UNIVERSIDAD COMPLUTENSE DE MADRID

Facultad de Informática

Departamento de Arquitectura de Computadores y Automática

Desarrollo de una plataforma de inyección de errores sobre hardware reconfigurable

Proyecto de Sistemas Informáticos

Felipe Serrano Santos
Rubén Tarancón Garijo

Profesor Director:
Hortensia Mecha López

Madrid, 2011

UNIVERSIDAD COMPLUTENSE DE MADRID

Facultad de Informática

Departamento de Arquitectura de Computadores y Automática

Desarrollo de una plataforma de inyección de errores sobre hardware reconfigurable

Proyecto de Sistemas Informáticos

Felipe Serrano Santos
Rubén Tarancón Garijo

Profesor Director:
Hortensia Mecha López

Madrid, 2011

Autorización

Los autores de este proyecto autorizamos a la Universidad Complutense a difundir y utilizar con fines académicos no comerciales tanto la memoria, como el código, la documentación y el prototipo de la plataforma desarrollada.

Felipe Serrano Santos

Rubén Tarancón Garijo

Resumen

Las mejoras tecnológicas alcanzadas por el hardware reconfigurable, su bajo coste de desarrollo y facilidad de implementación de circuitos, junto con la demanda de flexibilidad de las aplicaciones, hacen que los dispositivos reconfigurables sean una opción a tener en cuenta frente a la de los circuitos tradicionales para usarlos en aplicaciones aeroespaciales.

El problema que hay en estos entornos son las partículas de alta energía que pueden ocasionalmente colisionar con los circuitos y modificar la memoria de configuración del hardware reconfigurable, haciendo que tengan un comportamiento erróneo, algo inadmisible en este tipo de sistemas, donde los circuitos deben ofrecer una fiabilidad y durabilidad muy alta. En particular, debido al carácter crítico del correcto funcionamiento de un proyecto de tal envergadura y presupuesto, se hacen necesarios mecanismos para evitar estas alteraciones provocadas por las partículas cósmicas.

Para probar la vulnerabilidad de un circuito frente a alteraciones en el mapa de bits, en este proyecto se presenta una plataforma que simula el comportamiento de una partícula y permite probar distintos circuitos y comprobar su robustez.

Abstract

The technological improvements reached by reconfigurable hardware, its low development cost, the facility to implement circuits with them and the demand of flexibility in applications, it makes the reconfigurable devices a useful option to have in mind instead the traditional circuits in order to use it in aerospace applications.

The problem in these environments is the high energy particles, that occasionally may collide with the circuits and modify the reconfiguration memory of the reconfigurable hardware, making them work wrong. This is inadmissible in this kind of systems, where the circuits have to prove a large fiability and durability.

Particularly, due to the critical behavior of a project so big and costly, mechanisms are needed to avoid this modifications made by cosmic particles.

In order to prove the vulnerability of a circuit against changes in its bitmap, in this project we introduce a platform that simulate the behavioral of a particle and allow us to prove different circuits and his robustness.

Lista de palabras

Hardware reconfigurable
FPGA
Reconfiguración parcial
Bitflip
Icap
Aplicaciones aeroespaciales
Reconfiguración dinámica

Índice

Lista de figuras

Capítulo 1: Objetivo

El objetivo de este proyecto es proveer una plataforma que simule los impactos de las partículas cósmicas en un diseño implementado sobre una FPGA (field programmable gate array) tipo Virtex, modificando los bits de configuración y viendo como éstos afectan al funcionamiento del circuito para así poder ver qué partes son críticas.

Esto se conseguirá mediante la ayuda de un software que servirá de comunicación entre la FPGA y el ordenador y un hardware adicional que de soporte a una serie de operaciones tales como:

- Cargar un testbench (una serie de entradas que tomará el circuito)
- Generar un golden (las salidas correspondientes a un testbench sabiendo que el funcionamiento del circuito es correcto)
- Cargar un golden
- Ejecutar el testbench (estimular al circuito con las entradas del testbench y comprobar que sus salidas coincidan con el golden asociado para corroborar el correcto funcionamiento)
- Inyección de errores (barrido del mapa de bits de configuración del circuito invirtiéndolos 1 a 1 y comprobar cómo éstos cambios afectan al funcionamiento del circuito)

Un esquema general de como se quiere hacer esto viene dado en la Figura 1.

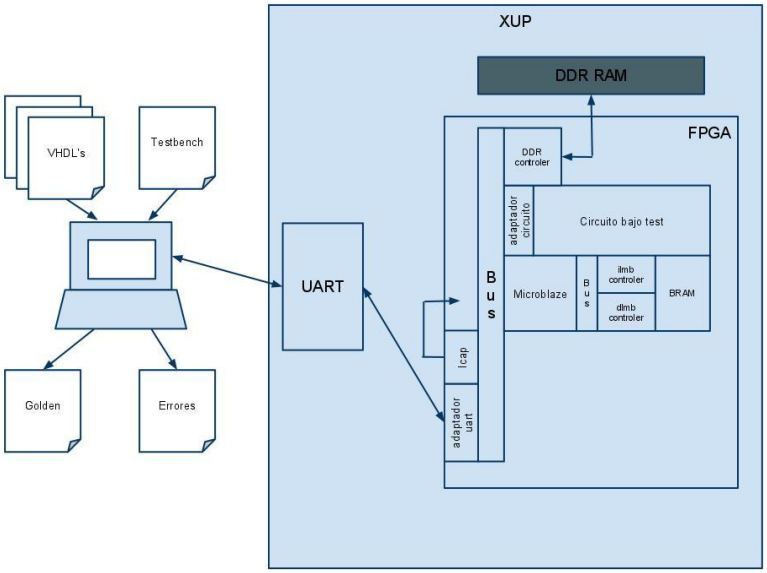

Figura 1: Esquema general del proyecto

Capítulo 2: Entorno de desarrollo

2.1. Entorno hardware

El sistema desarrollado en este proyecto está diseñado para funcionar sobre un dispositivo dinámicamente reconfigurable, en concreto sobre una FPGA de la familia Virtex II Pro empotrada en la placa de desarrollo XUP Virtex II Pro Development System o en una Virtex V empotrada en la placa de desarrollo XUPV5-LX110T.

El lenguaje de descripción hardware utilizado ha sido VHDL.

2.1.1. FPGA

Las FPGAs son circuitos electrónicos prefabricados reconfigurables que contienen bloques de lógica cuya funcionalidad e interconexión se puede programar. Esto nos permite crear nuevos circuitos que se comportarán como nosotros queramos. Son típicamente volátiles.

Las FPGAs aparecieron como una evolución de los CPLDs (Complex Programmable Logic Device). Las ventajas que tienen frente a éstos, son su mayor densidad de puertas, la flexibilidad de su arquitectura y que habitualmente contienen funciones de alto nivel, como sumadores o multiplicadores, y bloques de memoria empotrados en la matriz de interconexionado.

La ventaja de la FPGA es que al diseñar un sistema no es necesaria su fabricación, sino que se puede realizar programando correctamente el dispositivo, por lo que los costes se reducen.

Una FPGA es programable a nivel hardware aunque actualmente existen dispositivos programables que llevan además procesadores empotrados capaces de ejecutar programas especificados con lenguajes de alto nivel. Debido a esto proporciona las ventajas de un procesador de propósito general y un circuito especializado (sistema empotrado).

La utilización de FPGAs se está extendiendo en todos los ámbitos del diseño de sistemas digitales debido a que los costes y el tiempo de desarrollo son mucho menores. En muchos casos ni siquiera se procede a la posterior fabricación del circuito final sino que se utiliza directamente la FPGA.

2.1.1.1. Aplicaciones

Debido a la flexibilidad, capacidad de procesado y a la rápida implementación de sistemas sobre FPGA, estos dispositivos se emplean habitualmente en:

- Procesamiento de señales digitales.
- Módulos de comunicación.
- Sistemas aeronáuticos.
- Formación en el diseño de sistemas hardware.
- Simulación y depuración en el diseño de microprocesadores y microcontroladores.

2.1.1.2. Arquitectura Virtex II pro

La arquitectura básica de una FPGA Virtex II pro consiste en una matriz de CLBs (Configurable Logic Block), IOBs (células de entrada y salida), canales de comunicación, DCMs (Digital Clock Manager), BRAM (Block SelectRAM), multiplicadores y dos procesadores. En la Figura 2 se puede apreciar un esquema de cómo están organizados los componentes anteriores. Se trata de un esquema de una Virtex II, por lo que faltarían los dos procesadores.

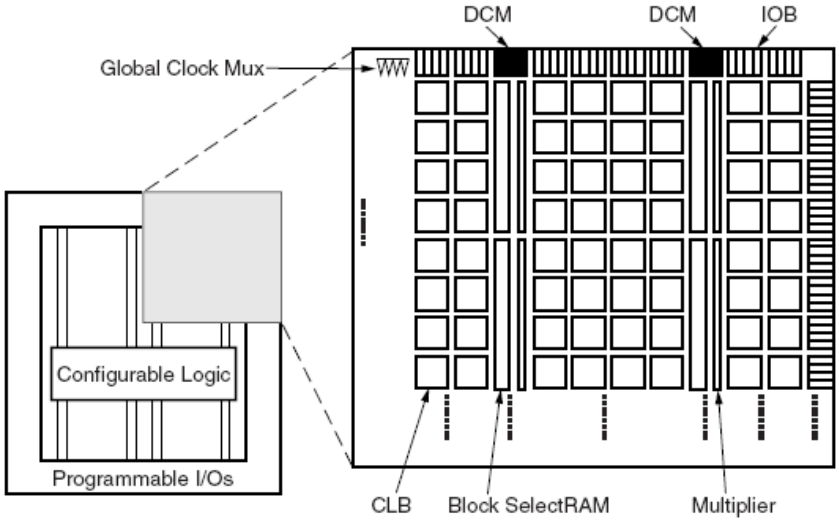

Figura 2: Estructura interna de una FPGA Virtex II

- IOB:

Comunican el dispositivo con el exterior y están situados rodeando la matriz de bloques lógicos, próximos a las patillas del chip. Se puede programar su forma de trabajo para que se comporten como puertos de entrada, salida o entrada-salida (Ver Figura 3).

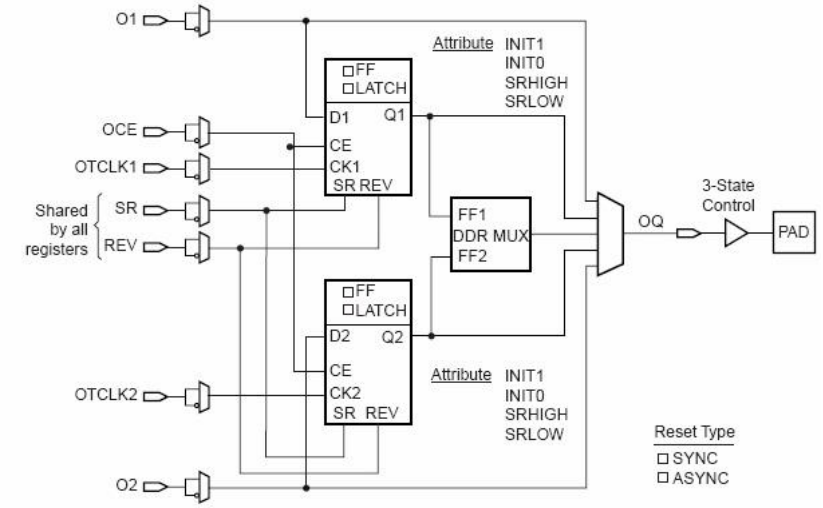

Figura 3: Estructura de un IOB en Virtex II pro

- CLB:

La estructura de un CLB en una Virtex II pro es la siguiente (Ver Figura 4):

Slices	LUTs	Flip-Flops	MULT_ANDs	Arithmetic & Carry-Chains	SOP Chains	Distributed SelectRAM	Shift Registers	TBUF
4	8	8	8	2	2	128 bits	128 bits	2

Figura 4: Composición de un CLB en Virtex II pro

Cada CLB contiene 4 slices interconectados entre sí por conexiones rápidas. Existe lógica e interconexionado especial para aceleración de carry (2 carrys por CLB). (Ver Figura 5)

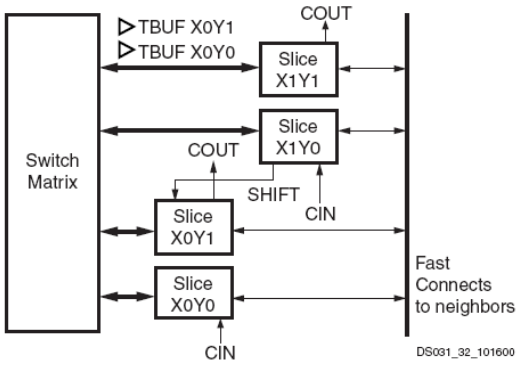

Figura 5: Estructura de un CLB en Virtex II pro

Cada slice tiene dos generadores de funciones, también llamadas LUT (look up table) F y G, dos elementos de almacenamiento, puertas lógicas, multiplexores, encadenamiento de carry y encadenamiento horizontal (puerta OR) (Ver Figura 6).

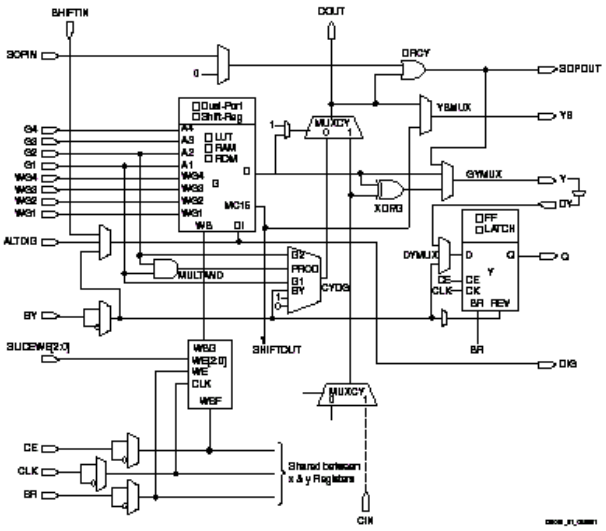

Figura 6: Parte superior de un SLICE en Virtex II pro

Una LUT contiene una tabla de funciones lógicas de cuatro entradas, tal y como se muestra en la Figura 7.

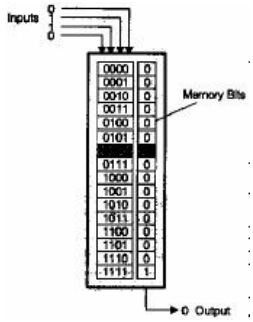

Figura 7: Estructura de una LUT en Virtex II pro

- BLOQUES DE INTERCONEXIÓN

Están formados por un canal de rutado vertical y otro horizontal a los que se puede conectar cualquier CLB próximo mediante un punto de interconexión. En los cruces de los dos canales se encuentran las matrices de conmutación (Ver Figura 8).

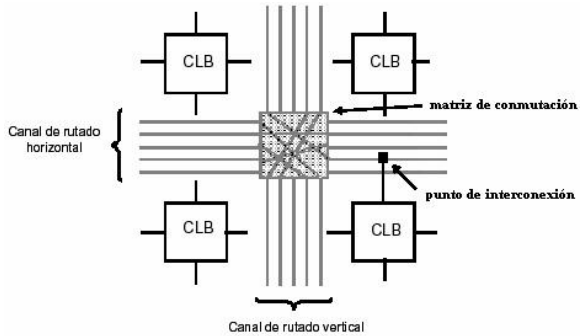

Figura 8: Estructura de interconexión en Virtex II pro

- BRAM:

Están integrados dentro de la misma FPGA y la existencia de estos evita que se deban usar CLBs para sintetizar módulos de memoria menos eficientes cuando el diseño lo requiere. Son dispositivos de almacenamiento muy rápidos y de poca capacidad. En Virtex II se pueden almacenar hasta 18 KB.

- MULTIPLICADORES:

Mismo propósito que la BRAM. Al estar implementados físicamente y no mediante CLBs, son más rápidos y ocupan menos área. El multiplicador y la memoria SelectRAM pueden usarse simultáneamente, aunque comparten parte del interconexionado.

- PROCESADORES:

Cuenta con dos procesadores RISC PowerPC 405.

- DCM:

Este módulo se encarga de la gestión del reloj y realiza tareas como multiplicar o dividir su frecuencia, eliminar el skew dentro del dispositivo o en los componentes externos, asegurar una señal de reloj limpia, desfasar el reloj una cantidad fija o incremental o devolver una señal espejo del reloj. Las funciones realizadas y las señales generadas se pueden ver en la Figura 9.

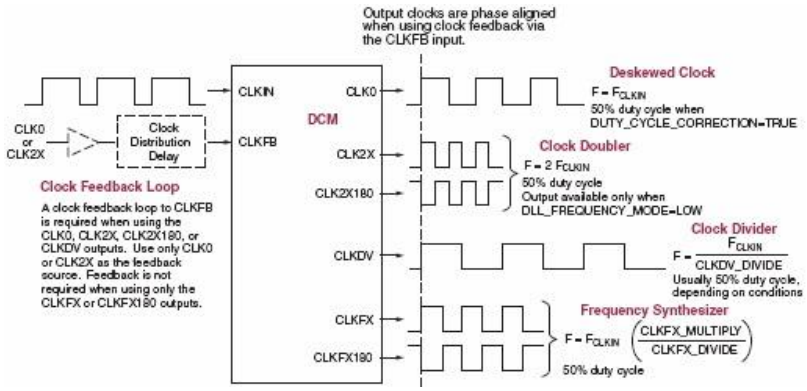

Figura 9: Relojes generados por el DCM

- clk0: es el reloj de entrada sin skew.
- clk2x: es el reloj con el doble de frecuencia y sin skew.
- clk2x180: reloj clk2x desfasado 180 grados.
- clkdv: reloj de entrada dividido por un parámetro configurable.
- clkfx: reloj de entrada dividido y multiplicado por unos parámetros configurables.
- clkfx180: clkfx desfasado 180 grados.

Es importante destacar que en la Virtex II pro la unidad mínima de reconfiguración parcial es una frame. Una frame tiene un bit de ancho y de alto, el alto de la FPGA (Figura 10).

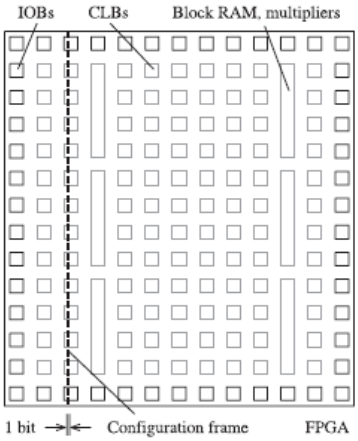

Figura 10: Frame en Virtex II pro

2.1.1.3. Arquitectura Virtex V

La arquitectura básica de una FPGA Virtex V consiste en una matriz de CLBs, IOBs**,** canales de comunicación, CTMs (Clock Manager Tile), BRAM y DSP (Digital Signal Processing).

- CLB:

La composición de un CLB en una Virtex II pro es la siguiente:

Slices	LUTs	Flip-Flops	Arithmetic and Carry Chains	Distributed RAM(1)	Shift Registers(1)
2	8	8	2	256 bits	128 bits

Figura 11: Composición de un CLB en Virtex V

A diferencia de la Virtex V, esta FPGA contiene únicamente dos slices en cada CLB. En la siguiente imagen aparecen cómo están estructurados ambos slices dentro del CLB.

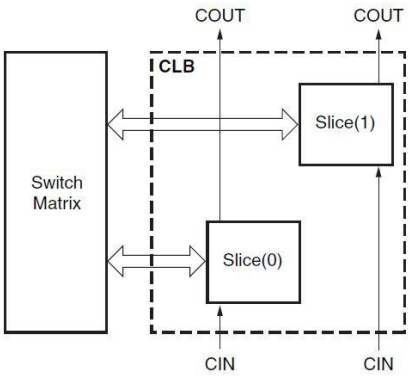

Figura 12: Estructura de un CLB en Virtex V

En Virtex V existen dos tipos de slice: slicel y slicem. Los primeros se usan cuando lo que se va a implementar en ellos es lógica combinacional, y los segundos cuando se necesita almacenar memoria. En la siguiente figura se aprecia la estructura de un slicel.

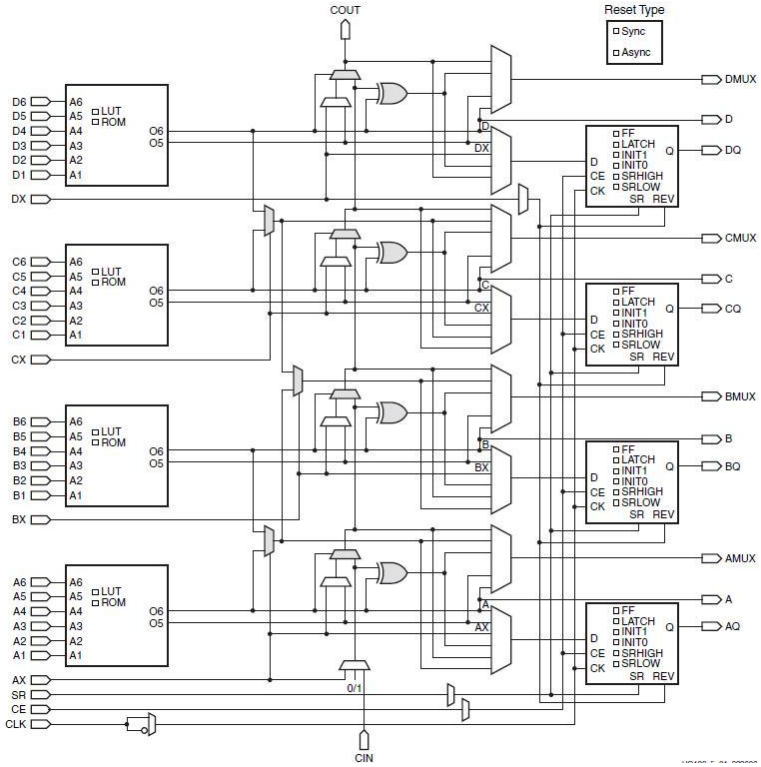

Figura 13: SliceL en Virtex V

Las tablas tradicionales LUT de cuatro entradas se han vuelto extremadamente limitantes y requieren muchos niveles de lógica para implementar código complejo. Xilinx ha extendido la tabla LUT a una de seis entradas para obtener más capacidad. Estas LUTs pueden configurarse como funciones de 6 entradas o como 2 funciones de 5 entradas (Ver Figura 14).

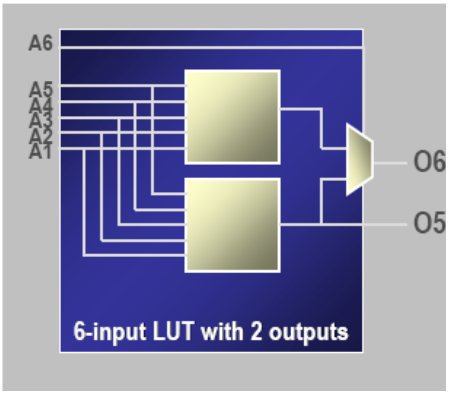

Figura 14: LUT en Virtex V

- BRAM:

La única diferencia con la BRAM de la Virtex II pro es el tamaño de la misma. En Virtex V es de 36 KB.

- DSP:

En Virtex V desaparecen los multiplicadores y aparecen dentro de los DSPs. Un DSP se utiliza para operaciones aritmético-lógicas. Está compuesto por un multiplicador de 25bits*15bits y un sumador de 48bits (Ver Figura 15):

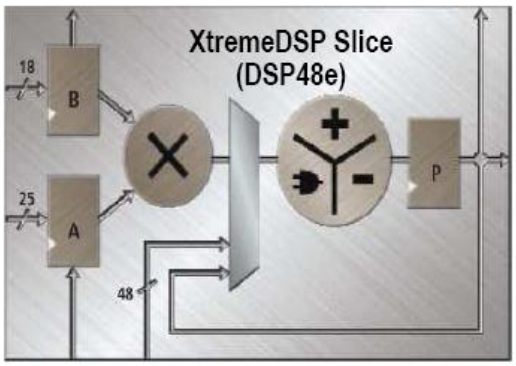

Figura 15: DSP en Virtex V

- CMT:

En Virtex V han ampliado los DCMs y los agrupan en este nuevo componente. Los CMT están compuestos de dos DCMs y de un PLL (Ver Figura 16). Los PLL son componentes analógicos con realimentación cuya misión es que la señal de reloj que generan sea lo más perfecta posible.

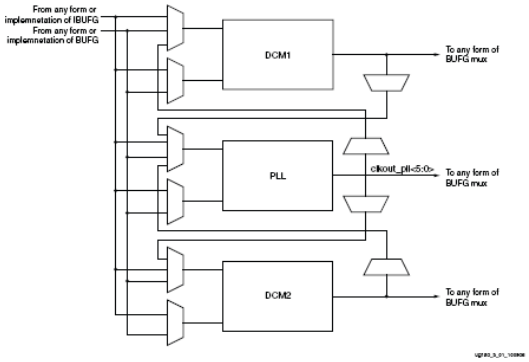

Figura 16: CMT en Virtex V

En Virtex V, a diferencia de la Virtex II pro, la unidad mínima de reconfiguración no la altura del dispositivo. Cada frame tiene un bit de ancho y de alto la altura de 20 CLBs. Una estructura así favorece la reconfiguración parcial, ya que es más fácil agrupar los componentes en frames completas.

2.1.3.4. Puerto ICAP (Internal Configuration Access Port)

La reconfiguración parcial es una característica que permite cargar un diseño en una parte de la FPGA mientras el resto del dispositivo esté funcionando normalmente.

Este tipo de reconfiguración es útil para aplicaciones que requieren cargar diferentes diseños en el mismo área del dispositivo o la flexibilidad para cambiar parte de un diseño, en tiempo de ejecución, sin necesidad de resetear o reconfigurar el dispositivo al completo. De ahí que este mecanismo sea idóneo para la corrección de errores en nuestro circuito en tiempo de ejecución.

La reconfiguración parcial en la familia de FPGAs Virtex II Pro y Virtex V se puede realizar mediante una interfaz interna de reconfiguración conocida como ICAP**.** También se pueden usar los modos SelectMap o el modo JTAG que necesitan un controlador externo para enviar los datos de reconfiguración. La Figura 17 muestra la interfaz de este puerto.

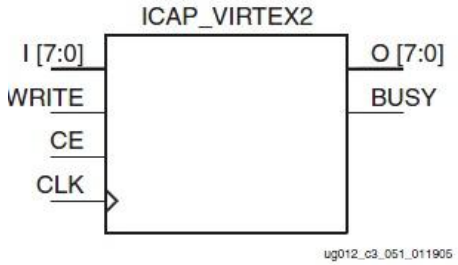

Figura 17: Interface Icap

2.1.2. Placas de desarrollo XUP

Los dispositivos utilizados han sido una FPGA XC2VP30, de la familia Virtex II pro, montada sobre una placa de desarrollo XUP Virtex II pro Development System (Ver Figura 18) y una FPGA XC5VLX110T, de la familia Virtex V, en la placa XUPV5-LX110T Evaluation platform (Ver Figura 19).

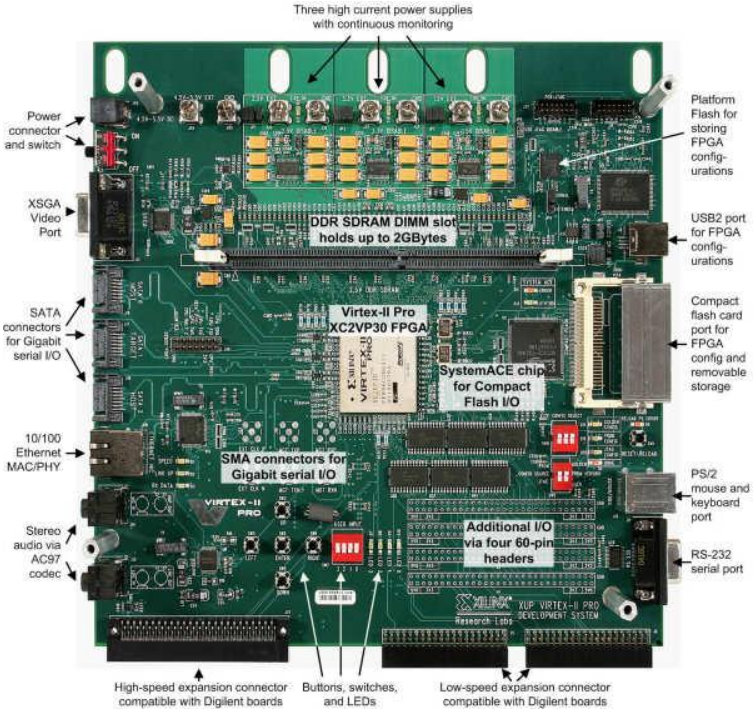

Figura 18: XUPV2

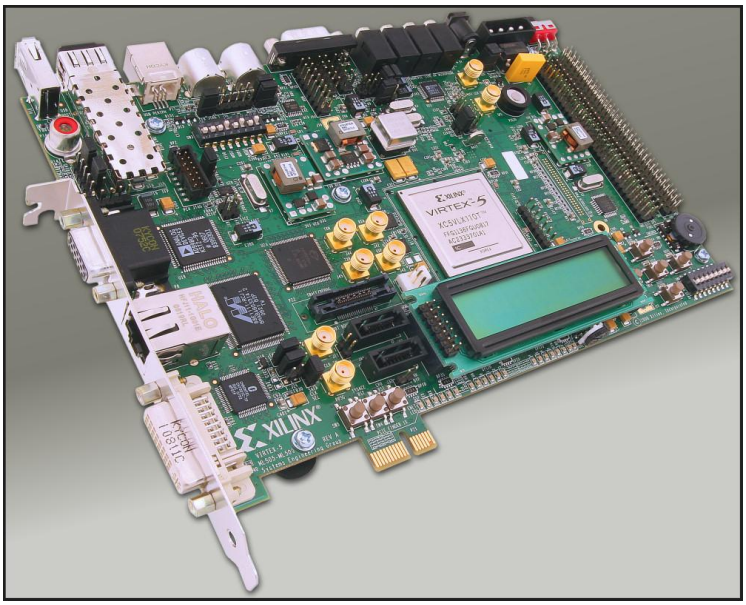

Figura 19: XUPV5

Las placas XUP aportan algunos componentes periféricos, como una memoria DDR RAM (puede ser desde 64MB a 2 GB), puerto Ethernet 10/100, códecs de audio y video, pines de entrada y salida para una tarjeta compact flash, puertos PS/2, switches, leds, puertos USBs, puertos XSGA video y conectores SATA, entre otras cosas y dependiendo del modelo.

La forma en que la FPGA Virtex II pro queda conectada en la placa de desarrollo se muestra en la Figura 20. Para la Virtex V es muy similar.

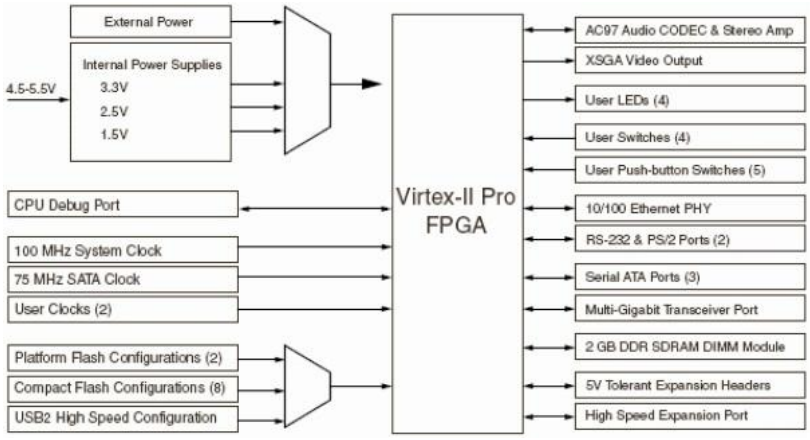

Figura 20: Conexiones Virtex II Pro con la XUP

2.2. Entorno software

En cuanto al software, se ha utilizado Xilinx ISE 9.1, Xilinx EDK 9.1, Xilinx SDK 9.1 y iMPACT 9.1 para el desarrollo relacionado con la Virtex II pro y los mismos programas en las versiones 12.1, además del PlanAhead 12.1 para el desarrollo de la Virtex V.

2.2.1. Xilinx ISE

El entorno de desarrollo ISE (Integrated Software Environment) es una herramienta de diseño de sistemas digitales. Las versiones utilizadas han sido la 9.1 y la 12.1.

Con esta herramienta se puede crear, verificar, simular y sintetizar diseños basados en una FPGA o CPDL. Los pasos a seguir en el diseño e implementación se muestran en la siguiente figura:

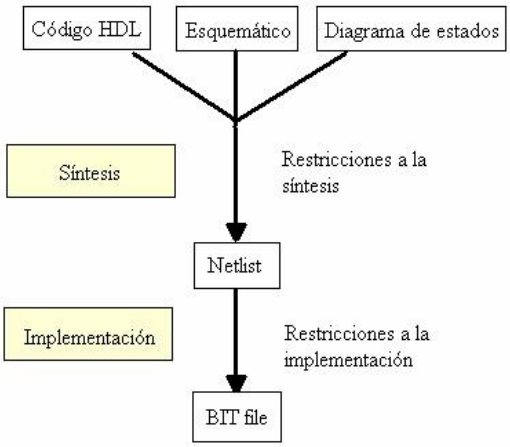

Figura 21: Flujo de diseño de circuitos

El editor en el que se realiza el diseño es el Project Navigator. Los diseños se pueden generar en un lenguaje VHD (VHDL o Verilog), o mediante esquemáticos o dibujando diagramas de estado que transforma en código.

Este es el aspecto general del programa (Ver Figura 22):

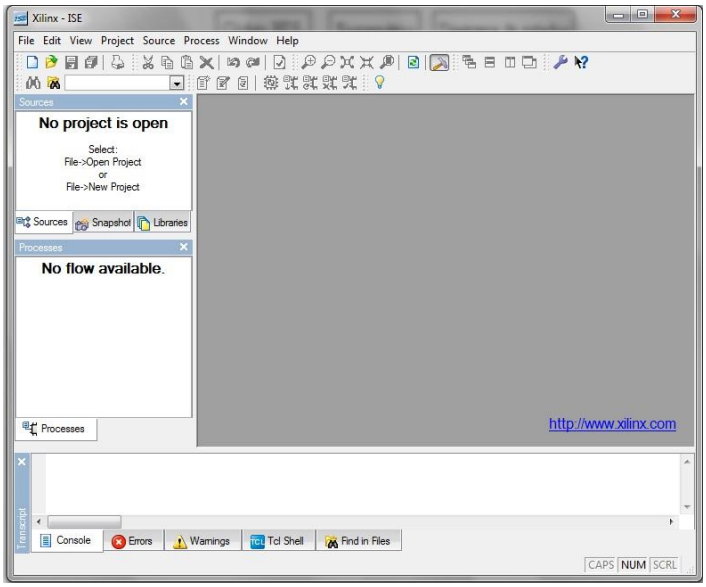

Figura 22: Xilinx ISE 9.1

Una vez abierto o creado un proyecto podremos ver el cuadro de procesos. Este cuadro nos muestra las operaciones que se pueden realizar sobre el fichero seleccionado en el cuadro de fuentes. Las opciones de síntesis e implementación solamente estarán disponibles si el fichero seleccionado es el más alto en la jerarquía. En la Figura 23 se pueden ver los procesos disponibles en ISE.

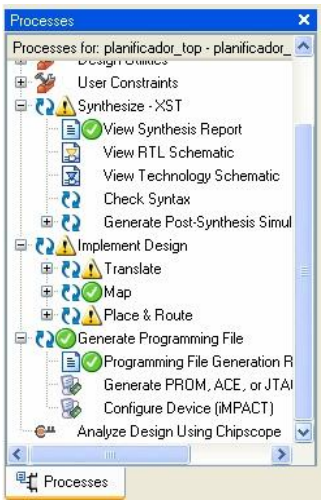

Figura 23: Cuadro de procesos

La función de síntesis transforma el código inicial a una netlist descrita en un lenguaje interno para un hardware concreto. Se definen restricciones temporales, asignación de pines y optimizaciones de cada uno de los componentes a partir de la descripción de comportamiento del sistema. Para realizar este proceso se deberá seleccionar la opción Synthesize-XST. Una vez finalizado ISE nos muestra un informe con toda la información de la síntesis.

El paso de implementación está dividido en tres fases: translate, map y place&route. Lo que se hace en este proceso es enlazar los módulos con las condiciones indicadas en el fichero de restricciones (.ucf) y reducirla a primitivas de Xilinx. A continuación se adapta al dispositivo en el cual se va a volcar el diseño y se colocan y se rutan los componentes físicos finales. Para realizar este proceso se seleccionará Implement Design. Cada una de las fases generará un informe.

En el fichero de restricciones se especificarán los pines a los que se deberá conectar cada señal de entrada y salida y su tecnología. Se pueden añadir restricciones temporales, como retardo máximo de una señal o frecuencia del reloj, y colocación de bloques en la FPGA.

El último paso será la generación del bitstream (fichero .bit) que será lo que se cargue en la FPGA. Este fichero contiene toda la información de localización de los dispositivos y los elementos de rutado. Para este proceso se seleccionará la opción *"Generate Programming File".*

2.2.2. Xilinx EDK

Xilinx platform studio EDK permite diseñar sistemas complejos en FPGA con solo unos clicks. Este software integra a Xilinx ISE y usa los procesos de síntesis e implementación de éste con nuestros módulos de una manera muy cómoda para el usuario. Además, permite realizar aplicaciones software que luego irán mapeadas en memoria y serán ejecutas por los distintos procesadores que implementemos. El flujo de diseño para crear un sistema empotrado está ilustrado en la Figura 24:

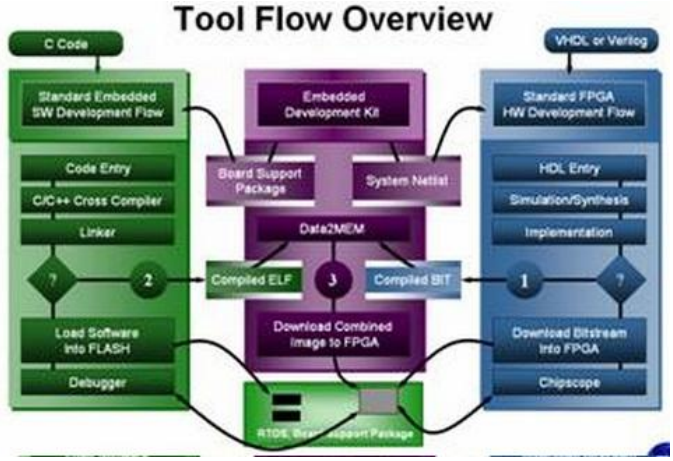

Figura 24: Flujo de diseño EDK

EDK nos provee una forma de desarrollar hardware de más alto nivel encapsulando cada módulo que desarrollemos en "cores" y nos da una visión del sistema más global (Ver Figura 25) mostrando como los cores están relacionados entre sí sin entrar en demasiados detalles, lo que permite desarrollar sistemas complejos que de otra manera sería muy engorroso hacerlos.

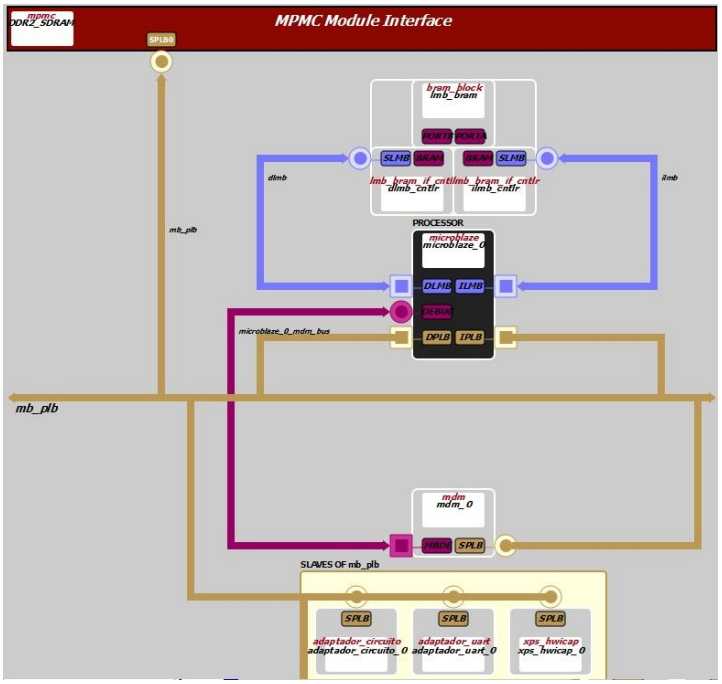

Figura 25: Visión general del sistema en EDK

Además permite usar cores ya desarrollados (por Xilinx, otros usuarios o por nosotros mismos) e integrarlos en nuestro diseño mediante drag & drop. En la Figura 26 se muestra el aspecto general del programa con un proyecto abierto.

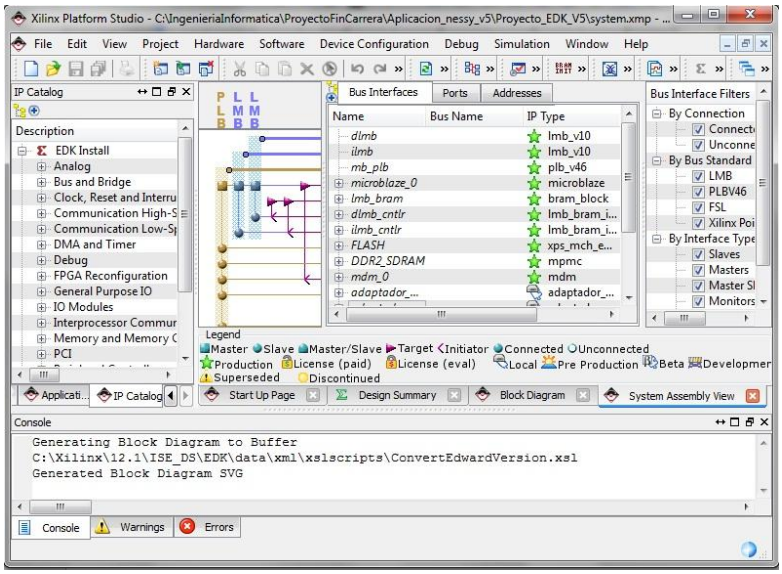

Figura 26: Xilinx EDK 12.1

En la imagen anterior podemos distinguir varias ventanas:

- La ventana izquierda se compone de 3 pestañas:
  a. La mostrada en la captura nos permite buscar en las librerías de cores para añadir uno nuevo con tan solo arrastrarlo a la pestaña central. La otra pestaña oculta muestra opciones del proyecto tales como el .ucf, el archivo .ut con los parámetros del bitgen etc.
  b. *"Aplications"* sirve para gestionar las aplicaciones software que luego irán mapeadas en la memoria de algún procesador (en caso de haberlo).
  c. *"Project"* (no se ve en la captura) sirve para la gestión de las opciones nuestro proyecto tales como los parámetros con los que se llamará a bitgen, el .ucf, etc.
- En la ventana central tenemos tres pestañas:
  a. Por una parte, *"bus interfaces"* muestra la estructura de buses y qué módulos están conectados a cada uno.
  b. *"Ports"* nos muestra los puertos de cada módulo y como están conectados entre ellos..
  c. La última pestaña, *"addresses"*, nos permite asignar direcciones de memoria a cada módulo dependiendo en qué bus estén conectados.
- En la parte de abajo tenemos la consola donde ver los logs, warnings y errores al realizar algún cambio a nuestro proyecto.

De los botones en la barra superior, los más importantes son *"hardware", "software"* y *"device configuration*" (Ver Figura 27). Son los que nos permiten sintetizar e implementar un diseño, el primero para la parte hardware (usando los procesos del ISE), el segundo compila los fuentes en C de las aplicaciones que queramos mapear en algún procesador y el tercer botón hace todos los procesos en conjunto: sintetiza e implementa los módulos hardware, compila los fuentes en C y mapea el .elf en el rango de memoria asociada a los procesadores donde se deban ejecutar.

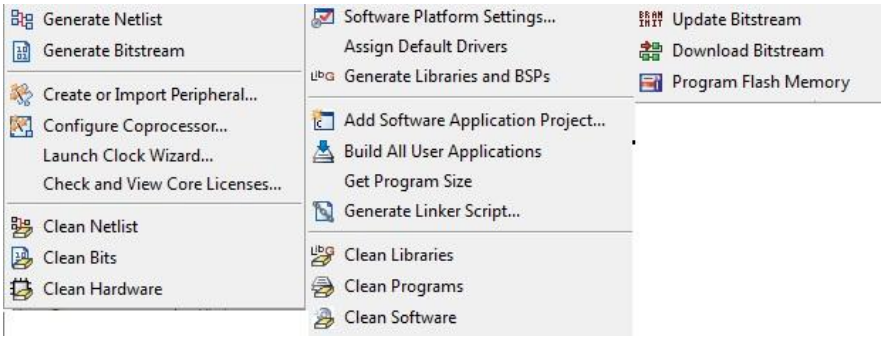

Figura 27: Botones de procesos en EDK

A Xilinx EDK se le puede pasar como parámetro un script mediante la opción `-scr <script>`. Además, también se puede ejecutar en modo consola con la opción `-nw`. Esta es la forma en que usaremos el programa en nuestro proyecto.

2.2.3. Xilinx SDK

Xilinx SDK (Ver Figura 28) es una versión modificada de eclipse que está integrado con Xilinx EDK y así se tiene un IDE donde desarrollar las aplicaciones software de una manera cómoda y con las ventajas que un IDE de este tipo supone.

Los proyectos creados en SDK dependen del EDK puesto que tienen que importar unas librerias que variarán según qué placa se esté usando en el proyecto EDK. Es por eso que el SDK mantiene toda la información relevante relacionada con el hardware en un proyecto aparte llamado *"hw_platform_0"*.

También tiene una plataforma software llamada en este caso, *"standalone_bsp",* necesaria para poder acceder a las características de bajo nivel de los dispositivos implementados en la placa mediante funciones en C genéricas tales como la entrada/salida, interrupciones, caché etc.

Se podría ver como un SO aunque en este caso no lo es, ya que solo nos provee con funciones de alto nivel adaptadas a nuestro hardware para poder usar los dispositivos, pero en ningún momento hay cargado en la memoria asociada a los procesadores nada más que nuestras aplicaciones. Para tener un SO como tal, Xilinx da la opción de usar un kernel reducido de linux en el que se permite el uso de threads, sistema de ficheros, señales, etc.

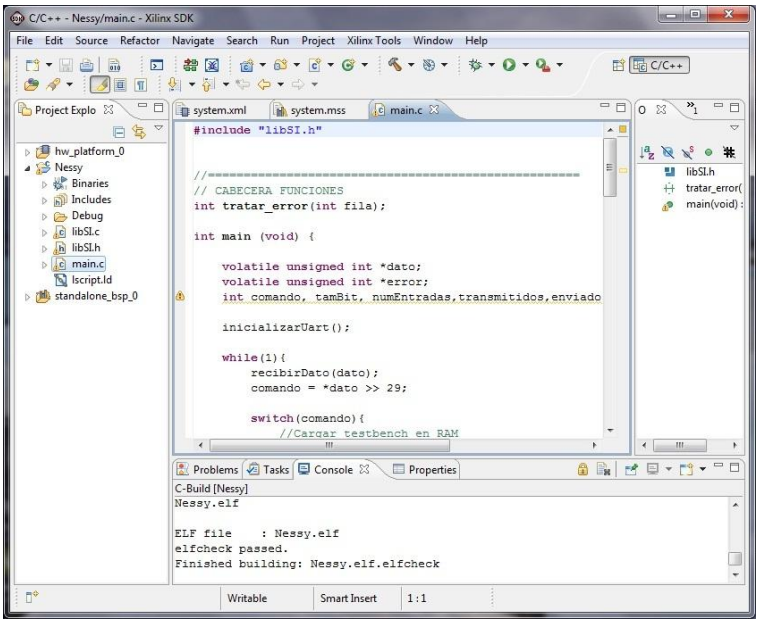

Figura 28: Xilinx SDK 12.1

2.2.4. iMPACT

iMPACT es el software desarrollado por Xilinx para programar sus FPGAs. Con este programa también se pueden generar las imágenes y programar memorias flash. Permite comunicarse con la FPGA de varias formas, mediante cable paralelo IV, cable paralelo III o USB. La comunicación con los dispositivos se puede hacer automáticamente mediante boundary scan. Una vez que se ha establecido la conexión entre la FPGA e iMPACT, (Ver Figura 29) solo hay que determinar qué bitstream o imagen flash quieres utilizar y en qué dispositivo y dar a “program”.

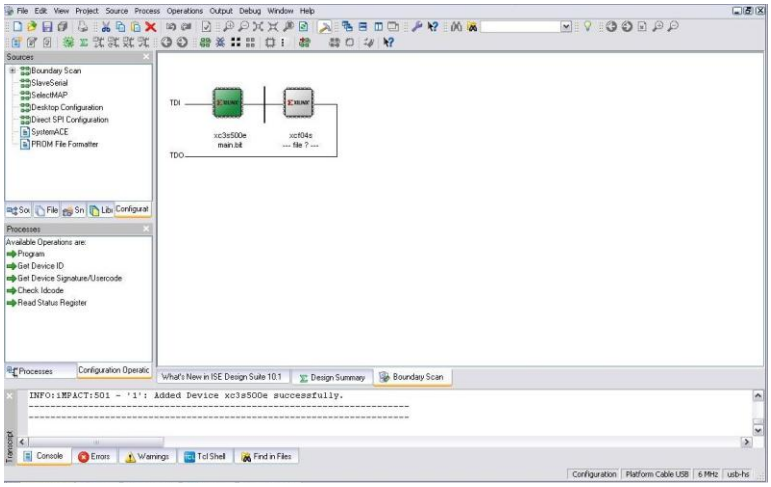

Figura 29: iMPACT

iMPACT puede recibir como parámetro un script mediante la opción `-batch <script>` que además hará que se ejecute en modo consola. Esta es la forma en que usaremos el programa en nuestro proyecto.

2.2.5. PlanAhead

Este programa lo hemos utilizado en nuestra versión realizada para la FPGA Virtex V. La utilidad y la necesidad de utilizar esta herramienta reside en que es fácil crear un bitstream parcial de un componente de nuestro sistema.

Es muy importante poder crear un bitstream parcial porque con la versión para Virtex II pro, agrupábamos nuestro circuito en zonas de la FPGA mediante el comando *"AREA_GROUP"* en el fichero de restricciones, pero eso no implica que la herramienta de Xilinx, al mapear, introduzca cableado de otros componentes que atraviese nuestra zona de reconfiguración.

En este proyecto utilizaremos la aplicación Partial Reconfiguration (Ver Figura 30) de la herramienta, ya que esta herramienta permite realizar múltiples utilidades. Como se puede apreciar en la siguiente imagen, las opciones que tenemos al abrir la herramienta son variadas. Nosotros tenemos que seleccionar las qué están marcadas. Hay que tener en cuenta que para utilizar esta utilidad del PlanAhead hace falta tener una licencia especial.

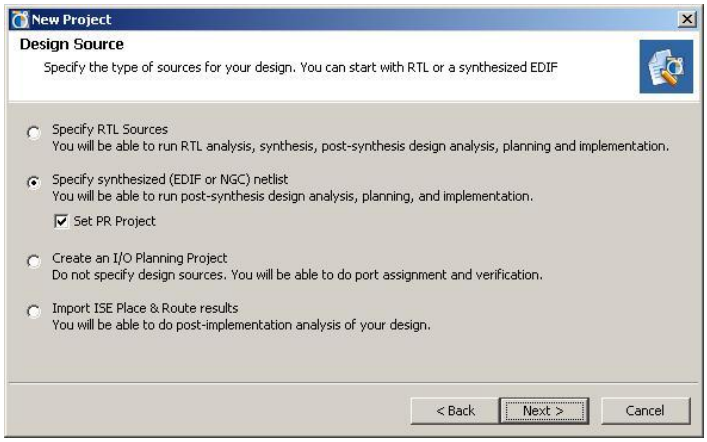

Figura 30: Nuevo proyecto PlanAhead

A continuación la herramienta nos pedirá dos archivos: el .ngc, que es el resultante después de haber realizado la síntesis y el .ucf, que es el fichero de restricciones del nuestro sistema. Después de introducir ambos ficheros el aspecto de la herramienta es el de la siguiente figura:

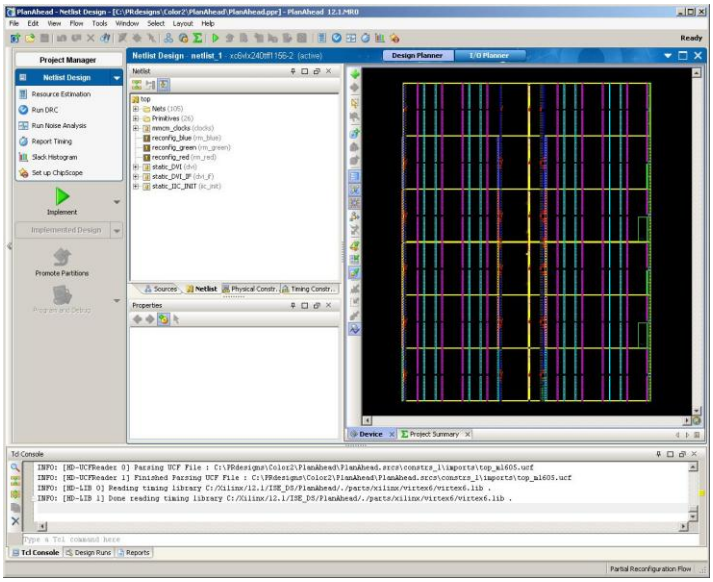

Figura 31: Visión general de PlanAhead

A continuación tenemos que definir el área en el que vamos a colocar nuestro circuito que queremos reconfigurar. Para ello hay que seleccionar nuestro componente, hacer click derecho y seleccionar *"Add Reconfigurable Module"*. Para definir el tamaño del componente, hacer click derecho en *"Set Pblock Size"* y dibujar en la pantalla de la derecha un rectángulo que indicará el lugar donde la herramienta colocará el circuito (Figura 32).

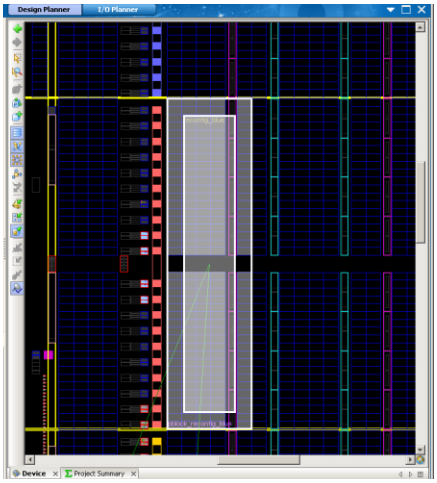

Figura 32: Crear región reconfigurable en PlanAhead

Como en nuestro caso queremos crear en una configuración, la creación de varios bitstreams (.bit general del sistema y .bit del componente a reconfigurar), tenemos que decírselo a la herramienta. Para ello hay que añadir a nuestra configuración los módulos que queramos. En el ejemplo de la Figura 33 se añaden 3 módulos (reconfig_blue, reconfig_green, reconfig_red) para los cuales se crearan unos bitstream parciales.

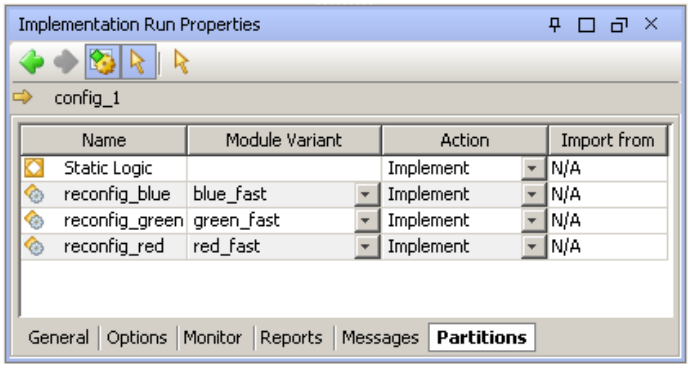

Figura 33: Varias implementaciones en PlanAhead

Por último ya solo hace falta accionar la opción "*Tools → Create Multiple Runs".* Estas acciones elegidas las introducimos en nuestra herramienta mediante la *opción* `-mode batch -source scrip.tcl`. Script.tcl lo generamos automáticamente con Nessy.

Capítulo 3: Plataforma de inyección de errores. Nessy 2.0

3.1. Nessy 2.0

Para el desarrollo de la plataforma no hemos partido de cero, hemos usado el proyecto desarrollado por Carlos Sánchez Vellisco Sánchez, Antonio José García Martínez y David Fernández Maíquez durante el curso 2009/2010 con nombre "Plataforma de inyección, detección y corrección de errores sobre FPGAs".

El aspecto de Nessy 2.0 es el siguiente:

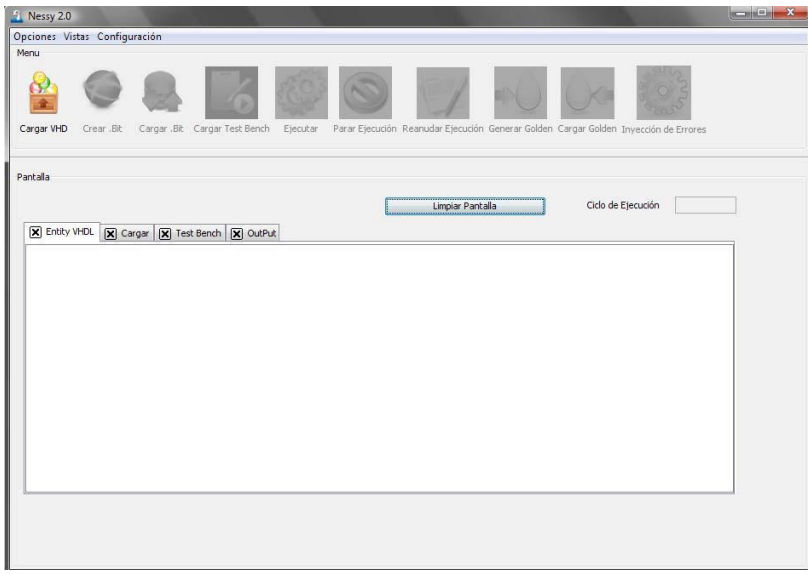

Figura 34: Nessy 2.0

En la imagen anterior se puede apreciar los botones que realizan las acciones necesarias para nuestro proyecto tales como “cargar testbench”, “generar golden” o *“inyección de errores”*.

El programa Nessy 2.0 sirve como plataforma para establecer una comunicación entre la FPGA y el ordenador y así poder realizar las acciones deseadas por nuestro proyecto tales como cargar un testbench, cargar o generar golden y ejecutar el testbench en el circuito.

Para tener una idea general de cómo se estructura Nessy 2.0 ver la Figura 35.

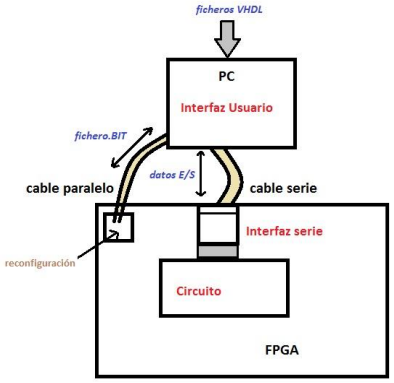

Figura 35: Visión global Nessy 2.0

3.2. Problemas

El proyecto tal y como estaba planteado tenía una serie de problemas y es que se abusaba de la comunicación por el puerto serie, lo que acababa siendo un cuello de botella, puesto que para el testeo de un diseño se necesitaban varios días de ejecución.

Nessy delegaba todo el trabajo a la parte software del ordenador, limitando a la FPGA a ejecutar el diseño a testear y una interfaz de la UART. En este caso, la FPGA es incapaz de almacenar ninguna información, por lo que los datos necesarios para la plataforma de inyección de errores (testbenches, golden, bitstreams modificados y restorers) se alojan en la memoria del ordenador teniéndose que enviar a través del puerto serie cada vez que se quiera hacer uso de ellos.

Para hacer una inyección de errores, para cada modificación de los bits de configuración (1 bit), se tiene que:

- Hacer la reconfiguración parcial enviando por el puerto jtag el bitstream modificado.
- Enviar el testbench entero a través del puerto serie.
- Ejecutarlo en el circuito.
- Recibir todo el fichero de resultados a través del puerto serie, sin tener en cuenta si había errores o no, ya que no hay forma de comprobarlo desde la FPGA.

- Hacer la reconfiguración parcial al circuito original enviando el bitstream restorer por el puerto jtag.

Hay que tener en cuenta que para un circuito pequeño como un contador de 2^8 colocado en un cuadro de 2x4 slices en la FPGA, se deben modificar más de 4000 bits. Son muchas reconfiguraciones para un circuito relativamente pequeño, que añadiendo el tiempo requerido por el puerto serie es de la magnitud de días. La conclusión es que es inviable hacer una inyección de errores de esta manera.

La solución a este problema consiste en añadir un hardware adicional al circuito a testear que sea capaz de realizar dentro de la placa casi todo el trabajo que antes hacía el computador, para así no depender tanto de la comunicación serie.

Este hardware adicional tiene que:

- Poder almacenar el testbench, golden y bitstream parciales en la memoria de la placa.
- Comunicarse con el ordenador para poder recibir testbenchs y enviar los resultados de la inyección de errores.
- Ejecutar el testbench localmente en el circuito.
- Generar y almacenar en la placa un golden partiendo de la ejecución del testbench en el circuito original.
- Obtener y guardar en memoria local las salidas del circuito.
- Crear los bitstream parciales modificados internamente desde la placa.
- Reconfigurar internamente solo la parte de la FPGA donde está instanciado el circuito a testear.
- Comparar, desde la placa, las salidas del circuito con los resultados del golden para detectar errores.

De esta manera necesitamos transmitir muchísimos menos datos por el puerto serie: **enviar el testbench y el bitstream parcial original del circuito una única vez y las órdenes necesarias** para que el hardware adicional sepa qué acción realizar en cada momento.

Capítulo 4: Modificaciones para la tarjeta Virtex II pro. Nessy 5.0

4.1. Modificaciones Hardware

Conociendo los problemas del proyecto anterior y cómo solucionarlos, se ha implementado un sistema (Ver Figura 36) que sea capaz de eliminar el cuello de botella causado por el puerto serie. Cómo se ha dicho anteriormente, se ha añadido un hardware que envuelve al circuito a testear.

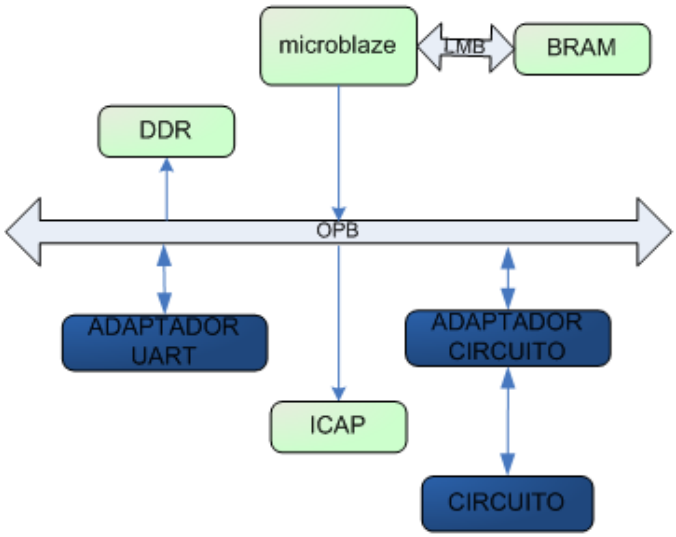

Figura 36: Modificaciones hardware en Virtex II pro

En el diagrama de bloques anterior se puede ver cómo están interconectados los módulos a nivel de buses.

La asignación de direcciones de los módulos en los buses es:

mb_opb					U
dlmb_cntlr	SLMB		0x00000000	0x0000ffff	64K
ilmb_cntlr	SLMB		0x00000000	0x0000ffff	64K
adaptador_circuito_0	SOPB		0x70000000	0x700001FF	512
adaptador_uart_0	SOPB		0x60000000	0x600001FF	512
opb_hwicap_0	SOPB		0x80000000	0x80007FFF	32K
DDR_512MB_64Mx64_rank2_row13_col10_cl2_5	SOPB:MCH0:MCH1:MCH2:MCH3	MEM0	0x40000000	0x4FFFFFFF	256M

Figura 37: Direcciones de los módulos en los buses

Para tener una idea más clara de sus puertos y conexiones entre módulos ver las Figura 38, 39, 40 y 41 que explicamos a continuación:

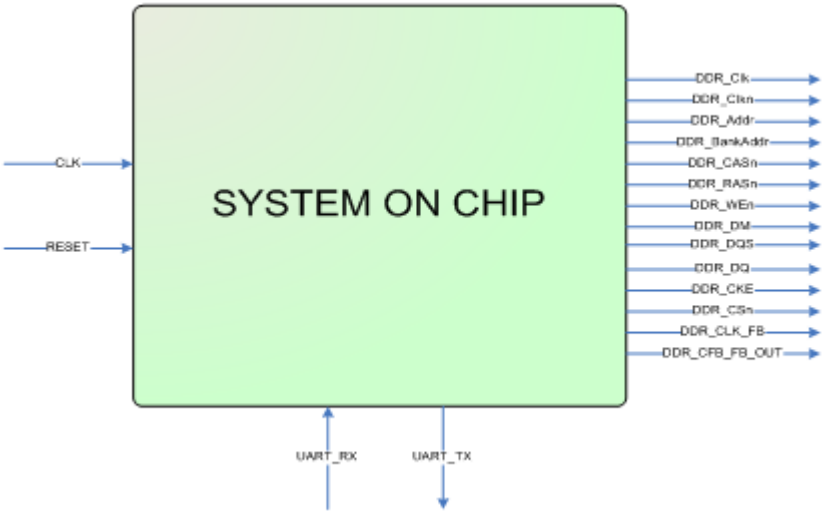

Figura 38: Puertos externos, Virtex II pro

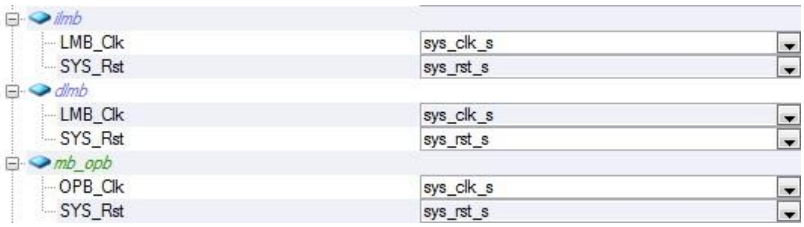

Figura 39: Puertos de los buses, Virtex II pro

- Por un lado, vemos que los buses están conectados obviamente al reloj y al reset del sistema (Ver Figura 39). EDK usa el reloj de un bus para propagarlo a todos los módulos que estén conectados a éste (microblaze, controladores de BRAM, controlador de DDR RAM, adaptadores e icap en nuestro caso).

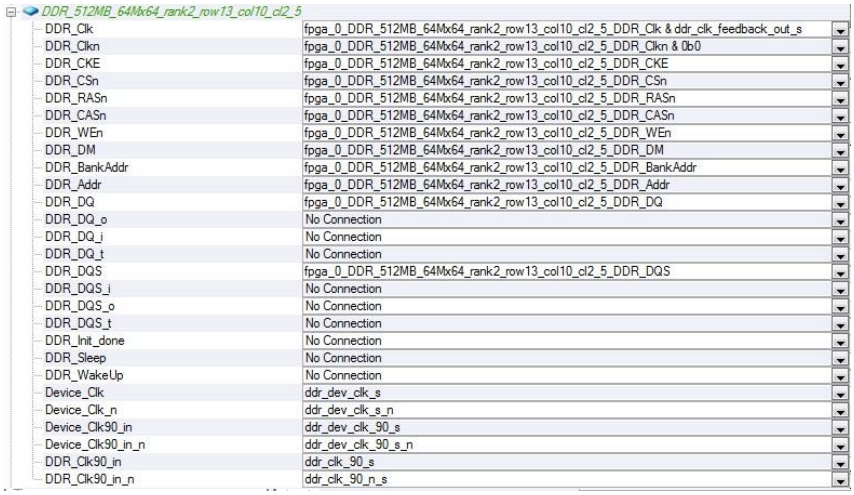

Figura 40: Puertos del controlador DDR RAM, virtex II pro

- En el caso del controlador de DDR la mayoría de los puertos son externos que van a la tarjeta DIMM de memoria de la placa XUP (Ver Figura 38).

Figura 41: Puertos de los adaptadores y del circuito, Virtex II pro

- Por otro lado, los únicos puertos relevantes son los de los adaptadores y el circuito.
    a. Vemos como en el caso de la uart este adaptador hace de interfaz entre la uart y el bus OPB. Éstos son puertos que van a los pines de la FPGA conectados a la uart de la placa XUP
    b. Adaptador circuito hace de interfaz entre el bus OPB y el circuito a testear.
- Es importante destacar que la frecuencia de reloj es de 50 Mhz. Ha estado restringida por la utilización de ICAP, ya que sino el sistema podría estar funcionando con una frecuencia de reloj de 100Mhz.
- También hemos intentado optimizar la velocidad de la uart, consiguiendo una velocidad de transferencia final de 19200 baudios. Para ello hemos restringido el área donde pueden localizarse cada uno de los componentes, de tal forma que los componentes que tengan dependencias estén en áreas cercanas, facilitando así a la herramienta su síntesis y que el tiempo de rutado sea menor. Los componentes

que se comunican con el exterior de la FPGA (DDR, adaptador_uart e ICAP) están colocados cerca de los pines de la FPGA.
En la siguiente imagen se puede apreciar la colocación de los componentes en el área de la FPGA. Estas restricciones irán en el fichero .ucf del sistema.

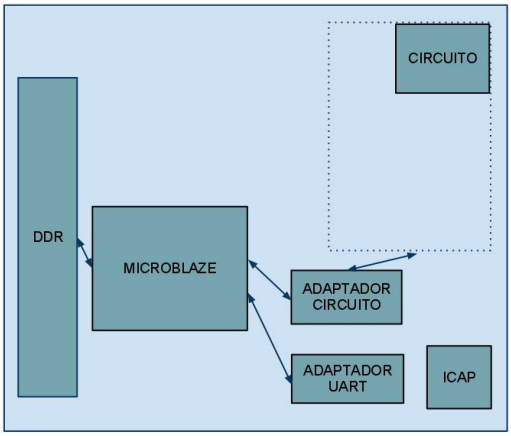

Figura 42: Distribución de dispositivos en la FPGA

4.1.1. Microblaze

El procesador empotrado microblaze softcore es un procesador con un conjunto de instrucciones reducido (RISC) optimizado para la implementación en FPGA's de Xilinx. La siguiente figura muestra el diagrama funcional del core:

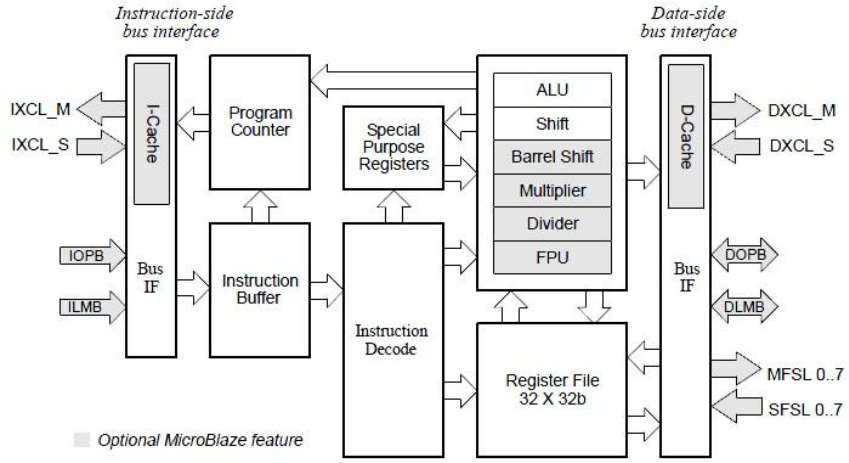

Figura 43: Diagrama funcional microblaze

Este procesador es altamente configurable, permitiéndote seleccionar un conjunto específico de características requeridas por su diseño.
Algunas características fijas del core son:

- Registros de propósito general de 32 bits.
- Instrucciones de 32 bits con 3 operandos y 2 modos de direccionamiento.

- Direcciones de bus de 32 bits.
- No segmentación

La versión de core usada en el proyecto para la Virtex II pro es la 6.00b y se usa para el proceso y gestión del hardware añadido a la plataforma. El microblaze está escuchando el puerto serie a la espera de algún comando enviado desde Nessy a través del computador y según sea el comando recibido se encarga de:

- **Cargar testbench**: Leer de la uart y almacenar un testbench recibido por el puerto serie.
- **Generar golden**: Leer de la DDR los datos del testbench, escribirlos en la entrada del circuito y leer y almacenar sus salidas en forma de golden en la RAM local.
- **Ejecutar**: Comparar las salidas del circuito con las del golden y guardar la información (en caso de haberla) de los errores.
- **Enviar golden**: Leer de la DDR los datos correspondientes al golden y enviarlos por el puerto serie a Nessy 5.0.
- **Cargar golden**: Recibir por la uart un golden y escribirlo en su región de DDR correspondiente.
- **Reconfiguración**:
    - Primero recibir el bitstream parcial original del circuito por el puerto serie y almacenarlo en su región de DDR.
    - Después generar un bitstream parcial modificado partiendo del original y guardarlo en BRAM.
    - A continuación realizar una reconfiguración parcial del diseño a testear usando el ICAP y el bitstream parcial modificado.
    - Usar la función de "*ejecutar"* y *"enviar los errores"* por el puerto serie.
    - Hacer la reconfiguración parcial del circuito con el diseño original.

  Repetir desde 2. para cada bit de configuración del bitstream parcial.
- **Comprobar comunicación**: Leer una palabra de la entrada de la uart y si coincide con lo esperado, enviar una respuesta predeterminada a Nessy 5.0. por la uart también.

4.1.2. Bus OPB y LMB

El core OPB_v20 (Ver Figura 44) es usado como bus de interconexión en los sistemas basados en procesadores empotrados en FPGA's de Xilinx. Este bus incluye un árbitro OPB desarrollado por IBM.

En el proyecto para la Virtex II pro usamos la versión v1.10c. Sus características son:

- Incluye un árbitro OPB configurable.
- Incluye soporte para hasta 16 maestros y cualquier número de esclavos.
- Incluye una entrada de reset para el Watchdog Timer

El bus LMB, es un bus especial usado solo para conectar el microblaze con los puertos de datos e instrucciones a dispositivos de alta velocidad, normalmente BRAM.

La versión usada en nuestro proyecto para la Virtex II pro es la v1.00a. Algunas características de este bus son:

- Poca utilización de recursos de la FPGA.
- Buses de lectura y escritura separados.
- No requiere árbitro.

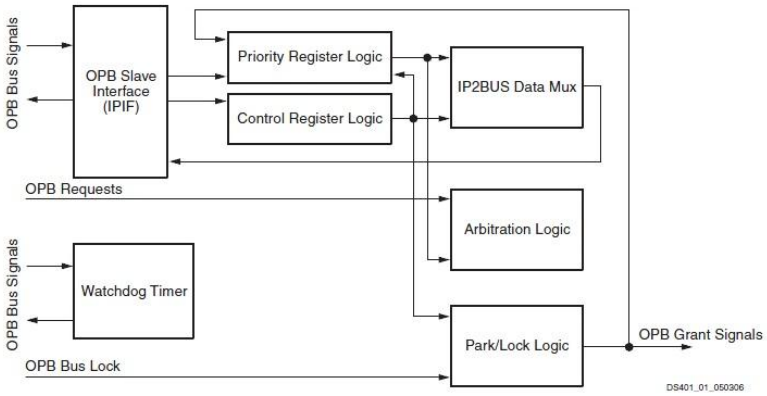

Figura 44: Descripción funcional OPB_V20

En nuestro proyecto, usamos un bus OPB como bus del sistema, con el que conectamos el microblaze a los demás dispositivos como se ha visto anteriormente y dos buses LMB conectados al microblaze que van a dos controladores de BRAM, uno para datos y otro para instrucciones, para poder leer a la vez un dato y una instrucción, tal como se puede apreciar en la siguiente figura:

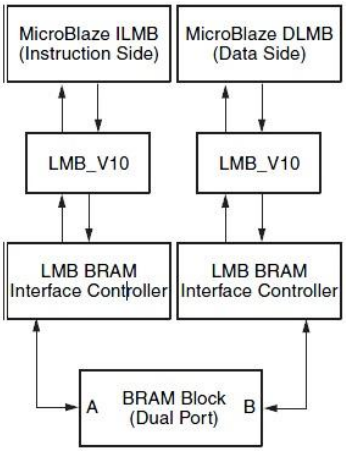

Figura 45: Caso de uso bus LMB

4.1.3. Memoria DDR RAM

Debido a que con nuestra solución debemos almacenar toda la información en la placa XUP para evitar la transmisión por el puerto serie de datos repetidos, necesitamos un dispositivo de almacenamiento. Ambas placas XUP poseen una memoria DIM DDR RAM de 512MB.

Para poder acceder a la RAM necesitamos un controlador. Xilinx provee un controlador de memoria multi-puerto (MPMC). (Ver Figura 46)

Este core soporta memorias SDRAM, DDR, DDR2. Provee acceso a memoria desde uno de hasta ocho puertos, donde cada puerto puede ser elegido de un conjunto de "Personality Interface Modules" (PIMs) que permiten conectarse a los PowerPC y microblaze cores a través de los buses PLB o OPB.

Algunas características del MPMC:

- Soporte para SDMA.
- Soporte para lectura de 2 datos a la vez (DDR/DDR2/DDR3/LPDDR) y lectura de 1 dato a la vez (SDR).
- Soporte para código de corrección de errores (ECC), monitoreo de rendimiento (PM) y registro de depuración.
- Parametrizable: número de puertos, número de bits por palabra, configuración de FIFO's.
- Configuración de algoritmos de arbitraje.
- Generador de interfaces de memoria basado en PHY

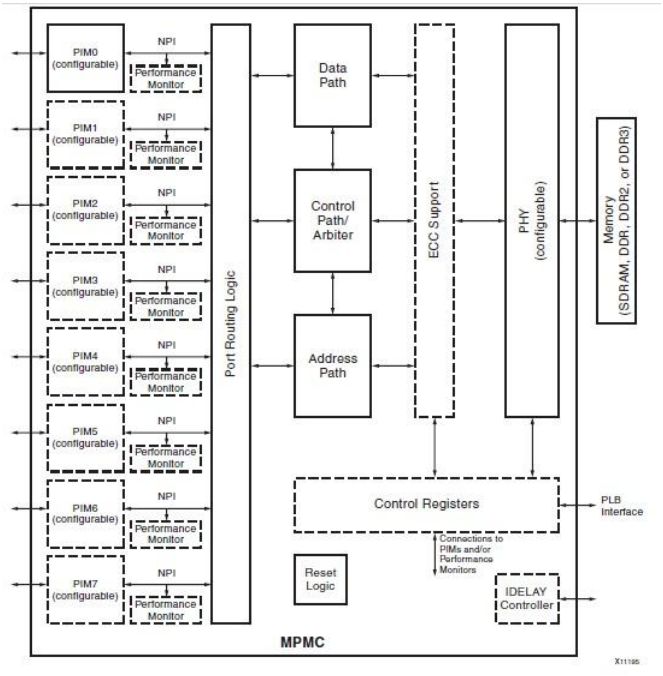

Figura 46: Diagrama de bloques, mpmc

En la siguiente figura se ve cómo hemos configurado el MPMC en nuestro diseño y de esta forma, es nuestro procesador microblaze, quien se encarga de leer y escribir datos en la DDR.

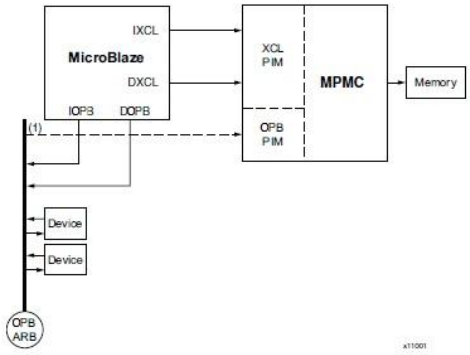

Figura 47: Caso de uso DDR RAM con microblaze y OPB

4.1.4. Componentes propios

Para el desarrollo del proyecto hemos tenido que implementar componentes propios que no estaban disponibles en las librerías del EDK y que nos resultaban imprescindibles. Estos módulos son *"adaptador_uart"*, "adaptador_UART, *adaptador_circuito"* y *"circuito"*. Más adelante se explicarán en detalle.

La peculiaridad de los dos primeros módulos es la comunicación con el bus OPB, ya que en el software del microblaze es imprescindible poder acceder a estos componentes. En nuestro caso para acceder a estos componentes hemos definido diferentes registros de 32 bits de anchura que se corresponderán con diferentes direcciones de memoria. De esta forma, el microblaze podrá realizar lecturas y escrituras en direcciones de memoria para comunicarse con cada componente.

Un bus OPB posee las siguientes señales: `Bus2IP_Clk`, `Bus2IP_Reset`, `Bus2IP_Data`, `Bus2IP_BE`, `Bus2IP_RdCE` y `Bus2IP_WrCE`. En el siguiente código se puede apreciar un código VHDL donde se escribe y lee respectivamente en diferentes registros:

```
PROCESO_DE_ESCRITURA : process( Bus2IP_Clk ) is
begin
  if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
      if Bus2IP_Reset = '1' then
          -- INICIALIZAR EL VALOR DEL REGISTRO CUANDO HAYA UN RESET
          registro_x <= (others => '0');
      else
          case Bus2IP_WrCE is
              when "1000" =>
                  -- CARGAR EL VALOR DEL REGISTRO_X TOMANDO EL VALOR DE Bus2IP_Data
                  -- Hay que tener en cuenta la mascara Bus2IP_BE
                  for byte_index in 0 to (C_DWIDTH/8)-1 loop
                      if ( Bus2IP_BE(byte_index) = '1' ) then
                          registro_x(byte_index*8 to byte_index*8+7) <= Bus2IP_Data(byte_index*8 to byte_index*8+7);
                      end if;
                  end loop;
              when "0100" =>
                  -- DE FORMA ANALOGA
              when "0010" =>
                  -- DE FORMA ANALOGA
              when "0001" =>
                  -- DE FORMA ANALOGA
              when others => null;
          end case;
      end if;
  end if;
```

```
PROCESO_DE_LECTURA : process( Bus2IP_RdCE, registro_x) is
begin
    -- DEPENDIENDO DE LA DIRECCION A LA QUE ACCEDAS
    case Bus2IP_RdCE is
        when "1000" =>
            -- CARGAS EL DATO DEL REGISTRO EN EL BUS DE DATOS
            IP2Bus_Data <= registro_x;
        when "0100" =>
            -- ANALOGO
        when "0010" =>
            -- ANALOGO
        when "0001" =>
            -- ANALOGO
        when others =>
            IP2Bus_Data<= (others => '0');
    end case;
end process PROCESO_DE_LECTURA;
```

4.1.4.1. Adaptador uart

Este módulo es el encargado de comunicar la placa con el exterior mediante el protocolo de comunicación RS232. Posee una interfaz de comunicación con el bus OPB, de forma que mediante la escritura/lectura de diferentes registros el microblaze puede intercambiar datos con el exterior.

Como cualquier componente que se utiliza en la herramienta de Xilinx EDK, tiene que tener unos ficheros concretos y con una determinada estructura. En la siguiente imagen se puede apreciar los directorios y ficheros de este módulo, todos dentro de la carpeta *"pcores"* dentro del directorio del *"proyecto_EDK"*.

Figura 48: Estructura de carpetas adaptador_uart

En la carpeta *"projnav"* está el proyecto en la herramienta Xilinx ISE, en la carpeta *"vhdl"* estas los archivos .vhd y en la carpeta *"data"* están el fichero *"adaptador_uart_v2_1_0.pao"* y *"adaptador_uart_v2_1_0.mpd".*

El fichero .mdp contiene la información sobre las entradas/salidas del modulo, la señal de reset y el reloj. En este caso aparte de las señales de control del bus OPB, reset y reloj, están las señales rx y tx que son señales externas que irán conectadas a pines de la FPGA y servirán para enviar/recibir los datos.

- **Interfaz software**
  - Registro de salida de la uart:
    - Dirección: 0x00.
    - Mapa de memoria:

Bits	**31-0**
Función	Dato de 32 bits para la salida uart

    - Campos:

      Bits 31-0: Es un registro de escritura y la información que se escribe son los datos que se quieren enviar por la salida de la uart

  - Registro de salida ocupada
    - Dirección: 0x04.
    - Mapa de memoria:

Bits	**31-0**
Función	Información de si se está enviando un dato por la uart

    - Campos:

      Bits 31-0: Es un registro de lectura y si todos los bits están a '1' es que todavía se está enviando una palabra.

  - Registro de entrada de la uart
    - Dirección: 0x08.
    - Mapa de memoria:

Bits	**31-0**
Función	Dato de 32 bits de la entrada uart

- Campos:
  Bits 31-0: Es un registro de lectura y se leerán los datos de 32 bits que entran por la uart

- Registro de entrada lista
  - Dirección: 0x0C.
  - Mapa de memoria:

Bits	31-0
Función	Información de si hay un nuevo dato recibido por la uart

  - Campos:
    Bits 31-0: Es un registro de lectura y si todos los bits están a '1' será que hay un nuevo dato de 32 bits que ha entrado por la uart.

NOTA: Cuando se habla de lectura y escritura, se hace pensando en el punto de vista del programa software que va a ejecutarse, es decir, desde el punto de vista del microblaze.

- **Diagrama de bloques**

Este modulo puede está dividido en dos grandes bloques:

- Control de OPB (ficheros *"adaptador_uart.vhd"* y *"user_logic.vhd"*)
- Control de datos (ficheros *"Entrada_salida.vhd"*, *"rx_serie.vhd"* y *"tx_serie.vhd"*)

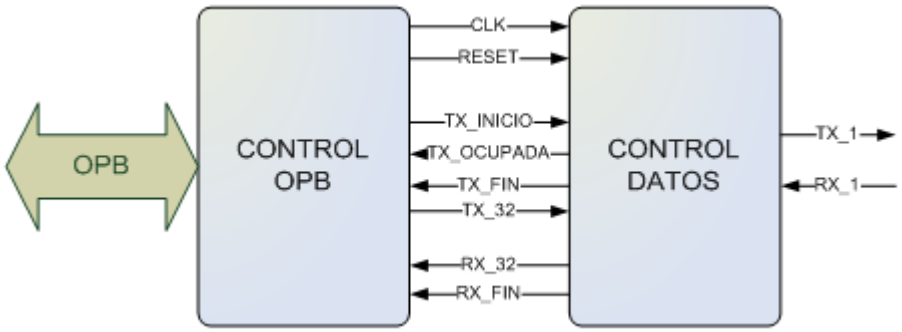

Figura 49: Diagrama de bloques adaptador_uart

Como se ha explicado en el apartado anterior, nuestro modulo posee cuatro direcciones de memoria accesibles desde el microblaze. A partir de la escritura/lectura de estos registros se consiguen todas las señales que aparecen en la imagen anterior.

- La señal `TX_INICIO` se activa cuando el microblaze realiza una escritura en el registro de salida de la uart.
- El dato `TX_32` se valida cuando la señal anterior se pone a '1' y en ese dato estará el dato que se quiere transmitir por la uart.

- La señal `TX_OCUPADA` permanecerá a '1' mientras se esté transmitiendo una palabra de 32 bits. Que esta señal permanezca a '1' quiere decir que el registro de salida ocupada permanecerá con el valor 0xFFFFFFFF.
- La señal `TX_FIN` se activará cuando se haya terminado de enviar la palabra de 32 bits por la uart
- El dato `RX_32` se cargará cuando por la entrada de la uart se han recibido 32 bits de datos. Este dato se cargará en el registro de entrada de la uart.
- La señal `RX_FIN` indicará cuando la señal `RX_32` posee un valor correcto. Que esta señal esté activa significa que el registro de entrada lista tenga un valor de 0xFFFFFFFF.

CONTROL DE DATOS

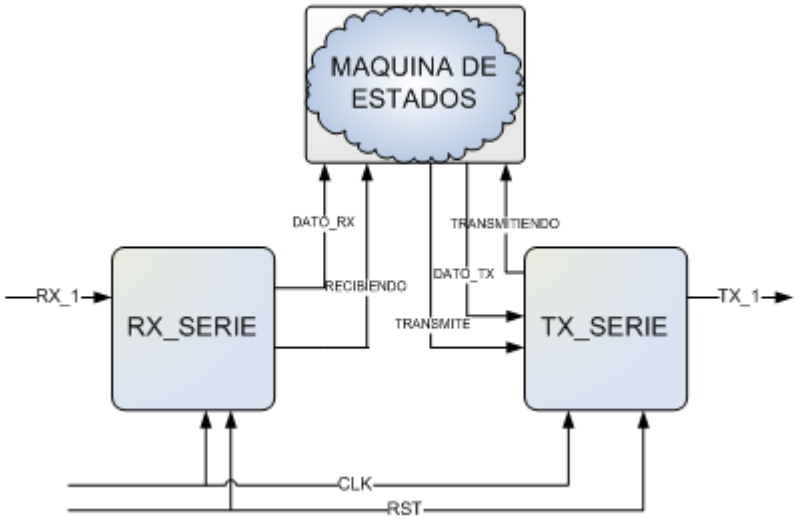

Figura 50: Control adaptador_uart

El control de datos se encarga principalmente de unir datos de 8 bits que enviar/reciben los modulos de *"tx_serie"* y *"rx_serie",* respectivamente, en datos de 32 bits, que son los que se van a escribir y leer de los registros del módulo *"adaptador_uart".*

Para implementar este modulo se han necesitado dos maquinas de estados, una para la transmisión y otra para la recepción.

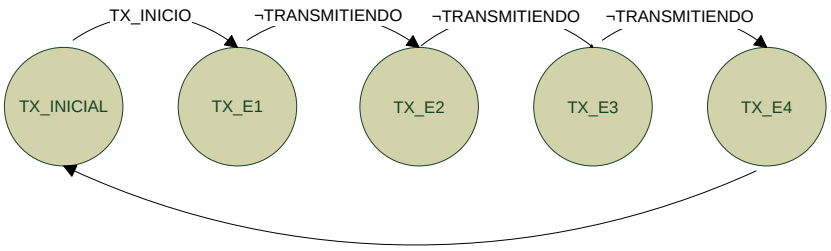

Figura 51: Máquina de estados transmisión adaptador_uart

Para la transmisión, como se puede ver en la figura de arriba, hacen falta los siguientes estados:

- TX_INICIAL: Se permanecerá en este estado hasta que la señal `TX_INICIO`, procedente del módulo control OPB se active. Se pasará al estado TX_E1.
- TX_E1**:** Se permanecerá en este estado mientras la señal `TRANSMITIENDO`=‘1’, o lo que es lo mismo, mientras se transmita el primer dato de 8 bits (primer byte de `TX_32`). La señal `TRANSMITE` permanecerá a ‘1’ y se cambiará de estado cuando `TRANSMITIENDO` pase a valer ‘0’.
- TX_E2**:** Se permanecerá en este estado mientras la señal `TRANSMITIENDO`=‘1’, o lo que es lo mismo, mientras se transmita el segundo dato de 8 bits (segundo byte de `TX_32`). La señal `TRANSMITE` permanecerá a ‘1’ y se cambiará de estado cuando `TRANSMITIENDO` pase a valer ‘0’.
- TX_E3: Se permanecerá en este estado mientras la señal `TRANSMITIENDO`=‘1’, o lo que es lo mismo, mientras se transmita el tercer dato de 8 bits (tercer byte de `TX_32`). La señal `TRANSMITE` permanecerá a ‘1’ y se cambiará de estado cuando `TRANSMITIENDO` pase a valer ‘0’.
- TX_E4**:** Se permanecerá en este estado mientras la señal `TRANSMITIENDO`=‘1’, o lo que es lo mismo, mientras se transmita el cuarto dato de 8 bits (cuarto byte de `TX_32`). La señal `TRANSMITE` permanecerá a ‘1’ y se cambiará de estado cuando `TRANSMITIENDO` pase a valer ‘0’. En este momento se activará la señal `TX_FIN`.

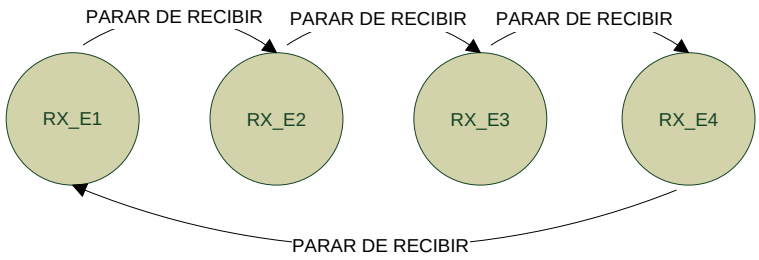

Figura 52: Máquina de estados recepción adaptador_uart

Para la recepción, según muestra la figura de arriba, hacen falta los siguientes estados:

- RX_1**:** Se permanece en este estado hasta que se para de recibir bits del módulo *“rx_serie”*, es decir, la señal `RECIBIENDO` pasa de valer ‘1’ a valer ‘0’. Cuando se termina de recibir se carga el primer byte de la palabra `RX_32` con los 8 bits recibidos.
- RX_2**:** Se permanece en este estado hasta que se para de recibir bits del módulo *“rx_serie”*, es decir, la señal `RECIBIENDO` pasa de valer ‘1’ a valer ‘0’. Cuando se termina de recibir se carga el segundo byte de la palabra `RX_32` con los 8 bits recibidos.

- RX_3**:** Se permanece en este estado hasta que se para de recibir bits del módulo *"rx_serie"*, es decir, la señal `RECIBIENDO` pasa de valer '1' a valer '0'. Cuando se termina de recibir se carga el tercer byte de la palabra `RX_32` con los 8 bits recibidos.
- RX_4**:** Se permanece en este estado hasta que se para de recibir bits del módulo *"rx_serie"*, es decir, la señal `RECIBIENDO` pasa de valer '1' a valer '0'. Cuando se termina de recibir se carga el cuarto byte de la palabra `RX_32` con los 8 bits recibidos.

4.1.4.2. Adaptador circuito

Este módulo sirve para generalizar nuestro sistema, de tal forma que cualquier diseño que se utilice en nuestra herramienta se pueda conectar a un bus OPB. Para ello nuestro modulo genérico tendrá tres salidas: el reloj de nuestro circuito, la entrada del circuito y la salida del circuito.

Como cualquier componente que se utiliza en la herramienta de Xilinx EDK, tiene que tener unos ficheros concretos y una determinada estructura. En la siguiente imagen se puede apreciar los directorios y ficheros de este módulo, todos dentro de la carpeta *"pcores"* dentro del directorio del "*proyecto_EDK".*

Figura 53: Estructura carpetas adaptador_circuito

En la carpeta *"projnav"* está el proyecto realizado con la herramienta xilinx ISE, en la carpeta *"vhdl"* estas los archivos .vhd y en la carpeta *"data"* están el fichero *"adaptador_circuito_v2_1_0.pao"* y *"adaptador_circuito_v2_1_0.mpd".*

El fichero .mdp contiene la información sobre las entradas/salidas del modulo, la señal de reset y el reloj. En este caso, aparte de las señales de control del bus OPB, reset y reloj, no habrá ninguna señal más, ya que este modulo no necesita intercambiar información con el exterior (se hace toda la comunicación a través de la memoria).

- **Interfaz software**
  - Registro de entrada del circuito:
    - Dirección: `0x00`.
    - Mapa de memoria:

Bits	**31-0**
Función	Entrada del circuito

- Campos:

  Bits 31-0: Es un registro de escritura y será la entrada que tenga el circuito en un determinado ciclo de reloj.

- Registro de entrada del circuito:
  - Dirección: `0x04`.
  - Mapa de memoria:

Bits	31-0
Función	Salida del circuito

  - Campos:

    Bits 31-0: Es un registro de lectura e indicará la salida de nuestro circuito en un determinado flanco de reloj.

    NOTA: Cuando se habla de lectura y escritura, se hace pensando en el punto de vista del programa software que va a ejecutarse, es decir, desde el punto de vista del microblaze.

- **Interfaz software**

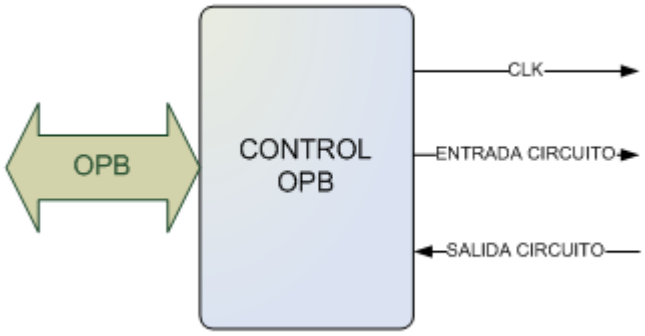

Figura 54: Interfaz adaptador_circuito

`ENTRADA CIRCUITO` se actualiza cuando escribes en el registro de entrada del circuito. Es un nuevo ciclo, cuando se ha terminado de escribir en el registro, `CLK` dará un pulso de reloj, para que el reloj del circuito que vaya conectado al otro extremo cambie de valor los registros internos de su circuito con los nuevos valores de la entrada.

Un ciclo más tarde se actualiza el valor de `SALIDA CIRCUITO`, que se escribirá en el registro definido para ello. A partir de este momento el microblaze podrá hacer lecturas para ver la salida del circuito.

4.1.4.3. Circuito bajo test

Este módulo no tiene ninguna característica en especial. Es una entidad en vhdl, con unas entradas, unas salidas y en caso de no ser combinacional, un reloj. No hace falta

decir que la salida de este módulo, en caso de ser un modulo combinacional estará disponible antes del flanco de reloj.

Como cualquier componente que se utiliza en la herramienta de xilinx EDK, tiene que tener unos ficheros concretos y con una determinada estructura. En la siguiente imagen se puede apreciar los directorios y ficheros de este módulo, todos dentro de la carpeta *"pcores"* dentro del directorio del *"proyecto_EDK".*

Figura 55: Estructura carpetas circuito

En la carpeta *"projnav"* está el proyecto en la herramienta xilinx ISE, en la carpeta "*vhdl*" estas los archivos .vhd y en la carpeta *"data"* están el fichero *"adaptador_circuito_v2_1_0.pao"*.

En este caso no el fichero .mdp no contendrá ninguna información sobre las señales del bus OPB, ya que va a ser un modulo que irá conectado a otro modulo y no al bus. Solo va a contener la información de `CLK`, `ENTRADA CIRCUITO` y `SALIDA CIRCUITO`.

- **Diagramas de bloques**

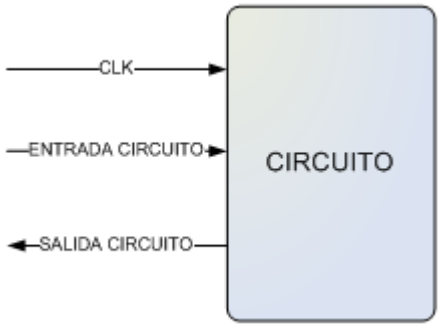

Figura 56: Interfaz circuito

Como se puede apreciar en el diagrama anterior, este módulo es la continuación del adaptador circuito, con las señales de `CLK`, `ENTRADA CIRCUITO` y `SALIDA CIRCUITO`.

En el código siguiente se puede apreciar un ejemplo de código vhdl donde CONTADOR habrá que sustituirlo por el nombre de nuestro circuito:

```
library ieee;
use ieee.std_logic_1164.all;

entity circuito is
port(
    dato_listo : in std_logic;
    entrada_circuito : in std_logic_vector(0 to 31);
    salida_circuito :out std_logic_vector(0 to 31)
);
end circuito;

architecture Behavioral of circuito is

    ----------------- SEÑALES DEL CIRCUITO ----------------------
    signal SENAL_SALIDA          : std_logic_vector(0 to 7);
    -------------------------------------------------------------

    --------------------- COMPONENTE ----------------------------
    COMPONENT CONTADOR PORT (
        signal RESET : in std_logic;
        signal CLK : in std_logic;
        signal ENABLE : in std_logic;
        signal LOAD : in std_logic;
        signal DATA_LOAD : in std_logic_vector(0 to 7);
        signal SALIDA : out std_logic_vector(0 to 7)
        );
    END COMPONENT;
    -------------------------------------------------------------

begin

    ---------------- ASIGNACIÓN DE LAS SALIDAS -------------------
    salida_circuito(0 to 7) <= SENAL_SALIDA(0 to 7);
    -------------------------------------------------------------
    ------------------------ MAPEO --------------------------------
    instancia : CONTADOR PORT MAP(
    RESET => entrada_circuito(0),
    CLK => dato_listo,
    ENABLE => entrada_circuito(1),
    LOAD => entrada_circuito(2),
    DATA_LOAD(0 to 7) => entrada_circuito(3 to 10),
    SALIDA => SENAL_SALIDA
    );
    -------------------------------------------------------------
end Behavioral;
```

4.1.5. Icap

La reconfiguración parcial en la familia de FPGAs Virtex II Pro y Virtex V se puede realizar mediante una interfaz interna de reconfiguración conocida como ICAP. Existen otras opciones como el modo SelectMap o el modo JTAG que necesitan un controlador externo para enviar los datos de reconfiguración, pero resultan mucho más lentos.

La interfaz de este puerto viene mostrada en la siguiente figura:

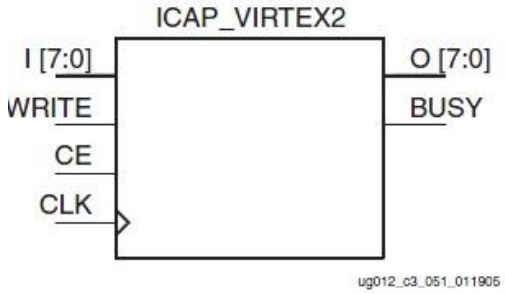

Figura 57: Interface Icap

ICAP Port		Equivalent SelectMAP Port
Name	Direction	
BUSY	Output	BUSY
O[0:7]	Output	D[0:7] when SMAP port is set for read control
CE	Input	CS_B
CLK	Input	CCLK
I[0:7]	Input	D[0:7] when SMAP port is set for write control
WRITE	Input	RDWR_B

Figura 58: Equivalencias entre interfaces Icap y SelectMap

La interfaz del ICAP es un subconjunto de la interfaz SelectMap (Ver Figura 58) y usa el mismo protocolo que ésta en modo esclavo, aunque hay una diferencia y es que el ICAP separa los datos de entrada y los de salida en vez de tener un único puerto bidireccional de datos.

Algunas características de este core son:

- Interface para bus OPB.
- Permite la lectura/escritura de CLB LUTs
- Permite la lectura/escritura de las propiedades de los CLB Flip-flops.
- Soporte para los procesadores empotrados microblaze y powerPC.
- Frecuencia máxima de trabajo a 50Mhz en la virtex II pro.

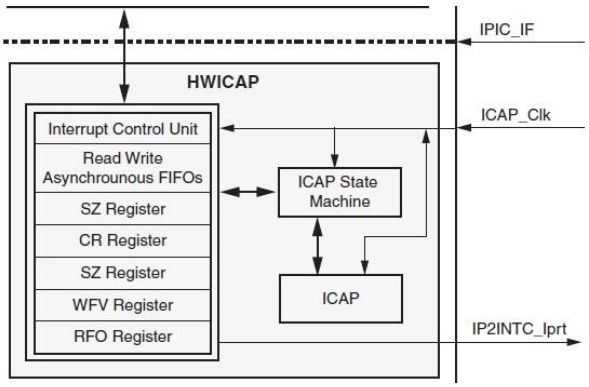

Figura 59: Diagrama de bloques, hwicap core

En el proyecto, hemos usado el core dado por Xilinx *"hwicap v1.00b"* (Ver Figura 59) para la Virtex II. Tenemos el core conectado al bus del sistema OPB (Ver Figura 60) por lo que podemos comunicarnos con el ICAP a través del microblaze escribiendo o leyendo en los registros pertinentes.

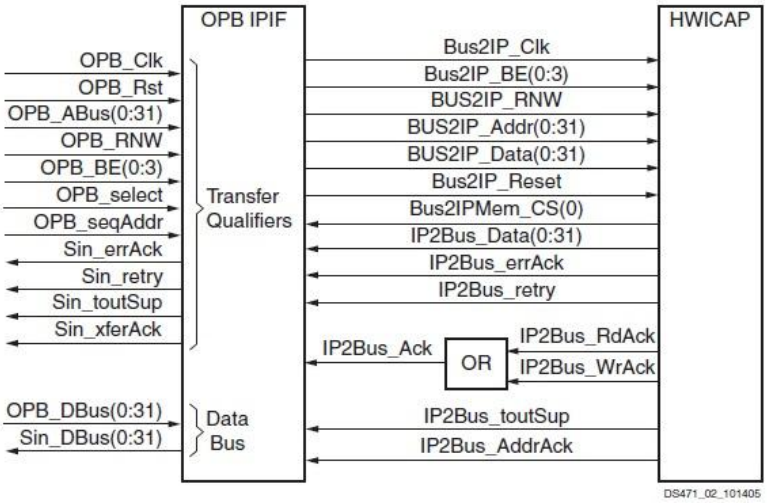

Figura 60: Interface en modo esclavo con el bus OPB

Para lograr la reconfiguración partimos de un bitstream parcial enviado a través de la uart y almacenado en RAM local previamente. Para modificar el circuito de la FPGA, determinamos qué frames deben ser cambiados, el microblaze genera un bitstream parcial modificado, lo almacena en la BRAM y da la orden de que inicie la reconfiguración parcial.

Ahí es donde se encarga el ICAP de ir leyendo el bitstream modificado de la BRAM a través del OPB e ir escribiéndolo en la memoria de configuración de la FPGA.

4.2. Modificaciones software

El programa mapeado en el microblaze es el encargado de realizar la mayor parte de las acciones que antes hacía Nessy 2.0 en el computador. Esto es, está a la espera de recibir un comando proveniente de Nessy 5.0 a través del puerto serie y en función del comando, hace una cosa u otra. Esto significa que tiene que estar sincronizado con Nessy 5.0 en todo momento.

Los ficheros *libSI.h* y *libSI.c* contienen las definiciones de macros y variables necesarios así, como la implementación de funciones para el manejo del controlador de uart y del ICAP.

Por un lado, en *libSI.h* definimos los registros del controlador de uart y los del controlador del circuito, así como los distintos rangos de memoria donde irá el testbench, el golden, los errores, el bitstream original, el bitstream modificado y el bitstream restorer. En la siguiente región de código podemos apreciar esto:

```
    #define UART_SALIDA                 (XPAR_ADAPTADOR_UART_0_BASEADDR)
    #define UART_SALIDA_OCUPADA         (XPAR_ADAPTADOR_UART_0_BASEADDR + 4)
    #define UART_ENTRADA                (XPAR_ADAPTADOR_UART_0_BASEADDR + 8)
    #define UART_ENTRADA_LISTA          (XPAR_ADAPTADOR_UART_0_BASEADDR + 12)

    #define CIRCUITO_ENTRADA            (XPAR_ADAPTADOR_CIRCUITO_0_BASEADDR)
    #define CIRCUITO_SALIDA             (XPAR_ADAPTADOR_CIRCUITO_0_BASEADDR + 4)

    #define ICAP                        (XPAR_OPB_HWICAP_0_BASEADDR)

    #define RAM_TESTBENCH
(XPAR_DDR_512MB_64MX64_RANK2_ROW13_COL10_CL2_5_MEM0_BASEADDR + 0x00100000)
    #define RAM_GOLDEN
(XPAR_DDR_512MB_64MX64_RANK2_ROW13_COL10_CL2_5_MEM0_BASEADDR + 0x00200000)
    #define RAM_ERRORES
(XPAR_DDR_512MB_64MX64_RANK2_ROW13_COL10_CL2_5_MEM0_BASEADDR + 0x00300000)
    #define RAM_BITSTREAM_ORIGINAL
(XPAR_DDR_512MB_64MX64_RANK2_ROW13_COL10_CL2_5_MEM0_BASEADDR + 0x00400000)
    #define RAM_BITSTREAM_MODIFICADO    (0x0000E000)
    #define RAM_BITSTREAM_RESTORER      (0x0000F000)
```

Con esto nos hacemos una idea de la distribución de los archivos almacenados en la BRAM y RAM según lo anterior.

En la BRAM solo estará el código del programa que ejecutará el microblaze desde la posición 0x0 y los bitstream modificados con errores inyectados en las posiciones 0xE000 y 0xF000 respectivamente. Se ha decidido almacenar estos archivos en la BRAM debido a que la BRAM es una memoria más rápida que la RAM externa y estos archivos van a ser leídos y escritos muchas veces a lo largo de una inyección.

Para la RAM, la distribución de los archivos se ve en la Figura 61.

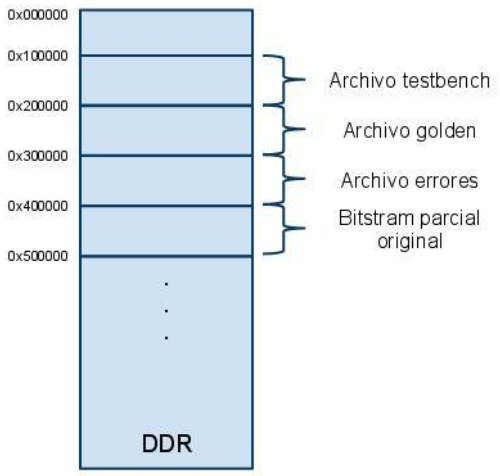

Figura 61: Distribución de archivos en la RAM de la FPGA

Y por último en *libSI.h* definimos la cabecera de las funciones que luego implementaremos en *libSI.c*:

```
void enviarDato(volatile unsigned int *dato);
void recibirDato(volatile unsigned int *dato);
void inicializarUart();

void inicializarIcap(XHwIcap * icap);
void programar_icap(XHwIcap * icap,Xuint32 *data, Xuint32 tamano);
```

Las funciones implementadas en *libSI.c* se dividen en dos grupos, las relacionadas con el manejo de la uart y las relacionadas con el manejo del ICAP. El código de las primeras viene mostrado a continuación:

```
void enviarDato(volatile unsigned int *dato){
    while(*pto_uart_salida_ocupada == 0xFFFFFFFF)
        ;
    *pto_uart_salida = *dato;
}

void recibirDato(volatile unsigned int *dato){
    while(*pto_uart_entrada_lista != 0xFFFFFFFF)
        ;
    *dato = *pto_uart_entrada;
    *pto_uart_entrada_lista = 0x00000000;
}

void inicializarUart(){
    *pto_uart_entrada_lista = 0x00000000;
 }
```

Con estas funciones ya se puede recibir y enviar datos a través de los registros del módulo “*adaptador_uart”*.

Antes de hacer uso de la uart, es necesario hacer una llamada a `inicializarUart()`.
`enviarDato()` y `recibirDato()`, reciben un puntero del dato que se quiere enviar o el puntero donde se quiere escribir el dato recibido respectivamente.

```
    void inicializarIcap(XHwIcap * icap){
        XHwIcap_Initialize(icap, XPAR_OPB_HWICAP_0_DEVICE_ID,
XHI_XC2VP30);
    }

    void programar_icap(XHwIcap * icap, Xuint32 *data, Xuint32 tamano){
        XStatus status = XHwIcap_SetConfiguration(icap, data, tamano);
    }
```

En el código anterior se muestran las 2 funciones necesarias para el manejo del ICAP. Es necesario hacer una llamada a `iniciarlizarIcap()` antes de hacer uso de éste..

`programar_icap()` necesita como parámetros: un puntero a la estructura de datos *XHwIcap* dada por Xilinx e inicializada anteriormente, un puntero al primer dato del bitstream parcial y por último el tamaño de éste. Con esto, al hacer una llamada a esta función la FPGA se reconfigurará parcialmente y la función devolverá 2 en caso de que haya ocurrido algún fallo y 1 en caso de que haya sido posible la reconfiguración.

Una vez explicadas las funciones para el manejo de los dispositivos involucrados en nuestro sistema, la estructura del programa principal (*main.c*) es la siguiente:

```
    int main (void) {
        dato_syn = 204;

        inicializarUart();
        inicializarIcap(&icap);

        while(1){
            recibirDato(dato);
            comando = *dato >> 29;

            switch(comando){
            ...
    }
```

La estructura del código es un bucle *while* con condición siempre verdadera, en el que se lee la entrada del "*adaptador_uart*" y mediante un bucle *switch*, si el dato leído coincide con algún comando conocido, entra a ejecutar su código asociado.
La instrucción "`comando = *dato >> 29;`" realiza un desplazamiento de 29 bits a la derecha para quedarse con los 3 bits más significativos que han sido recibidos por la uart, los cuales, corresponden al comando. Los comandos son:

4.2.1. Cargar testbench

En los bits menos significativos del dato que contiene al comando, se envía el tamaño del testbench. Después se procede a leerlo de la entrada de la uart y escribirlos en la DDR:

```
    case 0x00000000 :
        num_entradas_tb = *dato;
        recibidos = 0;
        while(recibidos<num_entradas_tb){
            recibirDato(pto_ram_testbench + recibidos);
            recibidos = recibidos + 1;
        }
        break;
```

4.2.2. Generar golden

Se ponen los datos del testbench en la entrada del circuito y se guardan las salidas generadas por el circuito en una zona de la DDR asignada para el golden.

```
    case 0x00000001 :
        for(i=0;i<num_entradas_tb;i++){
            *(pto_circuito_entrada) = *(pto_ram_testbench+i);
            *(pto_ram_golden+i) = *(pto_circuito_salida);
        }
        break;
```

4.2.3. Ejecutar

Se ponen los datos del testbench en la entrada del circuito y se comprueban las salidas con su correspondiente dato del golden. En caso de no coincidir una entrada, se escribe en una zona de la DDR asignada para los errores qué bits han fallado para qué número de entrada.

```
    case 0x00000002 :
        ejecutar();
        enviarErrores();
        break;
```

La función ejecutar, lee los datos de la zona de memoria donde está el testbench y los escribe en el registro de la entrada del circuito. Uno por ciclo de reloj. A su vez, cada ciclo de reloj, lee el dato que hay en el registro de salida del circuito y lo compara con el dato correspondiente del golden. En el caso de que no coincidan todos los bits, se escribe la variable "*ciclo_error*" indicando en que ciclo del testbench ha sucedido el error y el vector error indica con "1"s que bits han sido erróneos.

```
ejecutar(){
    ciclo_error = 0;
    error = 0x00000000;
    m=0;
    while(m < num_entradas_tb & error == 0x00000000){
        *(pto_circuito_entrada) = *(pto_ram_testbench+m);
        error = (*(pto_circuito_salida)) ^ (*(pto_ram_golden+m));
        if (error != 0x00000000)
            ciclo_error = m + 1;
        m = m + 1;
    }
}
```

La función `enviarErrores()`, se limita a enviar por el puerto serie el número de ciclo donde ha sucedido el error. Esta decisión de solo enviar el número de ciclo, se ha tomado para no saturar demasiado el puerto.

```
enviarErrores(){
    enviarDato(&ciclo_error);
}
```

4.2.4. Enviar golden

Se envía el golden por la salida de la uart:

```
case 0x00000003 :
    enviados = 0;
    while (enviados<num_entradas_tb){
        enviarDato(pto_ram_golden + enviados);
        enviados = enviados+1;
    }
    break;
```

4.2.5. Cargar golden

Similar al cargar testbench, excepto porque los datos se escriben en la zona de la DDR asignada para el golden.

```
case 0x00000004 :
    recibidos = 0;
    while(recibidos<num_entradas_tb){
        recibirDato(pto_ram_golden + recibidos);
        recibidos = recibidos + 1;
    }
break;
```

4.2.6. Reconfiguración

En la inyección de errores, es el propio microblaze quien genera un bitstream parcial modificado y reconfigura la FPGA para cada bit de la memoria de configuración asignada al diseño a testear.

En primer lugar, pasa el bitstream parcial que contiene las frames que afectan a nuestro circuito a la DDR, el microblaze ejecuta el siguiente código:

```
recibidos = 0;
while(recibidos<tam_bit){
    recibirDato(pto_ram_bitOriginal + recibidos);
    recibidos = recibidos + 1;
}
```

La segunda parte ha sido realizada en conjunto con el estudiante de Master en Investigación en Informática en la Facultad de Informática de la Universidad Complutense de Madrid, Víctor Alaminos. Este estudiante ha desarrollado como proyecto de fin de máster, una función escrita en C `( int = partial(…)`) capaz de generar los bitstream parciales para la Virtex II pro. Estos bit parciales producen en el mapa de bits original, un bitflip en el bit que se pasa por parámetro. Gracias al trabajo conjunto con Víctor, nos ha sido posible integrar su proyecto en la palabra y el bit pasados por parámetro el código que ejecuta nuestro microblaze.

La función `partial()` está cargada en la BRAM del microblaze y es llamada para cada bit que es necesario modificar, encargándose ésta de leer el bitstream parcial original almacenado en la DDR de la FPGA, creando el bitstream parcial modificado que corresponda y almacenándolo en otro rango de direcciones reservados para este propósito.

Como se ha explicado en el apartado 2.1.1.2, la unidad mínima de reconfiguración en una Virtex II pro es una frame. Es importante destacar que una columna entera de CLBs se configura a través de 22 frames para la Virtex II pro de nuestra placa. Cada frame está compuesta de 206 palabras de 32 bits. Por tanto esas 206 palabras de cada frame se encargan de reconfigurar parte de cada uno de los CLBs y de los IOBs que se sitúan arriba y debajo de la frame. En concreto 3 palabras de la frame se corresponden a la configuración de cada IOB.

También es importante destacar que hay frames que reconfiguran IOBs y que las frames que reconfiguran los slices comienzan a partir de la 30.

El código de esta segunda parte es el siguiente:

```
    palabra_inicial = (frame_inicio * 206) + 1;
    for(i = frame_inicio; i < frame_fin ; i++){
        //calculos
        palabra_inicio = (i * 206) + 1;
        gap_inicio = (((159 - sliceY1) / 2) * 80) / 32;
        palabra_gap_inicio = palabra_inicio + 3 + gap_inicio;
        bit_inicio = (32 - ((((159 - sliceY1) / 2) * 80) % 32)) - 1;
        gap_fin = ((((159 - sliceY0) / 2) + 1) * 80) / 32;
        palabra_gap_fin = palabra_inicio + 3 + gap_fin;
        bit_fin = 32 - (((((159 - sliceY0) / 2) + 1) * 80) % 32);
        bit_fin_aux = 0;

        for(j = palabra_gap_inicio; j <= palabra_gap_fin; j++){
            if (j == palabra_gap_fin){
                bit_fin_aux = bit_fin;
            }
            for(k = 0; k <= 31; k++){

                //CREAR BIT ERRONEO Y BIT RESTORER
                partial(j,k,0,palabra_inicial);

                //CARGAR BIT ERRONEO
                programar_icap(&icap ,
               (Xuint32
*)pto_ram_bitModificado,(Xuint32)tam_bit_modificado);

                //Ejecutar el testbench
                ejecutar();
                enviarDato(&dato_syn);
                enviarErrores();

                //CARGAR BIT RESTORER
                programar_icap(&icap ,
               (Xuint32 *)pto_ram_bitRestorer,(Xuint32)tam_bit_restorer);

            }
            //bit_inicio = 31;
        }
    }
```

La información que le pasamos al microblaze es "*frame_inicio*", "*frame_final*", "*palabra_inicio*" y "*palabra_fin*". Estos valores los calculamos mediante las siguientes operaciones:

```
frame_inicio = 30+ (slice_inicialX/2*22);
frame_fin = 30+ ((((slice_finalX)/2)+1)*22);
palabra_inicio = (frame_inicio * 206) + 1;
palabra_fin = (frame_fin * 206);
```

4.2.7. Comprobar comunicación

Al recibir este comando desde Nessy 5.0, lo que hace es enviar otro dato por la uart (el cual Nessy 5.0 estará esperando) para sincronizar FPGA y computador.

```
    case 0x00000006 :
        *respuesta=0xf0f0f0f0;
        enviarDato(respuesta);
```

4.3. Modificaciones en la Aplicación

Las modificaciones software realizadas en nuestro proyecto se han centrado en adaptar la funcionalidad de Nessy 2.0 a nuestros propósitos de pasar más carga de trabajo a la FPGA y usar menos la comunicación por el puerto serie. Además, hemos aplicado una serie de mejoras útiles para futuras ampliaciones del proyecto tales como la gestión de procesos externos y su sincronización, configuración de distintas versiones de Xilinx o la configuración del puerto serie.

La mayoría de las modificaciones han sido internas, por lo que la interfaz de Nessy 5.0 presenta un aspecto muy similar al de la versión 2.0. (Ver Figura 62).

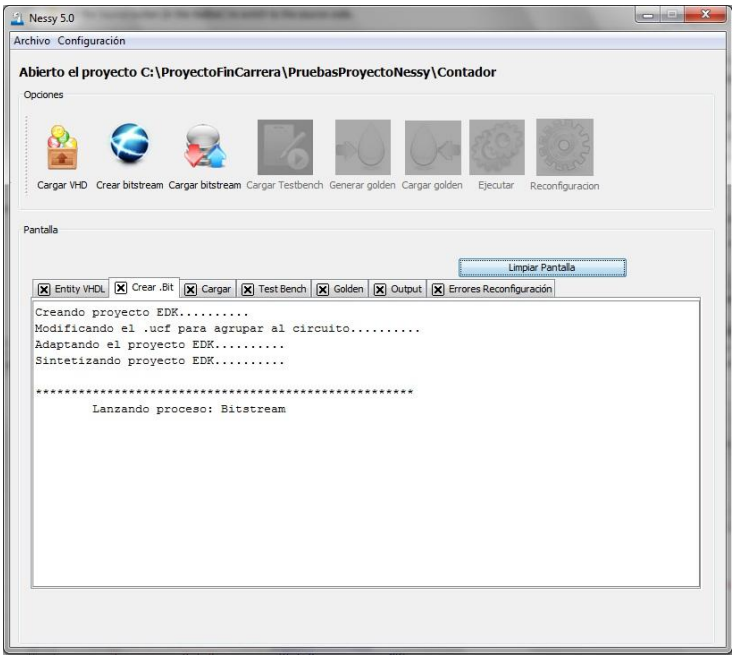

Figura 62: Nessy 5.0

La diferencia más notable en la interfaz es la ausencia de los botones "*parar ejecución*" y "*reanudar ejecución*". Estos eran inservibles para nuestra manera de realizar la ejecución del circuito, puesto que la única relación en una ejecución entre la FPGA con

Nessy 5.0 es al comienzo al enviar la orden "*ejecución*" y después perdemos el control de ésta hasta que haya terminado, ya que todo el trabajo queda delegado al microblaze que es quien se encarga de proveer de entradas al circuito. Al ser el microblaze monothread y estar ocupado ejecutando, no hay forma de detener la ejecución.

Los cambios más significativos en el funcionamiento se han visto reflejados, en su mayoría, en el código asociado a los botones principales. Antes cada botón tenía una función que realizaba el propio Nessy 2.0 en el computador. Ahora esa función se convierte, en la mayoría, en el envío de un comando por el puerto serie para que sea el microblaze el que se encargue de hacerlo en la FPGA.

4.3.1. Funcionalidad de los botones de la interfaz

4.3.1.1. Cargar VHD

Esta opción, que sirve para parsear un archivo fuente VHDL, ha quedado intacta excepto por algunas adaptaciones menores a nuestro gestor de proyectos, tales como la copia de los archivos fuentes a la carpeta *"vhdl"* del proyecto en cuestión o la escritura del nombre del archivo top del diseño en *"proyecto.nessy"*

Por lo demás, permanece igual que en Nessy 2.0. Realiza su función de parsear un archivo .vhd, cargarlo en memoria como antes y mostrar por pantalla la información formateada de la entidad top (Figura 63).

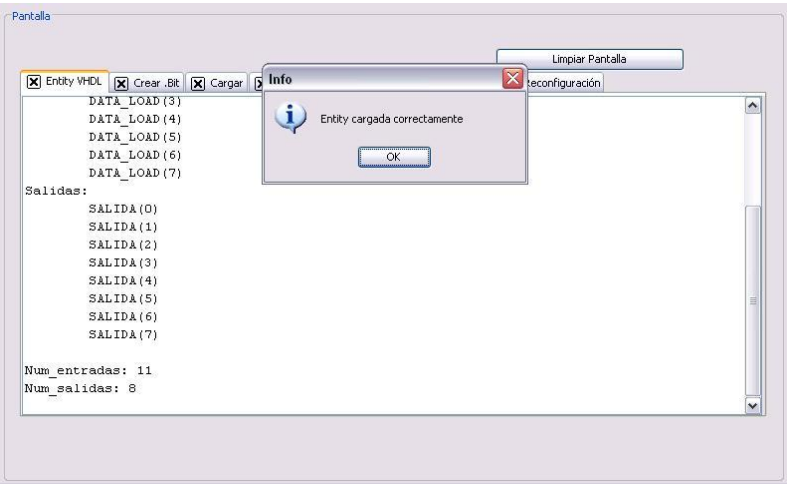

Figura 63: Cargar VHD

4.3.1.2. Crear bitstream

El código que se ejecuta cuando se produce el evento de pulsar el botón "*crear bitstream*" de la interfaz tiene y tenía como objetivo generar un archivo bitstream con el código vhdl y los interfaces añadidos por nuestro sistema. En caso de que la generación no de error, se habilitará el botón "*cargar bitstream*". En el siguiente código se muestra el proceso:

```
    private void _btnCrearBitActionPerformed(java.awt.event.ActionEvent
evt) {
        try {
            if (this.crearBit()) {
                 _btnCargarBit.setEnabled(true);
            } else {
                JOptionPane.showMessageDialog(this, "No se ha podido
generar el .bit correctamente", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
```

En caso de que la función `crearBit()` termine correctamente habilitamos el botón de la interfaz *"cargar bit.*,En caso contrario, mostramos por pantalla un mensaje de error.
En `crearBit()` los cambios son significativos. Antes se limitaba a añadir los ficheros .vhdl del diseño a una entidad top que utilizaba una interfaz entre la uart y el diseño para poder recibir datos desde Nessy 2.0 y transmitir sus salidas de vuelta para después sintetizarlo llamando al ISE. Esta función ahora hace lo siguiente:

1. **Introducir los SLICES** en donde se quiere reservar espacio para ubicar el circuito a testear.
2. **Copiar el proyecto EDK** que contiene todo el sistema (microblaze, buses, controladores, adaptadores, ICAP etc.) a la carpeta del proyecto (del gestor de proyectos se hablará más tarde en esta misma sección). Además, añadir al pcore *"circuito"* los archivos VHDL del diseño a testear.
3. **Modificar el archivo de restricciones** system.ucf del proyecto EDK para añadir la restricción de área del circuito a testear.
4. **Crear el archivo circuito.vhd** del pcore *"circuito"*, que tenga una instancia del top del diseño VHDL que se quiere testear.
5. **Modificar el archivo .pao**
6. **Generar el bitstream del sistema** con el circuito a testear incluido. Esto lo hacemos en dos partes para facilitar la depuración:
    a. Generar los netlists del sistema mediante el programa *bits* de Xilinx.
    b. Mapear el software en la BRAM asociada al microblaze mediante el programa *data2mem* de Xilinx también.

Como las demás acciones asociadas a botones en la interfaz, Nessy 5.0 mostrará en tiempo real el estado de la creación del bitstream. Ver Figura 64:

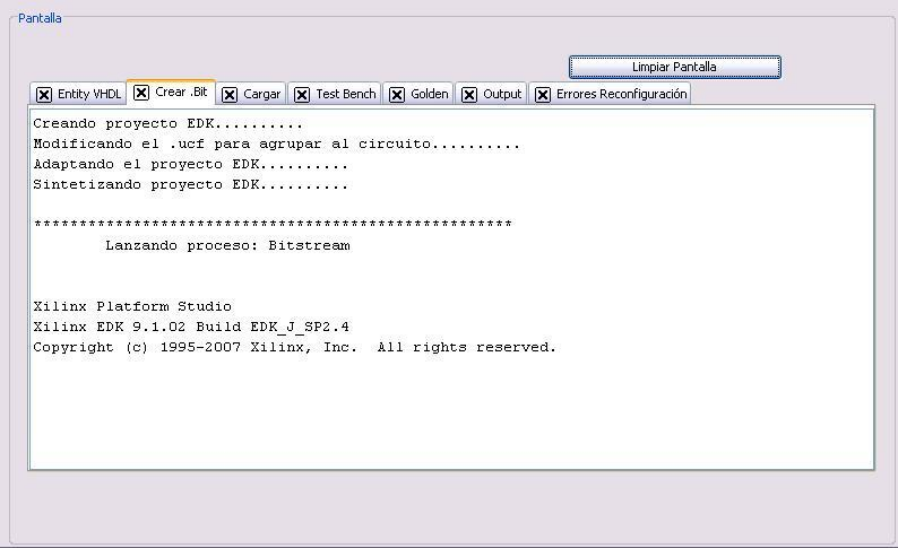

Figura 64: Crear bitstream

A continuación se explican los puntos anteriores con más detalle:

- **Introducir los slices**: Los slices los introduce el usuario mediante una interfaz gráfica. Como se puede apreciar en la siguiente imagen el usuario introduce el número de slices de ancho y el número de slices de alto. Estos slices empezarán a contarse desde la esquina superior derecha de la FPGA.

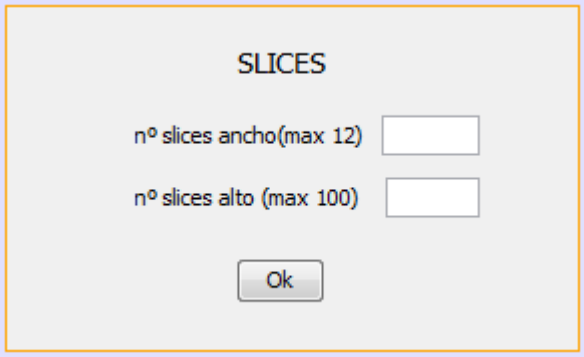

Figura 65: GUIInyeccionErrores

Estas cantidades introducidas por el usuario, se suman a una restricción de tamaño para evitar que el circuito quede instanciado en una zona de la FPGA en donde hay más componentes y por tanto solo se inyecte bitflips en nuestro circuito.

- **Copiar el proyecto EDK:** La copia de estos archivos se hace mediante el programa *xcopy* de Windows.

```
    private boolean copiarArchivos() throws Exception{
        Process proc_proyecto = Runtime.getRuntime().exec("xcopy.exe /Y
/E /Q "+"\""+RUTA_NESSY+"\\Proyecto_EDK_V2\" \"" +
RUTA_PROYECTO_ACTUAL_EDK+"\"");
        proc_proyecto.waitFor();
        if(proc_proyecto.exitValue() != 0){
            return false;
        }
        Process proc_archivo = Runtime.getRuntime().exec("xcopy /Y /Q
\""+RUTA_PROYECTO_ACTUAL_VHDL+"\"
\""+RUTA_PROYECTO_ACTUAL_EDK+"\\pcores\\circuito_v1_00_a\\hdl\\vhdl\\\"")
;
        proc_archivo.waitFor();
        if(proc_archivo.exitValue() != 0){
            return false;
        }


        return true;
    }
```

- **Modificar el archivo de restricciones**: Esto lo hacemos a través de la función `modificaUcf(int nsliceX, int nsliceY)`. Que usa la propiedad *"AREA_GROUP"* para añadir una restricción al archivo system.ucf.

  Para el ejemplo de un circuito al que se le ha asignado un tamaño de 2x4 slices, el texto a añadir en system.ucf es el siguiente (X91 e Y159 es el slice situado en la esquina superior izquierda):

```
INST "circuito_0" AREA_GROUP = "pblock_circuito_0";
AREA_GROUP                                  "pblock_adap_circuito_0"
RANGE=SLICE_X90Y156:SLICE_x91Y159
```

- **Crear el archivo circuito.vhdl:** La función *`generarVhdl()`* se encarga de esta tarea. La función se divide en tres partes: inicializar variables y abrir el archivo, escribir todo el código VHDL adaptado al circuito a testear en cuestión y por último, cerrar el archivo y mostrar los errores si corresponde.

  La parte más importante es la escritura del archivo .vhd. Usando las variables creadas en el parseo realizado en *"cargar VHD"* tenemos toda la información necesaria para instanciar una entidad del circuito a testear y conectar todas sus señales a las del core "*circuito*". Hemos hecho esto llamando secuencialmente a una serie de funciones que por su nombre, dan una idea de que hacen cada una teniendo en mente la estructura de un archivo .vhd.

```
  public void crearFichero() {
        comentariosCabecera();
        librerias();
```

```
        entidadGeneral();
        inicioArquitectura();
        senalesCircuito();
        componenteCircuito();
        beginArquitectura();
        asignacionSenalesCircuito();
        asignacionComponenteCircuito();
        end();
    }
```

- **Modificar el archivo .pao:** Necesitamos este paso para que cuando Xilinx EDK sintetice nuestro proyecto, sepa qué VHDLs pertenecen al pcore "*circuito".*

  Se implementa en la función `modificaPao()` que se encarga de añadir una línea por cada archivo VHDL del circuito a testear.

  Estas líneas deben seguir la sintaxis:

  ```
  lib circuito_v1_00_a <nombre_vhdl> vhdl
  ```

  Donde <*nombre_vhdl*> es el nombre de los distintos archivos VHDL del circuito a testear.

- **Modificar el bitstream del sistema:** Haciendo uso de las mejoras introducidas en Nessy 5.0, dividimos la síntesis del proyecto EDK en 2 partes, lanzando 2 procesos consecutivos: la generación de los netlist, y el mapeo del software en las BRAM asociadas al microblaze.

  Existen las funciones `compilarProyecto(Semaforo sem)` y `data2mem(Semaforo sem)` que se encargan de las tareas mencionadas anteriormente.

  Primero, "*compilarProyecto"* crea un proceso nuevo lanzando el programa xps.exe y a continuación, "*data2mem"* crea otro proceso llamando al programa data2mem.exe haciendo uso de la clase *HiloProceso* y usando un objeto de la clase *Semaforo*. Ambas clases y cómo se crean estos procesos viene explicado más adelante en el apartado de "Hilos: sincronización y paralelismo".

4.3.1.3. Cargar bitstream.

Al igual que "*cargar VHD"* realiza su función de la misma manera que Nessy 2.0 mediante el programa iMPACT.

La única mejora realizada en la acción de este botón ha sido la manera de llamar al programa iMPACT. En Nessy 5.0 lo llamamos haciendo uso de las clases creadas para este fin como son *HiloProyecto, HiloNessy* y *Semaforo,* quedando como resultado la misma funcionalidad que tenía Nessy 2.0, pero permitiendo añadir al botón más funcionalidades fácilmente en una futura versión.

En la Figura 66 se ve cómo se muestra la salida de iMPACT por la pestaña *"cargar"* haciendo uso de nuestras clases creadas para lanzar procesos externos:

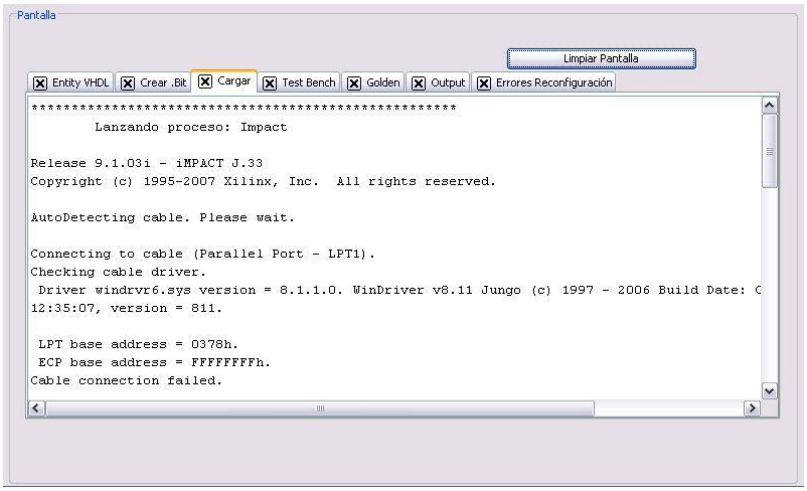

Figura 66: Cargar bitstream

4.3.1.4. Cargar testbench.

Han sido necesarios algunos cambios a la acción de este botón, no a la función en sí misma, sino a la manera de hacerla, puesto que ahora es en la FPGA donde se almacena el archivo testbench y no en el ordenador.

El código ejecutado cuando se detecta el evento de pulsar el botón es:

```
    private void _btnCargarTBActionPerformed(java.awt.event.ActionEvent
evt) {
        try {
            if (this.cargarTB()) {
                _btnCargarGolden.setEnabled(true);
                _btnGenerarGolden.setEnabled(true);
            } else {
                JOptionPane.showMessageDialog(this, "Error al cargar el
testbench", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
```

Vemos que comparte la misma estructura que los demás botones. En caso de haber retornado bien la función `cargarTB()`, habilita los botones *"cargar golden"* y *"generar golden"*. En caso contrario muestra un error por pantalla.

La función `cargarTB()`, que es quien realiza todo el trabajo, se descompone en: configurar el puerto serie (ya que aquí es donde se usa por primera vez), comprobar comunicación con la FPGA, leer el fichero testbench y enviarlo a través del puerto serie.

```
    public boolean cargarTB() throws Exception{
        if(entsal==null) configurarPuertoSerie();
        if(!comprobarComunicacion())return false;
```

```
        if(!cargaFicheroTB()) return false;
        enviarTB(fichero_tb);
        return true;
    }
```

En `enviarTB` se debe enviar primero el comando *"enviar testbench"* para preparar a la FPGA a que lo reciba.

4.3.1.5. Generar golden.

En este botón, al igual que con cargar testbench, se conserva la finalidad de la acción pero no la manera de realizarla.

Antes se enviaba por el puerto serie todo el testbench y se recibía el golden. Ahora solo se envía el comando *"generar golden"* por el puerto serie y espera a recibirlo desde la FPGA.

Lo que muestra Nessy 5.0 por la pestañas son los resultados del circuito al recibir vectores del testbench en su entrada.

4.3.1.6. Cargar golden.

Mientras que en Nessy 2.0 esta acción solo tenía que leer un archivo de texto, ahora también debe enviar el comando *"cargar golden"* a la FPGA y posteriormente enviar el mismo por el puerto serie.

4.3.1.7. Ejecutar.

La opción ejecutar sirve para correr el testbench almacenado en la memoria RAM de la FPGA sobre el circuito a testear y comparar la salida con las entradas del golden.

El código asociado al botón *"ejecutar"* de la interfaz, es el siguiente:

```
    private void _btnEjecutarActionPerformed(java.awt.event.ActionEvent
evt) {
        try {
            if (this.ejecutar()) {

            } else {
                JOptionPane.showMessageDialog(this, "Ejecucion sin
terminar", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
```

Como se puede ver, se llama a la función `ejecutar()` y en caso de haber ido algo mal, muestra por pantalla un mensaje de error. En caso de ir bien no hace falta mostrar mensaje ya que Nessy mostrará en tiempo real el estado de la ejecución por pantalla.

En la función ejecutar lo primero que se hace es comprobar la comunicación entre ordenador y FPGA para inmediatamente después enviarle por el puerto serie el comando de *"ejecutar"*. A partir de ahí, la función se limita a recibir los errores que envía la FPGA y mostrarlos por la pestaña "*Output"* de la interfaz. El código que hace esto se muestra a continuación:

```
  if(!comprobarComunicacion()) return false;

        entsal.enviarBinaria("01000000000000000000000000000000");
        numErroresString = entsal.recibirBinaria(32);
        //IMPORTANTE ERROR POR LA SALIDA DE LA FPGA
        numErrores = this.traduceString(numErroresString);
        this._TextSalida.append("Ha habido " + numErrores + " ciclos en
la ejecucion en los que hay errores: \n");
        for (i = 0; i < numErrores; i++) {
            cicloError = entsal.recibirBinaria(32);
            bitError = entsal.recibirBinaria(32);
           this._TextSalida.append("\t Errores en el ciclo " +
this.traduceString(cicloError) + ": bits " + this.erroresString(bitError)
+ "\n");


this._TextSalida.setCaretPosition(_TextSalida.getText().length());
        }
  }
```

4.3.1.8. Reconfiguración

Esta opción es la más importante, se encarga de inyectar un bitflip en cada bit de la memoria de configuración asociada al circuito a testear y comprueba las salidas generadas del circuito al pasar el testbench con los resultados obtenidos en el golden. Al usar la opción de "*reconfiguración"* de Nessy 5.0, se ejecuta el siguiente código:

```
   private void
_btnReconfiguracionActionPerformed(java.awt.event.ActionEvent evt) {
      try {
        if (this.reconfiguracion()) {

        } else {
            JOptionPane.showMessageDialog(this, "Inyeccion de errores
finalizada incorrectamente" ,"Error", JOptionPane.ERROR_MESSAGE);
        }
      } catch (Exception ex) {

java.util.logging.Logger.getLogger(GUIPrincipal.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
      }
   }
```

En este caso no se muestra nada si la función `reconfiguración()` ha terminado correctamente (ya que el usuario lo sabrá debido a los mensajes mostrados en la pestaña

"errores reconfiguración") y muestra un mensaje en caso de que haya sucedido algo inesperado. En la función `reconfiguración()` primero se hacen una serie de cálculos (explicados en el apartado *3.2.6*) que sirven para saber qué bits de la memoria de configuración de la FPGA se corresponden con los del circuito a testear.

Seguido, se crea un bitstream parcial que se corresponda a la región que el usuario ha indicado que quiere reservar para el circuito a testear. Este bitstream parcial se genera usando el programa *Virtex_II_Configuration_Cleaner.jar* desarrollado por Víctor Alaminos.

A continuación, se envía por el puerto serie, el comando correspondiente a la acción *reconfiguración* y los datos previamente calculados referentes a la memoria de configuración de la FPGA.

Por último, se crea un hilo (objeto *HiloInyeccion; La razón de esta clase es explicada más adelante en el apartado "Hilos: sincronización y paralelismo"*) que se encargará de gestionar la inyección de errores. La función run() de este hilo realiza dos acciones básicas:

```
public void run(){
    enviarBitstream();
    inyectarErrores();
}
```

Por un lado, *`enviarBitstram()`*. Esta función lee el bitstream parcial generado previamente y lo envía por el puerto serie a la FPGA. Para enviar el bitstream por el puerto serie se usa el siguiente código (se ha omitido parte de la función de `enviarBitstream()` tal como la inicialización de variables o algunas sentencias relacionadas con ficheros).

```
//Enviar el bitstream por el puerto serie
byteleido = bi.read();
while (byteleido != -1) {
    enviados++;
    cad = cad + gui.intToBinario8(byteleido);
    if (enviados % 4 == 0) {
    entsal.enviarBinaria(cad);
    bw.append(cad+"\n");
    palabrasEnviadas = "("+enviados/4+"/"+tam_bit+")";
    texto = cabecera + palabrasEnviadas + cola;
    panel.setText(texto);
    cad = "";
    }
    byteleido = bi.read();
}
// Envía la última palabra
if (!cad.equals("")) {
    entsal.enviarBinaria(cad);
}
```

Después de enviar el bitstream parcial, se ejecuta *`inyectarErrores()`*.

```
    for(int i = frame_inicio ; i < frame_fin; i++){
        //calculos
        palabra_inicio = (i * 206) + 1;
        gap_inicio = (((159 - sliceY1) / 2) * 80) / 32;
        palabra_gap_inicio = palabra_inicio + 3 + gap_inicio;
        bit_inicio = (32 - ((((159 - sliceY1) / 2) * 80) % 32)) - 1;
        gap_fin = ((((159 - sliceY0) / 2) + 1) * 80) / 32;
        palabra_gap_fin = palabra_inicio + 3 + gap_fin;
        bit_fin = 32 - (((((159 - sliceY0) / 2) + 1) * 80) % 32);
        bit_fin_aux = 0;

        for(int j = palabra_gap_inicio; j <= palabra_gap_fin; j++){
            if (j == palabra_gap_fin){
                bit_fin_aux = bit_fin;
            }
            for(int k = 0; k <= 31; k++){
                //RECONFIGURACION
                numeroConf++;

                esperar_palabra_syn();
                String cicloErrorString = entsal.recibirBinaria(32);
                cicloError = gui.traduceString(cicloErrorString);
                if (cicloError != 0){
                    numeroConfErroneas++;
                    bw.write(i+"\t"+j+"\t"+k+"\t"+cicloError+"\n");
                }
            }
            bit_inicio = 31;
        }
    }
```

Aparte de la modificación del comportamiento de los botones de manera general, también hemos realizado una serie de cambios mayores explicados en los siguientes apartados.

4.3.2. Configuración de Nessy

Debido a que el proyecto ha sido desarrollado para las plataformas Virtex II pro y Virtex V, y estas placas son soportadas por distintas versiones del software de Xilinx, era necesario eliminar la dependencia de Nessy a una versión concreta de software. Por eso, se ha creado una opción de configuración en el menú horizontal superior, que permite configurar la versión del software usado por Nessy 5.0 fácilmente.

Seleccionando la opción *"configuración Nessy"* dentro del menú superior *"configuración"*, nos aparecerá una interfaz en donde poder poner la ruta de donde están instalado los programas necesarios para el proceso de generación de los bitstreams (Ver Figura 67).

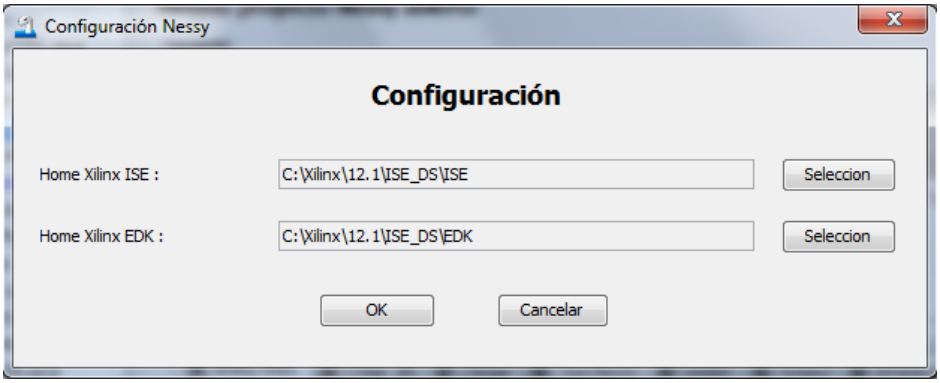

Figura 67: GUIConfig

Pulsando el botón de *"selección"* nos aparecerá una ventana con la cual podemos buscar la carpeta raíz en donde están instalados los programas de Xilinx y así Nessy sea capaz de encontrarlos.

Inicialmente, esos valores se leen de un archivo de texto plano llamado *"config.properties"* y modificando los valores de las variables HomeXilinxISE y HomeXilinxEDK se logra el mismo efecto. Esto ya estaba implementado en Nessy 2.0 junto con la opción de cargar un archivo **.propertie,.* en donde se indica la ruta de los programas, pero debido a que existían errores en el sistema de Nessy 2.0, se rehízo la manera de configurar Nessy para la versión 5.0.

4.3.3. Configuración del puerto serie

Nessy 2.0 suponía que el cable serie iba a estar conectado al puerto "*COM 1*" sin posibilidad de cambiarlo. Mediante una interfaz gráfica simple, permitimos al usuario que configure el puerto serie por el que Nessy escuchará.

A esta interfaz se puede acceder dentro del menú superior *"configuración"*, seleccionando la opción *"configuración puerto serie"* y nos aparecerá una ventana como se muestra a continuación:

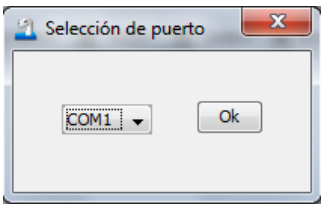

Figura 68: GUIComConfig

Esta interfaz también se mostrará en caso de que queramos hacer uso de alguna opción que requiera uso del puerto serie sin haberlo configurado previamente. Este es el caso del botón *"enviar testbench"*, que comprueba primero si se ha configurado o no, para mostrar al usuario la interfaz *"GUIComConfig"*.

Para comprobar si el puerto serie ha sido inicializado se utiliza la función `configurarPuertoSerie()`. El código de esta función es:

```
    public boolean configurarPuertoSerie(){
        boolean r = true;
        GUIComConfig GUIcom = new GUIComConfig(this, r);
        GUIcom.setLocationRelativeTo(null);
        GUIcom.setVisible(true);
        entsal = new EntradaSalida(GUIcom.getSel());
        r = entsal.inicializarPuertoSerie();
        if(!r){
            JOptionPane.showMessageDialog(this, "Error al abrir el puerto
"+GUIcom.getSel(), "Error", JOptionPane.ERROR_MESSAGE);
            entsal = null;
        }
        return r;
    }
```

Esta función crea un objeto que muestra la interfaz *"GUIComConfig"* y una vez que el usuario ha introducido el número de puerto, el objeto "*entsal*" al que se le ha pasado como parámetro a su constructora el número de puerto, llama a la función `inicializarPuertoSerie()`, la cual se encarga de toda la configuración relacionada con el puerto.

4.3.4. Gestor de proyectos: crear, abrir y cerrar proyecto

Debido a la cantidad de archivos necesarios para generar un bitstream , el tiempo necesario para realizar todos los pasos requeridos por Nessy y la necesidad de poder trabajar con distintos diseños en paralelo, hemos añadido un gestor de proyectos.

Este gestor crea una jerarquía de carpetas para cada proyecto. De esta manera los archivos relacionados a un proyecto quedan ordenados y fácilmente localizables. (Ver Figura 69)

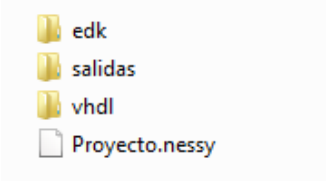

Figura 69: Jerarquía de carpetas de los proyectos

La carpeta *"edk"* contiene los archivos relacionados con el proyecto EDK que contiene todo el sistema que irá instanciado en la FPGA.

La carpeta *"vhdl"* contiene los archivos fuente .vhd del proyecto.

La carpeta *"salidas"* tiene todos los logs generados durante todo el proceso de inyección de errores.

El archivo *"proyecto.nessy"* contiene la meta información necesaria para que Nessy sepa el estado en el que se encuentra el proyecto, así como la entidad top o la región de la FPGA donde se va a instanciar el circuito a testear.

Al manejo de proyectos se accede mediante la opción Archivo en la barra horizontal superior de Nessy. Se verán las siguientes opciones:

Figura 70: Gestor de proyectos

4.3.4.1. Nuevo proyecto.

Esta opción lo que hace es preparar la estructura de carpetas y archivos que Nessy 5.0 irá usando/modificando según sea necesario. Cuando seleccionamos la opción "nuevo proyecto" nos saldrá una ventana como la Figura 71:

Figura 71: Nuevo proyecto

Aquí debemos poner la ruta donde se creará el proyecto y el nombre del proyecto del que colgará toda la jerarquía de carpetas y archivos relacionados con este.
La jerarquía creada consta de:

- Un archivo *"proyecto.nessy"* que contiene meta información del proyecto.
- Una carpeta *"edk"* donde irán todos los archivos del proyecto EDK.
- Una carpeta *"salidas"* donde se crearán todos los logs.
- Una carpeta *"vhdl"* donde se copiarán todos los .vhd una vez se haya cargado una entidad

4.3.4.2. Abrir proyecto.

Esta acción permite abrir un proyecto previamente creado (Ver Figura 72) y así ahorrarnos pasar por etapas ya realizadas, esto es, cargar de nuevo el .vhd o introducir los slices y sintetizar el diseño.

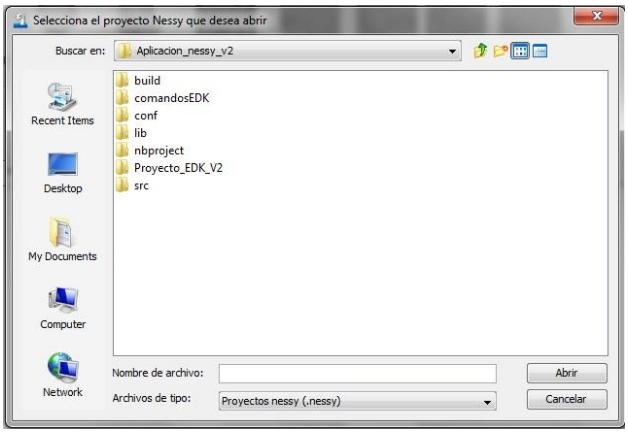

Figura 72: Abrir proyecto

Debemos seleccionar un archivo .nessy que es el que contiene meta información referente al proyecto (número de slices y .vhd top)

La función de abrir proyecto tiene una serie de etapas y condiciones:

- Se comprueba si existe algún top en el archivo de proyecto *"proyecto.nessy"*
  a. En caso de que no haya ningún top, el proceso termina dejando solo como opciones activadas *"cargar .vhd"* en la interfaz de Nessy 5.0.
  b. En caso afirmativo, se busca el archivo top en `<carpeta_raiz>/vhdl/`, se parsea y se muestra por pantalla. A continuación, se comprueba si existe el archivo *"download.bit"* en la carpeta raíz del proyecto.
     i. Si no existe, el proceso termina devolviendo el control al usuario y activando como opciones válidas, los botones de "cargar .vhd" y *"generar bitstream".*
     ii. En caso de que si exista el archivo, se activan los botones *"cargar .vhd", "generar bitstream*" y *"cargar bitstream"*, terminando el proceso.

4.3.4.3. Cerrar proyecto

Como su nombre indica, esta opción simplemente cierra el proyecto, liberando la memoria asociada a este y dejando deshabilitados todos los botones de la interfaz gráfica.

4.3.5. Hilos: sincronización, paralelismo

Puesto que las funcionalidades que tenía Nessy 2.0 no necesitaban demasiado tiempo de ejecución, apenas se notaba la congelación que sufría la interfaz cuando se estaba realizando alguna acción. Además por cada botón de la interfaz, solo se necesitaba lanzar un proceso externo.

En el caso de Nessy 5.0, hay acciones que requieren bastante tiempo, como la generación del bitstream o la inyección de errores. Además, hay acciones que requieren la llamada, en el mismo botón de la interfaz, a dos o más procesos externos, donde uno

depende del resultado generado del otro, por lo tanto deben sincronizarse de alguna manera.

Por eso hemos introducido una serie de clases que permitan ejecutar un proceso externo en paralelo al hilo de la interfaz gráfica de Nessy y así, ésta pueda responder al usuario en todo momento, mostrando la salida de dichos procesos en tiempo real sin tener que esperar a su finalización y mostrar todo de golpe. También, gracias a estas clases, es posible lanzar varios procesos externos en paralelo y que estén sincronizados en todo momento mediante un semáforo.

Las clases creadas para este propósito son:

- HiloProceso
- HiloNessy
- Semáforo
- HiloInyección

Excepto *Semaforo,* las otras tres clases heredan de la clase *Thread.* Las dos primeras sirven para ejecutar un proceso externo mediante la creación de un objeto *HiloProceso*, al cual se le puede pasar, entre otros, una pestaña como parámetro para mostrar la salida estándar del proceso en la correspondiente pestaña de Nessy 5.0.

4.3.5.1. Clase HiloProceso

Como se puede ver en el código siguiente, para lanzar un proceso en Nessy 5.0 se debe crear un objeto de la clase *HiloProceso,* asignarle nombre y lanzarlo mediante la llamada a `start()`.

```
    String comando = RUTA_EDK+"\\bin\\nt\\xps.exe -nw -scr
"+RUTA_NESSY+"\\comandosEDK\\generar_bit.tcl
                "+RUTA_PROYECTO_ACTUAL_EDK+"\\system.xmp";
    String salida = RUTA_PROYECTO_ACTUAL_SALIDAS+"\\EDK.log";HiloProceso
bitstream = new HiloProceso(this._TextCrearBit,comando,salida,sem);
    bitstream.setName("Bitstream");
    bitstream.start();
```

La constructora de *HiloProceso* tiene como cabecera:

public HiloProceso(JTextArea pestaña, String comando, String salida, Semaforo sem)

El parámetro *pestaña sirve como puntero* a un objeto *JTextArea* de la interfaz gráfica de Nessy 5.0. Este objeto es el encargado de mostrar por pantalla información relevante a cada acción de Nessy, por lo que lo usamos para mostrar en Nessy 5.0 información relacionada con el proceso en cuestión.

El parámetro *comando* es el nombre del proceso a lanzar seguido de sus parámetros, `salida` indica el nombre del archivo .log que almacenará toda la información en un fichero plano de texto. Si es nulo, no se generará archivo .log ni se mostrará por pantalla la salida estándar del proceso.

Por último, *sem* es un puntero a un objeto *Semaforo*. Esto tiene por objetivo sincronizar varios procesos mediante la compartición del mismo objeto *Semaforo.*

El código de la función `run()` de la clase *HiloProceso* viene mostrado a continuación:

```
    public void run(){
        try {
            sem.lock();

            pestaña.append("\tLanzando proceso: "+this.getName()+"\n\n");
            pestaña.setCaretPosition(pesta□span>.getText().length());
            Process proceso = Runtime.getRuntime().exec(comando);
            if(pestaña!=null){
                HiloNessy hilo = new HiloNessy(pestaña,proceso,salida);
                hilo.setName("Salida "+Thread.currentThread().getName());
                hilo.start();
            }
            proceso.waitFor();
            pestaña.append("\n\n\tProceso finalizado\n");

            pestaña.setCaretPosition(pestaña.getText().length());
            sem.unlock();
        } catch (InterruptedException ex) {
      Logger.getLogger(HiloProceso.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (IOException ex) {

Logger.getLogger(HiloProceso.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
```

Cuando se llama a la función `start()` de *HiloProceso,* ésta lo primero que hace es bloquear el semáforo (su funcionamiento se explica más adelante) para que no se puedan lanzar más procesos en paralelo con el mismo semáforo.

Después muestra a través de la interfaz gráfica de Nessy 5.0 qué proceso se está ejecutando actualmente y lanza a ejecución el proceso externo contenido en *comando.*

En el caso de que *salida* no sea un parámetro nulo, entonces se creará otro objeto de tipo *HiloNessy* que será el encargado de leer la salida estándar del proceso e ir mostrándola por la pestaña correspondiente de la interfaz de Nessy 5.0, además de escribirlo todo en un registro (archivo .log).

Por último, una vez haya finalizado el proceso, se muestra por pantalla que éste ha terminado y se desbloquea el semáforo para que otro proceso pueda ser lanzado.

4.3.5.2. Clase HiloNessy

La clase *HiloNessy,* como se ha mencionado brevemente antes, se encarga de la tarea de leer la salida estándar de un proceso e ir mostrándola por la interfaz de Nessy 5.0, además de escribirla en un registro. La constructora de esta clase tiene la cabecera:

```
public HiloNessy(JTextArea pestaña,Process proceso,String
salida)
```

Los parámetros *pestaña* y *salida* se corresponden con los explicados en *HiloProceso.* El parámetro `proceso` es un puntero al proceso creado en *HiloProceso* y se usa para leer la salida estándar de éste.

A continuación se muestra el código de la función `run()`:

```
    public void run() {
    String line;
        try {
            FileOutputStream os = new FileOutputStream(file);
            BufferedWriter salida = new BufferedWriter(new
OutputStreamWriter(os));
            BufferedReader is = new BufferedReader(new
InputStreamReader(p.getInputStream()));
            while ((line = is.readLine()) != null){
                this.text.append(line+"\n");
                this.text.setCaretPosition(text.getText().length());
                salida.write(line+"\n");
            }
            salida.close();
        }catch (IOException ex) {
            Logger.getLogger(HiloNessy.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
```

La función se dedica a leer de la salida estándar del proceso e ir escribiéndolo en el log con nombre *salida* y en el *JTextArea pestaña.*

4.3.5.3. Clase Semaforo

Semaforo se usa para sincronizar varios procesos creados mediante la clase *HiloProceso* cuando son ejecutados al pulsar un mismo botón, pasándoles un objeto *Semaforo* como parámetro. En el caso de *"generar bitstream"* se lanzan varios procesos que dependen los unos de los otros, por lo que se hace necesaria una sincronización.

```
    Semaforo sem = new Semaforo();
    compilarProyecto(sem);
    data2mem(sem);
```

En el ejemplo anterior se crea un objeto *Semaforo* y se pasa como parámetro a dos funciones que lanzarán un proceso cada una. Estos procesos estarán sincronizados, ya que al ejecutarse primero, *compilarProyecto* bloqueará el semáforo primero por lo que el proceso asociado a la función *data2mem* tendrá que esperar a que el primero finalice y así poder tomar el semáforo.

La clase *Semaforo* tiene una constructora sin parámetros:

```
    public Semaforo(){
        v = true;
    }
```

Lo único que hace es igualar el único atributo de clase *v* a true. Esto significa que el semáforo está disponible para ser tomado.

El tomar un semáforo o soltarlo se hace mediante las funciones `lock()` y `unlock()`. El código de estas funciones se muestra a continuación.

```
    public synchronized void unlock(){
        v=true;
        notify();
    }

    public synchronized void lock(){
        try {
            if (isLocked()) {
                wait();
            }
            v = false;
        } catch (InterruptedException ex) {
            Logger.getLogger(Semaforo.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
```

Cuando se quiere entrar en la región crítica, en nuestro caso con Nessy 5.0. de que un objeto *HiloProceso* lance un proceso, debe llamar a la función `lock()` y eso garantizará que ningún otro proceso que tenga que ser ejecutado al pulsar ese mismo botón de la interfaz, lo esté en paralelo.

Cuando el proceso haya terminado, el objeto *HiloProceso* asociado a éste, debe llamar a la función *`unlock()`* permitiendo que los otros objetos *HiloProceso* que estén a la espera, puedan continuar con su código y ejecutar los procesos correspondientes.

En la Figura 73 se muestra un ejemplo (*crear bitstream*) de dos procesos lanzados consecutivamente, pero *data2mem* se espera a que termine el proceso anterior.

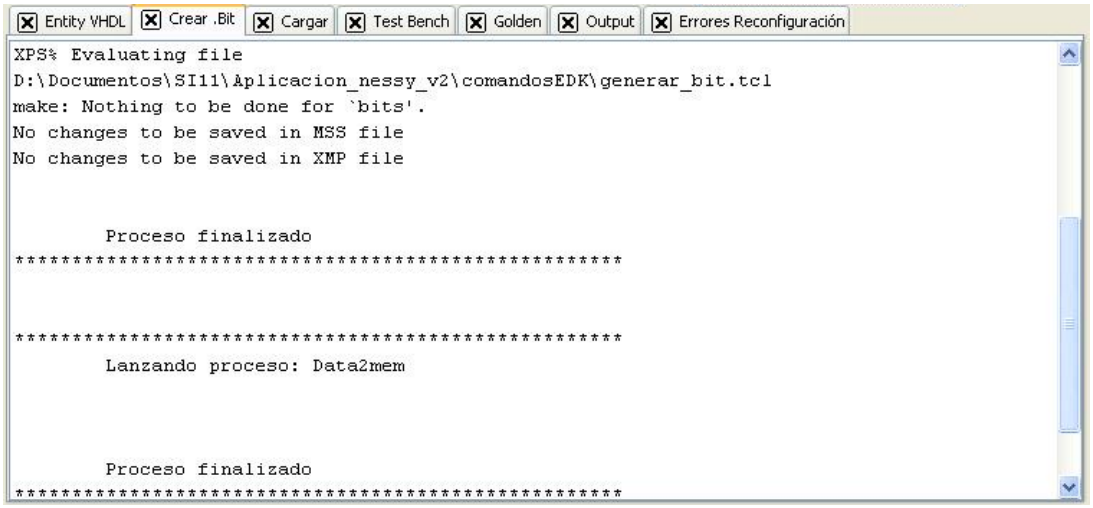

Figura 73: Procesos sincronizados

4.3.5.4. Clase HiloInyeccion

La última clase es algo particular. Se usa para crear un hilo que se encargue de lo referente a la inyección de errores, mientras la interfaz puede ir mostrando los errores recibidos desde la FPGA.

Tiene un trato especial puesto que no es un proceso externo si no que ejecuta código propio de Nessy 5.0, pero requiere demasiado tiempo como para ejecutarse en el hilo principal ya que congelaría la interfaz gráfica en algunos casos varias horas.

Se crea un hilo de esta clase en la acción asociada al botón *"reconfiguración"* de Nessy, una vez se hayan hecho los cálculos necesarios y todo esté preparado para iniciar la inyección de errores en el circuito.

Capítulo 5: Resultados para la tarjeta Virtex II pro

En una inyección de errores con Nessy 5.0, podemos obtener toda la información relevante sobre qué ha ocurrido en un circuito testado. Con los resultados que envían el microblaze de la FPGA al ordenador, podemos saber qué frame, qué palabra y qué bit han sido modificados y en qué ciclo del testbench el circuito ha fallado. Solo nos ha interesado obtener el primer ciclo del testbench en el que falla puesto que un fallo es una condición suficiente para descartar el circuito en aplicaciones reales.

Nessy 5.0 solo mostrará por la pestaña correspondiente a la inyección de errores el número de frame que se está reconfigurando, además de un resumen al final de la inyección con el tiempo necesario para la inyección, el número de reconfiguraciones realizadas, el número de reconfiguraciones erróneas y el porcentaje de errores.

Si se quiere acceder a los datos detallados comentados anteriormente, se debe ir a la carpeta *salidas del proyecto* y consultar el archivo *errores_reconfiguracion.txt.* Este archivo está diferenciado en dos partes, la primera con la información sobre frames, palabras, bits y ciclos; mostrado por columnas en la forma:

```
<frame> <palabra> <bit> <ciclo de testbench>
```

Es importante resaltar que a la hora de reservar una región de slices para un circuito a través de Nessy 5.0, se debe ajustar al máximo el tamaño reservado para el circuito, para así, minimizar los slices en los que se han inyectado errores y no han sido usados para instanciar el circuito. Ya que cuanto mejor sea el ajuste, mayor será el número de slices al que se le hayan inyectado errores que tengan que ver con la instanciación del circuito y por lo tanto, mayor será la precisión de los resultados.

Lamentablemente esto no nos ha sido posible hacerlo de una manera automática y óptima. Se ha optado por la visualización del circuito mediante la herramienta de Xilinx, FPGA editor y de esta manera, ver cuántas slices requiere el circuito en cuestión.

5.1. Contador 8 bits

Para un circuito contador de 8 bits, se le ha asignado una región en la FPGA de 2x4. El testbench usado consta de 200 ciclos (200 entradas) y consiste en la activación del "*reset*" al comienzo y la activación de la señal *"contar"* en el resto de ciclos.

```
Tiempo tardado:                          0:0:19
Número de reconfiguraciones:             4224
Número de reconfiguraciones erróneas:    231
Porcentaje de errores:                   5.46875%
```

Usando los datos más detallados e importándolos desde Matlab, podemos con un simple script, mostrar algunas relaciones tales como el número de errores por frame (Ver Figura 74) o el total del número de errores que surgen en el mismo bit de las distintas palabras de configuración (Ver Figura 75) y así intentar sacar conclusiones.

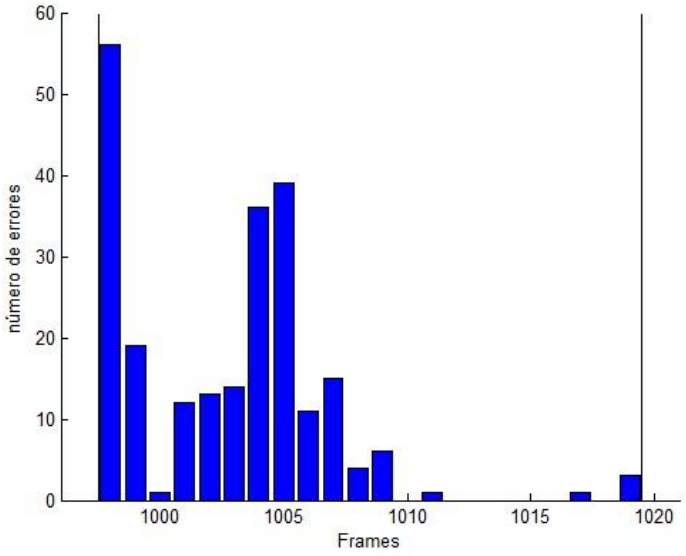

Figura 74: Número de errores por frame en el contador

Del número de errores por frame podemos ver que existe una mitad del circuito en la que casi no se ha producido ningún error. En este ejemplo, al contador se le ha asignado una región de 2x4 slices de la FPGA, esto es, un CLB y por tanto, su configuración depende de 22 frames (desde el 998 hasta la 1019) y de estas 22 frames solo en la primera mitad se han producido errores. Las líneas negras verticales indican el comienzo y final de una columna de CLB's.

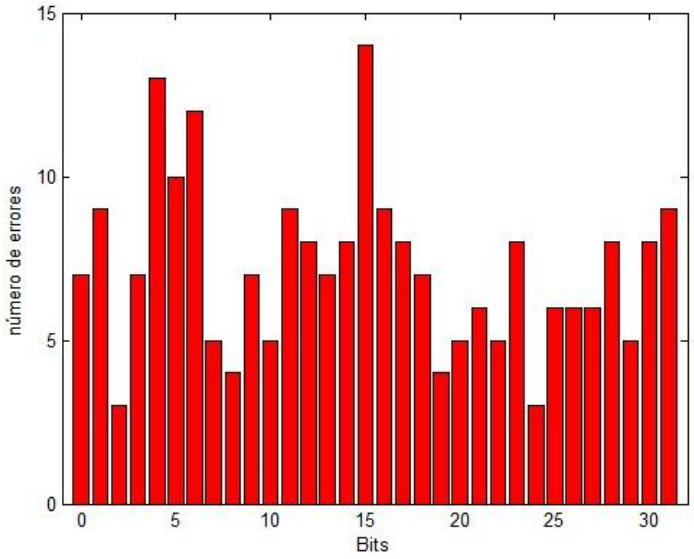

Figura 75: Número de errores por bit de palabra en el contador

5.2. FFE (Feed-Fordward Equalization)

Los resultados para un circuito FFE en una región de 10x80 slices y con un testbench de 200 ciclos son:

```
Tiempo tardado:                          0:27:49
Número de reconfiguraciones:             355520
Número de reconfiguraciones erróneas:    27908
Porcentaje de errores:
7.8499099909991%
```

Como en el caso del contador, también hemos obtenido las gráficas mostrando las relaciones comentadas anteriormente. Estas se pueden ver en las siguientes dos figuras:

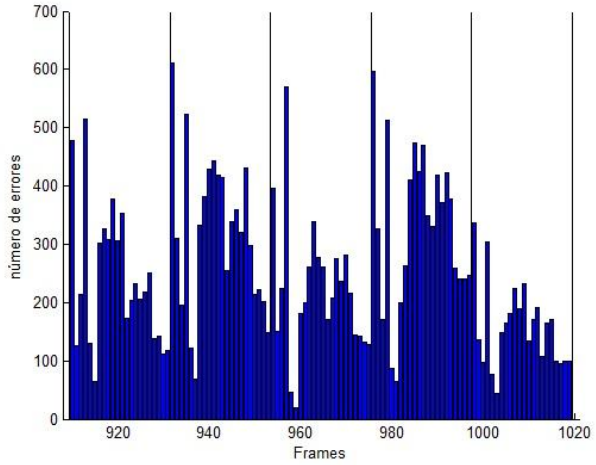

Figura 76: Número de errores por frame en el FFE

Vemos que aquí los errores siguen un patrón que se repite cada aproximadamente 22 frames. Este es el número de frames que configuran una columna de CLB's, indicado como en el caso del contador, mediante líneas verticales negras. Vemos que para cada columna de CLB's de la FPGA las inyecciones realizadas en su primera y cuarta frame son las que producen más errores. Aparte, vemos que hay una mayor densidad de errores cuando se trata de las frames del medio, siendo las frames del final y del principio (excepto por la primera y la cuarta) las menos sensibles a errores.

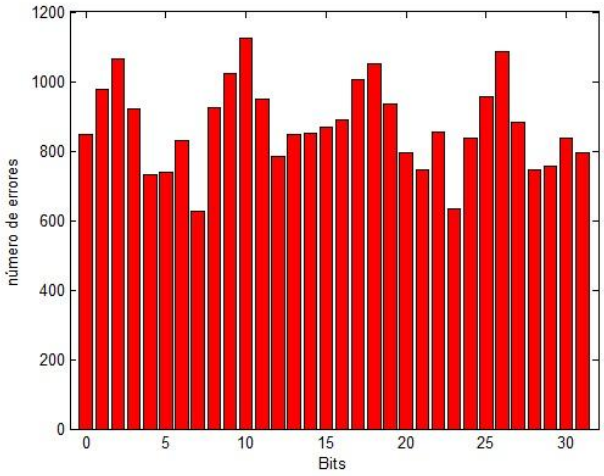

Figura 77: Número de errores por bit de palabra en el FFE

Como en el caso del contador, no podemos decir nada de los errores según la posición del bit en la palabra de configuración puesto que los errores están distribuidos uniformemente.

5.3. Comparación con Nessy 2.0

Como se ha visto, con Nessy 5.0 el tiempo necesario para una inyección en circuitos de tamaño tan dispares como es un contador de 8 bits y el FFE, es del orden de magnitud de minutos.

En Nessy 2.0, una inyección de errores tardaba de media **28 segundos** por bit modificado. En cambio, con Nessy 5.0 y tomando como base los datos de la inyección del de errores del contador y el FFE, vemos que tarda de media **4,59 milisegundos** bit modificado.

Esto supone que el tiempo necesario para la inyección de errores con Nessy 5.0 es de **6100 veces menor**. Es decir, una mejoría de 3 órdenes de magnitud. Una inyección antes necesitaba semanas o meses, ahora necesita minutos.

Capítulo 6: Modificaciones para la tarjeta Virtex V

Hay que aclarar que esta versión no está funcional 100% ya que no disponemos de la función partial la cual crea los bitstream parciales modificados y depende del estudiante de Máster en Investigación en Informática, Víctor Alaminos.
Así, queda pendiente integrar dicha función en el programa que correrá en el microblaze con la interfaz que provee el pcore icap para su manejo en la versión 12.1.

6.1. Modificaciones Hardware

Al haber usado Xilinx EDK el esfuerzo a la hora de migrar el sistema a otra FPGA como en este caso, a la Virtex V, ha sido mínimo debido a la modularidad del EDK.

El nuevo sistema pensado para la Virtex V mantiene la misma estructura y los únicos cambios se realizan en las versiones de los dispositivos o en el peor de los casos, se sustituyen por otros módulos de idéntica funcionalidad debido a que Xilinx ha dejado de dar soportes a los usados para la versión de la Virtex II pro. Este es el caso del bus OPB, el cual Xilinx no soporta en la versión 12.1.

Comentar que gracias al uso de una versión posterior del core icap, nos ha sido posible usar un reloj del sistema de hasta 100Mhz. En la versión Nessy 5.0 para Virtex II pro, la velocidad máxima del reloj estaba limitada a 50Mhz debido a la versión del icap para esa FPGA. Esto supone una gran mejora en velocidad sin apenas hacer ningún esfuerzo por nuestra parte, puesto que el problema estaba en el diseño del core en sí y no dependía de nosotros. Claro está, siempre seguirá existiendo el cuello de botella impuesto por la velocidad de transmisión del puerto serie.

6.1.1. Bus PLB

Para la versión 12.1 del software de Xilinx, no hay soporte para el bus OPB y este es sustituido por el processor local bus (PLB). Así que hemos tenido que sustituir nuestro bus OPB del sistema usado en la Virtex II pro por un PLB v4.6, que sí es soportado por Xilinx EDK.

El PLB de Xilinx consiste en un árbitro central, la lógica necesaria de control y todas las estructuras OR/MUX necesarias para el bus. El Xilinx PLB v4.6 core provee toda la estructura del PLB y permite por tanto, configurar el número de esclavos y/o maestros que tendrá.

En el ejemplo siguiente, se muestran las conexiones de un PLB para un sistema con 3 maestros y 3 esclavos.

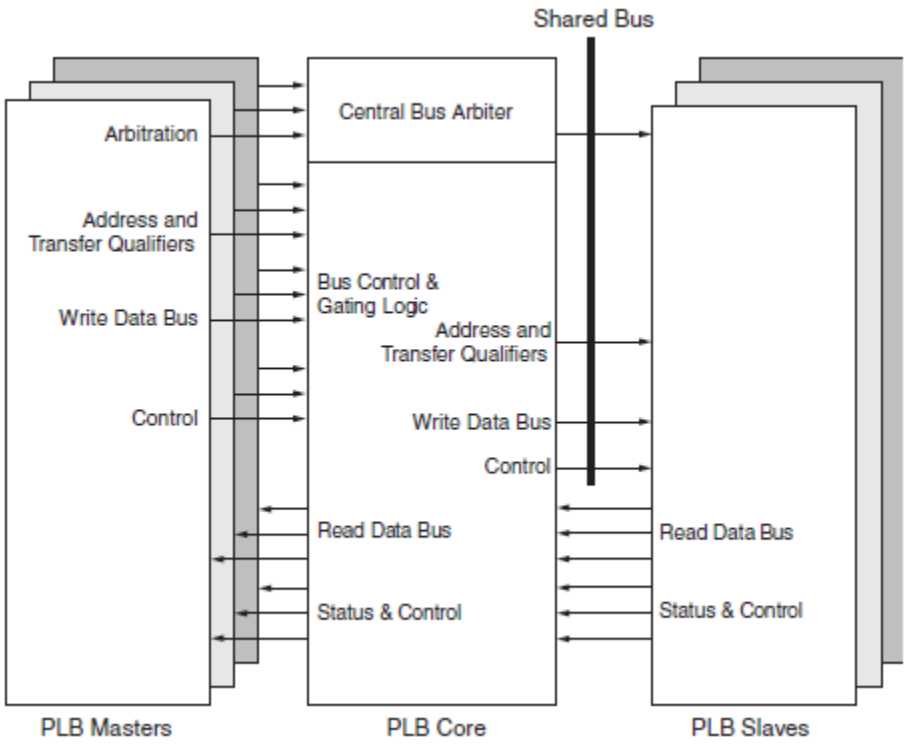

Figura 78: Ejemplo PLB

No hay mucho más que comentar desde el momento en el que se usa Xilinx EDK para su integración en el sistema debido a la similitud con el bus OPB.

6.1.2. Componentes propios

Al haberse sustituido el bus del sistema por un bus PLB, hemos tenido que rehacer nuestros componentes propios con interfaz a bus (adaptador_uart y adaptador_circuito) para adaptarlos a los cambios.

Esto se hace con el wizard de pcores de Xilinx EDK indicándole que la interfaz debe ser con un bus PLB y una vez creado el esqueleto, adaptar el código de los adaptadores desarrollados por nosotros a las particularidades de esta nueva interfaz.

6.2. Modificaciones software

6.2.1. Reconfiguración

Esta parte nos ha sido imposible terminarla debido a que no tenemos de la función necesaria para crear los bitstream parciales modificados, esto es, el programa de Víctor Alaminos. Por lo tanto, se deja pendiente esta parte del proyecto por si en un futuro tiene continuidad por parte de otros alumnos.

6.3. Modificaciones en la Aplicación

Gracias a la abstracción a la hora de desarrollar Nessy 5.0, tenemos una aplicación que sirve para cualquier placa de Xilinx.

La aplicación no necesita de ninguna modificación “obligatoria” con el fin de que funcione en otra placa salvo cambiar las variables del fichero config.properties en el que se debe de poner la ruta del software de Xilinx (y esto al fin y al cabo, no es ninguna modificación a nivel de desarrollo si no que es de parte de usuario).

Dicho esto, comentamos la mejora realizada desde el punto de vista de la optimización y no la funcionalidad y a qué partes de la aplicación afecta.

Esta optimización ha sido el uso de Xilinx PlanAhead para crear el bitstream parcial de nuestro circuito en vez de usar directamente la opción AREA_GROUP en el fichero de restricciones .ucf y el programa Virtex_II_Configuration_Cleaner.jar como se hacía anteriormente.

Este cambio viene motivado por la forma de crear los bitstream parciales del PlanAhead y de la opción AREA_GROUP y es que con Xilinx PlanAhead se garantiza que en la región designada solo estará instanciado el circuito a testar, mientras que haciendo uso del AREA_GROUP y Xilinx EDK, no se garantiza que en el enrrutado se meta alguna señal ajena al circuito a testar (por lo que puede causar problemas en la inyección de errores).

6.3.1. Funcionalidad de los botones de la interfaz

6.3.1.1. Crear bitstream.

Ahora la opción “crear bitstream” de Nessy 5.0. también crea el bitstream parcial en vez de solo declarar la región donde se quería que fuera el circuito como antes. Esto se hace llamando al proceso de Xilinx PlanAhead con la configuración necesaria, esto es, pasar como parámetro un script. El resto permanece como antes.

6.3.1.2. Reconfiguración

Ahora ya no hacemos uso del programa Virtex_II_Configuration_Cleaner.jar para crear el bitstream parcial puesto que este ya ha sido creado previamente en “crear bitstream” por el PlanAhead. Por lo demás, es igual.

6.3.2. Configuración de Nessy

Esta modificación respecto a la versión de Nessy 5.0 para la Virtex II pro se debe a que en las versiones del software de Xilinx que soportan esa placa (9.1i) no estaba el PlanAhead. Con la introducción de este debemos tener en cuenta que Nessy 5.0 para Virtex V debe conocer la ruta de tanto Xilinx ISE, Xilinx EDK y Xilinx PlanAhead, por lo que añadimos una opción más tanto al archivo config.properties (mostrado a continuación) como a la GUI de configuración de Nessy (Figura 79).

```
HomeXilinxEDK=C\:\\Xilinx\\12.1\\ISE_DS\\EDK
HomeXilinxPLANAHEAD=C\:\\Xilinx\\12.1\\ISE_DS\\PlanAhead
HomeXilinxISE=C\:\\Xilinx\\12.1\\ISE_DS\\ISE
```

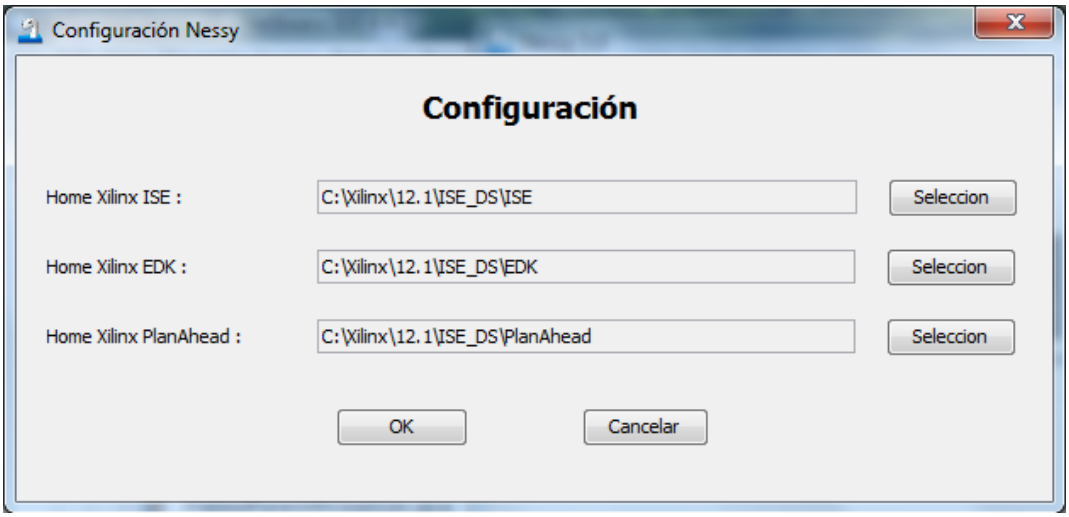

Figura 79: GUIconfig para Virtex V

6.3.3. Gestor de proyectos: crear, abrir y cerrar proyecto

Tenemos que la estructura de los proyectos también sufre una modificación menor debido a lo mismo que la configuración de Nessy 5.0, por la introducción de PlanAhead.
Ahora para cada proyecto creado, se tiene una nueva carpeta llamada planahead. Esta carpeta contiene obviamente todos los archivos creados por este programa durante la síntesis y creación de bits parciales del diseño.

La jerarquía de carpetas para Nessy 5.0 para la Virtex V es por tanto:

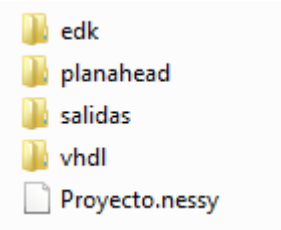

Figura 80: Jerarquía de carpetas para la Virtex V

Capítulo 7: Apéndices

A. Ficheros hardware

A.1. System.xmp

```
XmpVersion: 9.1.02
VerMgmt: 9.1.02
IntStyle: default
MHS File: system.mhs
MSS File: system.mss
NPL File: projnav/system.ise
Architecture: virtex2p
Device: xc2vp30
Package: ff896
SpeedGrade: -7
UseProjNav: 0
PNImportBitFile:
PNImportBmmFile:
UserCmd1:
UserCmd1Type: 0
UserCmd2:
UserCmd2Type: 0
TopInst: system_i
GenSimTB: 0
InsertNoPads: 0
WarnForEAArch: 1
HdlLang: VHDL
Simulator: mti
SimModel: BEHAVIORAL
MixLangSim: 1
UcfFile: data/system.ucf
FpgaImpMode: 0
ShowLicenseDialog: 1
Processor: microblaze_0
BootLoop: 1
XmdStub: 0
```

A.2. System.mhs

```
##############################################################################
########
# Created by Base System Builder Wizard for Xilinx EDK 9.1.02 Build
EDK_J_SP2.4
# Tue Apr 19 19:04:21 2011
# Target Board:  Xilinx XUP Virtex-II Pro Development System Rev
# Family:    virtex2p
# Device:    xc2vp30
# Package:   ff896
# Speed Grade:   -7
# Processor: Microblaze
# System clock frequency: 50.000000 MHz
# Debug interface: No Debug
# On Chip Memory :  64 KB
# Total Off Chip Memory : 512 MB
# - DDR_SDRAM_64Mx64 Dual Rank = 256 MB
# - DDR_512MB_64Mx64_rank2_row13_col10_cl2_5 = 256 MB
#
##############################################################################
########


 PARAMETER VERSION = 2.1.0


 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk, DIR = O, VEC
= [0:2]
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn, DIR = O, VEC
= [0:2]
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr, DIR = O, VEC
= [0:12]
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_BankAddr_pin
= fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_BankAddr, DIR =
O, VEC = [0:1]
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CASn_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CASn, DIR = O
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_RASn_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_RASn, DIR = O
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_WEn_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_WEn, DIR = O
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM, DIR = O, VEC =
[0:7]
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS, DIR = IO, VEC
= [0:7]
```

```
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ, DIR = IO, VEC
= [0:63]
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CKE_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CKE, DIR = O, VEC
= [0:1]
 PORT fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CSn_pin =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CSn, DIR = O, VEC
= [0:1]
 PORT fpga_0_net_gnd_pin = net_gnd, DIR = O
 PORT fpga_0_net_gnd_1_pin = net_gnd, DIR = O
 PORT fpga_0_net_gnd_2_pin = net_gnd, DIR = O
 PORT fpga_0_net_gnd_3_pin = net_gnd, DIR = O
 PORT fpga_0_net_gnd_4_pin = net_gnd, DIR = O
 PORT fpga_0_net_gnd_5_pin = net_gnd, DIR = O
 PORT fpga_0_net_gnd_6_pin = net_gnd, DIR = O
 PORT fpga_0_DDR_CLK_FB = ddr_feedback_s, DIR = I, SIGIS = CLK,
CLK_FREQ = 100000000
 PORT fpga_0_DDR_CLK_FB_OUT = ddr_clk_feedback_out_s, DIR = O
 PORT sys_clk_pin = dcm_clk_s, DIR = I, SIGIS = CLK, CLK_FREQ =
100000000
 PORT sys_rst_pin = sys_rst_s, DIR = I, RST_POLARITY = 0, SIGIS = RST
 PORT circuito_0_salida_circuito_pin = circuito_0_salida_circuito, DIR
= O, VEC = [0:31]
 PORT adaptador_uart_0_entrada_serie_pin =
adaptador_uart_0_entrada_serie, DIR = I
 PORT adaptador_uart_0_salida_serie_pin =
adaptador_uart_0_salida_serie, DIR = O


BEGIN microblaze
 PARAMETER INSTANCE = microblaze_0
 PARAMETER HW_VER = 6.00.b
 PARAMETER C_USE_FPU = 0
 BUS_INTERFACE DLMB = dlmb
 BUS_INTERFACE ILMB = ilmb
 BUS_INTERFACE DOPB = mb_opb
 BUS_INTERFACE IOPB = mb_opb
END

BEGIN lmb_v10
 PARAMETER INSTANCE = ilmb
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_EXT_RESET_HIGH = 0
 PORT SYS_Rst = sys_rst_s
 PORT LMB_Clk = sys_clk_s
END

BEGIN lmb_v10
 PARAMETER INSTANCE = dlmb
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_EXT_RESET_HIGH = 0
 PORT SYS_Rst = sys_rst_s
 PORT LMB_Clk = sys_clk_s
END

BEGIN lmb_bram_if_cntlr
```

```
 PARAMETER INSTANCE = dlmb_cntlr
 PARAMETER HW_VER = 2.00.a
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x0000ffff
 BUS_INTERFACE SLMB = dlmb
 BUS_INTERFACE BRAM_PORT = dlmb_port
END

BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = ilmb_cntlr
 PARAMETER HW_VER = 2.00.a
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x0000ffff
 BUS_INTERFACE SLMB = ilmb
 BUS_INTERFACE BRAM_PORT = ilmb_port
END

BEGIN bram_block
 PARAMETER INSTANCE = lmb_bram
 PARAMETER HW_VER = 1.00.a
 BUS_INTERFACE PORTA = ilmb_port
 BUS_INTERFACE PORTB = dlmb_port
END

BEGIN opb_v20
 PARAMETER INSTANCE = mb_opb
 PARAMETER HW_VER = 1.10.c
 PARAMETER C_EXT_RESET_HIGH = 0
 PORT SYS_Rst = sys_rst_s
 PORT OPB_Clk = sys_clk_s
END

BEGIN mch_opb_ddr
 PARAMETER INSTANCE = DDR_512MB_64Mx64_rank2_row13_col10_cl2_5
 PARAMETER HW_VER = 1.00.c
 PARAMETER C_MCH_OPB_CLK_PERIOD_PS = 20000
 PARAMETER C_NUM_BANKS_MEM = 2
 PARAMETER C_NUM_CLK_PAIRS = 4
 PARAMETER C_REG_DIMM = 0
 PARAMETER C_DDR_TMRD = 20000
 PARAMETER C_DDR_TWR = 20000
 PARAMETER C_DDR_TRAS = 60000
 PARAMETER C_DDR_TRC = 90000
 PARAMETER C_DDR_TRFC = 100000
 PARAMETER C_DDR_TRCD = 30000
 PARAMETER C_DDR_TRRD = 20000
 PARAMETER C_DDR_TRP = 30000
 PARAMETER C_DDR_AWIDTH = 13
 PARAMETER C_DDR_COL_AWIDTH = 10
 PARAMETER C_DDR_BANK_AWIDTH = 2
 PARAMETER C_DDR_DWIDTH = 64
 PARAMETER C_DDR_ASYNC_SUPPORT = 1
 PARAMETER C_MEM0_BASEADDR = 0xc0000000
 PARAMETER C_MEM0_HIGHADDR = 0xcfffffff
 PARAMETER C_MEM1_BASEADDR = 0xd0000000
 PARAMETER C_MEM1_HIGHADDR = 0xdfffffff
 BUS_INTERFACE SOPB = mb_opb
```

```
 PORT DDR_Addr =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr
 PORT DDR_BankAddr =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_BankAddr
 PORT DDR_CASn =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CASn
 PORT DDR_CKE =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CKE
 PORT DDR_CSn =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CSn
 PORT DDR_RASn =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_RASn
 PORT DDR_WEn =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_WEn
 PORT DDR_DM = fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM
 PORT DDR_DQS =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS
 PORT DDR_DQ = fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ
 PORT DDR_Clk =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk &
ddr_clk_feedback_out_s
 PORT DDR_Clkn =
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn & 0b0
 PORT Device_Clk90_in = ddr_dev_clk_90_s
 PORT Device_Clk90_in_n = ddr_dev_clk_90_s_n
 PORT Device_Clk = ddr_dev_clk_s
 PORT Device_Clk_n = ddr_dev_clk_s_n
 PORT DDR_Clk90_in = ddr_clk_90_s
 PORT DDR_Clk90_in_n = ddr_clk_90_n_s
END

BEGIN util_vector_logic
 PARAMETER INSTANCE = devclk_inv
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_SIZE = 1
 PARAMETER C_OPERATION = not
 PORT Op1 = ddr_dev_clk_s
 PORT Res = ddr_dev_clk_s_n
END

BEGIN util_vector_logic
 PARAMETER INSTANCE = devclk90_inv
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_SIZE = 1
 PARAMETER C_OPERATION = not
 PORT Op1 = ddr_dev_clk_90_s
 PORT Res = ddr_dev_clk_90_s_n
END

BEGIN util_vector_logic
 PARAMETER INSTANCE = ddr_clk90_inv
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_SIZE = 1
 PARAMETER C_OPERATION = not
 PORT Op1 = ddr_clk_90_s
 PORT Res = ddr_clk_90_n_s
END
```

```
BEGIN dcm_module
 PARAMETER INSTANCE = dcm_0
 PARAMETER HW_VER = 1.00.c
 PARAMETER C_CLK0_BUF = TRUE
 PARAMETER C_CLK90_BUF = TRUE
 PARAMETER C_CLKDV_BUF = TRUE
 PARAMETER C_CLKDV_DIVIDE = 2.000000
 PARAMETER C_CLKIN_PERIOD = 10.000000
 PARAMETER C_CLK_FEEDBACK = 1X
 PARAMETER C_DLL_FREQUENCY_MODE = LOW
 PARAMETER C_EXT_RESET_HIGH = 1
 PORT CLKIN = dcm_clk_s
 PORT CLK0 = ddr_dev_clk_s
 PORT CLK90 = ddr_dev_clk_90_s
 PORT CLKDV = sys_clk_s
 PORT CLKFB = ddr_dev_clk_s
 PORT RST = net_gnd
 PORT LOCKED = dcm_0_lock
END

BEGIN dcm_module
 PARAMETER INSTANCE = dcm_1
 PARAMETER HW_VER = 1.00.c
 PARAMETER C_CLK0_BUF = TRUE
 PARAMETER C_CLK90_BUF = TRUE
 PARAMETER C_CLKIN_PERIOD = 10.000000
 PARAMETER C_CLK_FEEDBACK = 1X
 PARAMETER C_DLL_FREQUENCY_MODE = LOW
 PARAMETER C_PHASE_SHIFT = 60
 PARAMETER C_CLKOUT_PHASE_SHIFT = FIXED
 PARAMETER C_EXT_RESET_HIGH = 0
 PORT CLKIN = ddr_feedback_s
 PORT CLK90 = ddr_clk_90_s
 PORT CLK0 = dcm_1_FB
 PORT CLKFB = dcm_1_FB
 PORT RST = dcm_0_lock
 PORT LOCKED = dcm_1_lock
END

BEGIN opb_hwicap
 PARAMETER INSTANCE = opb_hwicap_0
 PARAMETER HW_VER = 1.00.b
 PARAMETER C_BASEADDR = 0x40200000
 PARAMETER C_HIGHADDR = 0x4020ffff
 BUS_INTERFACE SOPB = mb_opb
END

BEGIN adaptador_circuito
 PARAMETER INSTANCE = adaptador_circuito_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x76000000
 PARAMETER C_HIGHADDR = 0x760001FF
 BUS_INTERFACE SOPB = mb_opb
 PORT dato_listo = circuito_0_dato_listo
 PORT entrada_circuito = circuito_0_entrada_circuito
 PORT salida_circuito = circuito_0_salida_circuito
```

```
END

BEGIN adaptador_uart
 PARAMETER INSTANCE = adaptador_uart_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x79400000
 PARAMETER C_HIGHADDR = 0x794001FF
 BUS_INTERFACE SOPB = mb_opb
 PORT entrada_serie = adaptador_uart_0_entrada_serie
 PORT salida_serie = adaptador_uart_0_salida_serie
END

BEGIN circuito
 PARAMETER INSTANCE = circuito_0
 PARAMETER HW_VER = 1.00.a
 PORT dato_listo = circuito_0_dato_listo
 PORT entrada_circuito = circuito_0_entrada_circuito
 PORT salida_circuito = circuito_0_salida_circuito
END
```

A.3. System.ucf

```
## RELOJ Y RESET

Net sys_clk_pin LOC=AJ15;
Net sys_clk_pin IOSTANDARD = LVCMOS25;
Net sys_rst_pin LOC=AH5;
Net sys_rst_pin IOSTANDARD = LVTTL;
Net fpga_0_DDR_CLK_FB LOC=C16;
Net fpga_0_DDR_CLK_FB IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_CLK_FB_OUT LOC=G23;
Net fpga_0_DDR_CLK_FB_OUT IOSTANDARD = SSTL2_II;

## System level constraints
Net sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 20000 ps;
Net sys_rst_pin TIG;
Net fpga_0_DDR_CLK_FB TNM_NET = fpga_0_DDR_CLK_FB;
TIMESPEC TS_fpga_0_DDR_CLK_FB = PERIOD fpga_0_DDR_CLK_FB 7499 ps;


## UART

Net adaptador_uart_0_entrada_serie_pin LOC=AJ8;
Net adaptador_uart_0_entrada_serie_pin IOSTANDARD = LVCMOS25;
Net adaptador_uart_0_salida_serie_pin LOC=AE7;
Net adaptador_uart_0_salida_serie_pin IOSTANDARD = LVCMOS25;
Net adaptador_uart_0_salida_serie_pin SLEW = SLOW;
Net adaptador_uart_0_salida_serie_pin DRIVE = 12;

## DDRR

Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<12>
LOC=M25;
```

```
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<12>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<11>
LOC=N25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<11>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<10>
LOC=L26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<10>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<9>
LOC=M29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<9>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<8>
LOC=K30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<8>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<7>
LOC=G25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<7>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<6>
LOC=G26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<6>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<5>
LOC=D26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<5>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<4>
LOC=J24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<4>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<3>
LOC=K24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<3>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<2>
LOC=F28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<2>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<1>
LOC=F30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<1>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<0>
LOC=M24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Addr_pin<0>
IOSTANDARD = SSTL2_II;
Net
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_BankAddr_pin<1>
LOC=M26;
Net
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_BankAddr_pin<1>
IOSTANDARD = SSTL2_II;
```

```
Net
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_BankAddr_pin<0>
LOC=K26;
Net
fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_BankAddr_pin<0>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CASn_pin
LOC=L27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CASn_pin
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CKE_pin<1>
LOC=R26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CKE_pin<1>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CKE_pin<0>
LOC=R25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CKE_pin<0>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CSn_pin<1>
LOC=R24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CSn_pin<1>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CSn_pin<0>
LOC=R23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_CSn_pin<0>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_RASn_pin
LOC=N29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_RASn_pin
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_WEn_pin
LOC=N26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_WEn_pin
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<7>
LOC=U26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<7>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<6>
LOC=V29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<6>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<5>
LOC=W29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<5>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<4>
LOC=T22;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<4>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<3>
LOC=W28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<3>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<2>
LOC=W27;
```

```
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<2>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<1>
LOC=W26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<1>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<0>
LOC=W25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DM_pin<0>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<7>
LOC=E30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<7>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<6>
LOC=J29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<6>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<5>
LOC=M30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<5>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<4>
LOC=P29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<4>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<3>
LOC=V23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<3>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<2>
LOC=AA25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<2>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<1>
LOC=AC25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<1>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<0>
LOC=AH26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQS_pin<0>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<63>
LOC=C27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<63>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<62>
LOC=D28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<62>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<61>
LOC=D29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<61>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<60>
LOC=D30;
```

```
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<60>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<59>
LOC=H25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<59>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<58>
LOC=H26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<58>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<57>
LOC=E27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<57>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<56>
LOC=E28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<56>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<55>
LOC=J26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<55>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<54>
LOC=G27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<54>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<53>
LOC=G28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<53>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<52>
LOC=G30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<52>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<51>
LOC=L23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<51>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<50>
LOC=L24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<50>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<49>
LOC=H27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<49>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<48>
LOC=H28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<48>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<47>
LOC=J27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<47>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<46>
LOC=J28;
```

```
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<46>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<45>
LOC=K29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<45>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<44>
LOC=L29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<44>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<43>
LOC=N23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<43>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<42>
LOC=N24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<42>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<41>
LOC=K27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<41>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<40>
LOC=K28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<40>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<39>
LOC=R22;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<39>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<38>
LOC=M27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<38>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<37>
LOC=M28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<37>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<36>
LOC=P30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<36>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<35>
LOC=P23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<35>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<34>
LOC=P24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<34>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<33>
LOC=N27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<33>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<32>
LOC=N28;
```

```
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<32>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<31>
LOC=V27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<31>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<30>
LOC=Y30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<30>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<29>
LOC=U24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<29>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<28>
LOC=U23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<28>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<27>
LOC=V26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<27>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<26>
LOC=V25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<26>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<25>
LOC=Y29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<25>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<24>
LOC=AA29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<24>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<23>
LOC=Y26;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<23>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<22>
LOC=AA28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<22>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<21>
LOC=AA27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<21>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<20>
LOC=W24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<20>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<19>
LOC=W23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<19>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<18>
LOC=AB28;
```

```
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<18>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<17>
LOC=AB27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<17>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<16>
LOC=AC29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<16>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<15>
LOC=AB25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<15>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<14>
LOC=AE29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<14>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<13>
LOC=AA24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<13>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<12>
LOC=AA23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<12>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<11>
LOC=AD28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<11>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<10>
LOC=AD27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<10>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<9>
LOC=AF30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<9>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<8>
LOC=AF29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<8>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<7>
LOC=AF25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<7>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<6>
LOC=AG30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<6>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<5>
LOC=AG29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<5>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<4>
LOC=AD26;
```

```
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<4>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<3>
LOC=AD25;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<3>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<2>
LOC=AG28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<2>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<1>
LOC=AH27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<1>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<0>
LOC=AH29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_DQ_pin<0>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk_pin<2>
LOC=AC27;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk_pin<2>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk_pin<1>
LOC=AD29;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk_pin<1>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk_pin<0>
LOC=AB23;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clk_pin<0>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn_pin<2>
LOC=AC28;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn_pin<2>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn_pin<1>
LOC=AD30;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn_pin<1>
IOSTANDARD = SSTL2_II;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn_pin<0>
LOC=AB24;
Net fpga_0_DDR_512MB_64Mx64_rank2_row13_col10_cl2_5_DDR_Clkn_pin<0>
IOSTANDARD = SSTL2_II;

Net fpga_0_net_gnd_pin LOC=G12;
Net fpga_0_net_gnd_pin IOSTANDARD = LVTTL;
Net fpga_0_net_gnd_pin SLEW = SLOW;
Net fpga_0_net_gnd_pin DRIVE = 6;
Net fpga_0_net_gnd_1_pin LOC=D15;
Net fpga_0_net_gnd_1_pin IOSTANDARD = LVTTL;
Net fpga_0_net_gnd_1_pin SLEW = SLOW;
Net fpga_0_net_gnd_1_pin DRIVE = 6;
Net fpga_0_net_gnd_2_pin LOC=E15;
Net fpga_0_net_gnd_2_pin IOSTANDARD = LVTTL;
Net fpga_0_net_gnd_2_pin SLEW = SLOW;
Net fpga_0_net_gnd_2_pin DRIVE = 6;
Net fpga_0_net_gnd_3_pin LOC=G10;
Net fpga_0_net_gnd_3_pin IOSTANDARD = LVTTL;
```

```
Net fpga_0_net_gnd_3_pin SLEW = SLOW;
Net fpga_0_net_gnd_3_pin DRIVE = 6;
Net fpga_0_net_gnd_4_pin LOC=E10;
Net fpga_0_net_gnd_4_pin IOSTANDARD = LVTTL;
Net fpga_0_net_gnd_4_pin SLEW = SLOW;
Net fpga_0_net_gnd_4_pin DRIVE = 6;
Net fpga_0_net_gnd_5_pin LOC=G8;
Net fpga_0_net_gnd_5_pin IOSTANDARD = LVTTL;
Net fpga_0_net_gnd_5_pin SLEW = SLOW;
Net fpga_0_net_gnd_5_pin DRIVE = 6;
Net fpga_0_net_gnd_6_pin LOC=H9;
Net fpga_0_net_gnd_6_pin IOSTANDARD = LVTTL;
Net fpga_0_net_gnd_6_pin SLEW = SLOW;
Net fpga_0_net_gnd_6_pin DRIVE = 6;

## RESTRICCIONES DE LUGAR

INST "microblaze_0" AREA_GROUP = "pblock_microblaze_0";
AREA_GROUP "pblock_microblaze_0" RANGE = SLICE_X20Y20:SLICE_x59Y89;
AREA_GROUP "pblock_microblaze_0" RANGE =
MULT18X18_X2Y2:MULT18X18_X3Y4;

INST "lmb_bram" AREA_GROUP = "pblock_bram";
AREA_GROUP "pblock_bram" RANGE = RAMB16_X2Y1:RAMB16_X6Y7 ;

INST "ddr_512mb_64mx64_rank2_row13_col10_cl2_5" AREA_GROUP =
"pblock_ddr";
AREA_GROUP "pblock_ddr" RANGE = SLICE_X0Y4:SLICE_x9Y149 ;

INST "opb_hwicap_0" AREA_GROUP = "pblock_icap";
AREA_GROUP "pblock_icap" RANGE = SLICE_X86Y0:SLICE_x91Y29;
AREA_GROUP "pblock_icap" RANGE = RAMB16_X7Y0:RAMB16_X7Y1;

INST "adaptador_uart_0" AREA_GROUP = "pblock_uart_0";
AREA_GROUP "pblock_uart_0" RANGE = SLICE_X60Y0:SLICE_x75Y35 ;

INST "adaptador_circuito_0" AREA_GROUP = "pblock_adap_circuito_0";
AREA_GROUP "pblock_adap_circuito_0" RANGE = SLICE_X60Y60:SLICE_x69Y99
;
```

A.4. Adaptador circuito

A.4.1. Fichero .pao

```
##############################################################################
########
## Filename:
D:/Ruben/SI11/Aplicacion_nessy_v2/Proyecto_EDK_V2/pcores/adaptador_cir
cuito_v1_00_a/data/adaptador_circuito_v2_1_0.pao
## Description:       Peripheral Analysis Order
## Date:              Fri Feb 18 11:32:47 2011 (by Create and Import
Peripheral Wizard)
##############################################################################
########
```

```
lib proc_common_v2_00_a proc_common_pkg vhdl
lib proc_common_v2_00_a family vhdl
lib proc_common_v2_00_a or_muxcy vhdl
lib proc_common_v2_00_a or_gate vhdl
lib proc_common_v2_00_a counter_bit vhdl
lib proc_common_v2_00_a counter vhdl
lib proc_common_v2_00_a inferred_lut4 vhdl
lib proc_common_v2_00_a srl_fifo2 vhdl
lib proc_common_v2_00_a pf_counter_bit vhdl
lib proc_common_v2_00_a pf_counter vhdl
lib proc_common_v2_00_a pf_counter_top vhdl
lib proc_common_v2_00_a pf_occ_counter vhdl
lib proc_common_v2_00_a pf_occ_counter_top vhdl
lib proc_common_v2_00_a pf_adder_bit vhdl
lib proc_common_v2_00_a pf_adder vhdl
lib proc_common_v2_00_a pf_dpram_select vhdl
lib proc_common_v2_00_a srl16_fifo vhdl
lib proc_common_v2_00_a pselect vhdl
lib proc_common_v2_00_a valid_be vhdl
lib proc_common_v2_00_a ld_arith_reg vhdl
lib proc_common_v2_00_a mux_onehot vhdl
lib proc_common_v2_00_a down_counter vhdl
lib proc_common_v2_00_a ipif_pkg vhdl
lib proc_common_v2_00_a ipif_steer vhdl
lib proc_common_v2_00_a direct_path_cntr_ai vhdl
lib interrupt_control_v1_00_a interrupt_control vhdl
lib wrpfifo_v1_01_b pf_dly1_mux vhdl
lib wrpfifo_v1_01_b ipif_control_wr vhdl
lib wrpfifo_v1_01_b wrpfifo_dp_cntl vhdl
lib wrpfifo_v1_01_b wrpfifo_top vhdl
lib rdpfifo_v1_01_b ipif_control_rd vhdl
lib rdpfifo_v1_01_b rdpfifo_dp_cntl vhdl
lib rdpfifo_v1_01_b rdpfifo_top vhdl
lib opb_ipif_v3_01_c reset_mir vhdl
lib opb_ipif_v3_01_c brst_addr_cntr vhdl
lib opb_ipif_v3_01_c opb_flex_addr_cntr vhdl
lib opb_ipif_v3_01_c brst_addr_cntr_reg vhdl
lib opb_ipif_v3_01_c opb_be_gen vhdl
lib opb_ipif_v3_01_c srl_fifo3 vhdl
lib opb_ipif_v3_01_c write_buffer vhdl
lib opb_ipif_v3_01_c opb_bam vhdl
lib opb_ipif_v3_01_c opb_ipif vhdl
lib adaptador_circuito_v1_00_a user_logic vhdl
lib adaptador_circuito_v1_00_a adaptador_circuito vhdl
```

A.4.2. Fichero .mpd

```
##############################################################################
##
## Name     : adaptador_circuito
## Desc     : Microprocessor Peripheral Description
##          : Automatically generated by PsfUtility
##
##############################################################################
```

```
BEGIN adaptador_circuito

## Peripheral Options
OPTION IPTYPE = PERIPHERAL
OPTION IMP_NETLIST = TRUE
OPTION HDL = VHDL
OPTION IP_GROUP = MICROBLAZE:PPC:USER
OPTION DESC = ADAPTADOR_CIRCUITO


## Bus Interfaces
BUS_INTERFACE BUS = SOPB, BUS_TYPE = SLAVE, BUS_STD = OPB

## Generics for VHDL or Parameters for Verilog
PARAMETER C_BASEADDR = 0xffffffff, DT = std_logic_vector, MIN_SIZE =
0x200, BUS = SOPB, ADDRESS = BASE, PAIR = C_HIGHADDR
PARAMETER C_HIGHADDR = 0x00000000, DT = std_logic_vector, BUS = SOPB,
ADDRESS = HIGH, PAIR = C_BASEADDR
PARAMETER C_OPB_AWIDTH = 32, DT = INTEGER, BUS = SOPB, ASSIGNMENT =
CONSTANT
PARAMETER C_OPB_DWIDTH = 32, DT = INTEGER, BUS = SOPB, RANGE = (8, 16,
32)
PARAMETER C_USER_ID_CODE = 3, DT = INTEGER, RANGE = (0:255)
PARAMETER C_FAMILY = virtex2p, DT = STRING

## Ports
PORT dato_listo = "", DIR = O
PORT entrada_circuito = "", DIR = O, VEC = [0:31]
PORT salida_circuito = "", DIR = I, VEC = [0:31]

PORT OPB_Clk = "", DIR = I, SIGIS = Clk, BUS = SOPB
PORT OPB_Rst = OPB_Rst, DIR = I, SIGIS = Rst, BUS = SOPB
PORT Sl_DBus = Sl_DBus, DIR = O, VEC = [0:(C_OPB_DWIDTH-1)], BUS =
SOPB
PORT Sl_errAck = Sl_errAck, DIR = O, BUS = SOPB
PORT Sl_retry = Sl_retry, DIR = O, BUS = SOPB
PORT Sl_toutSup = Sl_toutSup, DIR = O, BUS = SOPB
PORT Sl_xferAck = Sl_xferAck, DIR = O, BUS = SOPB
PORT OPB_ABus = OPB_ABus, DIR = I, VEC = [0:(C_OPB_AWIDTH-1)], BUS =
SOPB
PORT OPB_BE = OPB_BE, DIR = I, VEC = [0:((C_OPB_DWIDTH/8)-1)], BUS =
SOPB
PORT OPB_DBus = OPB_DBus, DIR = I, VEC = [0:(C_OPB_DWIDTH-1)], BUS =
SOPB
PORT OPB_RNW = OPB_RNW, DIR = I, BUS = SOPB
PORT OPB_select = OPB_select, DIR = I, BUS = SOPB
PORT OPB_seqAddr = OPB_seqAddr, DIR = I, BUS = SOPB

END
```

A.4.3. Adaptador_circuito.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

```
library proc_common_v2_00_a;
use proc_common_v2_00_a.proc_common_pkg.all;
use proc_common_v2_00_a.ipif_pkg.all;
library opb_ipif_v3_01_c;
use opb_ipif_v3_01_c.all;

library adaptador_circuito_v1_00_a;
use adaptador_circuito_v1_00_a.all;


entity adaptador_circuito is
  generic
  (
    C_BASEADDR                     : std_logic_vector     :=
X"00000000";
    C_HIGHADDR                     : std_logic_vector     :=
X"0000FFFF";
    C_OPB_AWIDTH                   : integer              := 32;
    C_OPB_DWIDTH                   : integer              := 32;
    C_USER_ID_CODE                 : integer              := 3;
    C_FAMILY                       : string               :=
"virtex2p"
    -- DO NOT EDIT ABOVE THIS LINE --------------------
  );
  port
  (
    dato_listo                     : out std_logic;
    salida_circuito                : in std_logic_vector(0 to 31);
    entrada_circuito               : out std_logic_vector(0 to 31);

    OPB_Clk                        : in  std_logic;
    OPB_Rst                        : in  std_logic;
    Sl_DBus                        : out std_logic_vector(0 to
C_OPB_DWIDTH-1);
    Sl_errAck                      : out std_logic;
    Sl_retry                       : out std_logic;
    Sl_toutSup                     : out std_logic;
    Sl_xferAck                     : out std_logic;
    OPB_ABus                       : in  std_logic_vector(0 to
C_OPB_AWIDTH-1);
    OPB_BE                         : in  std_logic_vector(0 to
C_OPB_DWIDTH/8-1);
    OPB_DBus                       : in  std_logic_vector(0 to
C_OPB_DWIDTH-1);
    OPB_RNW                        : in  std_logic;
    OPB_select                     : in  std_logic;
    OPB_seqAddr                    : in  std_logic
    -- DO NOT EDIT ABOVE THIS LINE --------------------
  );

  attribute SIGIS : string;
  attribute SIGIS of OPB_Clk       : signal is "Clk";
  attribute SIGIS of OPB_Rst       : signal is "Rst";

end entity adaptador_circuito;
```

```
------------------------------------------------------------------------
--------
-- Architecture section
------------------------------------------------------------------------
--------

architecture IMP of adaptador_circuito is

  ---------------------------------------
  -- Constant: array of address range identifiers
  ---------------------------------------
  constant ARD_ID_ARRAY                    : INTEGER_ARRAY_TYPE   :=
    (
      0  => USER_00,                          -- user logic S/W
register address space
      1  => IPIF_RST                          -- include IPIF S/W
Reset/MIR service
    );

  ---------------------------------------
  -- Constant: array of address pairs for each address range
  ---------------------------------------
  constant ZERO_ADDR_PAD                   : std_logic_vector(0 to 64-
C_OPB_AWIDTH-1) := (others => '0');

  constant USER_BASEADDR                   : std_logic_vector     :=
C_BASEADDR or X"00000000";
  constant USER_HIGHADDR                   : std_logic_vector     :=
C_BASEADDR or X"000000FF";

  constant RST_BASEADDR                    : std_logic_vector     :=
C_BASEADDR or X"00000100";
  constant RST_HIGHADDR                    : std_logic_vector     :=
C_BASEADDR or X"000001FF";

  constant ARD_ADDR_RANGE_ARRAY            : SLV64_ARRAY_TYPE     :=
    (
      ZERO_ADDR_PAD & USER_BASEADDR,             -- user logic base
address
      ZERO_ADDR_PAD & USER_HIGHADDR,             -- user logic high
address
      ZERO_ADDR_PAD & RST_BASEADDR,              -- MIR/Reset
register base address
      ZERO_ADDR_PAD & RST_HIGHADDR               -- MIR/Reset
register high address
    );

  ---------------------------------------
  -- Constant: array of data widths for each target address range
  ---------------------------------------
  constant USER_DWIDTH                     : integer              :=
32;

  constant ARD_DWIDTH_ARRAY                : INTEGER_ARRAY_TYPE   :=
    (
      0  => USER_DWIDTH,                      -- user logic data width
```

```
      1  => C_OPB_DWIDTH                        -- MIR/Reset register
data width
    );

  ------------------------------------------
  -- Constant: array of desired number of chip enables for each
address range
  ------------------------------------------
  constant USER_NUM_CE                    : integer              := 2;

  constant ARD_NUM_CE_ARRAY               : INTEGER_ARRAY_TYPE   :=
    (
      0  => pad_power2(USER_NUM_CE),          -- user logic number of
CEs
      1  => 1                                 -- MIR/Reset register -
1 CE
    );

  ------------------------------------------
  -- Constant: array of unique properties for each address range
  ------------------------------------------
  constant ARD_DEPENDENT_PROPS_ARRAY      : DEPENDENT_PROPS_ARRAY_TYPE
:=
    (
      0  => (others => 0),                    -- user logic slave
space dependent properties (none defined)
      1  => (others => 0)                     -- IPIF reset/mir
dependent properties (none defined)
    );

  ------------------------------------------
  -- Constant: pipeline mode
  -- 1 = include OPB-In pipeline registers
  -- 2 = include IP pipeline registers
  -- 3 = include OPB-In and IP pipeline registers
  -- 4 = include OPB-Out pipeline registers
  -- 5 = include OPB-In and OPB-Out pipeline registers
  -- 6 = include IP and OPB-Out pipeline registers
  -- 7 = include OPB-In, IP, and OPB-Out pipeline registers
  -- Note:
  -- only mode 4, 5, 7 are supported for this release
  ------------------------------------------
  constant PIPELINE_MODEL                 : integer              := 5;

  ------------------------------------------
  -- Constant: user core ID code
  ------------------------------------------
  constant DEV_BLK_ID                     : integer              :=
C_USER_ID_CODE;

  ------------------------------------------
  -- Constant: enable MIR/Reset register
  ------------------------------------------
  constant DEV_MIR_ENABLE                 : integer              := 1;

  ------------------------------------------
  -- Constant: array of IP interrupt mode
```

```
  -- 1 = Active-high interrupt condition
  -- 2 = Active-low interrupt condition
  -- 3 = Active-high pulse interrupt event
  -- 4 = Active-low pulse interrupt event
  -- 5 = Positive-edge interrupt event
  -- 6 = Negative-edge interrupt event
  ------------------------------------------
  constant IP_INTR_MODE_ARRAY             : INTEGER_ARRAY_TYPE    :=
    (
      0  => 0  -- not used
    );

  ------------------------------------------
  -- Constant: enable device burst
  ------------------------------------------
  constant DEV_BURST_ENABLE               : integer               := 0;

  ------------------------------------------
  -- Constant: include address counter for burst transfers
  ------------------------------------------
  constant INCLUDE_ADDR_CNTR              : integer               := 0;

  ------------------------------------------
  -- Constant: include write buffer that decouples OPB and IPIC write
transactions
  ------------------------------------------
  constant INCLUDE_WR_BUF                 : integer               := 0;

  ------------------------------------------
  -- Constant: index for CS/CE
  ------------------------------------------
  constant USER00_CS_INDEX                : integer               :=
get_id_index(ARD_ID_ARRAY, USER_00);

  constant USER00_CE_INDEX                : integer               :=
calc_start_ce_index(ARD_NUM_CE_ARRAY, USER00_CS_INDEX);

  ------------------------------------------
  -- IP Interconnect (IPIC) signal declarations -- do not delete
  -- prefix 'i' stands for IPIF while prefix 'u' stands for user logic
  -- typically user logic will be hooked up to IPIF directly via
i<sig>
  -- unless signal slicing and muxing are needed via u<sig>
  ------------------------------------------
  signal iBus2IP_RdCE                   : std_logic_vector(0 to
calc_num_ce(ARD_NUM_CE_ARRAY)-1);
  signal iBus2IP_WrCE                   : std_logic_vector(0 to
calc_num_ce(ARD_NUM_CE_ARRAY)-1);
  signal iBus2IP_Data                   : std_logic_vector(0 to
C_OPB_DWIDTH-1);
  signal iBus2IP_BE                     : std_logic_vector(0 to
C_OPB_DWIDTH/8-1);
  signal iIP2Bus_Data                   : std_logic_vector(0 to
C_OPB_DWIDTH-1)   := (others => '0');
  signal iIP2Bus_Ack                    : std_logic    := '0';
  signal iIP2Bus_Error                  : std_logic    := '0';
  signal iIP2Bus_Retry                  : std_logic    := '0';
```

```
  signal iIP2Bus_ToutSup                : std_logic   := '0';
  signal DISABLE_POSTED_WRITE           : std_logic_vector(0 to
ARD_ID_ARRAY'length-1)   := (others => '1'); -- disable posted write
behavior for acknowledged write behavior
  signal ZERO_IP2RFIFO_Data             : std_logic_vector(0 to
ARD_DWIDTH_ARRAY(get_id_index_iboe(ARD_ID_ARRAY, IPIF_RDFIFO_DATA))-1)
:= (others => '0'); -- work around for XST not taking (others => '0')
in port mapping
  signal ZERO_WFIFO2IP_Data             : std_logic_vector(0 to
ARD_DWIDTH_ARRAY(get_id_index_iboe(ARD_ID_ARRAY, IPIF_WRFIFO_DATA))-1)
:= (others => '0'); -- work around for XST not taking (others => '0')
in port mapping
  signal ZERO_IP2Bus_IntrEvent          : std_logic_vector(0 to
IP_INTR_MODE_ARRAY'length-1)   := (others => '0'); -- work around for
XST not taking (others => '0') in port mapping
  signal iBus2IP_Clk                    : std_logic;
  signal iBus2IP_Reset                  : std_logic;
  signal uBus2IP_Data                   : std_logic_vector(0 to
USER_DWIDTH-1);
  signal uBus2IP_BE                     : std_logic_vector(0 to
USER_DWIDTH/8-1);
  signal uBus2IP_RdCE                   : std_logic_vector(0 to
USER_NUM_CE-1);
  signal uBus2IP_WrCE                   : std_logic_vector(0 to
USER_NUM_CE-1);
  signal uIP2Bus_Data                   : std_logic_vector(0 to
USER_DWIDTH-1);

begin

  ------------------------------------------
  -- instantiate the OPB IPIF
  ------------------------------------------
  OPB_IPIF_I : entity opb_ipif_v3_01_c.opb_ipif
    generic map
    (
      C_ARD_ID_ARRAY                => ARD_ID_ARRAY,
      C_ARD_ADDR_RANGE_ARRAY        => ARD_ADDR_RANGE_ARRAY,
      C_ARD_DWIDTH_ARRAY            => ARD_DWIDTH_ARRAY,
      C_ARD_NUM_CE_ARRAY            => ARD_NUM_CE_ARRAY,
      C_ARD_DEPENDENT_PROPS_ARRAY   => ARD_DEPENDENT_PROPS_ARRAY,
      C_PIPELINE_MODEL              => PIPELINE_MODEL,
      C_DEV_BLK_ID                  => DEV_BLK_ID,
      C_DEV_MIR_ENABLE              => DEV_MIR_ENABLE,
      C_OPB_AWIDTH                  => C_OPB_AWIDTH,
      C_OPB_DWIDTH                  => C_OPB_DWIDTH,
      C_FAMILY                      => C_FAMILY,
      C_IP_INTR_MODE_ARRAY          => IP_INTR_MODE_ARRAY,
      C_DEV_BURST_ENABLE            => DEV_BURST_ENABLE,
      C_INCLUDE_ADDR_CNTR           => INCLUDE_ADDR_CNTR,
      C_INCLUDE_WR_BUF              => INCLUDE_WR_BUF
    )
    port map
    (
      OPB_select                    => OPB_select,
      OPB_DBus                      => OPB_DBus,
      OPB_ABus                      => OPB_ABus,
```

```
    OPB_BE                          => OPB_BE,
    OPB_RNW                         => OPB_RNW,
    OPB_seqAddr                     => OPB_seqAddr,
    Sln_DBus                        => Sl_DBus,
    Sln_xferAck                     => Sl_xferAck,
    Sln_errAck                      => Sl_errAck,
    Sln_retry                       => Sl_retry,
    Sln_toutSup                     => Sl_toutSup,
    Bus2IP_CS                       => open,
    Bus2IP_CE                       => open,
    Bus2IP_RdCE                     => iBus2IP_RdCE,
    Bus2IP_WrCE                     => iBus2IP_WrCE,
    Bus2IP_Data                     => iBus2IP_Data,
    Bus2IP_Addr                     => open,
    Bus2IP_AddrValid                => open,
    Bus2IP_BE                       => iBus2IP_BE,
    Bus2IP_RNW                      => open,
    Bus2IP_Burst                    => open,
    IP2Bus_Data                     => iIP2Bus_Data,
    IP2Bus_Ack                      => iIP2Bus_Ack,
    IP2Bus_AddrAck                  => '0',
    IP2Bus_Error                    => iIP2Bus_Error,
    IP2Bus_Retry                    => iIP2Bus_Retry,
    IP2Bus_ToutSup                  => iIP2Bus_ToutSup,
    IP2Bus_PostedWrInh              => DISABLE_POSTED_WRITE,
    IP2RFIFO_Data                   => ZERO_IP2RFIFO_Data,
    IP2RFIFO_WrMark                 => '0',
    IP2RFIFO_WrRelease              => '0',
    IP2RFIFO_WrReq                  => '0',
    IP2RFIFO_WrRestore              => '0',
    RFIFO2IP_AlmostFull             => open,
    RFIFO2IP_Full                   => open,
    RFIFO2IP_Vacancy                => open,
    RFIFO2IP_WrAck                  => open,
    IP2WFIFO_RdMark                 => '0',
    IP2WFIFO_RdRelease              => '0',
    IP2WFIFO_RdReq                  => '0',
    IP2WFIFO_RdRestore              => '0',
    WFIFO2IP_AlmostEmpty            => open,
    WFIFO2IP_Data                   => ZERO_WFIFO2IP_Data,
    WFIFO2IP_Empty                  => open,
    WFIFO2IP_Occupancy              => open,
    WFIFO2IP_RdAck                  => open,
    IP2Bus_IntrEvent                => ZERO_IP2Bus_IntrEvent,
    IP2INTC_Irpt                    => open,
    Freeze                          => '0',
    Bus2IP_Freeze                   => open,
    OPB_Clk                         => OPB_Clk,
    Bus2IP_Clk                      => iBus2IP_Clk,
    IP2Bus_Clk                      => '0',
    Reset                           => OPB_Rst,
    Bus2IP_Reset                    => iBus2IP_Reset
  );

  ------------------------------------------
  -- instantiate the User Logic
  ------------------------------------------
```

```
  USER_LOGIC_I : entity adaptador_circuito_v1_00_a.user_logic
    generic map
    (
      -- MAP USER GENERICS BELOW THIS LINE ---------------
      --USER generics mapped here
      -- MAP USER GENERICS ABOVE THIS LINE ---------------

      C_DWIDTH                       => USER_DWIDTH,
      C_NUM_CE                       => USER_NUM_CE
    )
    port map
    (
      -- MAP USER PORTS BELOW THIS LINE ------------------
      --USER ports mapped here
      -- MAP USER PORTS ABOVE THIS LINE ------------------

      dato_listo                     => dato_listo,
      salida_circuito                => salida_circuito,
      entrada_circuito               => entrada_circuito,

      Bus2IP_Clk                     => iBus2IP_Clk,
      Bus2IP_Reset                   => iBus2IP_Reset,
      Bus2IP_Data                    => uBus2IP_Data,
      Bus2IP_BE                      => uBus2IP_BE,
      Bus2IP_RdCE                    => uBus2IP_RdCE,
      Bus2IP_WrCE                    => uBus2IP_WrCE,
      IP2Bus_Data                    => uIP2Bus_Data,
      IP2Bus_Ack                     => iIP2Bus_Ack,
      IP2Bus_Retry                   => iIP2Bus_Retry,
      IP2Bus_Error                   => iIP2Bus_Error,
      IP2Bus_ToutSup                 => iIP2Bus_ToutSup
    );

  ----------------------------------------
  -- hooking up signal slicing
  ----------------------------------------
  uBus2IP_BE <= iBus2IP_BE(0 to USER_DWIDTH/8-1);
  uBus2IP_Data <= iBus2IP_Data(0 to USER_DWIDTH-1);
  uBus2IP_RdCE <= iBus2IP_RdCE(USER00_CE_INDEX to
USER00_CE_INDEX+USER_NUM_CE-1);
  uBus2IP_WrCE <= iBus2IP_WrCE(USER00_CE_INDEX to
USER00_CE_INDEX+USER_NUM_CE-1);
  iIP2Bus_Data(0 to USER_DWIDTH-1) <= uIP2Bus_Data;

end IMP;
```

A.4.4. User_logic.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

library proc_common_v2_00_a;
use proc_common_v2_00_a.proc_common_pkg.all;
```

```
entity user_logic is
  generic
  (

    C_DWIDTH                        : integer              := 32;
    C_NUM_CE                        : integer              := 2
  );
  port
  (
    dato_listo                      : out std_logic;
    salida_circuito                 : in std_logic_vector(0 to 31);
    entrada_circuito                : out std_logic_vector(0 to 31);

    Bus2IP_Clk                      : in  std_logic;
    Bus2IP_Reset                    : in  std_logic;
    Bus2IP_Data                     : in  std_logic_vector(0 to
C_DWIDTH-1);
    Bus2IP_BE                       : in  std_logic_vector(0 to
C_DWIDTH/8-1);
    Bus2IP_RdCE                     : in  std_logic_vector(0 to
C_NUM_CE-1);
    Bus2IP_WrCE                     : in  std_logic_vector(0 to
C_NUM_CE-1);
    IP2Bus_Data                     : out std_logic_vector(0 to
C_DWIDTH-1);
    IP2Bus_Ack                      : out std_logic;
    IP2Bus_Retry                    : out std_logic;
    IP2Bus_Error                    : out std_logic;
    IP2Bus_ToutSup                  : out std_logic
  );
end entity user_logic;

------------------------------------------------------------------------
--------
-- Architecture section
------------------------------------------------------------------------
--------

architecture IMP of user_logic is

  signal slv_reg0                       : std_logic_vector(0 to
C_DWIDTH-1);
  signal slv_reg1                       : std_logic_vector(0 to
C_DWIDTH-1);
  signal slv_reg_write_select           : std_logic_vector(0 to 1);
  signal slv_reg_read_select            : std_logic_vector(0 to 1);
  signal slv_ip2bus_data                : std_logic_vector(0 to
C_DWIDTH-1);
  signal slv_read_ack                   : std_logic;
  signal slv_write_ack                  : std_logic;

  ----------------- SEcLES DEL CIRCUITO ----------------------
  type estado is (E0,E1,E2,E3);
  signal est, sig_est           : estado := E0;
  signal entrada_lista          : std_logic;
  signal senal_clk              : std_logic;
  signal senal_entrada_lista      : std_logic_vector(0 to 3);
```

```
  ---------------------------------------------------------------

begin

  slv_reg_write_select <= Bus2IP_WrCE(0 to 1);
  slv_reg_read_select  <= Bus2IP_RdCE(0 to 1);
  slv_write_ack        <= Bus2IP_WrCE(0) or Bus2IP_WrCE(1);
  slv_read_ack         <= Bus2IP_RdCE(0) or Bus2IP_RdCE(1);
  IP2Bus_Data          <= slv_ip2bus_data;
  IP2Bus_Ack           <= slv_write_ack or slv_read_ack;
  IP2Bus_Error         <= '0';
  IP2Bus_Retry         <= '0';
  IP2Bus_ToutSup       <= '0';

  -- implement slave model register(s)
  SLAVE_REG_WRITE_PROC : process( Bus2IP_Clk ) is
  begin

    if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
      if Bus2IP_Reset = '1' then
        slv_reg0 <= (others => '0');
        entrada_lista <= '0';
        senal_entrada_lista <= "0000";
      else
        case slv_reg_write_select is
          when "10" =>
            for byte_index in 0 to (C_DWIDTH/8)-1 loop
              if ( Bus2IP_BE(byte_index) = '1' ) then
                slv_reg0(byte_index*8 to byte_index*8+7) <=
Bus2IP_Data(byte_index*8 to byte_index*8+7);
                senal_entrada_lista(byte_index) <= '1';
              end if;
            end loop;
          when "01" =>
          when others => null;
        end case;
      end if;
      if senal_entrada_lista = "1111" then
        entrada_lista <= '1';
        senal_entrada_lista <= "0000";
      end if;
      if entrada_lista = '1' then
        entrada_lista <= '0';
      end if;
    end if;

  end process SLAVE_REG_WRITE_PROC;

  -- implement slave model register read mux
  SLAVE_REG_READ_PROC : process( slv_reg_read_select, slv_reg0,
slv_reg1 ) is
  begin

    case slv_reg_read_select is
      when "10" => slv_ip2bus_data <= slv_reg0;
      when "01" => slv_ip2bus_data <= slv_reg1;
      when others => slv_ip2bus_data <= (others => '0');
```

```
    end case;

  end process SLAVE_REG_READ_PROC;


---------------- MAQUINA DE ESTADOS ----------------------
    SINCRONO: process(Bus2IP_Clk,Bus2IP_Reset)
    begin
        if Bus2IP_Reset = '1' then
            est <= E0;
        elsif Bus2IP_Clk'event and Bus2IP_Clk = '1' then
            est <= sig_est;
        end if;
    end process SINCRONO;


    COMB: process(entrada_lista,senal_clk)
    begin
        sig_est <= est;
        case est is
            when E0 =>
                senal_clk <= '0';
                if entrada_lista = '1' then
                    sig_est <= E1;
                end if;
            when E1 =>
                senal_clk <= '0';
                sig_est <= E2;
            when E2 =>
                senal_clk <= '0';
                sig_est <= E3;
            when E3 =>
                senal_clk <= '1';
                sig_est <= E0;
        end case;
    end process COMB;


    process(Bus2IP_Clk)
    begin
        if Bus2IP_Clk'event and Bus2IP_Clk ='1' then
            slv_reg1 <= salida_circuito;
        end if;
    end process;

    dato_listo <= senal_clk;
    entrada_circuito <= slv_reg0;


end IMP;
```

A.5. Adaptador uart

A.5.1. Fichero .pao

```
##############################################################################
########
## Filename:
D:/Ruben/SI11/pruebasProyectosNessy/contador_v2/edk/pcores/adaptador_u
art_v1_00_a/data/adaptador_uart_v2_1_0.pao
## Description:       Peripheral Analysis Order
## Date:              Thu Feb 24 12:52:04 2011 (by Create and Import
Peripheral Wizard)
##############################################################################
########

lib proc_common_v2_00_a proc_common_pkg vhdl
lib proc_common_v2_00_a family vhdl
lib proc_common_v2_00_a or_muxcy vhdl
lib proc_common_v2_00_a or_gate vhdl
lib proc_common_v2_00_a counter_bit vhdl
lib proc_common_v2_00_a counter vhdl
lib proc_common_v2_00_a inferred_lut4 vhdl
lib proc_common_v2_00_a srl_fifo2 vhdl
lib proc_common_v2_00_a pf_counter_bit vhdl
lib proc_common_v2_00_a pf_counter vhdl
lib proc_common_v2_00_a pf_counter_top vhdl
lib proc_common_v2_00_a pf_occ_counter vhdl
lib proc_common_v2_00_a pf_occ_counter_top vhdl
lib proc_common_v2_00_a pf_adder_bit vhdl
lib proc_common_v2_00_a pf_adder vhdl
lib proc_common_v2_00_a pf_dpram_select vhdl
lib proc_common_v2_00_a srl16_fifo vhdl
lib proc_common_v2_00_a pselect vhdl
lib proc_common_v2_00_a valid_be vhdl
lib proc_common_v2_00_a ld_arith_reg vhdl
lib proc_common_v2_00_a mux_onehot vhdl
lib proc_common_v2_00_a down_counter vhdl
lib proc_common_v2_00_a ipif_pkg vhdl
lib proc_common_v2_00_a ipif_steer vhdl
lib proc_common_v2_00_a direct_path_cntr_ai vhdl
lib interrupt_control_v1_00_a interrupt_control vhdl
lib wrpfifo_v1_01_b pf_dly1_mux vhdl
lib wrpfifo_v1_01_b ipif_control_wr vhdl
lib wrpfifo_v1_01_b wrpfifo_dp_cntl vhdl
lib wrpfifo_v1_01_b wrpfifo_top vhdl
lib rdpfifo_v1_01_b ipif_control_rd vhdl
lib rdpfifo_v1_01_b rdpfifo_dp_cntl vhdl
lib rdpfifo_v1_01_b rdpfifo_top vhdl
lib opb_ipif_v3_01_c reset_mir vhdl
lib opb_ipif_v3_01_c brst_addr_cntr vhdl
lib opb_ipif_v3_01_c opb_flex_addr_cntr vhdl
lib opb_ipif_v3_01_c brst_addr_cntr_reg vhdl
lib opb_ipif_v3_01_c opb_be_gen vhdl
lib opb_ipif_v3_01_c srl_fifo3 vhdl
lib opb_ipif_v3_01_c write_buffer vhdl
lib opb_ipif_v3_01_c opb_bam vhdl
lib opb_ipif_v3_01_c opb_ipif vhdl
```

```
lib adaptador_uart_v1_00_a user_logic vhdl
lib adaptador_uart_v1_00_a adaptador_uart vhdl
lib adaptador_uart_v1_00_a Entrada_salida vhdl
lib adaptador_uart_v1_00_a Tx_serie vhdl
lib adaptador_uart_v1_00_a Rx_serie vhdl
```

A.5.2. Fichero .mpd

```
##############################################################################
##
## Name     : adaptador_uart
## Desc     : Microprocessor Peripheral Description
##          : Automatically generated by PsfUtility
##
##############################################################################

BEGIN adaptador_uart

## Peripheral Options
OPTION IPTYPE = PERIPHERAL
OPTION IMP_NETLIST = TRUE
OPTION HDL = VHDL
OPTION IP_GROUP = MICROBLAZE:PPC:USER
OPTION DESC = ADAPTADOR_UART


## Bus Interfaces
BUS_INTERFACE BUS = SOPB, BUS_TYPE = SLAVE, BUS_STD = OPB

## Generics for VHDL or Parameters for Verilog
PARAMETER C_BASEADDR = 0xffffffff, DT = std_logic_vector, MIN_SIZE =
0x200, BUS = SOPB, ADDRESS = BASE, PAIR = C_HIGHADDR
PARAMETER C_HIGHADDR = 0x00000000, DT = std_logic_vector, BUS = SOPB,
ADDRESS = HIGH, PAIR = C_BASEADDR
PARAMETER C_OPB_AWIDTH = 32, DT = INTEGER, BUS = SOPB, ASSIGNMENT =
CONSTANT
PARAMETER C_OPB_DWIDTH = 32, DT = INTEGER, BUS = SOPB, RANGE = (8, 16,
32)
PARAMETER C_USER_ID_CODE = 3, DT = INTEGER, RANGE = (0:255)
PARAMETER C_FAMILY = virtex2p, DT = STRING

## Ports
PORT entrada_serie = "", DIR = I
PORT salida_serie = "", DIR = O
PORT OPB_Clk = "", DIR = I, SIGIS = Clk, BUS = SOPB
PORT OPB_Rst = OPB_Rst, DIR = I, SIGIS = Rst, BUS = SOPB
PORT Sl_DBus = Sl_DBus, DIR = O, VEC = [0:(C_OPB_DWIDTH-1)], BUS =
SOPB
PORT Sl_errAck = Sl_errAck, DIR = O, BUS = SOPB
PORT Sl_retry = Sl_retry, DIR = O, BUS = SOPB
PORT Sl_toutSup = Sl_toutSup, DIR = O, BUS = SOPB
PORT Sl_xferAck = Sl_xferAck, DIR = O, BUS = SOPB
PORT OPB_ABus = OPB_ABus, DIR = I, VEC = [0:(C_OPB_AWIDTH-1)], BUS =
SOPB
PORT OPB_BE = OPB_BE, DIR = I, VEC = [0:((C_OPB_DWIDTH/8)-1)], BUS =
SOPB
```

```
PORT OPB_DBus = OPB_DBus, DIR = I, VEC = [0:(C_OPB_DWIDTH-1)], BUS =
SOPB
PORT OPB_RNW = OPB_RNW, DIR = I, BUS = SOPB
PORT OPB_select = OPB_select, DIR = I, BUS = SOPB
PORT OPB_seqAddr = OPB_seqAddr, DIR = I, BUS = SOPB

END
```

A.5.3. Adaptador_uart.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

library proc_common_v2_00_a;
use proc_common_v2_00_a.proc_common_pkg.all;
use proc_common_v2_00_a.ipif_pkg.all;
library opb_ipif_v3_01_c;
use opb_ipif_v3_01_c.all;

library adaptador_uart_v1_00_a;
use adaptador_uart_v1_00_a.all;


entity adaptador_uart is
  generic
  (
    C_BASEADDR                     : std_logic_vector     :=
X"00000000";
    C_HIGHADDR                     : std_logic_vector     :=
X"0000FFFF";
    C_OPB_AWIDTH                   : integer              := 32;
    C_OPB_DWIDTH                   : integer              := 32;
    C_USER_ID_CODE                 : integer              := 3;
    C_FAMILY                       : string               :=
"virtex2p"
    -- DO NOT EDIT ABOVE THIS LINE ---------------------
  );
  port
  (
    entrada_serie                      : in std_logic;
    salida_serie                       : out std_logic;

    OPB_Clk                        : in  std_logic;
    OPB_Rst                        : in  std_logic;
    Sl_DBus                        : out std_logic_vector(0 to
C_OPB_DWIDTH-1);
    Sl_errAck                      : out std_logic;
    Sl_retry                       : out std_logic;
    Sl_toutSup                     : out std_logic;
    Sl_xferAck                     : out std_logic;
    OPB_ABus                       : in  std_logic_vector(0 to
C_OPB_AWIDTH-1);
    OPB_BE                         : in  std_logic_vector(0 to
C_OPB_DWIDTH/8-1);
```

```
    OPB_DBus                       : in  std_logic_vector(0 to
C_OPB_DWIDTH-1);
    OPB_RNW                        : in  std_logic;
    OPB_select                     : in  std_logic;
    OPB_seqAddr                    : in  std_logic
    -- DO NOT EDIT ABOVE THIS LINE ---------------------
  );

  attribute SIGIS : string;
  attribute SIGIS of OPB_Clk       : signal is "Clk";
  attribute SIGIS of OPB_Rst       : signal is "Rst";

end entity adaptador_uart;

------------------------------------------------------------------------
--------
-- Architecture section
------------------------------------------------------------------------
--------

architecture IMP of adaptador_uart is

  ---------------------------------------
  -- Constant: array of address range identifiers
  ---------------------------------------
  constant ARD_ID_ARRAY                    : INTEGER_ARRAY_TYPE   :=
    (
      0  => USER_00,                          -- user logic S/W
register address space
      1  => IPIF_RST                          -- include IPIF S/W
Reset/MIR service
    );

  ---------------------------------------
  -- Constant: array of address pairs for each address range
  ---------------------------------------
  constant ZERO_ADDR_PAD                   : std_logic_vector(0 to 64-
C_OPB_AWIDTH-1) := (others => '0');

  constant USER_BASEADDR                   : std_logic_vector     :=
C_BASEADDR or X"00000000";
  constant USER_HIGHADDR                   : std_logic_vector     :=
C_BASEADDR or X"000000FF";

  constant RST_BASEADDR                    : std_logic_vector     :=
C_BASEADDR or X"00000100";
  constant RST_HIGHADDR                    : std_logic_vector     :=
C_BASEADDR or X"000001FF";

  constant ARD_ADDR_RANGE_ARRAY            : SLV64_ARRAY_TYPE     :=
    (
      ZERO_ADDR_PAD & USER_BASEADDR,             -- user logic base
address
      ZERO_ADDR_PAD & USER_HIGHADDR,             -- user logic high
address
      ZERO_ADDR_PAD & RST_BASEADDR,              -- MIR/Reset
register base address
```

```
      ZERO_ADDR_PAD & RST_HIGHADDR                -- MIR/Reset
register high address
    );

  ------------------------------------------
  -- Constant: array of data widths for each target address range
  ------------------------------------------
  constant USER_DWIDTH                    : integer              :=
32;

  constant ARD_DWIDTH_ARRAY               : INTEGER_ARRAY_TYPE   :=
    (
      0  => USER_DWIDTH,                      -- user logic data width
      1  => C_OPB_DWIDTH                      -- MIR/Reset register
data width
    );

  ------------------------------------------
  -- Constant: array of desired number of chip enables for each
address range
  ------------------------------------------
  constant USER_NUM_CE                    : integer              := 4;

  constant ARD_NUM_CE_ARRAY               : INTEGER_ARRAY_TYPE   :=
    (
      0  => pad_power2(USER_NUM_CE),          -- user logic number of
CEs
      1  => 1                                 -- MIR/Reset register -
1 CE
    );

  ------------------------------------------
  -- Constant: array of unique properties for each address range
  ------------------------------------------
  constant ARD_DEPENDENT_PROPS_ARRAY      : DEPENDENT_PROPS_ARRAY_TYPE
:=
    (
      0  => (others => 0),                    -- user logic slave
space dependent properties (none defined)
      1  => (others => 0)                     -- IPIF reset/mir
dependent properties (none defined)
    );

  ------------------------------------------
  -- Constant: pipeline mode
  -- 1 = include OPB-In pipeline registers
  -- 2 = include IP pipeline registers
  -- 3 = include OPB-In and IP pipeline registers
  -- 4 = include OPB-Out pipeline registers
  -- 5 = include OPB-In and OPB-Out pipeline registers
  -- 6 = include IP and OPB-Out pipeline registers
  -- 7 = include OPB-In, IP, and OPB-Out pipeline registers
  -- Note:
  -- only mode 4, 5, 7 are supported for this release
  ------------------------------------------
  constant PIPELINE_MODEL                 : integer              := 5;
```

```
  ------------------------------------------
  -- Constant: user core ID code
  ------------------------------------------
  constant DEV_BLK_ID                     : integer              :=
C_USER_ID_CODE;

  ------------------------------------------
  -- Constant: enable MIR/Reset register
  ------------------------------------------
  constant DEV_MIR_ENABLE                 : integer              := 1;

  ------------------------------------------
  -- Constant: array of IP interrupt mode
  -- 1 = Active-high interrupt condition
  -- 2 = Active-low interrupt condition
  -- 3 = Active-high pulse interrupt event
  -- 4 = Active-low pulse interrupt event
  -- 5 = Positive-edge interrupt event
  -- 6 = Negative-edge interrupt event
  ------------------------------------------
  constant IP_INTR_MODE_ARRAY             : INTEGER_ARRAY_TYPE   :=
    (
      0  => 0  -- not used
    );

  ------------------------------------------
  -- Constant: enable device burst
  ------------------------------------------
  constant DEV_BURST_ENABLE               : integer              := 0;

  ------------------------------------------
  -- Constant: include address counter for burst transfers
  ------------------------------------------
  constant INCLUDE_ADDR_CNTR              : integer              := 0;

  ------------------------------------------
  -- Constant: include write buffer that decouples OPB and IPIC write
transactions
  ------------------------------------------
  constant INCLUDE_WR_BUF                 : integer              := 0;

  ------------------------------------------
  -- Constant: index for CS/CE
  ------------------------------------------
  constant USER00_CS_INDEX                : integer              :=
get_id_index(ARD_ID_ARRAY, USER_00);

  constant USER00_CE_INDEX                : integer              :=
calc_start_ce_index(ARD_NUM_CE_ARRAY, USER00_CS_INDEX);

  ------------------------------------------
  -- IP Interconnect (IPIC) signal declarations -- do not delete
  -- prefix 'i' stands for IPIF while prefix 'u' stands for user logic
  -- typically user logic will be hooked up to IPIF directly via
i<sig>
  -- unless signal slicing and muxing are needed via u<sig>
  ------------------------------------------
```

```
  signal iBus2IP_RdCE                   : std_logic_vector(0 to
calc_num_ce(ARD_NUM_CE_ARRAY)-1);
  signal iBus2IP_WrCE                   : std_logic_vector(0 to
calc_num_ce(ARD_NUM_CE_ARRAY)-1);
  signal iBus2IP_Data                   : std_logic_vector(0 to
C_OPB_DWIDTH-1);
  signal iBus2IP_BE                     : std_logic_vector(0 to
C_OPB_DWIDTH/8-1);
  signal iIP2Bus_Data                   : std_logic_vector(0 to
C_OPB_DWIDTH-1)   := (others => '0');
  signal iIP2Bus_Ack                    : std_logic    := '0';
  signal iIP2Bus_Error                  : std_logic    := '0';
  signal iIP2Bus_Retry                  : std_logic    := '0';
  signal iIP2Bus_ToutSup                : std_logic    := '0';
  signal DISABLE_POSTED_WRITE           : std_logic_vector(0 to
ARD_ID_ARRAY'length-1)   := (others => '1'); -- disable posted write
behavior for acknowledged write behavior
  signal ZERO_IP2RFIFO_Data             : std_logic_vector(0 to
ARD_DWIDTH_ARRAY(get_id_index_iboe(ARD_ID_ARRAY, IPIF_RDFIFO_DATA))-1)
:= (others => '0'); -- work around for XST not taking (others => '0')
in port mapping
  signal ZERO_WFIFO2IP_Data             : std_logic_vector(0 to
ARD_DWIDTH_ARRAY(get_id_index_iboe(ARD_ID_ARRAY, IPIF_WRFIFO_DATA))-1)
:= (others => '0'); -- work around for XST not taking (others => '0')
in port mapping
  signal ZERO_IP2Bus_IntrEvent          : std_logic_vector(0 to
IP_INTR_MODE_ARRAY'length-1)   := (others => '0'); -- work around for
XST not taking (others => '0') in port mapping
  signal iBus2IP_Clk                    : std_logic;
  signal iBus2IP_Reset                  : std_logic;
  signal uBus2IP_Data                   : std_logic_vector(0 to
USER_DWIDTH-1);
  signal uBus2IP_BE                     : std_logic_vector(0 to
USER_DWIDTH/8-1);
  signal uBus2IP_RdCE                   : std_logic_vector(0 to
USER_NUM_CE-1);
  signal uBus2IP_WrCE                   : std_logic_vector(0 to
USER_NUM_CE-1);
  signal uIP2Bus_Data                   : std_logic_vector(0 to
USER_DWIDTH-1);

begin

  ------------------------------------------
  -- instantiate the OPB IPIF
  ------------------------------------------
  OPB_IPIF_I : entity opb_ipif_v3_01_c.opb_ipif
    generic map
    (
      C_ARD_ID_ARRAY                => ARD_ID_ARRAY,
      C_ARD_ADDR_RANGE_ARRAY        => ARD_ADDR_RANGE_ARRAY,
      C_ARD_DWIDTH_ARRAY            => ARD_DWIDTH_ARRAY,
      C_ARD_NUM_CE_ARRAY            => ARD_NUM_CE_ARRAY,
      C_ARD_DEPENDENT_PROPS_ARRAY   => ARD_DEPENDENT_PROPS_ARRAY,
      C_PIPELINE_MODEL              => PIPELINE_MODEL,
      C_DEV_BLK_ID                  => DEV_BLK_ID,
      C_DEV_MIR_ENABLE              => DEV_MIR_ENABLE,
```

```
    C_OPB_AWIDTH                    => C_OPB_AWIDTH,
    C_OPB_DWIDTH                    => C_OPB_DWIDTH,
    C_FAMILY                        => C_FAMILY,
    C_IP_INTR_MODE_ARRAY            => IP_INTR_MODE_ARRAY,
    C_DEV_BURST_ENABLE              => DEV_BURST_ENABLE,
    C_INCLUDE_ADDR_CNTR             => INCLUDE_ADDR_CNTR,
    C_INCLUDE_WR_BUF                => INCLUDE_WR_BUF
  )
  port map
  (
    OPB_select                      => OPB_select,
    OPB_DBus                        => OPB_DBus,
    OPB_ABus                        => OPB_ABus,
    OPB_BE                          => OPB_BE,
    OPB_RNW                         => OPB_RNW,
    OPB_seqAddr                     => OPB_seqAddr,
    Sln_DBus                        => Sl_DBus,
    Sln_xferAck                     => Sl_xferAck,
    Sln_errAck                      => Sl_errAck,
    Sln_retry                       => Sl_retry,
    Sln_toutSup                     => Sl_toutSup,
    Bus2IP_CS                       => open,
    Bus2IP_CE                       => open,
    Bus2IP_RdCE                     => iBus2IP_RdCE,
    Bus2IP_WrCE                     => iBus2IP_WrCE,
    Bus2IP_Data                     => iBus2IP_Data,
    Bus2IP_Addr                     => open,
    Bus2IP_AddrValid                => open,
    Bus2IP_BE                       => iBus2IP_BE,
    Bus2IP_RNW                      => open,
    Bus2IP_Burst                    => open,
    IP2Bus_Data                     => iIP2Bus_Data,
    IP2Bus_Ack                      => iIP2Bus_Ack,
    IP2Bus_AddrAck                  => '0',
    IP2Bus_Error                    => iIP2Bus_Error,
    IP2Bus_Retry                    => iIP2Bus_Retry,
    IP2Bus_ToutSup                  => iIP2Bus_ToutSup,
    IP2Bus_PostedWrInh              => DISABLE_POSTED_WRITE,
    IP2RFIFO_Data                   => ZERO_IP2RFIFO_Data,
    IP2RFIFO_WrMark                 => '0',
    IP2RFIFO_WrRelease              => '0',
    IP2RFIFO_WrReq                  => '0',
    IP2RFIFO_WrRestore              => '0',
    RFIFO2IP_AlmostFull             => open,
    RFIFO2IP_Full                   => open,
    RFIFO2IP_Vacancy                => open,
    RFIFO2IP_WrAck                  => open,
    IP2WFIFO_RdMark                 => '0',
    IP2WFIFO_RdRelease              => '0',
    IP2WFIFO_RdReq                  => '0',
    IP2WFIFO_RdRestore              => '0',
    WFIFO2IP_AlmostEmpty            => open,
    WFIFO2IP_Data                   => ZERO_WFIFO2IP_Data,
    WFIFO2IP_Empty                  => open,
    WFIFO2IP_Occupancy              => open,
    WFIFO2IP_RdAck                  => open,
    IP2Bus_IntrEvent                => ZERO_IP2Bus_IntrEvent,
```

```
      IP2INTC_Irpt                  => open,
      Freeze                        => '0',
      Bus2IP_Freeze                 => open,
      OPB_Clk                       => OPB_Clk,
      Bus2IP_Clk                    => iBus2IP_Clk,
      IP2Bus_Clk                    => '0',
      Reset                         => OPB_Rst,
      Bus2IP_Reset                  => iBus2IP_Reset
    );

  ----------------------------------------
  -- instantiate the User Logic
  ----------------------------------------
  USER_LOGIC_I : entity adaptador_uart_v1_00_a.user_logic
    generic map
    (
      -- MAP USER GENERICS BELOW THIS LINE ---------------
      --USER generics mapped here
      -- MAP USER GENERICS ABOVE THIS LINE ---------------

      C_DWIDTH                      => USER_DWIDTH,
      C_NUM_CE                      => USER_NUM_CE
    )
    port map
    (
      -- MAP USER PORTS BELOW THIS LINE ------------------
      --USER ports mapped here
      -- MAP USER PORTS ABOVE THIS LINE ------------------

      Bus2IP_Clk                    => iBus2IP_Clk,
      Bus2IP_Reset                  => iBus2IP_Reset,
      Bus2IP_Data                   => uBus2IP_Data,
      Bus2IP_BE                     => uBus2IP_BE,
      Bus2IP_RdCE                   => uBus2IP_RdCE,
      Bus2IP_WrCE                   => uBus2IP_WrCE,
      IP2Bus_Data                   => uIP2Bus_Data,
      IP2Bus_Ack                    => iIP2Bus_Ack,
      IP2Bus_Retry                  => iIP2Bus_Retry,
      IP2Bus_Error                  => iIP2Bus_Error,
      IP2Bus_ToutSup                => iIP2Bus_ToutSup,

      entrada_serie                 => entrada_serie,
      salida_serie                  => salida_serie
    );

  ----------------------------------------
  -- hooking up signal slicing
  ----------------------------------------
  uBus2IP_BE <= iBus2IP_BE(0 to USER_DWIDTH/8-1);
  uBus2IP_Data <= iBus2IP_Data(0 to USER_DWIDTH-1);
  uBus2IP_RdCE <= iBus2IP_RdCE(USER00_CE_INDEX to
USER00_CE_INDEX+USER_NUM_CE-1);
  uBus2IP_WrCE <= iBus2IP_WrCE(USER00_CE_INDEX to
USER00_CE_INDEX+USER_NUM_CE-1);
  iIP2Bus_Data(0 to USER_DWIDTH-1) <= uIP2Bus_Data;

end IMP;
```

A.5.4. User_logic.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

library proc_common_v2_00_a;
use proc_common_v2_00_a.proc_common_pkg.all;

entity user_logic is
  generic
  (
    C_DWIDTH                       : integer              := 32;
    C_NUM_CE                       : integer              := 4
    -- DO NOT EDIT ABOVE THIS LINE ----------------------
  );
  port
  (
    Bus2IP_Clk                     : in  std_logic;
    Bus2IP_Reset                   : in  std_logic;
    Bus2IP_Data                    : in  std_logic_vector(0 to
C_DWIDTH-1);
    Bus2IP_BE                      : in  std_logic_vector(0 to
C_DWIDTH/8-1);
    Bus2IP_RdCE                    : in  std_logic_vector(0 to
C_NUM_CE-1);
    Bus2IP_WrCE                    : in  std_logic_vector(0 to
C_NUM_CE-1);
    IP2Bus_Data                    : out std_logic_vector(0 to
C_DWIDTH-1);
    IP2Bus_Ack                     : out std_logic;
    IP2Bus_Retry                   : out std_logic;
    IP2Bus_Error                   : out std_logic;
    IP2Bus_ToutSup                 : out std_logic;

    entrada_serie                  : in std_logic;
    salida_serie                   : out std_logic
    -- DO NOT EDIT ABOVE THIS LINE ----------------------
  );
end entity user_logic;

------------------------------------------------------------------------
--------
-- Architecture section
------------------------------------------------------------------------
--------

architecture IMP of user_logic is

  --USER signal declarations added here, as needed for user logic

  ---------------------------------------
  -- Signals for user logic slave model s/w accessible register
example
```

```
  ---------------------------------------
  signal slv_reg0                       : std_logic_vector(0 to
C_DWIDTH-1);
  signal slv_reg1                       : std_logic_vector(0 to
C_DWIDTH-1);
  signal slv_reg2                       : std_logic_vector(0 to
C_DWIDTH-1);
  signal slv_reg3                       : std_logic_vector(0 to
C_DWIDTH-1);
  signal slv_reg_write_select           : std_logic_vector(0 to 3);
  signal slv_reg_read_select            : std_logic_vector(0 to 3);
  signal slv_ip2bus_data                : std_logic_vector(0 to
C_DWIDTH-1);
  signal slv_read_ack                   : std_logic;
  signal slv_write_ack                  : std_logic;

   ---------------------------------------------------------------------
---------
  signal senal_tx_ocupada                    : std_logic_vector(0 to
31);
  signal senal_tx_inicio                     : std_logic;
  signal senal_tx_fin                        : std_logic;
  signal senal_tx_32                         : std_logic_vector(0 to
31);
  signal senal_tx_inicio_aux                 : std_logic_vector(0 to
3);

  signal senal_rx_fin                        : std_logic;
  signal senal_rx_32                         : std_logic_vector(0 to
31);

  signal tiempo                              : integer := 0;

  COMPONENT Entrada_salida PORT (
        clk : in  STD_LOGIC;
        resetN : in  STD_LOGIC;
        tx_32 : in STD_LOGIC_VECTOR(0 to 31);
        tx_1 : out  STD_LOGIC;
        tx_ocupada: out STD_LOGIC_vector(0 to 31);
        tx_inicio : in std_logic;
        tx_fin : out std_logic;

        rx_32 : out std_logic_vector(0 to 31);
        rx_1  : in std_logic;
        rx_fin : out std_logic
    );
  END COMPONENT;
---------------------------------------------------------------------------
------

begin

  --USER logic implementation added here

  ---------------------------------------
  -- Example code to read/write user logic slave model s/w accessible
registers
```

```
  --
  -- Note:
  -- The example code presented here is to show you one way of
reading/writing
  -- software accessible registers implemented in the user logic slave
model.
  -- Each bit of the Bus2IP_WrCE/Bus2IP_RdCE signals is configured to
correspond
  -- to one software accessible register by the top level template.
For example,
  -- if you have four 32 bit software accessible registers in the user
logic, you
  -- are basically operating on the following memory mapped registers:
  --
  --    Bus2IP_WrCE or   Memory Mapped
  --       Bus2IP_RdCE   Register
  --            "1000"   C_BASEADDR + 0x0
  --            "0100"   C_BASEADDR + 0x4
  --            "0010"   C_BASEADDR + 0x8
  --            "0001"   C_BASEADDR + 0xC
  --
  ---------------------------------------
  slv_reg_write_select <= Bus2IP_WrCE(0 to 3);
  slv_reg_read_select  <= Bus2IP_RdCE(0 to 3);
  slv_write_ack        <= Bus2IP_WrCE(0) or Bus2IP_WrCE(1) or
Bus2IP_WrCE(2) or Bus2IP_WrCE(3);
  slv_read_ack         <= Bus2IP_RdCE(0) or Bus2IP_RdCE(1) or
Bus2IP_RdCE(2) or Bus2IP_RdCE(3);

  -- implement slave model register(s)
  SLAVE_REG_WRITE_PROC : process( Bus2IP_Clk ) is
  begin

    if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
      if Bus2IP_Reset = '1' then
        slv_reg0 <= (others => '0');
        slv_reg1 <= (others => '0');
        slv_reg2 <= (others => '0');
        slv_reg3 <= (others => '0');
      else
        case slv_reg_write_select is
                when "1000" =>
                    for byte_index in 0 to (C_DWIDTH/8)-1 loop
                        if ( Bus2IP_BE(byte_index) = '1' ) then
                            slv_reg0(byte_index*8 to byte_index*8+7)
<= Bus2IP_Data(byte_index*8 to byte_index*8+7);
                            senal_tx_inicio_aux(byte_index) <= '1';
                            tiempo <= 0;
                        end if;
                    end loop;
                when "0100" =>
                    for byte_index in 0 to (C_DWIDTH/8)-1 loop
                        if ( Bus2IP_BE(byte_index) = '1' ) then
                            slv_reg1(byte_index*8 to byte_index*8+7)
<= Bus2IP_Data(byte_index*8 to byte_index*8+7);
                        end if;
                    end loop;
```

```
                when "0010" =>
                    for byte_index in 0 to (C_DWIDTH/8)-1 loop
                        if ( Bus2IP_BE(byte_index) = '1' ) then
                            slv_reg2(byte_index*8 to byte_index*8+7)
<= Bus2IP_Data(byte_index*8 to byte_index*8+7);
                        end if;
                    end loop;
                when "0001" =>
                    for byte_index in 0 to (C_DWIDTH/8)-1 loop
                        if ( Bus2IP_BE(byte_index) = '1' ) then
                            slv_reg3(byte_index*8 to byte_index*8+7)
<= Bus2IP_Data(byte_index*8 to byte_index*8+7);
                        end if;
                    end loop;
              when others => null;
        end case;
      end if;
      -- para que senal_tx_inicio se ponga a 1 8 ciclos despues de
escribir en el bus
        if senal_tx_inicio_aux = "1111" then
            tiempo <= tiempo + 1;
            if tiempo = 8 then
                senal_tx_inicio <= '1';
                tiempo <= 0;
                senal_tx_inicio_aux <= "0000";
            end if;
        end if;
        if senal_tx_inicio = '1' then
            senal_tx_inicio <= '0';
        end if;

        --TRANSMISION
        senal_tx_32 <= slv_reg0;
        slv_reg1 <= senal_tx_ocupada;

        --RECEPCION
        slv_reg2 <= senal_rx_32;
        if senal_rx_fin = '1' then
            slv_reg3 <= x"FFFFFFFF";
        end if;

    end if;

  end process SLAVE_REG_WRITE_PROC;

  -- implement slave model register read mux
  SLAVE_REG_READ_PROC : process( slv_reg_read_select, slv_reg0,
slv_reg1, slv_reg2, slv_reg3 ) is
  begin

    case slv_reg_read_select is
      when "1000" => slv_ip2bus_data <= slv_reg0;
      when "0100" => slv_ip2bus_data <= slv_reg1;
      when "0010" => slv_ip2bus_data <= slv_reg2;
      when "0001" => slv_ip2bus_data <= slv_reg3;
      when others => slv_ip2bus_data <= (others => '0');
    end case;
```

```
  end process SLAVE_REG_READ_PROC;

  ---------------------------------------
  -- Example code to drive IP to Bus signals
  ---------------------------------------
  IP2Bus_Data        <= slv_ip2bus_data;

  IP2Bus_Ack         <= slv_write_ack or slv_read_ack;
  IP2Bus_Error       <= '0';
  IP2Bus_Retry       <= '0';
  IP2Bus_ToutSup     <= '0';

  ------------------------------------------------------------------------
--------
  ES : Entrada_salida PORT MAP(
        clk => Bus2IP_Clk,
        resetN => not Bus2IP_Reset,
        tx_32 => senal_tx_32,
        tx_1 => salida_serie,
        tx_ocupada => senal_tx_ocupada,
        tx_inicio => senal_tx_inicio,
        tx_fin => senal_tx_fin,

        rx_32 => senal_rx_32,
        rx_1  => entrada_serie,
        rx_fin => senal_rx_fin
  );
------------------------------------------------------------------------
------
end IMP;
```

A.5.5. Entrada_salida.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--Descripcion de la entidad
entity Entrada_salida is
    Port (
        clk : in  STD_LOGIC;
        resetN : in  STD_LOGIC;
        tx_32 : in STD_LOGIC_VECTOR(0 to 31);
        tx_1 : out  STD_LOGIC;
        tx_ocupada : out std_logic_vector(0 to 31);
        tx_inicio : in std_logic;
        tx_fin : out std_logic;

        rx_32 : out std_logic_vector(0 to 31);
        rx_1  : in std_logic;
        rx_fin : out std_logic
    );
end Entrada_salida;
```

```
architecture Behavioral of Entrada_salida is
    component Tx_serie
        Port ( RstN : in  STD_LOGIC;
               clk : in  STD_LOGIC;
               Transmite : in  STD_LOGIC;
               DatoTxIn : in  STD_LOGIC_VECTOR (7 downto 0);
               Transmitiendo : out  STD_LOGIC;
               DatoSerieOut : out  STD_LOGIC);
    end component;

    component Rx_serie
        Port( Rstn : in  STD_LOGIC;
              Clk : in  STD_LOGIC;
           RxDatoSerie : in  STD_LOGIC;
           DatoRxOut : out  STD_LOGIC_VECTOR (0 to 7);
           AvisoRx : out  STD_LOGIC;
           Recibiendo : out  STD_LOGIC);
    end component;

    type txEstado is (TX_INICIAL, TX_E1, TX_E2, TX_E3, TX_E4);
    type rxEstado is (RX_INICIAL, RX_E1, RX_E2, RX_E3);
    type estado is (E0,E1,E2,E3);

    signal tx_estado, tx_sig_estado : txEstado := TX_INICIAL;
    signal rx_estado, rx_sig_estado : rxEstado := RX_INICIAL;
    signal est,sig_est : estado := E0;

    signal senal_tx_fin : std_logic;
    signal senal_tx_ocupada : std_logic_vector(0 to 31);

    signal senal_rx_fin : std_logic;
    signal fin_recepcion : std_logic;


   --Señales intermedias para la entrada y la salida. Se conectarán a
las entradas y las salidas del circuito principal
    signal Reg_entradas: std_logic_vector(0 to 31);
    signal Reg_salidas: std_logic_vector(0 to 31);

    --Señales para la Transmision Serie
    signal mi_resetserie:std_logic;
    signal mi_transmite:std_logic;
    signal mi_datotxin:std_logic_vector(7 downto 0);
    signal mi_transmitiendo:std_logic;
    signal mi_datoserieout:std_logic;

    --se□s para la Recepcion Serie
    signal mi_rxdatoserie : std_logic;
    signal mi_datorxout : std_logic_vector(7 downto 0);
    signal mi_avisorx : std_logic;
    signal mi_recibiendo : std_logic;


begin
    --Asignación de señales a los componentes de la entrada/salida
```

```
    T: Tx_serie port
map(mi_resetserie,clk,mi_transmite,mi_datotxin,mi_transmitiendo,mi_dat
oserieout);
    R: Rx_serie port
map(mi_resetserie,clk,mi_rxdatoserie,mi_datorxout,mi_avisorx,mi_recibi
endo);


   --PROCESO DE TRANSMISION DE DATOS--
    ------------------------------------------------------------------
-------------------------
    TX_SINCRONO:process(clk,resetN)
    begin
        if resetN = '0' then
            tx_estado <= TX_INICIAL;
        elsif clk'event and clk='1' then
            tx_estado <= tx_sig_estado;
            Reg_salidas <=   tx_32;

            case tx_estado is
                when TX_INICIAL =>
                    if tx_inicio = '1'  then
                        mi_transmite <= '1';
                        senal_tx_ocupada <= x"FFFFFFFF";
                    else
                        mi_transmite <= '0';
                        senal_tx_ocupada <= x"00000000";
                    end if;
                when TX_E1 =>
                    mi_datotxin <= Reg_salidas(0 to 7);
                    mi_transmite <= '1';
                    senal_tx_ocupada <= x"FFFFFFFF";
                when TX_E2 =>
                    mi_datotxin <= Reg_salidas(8 to 15);
                    mi_transmite <= '1';
                    senal_tx_ocupada <= x"FFFFFFFF";
                when TX_E3 =>
                    mi_datotxin <= Reg_salidas(16 to 23);
                    mi_transmite <= '1';
                    senal_tx_ocupada <= x"FFFFFFFF";
                when TX_E4 =>
                    mi_datotxin <= Reg_salidas(24 to 31);
                    mi_transmite <= '1';
                    senal_tx_ocupada <= x"FFFFFFFF";
            end case;
        end if;
    end process TX_SINCRONO;

    TX_COMB:process(tx_estado, tx_inicio, mi_transmitiendo)
    begin
            tx_sig_estado <= tx_estado;
            case tx_estado is
                when TX_INICIAL =>
                    senal_tx_fin <= '0';
                    if tx_inicio = '1'  then
                        tx_sig_estado <= TX_E1;
                    else
```

```
                    tx_sig_estado <= TX_INICIAL;
                end if;
            when TX_E1 =>
                senal_tx_fin <= '0';
                if mi_transmitiendo = '0' then
                    tx_sig_estado <= TX_E2;
                else
                    tx_sig_estado <= TX_E1;
                end if;
            when TX_E2 =>
                senal_tx_fin <= '0';
                if mi_transmitiendo = '0' then
                    tx_sig_estado <= TX_E3;
                else
                    tx_sig_estado <= TX_E2;
                end if;
            when TX_E3 =>
                senal_tx_fin <= '0';
                if mi_transmitiendo = '0' then
                    tx_sig_estado <= TX_E4;
                else
                    tx_sig_estado <= TX_E3;
                end if;
            when TX_E4 =>
                if mi_transmitiendo = '0' then
                    tx_sig_estado <= TX_INICIAL;
                    senal_tx_fin <= '1';
                else
                    tx_sig_estado <= TX_E4;
                    senal_tx_fin <= '0';
                end if;
        end case;
end process TX_COMB;

RX_SINCRONO:process(clk,mi_resetserie)
begin
    if resetN = '0' then
        rx_estado <= RX_INICIAL;
    elsif clk'event and clk='1' then
        rx_estado <= rx_sig_estado;
        rx_32 <= Reg_entradas;
        case rx_estado is
            when RX_INICIAL =>

            when RX_E1 =>

            when RX_E2 =>

            when RX_E3 =>

        end case;
    end if;
end process RX_SINCRONO;

RX_COMB:process(mi_recibiendo,rx_estado)
begin
    if mi_recibiendo'event and mi_recibiendo = '0'then
```

```
            fin_recepcion <= '0';
            case rx_estado is
                when RX_INICIAL =>
                    fin_recepcion <= '0';
                    Reg_entradas(0 to 7) <= mi_datorxout;
                    rx_sig_estado <= RX_E1;
                when RX_E1 =>
                    fin_recepcion <= '0';
                    Reg_entradas(8 to 15) <= mi_datorxout;
                    rx_sig_estado <= RX_E2;
                when RX_E2 =>
                    fin_recepcion <= '0';
                    Reg_entradas(16 to 23) <= mi_datorxout;
                    rx_sig_estado <= RX_E3;
                when RX_E3 =>
                    fin_recepcion<= '1';
                    Reg_entradas(24 to 31) <= mi_datorxout;
                    rx_sig_estado <= RX_INICIAL;
            end case;
        end if;
    end process RX_COMB;

    process(clk,resetN)
    begin
        if resetN = '0' then
            est <= E0;
        elsif clk'event and clk='1' then
            est <= sig_est;
        end if;
    end process;

    process(senal_rx_fin,fin_recepcion)
    begin
        sig_est <= est;
        case est is
            when E0 =>
                senal_rx_fin <= '0';
                if fin_recepcion = '1' then
                    sig_est <= E1;
                end if;
            when E1 =>
                senal_rx_fin <= '1';
                sig_est <= E2;
            when E2 =>
                senal_rx_fin <= '1';
                sig_est <= E3;
            when E3 =>
                senal_rx_fin <= '0';
                if fin_recepcion = '0' then
                    sig_est <= E0;
                end if;
        end case;
    end process;

    tx_1 <= mi_datoserieout;
    tx_fin <= senal_tx_fin;
    tx_ocupada <= senal_tx_ocupada;
```

```
    mi_rxdatoserie <= rx_1;
    rx_fin <= senal_rx_fin;

    mi_resetserie <= resetN;

end Behavioral;
```

A.5.6. Rx_serie.vhd

```
----------------------------------------------------------------------------
------------
-- Company:         UCM Facultad de Inform□ca
-- Engineer:        Carlos S□ hez-Vellisco S□ hez
--                  Antonio Jos頂arc▪ Mart□z
--                  David Fern□ez Maiquez
--
-- Create Date:     18:31:36 11/18/2009
-- Design Name:     Transmisor Serie
-- Module Name:     Rx_Serie - Behavioral
-- Project Name:    Nessy 2.0
-- Target Devices:  XC2VP30
-- Tool versions:   Xilinx 10.1
-- Description:     Receptor Serie ( Protocolo RS232)
--                  Se divide el dise□n bloques
--
--                      ->Divisor de frecuencia
--                      ->Registro de desplazamiento: El registro de
desplazamiento
--                      Serial In/Parallel Out (Desplz_SIPO) es un
registro al que se
--                      le van cargando los datos en serie y los
devuelve en paralelo.
--                      Como el primer bit que se recibe es el 0, si
la carga serie se
--                      hace por el bit m□ significativo del registro y
se hacen
--                      desplazar los bits hacia la derecha, en el
ltimo desplazamiento
--                      el bit 0 recibido estar□n el bit menos
significativo del
--                      registro, estando as□odos los bits ordenados.
La carga y el
--                      desplazamiento se realizan bajo la orden de la
se□Desplaza
--                      que sale del bloque de control.
--                      ->Registro de entrada: Como la entrada de la
comunicaci□erie
--                      RxDatoserie es as□rona, se almacena en un
registro RxDatoReg
--                      para evitar pulsos y entradas no deseadas.
--                      ->Circuito de control
----------------------------------------------------------------------------
--------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
----------------------------------------------------------------------
-------------
-- PUERTO       NBits      INFO
--
-- RstN             1       Entrada Se□de reset as□rono (activo a 0)
-- Clk              1       Entrada Se□de reloj de la placa, en
principio de 100MHz,
--                          pero configurable por gFrecClk
-- RxDatoSerie      1       Entrada Trama que recibe del PC, que sigue
el protocolo RS232
-- DatoRxOut        8       Salida  El dato que se ha recibido. Este
dato s□es v□ do
--                          desde que AvisoRx vale 1 y mientras
recibiendo sea 0.
-- AvisoRx          1       Salida  Aviso de que se ha recibido un
nuevo dato y que est˜Hspan>
--                          disponible en DatoRxOut. El aviso se
dar□oniendo la se□a
--                          1 durante un ciclo de reloj.
-- Recibiendo       1       Salida  Indica que el receptor se
encuentra recibiendo un dato
--                          y por lo tanto el valor de DatorxOut no es
v□ do.

----------------------------------------------------------------------
-------------
entity Rx_Serie is
     generic (
                gFrecClk : integer :=50000000;
                gBaud : integer :=19200
                );
    Port ( Rstn : in  STD_LOGIC;
           Clk : in  STD_LOGIC;
           RxDatoSerie : in  STD_LOGIC;
           DatoRxOut : out  STD_LOGIC_VECTOR (7 downto 0);
           AvisoRx : out  STD_LOGIC;
           Recibiendo : out  STD_LOGIC);
end Rx_Serie;

----------------------------------------------------------------------
------------

architecture Behavioral of Rx_Serie is


   -- Tipo nuevo para los estados --

    type t_Estado is (eInit,eBitInit,eBitsDato,eBitFin);
    constant cFinCuenta : natural := (gFrecClk/gBaud)-1;
```

```
    constant cFinCuenta_Medio : natural := cFinCuenta / 2 ;

    -- Se□s auxiliares

    signal Cuenta,CuentaBits: integer;
    signal ClkBaud,EnableCont,Dsplza,FinDsplza8bits,ClkBaudMedio :
std_logic;

    signal Estado : t_Estado;
    signal Rx_Registro: STD_LOGIC_VECTOR (7 downto 0);
    signal MiDatoTxIn:STD_LOGIC_VECTOR (7 downto 0);
    signal SalidaSeleccion:std_logic;
    signal RxDatoReg : std_logic;

begin

------------------------------------------------------------------------
------------
    -- A partir del reloj de la placa de 100 MHz (Clk), queremos
proporcionar una
    -- se□con frecuencia de 9600 Hz (ClkBaud). Este reloj tendr□or
tanto un
    -- periodo de 104,167 s, y estar□ 1 durante un solo ciclo de
reloj, estando
    -- el resto de tiempo a 0.

    P_DivFrec: Process (RstN, Clk)
    begin
            if RstN = '0' then
                Cuenta <= 0;
            ClkBaud <= '0';
                ClkBaudMedio <= '0';
                --Rx_Registro<="11111111";----??
            elsif EnableCont = '1' then
                if Clk'event and Clk='1' then
                        if Cuenta = cFinCuenta then
                            Cuenta <= 0;
                            ClkBaud <= '1';
                            ClkBaudMedio <= '0';
                        elsif Cuenta = cFinCuenta_Medio then
                            ClkBaudMedio <= '1';
                            Cuenta <= Cuenta + 1;
                            ClkBaud <= '0';
                        else
                            Cuenta <= Cuenta + 1;
                            ClkBaud <= '0';
                            ClkBaudMedio <= '0';
                        end if;
                end if;
            end if;
    end process;

------------------------------------------------------------------------
------------

    -- Proceso encargado de contar los 8 bits de datos.
```

```
    -- Solo cuenta cuando estamos enviando datos, es decir en el
estado eBitsDato

    P_CuentaBits: Process (RstN, Clk)
    begin
        if RstN = '0' then
            CuentaBits <= 0;
        elsif Clk'event and Clk='1' then
            if Estado = eBitsDato then
                if ClkBaud = '1' then
                    if CuentaBits = 7 then
                        CuentaBits <= 0;
                    else
                        CuentaBits <= CuentaBits + 1;
                    end if;
                end if;
            else
                CuentaBits <= 0;
            end if;
        end if;
    end process;

-------------------------------------------------------------------------
------------

    -- Proceso que controla las se□s internas necesarias para el
    -- registro de desplazamiento y para el contador de los 8 bits de
datos

    POut: Process (Estado, RxDatoReg, ClkBaudMedio,ClkBaud) --
RxDatoSerie
    begin
        EnableCont <= '0';
        Dsplza <= '0';
        --recibiendo <= '1';
        case Estado is
            when eInit =>
                EnableCont <= '0';
                Recibiendo <= '0';
                AvisoRx <= '0';
                Dsplza <= '0';
                if RxDatoReg = '0' then
                   AvisoRx <= '1';
                    EnableCont <= '1';
                end if;
            when eBitInit =>
                EnableCont <= '1';
                Recibiendo <= '1';
            when eBitsDato =>
                EnableCont <= '1';
                Recibiendo <= '1';
                if ClkBaudMedio = '1' then
                    Dsplza <= '1';
                end if;
            when eBitFin =>
                Recibiendo <= '1';
                if ClkBaud = '1' then
```

```
                    EnableCont <= '0';
                else
                    EnableCont <= '1';
                end if;
            end case;
    end process;
-------------------------------------------------------------------------
------------

-- Proceso de la m□ina de estados
-- eInit:    Es el estado inicial. El sistema est□n reposo esperando
la orden de
--           transmitir. Cuando la se□Transmite se ponga a 1, se
pasar□ enviar el
--           bit de inicio, pasando para ello al siguiente estado
(eBitInit). En ese momento
--           se dar□a orden de cargar el dato (DatoTxIn) en el
registro de desplazamiento.
--           Tambi鯡tendremos que sincronizar el contador del divisor
de frecuencia; para
--           esto haremos que en el estado inicial no cuente, y en el
resto se habilite el
--           contador.
-- eBitInit: En este estado se est□nviando el bit de inicio. Se
saldr□e este estado
--           al recibir un pulso de ClkBaud, que nos dir□ue debemos
pasar a enviar los bits
--           de dato. El siguiente estado es eBitsDato.
-- eBitsDato:Este estado se encarga de enviar los 8 bits del dato.
Utilizando un contador,
--           llevaremos la cuenta del nmero de bits que se han
enviado. Cuando se hayan
--           enviado los 8 bits es decir, cuando hayan llegado 8
pulsos de Clkbaud-  se
--           activar□a se□FinDsplza8bits que har□ue cambiemos al
siguiente estado
--           (eBitFin).
-- eBitFin:  Este estado env▪el bit de fin. Al llegar el siguiente
pulso de ClkBaud,
--           cambiaremos al estado inicial eInit.

    P_Control_FSM: Process (RstN, Clk)
    begin
        if RstN = '0' then
            Estado <= eInit;
        elsif Clk'event and Clk='1' then
            case Estado is
                when eInit =>
                    if RxDatoReg = '0' then
                        Estado <= eBitInit;
                    end if;
                when eBitInit =>
                    if ClkBaud = '1' then
                        Estado <= eBitsDato;
                    end if;
                when eBitsDato =>
```

```
                    if FinDsplza8bits = '1' then
                        Estado <= eBitFin;
                    end if;
                when eBitFin =>
                    if ClkBaud = '1' then
                        Estado <= eInit;
                    end if;
            end case;
        end if;
    end process;
------------------------------------------------------------------------
------------

    -- Proceso que se encarga de ir poniendo en SalReg el bit del
registro
    -- de desplazamiento concecreto.

    recibe_desplazamiento: process(Dsplza)
    begin
        if Estado = eBitsDato and Dsplza = '1' then
            Rx_Registro(CuentaBits) <= RxDatoReg ;
        end if;
    end process;

------------------------------------------------------------------------
------------
    -- Proceso que funciona como un biestable con la se□de entrada
    -- RxDatoReg seleccionando un 1 cuando no hay nada en la l□a.

    biestable: process(Clk,RstN)
    begin
        if (RstN = '0') then
            RxDatoReg <= '1';
        elsif (Clk'event and Clk = '1') then
            RxDatoReg <= RxDatoSerie;
        end if;
    end process;

DatoRxOut <= Rx_Registro;
FinDsplza8bits <= '1' when CuentaBits=7 and ClkBaud = '1' else '0';

end Behavioral;
```

A.5.7. Tx_serie.vhd

```
------------------------------------------------------------------------
------------
-- Company:  UCM Facultad de Inform□ca
-- Engineer:    Carlos S□ hez-Vellisco S□ hez
--              Antonio Jos頂arc▪ Mart□z
--              David Fern□ez Maiquez
--
-- Create Date:     19:27:16 11/05/2009
-- Design Name:     Transmisor Serie
-- Module Name:     Tx_serie - Behavioral
-- Project Name:    Nessy 2.0
```

```
-- Target Devices:  XC2VP30
-- Tool versions:   Xilinx 10.1
-- Description:     Transmisor serie  ( Protocolo RS232)
--                  Lo dividiremos en cuatro bloques principales:
--
--                      ->DivFrec: Un divisor de frecuencia. Dividir□a
frecuencia
--                      de reloj tantas veces como indique gFrecClk.
--                      ->Control: Una m□ ina de estados finitos.
--                      ->Carga_desplaz: Un registro de carga en
paralelo.
--                      ->Selecci□Un multiplexor que selecciona la
se□de salida
--                      segn el estado actual. Este multiplexor
termina en un biestable
--                      para evitar pulsos no deseados, ya que su
salida (DatoSerieOut)
--                      es la salida del circuito.
--
----------------------------------------------------------------------
------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;



----------------------------------------------------------------------
------------

----------------------------------------------------------------------
------------
--  PUERTO      NBits      INFO
--
--  RstN           1    Entrada Se□de reset as□rono (activo a 0)
--  Clk            1    Entrada Se□de reloj de la placa, en principio
de 100MHz,
--                      pero configurable por gFrecClk
--  Transmite      1    Entrada Se□del sistema que ordena al m□o la
transmisi□span>
--                      del dato que se encuentra en DatoTxIn.
--  DatoTxIn       8    Entrada El dato a enviar. Se proporciona de
manera simult□ a
 --                     cuando Transmite = 1
--  Transmitiendo  1    Salida  Se□del sistema que indica que en ese
instante se est˜Hspan>
--                      transmitiendo un dato.
--  DatoSerieOut   1    Salida  Trama de datos que se env▪al PC y que
sigue el
--                      protocolo RS232
```

```
--
----------------------------------------------------------------------------
------------
entity Tx_serie is
--
 generic (
                gFrecClk : integer := 50000000;
                gBaud : integer := 19200
            );
    Port ( RstN : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           Transmite : in  STD_LOGIC;
           DatoTxIn : in  STD_LOGIC_VECTOR (7 downto 0);
           Transmitiendo : out  STD_LOGIC;
           DatoSerieOut : out  STD_LOGIC);
end Tx_serie;

----------------------------------------------------------------------------
------------

architecture Behavioral of Tx_serie is


   -- Tipo nuevo para los estados --

    type t_Estado is (eBitInit,eBitsDato,eBitFin,eInit);

    constant cFinCuenta : natural := (gFrecClk/gBaud)-1;

    -- Se□s auxiliares

    signal Cuenta,CuentaBits: integer;
    signal ClkBaud,EnableCont,Dsplza,FinDsplza8bits,CargaDato :
std_logic;
    signal Estado : t_Estado;
    signal Registro: STD_LOGIC_VECTOR (7 downto 0);
    signal SalReg: std_logic;
    signal MiDatoTxIn:STD_LOGIC_VECTOR (7 downto 0);
    signal SalidaSeleccion:std_logic;

begin

----------------------------------------------------------------------------
------------
    -- A partir del reloj de la placa de 100 MHz (Clk), queremos
proporcionar una
    -- se□con frecuencia de 9600 Hz (ClkBaud). Este reloj tendr□or
tanto un
    -- periodo de 104,167 s, y estar□ 1 durante un solo ciclo de
reloj, estando
    -- el resto de tiempo a 0.

    P_DivFrec: Process (RstN, Clk)
    begin
            if RstN = '0' then
                Cuenta <= 0;
                ClkBaud <= '0';
```

```
            elsif Clk'event and Clk='1' then
                if EnableCont = '1' then
                    if Cuenta = cFinCuenta then
                        Cuenta <= 0;
                        ClkBaud <= '1';
                    else
                        Cuenta <= Cuenta + 1;
                        ClkBaud <= '0';
                    end if;
                end if;
            end if;
    end process;
-------------------------------------------------------------------------
------------

    -- Proceso encargado de contar los 8 bits de datos.
    -- Solo cuenta cuando estamos enviando datos, es decir en el
estado eBitsDato

    P_CuentaBits: Process (RstN, Clk)
    begin
        if RstN = '0' then
            CuentaBits <= 0;
        elsif Clk'event and Clk='1' then
            if Estado = eBitsDato then
                if ClkBaud = '1' then
                    if CuentaBits = 7 then
                        CuentaBits <= 0;
                    else
                        CuentaBits <= CuentaBits + 1;
                    end if;
                end if;
            else
                CuentaBits <= 0;
            end if;
        end if;
    end process;

-------------------------------------------------------------------------
------------

    -- Proceso que controla las se□s internas necesarias para el
    -- registro de desplazamiento y para el contador de los 8 bits de
datos

    POut: Process (Estado, Transmite, ClkBaud)
    begin
        Dsplza <= '0';
        CargaDato <= '0';
        EnableCont <= '1';

        case Estado is
            when eInit =>
                EnableCont <= '0';
                Transmitiendo <= '0';
                if Transmite = '1' then
                    CargaDato <= '1';
```

```
                EnableCont <= '1';
            end if;
        when eBitInit =>    Transmitiendo <= '1';
        when eBitsDato => Transmitiendo <= '1';
            if ClkBaud = '1' then
                Dsplza <= '1';
            end if;
        when eBitFin =>
            Transmitiendo <= '1';
            if ClkBaud = '1' then
                EnableCont <= '0';

            end if;
        end case;
    end process;
-------------------------------------------------------------------------
------------

-- Proceso de la m□ ina de estados
-- eInit:    Es el estado inicial. El sistema est□n reposo esperando
la orden de
--           transmitir. Cuando la se□Transmite se ponga a 1, se
pasar□ enviar el
--           bit de inicio, pasando para ello al siguiente estado
(eBitInit). En ese momento
--           se dar□a orden de cargar el dato (DatoTxIn) en el
registro de desplazamiento.
--           Tambi鯡tendremos que sincronizar el contador del divisor
de frecuencia; para
--           esto haremos que en el estado inicial no cuente, y en el
resto se habilite el
--           contador.
-- eBitInit: En este estado se est□nviando el bit de inicio. Se
saldr□e este estado
--           al recibir un pulso de ClkBaud, que nos dir□ue debemos
pasar a enviar los bits
--           de dato. El siguiente estado es eBitsDato.
-- eBitsDato:Este estado se encarga de enviar los 8 bits del dato.
Utilizando un contador,
--           llevaremos la cuenta del nmero de bits que se han
enviado. Cuando se hayan
--           enviado los 8 bits es decir, cuando hayan llegado 8
pulsos de Clkbaud-  se
--           activar□a se□FinDsplza8bits que har□ue cambiemos al
siguiente estado
--           (eBitFin).
-- eBitFin:  Este estado env"el bit de fin. Al llegar el siguiente
pulso de ClkBaud,
--           cambiaremos al estado inicial eInit.


    P_Control_FSM: Process (RstN, Clk)
    begin
        if RstN = '0' then
            Estado <= eInit;
        elsif Clk'event and Clk='1' then
```

```
            case Estado is
                when eInit =>
                    if Transmite = '1' then
                        Estado <= eBitInit;
                    end if;
                when eBitInit =>
                    if ClkBaud = '1' then
                        Estado <= eBitsDato;
                    end if;
                when eBitsDato =>
                    if FinDsplza8bits = '1' then
                        Estado <= eBitFin;
                    end if;
                when eBitFin =>
                    if ClkBaud = '1' then
                        Estado <= eInit;
                    end if;
            end case;
        end if;
    end process;
-------------------------------------------------------------------------
------------

    -- Proceso que se encarga de ir poniendo en SalReg el bit del
registro
    -- de desplazamiento concecreto.

    carga_desplazamiento: process(Dsplza,CargaDato)
    begin
        if CargaDato='1' then
            Registro <= DatoTxIn;
        else
            SalReg <= Registro(CuentaBits);
        end if;
    end process;

-------------------------------------------------------------------------
------------
    -- Proceso encargado de elegir la salida en funci□el Estado en el
que
    -- estemos

    seleccion: process(Estado)
        begin
        if Estado = eInit then
            SalidaSeleccion <= '1';
        elsif Estado = eBitInit then
            SalidaSeleccion <= '0';
        elsif Estado = eBitsDato then
            SalidaSeleccion <= SalReg;
        elsif Estado = eBitFin then
            SalidaSeleccion <= '1';
        end if;
        --  end if;
    end process;

    DatoSerieOut <= SalidaSeleccion;
```

```
    FinDsplza8bits <= '1' when CuentaBits=7 and ClkBaud = '1' else
'0';


end Behavioral;
```

B. Ficheros sofware

B.1. Main.c

```
#include "libSI.h"

#include <xstatus.h>
#include <xbasic_types.h>

//*****************************************************************
int partial(int nWord, int nBit, int rubish, int initWord);

//VARIABLES
volatile unsigned int *dato, *respuesta;
volatile unsigned int error;
volatile unsigned int comando, num_entradas_tb, recibidos, enviados =
0;
volatile unsigned int ciclo_error = 0;
volatile unsigned int i, j, k, m, u = 0;
volatile unsigned int frame_inicio, frame_fin, sliceY0, sliceY1,
palabra, bit= 0;
volatile unsigned int reconfiguracion_correcta, bit_parcial_correcto =
0;
volatile unsigned int tam_bit, tam_bit_modificado, tam_bit_restorer =
0;
volatile unsigned int dato_syn = 0;

XStatus status;

XHwIcap icap;

int main (void) {
    dato_syn = 204;

    inicializarUart();
    inicializarIcap(&icap);

    while(1){
        recibirDato(dato);
        comando = *dato >> 29;

        switch(comando){
            //Cargar testbench en RAM
            case 0x00000000 :
                num_entradas_tb = *dato;
                recibidos = 0;
                while(recibidos<num_entradas_tb){
```

```
                    recibirDato(pto_ram_testbench + recibidos);
                    recibidos = recibidos + 1;
                }
                break;

            //Generar Golden
            case 0x00000001 :
                for(i=0;i<num_entradas_tb;i++){
                    *(pto_circuito_entrada) = *(pto_ram_testbench+i);
                    *(pto_ram_golden+i) = *(pto_circuito_salida);
                }
                break;

            //Ejecutar
            case 0x00000002 :
                ejecutar();
                enviarErrores();
                break;

            //Pasar los datos del golden a la salida serie
            case 0x00000003 :
                enviados = 0;
                while (enviados<num_entradas_tb){
                    enviarDato(pto_ram_golden + enviados);
                    enviados = enviados+1;
                }
                break;
            //Cargar un golden por la entrada serie
            case 0x00000004 :
                recibidos = 0;
                while(recibidos<num_entradas_tb){
                    recibirDato(pto_ram_golden + recibidos);
                    recibidos = recibidos + 1;
                }
                break;
            //Reconfiguraci□arcial
            case 0x00000005 :
                tam_bit = (*dato<<3)>>3; // Con esto elimino los 3
primeros bits correspondientes al comando

                recibirDato(&frame_inicio);
                recibirDato(&frame_fin);
                recibirDato(&sliceY0);
                recibirDato(&sliceY1);

                //Cargar en RAM el .bit recibido por la uart
                recibidos = 0;
                while(recibidos<tam_bit){
                    recibirDato(pto_ram_bitOriginal + recibidos);
                    recibidos = recibidos + 1;
                }

                reconfiguracion();

                break;
            //Comprobar la comunicaci□ntre FPGA y Nessy
            case 0x00000006 :
```

```
                *respuesta=0xf0f0f0f0;
                enviarDato(respuesta);
            default :
                break;
        }
    }
   return 0;
}

ejecutar(){
    ciclo_error = 0;
    error = 0x00000000;
    m=0;
    while(m < num_entradas_tb & error == 0x00000000){
        *(pto_circuito_entrada) = *(pto_ram_testbench+m);
        error = (*(pto_circuito_salida)) ^ (*(pto_ram_golden+m));
        if (error != 0x00000000)
            ciclo_error = m + 1;
        m = m + 1;
    }
}

enviarErrores(){
    enviarDato(&ciclo_error);
}

reconfiguracion(){
    //VARIABLES AUXILIARES
    volatile unsigned int palabra_inicio, gap_inicio,
palabra_gap_inicio, bit_inicio, gap_fin, palabra_gap_fin, bit_fin,
bit_fin_aux, palabra_inicial = 0;

    palabra_inicial = (frame_inicio * 206) + 1;
    for(i = frame_inicio; i < frame_fin ; i++){
        //calculos
        palabra_inicio = (i * 206) + 1;
        gap_inicio = (((159 - sliceY1) / 2) * 80) / 32;
        palabra_gap_inicio = palabra_inicio + 3 + gap_inicio;
        bit_inicio = (32 - ((((159 - sliceY1) / 2) * 80) % 32)) - 1;
        gap_fin = ((((159 - sliceY0) / 2) + 1) * 80) / 32;
        palabra_gap_fin = palabra_inicio + 3 + gap_fin;
        bit_fin = 32 - (((((159 - sliceY0) / 2) + 1) * 80) % 32);
        bit_fin_aux = 0;

        for(j = palabra_gap_inicio; j <= palabra_gap_fin; j++){
            if (j == palabra_gap_fin){
                bit_fin_aux = bit_fin;
            }
            for(k = 0; k <= 31; k++){

                //CREAR BIT ERRONEO Y BIT RESTORER
                partial(j,k,0,palabra_inicial);

                //CARGAR BIT ERRONEO
                programar_icap(&icap , (Xuint32
*)pto_ram_bitModificado,(Xuint32)tam_bit_modificado);
```

```
                //Ejecutar el testbench
                ejecutar();
                enviarDato(&dato_syn);
                enviarErrores();

                //CARGAR BIT RESTORER
                programar_icap(&icap , (Xuint32
*)pto_ram_bitRestorer,(Xuint32)tam_bit_restorer);

            }
        }
    }
}



//******************* CѣIGO DE LA RECONFIGURACIѪ PARCIAL
**********************
#define SYNC_WORD               0xAA995566
#define NOOP                    0x20000000
#define PACKET_MASK             0x60000000
#define TYPE_1_WORD             0x20000000
#define TYPE_2_WORD             0x40000000
#define READ_WRITE_MASK         0x18000000
#define TYPE_READ               0x08000000
#define TYPE_WRITE              0x10000000
#define REG_MASK                0x0003E000
#define CRC                     0x00000000
#define FAR                     0x00002000
#define FDRI                    0x00004000
#define FDRO                    0x00006000
#define CMD                     0x00008000
#define CMD_MASK                0x0000000F
#define CTL                     0x0000A000
#define MASK                    0x0000C000
#define STAT                    0x0000E000
#define LOUT                    0x00010000
#define COR                     0x00012000
#define MFWR                    0x00014000
#define FLR                     0x00016000
#define KEY                     0x00018000
#define CBC                     0x0001A000
#define IDCODE                  0x0001C000
#define WORD_COUNT_1            0x000007FF
#define WORD_COUNT_2            0x07FFFFFF
#define INST_NO_CRC             0x0000DEFC
#define TOTAL_FRAMES            1756
#define TOTAL_FRAME_LENGTH      206
#define N_GCLK_COLUMNS          1
#define N_GCLK_FRAMES_PER_COLUMN    4
#define N_IOB_COLUMNS               2
#define N_IOB_FRAMES_PER_COLUMN     4
#define N_IOI_COLUMNS               2
#define N_IOI_FRAMES_PER_COLUMN     22
#define N_CLB_COLUMNS               46
#define N_CLB_FRAMES_PER_COLUMN     22
#define N_BRAM_COLUMNS              8
```

```
#define N_BRAM_FRAMES_PER_COLUMN    64
#define N_BRAM_INTERCONNECT_COLUMNS     8
#define N_BRAM_INTERCONNECT_FRAMES_PER_COLUMN   22



int partial(int nWord, int nBit, int rubish, int initWord) //, int *
pto_ram_bitOriginal,int tamBit,int * pto_ram_bitModificado,int *
pto_ram_bitRestorer)
    {
    volatile int index = 0;
    volatile int indexAux = 0;
    volatile int index_mod = 0;
    volatile int index_rest = 0;
    volatile unsigned int instruccion, instruccionAux;
    volatile int
numWords,numWordsAux,iniWord,auxWord,nFrame,BA,MJA,MNA;
    volatile int sync = 0;

    while ((!sync) && (index != tam_bit))
        {
        instruccion = *(pto_ram_bitOriginal + index);
        if (instruccion != SYNC_WORD)
            {
            if (rubish)
                {
                *(pto_ram_bitModificado + index_mod) = instruccion;
                *(pto_ram_bitRestorer + index_rest) = instruccion;
                index_mod ++;
                index_rest ++;
                }
            index ++;
            }
        else
            sync = 1;
        }

    while (index < tam_bit)
        {
        instruccion = *(pto_ram_bitOriginal + index);
        if (instruccion == SYNC_WORD)
            {
            *(pto_ram_bitModificado + index_mod) = instruccion;
            *(pto_ram_bitRestorer + index_rest) = instruccion;
            index_mod ++;
            index_rest ++;
            }
        else if (instruccion == NOOP)
            {
            *(pto_ram_bitModificado + index_mod) = instruccion;
            *(pto_ram_bitRestorer + index_rest) = instruccion;
            index_mod ++;
            index_rest ++;
            }
        else if ((instruccion & PACKET_MASK) == TYPE_1_WORD)
            {
```

```
            if ((instruccion & READ_WRITE_MASK) == TYPE_WRITE)
                {
                switch (instruccion & REG_MASK)
                    {
                    case CRC :
                        numWords = instruccion & WORD_COUNT_1;
                        *(pto_ram_bitModificado + index_mod) =
instruccion;
                        *(pto_ram_bitRestorer + index_rest) =
instruccion;
                        index ++;
                        index_mod ++;
                        index_rest++;
                        if (index == tam_bit)
                            return -1;
                        instruccion = *(pto_ram_bitOriginal + index);
                        if (numWords == 0)
                            {
                            if ((instruccion & PACKET_MASK) !=
TYPE_2_WORD)
                                {
                                index --;
                                break;
                                }
                            else
                                {
                                numWords = instruccion & WORD_COUNT_2;
                                *(pto_ram_bitModificado + index_mod) =
instruccion;
                                *(pto_ram_bitRestorer + index_rest) =
instruccion;
                                index ++;
                                index_mod ++;
                                index_rest ++;
                                if (index == tam_bit)
                                    return -1;
                                instruccion = *(pto_ram_bitOriginal +
index);
                                }
                            }
                        for (u = 1 ; u <= numWords ; u++)
                            {
                            instruccion = INST_NO_CRC;
                            *(pto_ram_bitModificado + index_mod) =
instruccion;
                            *(pto_ram_bitRestorer + index_rest) =
instruccion;
                            index ++;
                            index_mod ++;
                            index_rest ++;
                            if (index == tam_bit)
                                return -1;
                            instruccion = *(pto_ram_bitOriginal +
index);
                            }
                        index --;
                        break;
```

```
                    case FAR :
                        numWords = instruccion & WORD_COUNT_1;
                        *(pto_ram_bitModificado + index_mod) =
instruccion;
                        *(pto_ram_bitRestorer + index_rest) =
instruccion;
                        index ++;
                        index_mod ++;
                        index_rest ++;
                        if (index == tam_bit)
                            return -1;
                        instruccion = *(pto_ram_bitOriginal + index);
                        if (numWords == 0)
                            {
                            if ((instruccion & PACKET_MASK) !=
TYPE_2_WORD)
                                {
                                index --;
                                break;
                                }
                            else
                                {
                                numWords = instruccion & WORD_COUNT_2;
                                *(pto_ram_bitModificado + index_mod) =
instruccion;
                                *(pto_ram_bitRestorer + index_rest) =
instruccion;
                                index ++;
                                index_mod ++;
                                index_rest ++;
                                if (index == tam_bit)
                                    return -1;
                                instruccion = *(pto_ram_bitOriginal +
index);
                                }
                            }
                        nFrame = (nWord - 1) / TOTAL_FRAME_LENGTH;
                        for (u = 1 ; u <= numWords ; u++)
                            {
                            BA = 0;
                                    MJA = 0;
                            MNA = nFrame - (N_GCLK_COLUMNS *
N_GCLK_FRAMES_PER_COLUMN);
                                    if(MNA < 0)
                                        MNA = MNA + (N_GCLK_COLUMNS *
N_GCLK_FRAMES_PER_COLUMN);
                                    else
                                        {
                                        MNA = MNA - (N_IOB_COLUMNS/2 *
N_IOB_FRAMES_PER_COLUMN);
                                        if(MNA < 0)
                                                {
                                                MJA = 1;
                                                MNA = MNA +
(N_IOB_COLUMNS/2 * N_IOB_FRAMES_PER_COLUMN);
                                                }
                                        else
```

```
                                                    {
                                                    MNA = MNA -
(N_IOI_COLUMNS/2 * N_IOI_FRAMES_PER_COLUMN);
                                                    if(MNA < 0)
                                                        {
                                                        MJA = 2;
                                                        MNA = MNA +
(N_IOI_COLUMNS/2 * N_IOI_FRAMES_PER_COLUMN);
                                                        }
                                                    else
                                                        {
                                                        if(MNA /
(N_CLB_COLUMNS * N_CLB_FRAMES_PER_COLUMN) == 0)
                                                                {
                                                                MJA = (MNA
/ N_CLB_FRAMES_PER_COLUMN) + N_GCLK_COLUMNS +
N_IOB_COLUMNS/2 + N_IOI_COLUMNS/2;
                                                                MNA = MNA
% N_CLB_FRAMES_PER_COLUMN;
                                                                }
                                                            else
                                                                {
                                                                MNA = MNA
- (N_CLB_COLUMNS * N_CLB_FRAMES_PER_COLUMN) -
(N_IOI_COLUMNS/2 * N_IOI_FRAMES_PER_COLUMN);
                                                                MJA =
N_GCLK_COLUMNS + N_IOB_COLUMNS/2 + N_IOI_COLUMNS/2 +
N_CLB_COLUMNS;
                                                                if(MNA <
0)
                                                                    MNA =
MNA + (N_IOI_COLUMNS/2 * N_IOI_FRAMES_PER_COLUMN);
                                                                else
                                                                    {
                                                                    MNA =
MNA - (N_IOB_COLUMNS/2 * N_IOB_FRAMES_PER_COLUMN);
                                                                    MJA =
MJA + 1;
                                                                    if(MNA
< 0)
MNA = MNA + (N_IOB_COLUMNS/2 *
N_IOB_FRAMES_PER_COLUMN);
                                                                    else
{
BA = 1;
MJA = 0;
if(MNA / (N_BRAM_COLUMNS *
```

```
N_BRAM_FRAMES_PER_COLUMN) == 0)

{

MJA = MNA / N_BRAM_FRAMES_PER_COLUMN;

MNA = MNA % N_BRAM_FRAMES_PER_COLUMN;

}

else

{

BA = 2;

MNA = MNA - (N_BRAM_COLUMNS *

N_BRAM_FRAMES_PER_COLUMN);

MJA = MNA /

N_BRAM_INTERCONNECT_FRAMES_PER_COLUMN;

MNA = MNA %

N_BRAM_INTERCONNECT_FRAMES_PER_COLUMN;

}

}
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                BA = BA << 25;
                                MJA = MJA << 17;
                                MNA = MNA << 9;
                                instruccion = BA | MJA | MNA;
                                *(pto_ram_bitModificado + index_mod) =
instruccion;
                                *(pto_ram_bitRestorer + index_rest) =
instruccion;
                                index ++;
                                index_mod ++;
                                index_rest ++;
                                if (index == tam_bit)
                                    return -1;
                                instruccion = *(pto_ram_bitOriginal +
index);
                                }
                            index --;
                            break;
                        case FDRI :
                            numWords = instruccion & WORD_COUNT_1;
```

```
                        instruccionAux = instruccion;
                        *(pto_ram_bitModificado + index_mod) =
instruccion;
                        *(pto_ram_bitRestorer + index_rest) =
instruccion;
                        index ++;
                        index_mod ++;
                        index_rest ++;
                        if (index == tam_bit)
                            return -1;
                        instruccion = *(pto_ram_bitOriginal + index);
                        if (numWords == 0)
                            {
                            if ((instruccion & PACKET_MASK) !=
TYPE_2_WORD)
                                {
                                index --;
                                break;
                                }
                            else
                                {
                                index_mod --;
                                index_rest --;
                                numWords = instruccion & WORD_COUNT_2;
                                instruccionAux = instruccionAux |
(TOTAL_FRAME_LENGTH * 2);
                                *(pto_ram_bitModificado + index_mod) =
instruccionAux;
                                *(pto_ram_bitRestorer + index_rest) =
instruccionAux;
                                index ++;
                                index_mod ++;
                                index_rest ++;
                                if (index == tam_bit)
                                    return -1;
                                instruccion = *(pto_ram_bitOriginal +
index);
                                }
                            }
                        int wordGap = (nWord -1) / TOTAL_FRAME_LENGTH;
                        wordGap = wordGap - ((initWord - 1) /
TOTAL_FRAME_LENGTH);
                        wordGap = wordGap * TOTAL_FRAME_LENGTH;
                        indexAux = index;
                        index = index + wordGap;
                        for (u = initWord + wordGap ; u < (initWord +
wordGap + TOTAL_FRAME_LENGTH) ; u++)
                            {
                            instruccion = *(pto_ram_bitOriginal +
index);
                            *(pto_ram_bitRestorer + index_rest) =
instruccion;
                            if(nWord == u)
                                instruccion = instruccion ^
(0x00000001 << nBit);
                            *(pto_ram_bitModificado + index_mod) =
instruccion;
```

```
                            index_mod ++;
                            index_rest ++;
                            index ++;
                            if (index == tam_bit)
                                return -1;
                            }
                        index = indexAux + (numWords -
TOTAL_FRAME_LENGTH);
                        for (u = 1 ; u <= TOTAL_FRAME_LENGTH ; u++)
                            {
                            instruccion = *(pto_ram_bitOriginal +
index);
                            *(pto_ram_bitRestorer + index_rest) =
instruccion;
                            *(pto_ram_bitModificado + index_mod) =
instruccion;
                            index_mod ++;
                            index_rest ++;
                            index ++;
                            if (index == tam_bit)
                                return -1;
                            }
                        instruccion = INST_NO_CRC;
                        *(pto_ram_bitModificado + index_mod) =
instruccion;
                        *(pto_ram_bitRestorer + index_rest) =
instruccion;
                        index_mod ++;
                        index_rest ++;
                        break;
                    case FDRO :
                        break;
                    case CMD :
                        numWords = instruccion & WORD_COUNT_1;
                        *(pto_ram_bitModificado + index_mod) =
instruccion;
                        *(pto_ram_bitRestorer + index_rest) =
instruccion;
                        index ++;
                        index_mod ++;
                        index_rest ++;
                        if (index == tam_bit)
                            return -1;
                        instruccion = *(pto_ram_bitOriginal + index);
                        if (numWords == 0)
                            {
                            if ((instruccion & PACKET_MASK) !=
TYPE_2_WORD)
                                {
                                index --;
                                break;
                                }
                            else
                                {
                                numWords = instruccion & WORD_COUNT_2;
                                *(pto_ram_bitModificado + index_mod) =
instruccion;
```

```
                                    *(pto_ram_bitRestorer + index_rest) =
instruccion;
                                    index ++;
                                    index_mod ++;
                                    index_rest ++;
                                    if (index == tam_bit)
                                        return -1;
                                    instruccion = *(pto_ram_bitOriginal +
index);
                                    }
                                }
                            numWordsAux = numWords;
                            for (u = 1 ; u <= numWords ; u++)
                                {
                                if (((instruccion & CMD_MASK) != 5) &&
((instruccion & CMD_MASK) != 9) &&
                                    ((instruccion & CMD_MASK) != 10))
                                    {
                                    *(pto_ram_bitModificado + index_mod) =
instruccion;
                                    *(pto_ram_bitRestorer + index_rest) =
instruccion;
                                    index ++;
                                    index_mod ++;
                                    index_rest ++;
                                    if (index == tam_bit)
                                        return -1;
                                    }
                                else
                                    {
                                    index ++;
                                    if (index == tam_bit)
                                        return -1;
                                    index_mod --;
                                    index_rest --;
                                    numWordsAux --;
                                    if (numWordsAux == 0)
                                        {
                                        index --;
                                        break;
                                        }
                                    instruccion = *(pto_ram_bitModificado
+ index_mod);
                                    if ((instruccion & PACKET_MASK) ==
TYPE_2_WORD)
                                        {
                                        instruccion = instruccion &
(!WORD_COUNT_2);
                                        instruccion = instruccion |
numWordsAux;
                                        *(pto_ram_bitModificado +
index_mod) = instruccion;
                                        *(pto_ram_bitRestorer +
index_rest) = instruccion;
                                        index_mod ++;
                                        index_rest ++;
                                        }
```

```
                                else
                                    {
                                    instruccion = instruccion &
(!WORD_COUNT_1);
                                    instruccion = instruccion |
numWordsAux;
                                    *(pto_ram_bitModificado +
index_mod) = instruccion;
                                    *(pto_ram_bitRestorer +
index_rest) = instruccion;
                                    index_mod ++;
                                    index_rest ++;
                                    }
                                }
                            instruccion = *(pto_ram_bitOriginal +
index);
                            }
                        index --;
                        break;
                    case CTL :
                    case MASK :
                    case FLR :
                        numWords = instruccion & WORD_COUNT_1;
                        index ++;
                        if (index == tam_bit)
                            return -1;
                        instruccion = *(pto_ram_bitOriginal + index);
                        if (numWords == 0)
                            {
                            if ((instruccion & PACKET_MASK) !=
TYPE_2_WORD)
                                {
                                index --;
                                break;
                                }
                            else
                                {
                                numWords = instruccion & WORD_COUNT_2;
                                index ++;
                                if (index == tam_bit)
                                    return -1;
                                }
                            }
                        for (u = 1 ; u <= numWords ; u++)
                            {
                            index ++;
                            if (index == tam_bit)
                                return -1;
                            }
                        index --;
                        break;
                    case STAT :
                        break;
                    case LOUT :
                    case MFWR :
                    case KEY :
                    case CBC :
```

```
                    case IDCODE :
                        numWords = instruccion & WORD_COUNT_1;
                        *(pto_ram_bitModificado + index_mod) =
instruccion;
                        *(pto_ram_bitRestorer + index_rest) =
instruccion;
                        index ++;
                        index_mod ++;
                        index_rest ++;
                        if (index == tam_bit)
                            return -1;
                        instruccion = *(pto_ram_bitOriginal + index);
                        if (numWords == 0)
                            {
                            if ((instruccion & PACKET_MASK) !=
TYPE_2_WORD)
                                {
                                index --;
                                break;
                                }
                            else
                                {
                                numWords = instruccion & WORD_COUNT_2;
                                *(pto_ram_bitModificado + index_mod) =
instruccion;
                                *(pto_ram_bitRestorer + index_rest) =
instruccion;
                                index ++;
                                index_mod ++;
                                index_rest ++;
                                if (index == tam_bit)
                                    return -1;
                                instruccion = *(pto_ram_bitOriginal +
index);
                                }
                            }
                        for (u = 1 ; u <= numWords ; u++)
                            {
                            *(pto_ram_bitModificado + index_mod) =
instruccion;
                            *(pto_ram_bitRestorer + index_rest) =
instruccion;
                            index ++;
                            index_mod ++;
                            index_rest ++;
                            if (index == tam_bit)
                                return -1;
                            instruccion = *(pto_ram_bitOriginal +
index);
                            }
                        index --;
                        break;
                    case COR :
                        numWords = instruccion & WORD_COUNT_1;
                        *(pto_ram_bitModificado + index_mod) =
instruccion;
```

```
                        *(pto_ram_bitRestorer + index_rest) =
instruccion;
                        index ++;
                        index_mod ++;
                        index_rest ++;
                        if (index == tam_bit)
                            return -1;
                        instruccion = *(pto_ram_bitOriginal + index);
                        if (numWords == 0)
                            {
                            if ((instruccion & PACKET_MASK) !=
TYPE_2_WORD)
                                {
                                index --;
                                break;
                                }
                            else
                                {
                                numWords = instruccion & WORD_COUNT_2;
                                *(pto_ram_bitModificado + index_mod) =
instruccion;
                                *(pto_ram_bitRestorer + index_rest) =
instruccion;
                                index ++;
                                index_mod ++;
                                index_rest ++;
                                if (index == tam_bit)
                                    return -1;
                                instruccion = *(pto_ram_bitOriginal +
index);
                                }
                            }
                        for (u = 1 ; u <= numWords ; u++)
                            {
                            instruccion = instruccion | (0x00000001 <<
29);
                            *(pto_ram_bitModificado + index_mod) =
instruccion;
                            *(pto_ram_bitRestorer + index_rest) =
instruccion;
                            index ++;
                            index_mod ++;
                            index_rest ++;
                            if (index == tam_bit)
                                return -1;
                            instruccion = *(pto_ram_bitOriginal +
index);
                            }
                        index --;
                        break;
                    }
                }
            else if ((instruccion & READ_WRITE_MASK) == TYPE_READ)
                {
                numWords = instruccion & WORD_COUNT_1;
                *(pto_ram_bitModificado + index_mod) = instruccion;
                *(pto_ram_bitRestorer + index_rest) = instruccion;
```

```
                index ++;
                index_mod ++;
                index_rest++;
                if (index == tam_bit)
                    return -1;
                instruccion = *(pto_ram_bitOriginal + index);
                if (numWords == 0)
                    {
                    if ((instruccion & PACKET_MASK) != TYPE_2_WORD)
                        {
                        index --;
                        break;
                        }
                    else
                        {
                        numWords = instruccion & WORD_COUNT_2;
                        *(pto_ram_bitModificado + index_mod) =
instruccion;
                        *(pto_ram_bitRestorer + index_rest) =
instruccion;
                        index ++;
                        index_mod ++;
                        index_rest ++;
                        if (index == tam_bit)
                            return -1;
                        instruccion = *(pto_ram_bitOriginal + index);
                        }
                    }
                for (u = 1 ; u <= numWords ; u++)
                    {
                    *(pto_ram_bitModificado + index_mod) =
instruccion;
                    *(pto_ram_bitRestorer + index_rest) = instruccion;
                    index ++;
                    index_mod ++;
                    index_rest ++;
                    if (index == tam_bit)
                        return -1;
                    instruccion = *(pto_ram_bitOriginal + index);
                    }
                index --;
                }
            }
        index ++;
        }
    tam_bit_modificado = index_mod;
    tam_bit_restorer = index_rest;
    return 1;
    }
```

B.2. LibSI.c

```
#include "libSI.h"
```

```
/**************************** Include Files
*****************************/
#include <xbasic_types.h>
#include <xstatus.h>

//*********************************************************************
*
// FUNCIONES PARA EL ADAPTADOR_UART - ENVIAR/RECIBIR POR PUERTO SERIE

void enviarDato(volatile unsigned int *dato){
    while(*pto_uart_salida_ocupada == 0xFFFFFFFF)
        ;
    *pto_uart_salida = *dato;
}

void recibirDato(volatile unsigned int *dato){
    while(*pto_uart_entrada_lista != 0xFFFFFFFF)
        ;
    *dato = *pto_uart_entrada;
    *pto_uart_entrada_lista = 0x00000000;
}

void inicializarUart(){
    *pto_uart_entrada_lista = 0x00000000;
}

//*********************************************************************
*
// FUNCIONES PARA EL ICAP

void inicializarIcap(XHwIcap * icap){
    XHwIcap_Initialize(icap, XPAR_OPB_HWICAP_0_DEVICE_ID,
XHI_XC2VP30);
}

void programar_icap(XHwIcap * icap, Xuint32 *data, Xuint32 tamano){
    XStatus status = XHwIcap_SetConfiguration(icap, data, tamano);
}
```

B.3. LibSi.h

```
#ifndef LIBSI_H_
#define LIBSI_H_

    #include "xparameters.h"
    #include "xutil.h"
    #include <xhwicap.h>


    #define UART_SALIDA                 XPAR_ADAPTADOR_UART_0_BASEADDR
    #define UART_SALIDA_OCUPADA
(XPAR_ADAPTADOR_UART_0_BASEADDR + 4)
    #define UART_ENTRADA
(XPAR_ADAPTADOR_UART_0_BASEADDR + 8)
```

```
    #define UART_ENTRADA_LISTA
(XPAR_ADAPTADOR_UART_0_BASEADDR + 12)

    #define CIRCUITO_ENTRADA
XPAR_ADAPTADOR_CIRCUITO_0_BASEADDR
    #define CIRCUITO_SALIDA
(XPAR_ADAPTADOR_CIRCUITO_0_BASEADDR + 4)

    #define ICAP                        (XPAR_OPB_HWICAP_0_BASEADDR)

    #define RAM_TESTBENCH
(XPAR_DDR_512MB_64MX64_RANK2_ROW13_COL10_CL2_5_MEM0_BASEADDR +
0x00100000)
    #define RAM_GOLDEN
(XPAR_DDR_512MB_64MX64_RANK2_ROW13_COL10_CL2_5_MEM0_BASEADDR +
0x00200000)
    #define RAM_ERRORES
(XPAR_DDR_512MB_64MX64_RANK2_ROW13_COL10_CL2_5_MEM0_BASEADDR +
0x00300000)
    #define RAM_BITSTREAM_ORIGINAL
(XPAR_DDR_512MB_64MX64_RANK2_ROW13_COL10_CL2_5_MEM0_BASEADDR +
0x00400000)
    #define RAM_BITSTREAM_MODIFICADO    (0x0000E000)
    #define RAM_BITSTREAM_RESTORER      (0x0000F000)

    static volatile unsigned int * pto_uart_salida =(volatile int *)
UART_SALIDA;
    static volatile unsigned int * pto_uart_salida_ocupada = (volatile
int *) UART_SALIDA_OCUPADA;
    static volatile unsigned int * pto_uart_entrada =(volatile int *)
UART_ENTRADA;
    static volatile unsigned int * pto_uart_entrada_lista =(volatile
int *) UART_ENTRADA_LISTA;

    static volatile unsigned int * pto_circuito_entrada =(volatile int
*) CIRCUITO_ENTRADA;
    static volatile unsigned int * pto_circuito_salida =(volatile int
*) CIRCUITO_SALIDA;

    static volatile unsigned int * pto_icap =(volatile int *) ICAP;

    static volatile unsigned int * pto_ram_testbench = (volatile
unsigned int *) RAM_TESTBENCH;
    static volatile unsigned int * pto_ram_golden = (volatile unsigned
int *) RAM_GOLDEN;
    static volatile unsigned int * pto_ram_errores = (volatile
unsigned int *) RAM_ERRORES;
    static volatile unsigned int * pto_ram_bitOriginal = (volatile
unsigned int *) RAM_BITSTREAM_ORIGINAL;
    static volatile unsigned int * pto_ram_bitModificado = (volatile
unsigned int *) RAM_BITSTREAM_MODIFICADO;
    static volatile unsigned int * pto_ram_bitRestorer = (volatile
unsigned int *) RAM_BITSTREAM_RESTORER;

    void enviarDato(volatile unsigned int *dato);
    void recibirDato(volatile unsigned int *dato);
    void inicializarUart();
```

```
    void inicializarIcap(XHwIcap * icap);
    void programar_icap(XHwIcap * icap,Xuint32 *data, Xuint32 tamano);

#endif /*LIBSI_H_*/
```

C. Ficheros código de la Aplicación

C.1. Paquete Nessy

C.1.1. Main.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package nessy20;

import javax.swing.UIManager;

/**
 * Clase principal. Lanza la interfaz gráfica
 *
 * @author Tony, David y Carlos
 */
public class Main {

    /** Ejecuta la aplicación.
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //Para cambiar el lookAndFeel  de la aplicacion
        //Windows
        for(UIManager.LookAndFeelInfo
laf:UIManager.getInstalledLookAndFeels()){
            if("Windows".equals(laf.getName()))
                try {
                UIManager.setLookAndFeel(laf.getClassName());
            } catch (Exception ex) {
            }
        }

       GUIPrincipal gui;
       gui = new GUIPrincipal();
       gui.setLocationRelativeTo(null);
       gui.setVisible(true);

    }
}
```

C.1.2. GUIPrincipal.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * GUIPrincipal.java
 * Created on 01-mar-2010, 22:50:42
 */
package nessy20;

import java.util.logging.Level;
import java.util.logging.Logger;
import compiladorEntidad.Errores;
import compiladorEntidad.SintacticoEntidad;
import compiladorEntidad.Entidad;
import IOFPGA.EntradaSalida;
import generadorVHDL.GeneraVhdl_V2;
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStreamWriter;
import java.util.ArrayList;
import java.util.Properties;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.filechooser.FileNameExtensionFilter;
import procesos.HiloProceso;
import procesos.HiloInyeccion;
import procesos.Semaforo;
/**
 * Frame principal de la aplicación
 * @author Rubén y Felipe
 */
public class GUIPrincipal extends JFrame{
    /**
     * Ruta donde se encuentra Nessy
     */
    public final static String RUTA_NESSY =
System.getProperties().getProperty("user.dir");
    /**
     * Ruta en la que se encuentra el Xilinx. Será introducida por el
usuario
```

```
     * o cargada mediante el fichero de configuración
     */
    public static String RUTA_ISE;
    /**
     * Ruta en la que se encuentra el EDK
     */
    public static String RUTA_EDK;
    /**
     * Ruta del proyecto NESSY
     */
    public static String RUTA_PROYECTO_ACTUAL;
    /**
     * Ruta del proyecto NESSY_EDK
     */

    public static String RUTA_PROYECTO_ACTUAL_EDK;
    /**
     * Ruta de los log que genera Nessy
     */
    public static String RUTA_PROYECTO_ACTUAL_SALIDAS;
    /*
     * Ruta donde Nessy copia los vhdl selecionados
     */
    public static String RUTA_PROYECTO_ACTUAL_VHDL;
    /**
     * Fichero VHD en el que está definida la entidad top que se desea
ejecutar.
     */
    private String ficheroEntidadVHD;
    /**
     * Conjunto de todos los ficheros VHD que pueden formar parte de
un proyecto
     * a la hora de generar un .BIT
     */
    private ArrayList<File> ficherosVHD;
    /*
     * Conjunto de todos las rutas de los vhdl que forman el proyecto
     */
    private ArrayList<String> stringsVHD;
    /**
     * Indica la ruta fichero de banco de pruebas
     */
    private String fichero_tb;
    /**
      * Indica la ruta fichero de banco de pruebas
     */
    private String fichero_golden;
    /**
     * Indica la ruta del fichero .bit que se desea cargar
     */
    private String fichero_bit;
    /**
     * enteros que determinan la zona (slices) donde ha sido colocado
nuestro circuito
     */
    private int nsliceX,nsliceY;
```

```
    /**
     * Indica el numero de ciclos del fichero_tb
     */
    private int numCiclosTB;
    /**
     * Atributo para enviar y recibir datos por la entrada salida
     */
    private EntradaSalida entsal;
    /**
     * Representación interna de la entidad que se quiere ejecutar
     */
    private Entidad entidad;
    private SintacticoEntidad compilador = null;
    /**
     * Número de fichero que representa el top dentro del conjunto de
ficheros
     */
    private int top;
    /**
     * Indica que el menú de selección top se ha cerrado
     */
    private boolean cerradoTop;
    /**
     * Indica si se ha configurado por 1ª vez el puerto serie
     */
    boolean comConfigurado = false;

////////////////////////////////////////////////////////////////////////
////

//////////////////////////////CONSTRUCTORA//////////////////////////////
////

////////////////////////////////////////////////////////////////////////
////
    public GUIPrincipal() {
        config(RUTA_NESSY+"\\conf\\config.properties",false);

        stringsVHD = new ArrayList<String>();

        if(RUTA_ISE == null || RUTA_EDK == null){
            JOptionPane.showMessageDialog(this, "No se ha podido
encontrar la ruta de Xilinx ISE o Xilinx EDK", "Error",
JOptionPane.INFORMATION_MESSAGE);
            System.exit(-1);
        }

        String ruta = System.getProperty( "java.class.path" );
        ruta=ruta.substring(0, ruta.length()-12);
        new File(ruta+"conf").mkdir();

        initComponentsAux();
        initComponents();

        this.ficherosVHD = new ArrayList<File>();
    }
```

```
////////////////////////////////////////////////////////////////////////
////
    ////////////////////////METODOS MODIFICAR PARAMETROS
////////////////////////

////////////////////////////////////////////////////////////////////////
////
    /**
     * Devuelve el Array de ficheros Vhdl Cargados en la aplicación.
     * @return Valor del atributo ficherosVHD
     */
    public ArrayList<File> getFiles() {
        return ficherosVHD;
    }
    /**
     * Establece el fichero Top
     * @param top Índice del archivo TOP.
     */
    public void setTop(int top) {
        this.top = top;
    }
    /**
     * Devuelve el objeto EntradaSalida para comunicarse con el
exterior
     */
    public EntradaSalida getEntradaSalida(){
        return entsal;
    }
    /**
     * Devuelve la Entidad con la que se está trabajado
     * @return Entidad con la que se está trabajando.
     */

    public Entidad getEntidad() {
        return entidad;
    }
    /**
     * Establece el campo cerradoTop, con el valor del argumento de la
función.
     * @param cerradoTop Nuevo valor de tipo boolean de cerradoTop.
     */
    public void setCerradoTop(boolean cerradoTop) {
        this.cerradoTop = cerradoTop;
    }
    /**
     * Devuelve la ruta en la que se encuentra el fichero .BIT elegido
por el
     * usuario
     * @return El valor del atributo fichero_bit
     */
    public String getFichero_bit() {
        return fichero_bit;
    }
    /**
```

```
     * @return String con la ruta donde se colocan los ficheros de
salida de la aplicacion
     */
    public String getRutaSalidas(){
        return this.RUTA_PROYECTO_ACTUAL_SALIDAS;
    }




////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS CARGAR VHDL
////////////////////////////////

////////////////////////////////////////////////////////////////////////
/////
    /**
     * Carga varios ficheros VHDL de los cuales elegirá uno como top
     */
    private void cargaVariosVHDL() {
        JFileChooser chooser;
        this._TxtEntityVHD.setText("");
        chooser = new JFileChooser();
        chooser.setMultiSelectionEnabled(true);
        Filtro filter = new Filtro("vhd");
        chooser.addChoosableFileFilter(filter);
        chooser.setCurrentDirectory(new java.io.File("."));
        chooser.setDialogTitle("Seleccionar Archivos VHDL");
        chooser.setAcceptAllFileFilterUsed(false);
        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            ArrayList<String> ficheros = new ArrayList<String>();
            File[] f = chooser.getSelectedFiles();
            ficherosVHD = new ArrayList<File>();
            for (int i = 0; i < f.length; i++) {
                ficherosVHD.add(f[i]);
            }

            for (int i = 0; i < f.length; i++) {
                ficheros.add(f[i].getName());
            }
            cerradoTop = false;
            GUISeleccionTop selTop = new GUISeleccionTop(this, true,
ficheros);
            selTop.setLocationRelativeTo(null);
            selTop.setVisible(true);
            if (!cerradoTop) {
                ficheroEntidadVHD =
ficherosVHD.get(top).getAbsolutePath(); //el fichero es el absoluto
                this.cargarVHDL();
            }

        }
    }
    /**
```

```
     * Carga un fichero VHD como top
     */
    private void cargaTopVHDL() {
        JFileChooser chooser;
        chooser = new JFileChooser();
        Filtro filter = new Filtro("vhd");
        chooser.addChoosableFileFilter(filter);
        chooser.setCurrentDirectory(new java.io.File("."));
        chooser.setDialogTitle("Seleccionar Archivo VHDL");
        chooser.setAcceptAllFileFilterUsed(false);
        ficherosVHD = new ArrayList<File>();
        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            ficherosVHD.add(chooser.getSelectedFile());
            ficheroEntidadVHD = ficherosVHD.get(0).getAbsolutePath();
            this.cargarVHDL();
        }
    }
    /**
     * Trata de compilar un fichero VHD y muestra un mensaje con lo
     * ocurrido en una ventana emergente
     */
    private void cargarVHDL() {
        boolean error = !compilarEntidad();
        if (!error) {
            seleccionaPanel(panelVHD);
            this._TxtEntityVHD.setText(this.entidad.toString());
            JOptionPane.showMessageDialog(this, "Entity cargada
correctamente", "Info", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "Error al cargar el
fichero de la entity", "Error", JOptionPane.ERROR_MESSAGE);
        }

    }
    /**
     * Compila la entidad y genera el archivo VHDL a partir de ella de
tal forma
     * que se interconecte con el módulo de entrada/salida.
     * @return Cierto si todo ha sido correcto, falso si ha habido
algún error.
     */
    public boolean compilarEntidad() {
        boolean correcto = true;
        Errores errores = new Errores();

        try {
            compilador = new SintacticoEntidad(ficheroEntidadVHD,
errores);
            compilador.inicia();

            boolean error = compilador.Entidad();
            if (!error) {
                this.entidad = compilador.getEntidad();
            } else {
                this.muestraErroresConsola(errores);
                correcto = false;
```

```
            }
        } catch (Exception e) {
            if (e.getMessage() != null) {
                errores.error(e.getMessage());
                this.muestraErroresConsola(errores);
            } else {
                e.printStackTrace();
            }
            correcto = false;

        }
        if (compilador != null) {
            compilador.cerrar();
        }
        return correcto;
    }
    /**
     * Muestra los errores obtenidos en el proceso de compilación de
la entidad
     * por la pestaña de EntityVHD.
     * @param errores
     */
    private void muestraErroresConsola(Errores errores) {
        this._TxtEntityVHD.setText("");
        for (int i = 0; i < errores.getErrores().size(); i++) {
            this._TxtEntityVHD.append(errores.getErrores().get(i) +
"\n");
        }
    }




////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS CREAR .BIT
////////////////////////////////

////////////////////////////////////////////////////////////////////////
//////
    private boolean crearBit() throws Exception{
        Semaforo sem = new Semaforo();
        boolean r = true;

        seleccionaPanel(panelCrearBit);
        this._TextCrearBit.setText(null);

        //Modificamos el .ucf para colocar adaptador_circuito en los
slices que queremos
        if(nsliceX==0 && nsliceY==0){
            GUIInyeccionErrores GuiErrores = new
GUIInyeccionErrores(this, r);
            GuiErrores.setLocationRelativeTo(null);
            GuiErrores.setVisible(true);
            nsliceX=GuiErrores.getnslicesX();
```

```
            nsliceY=GuiErrores.getnslicesY();
            BufferedWriter bw = new BufferedWriter(new
FileWriter(RUTA_PROYECTO_ACTUAL+"\\Proyecto.nessy",true));
            bw.write(String.valueOf(nsliceX)+"\n");
            bw.write(String.valueOf(nsliceY)+"\n");
            bw.close();
        }

        File download = new
File(RUTA_PROYECTO_ACTUAL+"\\download.bit");
        if(!download.exists()){
            //Copia los archivos edk
            this._TextCrearBit.append("Creando proyecto EDK........."
+ "\n");
            if(!copiarArchivos()){
                r = false;
                JOptionPane.showMessageDialog(this, "No se ha podido
copiar el Proyecto_EDK plantilla", "Error",
JOptionPane.ERROR_MESSAGE);
            }

            this._TextCrearBit.append("Modificando el .ucf para
agrupar al circuito........." + "\n");
            modificaUcf(nsliceX, nsliceY);

            //Generamos el archivo circuito.vhd
            this._TextCrearBit.append("Adaptando el proyecto
EDK........." + "\n");
            if(!generarVhdl()){
                JOptionPane.showMessageDialog(this, "No se ha podido
generar el vhdl necesario para adaptar el proyecto EDK", "Error",
JOptionPane.ERROR_MESSAGE);
                r=false;
            }

            //Modificamos el .pao del proyecto EDK
            modificaPao();
        }

        //Compilamos el proyecto y generamos el .bit sin software
        this._TextCrearBit.append("Sintetizando proyecto
EDK........." + "\n\n");
        compilarProyecto(sem);

        Thread.currentThread().sleep(1000);

        //Creamos un .bit con el software
        data2mem(sem);

        return r;
    }

    private boolean copiarArchivos() throws Exception{
        Process proc_proyecto = Runtime.getRuntime().exec("xcopy.exe
/Y /E /Q "+"\""+RUTA_NESSY+"\\Proyecto_EDK_V2\" \"" +
RUTA_PROYECTO_ACTUAL_EDK+"\"");
        proc_proyecto.waitFor();
```

```
        if(proc_proyecto.exitValue() != 0){
            return false;
        }
        Process proc_archivo = Runtime.getRuntime().exec("xcopy /Y /Q
\""+RUTA_PROYECTO_ACTUAL_VHDL+"\"
\""+RUTA_PROYECTO_ACTUAL_EDK+"\\pcores\\circuito_v1_00_a\\hdl\\vhdl\\\
"");
        proc_archivo.waitFor();
        if(proc_archivo.exitValue() != 0){
            return false;
        }


        return true;
    }
    /**
     * Crea el fichero user_logic integrando la interfaz con el bus
con el circuito
     */
    private boolean generarVhdl() throws Exception{
        Errores errores = new Errores();
        GeneraVhdl_V2 generador;

        generador = new
GeneraVhdl_V2(RUTA_PROYECTO_ACTUAL_EDK+"\\pcores\\circuito_v1_00_a\\hd
l\\vhdl\\circuito.vhd", compilador.getEntidad(), errores);
        if (generador.abrir()) {
            generador.crearFichero();
            generador.cerrar();
            return true;
        } else {
            this.muestraErroresConsola(errores);
            return false;
        }
    }
    /**
     * Modifica el fichero .pao del proyecto de EDK , concretamente el
creado en
     * pcores/adaptador_circuito/data hay que añadir todos los
ficheros que hemos
     * introducido desde el nessy
     */
    private void modificaPao() throws Exception {
        BufferedWriter bw = null;
        bw = new BufferedWriter(new
FileWriter(RUTA_PROYECTO_ACTUAL_EDK+"\\pcores\\circuito_v1_00_a\\data\
\circuito_v2_1_0.pao",true));
        for(int i=0;i<stringsVHD.size();i++){
            String fichero = stringsVHD.get(i);
            fichero =
fichero.substring(fichero.lastIndexOf('\\')+1,fichero.lastIndexOf('.')
);
            bw.write("\nlib circuito_v1_00_a "+fichero+" vhdl");
        }
        bw.close();
    }
    /**
```

```
     * Modifica el fichero .ucf del proyecto de EDK ubicado en la
carpeta data. Lo que
     * hacemos al introducir las siguientes lineas de código es decir
en que slices de la
     * fpga ubicamos adaptador_circuito
     */
    private void modificaUcf(int nsliceX, int nsliceY) throws
Exception {
        BufferedWriter bw = null;
            int sliceFinalX = 89;
            int sliceFinalY = 155;
            int slice_inicialX = sliceFinalX - nsliceX +1;
            int slice_inicialY = sliceFinalY - nsliceY +1;
            bw = new BufferedWriter(new
FileWriter(RUTA_PROYECTO_ACTUAL_EDK+"\\data\\system.ucf",true));
            bw.write("INST \"circuito_0\" AREA_GROUP =
\"pblock_circuito_0\";\n");
            bw.write("AREA_GROUP \"pblock_circuito_0\" RANGE =
SLICE_X"+slice_inicialX+"Y"+slice_inicialY+":SLICE_X"+sliceFinalX+"Y"+
sliceFinalY+";\n");
            bw.write("AREA_GROUP \"pblock_circuito_0\" MODE =
RECONFIG;\n\n");
            bw.close();
    }

    private void compilarProyecto(Semaforo sem){
        String comando = RUTA_EDK+"\\bin\\nt\\xps.exe -nw -scr
"+RUTA_NESSY+"\\comandosEDK\\generar_bit.tcl
"+RUTA_PROYECTO_ACTUAL_EDK+"\\system.xmp";
        String salida = RUTA_PROYECTO_ACTUAL_SALIDAS+"\\EDK.log";
        HiloProceso bitstream = new
HiloProceso(this._TextCrearBit,comando,salida,sem);
        bitstream.setName("Bitstream");
        bitstream.start();
    }

    private void data2mem(Semaforo sem){
        String comando2= RUTA_ISE+"\\bin\\nt\\data2mem.exe -bm " +
RUTA_PROYECTO_ACTUAL_EDK + "\\implementation\\system_bd.bmm " +
                                    "-bt " +
RUTA_PROYECTO_ACTUAL_EDK+"\\implementation\\system.bit "+
                                    "-bd " + RUTA_PROYECTO_ACTUAL_EDK +
"\\SDK_projects\\Nessy\\Debug\\Nessy.elf "+
                                    "tag microblaze_0 -o b
"+RUTA_PROYECTO_ACTUAL+"\\download.bit";
        String salida2 =
RUTA_PROYECTO_ACTUAL_SALIDAS+"\\Data2mem.log";
        HiloProceso data2mem = new
HiloProceso(_TextCrearBit,comando2,salida2,sem);
        data2mem.setName("Data2mem");
        data2mem.start();
    }

    private void copiarLog() throws Exception{
        BufferedReader br = new BufferedReader(new
FileReader(RUTA_PROYECTO_ACTUAL_SALIDAS+"\\EDK.log"));
        String linea = br.readLine();
```

```
        while(linea != null){
            _TextCrearBit.append(linea+"\n");
            linea = br.readLine();
        }
        br.close();
    }


////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS CARGAR .BIT
////////////////////////////////

////////////////////////////////////////////////////////////////////////
//////
    /**
     * Carga un fichero .BIT utilizando la interfaz de carga. Realiza
6 intentos
     * de carga por hubiera algún error ajeno a la aplicación que
impidiera
     * su carga. Si no estamos en el proceso de reconfiguración
parcial, se
     * mostrará un mensaje indicando que la carga del bitstream se ha
producido
     * correctamente.
     * @param fichero_bit El fichero a cargar
     * @param ab_mostrar_mensajes Indica si hay que mostrar mensajes
de
     * información de lo ocurrido.
     * @return true si ha sido correcta la ejecucion y false en caso
contrario
     */
    public boolean cargarBit(String fichero_bit) {
        Semaforo sem = new Semaforo();
        boolean error = false;
        try {
            seleccionaPanel(panelCargar);
            _TextCargarbit.setText(null);

            FileOutputStream os = new
FileOutputStream(RUTA_PROYECTO_ACTUAL_SALIDAS+"\\cargaBit.cmd");
            BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(os));
            String coms = "setMode -bscan \n"
                    + "setCable -p auto\n"
                    + "identify\n"
                    + "identifyMPM\n"
                    + "assignfile -p 3 -file \"" + fichero_bit +
"\"\n"
                    + "program -p 3\n"
                    + "quit";
            bw.write(coms);
            bw.close();

            String comando = this.RUTA_ISE + "\\bin\\nt\\impact.exe" +
" -batch " + RUTA_PROYECTO_ACTUAL_SALIDAS + "\\cargaBit.cmd";
```

```
            String salida = RUTA_PROYECTO_ACTUAL_SALIDAS +
"\\impact.txt";
            HiloProceso impact = new
HiloProceso(_TextCargarbit,comando,salida,sem);
            impact.setName("Impact");
            impact.start();
        } catch (Exception e) {
            error = true;
        }

        return !error;
    }

    /**
     * Comprueba si existe o no un fichero.
     * @param ruta Cadena de la ruta del fichero
     * @return Boolean,cierto si existe el fichero falso en caso
contrario.
     */
    public boolean existeFichero(String ruta) {
        File fichero = new File(ruta);
        return fichero.exists();
    }



////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS EJECUTAR
///////////////////////////////////

////////////////////////////////////////////////////////////////////////
//////
    public boolean ejecutar() throws Exception{
        boolean r = true;
        int i = 0;
        int numErrores = 0;
        String numErroresString = "";
        String cicloError = null;
        String bitError = "";
        this.seleccionaPanel(panelOutput);

        if(!comprobarComunicacion()) return false;

        entsal.enviarBinaria("01000000000000000000000000000000");
        numErroresString = entsal.recibirBinaria(32);
        //IMPORTANTE ERROR POR LA SALIDA DE LA FPGA
        numErrores = this.traduceString(numErroresString);
        this._TextSalida.append("Ha habido " + numErrores + " ciclos
en la ejecucion en los que hay errores: \n");
        for (i = 0; i < numErrores; i++) {
            cicloError = entsal.recibirBinaria(32);
            bitError = entsal.recibirBinaria(32);
            this._TextSalida.append("\t Errores en el ciclo " +
this.traduceString(cicloError) + ": bits " +
this.erroresString(bitError) + "\n");
```

```
this._TextSalida.setCaretPosition(_TextSalida.getText().length());
        }

        return r;
    }
    /**
     * Convierte a entero una una cadena en formato binario
     * @param s Cadena a traducir.
     * @return Entero equivalente al String introducido o -1 si el
formato no es
     * correcto (no son 0's y 1's)
     */
    public int traduceString(String s) {
        int n = 0;
        int peso = 1;
        for (int i = s.length() - 1; i >= 0; i--) {
            if (s.charAt(i) != '0' && s.charAt(i) != '1') {
                return -1;
            }
            if (s.charAt(i) == '1') {
                n = n + peso;
            }
            peso = peso * 2;
        }
        return n;
    }
    /**
     * Nos devuelve un String diciendo la posicion del String donde
hay 1's
     */
    public String erroresString(String s) {
        String r="";
        int i = 1;
        for(i=0;i<s.length();i++){
            if (s.charAt(i) == '1'){
                if(i<s.length()-1){
                    r = r + i+ ", ";
                }else{
                    r = r + i;
                }
            }
        }
        return r;
    }




//////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS CARGAR TESTBENCH
/////////////////////////
```

```
////////////////////////////////////////////////////////////////////////
//////
    private boolean cargaFicheroTB() {
        boolean error = false;
        JFileChooser chooser;

        this._TextTB.setText("");
        chooser = new JFileChooser();
        Filtro filter = new Filtro("txt");
        chooser.addChoosableFileFilter(filter);
        chooser.setCurrentDirectory(new java.io.File("."));
        chooser.setDialogTitle("Seleccionar TestBench");
        chooser.setAcceptAllFileFilterUsed(false);

        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            fichero_tb = chooser.getSelectedFile().getAbsolutePath();
        }else{
            error = true;
        }
        return !error;
    }

    public boolean cargarTB() throws Exception{
        if(entsal==null) configurarPuertoSerie();
        if(!comprobarComunicacion())return false;
        if(!cargaFicheroTB()) return false;
        enviarTB(fichero_tb);
        return true;
    }

    public boolean comprobarComunicacion(){
        boolean error = false;
        try {
            String pregunta = "11000000000000000000000000000000";
            entsal.enviarBinaria(pregunta);
            String respuesta = entsal.recibirBinaria(32);
            if(!respuesta.equals("11110000111100001111000011110000")){
                JOptionPane.showMessageDialog(this, "No se ha podido
establecer la comunicación con la FPGA", "Error",
JOptionPane.ERROR_MESSAGE);
                error = true;
            }
        } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
            error = true;
        }
        return !error;
    }

    /**
     * Devuelve una String de longitud 32-numEntradas con "0"
     * @param numEntradas
     * @return
```

```
     */
    public void enviarTB(String ficheroTB) throws Exception{
        seleccionaPanel(panelTB);
        _TextTB.setText("");

        //PRIMERO ENVIAS LOS 32 BITS DICIENDO EL TAMAÑO
        numCiclosTB = this.numeroCiclosFicheroTB(ficheroTB);
        String numLineasString = Integer.toBinaryString(numCiclosTB);
        String numLineasString32 =
StringHasta32bits(numLineasString.length()) + numLineasString;
        entsal.enviarBinaria(numLineasString32);

        //LUEGO ENVIAS TODOS LOS CICLOS DE RELOJ
        String linea = null;
        String lineaEnvio = null;
        int numLineas=1;

        BufferedReader bf;
        bf = new BufferedReader(new FileReader(ficheroTB));
        do {
            linea = bf.readLine();
            if (linea != null) {
                this._TextTB.append(numLineas+":
\t"+linea.substring(0,entidad.getBitsEntrada()) + "\n");

this._TextTB.setCaretPosition(_TextTB.getText().length());
                lineaEnvio= linea +
this.StringHasta32bits(entidad.getBitsEntrada());
                entsal.enviarBinaria(lineaEnvio);
                numLineas++;
            }
        } while (linea != null);
        bf.close();
    }

    public String StringHasta32bits(int numEntradas){
        int i;
        String s="";
        for(i=numEntradas; i<32; i++){
            s=s+"0";
        }
        return s;
    }
    public String StringHasta32bitsReverso(int numEntradas){
        int i;
        String s="";
        for(i=numEntradas; i<32; i++){
            s="0"+s;
        }
        return s;
    }

    /**
     * Comprueba el numero de lineas que tiene un testbench,
supuestamente antes has
     * comprobado que es correct0
     * @param ficheroTB Fichero de Test Bench a analizar
```

```
     * @return true si el formato es correcto y false en caso
contrario
     */
    public int numeroCiclosFicheroTB(String ficheroTB) throws
Exception{
        String linea = null;
        int numLineas = 0;

        BufferedReader bf;
        bf = new BufferedReader(new FileReader(ficheroTB));
        do {
            linea = bf.readLine();
            if (linea != null) {
                numLineas ++ ;
            }
        } while (linea != null );

        bf.close();

        return numLineas;
    }




////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS GENERAR GOLDEN
/////////////////////////////

////////////////////////////////////////////////////////////////////////
//////
    public boolean generarGolden() throws Exception{
        String ciclo_salida = null;
        boolean r=true;
        int i;
        seleccionaPanel(panelGolden);
        _TextGolden.setText(null);
        if(!comprobarComunicacion()) return false;
        BufferedWriter bw;
        this.fichero_golden = RUTA_PROYECTO_ACTUAL_SALIDAS +
"\\golden.txt";
        bw = new BufferedWriter(new FileWriter(fichero_golden));
        entsal.enviarBinaria("00100000000000000000000000000000");
        entsal.enviarBinaria("01100000000000000000000000000000");
        for (i = 0; i < this.numCiclosTB; i++) {
            ciclo_salida = entsal.recibirBinaria(32);
            bw.write(ciclo_salida.substring(0,
entidad.getBitsSalida()) + "\n");
            this._TextGolden.append(i + 1 + ": \t" +
ciclo_salida.substring(0, entidad.getBitsSalida()) + "\n");

this._TextGolden.setCaretPosition(_TextGolden.getText().length());
        }
        bw.close();
```

```
        return r;
    }



////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS CARGAR GOLDEN
/////////////////////////////

////////////////////////////////////////////////////////////////////////
//////
    private boolean cargarGolden() throws Exception{
        boolean r = true;
        JFileChooser chooser;

        if(!comprobarComunicacion()) return false;
        chooser = new JFileChooser();
        Filtro filter = new Filtro("txt");
        chooser.addChoosableFileFilter(filter);
        chooser.setCurrentDirectory(new java.io.File("."));
        chooser.setDialogTitle("Seleccionar Archivo Golden");
        chooser.setAcceptAllFileFilterUsed(false);
        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            fichero_golden =
chooser.getSelectedFile().getAbsolutePath();
            enviarGolden(fichero_golden);
        } else {
            r = false;
        }

        return r;
    }
    public void enviarGolden(String ficheroTB) throws
FileNotFoundException, IOException{
        seleccionaPanel(panelGolden);
        _TextGolden.setText(null);

        //codigo de golden
        entsal.enviarBinaria("10000000000000000000000000000000");

        //LUEGO ENVIAS TODOS LOS CICLOS DE RELOJ
        String linea = null;
        String lineaEnvio = null;
        int numLinea =1;

        BufferedReader bf;
        bf = new BufferedReader(new FileReader(ficheroTB));
        do {
            linea = bf.readLine();
            if (linea != null) {
```

```
                this._TextGolden.append(numLinea+":
\t"+linea.substring(0,entidad.getBitsSalida()) + "\n");
                lineaEnvio= linea +
this.StringHasta32bits(entidad.getBitsSalida()) ;
                entsal.enviarBinaria(lineaEnvio);
                numLinea++;
            }
        } while (linea != null);
        bf.close();
    }



//////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS RECONFIGURACION
//////////////////////////

//////////////////////////////////////////////////////////////////////////
//////
    public boolean reconfiguracion() {
        BufferedInputStream bi = null;
        boolean r = true;

        int tamaño_bit32bits = 0;
        int slice_finalX = 89;
        int slice_finalY = 155;
        int slice_inicialX = slice_finalX - nsliceX + 1;
        int slice_inicialY = slice_finalY - nsliceY + 1;
        seleccionaPanel(panelErroresReconfiguracion);
        try {
            File bitstream_completo = new File(RUTA_PROYECTO_ACTUAL +
"\\download.bit");
            bi = new BufferedInputStream(new
FileInputStream(bitstream_completo));
            if (bitstream_completo.exists()) {
                if (!comprobarComunicacion()) return false;

                int f_inicio = 30+ (slice_inicialX/2*22);
                int f_fin = 30+ ((((slice_finalX)/2)+1)*22);
                int palabra_inicio = (f_inicio * 206) + 1;
                int palabra_fin = (f_fin * 206);
                String comando = "java -jar
"+RUTA_NESSY+"\\Virtex_II_Configuration_Cleaner.jar -i " +
RUTA_PROYECTO_ACTUAL+"\\download.bit -o
"+RUTA_PROYECTO_ACTUAL+"\\download_clean -iw " + palabra_inicio+ " -ew
" + palabra_fin;
                Process proc_parcial =
Runtime.getRuntime().exec(comando);
                proc_parcial.waitFor();

                File bitstream_parcial = new File(RUTA_PROYECTO_ACTUAL
+ "\\download_clean.bit");
                bi = new BufferedInputStream(new
FileInputStream(bitstream_parcial));
```

```
                //calcular el tamaño del bit parcial
                long tamaño_bit = bitstream_parcial.length();
                if (tamaño_bit % 4 == 0)
                    tamaño_bit32bits = (int) (tamaño_bit/4);
                else
                    tamaño_bit32bits = (int) (tamaño_bit/4)+1;

                //Enviar el comando de reconfiguracion con el tamaño
del bit parcial
                String num_palabras32bits =
Integer.toBinaryString(tamaño_bit32bits);
                String palabra = "101" +
StringHasta32bits(num_palabras32bits.length() + 3) +
num_palabras32bits;
                entsal.enviarBinaria(palabra);

                //Calcular la informacion de los frame que hay que
reconfigurar
                String frame_inicio =
StringHasta32bitsReverso(Integer.toBinaryString(f_inicio).length()) +
Integer.toBinaryString(f_inicio);
                String frame_fin =
StringHasta32bitsReverso(Integer.toBinaryString(f_fin).length()) +
Integer.toBinaryString(f_fin);
                String sliceY0 =
StringHasta32bitsReverso(Integer.toBinaryString(slice_inicialY).length
()) + Integer.toBinaryString(slice_inicialY);
                String sliceY1 =
StringHasta32bitsReverso(Integer.toBinaryString(slice_finalY).length()
) + Integer.toBinaryString(slice_finalY);

                //Enviar la información sobre donde está instanciado
el circuito en la FPGA
                entsal.enviarBinaria(frame_inicio);
                entsal.enviarBinaria(frame_fin);
                entsal.enviarBinaria(sliceY0);
                entsal.enviarBinaria(sliceY1);

                //Enviar el .bit por la entrada serie para escribirlo
en ram y a continuación empezar con la inyección de errores
                HiloInyeccion inyeccion = new
HiloInyeccion(_TextErroresReconfiguracion,this,entsal,bi,tamaño_bit32b
its, f_inicio, f_fin, slice_inicialY, slice_finalY);
                inyeccion.setName("Inyección de errores");
                inyeccion.start();

            }
        } catch (Exception e) {

//Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return r;
    }

    public String intToBinario8(int n){
        String binaria = Integer.toBinaryString(n);
```

```
        int l = binaria.length();
        for(int i=l;i<8;i++){
            binaria = '0'+binaria;
        }
        return binaria;
    }


////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////// METODOS CONFIGUARACION
////////////////////////////

////////////////////////////////////////////////////////////////////////
//////
    /**
     * Vuelve a cargar la configuracón de la aplicación
     * @param fichConf El fichero de configuración
     * @param cargaFich Indica si hay que carga desde fichero
     */
    private void config(String fichConf, boolean cargaFich) {
        Properties prop = new Properties();
        InputStream is = null;
        try {
            is = new FileInputStream(fichConf);
            prop.load(is);
            // Enumeration enume=prop.propertyNames();

            RUTA_ISE = prop.getProperty("HomeXilinxISE");
            RUTA_EDK = prop.getProperty("HomeXilinxEDK");
            if (cargaFich) {
                if (RUTA_ISE == null || RUTA_ISE.equals("") ||
RUTA_EDK == null || RUTA_EDK.equals("")) {
                    JOptionPane.showMessageDialog(this, ""
                            + "El fichero de configuración
seleccionado "
                            + "no es valido", "Info",
JOptionPane.INFORMATION_MESSAGE);
                }


            } else {
                boolean cierre=false;
                while ((RUTA_ISE == null || RUTA_ISE.equals("") ||
RUTA_EDK == null || RUTA_EDK.equals(""))&& !cierre) {
                    GUIConfig config = new GUIConfig(this, true,
"","");
                    config.setVisible(true);
                    is = new FileInputStream(fichConf);
                    prop.load(is);
                    RUTA_ISE = prop.getProperty("HomeXilinxISE");
                    RUTA_EDK = prop.getProperty("HomeXilinxEDK");
                    cierre=config.getCierre();
                }
            }
```

```
        } catch (IOException ioe) {

             JOptionPane.showMessageDialog(this, "No ha sido posible
configurar" +
                     " nessy, debido a que no se ha encontrado archivo
de " +
                     "configuración, para poder ejecutar Nessy
correctamente" +
                     " acceda a configuración.", "Info",
JOptionPane.INFORMATION_MESSAGE);
        }

    }

    /*
     * Procedimiento para cambiar la ruta de barras \ a barras /
     */
    public String cambiarRuta(String ruta){
        String s="";
        s=ruta.replace('\\', '/');
        return s;
    }



////////////////////////////////////////////////////////////////////////
//////
    ////////////////////////////// OTROS METODOS
//////////////////////////////////

////////////////////////////////////////////////////////////////////////
//////

    private void deshabilitarBotones(boolean b) {
        _btnCargarBit.setEnabled(!b);
        _btnCargarGolden.setEnabled(!b);
        _btnCargarTB.setEnabled(!b);
        _btnCargarVhd.setEnabled(!b);
        _btnCrearBit.setEnabled(!b);
        _btnEjecutar.setEnabled(!b);
        _btnGenerarGolden.setEnabled(!b);
        _btnReconfiguracion.setEnabled(!b);
    }
    /**
     * Selecciona uno de las cuatro pestañas según lo pasado por
parámetro
     * @param panel Panel al que se quiere cambiar
     */
    public void seleccionaPanel(JPanel panel) {
        try {
            if ((Boolean) ((JTabbedPaneWithCloseIcon)
jTabbedPane1).getTablaPaneles().get(panel)) {
                jTabbedPane1.setSelectedComponent(panel);
            } else {
                if (panel.getName().equals(panelOutput)){
                    _TextSalida.setColumns(20);
```

```
                    _TextSalida.setEditable(false);
                    _TextSalida.setRows(5);
                    jScrollPane4.setViewportView(_TextSalida);

                    javax.swing.GroupLayout panelOutPutLayout = new
javax.swing.GroupLayout(panelOutput);
                    panelCrearBit.setLayout(panelOutPutLayout);
                    panelOutPutLayout.setHorizontalGroup(

panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 715, Short.MAX_VALUE));
                    panelOutPutLayout.setVerticalGroup(

panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 297, Short.MAX_VALUE));

                    jTabbedPane1.addTab("Golden", panelOutput);
                    jTabbedPane1.setSelectedComponent(panelOutput);
                }
                else if(panel.getName().equals(panelCrearBit))
                {
                    _TextCrearBit.setColumns(20);
                    _TextCrearBit.setEditable(false);
                    _TextCrearBit.setRows(5);
                    jScrollPane4.setViewportView(_TextCrearBit);

                    javax.swing.GroupLayout panelOutPutLayout = new
javax.swing.GroupLayout(panelCrearBit);
                    panelCrearBit.setLayout(panelOutPutLayout);
                    panelOutPutLayout.setHorizontalGroup(

panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 715, Short.MAX_VALUE));
                    panelOutPutLayout.setVerticalGroup(

panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 297, Short.MAX_VALUE));

                    jTabbedPane1.addTab("Crear bit", panelCrearBit);
                    jTabbedPane1.setSelectedComponent(panelCrearBit);
                }
                else if(panel.getName().equals(panelCargar))
                {
                    _TextCargarbit.setColumns(20);
                    _TextCargarbit.setEditable(false);
                    _TextCargarbit.setRows(5);
                    jScrollPane4.setViewportView(_TextCargarbit);

                    javax.swing.GroupLayout panelOutPutLayout = new
javax.swing.GroupLayout(panelCargar);
                    panelCrearBit.setLayout(panelOutPutLayout);
                    panelOutPutLayout.setHorizontalGroup(
```

```
panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 715, Short.MAX_VALUE));
                    panelOutPutLayout.setVerticalGroup(

panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 297, Short.MAX_VALUE));

                    jTabbedPane1.addTab("Crear bit parcial",
panelCargar);
                    jTabbedPane1.setSelectedComponent(panelCargar);
                }
                else
if(panel.getName().equals(panelErroresReconfiguracion)) {
                    _TextErroresReconfiguracion.setColumns(20);
                    _TextErroresReconfiguracion.setEditable(false);
                    _TextErroresReconfiguracion.setRows(5);

jScrollPane4.setViewportView(_TextErroresReconfiguracion);

                    javax.swing.GroupLayout panelOutPutLayout = new
javax.swing.GroupLayout(panelErroresReconfiguracion);

panelErroresReconfiguracion.setLayout(panelOutPutLayout);
                    panelOutPutLayout.setHorizontalGroup(

panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 715, Short.MAX_VALUE));
                    panelOutPutLayout.setVerticalGroup(

panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 297, Short.MAX_VALUE));

                    jTabbedPane1.addTab("OutPut",
panelErroresReconfiguracion);

jTabbedPane1.setSelectedComponent(panelErroresReconfiguracion);
                } else if (panel.getName().equals(panelTB)) {
                    _TextTB.setColumns(20);
                    _TextTB.setRows(5);
                    _TextTB.setMaximumSize(getMaximumSize());
                    jScrollPane2.setViewportView(_TextTB);

                    javax.swing.GroupLayout panelCargarLayout = new
javax.swing.GroupLayout(panelTB);
                    panelTB.setLayout(panelCargarLayout);
                    panelCargarLayout.setHorizontalGroup(

panelCargarLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 715, Short.MAX_VALUE));
                    panelCargarLayout.setVerticalGroup(
```

```
panelCargarLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 297, Short.MAX_VALUE));

                    jTabbedPane1.addTab("Cargar", panelTB);
                    jTabbedPane1.setSelectedComponent(panelTB);
                } else if (panel.getName().equals(panelGolden)) {
                    _TextGolden.setColumns(20);
                    _TextGolden.setRows(5);
                    jScrollPane3.setViewportView(_TextGolden);

                    javax.swing.GroupLayout panelTBLayout = new
javax.swing.GroupLayout(panelGolden);
                    panelGolden.setLayout(panelTBLayout);
                    panelTBLayout.setHorizontalGroup(

panelTBLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING).addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 715, Short.MAX_VALUE));
                    panelTBLayout.setVerticalGroup(

panelTBLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING).addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 297, Short.MAX_VALUE));

                    jTabbedPane1.addTab("Test Bench", panelGolden);
                    jTabbedPane1.setSelectedComponent(panelGolden);
                } else {
                    _TxtEntityVHD.setColumns(20);
                    _TxtEntityVHD.setEditable(false);
                    _TxtEntityVHD.setRows(5);
                    jScrollPane1.setViewportView(_TxtEntityVHD);

                    javax.swing.GroupLayout panelVHDLayout = new
javax.swing.GroupLayout(panelVHD);
                    panelVHD.setLayout(panelVHDLayout);
                    panelVHDLayout.setHorizontalGroup(

panelVHDLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING).addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 699, Short.MAX_VALUE));
                    panelVHDLayout.setVerticalGroup(

panelVHDLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING).addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 297, Short.MAX_VALUE));

                    jTabbedPane1.addTab("Entity VHDL", panelVHD);
                    jTabbedPane1.setSelectedComponent(panelVHD);

                }

            }
        } catch (Exception ex) {
```

```
//
Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
    private void configLog(String fichConf, boolean cargaFich) {
        Properties prop = new Properties();
        InputStream is = null;
        try {
            is = new FileInputStream(fichConf);
            prop.load(is);
            // Enumeration enume=prop.propertyNames();

_TxtEntityVHD.setText(prop.getProperty("log4j.appender.LOGFILE.file"))
;
        } catch (IOException ioe) {
             JOptionPane.showMessageDialog(this, "No ha sido posible
configurar" +
                    " nessy, debido a que no se ha encontrado archivo
de " +
                    "configuración, para poder ejecutar Nessy
correctamente" +
                    " acceda a configuración.", "Info",
JOptionPane.INFORMATION_MESSAGE);
        }

    }
    private void initComponentsAux() {
        jTabbedPane1 = new JTabbedPaneWithCloseIcon();
    }


    public boolean configurarPuertoSerie(){
        boolean r = true;
        GUIComConfig GUIcom = new GUIComConfig(this, r);
        GUIcom.setLocationRelativeTo(null);
        GUIcom.setVisible(true);
        entsal = new EntradaSalida(GUIcom.getSel());
        r = entsal.inicializarPuertoSerie();
        if(!r){
            JOptionPane.showMessageDialog(this, "Error al abrir el
puerto "+GUIcom.getSel(), "Error", JOptionPane.ERROR_MESSAGE);
            entsal = null;
        }
    return r;
    }

////////////////////////////////////////////////////////////////////////
////////////
//////////////////////////CODIGO CREADO POR
NETBEANS//////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////


    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
```

```
    private void initComponents() {

        jPanel2 = new javax.swing.JPanel();
        jToolBar1 = new javax.swing.JToolBar();
        _btnCargarVhd = new javax.swing.JButton();
        _btnCrearBit = new javax.swing.JButton();
        _btnCargarBit = new javax.swing.JButton();
        _btnCargarTB = new javax.swing.JButton();
        _btnGenerarGolden = new javax.swing.JButton();
        _btnCargarGolden = new javax.swing.JButton();
        _btnEjecutar = new javax.swing.JButton();
        _btnReconfiguracion = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        _btnClear = new javax.swing.JButton();
        jTabbedPane1 = new JTabbedPaneWithCloseIcon();
        panelVHD = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        _TxtEntityVHD = new javax.swing.JTextArea();
        panelCrearBit = new javax.swing.JPanel();
        jScrollPane5 = new javax.swing.JScrollPane();
        _TextCrearBit = new javax.swing.JTextArea();
        panelCargar = new javax.swing.JPanel();
        jScrollPane7 = new javax.swing.JScrollPane();
        _TextCargarbit = new javax.swing.JTextArea();
        panelTB = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        _TextTB = new javax.swing.JTextArea();
        panelGolden = new javax.swing.JPanel();
        jScrollPane3 = new javax.swing.JScrollPane();
        _TextGolden = new javax.swing.JTextArea();
        panelOutput = new javax.swing.JPanel();
        jScrollPane6 = new javax.swing.JScrollPane();
        _TextSalida = new javax.swing.JTextArea();
        panelErroresReconfiguracion = new javax.swing.JPanel();
        jScrollPane4 = new javax.swing.JScrollPane();
        _TextErroresReconfiguracion = new javax.swing.JTextArea();
        jSeparator1 = new javax.swing.JSeparator();
        _comentariosProyecto = new javax.swing.JLabel();
        jMenuBar1 = new javax.swing.JMenuBar();
        menuOpciones = new javax.swing.JMenu();
        _menuOpcionesNuevoProyecto = new javax.swing.JMenuItem();
        _menuOpcionesAbrirProyecto = new javax.swing.JMenuItem();
        _menuOpcionesCerrarProyecto = new javax.swing.JMenuItem();
        _menuOpcionesCerrarNessy = new javax.swing.JMenuItem();
        menuConfig = new javax.swing.JMenu();
        menuConfigNessy = new javax.swing.JMenuItem();
        menuConfigFichConf = new javax.swing.JMenuItem();
        menuConfigCom = new javax.swing.JMenuItem();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Nessy 5.0"); // NOI18N
        setIconImage(new
ImageIcon("src/recursos/Nessy.png").getImage());
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosed(java.awt.event.WindowEvent evt) {
```

```
                formWindowClosed(evt);
            }
            public void windowClosing(java.awt.event.WindowEvent evt)
{
                formWindowClosing(evt);
            }
        });


jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Opcion
es"));
        jPanel2.setOpaque(false);

        jToolBar1.setRollover(true);

        _btnCargarVhd.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnCargarVhdl.
png"))); // NOI18N
        _btnCargarVhd.setText("Cargar VHD"); // NOI18N
        _btnCargarVhd.setEnabled(false);
        _btnCargarVhd.setFocusable(false);

_btnCargarVhd.setHorizontalTextPosition(javax.swing.SwingConstants.CEN
TER);

_btnCargarVhd.setVerticalTextPosition(javax.swing.SwingConstants.BOTTO
M);
        _btnCargarVhd.setEnabled(false);
        _btnCargarVhd.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCargarVhdActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnCargarVhd);

        _btnCrearBit.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnCrearBit.pn
g"))); // NOI18N
        _btnCrearBit.setText("Crear bitstream"); // NOI18N
        _btnCrearBit.setEnabled(false);
        _btnCrearBit.setFocusable(false);

_btnCrearBit.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);

_btnCrearBit.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
);
        _btnCrearBit.setEnabled(false);
        _btnCrearBit.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCrearBitActionPerformed(evt);
            }
        });
```

```
        jToolBar1.add(_btnCrearBit);

        _btnCargarBit.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnCargarBit.p
ng"))); // NOI18N
        _btnCargarBit.setText("Cargar bitstream"); // NOI18N
        _btnCargarBit.setEnabled(false);
        _btnCargarBit.setFocusable(false);

_btnCargarBit.setHorizontalTextPosition(javax.swing.SwingConstants.CEN
TER);

_btnCargarBit.setVerticalTextPosition(javax.swing.SwingConstants.BOTTO
M);
        _btnCargarBit.setEnabled(false);
        _btnCargarBit.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCargarBitActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnCargarBit);

        _btnCargarTB.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnCargarTB.pn
g"))); // NOI18N
        _btnCargarTB.setText("Cargar Testbench"); // NOI18N
        _btnCargarTB.setEnabled(false);
        _btnCargarTB.setFocusable(false);

_btnCargarTB.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);

_btnCargarTB.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
);
        _btnCargarTB.setEnabled(false);
        _btnCargarTB.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCargarTBActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnCargarTB);

        _btnGenerarGolden.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnGeneraGolde
n.png"))); // NOI18N
        _btnGenerarGolden.setText("Generar golden"); // NOI18N
        _btnGenerarGolden.setEnabled(false);
        _btnGenerarGolden.setFocusable(false);

_btnGenerarGolden.setHorizontalTextPosition(javax.swing.SwingConstants
.CENTER);
```

```
_btnGenerarGolden.setVerticalTextPosition(javax.swing.SwingConstants.B
OTTOM);
        _btnGenerarGolden.setEnabled(false);
        _btnGenerarGolden.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnGenerarGoldenActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnGenerarGolden);

        _btnCargarGolden.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnCargaGolden
.jpg"))); // NOI18N
        _btnCargarGolden.setText("Cargar golden"); // NOI18N
        _btnCargarGolden.setEnabled(false);
        _btnCargarGolden.setFocusable(false);

_btnCargarGolden.setHorizontalTextPosition(javax.swing.SwingConstants.
CENTER);

_btnCargarGolden.setVerticalTextPosition(javax.swing.SwingConstants.BO
TTOM);
        _btnCargarGolden.setEnabled(false);
        _btnCargarGolden.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCargarGoldenActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnCargarGolden);

        _btnEjecutar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnEjec.png"))
); // NOI18N
        _btnEjecutar.setText("Ejecutar"); // NOI18N
        _btnEjecutar.setEnabled(false);
        _btnEjecutar.setFocusable(false);

_btnEjecutar.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);

_btnEjecutar.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
);
        _btnEjecutar.setEnabled(false);
        _btnEjecutar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnEjecutarActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnEjecutar);
```

```
        _btnReconfiguracion.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/reconParc.JPG"
))); // NOI18N
        _btnReconfiguracion.setText("Reconfiguracion"); // NOI18N
        _btnReconfiguracion.setEnabled(false);
        _btnReconfiguracion.setEnabled(false);
        _btnReconfiguracion.setFocusable(false);

_btnReconfiguracion.setHorizontalTextPosition(javax.swing.SwingConstan
ts.CENTER);

_btnReconfiguracion.setVerticalTextPosition(javax.swing.SwingConstants
.BOTTOM);
        _btnReconfiguracion.setEnabled(false);
        _btnReconfiguracion.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnReconfiguracionActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnReconfiguracion);

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addComponent(jToolBar1,
javax.swing.GroupLayout.PREFERRED_SIZE, 745,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(27, Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jToolBar1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );


jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Pantal
la"));
        jPanel1.setOpaque(false);

        _btnClear.setText("Limpiar Pantalla"); // NOI18N
        _btnClear.setAutoscrolls(true);
```

```
        _btnClear.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnClearActionPerformed(evt);
            }
        });

        _TxtEntityVHD.setColumns(20);
        _TxtEntityVHD.setEditable(false);
        _TxtEntityVHD.setRows(5);
        jScrollPane1.setViewportView(_TxtEntityVHD);

        javax.swing.GroupLayout panelVHDLayout = new
javax.swing.GroupLayout(panelVHD);
        panelVHD.setLayout(panelVHDLayout);
        panelVHDLayout.setHorizontalGroup(

panelVHDLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(panelVHDLayout.createSequentialGroup()
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 744,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        panelVHDLayout.setVerticalGroup(

panelVHDLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Entity VHDL", panelVHD);

        _TextCrearBit.setColumns(20);
        _TextCrearBit.setEditable(false);
        _TextCrearBit.setRows(5);
        jScrollPane5.setViewportView(_TextCrearBit);

        javax.swing.GroupLayout panelCrearBitLayout = new
javax.swing.GroupLayout(panelCrearBit);
        panelCrearBit.setLayout(panelCrearBitLayout);
        panelCrearBitLayout.setHorizontalGroup(

panelCrearBitLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
            .addComponent(jScrollPane5,
javax.swing.GroupLayout.DEFAULT_SIZE, 747, Short.MAX_VALUE)
        );
        panelCrearBitLayout.setVerticalGroup(

panelCrearBitLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
```

```
            .addComponent(jScrollPane5,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Crear .Bit", panelCrearBit);

        _TextCargarbit.setColumns(20);
        _TextCargarbit.setEditable(false);
        _TextCargarbit.setRows(5);
        jScrollPane7.setViewportView(_TextCargarbit);

        javax.swing.GroupLayout panelCargarLayout = new
javax.swing.GroupLayout(panelCargar);
        panelCargar.setLayout(panelCargarLayout);
        panelCargarLayout.setHorizontalGroup(

panelCargarLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addComponent(jScrollPane7,
javax.swing.GroupLayout.DEFAULT_SIZE, 747, Short.MAX_VALUE)
        );
        panelCargarLayout.setVerticalGroup(

panelCargarLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addComponent(jScrollPane7,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Cargar", panelCargar);

        _TextTB.setColumns(20);
        _TextTB.setRows(5);
        _TextTB.setMaximumSize(getMaximumSize());
        jScrollPane2.setViewportView(_TextTB);

        javax.swing.GroupLayout panelTBLayout = new
javax.swing.GroupLayout(panelTB);
        panelTB.setLayout(panelTBLayout);
        panelTBLayout.setHorizontalGroup(

panelTBLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 747, Short.MAX_VALUE)
        );
        panelTBLayout.setVerticalGroup(

panelTBLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Test Bench", panelTB);
```

```
        _TextGolden.setColumns(20);
        _TextGolden.setRows(5);
        jScrollPane3.setViewportView(_TextGolden);

        javax.swing.GroupLayout panelGoldenLayout = new
javax.swing.GroupLayout(panelGolden);
        panelGolden.setLayout(panelGoldenLayout);
        panelGoldenLayout.setHorizontalGroup(

panelGoldenLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 747, Short.MAX_VALUE)
        );
        panelGoldenLayout.setVerticalGroup(

panelGoldenLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Golden", panelGolden);

        _TextSalida.setColumns(20);
        _TextSalida.setEditable(false);
        _TextSalida.setRows(5);
        jScrollPane6.setViewportView(_TextSalida);

        javax.swing.GroupLayout panelOutputLayout = new
javax.swing.GroupLayout(panelOutput);
        panelOutput.setLayout(panelOutputLayout);
        panelOutputLayout.setHorizontalGroup(

panelOutputLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addComponent(jScrollPane6,
javax.swing.GroupLayout.DEFAULT_SIZE, 747, Short.MAX_VALUE)
        );
        panelOutputLayout.setVerticalGroup(

panelOutputLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addComponent(jScrollPane6,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Output", panelOutput);

        _TextErroresReconfiguracion.setColumns(20);
        _TextErroresReconfiguracion.setEditable(false);
        _TextErroresReconfiguracion.setRows(5);
        jScrollPane4.setViewportView(_TextErroresReconfiguracion);
```

```
        javax.swing.GroupLayout panelErroresReconfiguracionLayout =
new javax.swing.GroupLayout(panelErroresReconfiguracion);

panelErroresReconfiguracion.setLayout(panelErroresReconfiguracionLayou
t);
        panelErroresReconfiguracionLayout.setHorizontalGroup(

panelErroresReconfiguracionLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)
            .addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 747, Short.MAX_VALUE)
        );
        panelErroresReconfiguracionLayout.setVerticalGroup(

panelErroresReconfiguracionLayout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)
            .addComponent(jScrollPane4,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Errores Reconfiguraci\u00f3n",
panelErroresReconfiguracion);

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap(488, Short.MAX_VALUE)
                .addComponent(_btnClear,
javax.swing.GroupLayout.PREFERRED_SIZE, 190,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(94, 94, 94))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 752,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(19, 19, 19)
                .addComponent(_btnClear)
```

```
                .addContainerGap(405, Short.MAX_VALUE))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(40, 40, 40)
                    .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 350,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap(57, Short.MAX_VALUE)))
        );

        jTabbedPane1.getAccessibleContext().setAccessibleName(""); //
NOI18N

        _comentariosProyecto.setFont(new java.awt.Font("Tahoma", 1,
14));
        _comentariosProyecto.setText("Ningun proyecto Nessy abierto");

        menuOpciones.setText("Archivo"); // NOI18N
        menuOpciones.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesActionPerformed(evt);
            }
        });


_menuOpcionesNuevoProyecto.setAccelerator(javax.swing.KeyStroke.getKey
Stroke(java.awt.event.KeyEvent.VK_N,
java.awt.event.InputEvent.CTRL_MASK));
        _menuOpcionesNuevoProyecto.setText("Nuevo Proyecto"); //
NOI18N
        _menuOpcionesNuevoProyecto.setEnabled(true);
        _menuOpcionesNuevoProyecto.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _menuOpcionesNuevoProyectoActionPerformed(evt);
            }
        });
        menuOpciones.add(_menuOpcionesNuevoProyecto);


_menuOpcionesAbrirProyecto.setAccelerator(javax.swing.KeyStroke.getKey
Stroke(java.awt.event.KeyEvent.VK_A,
java.awt.event.InputEvent.CTRL_MASK));
        _menuOpcionesAbrirProyecto.setText("Abrir Proyecto"); //
NOI18N
        _menuOpcionesAbrirProyecto.setEnabled(true);
        _menuOpcionesAbrirProyecto.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _menuOpcionesAbrirProyectoActionPerformed(evt);
            }
```

```
        });
        menuOpciones.add(_menuOpcionesAbrirProyecto);


_menuOpcionesCerrarProyecto.setAccelerator(javax.swing.KeyStroke.getKe
yStroke(java.awt.event.KeyEvent.VK_C,
java.awt.event.InputEvent.CTRL_MASK));
        _menuOpcionesCerrarProyecto.setText("Cerrar Proyecto"); //
NOI18N
        _menuOpcionesCerrarProyecto.setEnabled(false);
        _menuOpcionesCerrarProyecto.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _menuOpcionesCerrarProyectoActionPerformed(evt);
            }
        });
        menuOpciones.add(_menuOpcionesCerrarProyecto);


_menuOpcionesCerrarNessy.setAccelerator(javax.swing.KeyStroke.getKeySt
roke(java.awt.event.KeyEvent.VK_ESCAPE, 0));
        _menuOpcionesCerrarNessy.setText("Cerrar Nessy"); // NOI18N
        _menuOpcionesCerrarNessy.setEnabled(true);
        _menuOpcionesCerrarNessy.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _menuOpcionesCerrarNessyActionPerformed(evt);
            }
        });
        menuOpciones.add(_menuOpcionesCerrarNessy);

        jMenuBar1.add(menuOpciones);

        menuConfig.setText("Configuración"); // NOI18N
        menuConfig.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuConfigActionPerformed(evt);
            }
        });

        menuConfigNessy.setText("Configurar Nessy"); // NOI18N
        menuConfigNessy.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuConfigNessyActionPerformed(evt);
            }
        });
        menuConfig.add(menuConfigNessy);

        menuConfigFichConf.setText("Cargar Fichero Configuración"); //
NOI18N
```

```
        menuConfigFichConf.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuConfigFichConfActionPerformed(evt);
            }
        });
        menuConfig.add(menuConfigFichConf);

        menuConfigCom.setText("Configurar puerto serie");
        menuConfigCom.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuConfigComActionPerformed(evt);
            }
        });
        menuConfig.add(menuConfigCom);

        jMenuBar1.add(menuConfig);

        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jSeparator1,
javax.swing.GroupLayout.DEFAULT_SIZE, 1, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(10, 10, 10)
                        .addComponent(_comentariosProyecto,
javax.swing.GroupLayout.PREFERRED_SIZE, 770,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.TRAILING)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                        .addGap(147, 147, 147)
                        .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(_comentariosProyecto,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void
_btnCargarVhdActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnCargarVhdActionPerformed
        Seleccion sel = new Seleccion();
        GUICargaVHDL guiVhdl = new GUICargaVHDL(this, true, sel);
        guiVhdl.setLocationRelativeTo(null);
        guiVhdl.setVisible(true);
        if
(sel.seleccion.equals(SeleccionCargaVHD.SELECCION_VHDL_TOP)) {
            cargaTopVHDL();
            _btnCrearBit.setEnabled(true);
        } else {
            if
(sel.seleccion.equals(SeleccionCargaVHD.SELECCION_VARIOS_VHDL)) {
                cargaVariosVHDL();
                _btnCrearBit.setEnabled(true);
            }
        }
        try {
            File dirVhdl = new File(RUTA_PROYECTO_ACTUAL_VHDL);
            File[] listaArchivos = dirVhdl.listFiles();
```

```
            for(int i=0;i<listaArchivos.length;i++){
                listaArchivos[i].delete();
            }
            for(int i=0;i<ficherosVHD.size();i++){
                Process proc_archivo =
Runtime.getRuntime().exec("xcopy /Y /Q \""+ficherosVHD.get(i)+"\"
\""+RUTA_PROYECTO_ACTUAL_VHDL+"\"");
                proc_archivo.waitFor();
            }
            stringsVHD.clear();
            for(int i=0;i<ficherosVHD.size();i++){
                stringsVHD.add(ficherosVHD.get(i).getName());
            }
            BufferedWriter bw2 = new BufferedWriter(new
FileWriter(RUTA_PROYECTO_ACTUAL+"\\Proyecto.nessy"));
            if(ficherosVHD.size()>1)
                bw2.write(ficherosVHD.get(top).getName()+"\n");
            else
                bw2.write(ficherosVHD.get(0).getName()+"\n");
            bw2.close();
        } catch (InterruptedException ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        } catch (IOException ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }//GEN-LAST:event_btnCargarVhdActionPerformed

    private void
_btnCargarBitActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnCargarBitActionPerformed
        try {
            File fichero_bit = new
File(this.RUTA_PROYECTO_ACTUAL+"\\download.bit");
            if (fichero_bit.exists()) {
                if (this.cargarBit(fichero_bit.toString())) {
                    _btnCargarTB.setEnabled(true);
                } else {
                    JOptionPane.showMessageDialog(this, "No se ha
podido Cargar el bitstream correctamente", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(this, "El bitstream no
ha sido generado aún.", "Información",
JOptionPane.INFORMATION_MESSAGE);
            }
        } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }//GEN-LAST:event_btnCargarBitActionPerformed
```

```
    private void
_btnEjecutarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnEjecutarActionPerformed
        try {
            if (this.ejecutar()) {

            } else {
                JOptionPane.showMessageDialog(this, "Ejecucion sin
terminar", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }//GEN-LAST:event__btnEjecutarActionPerformed

    private void
_btnCargarTBActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnCargarTBActionPerformed
        try {
            if (this.cargarTB()) {
                _btnCargarGolden.setEnabled(true);
                _btnGenerarGolden.setEnabled(true);
            } else {
                JOptionPane.showMessageDialog(this, "Error al cargar
el testbench", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }//GEN-LAST:event__btnCargarTBActionPerformed

    private void _btnClearActionPerformed(java.awt.event.ActionEvent
evt) {

        if (jTabbedPane1.getComponentCount() > 0) {
            javax.swing.JPanel panel = (javax.swing.JPanel)
jTabbedPane1.getSelectedComponent();
            javax.swing.JScrollPane scrPanel =
(javax.swing.JScrollPane) panel.getComponent(0);
            javax.swing.JViewport viewPort = (javax.swing.JViewport)
scrPanel.getComponent(0);
            javax.swing.JTextArea txtArea = (javax.swing.JTextArea)
viewPort.getComponent(0);
            txtArea.setText("");
        }
    }

    private void formWindowClosed(java.awt.event.WindowEvent evt)
{//GEN-FIRST:event_formWindowClosed

    }//GEN-LAST:event_formWindowClosed
```

```
private void
_btnGenerarGoldenActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnGenerarGoldenActionPerformed
    try {
        if (this.generarGolden()) {
            _btnEjecutar.setEnabled(true);
            _btnReconfiguracion.setEnabled(true);
        } else {
            JOptionPane.showMessageDialog(this, "No se ha podido
generar el golden correctamente", "Error", JOptionPane.ERROR_MESSAGE);
        }
    } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
    }
}//GEN-LAST:event__btnGenerarGoldenActionPerformed

private void
_btnCargarGoldenActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnCargarGoldenActionPerformed
    try {
        if (this.cargarGolden()) {
            _btnEjecutar.setEnabled(true);
            _btnReconfiguracion.setEnabled(true);
        } else {
            JOptionPane.showMessageDialog(this, "No se ha podido
cargar el golden correctamente", "Error", JOptionPane.ERROR_MESSAGE);
        }
    } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
    }
}//GEN-LAST:event__btnCargarGoldenActionPerformed

private void menuConfigNessyActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuConfigNessyActionPerformed
        String rutaISE = "";
        String rutaEDK = "";
        if (RUTA_ISE != null) {
            rutaISE = RUTA_ISE;
        }
        if (RUTA_EDK != null) {
            rutaEDK = RUTA_EDK;
        }
        GUIConfig config = new GUIConfig(this, true, rutaISE,
rutaEDK);
        config.setVisible(true);
        try {
            InputStream is = null;
            Properties prop = new Properties();
            is = new
FileInputStream(RUTA_NESSY+"\\conf\\config.properties");
            prop.load(is);
            RUTA_ISE = prop.getProperty("HomeXilinxISE");
            RUTA_EDK = prop.getProperty("HomeXilinxEDK");
```

```
        } catch (IOException ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
}//GEN-LAST:event_menuConfigNessyActionPerformed

private void
menuConfigFichConfActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuConfigFichConfActionPerformed

    JFileChooser chooser;
    chooser = new JFileChooser();
    Filtro filter = new Filtro("properties");
    chooser.addChoosableFileFilter(filter);
    chooser.setCurrentDirectory(new java.io.File("."));
    chooser.setDialogTitle("Seleccionar Archivo Configuración");
    chooser.setAcceptAllFileFilterUsed(false);
    chooser.setMultiSelectionEnabled(false);
    ficherosVHD = new ArrayList<File>();
    if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
        config(chooser.getSelectedFile().getAbsolutePath(), true);

    }

}//GEN-LAST:event_menuConfigFichConfActionPerformed

private void
_btnReconfiguracionActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnReconfiguracionActionPerformed
    try {
        if (this.reconfiguracion()) {

        } else {
            JOptionPane.showMessageDialog(this, "Inyeccion de errores
finalizada incorrectamente" , "Error", JOptionPane.ERROR_MESSAGE);
        }
    } catch (Exception ex) {

java.util.logging.Logger.getLogger(GUIPrincipal.class.getName()).log(j
ava.util.logging.Level.SEVERE, null, ex);
    }
}//GEN-LAST:event__btnReconfiguracionActionPerformed

private void formWindowClosing(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_formWindowClosing

}//GEN-LAST:event_formWindowClosing

private void _btnCrearBitActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btnCrearBitActionPerformed
    try {
        if (this.crearBit()) {
             _btnCargarBit.setEnabled(true);
        } else {
            JOptionPane.showMessageDialog(this, "No se ha podido
generar el .bit correctamente", "Error", JOptionPane.ERROR_MESSAGE);
```

```
        }
    } catch (Exception ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
    }
}//GEN-LAST:event_btnCrearBitActionPerformed

private void
_menuOpcionesCerrarNessyActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuOpcionesCerrarNessyActionPerformed
    System.exit(0);
}//GEN-LAST:event_menuOpcionesCerrarNessyActionPerformed

private void
_menuOpcionesCerrarProyectoActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuOpcionesCerrarProyectoActionPerformed
    _btnCargarVhd.setEnabled(false);
    _btnCrearBit.setEnabled(false);
    _btnCargarBit.setEnabled(false);
    _btnCargarTB.setEnabled(false);
    _btnGenerarGolden.setEnabled(false);
    _btnCargarGolden.setEnabled(false);
    _btnEjecutar.setEnabled(false);
    _btnReconfiguracion.setEnabled(false);
    _menuOpcionesCerrarProyecto.setEnabled(false);
    _TxtEntityVHD.setText("");
    _TextCrearBit.setText("");
    _TextCargarbit.setText("");
    _TextTB.setText("");
    _TextGolden.setText("");
    _TextSalida.setText("");
    _TextErroresReconfiguracion.setText("");
    _comentariosProyecto.setText("Ningun proyecto Nessy abierto");
    RUTA_PROYECTO_ACTUAL = null;
    RUTA_PROYECTO_ACTUAL_EDK = null;
    RUTA_PROYECTO_ACTUAL_SALIDAS = null;
}//GEN-LAST:event_menuOpcionesCerrarProyectoActionPerformed

private void
_menuOpcionesAbrirProyectoActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuOpcionesAbrirProyectoActionPerformed
    JFileChooser chooser;
    chooser = new JFileChooser();
    chooser.setCurrentDirectory(new java.io.File("."));
    chooser.setDialogTitle("Selecciona el proyecto Nessy que desea
abrir");
    chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
    chooser.setFileFilter(new FileNameExtensionFilter("Proyectos nessy
(.nessy)","nessy"));
    chooser.setAcceptAllFileFilterUsed(false);
    if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION){
       RUTA_PROYECTO_ACTUAL = chooser.getSelectedFile().getParent();
       RUTA_PROYECTO_ACTUAL_EDK = RUTA_PROYECTO_ACTUAL+"\\edk";
       RUTA_PROYECTO_ACTUAL_SALIDAS =
RUTA_PROYECTO_ACTUAL+"\\salidas";
       RUTA_PROYECTO_ACTUAL_VHDL = RUTA_PROYECTO_ACTUAL+"\\vhdl";
```

```
        deshabilitarBotones(true);
        _btnCargarVhd.setEnabled(true);
        _menuOpcionesCerrarProyecto.setEnabled(true);
        File proyecto = new
File(RUTA_PROYECTO_ACTUAL+"\\Proyecto.nessy");
        //File infoReconfig = new File(RUTA_PROYECTO_ACTUAL +
"\\infoReconf");
        //if(proyecto.exists()){
            try {
                BufferedReader br2 = new BufferedReader(new
FileReader(RUTA_PROYECTO_ACTUAL + "\\Proyecto.nessy"));
                String s;
                //Leer la 1º línea, en caso de que exista, es que ya se
han seleccionado los .vhd y hay un TOP
                s = br2.readLine();
                if(s!=null){
                    //TOP
                    ficheroEntidadVHD = RUTA_PROYECTO_ACTUAL_VHDL +
"\\" + s;

                    //Vhdls
                    File vhdls = new File(RUTA_PROYECTO_ACTUAL_VHDL);
                    String[] l = vhdls.list();
                    stringsVHD.clear();
                    for(int i=0;i<l.length;i++){
                        stringsVHD.add(l[i]);
                    }
                    this.cargarVHDL();

                    //Leer las 4 siguientes líneas, en caso de que
exista el bitstream, contienen la información sobre los slices
                    File bitstream = new
File(RUTA_PROYECTO_ACTUAL+"\\download.bit");
                    if(bitstream.exists()){
                        nsliceX=Integer.parseInt(br2.readLine());
                        nsliceY=Integer.parseInt(br2.readLine());
                        _btnCargarBit.setEnabled(true);
                    //Comprobar que el proceso de síntesis no haya
terminado mal y haya metainformación referente a las slices sin tener
                    //generado el bitstream. Borra y crea el archivo
.nessy con solo la información sobre el TOP
                    }else if(br2.readLine()!=null){
                        proyecto.delete();
                        proyecto.createNewFile();
                        BufferedWriter bw = new BufferedWriter(new
FileWriter(RUTA_PROYECTO_ACTUAL+"\\Proyecto.nessy"));
                        bw.write(s+"\n");
                        bw.close();
                    }

                }
                br2.close();
            } catch (IOException ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
```

```
            }
           _btnCrearBit.setEnabled(true);


       _comentariosProyecto.setText("Abierto el proyecto
"+RUTA_PROYECTO_ACTUAL);
    }
}//GEN-LAST:event__menuOpcionesAbrirProyectoActionPerformed

private void
_menuOpcionesNuevoProyectoActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__menuOpcionesNuevoProyectoActionPerformed
    String sel="";
    GUINuevoProyecto GuiProyecto = new GUINuevoProyecto(this,
true,sel);
    GuiProyecto.setLocationRelativeTo(null);
    GuiProyecto.setVisible(true);
    if (GuiProyecto.getSeleccion().equals("ok")){
            try {
                RUTA_PROYECTO_ACTUAL = GuiProyecto.getDirectorio() +
"\\" + GuiProyecto.getNombreProyecto();
                RUTA_PROYECTO_ACTUAL_EDK = RUTA_PROYECTO_ACTUAL +
"\\edk";
                RUTA_PROYECTO_ACTUAL_SALIDAS = RUTA_PROYECTO_ACTUAL +
"\\salidas";
                RUTA_PROYECTO_ACTUAL_VHDL = RUTA_PROYECTO_ACTUAL +
"\\vhdl";
                //Crea la estructura del proyecto
                File directorio = new File(RUTA_PROYECTO_ACTUAL);
                directorio.mkdir();
                File directorio2 = new File(RUTA_PROYECTO_ACTUAL_EDK);
                directorio2.mkdir();
                File directorio4 = new
File(RUTA_PROYECTO_ACTUAL_SALIDAS);
                directorio4.mkdir();
                File directorio5 = new
File(RUTA_PROYECTO_ACTUAL_VHDL);
                directorio5.mkdir();
                File proyecto = new File(RUTA_PROYECTO_ACTUAL +
"\\Proyecto.nessy");
                proyecto.createNewFile();
                deshabilitarBotones(true);
                _btnCargarVhd.setEnabled(true);
                _menuOpcionesCerrarProyecto.setEnabled(true);
                _comentariosProyecto.setText("Abierto el proyecto " +
RUTA_PROYECTO_ACTUAL);
            } catch (IOException ex) {

Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
            }
    }

}//GEN-LAST:event__menuOpcionesNuevoProyectoActionPerformed

private void menuConfigComActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuConfigComActionPerformed
```

```
   configurarPuertoSerie();
}//GEN-LAST:event_menuConfigComActionPerformed

private void menuConfigActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuConfigActionPerformed
    // TODO add your handling code here:
}//GEN-LAST:event_menuConfigActionPerformed

private void menuOpcionesActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuOpcionesActionPerformed
    // TODO add your handling code here:
}//GEN-LAST:event_menuOpcionesActionPerformed

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JTextArea _TextCargarbit;
    private javax.swing.JTextArea _TextCrearBit;
    private javax.swing.JTextArea _TextErroresReconfiguracion;
    private javax.swing.JTextArea _TextGolden;
    private javax.swing.JTextArea _TextSalida;
    private javax.swing.JTextArea _TextTB;
    private javax.swing.JTextArea _TxtEntityVHD;
    private javax.swing.JButton _btnCargarBit;
    private javax.swing.JButton _btnCargarGolden;
    private javax.swing.JButton _btnCargarTB;
    private javax.swing.JButton _btnCargarVhd;
    private javax.swing.JButton _btnClear;
    private javax.swing.JButton _btnCrearBit;
    private javax.swing.JButton _btnEjecutar;
    private javax.swing.JButton _btnGenerarGolden;
    private javax.swing.JButton _btnReconfiguracion;
    private javax.swing.JLabel _comentariosProyecto;
    private javax.swing.JMenuItem _menuOpcionesAbrirProyecto;
    private javax.swing.JMenuItem _menuOpcionesCerrarNessy;
    private javax.swing.JMenuItem _menuOpcionesCerrarProyecto;
    private javax.swing.JMenuItem _menuOpcionesNuevoProyecto;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JScrollPane jScrollPane6;
    private javax.swing.JScrollPane jScrollPane7;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JToolBar jToolBar1;
    private javax.swing.JMenu menuConfig;
    private javax.swing.JMenuItem menuConfigCom;
    private javax.swing.JMenuItem menuConfigFichConf;
    private javax.swing.JMenuItem menuConfigNessy;
    private javax.swing.JMenu menuOpciones;
    private javax.swing.JPanel panelCargar;
    private javax.swing.JPanel panelCrearBit;
    private javax.swing.JPanel panelErroresReconfiguracion;
    private javax.swing.JPanel panelGolden;
```

```
    private javax.swing.JPanel panelOutput;
    private javax.swing.JPanel panelTB;
    private javax.swing.JPanel panelVHD;
    // End of variables declaration//GEN-END:variables

}
```

C.1.3. GUISeleccionTop.java

```
/*
 * GUISeleccionTop.java
 *
 * Created on 10 de mayo de 2010, 18:34
 */

package nessy20;

import java.io.File;
import java.util.ArrayList;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

/**
 * Clase para la elección de TOP entre varios VHD
 *
 * @author  David,Carlos y Tony
 */
public class GUISeleccionTop extends javax.swing.JDialog {

    /**
     * Conjunto de ficheros VHD
     */
    private ArrayList<String> ficheros;

    /**
     * Componente que hace la llamada
     */
    private JFrame padre;

    /** Constructor de form GUISeleccionTop
     * @param parent Frame padre encargado de la llamada.
     * @param modal  Indica si es modal o no.
     * @param fich   ArrayList de ficheros seleccionados.
     */
    @SuppressWarnings("empty-statement")
    public GUISeleccionTop(java.awt.Frame parent, boolean
modal,ArrayList<String> fich) {
        super(parent, modal);
        initComponents();

        padre=(JFrame) parent;
        ficheros=fich;
        DefaultTableModel dtm= (DefaultTableModel) jTable1.getModel();
```

```
        for(int i=0;i<ficheros.size();i++)
        {
          Object[] row={ficheros.get(i),false};
          dtm.insertRow(i, row);
        }

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jButton1 = new javax.swing.JButton();
        _btn_AddVHDL = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("Selección de fichero VHDL como Top");
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt)
{
                formWindowClosing(evt);
            }
        });

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "VHDL", "Top"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.Boolean.class
            };
            boolean[] canEdit = new boolean [] {
                false, true
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }
```

```
            public boolean isCellEditable(int rowIndex, int
columnIndex) {
                return canEdit [columnIndex];
            }
        });
        jTable1.setColumnSelectionAllowed(true);
        jScrollPane1.setViewportView(jTable1);

jTable1.getColumnModel().getSelectionModel().setSelectionMode(javax.sw
ing.ListSelectionModel.SINGLE_SELECTION);
        jTable1.getColumnModel().getColumn(0).setResizable(false);
        jTable1.getColumnModel().getColumn(0).setPreferredWidth(300);
        jTable1.getColumnModel().getColumn(1).setResizable(false);
        jTable1.getColumnModel().getColumn(1).setPreferredWidth(30);

        jButton1.setText("Ok");
        jButton1.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                jButton1ActionPerformed(evt);
            }
        });

        _btn_AddVHDL.setText("Añadir VHDL");
        _btn_AddVHDL.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btn_AddVHDLActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(37, 37, 37)
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 373,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(_btn_AddVHDL))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(239, 239, 239)
                        .addComponent(jButton1)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
```

```
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(31, 31, 31)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                    .addComponent(_btn_AddVHDL)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 139,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(32, 32, 32)
                        .addComponent(jButton1)))
                .addContainerGap(39, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed


    int selecciones=0;
    int pos=0;
    DefaultTableModel dtm= (DefaultTableModel) jTable1.getModel();
    int i;
    for(i=0;i<ficheros.size();i++)
    {
        if((Boolean)dtm.getValueAt(i,1))
        {
            selecciones ++;
            pos=i;
        }
    }
```

```
    if(selecciones ==1)
    {
        ((GUIPrincipal)padre).setTop(pos);
        this.dispose();
    }
    else
        if(selecciones >1)
            JOptionPane.showMessageDialog(this, "Elija solo un fichero
como Top" +
                    " VHDL", "Error", JOptionPane.ERROR_MESSAGE);
        else//GEN-LAST:event_jButton1ActionPerformed
           JOptionPane.showMessageDialog(this, "Debe seleccionar un
fichero como Top" +
                    " VHDL", "Error", JOptionPane.ERROR_MESSAGE);

}
private void _btn_AddVHDLActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btn_AddVHDLActionPerformed

        JFileChooser chooser;
        chooser = new JFileChooser();
        chooser.setMultiSelectionEnabled(true);
        Filtro filter = new Filtro("vhd");
        chooser.addChoosableFileFilter(filter);
        chooser.setCurrentDirectory(new java.io.File("."));
        chooser.setDialogTitle("Seleccionar Archivos VHDL");
        chooser.setAcceptAllFileFilterUsed(false);
        String fichero;
        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            //try {
            File[] files;

            files=chooser.getSelectedFiles();

            for(int i=0; i<files.length;i++)
            {
                ficheros.add(files[i].getName());
                ((GUIPrincipal)padre).getFiles().add(files[i]);

            }
        }
        DefaultTableModel dtm= (DefaultTableModel) jTable1.getModel();
        int numFilas = dtm.getRowCount();
        for(int i=0;i<numFilas;i++)
        {
            dtm.removeRow(0);
        }

        for(int i=0;i<ficheros.size();i++)
        {
          Object[] row={ficheros.get(i),false};
          dtm.addRow( row);
        }


}//GEN-LAST:event__btn_AddVHDLActionPerformed
```

```
private void formWindowClosing(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_formWindowClosing
    ((GUIPrincipal)padre).setCerradoTop(true);
}//GEN-LAST:event_formWindowClosing


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton _btn_AddVHDL;
    private javax.swing.JButton jButton1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    // End of variables declaration//GEN-END:variables

}
```

C.1.4. GUICargaVHDL.java

```
/*
 * GUICargaVHDL.java
 *
 * Created on 10 de mayo de 2010, 15:35
 */

package nessy20;

import javax.swing.JFrame;

/**
 * Interfaz gráfica para elegir si se carga un sólo fichero VHD o
varios
 *
 * @author  David,Carlos y Tony
 */
public class GUICargaVHDL extends javax.swing.JDialog {

    /**
     * Tipo de selección que se hará. En este caso SeleccionCargaVHD
     */
    private Seleccion sel;


    /** Constructor del form GUICargaVHDL.
     * @param jf  Frame padre encargado de la llamada.
     * @param sel Tipo de Seleccion a elegir.
     * @param bol Indica si es modal o no.
     */
    public GUICargaVHDL(JFrame jf,boolean bol,Seleccion sel) {
        super(jf,bol);
        initComponents();
        this.sel=sel;

    }

    /** This method is called from within the constructor to
```

```
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        jPanel1 = new javax.swing.JPanel();
        _btn_CargarTop = new javax.swing.JRadioButton();
        _btn_CargarVariosVHDL = new javax.swing.JRadioButton();
        _btnOK = new javax.swing.JButton();
        _btnCancelar = new javax.swing.JButton();

        setTitle("Cargar VHDL");
        setResizable(false);

        _btn_CargarTop.setSelected(true);
        _btn_CargarTop.setText("Cargar un VHDL ( Top)");
        buttonGroup1.add(_btn_CargarTop);
        _btn_CargarTop.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btn_CargarTopActionPerformed(evt);
            }
        });

        _btn_CargarVariosVHDL.setText("Cargar Varios VHDL y
Seleccionar Top");
        buttonGroup1.add(_btn_CargarVariosVHDL);

        _btnOK.setText("OK");
        _btnOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnOKActionPerformed(evt);
            }
        });

        _btnCancelar.setText("Cancelar");
        _btnCancelar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCancelarActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
```

```
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap(77, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                    .addComponent(_btn_CargarTop)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.TRAILING, false)

.addGroup(jPanel1Layout.createSequentialGroup()
                            .addComponent(_btnOK,
javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(_btnCancelar))
                        .addComponent(_btn_CargarVariosVHDL,
javax.swing.GroupLayout.Alignment.LEADING)))
                .addGap(88, 88, 88))
        );

        jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL,
new java.awt.Component[] {_btnCancelar, _btnOK});

        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(57, 57, 57)
                .addComponent(_btn_CargarTop)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(_btn_CargarVariosVHDL)
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.BASELINE)
                    .addComponent(_btnCancelar)
                    .addComponent(_btnOK))
                .addContainerGap(26, Short.MAX_VALUE))
        );

        jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL,
new java.awt.Component[] {_btnCancelar, _btnOK});

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

private void _btn_CargarTopActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btn_CargarTopActionPerformed
// TODO add your handling code here:
}//GEN-LAST:event__btn_CargarTopActionPerformed

private void _btnOKActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnOKActionPerformed

    if(_btn_CargarTop.isSelected())
    {
        sel.seleccion=SeleccionCargaVHD.SELECCION_VHDL_TOP;
    }
    else
    {
        if(_btn_CargarVariosVHDL.isSelected())
        {
            sel.seleccion=SeleccionCargaVHD.SELECCION_VARIOS_VHDL;
        }
        else
            sel.seleccion=SeleccionCargaVHD.NADA;
    }
    this.dispose();
}//GEN-LAST:event__btnOKActionPerformed

private void _btnCancelarActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btnCancelarActionPerformed
    this.setVisible(false);
}//GEN-LAST:event__btnCancelarActionPerformed



    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton _btnCancelar;
```

```
    private javax.swing.JButton _btnOK;
    private javax.swing.JRadioButton _btn_CargarTop;
    private javax.swing.JRadioButton _btn_CargarVariosVHDL;
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration//GEN-END:variables

}
```

C.1.5. GUINuevoProyecto.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * GUINuevoProyecto.java
 *
 * Created on 28-dic-2010, 22:10:19
 */

package nessy20;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

/**
 *
 * @author Ruben
 */
public class GUINuevoProyecto extends javax.swing.JDialog {
    private String directorio;
    private String nombreProyecto;
    private String seleccion;

    /** Creates new form GUINuevoProyecto */
    public GUINuevoProyecto(java.awt.Frame parent, boolean
modal,String seleccion) {
        super(parent, modal);
        initComponents();
        this.seleccion = seleccion;
    }

    public String getDirectorio(){
        return directorio;
    }

    public String getNombreProyecto(){
        return nombreProyecto;
    }

    public String getSeleccion(){
        return seleccion;
    }
```

```
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        _btn_ok = new javax.swing.JButton();
        _btn_cancel = new javax.swing.JButton();
        _lbl_directorio = new javax.swing.JLabel();
        _lbl_nombreProyecto = new javax.swing.JLabel();
        _txt_directorio = new javax.swing.JTextField();
        _txt_nombreProyecto = new javax.swing.JTextField();
        _btn_directorio = new javax.swing.JButton();

        jLabel1.setText("jLabel1");

        jLabel2.setText("jLabel2");


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("Nuevo Proyecto Nessy");

        jLabel3.setFont(new java.awt.Font("Tahoma", 1, 18));

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel3.setText("Nuevo Proyecto Nessy");

        _btn_ok.setText("OK");
        _btn_ok.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btn_okActionPerformed(evt);
            }
        });

        _btn_cancel.setText("Cancel");
        _btn_cancel.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btn_cancelActionPerformed(evt);
            }
        });

        _lbl_directorio.setText("Directorio Proyecto:");

        _lbl_nombreProyecto.setText("Nombre del Proyecto");
```

```
        _btn_directorio.setText("...");
        _btn_directorio.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btn_directorioActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(72, 72, 72)
                .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 247,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(79, Short.MAX_VALUE))
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(50, 50, 50)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                            .addComponent(_lbl_nombreProyecto)
                            .addComponent(_lbl_directorio))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(_btn_ok,
javax.swing.GroupLayout.PREFERRED_SIZE, 47,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(_txt_directorio,
javax.swing.GroupLayout.PREFERRED_SIZE, 141,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                    .addComponent(_btn_directorio,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addComponent(_txt_nombreProyecto,
javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGroup(layout.createSequentialGroup()
                            .addGap(78, 78, 78)
                            .addComponent(_btn_cancel)))
                    .addGap(57, 57, 57))
            );
            layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(19, 19, 19)
                    .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(48, 48, 48)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                        .addComponent(_lbl_directorio)
                        .addComponent(_btn_directorio)
                        .addComponent(_txt_directorio,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                        .addComponent(_lbl_nombreProyecto)
                        .addComponent(_txt_nombreProyecto,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(27, 27, 27)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                        .addComponent(_btn_ok)
                        .addComponent(_btn_cancel))
                    .addContainerGap())
            );

            pack();
        }// </editor-fold>//GEN-END:initComponents

        private void _btn_okActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btn_okActionPerformed
            if(_txt_directorio.getText().equals("") ||
_txt_nombreProyecto.getText().equals("")){
```

```
            JOptionPane.showMessageDialog(this, "Tienes que introducir
el directorio y el nombre de proyecto", "Error" ,
JOptionPane.INFORMATION_MESSAGE);
        }
        else{
            directorio = _txt_directorio.getText();
            nombreProyecto = _txt_nombreProyecto.getText();
            seleccion = "ok";
            this.dispose();
        }
    }//GEN-LAST:event__btn_okActionPerformed

    private void
_btn_directorioActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btn_directorioActionPerformed
        JFileChooser chooser;
        chooser = new JFileChooser();
        chooser.setCurrentDirectory(new java.io.File("."));
        chooser.setDialogTitle("Selecciona el directorio donde se
creará el proyecto Nessy");
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        chooser.setAcceptAllFileFilterUsed(false);
        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            directorio = chooser.getSelectedFile().getPath();
            _txt_directorio.setText(directorio);
        }
    }//GEN-LAST:event__btn_directorioActionPerformed

    private void _btn_cancelActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btn_cancelActionPerformed
        seleccion="cancel";
        this.setVisible(false);
        this.dispose();
    }//GEN-LAST:event__btn_cancelActionPerformed


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton _btn_cancel;
    private javax.swing.JButton _btn_directorio;
    private javax.swing.JButton _btn_ok;
    private javax.swing.JLabel _lbl_directorio;
    private javax.swing.JLabel _lbl_nombreProyecto;
    private javax.swing.JTextField _txt_directorio;
    private javax.swing.JTextField _txt_nombreProyecto;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    // End of variables declaration//GEN-END:variables

}
```

C.1.6. GUIConfig.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package nessy20;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
import javax.swing.JFileChooser;

/**
 *
 *
 * Interfaz gráfica para la configuración de la ubicación de Xilinx
 *
 * Created on 04-jun-2010, 16:02:42
 */
public class GUIConfig extends javax.swing.JDialog {

    private String rutaXilinxISE;
    private String rutaXilinxEDK;
    private boolean cierre;

    /** Creates new form GUIConfig */
    public GUIConfig(java.awt.Frame parent, boolean modal,String
rutaISE,String rutaEDK) {
        super(parent, modal);
        rutaXilinxISE=rutaISE;
        rutaXilinxEDK=rutaEDK;
        initComponents();
        cierre=false;

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        _lbl_HomeXilinx_ISE = new javax.swing.JLabel();
        _txt_HomeXilinx_ISE = new javax.swing.JTextField();
        _btn_HomeXilinx_ISE = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        _btnOK = new javax.swing.JButton();
        _btnCancelar = new javax.swing.JButton();
        _txt_HomeXilinx_EDK = new javax.swing.JTextField();
        _lbl_HomeXilinx_EDK = new javax.swing.JLabel();
        _btn_HomeXilinx_EDK = new javax.swing.JButton();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("Configuración Nessy");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt)
{
                formWindowClosing(evt);
            }
        });

        _lbl_HomeXilinx_ISE.setText("Home Xilinx ISE :");

        _txt_HomeXilinx_ISE.setText(rutaXilinxISE);
        _txt_HomeXilinx_ISE.setEditable(false);

        _btn_HomeXilinx_ISE.setText("Seleccion");
        _btn_HomeXilinx_ISE.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btn_HomeXilinx_ISEActionPerformed(evt);
            }
        });

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 18));
        jLabel1.setText("Configuración");

        _btnOK.setText("OK");
        _btnOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnOKActionPerformed(evt);
            }
        });

        _btnCancelar.setText("Cancelar");
        _btnCancelar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCancelarActionPerformed(evt);
            }
        });

        _txt_HomeXilinx_EDK.setText(rutaXilinxEDK);
        _txt_HomeXilinx_EDK.setEditable(false);

        _lbl_HomeXilinx_EDK.setText("Home Xilinx EDK :");

        _btn_HomeXilinx_EDK.setText("Seleccion");
        _btn_HomeXilinx_EDK.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btn_HomeXilinx_EDKActionPerformed(evt);
            }
```

```
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap(18, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                            .addComponent(_lbl_HomeXilinx_ISE,
javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(_lbl_HomeXilinx_EDK,
javax.swing.GroupLayout.PREFERRED_SIZE, 99,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(60, 60, 60)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING, false)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                .addComponent(_txt_HomeXilinx_ISE,
javax.swing.GroupLayout.PREFERRED_SIZE, 318,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(_btn_HomeXilinx_ISE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                .addComponent(_txt_HomeXilinx_EDK)
                                .addGap(18, 18, 18)
                                .addComponent(_btn_HomeXilinx_EDK))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(27, 27, 27)
                                .addComponent(_btnOK,
javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(41, 41, 41)
                                .addComponent(_btnCancelar))))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(239, 239, 239)
                        .addComponent(jLabel1)))
                .addContainerGap(20, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```

```
                .addGap(19, 19, 19)
                .addComponent(jLabel1)
                .addGap(31, 31, 31)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                    .addComponent(_txt_HomeXilinx_ISE,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(_lbl_HomeXilinx_ISE)
                    .addComponent(_btn_HomeXilinx_ISE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                    .addComponent(_txt_HomeXilinx_EDK,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(_lbl_HomeXilinx_EDK)
                    .addComponent(_btn_HomeXilinx_EDK))
                .addGap(26, 26, 26)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                    .addComponent(_btnCancelar)
                    .addComponent(_btnOK))
                .addContainerGap(26, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void
_btn_HomeXilinx_ISEActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btn_HomeXilinx_ISEActionPerformed
        JFileChooser chooser;
        chooser = new JFileChooser();
        chooser.setCurrentDirectory(new java.io.File("C:/"));
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        chooser.setMultiSelectionEnabled(false);
        chooser.setDialogTitle("Seleccionar Home Xilinx ISE");
        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            rutaXilinxISE=chooser.getSelectedFile().getAbsolutePath();
             _txt_HomeXilinx_ISE.setText(rutaXilinxISE);
        } else {
            //log.info("Seleccion no llevada a cabo");
        }
    }//GEN-LAST:event__btn_HomeXilinx_ISEActionPerformed

    private void _btnOKActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnOKActionPerformed
        try {
            //TODO cargar configuracion en archivos.Properties prop =
new Properties();
```

```
            InputStream is = null;
            Properties prop = new Properties();
            prop.setProperty("HomeXilinxISE", rutaXilinxISE);
            prop.setProperty("HomeXilinxEDK", rutaXilinxEDK);
            prop.store(new FileOutputStream("conf/config.properties"),
"rutas");
            this.dispose();
        } catch (IOException ex) {
            System.out.println("Fallo");
        }
}//GEN-LAST:event__btnOKActionPerformed

    private void
_btnCancelarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnCancelarActionPerformed
        this.setVisible(false);
        this.dispose();

}//GEN-LAST:event__btnCancelarActionPerformed

    private void formWindowClosing(java.awt.event.WindowEvent evt)
{//GEN-FIRST:event_formWindowClosing
       cierre=true;
    }//GEN-LAST:event_formWindowClosing

    private void
_btn_HomeXilinx_EDKActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btn_HomeXilinx_EDKActionPerformed
        JFileChooser chooser;
        chooser = new JFileChooser();
        chooser.setCurrentDirectory(new java.io.File("C:/"));
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        chooser.setMultiSelectionEnabled(false);
        chooser.setDialogTitle("Seleccionar Home Xilinx EDK");
        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            rutaXilinxEDK=chooser.getSelectedFile().getAbsolutePath();
            _txt_HomeXilinx_EDK.setText(rutaXilinxEDK);
        } else {
            //log.info("Seleccion no llevada a cabo");
        }
    }//GEN-LAST:event__btn_HomeXilinx_EDKActionPerformed



    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton _btnCancelar;
    private javax.swing.JButton _btnOK;
    private javax.swing.JButton _btn_HomeXilinx_EDK;
    private javax.swing.JButton _btn_HomeXilinx_ISE;
    private javax.swing.JLabel _lbl_HomeXilinx_EDK;
    private javax.swing.JLabel _lbl_HomeXilinx_ISE;
    private javax.swing.JTextField _txt_HomeXilinx_EDK;
    private javax.swing.JTextField _txt_HomeXilinx_ISE;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables
```

```
    boolean getCierre() {
      return cierre;
    }

}
```

C.1.7. GUIComConfig.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * GUIcomConfig.java
 *
 * Created on 26-nov-2010, 20:39:37
 */

package nessy20;

/**
 *
 * @author Felipe
 */
public class GUIComConfig extends javax.swing.JDialog {

    private String com;

    /** Creates new form GUIcomConfig */
    public GUIComConfig(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    public String getSel(){return com;}

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        _btnOk = new javax.swing.JButton();
        _cmbCom = new javax.swing.JComboBox();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("Selección de puerto");

        _btnOk.setText("Ok");
```

```
        _btnOk.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnOkActionPerformed(evt);
            }
        });

        _cmbCom.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "COM1", "COM2", "COM3", "COM4", "COM5", "COM6", "COM7",
"COM8", "COM9", "COM10" }));
        _cmbCom.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _cmbComActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(32, 32, 32)
                .addComponent(_cmbCom,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(29, 29, 29)
                .addComponent(_btnOk)
                .addContainerGap(31, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(32, 32, 32)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.TRAILING)
                    .addComponent(_btnOk)
                    .addComponent(_cmbCom,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(38, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void _cmbComActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__cmbComActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event__cmbComActionPerformed
```

```
    private void _btnOkActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnOkActionPerformed
        com = _cmbCom.getSelectedItem().toString();
        this.dispose();
    }//GEN-LAST:event_btnOkActionPerformed



    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton _btnOk;
    private javax.swing.JComboBox _cmbCom;
    // End of variables declaration//GEN-END:variables

}
```

C.1.8. GUIInyeccionErrores.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * GuiInyeccionErrores.java
 *
 * Created on 04-dic-2010, 22:16:52
 */

package nessy20;

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Rubén y Felipe
 */
public class GUIInyeccionErrores extends javax.swing.JDialog {

    private int nslicesX;
    private int nslicesY;

    /** Creates new form GuiInyeccionErrores */
    public GUIInyeccionErrores(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }
    public int getnslicesX(){return nslicesX;}
    public int getnslicesY(){return nslicesY;}

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
```

```
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        text_x0 = new javax.swing.JTextField();
        text_y0 = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
        setTitle("Localizacion del circuito");

        jButton1.setText("Ok");
        jButton1.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                jButton1ActionPerformed(evt);
            }
        });

        text_x0.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                text_x0ActionPerformed(evt);
            }
        });

        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14));
        jLabel1.setText("SLICES");

        jLabel2.setText("n° slices ancho(max 12)");

        jLabel4.setText("n° slices alto (max 100)");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(114, 114, 114)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.TRAILING)
                            .addComponent(jLabel1)
```

```
                                .addComponent(jButton1)))
                        .addGroup(layout.createSequentialGroup()
                            .addGap(64, 64, 64)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(jLabel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 119, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                    .addComponent(text_x0,
javax.swing.GroupLayout.PREFERRED_SIZE, 49,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                    .addComponent(jLabel4,
javax.swing.GroupLayout.DEFAULT_SIZE, 115, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                    .addComponent(text_y0,
javax.swing.GroupLayout.PREFERRED_SIZE, 47,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                .addGap(49, 49, 49))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(11, 11, 11)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(text_x0,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(text_y0,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
20, Short.MAX_VALUE)
```

```
                .addComponent(jButton1)
                .addGap(23, 23, 23))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jButton1ActionPerformed
        this.nslicesX = Integer.parseInt(text_x0.getText());
        this.nslicesY = Integer.parseInt(text_y0.getText());
        this.dispose();

    }//GEN-LAST:event_jButton1ActionPerformed

    private void text_x0ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_text_x0ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_text_x0ActionPerformed


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JTextField text_x0;
    private javax.swing.JTextField text_y0;
    // End of variables declaration//GEN-END:variables

}
```

C.1.9. Seleccion.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package nessy20;

/**
 * Clase donde tenemos los tipos de selecciones que se hacen en la
aplicación.
 * Ya sea al elegir el tipo de carga de VHDL's o del TestBench
 * @author David,Tony y Carlos
 */
public class Seleccion {

    /**
     * Seleccion de VHD
     */
    public SeleccionNumIter selIter;

    public int numIter;
```

```
    public SeleccionCargaVHD seleccion;
    /**
     * Seleccion de carga de banco de pruebas
     */

    public SeleccionTB selTB;
    /**
     *
     */

    public Seleccion()
    {
        numIter=1;
        selIter=SeleccionNumIter.NADA;
        seleccion=SeleccionCargaVHD.NADA;
        selTB=SeleccionTB.NADA;
    }
}
```

C.1.10. SeleccionCargaVHDL.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package nessy20;

/**
 * Opciones al seleccionar la Carga los VHDL's en nuestra aplicación.
 * @author David,Tony y Carlos
 */
public enum SeleccionCargaVHD {

    /**
     *
     */
    NADA,
    /**
     *
     */
    SELECCION_VHDL_TOP,
    /**
     *
     */
    SELECCION_VARIOS_VHDL

}
```

C.1.11. SeleccionTB.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package nessy20;

/**
 * Opciones al seleccionar la Carga de un Test Bench en nuestra
aplicación.
 * @author David,Tony y Carlos
 */
public enum SeleccionTB {
    /**
     *
     */
    NADA,
    /**
     *
     */
    CARGA_PANTALLA,
    /**
     *
     */
    CARGA_FICHERO
}
```

C.1.12. SeleccionNumIteraciones.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package nessy20;

/**
 *
 * @author User
 */
public enum SeleccionNumIter { OK, CANCELAR, NADA

}
```

C.1.13. Filtro.java

```
package nessy20;

/**
 *
 * @author Tony,David y Carlos.
 */

import javax.swing.filechooser.*;
import java.io.File;


public class Filtro extends FileFilter {

  String[] extensions;
```

```
  String description;

  /**
   * Constructor de la clase.
   * @param ext Extensión de ficheros que deseamos que aparezcan.
   */
  public Filtro(String ext) {
    this (new String[] {ext}, null);
  }

  /**
   * Constructor de la clase.
   * @param exts Extensiones que deseamos que aparezcan.
   * @param descr Extensión de una de ellas.
   */
  public Filtro(String[] exts, String descr) {
    // Clone and lowercase the extensions
    extensions = new String[exts.length];
    for (int i = exts.length - 1; i >= 0; i--) {
      extensions[i] = exts[i].toLowerCase();
    }
    // Make sure we have a valid (if simplistic) description
    description = (descr == null ? exts[0] + " files" : descr);
  }

  public boolean accept(File f) {

    if (f.isDirectory()) { return true; }
    String name = f.getName().toLowerCase();
    for (int i = extensions.length - 1; i >= 0; i--) {
      if (name.endsWith(extensions[i])) {
        return true;
      }
    }
    return false;
  }

  public String getDescription() { return description; }
}
```

C.2. Paquete procesos

C.2.1. HiloNessy.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package procesos;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.IOException;
```

```
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTextArea;

/**
 *
 * @author Ruben
 */
public class HiloNessy extends Thread {

    private JTextArea text;
    private Process p;
    private String file;

    public HiloNessy(JTextArea text,Process p,String file){
        this.text= text;
        this.p = p;
        this.file = file;

    }

    public void run() {
        String line;
        try {
            FileOutputStream os = new FileOutputStream(file);
            BufferedWriter salida = new BufferedWriter(new
OutputStreamWriter(os));
            BufferedReader is = new BufferedReader(new
InputStreamReader(p.getInputStream()));
            while ((line = is.readLine()) != null){
                this.text.append(line+"\n");
                this.text.setCaretPosition(text.getText().length());
                salida.write(line+"\n");
            }
            salida.close();
        } catch (IOException ex) {

Logger.getLogger(HiloNessy.class.getName()).log(Level.SEVERE, null,
ex);
        }


    }
}
```

C.2.1. HiloProceso.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package procesos;
```

```
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTextArea;
import javax.swing.SwingWorker;

/**
 *
 * @author Felipe
 */
public class HiloProceso extends Thread{
    private String comando;
    private String salida;
    private JTextArea pestaña;
    Semaforo sem;

    public HiloProceso(JTextArea pestaña, String comando, String
salida, Semaforo sem){
        this.comando = comando;
        this.salida = salida;
        this.pestaña = pestaña;
        this.sem = sem;
    }


    public void run(){
        try {
                sem.lock();

pestaña.append("**************************************************\
n");
                pestaña.append("\tLanzando proceso:
"+this.getName()+"\n\n");
                pestaña.setCaretPosition(pestaña.getText().length());
                Process proceso = Runtime.getRuntime().exec(comando);
                if(pestaña!=null){
                    HiloNessy hilo = new
HiloNessy(pestaña,proceso,salida);
                    hilo.setName("Salida
"+Thread.currentThread().getName());
                    hilo.start();
                }
                proceso.waitFor();
                pestaña.append("\n\n\tProceso finalizado\n");

pestaña.append("**************************************************\
n\n\n");
                pestaña.setCaretPosition(pestaña.getText().length());
                sem.unlock();
        } catch (InterruptedException ex) {

Logger.getLogger(HiloProceso.class.getName()).log(Level.SEVERE, null,
ex);
        } catch (IOException ex) {

Logger.getLogger(HiloProceso.class.getName()).log(Level.SEVERE, null,
ex);
```

```
        }
    }
}
```

C.2.3. HiloInyeccion.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package procesos;

import IOFPGA.EntradaSalida;
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTextArea;
import nessy20.GUIPrincipal;

/**
 *
 * @author XPMUser
 */
public class HiloInyeccion extends Thread{
    private JTextArea panel;
    private GUIPrincipal gui;
    private EntradaSalida entsal;
    private BufferedInputStream bi;
    private int tam_bit;
    private int frame_inicio;
    private int frame_fin;
    private int sliceY0;
    private int sliceY1;


    public HiloInyeccion(JTextArea panel, GUIPrincipal gui,
EntradaSalida entsal, BufferedInputStream bi,int tam_bit, int
frame_inicio, int frame_fin, int sliceY0, int sliceY1){
        this.panel = panel;
        this.gui = gui;
        this.entsal = entsal;
        this.bi = bi;
        this.tam_bit = tam_bit;
        this.frame_inicio = frame_inicio;
        this.frame_fin = frame_fin;
        this.sliceY0 = sliceY0;
        this.sliceY1 = sliceY1;

    }
```

```
    @Override
    public void run(){
        enviarBitstream();
        inyectarErrores();
    }


    public void enviarBitstream(){
        int enviados = 0;
        int byteleido = 0;
        String cad = "";
        String texto = "";
        String cabecera = "Enviando bitstream original a la RAM... ";
        String cola = " palabras de 32bits enviadas";
        String palabrasEnviadas;

        byte[] hex = new byte[4];

        try{
            String fichero = gui.RUTA_PROYECTO_ACTUAL_SALIDAS +
"\\bit_parcial.txt";
            BufferedWriter bw = new BufferedWriter(new
FileWriter(fichero));

            //Enviar el bitstream por el puerto serie
            byteleido = bi.read();
            while (byteleido != -1) {
                enviados++;
                cad = cad + gui.intToBinario8(byteleido);
                if (enviados % 4 == 0) {
                    entsal.enviarBinaria(cad);
                    bw.append(cad+"\n");
                    palabrasEnviadas = "("+enviados/4+"/"+tam_bit+")";
                    texto = cabecera + palabrasEnviadas + cola;
                    panel.setText(texto);
                    cad = "";
                }
                byteleido = bi.read();
            }
            // Envía la última palabra, que lo más seguro es que esté
incompleta (<32bits)
            if (!cad.equals("")) {
                entsal.enviarBinaria(cad);
            }
            bi.close();
            bw.close();
            String envioBitstream = cabecera + "Ok\n\n";
            texto = envioBitstream;
            panel.setText(texto);
        } catch (Exception ex) {

Logger.getLogger(HiloInyeccion.class.getName()).log(Level.SEVERE,
null, ex);
            System.out.println("error");
        }
    }
```

```
    public void inyectarErrores(){
        int cicloError = 0;
        int palabra_inicio = 0;
        int palabra_gap_inicio, palabra_gap_fin, gap_inicio, gap_fin =
0;
        int bit_inicio, bit_fin, bit_fin_aux = 0;
        String resultados = "";

        //Variables para tratar los datos obtenidos
        long tiempoInicio = System.currentTimeMillis();
        int numeroConf=0, numeroConfErroneas=0;
        double por_ciento_erroneas;

        String fichero_errores =
GUIPrincipal.RUTA_PROYECTO_ACTUAL_SALIDAS+"\\errores_reconfiguracion.t
xt";

        try {
            BufferedWriter bw = new BufferedWriter(new
FileWriter(fichero_errores));
            //for(int t=0;t<44)

            panel.append("INICIO DE LA RECONFIGURACIÓN\n");
            for(int i = frame_inicio ; i < frame_fin; i++){
                //calculos
                palabra_inicio = (i * 206) + 1;
                gap_inicio = (((159 - sliceY1) / 2) * 80) / 32;
                palabra_gap_inicio = palabra_inicio + 3 + gap_inicio;
                bit_inicio = (32 - ((((159 - sliceY1) / 2) * 80) %
32)) - 1;
                gap_fin = ((((159 - sliceY0) / 2) + 1) * 80) / 32;
                palabra_gap_fin = palabra_inicio + 3 + gap_fin;
                bit_fin = 32 - (((((159 - sliceY0) / 2) + 1) * 80) %
32);
                bit_fin_aux = 0;

                panel.append("Reconfigurando la frame "+i+" ...\n");
                System.out.println("Reconfigurando la frame "+i+"
...");
                panel.setCaretPosition(panel.getText().length());

                for(int j = palabra_gap_inicio; j <= palabra_gap_fin;
j++){
                    if (j == palabra_gap_fin){
                       bit_fin_aux = bit_fin;
                    }
                    for(int k = 0; k <= 31; k++){
                        //RECONFIGURACION
                        numeroConf++;

                        esperar_palabra_syn();
                        String cicloErrorString =
entsal.recibirBinaria(32);
                        cicloError =
gui.traduceString(cicloErrorString);
                        if (cicloError != 0){
```

```
                                numeroConfErroneas++;

bw.write(i+"\t"+j+"\t"+k+"\t"+cicloError+"\n");
                            }
                        }
                        bit_inicio = 31;
                    }

                }

                //RESULTADOS
                long totalTiempo = System.currentTimeMillis() -
tiempoInicio;
                totalTiempo = totalTiempo / 1000; //Convertimos en
segundos, se desprecia el resto
                int segundos = (int)totalTiempo % 60 ;
                totalTiempo = totalTiempo / 60;
                int minutos = (int)totalTiempo % 60;
                int horas = (int)totalTiempo / 60;
                String tiempoFormateado = horas+":"+minutos+":"+segundos;
                por_ciento_erroneas =
(double)numeroConfErroneas/(double)numeroConf*100;
                resultados = "\tTiempo tardado:
\t\t\t"+tiempoFormateado+"\n"
                        + "\tNúmero de reconfiguraciones:
\t\t"+numeroConf+"\n"
                        + "\tNúmero de reconfiguraciones erróneas:
\t"+numeroConfErroneas+"\n"
                        + "\tPorcentaje de errores:
\t\t\t"+por_ciento_erroneas+"%\n";

                panel.append("FIN DE LA RECONFIGURACIÓN\n");
                panel.append(resultados);
                panel.setCaretPosition(panel.getText().length());
                bw.write(resultados);
                bw.close();
            } catch (Exception ex) {

Logger.getLogger(HiloInyeccion.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }

        public String esperar_palabra_syn() throws Exception{
            boolean encontrado = false;
            String dato = "";
            while(!encontrado){
                dato = entsal.recibirBinaria(8);
                // Los dos posibles datos son 1 (correcto) o 2(incorrecto)
                if (dato.equals("11001100"))
                    encontrado = true;
            }
            return dato;
        }

}
```

C.2.4. Semaforo.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package procesos;

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Felipe
 */
public class Semaforo {
    private boolean v;

    public Semaforo(){
        v = true;
    }

    public boolean isLocked(){return !v;};

    public synchronized void unlock(){
        v=true;
        notify();
    }

    public synchronized void lock(){
        try {
            if (isLocked()) {
                wait();
            }
            v = false;
        } catch (InterruptedException ex) {

Logger.getLogger(Semaforo.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

}
```

C.3. Paquete IOFPGA

C.3.1. EntradaSalida.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package IOFPGA;
```

```
import app.Com;
import app.Parameters;
import core.SerialPort;


/**
 *
 * @author Ghadir
 */
public class EntradaSalida {
    /**
     * Parametros de configuración que se introducen al puerto serie
     */
    private Parameters param;
    /**
     * Interfaz para el manejo del puerto RS232
     */
    private Com com;

    private String numPuerto;

    private SerialPort puerto;

    public EntradaSalida(String numPuerto){
        param = null;
        com = null;
        this.numPuerto = numPuerto;
    }

    public Com getCom(){
        return com;
    }

    public void setComNull(){
        com = null;
    }

    /**
     * Función que inicializa el puerto serie necesario para la
comuncicación
     * con la FPGA. Comprueba que esté libre el puerto y que la
maquina sobre
     * la que estamos ejecutando tenga el puerto COM1.
     * @return Cierto si todo ha sido correcto, falso si ha habido
algún error.
     */
    public boolean inicializarPuertoSerie() {
        boolean correcto = true;
        try {
            param = new Parameters();
            param.setPort(numPuerto);
            param.setBaudRate("9600");
            puerto = new SerialPort();
            if (puerto.getStateSerialPortC(numPuerto).equals("free"))
{
                com = new Com(param);
            } else {
```

```
                correcto = false;
            }
        } catch (Exception ex) {
            System.out.println(ex);
            //JOptionPane.showMessageDialog(this, "La aplicación ya se
encuentra ejecutándose, ciérrela para ejecutar nuevamente la
aplicación.", "Info", JOptionPane.INFORMATION_MESSAGE);
            //log.info("La aplicacion ya se encuentra ejecutandose" +
ex);
            System.exit(0);
        }
        return correcto;
    }

    /**
     * Envía una cadena binaria a la FPGA en 4 partes, porque la
instrucción es de 32 bits
     * y solo podemos enviar 8 por cada ciclo
     * @param Cadena de 32 de bits que deseamos enviar.
     */
    public void enviarBinaria(String s) {
        String cad0, cad1, cad2, cad3;
        int dif = 0;
        if (s.length() < 32) {
            dif = 32 - s.length();
        }
        for (int i = 0; i < dif; i++) {
            s = "0" + s; //se añaden 0's por la izquierda
        }
        cad0 = s.substring(0, 8);
        cad1 = s.substring(8, 16);
        cad2 = s.substring(16, 24);
        cad3 = s.substring(24);
        try {
            com.sendSingleData(traduceString(cad0));
            com.sendSingleData(traduceString(cad1));
            com.sendSingleData(traduceString(cad2));
            com.sendSingleData(traduceString(cad3));
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    /**
     * Convierte a entero una una cadena en formato binario
     * @param s Cadena a traducir.
     * @return Entero equivalente al String introducido o -1 si el
formato no es
     * correcto (no son 0's y 1's)
     */
    public int traduceString(String s) {
        int n = 0;
        int peso = 1;
        for (int i = s.length() - 1; i >= 0; i--) {
            if (s.charAt(i) != '0' && s.charAt(i) != '1') {
                return -1;
            }
            if (s.charAt(i) == '1') {
```

```
                n = n + peso;
            }
            peso = peso * 2;
        }
        return n;
    }
    /**
     * Procedimiento que lee de la FPGA un entero y lo transforma a la
cadena de bits.
     * @return Cadena de bits recibida.
     */
    public String recibirBinaria(int numBitsSalida) throws Exception {
        int num;
        String cadenaRecibida = "";
        for (int i = 0; i < numBitsSalida/8; i++) {
            num = com.receiveSingleDataInt();
            cadenaRecibida =cadenaRecibida +
this.convertirByteBinario(num);
        }
        return cadenaRecibida;
    }

    /**
     * Procedimiento que transforma un byte (representado mediante un
entero)
     * a una cadena de ceros y  unos. Utiliza el algoritmo de la
división por 2.
     * @param recibido Byte a traducir.
     * @return Cadena equivalente en bits al byte recibido.
     */
    private String convertirByteBinario(int recibido) {
        String salida = "";
        int numero;
        numero = recibido;
        int long_byte = 8;
        for (int i = 0; i < long_byte; i++) {
            if (numero % 2 == 0) {
                salida = "0" + salida;
            } else {
                salida = "1" + salida;
            }
            numero = numero / 2;
        }
        return salida;
    }


}
```

C.4. Paquete GeneradorVHDL

C.4.1. GeneraVhdl_v2

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package generadorVHDL;

import compiladorEntidad.*;
import java.io.*;
import java.util.Date;

/**
 *
 * @author Ruben
 */
public class GeneraVhdl_V2 {
    /**
     * Entidad que se quiere conectar al módulo de entrada/salida
     */
    private Entidad entidad;

    /**
     * Nombre de la entidad superior. Va a ser siempre Circuito_FPGA
     */
    private String nomEntidadGeneral;

    /**
     * Escritor de fichero
     */
    private BufferedWriter bw;

    /**
     * Ruta del fichero en la que se va a escribir
     */
    private String fichero;

    /**
     * Errores surgidos durante la generación del fichero
     */
    private Errores errores;

    /**
     * Constructor de la clase
     */
    public GeneraVhdl_V2(String fichero, Entidad entidad, Errores
errores) throws IOException {
        this.fichero = fichero;
        this.entidad = entidad;
        this.errores = errores;
    }

    /**
     * Abre el fichero para escribir el nuevo archivo vhdl.
```

```
     * @return Cierto si no ha habido ningún error, falso en caso
contrario.
     */
    public boolean abrir() {
        boolean correcto = fichero != null;
        try {
            bw = new BufferedWriter(new FileWriter(fichero));
        } catch (IOException e) {
            errores.error("Error al abrir el fichero " + fichero);
            correcto = false;
        }
        return correcto;
    }

    /**
     * Cierra el fichero VHDL que estamos construyendo
     * @return Cierto si no ha habido ningún error, falso en caso
contrario.
     */
    public boolean cerrar() {
        boolean correcto = bw != null;
        try {
            bw.close();
        } catch (IOException e) {
            errores.error("Error al cerrar el fichero " + fichero);
            correcto = false;
        }
        return correcto;
    }

    /**
     * Escribe una cadena en el fichero junto con un salto de linea
     * @param linea Cadena a escribir
     */
    private void escribirLinea(String linea) {
        try {
            bw.write(linea);
            bw.newLine();
        } catch (IOException ex) {
            errores.error("Error al escribir en fichero");
        }
    }

    /**
     * Método para saber si hay reloj
     * @return
     */
    private boolean hayReloj(){
        return this.entidad.getPosicionClk() >= 0;
    }


    /**
     * Escribe comentarios en la cabecera del fichero
     */
    private void comentariosCabecera(){
```

```
        escribirLinea("-----------------------------------------------
--------------------------------");
        escribirLinea("--Descripción:");
        escribirLinea("--\tEste fichero ha sido generado
automáticamente por la aplicación Nessy2.0");
        escribirLinea("--\tSe trata de un fichero que se llamara
user_logic y que estara situado en la carpeta
pcores/adaptador_circuito");
        escribirLinea("--");
        escribirLinea("--");
        escribirLinea("--Especificaciones:");
        escribirLinea("--\tCircuito a ejecutar:");
        escribirLinea("--\t\tNum. Entradas: "+
entidad.getBitsEntrada());
        escribirLinea("--\t\tNum. Salidas: " +
entidad.getBitsSalida());
        escribirLinea("--Autores:");
        escribirLinea("--\tRubén Tarancón Garijo");
        escribirLinea("--\tFelipe Serrano Santos");
        escribirLinea("--\tFacultad de Informatica. Universidad
Complutense de Madrid");
        Date date = new Date();
        escribirLinea("--Fecha: ");
        escribirLinea("--\t"+date.toString());
        escribirLinea("-----------------------------------------------
--------------------------------");
        escribirLinea("");
    }
    /**
     * Escribe la inclusión de librerías en el fichero
     */
    private void librerias() {
        escribirLinea("library ieee;");
        escribirLinea("use ieee.std_logic_1164.all;");
        escribirLinea("");
        escribirLinea("");
    }

    /**
     * Descripción de la entidad en la que se integrará nuestro
contador y que tendrá las señales para comunicarse con el bus
     */
    private void entidadGeneral() {
        escribirLinea("");
        escribirLinea("entity circuito is");
        escribirLinea("port(");
        escribirLinea("\tdato_listo : in std_logic;");
        escribirLinea("\tentrada_circuito : in std_logic_vector(0 to
31);");
        escribirLinea("\tsalida_circuito :out std_logic_vector(0 to
31)");
        escribirLinea(");");
        escribirLinea("end circuito;");
        escribirLinea("");
    }

    /**
```

```
     * Inicio de la arquitectura
     */
    private void inicioArquitectura() {
        escribirLinea("");
        escribirLinea("architecture Behavioral of circuito is");
        escribirLinea("");
    }

    /**
     * Señales del circuito que queremos adaptar
     */
    private void senalesCircuito() {
        escribirLinea("\t----------------- SEÑALES DEL CIRCUITO ------
-----------------");
        /*Para cada salida "sal_i" del circuito: senal_sal_i */
        for(int i=0;i<entidad.getNumSalidas();i++){
            Salida s = entidad.getSalida(i);
            if(s.getNumBits()==1){
                escribirLinea("\tsignal SENAL_"+s.getNombre()+"
: std_logic;");
            }else{
                escribirLinea("\tsignal SENAL_"+s.getNombre()+"
: std_logic_vector(0 to "+(s.getNumBits()-1)+");");
            }
        }
        escribirLinea("\t-------------------------------------------
-----------------");
        escribirLinea("");
    }

    /**
     * Componente del circuito que queremos adaptar
     */
    private void componenteCircuito() {
        escribirLinea("");
        escribirLinea("\t--------------------- COMPONENTE ------------
-----------------");

        escribirLinea("\tCOMPONENT "+entidad.getNombre()+" PORT (");
        for(int i=0;i<entidad.getNumEntradas();i++){
            Entrada e = entidad.getEntrada(i);
            if(e.getNumBits()==1){
                escribirLinea("\t\tsignal "+e.getNombre()+" : in
std_logic;");
            }else{
                escribirLinea("\t\tsignal "+e.getNombre()+" : in
std_logic_vector(0 to "+(e.getNumBits()-1)+");");
            }
        }
        for(int i=0;i<entidad.getNumSalidas();i++){
            Salida s = entidad.getSalida(i);
            if(entidad.getNumSalidas()-1==i){
                //EN EL CASO DE QUE SEA LA ÚLTIMA SALIDA NO SE PONE
";" AL FINAL
                if(s.getNumBits()==1){
                    escribirLinea("\t\tsignal "+s.getNombre()+" : out
std_logic");
```

```
                }else{
                    escribirLinea("\t\tsignal "+s.getNombre()+" : out
std_logic_vector(0 to "+(s.getNumBits()-1)+")");
                }
            }else{
                if(s.getNumBits()==1){
                    escribirLinea("\t\tsignal "+s.getNombre()+" : out
std_logic;");
                }else{
                    escribirLinea("\t\tsignal "+s.getNombre()+" : out
std_logic_vector(0 to "+(s.getNumBits()-1)+");");
                }
            }
        }
        escribirLinea("\t\t);");
        escribirLinea("\tEND COMPONENT;");
        escribirLinea("\t-------------------------------------------
-----------------");
        escribirLinea("");
    }

    /**
     * begin de la arquitectura
     */
    private void beginArquitectura() {
        escribirLinea("");
        escribirLinea("begin");
        escribirLinea("");
    }
    /**
     * Asignacion de las señales del circuito
     */
    private void asignacionSenalesCircuito() {
        int bit_usados=0;
        int tam_salida;

        escribirLinea("");
        escribirLinea("\t---------------- ASIGNACIÓN DE LAS SALIDAS --
------------------");
        for(int i=0;i<entidad.getNumSalidas();i++){
            Salida s = entidad.getSalida(i);
            tam_salida = s.getNumBits();
            if(tam_salida == 1){
                /* CASO STD_LOGIC */
                escribirLinea("\tsalida_circuito("+bit_usados+") <=
SENAL_"+s.getNombre()+";");
                bit_usados++;
            }else{
                /* CASO_ STD_LOGIC_VECTOR */
                escribirLinea("\tsalida_circuito("+bit_usados+" to
"+(bit_usados+tam_salida-1)+") <= SENAL_"+s.getNombre()+"(0 to
"+(tam_salida-1)+");");
                bit_usados = bit_usados + tam_salida;
            }
        }
        escribirLinea("\t-------------------------------------------
------------------");
```

```
        escribirLinea("");
    }

    /**
     * port map del componente Circuito con las señales del circuito
     */
    private void asignacionComponenteCircuito() {
        int bit_usados=0;
        int tam_entrada;
        escribirLinea("");
        escribirLinea("\t------------------------ MAPEO --------------
------------------");
        escribirLinea("\tinstancia : "+entidad.getNombre()+" PORT
MAP(");
        for(int i=0;i<entidad.getNumEntradas();i++){
            Entrada e = entidad.getEntrada(i);
            tam_entrada = e.getNumBits();
            if(tam_entrada==1){
                /* CASO STD_LOGIC */
                if(e.getEsReloj()){
                    /* En el caso de que sea el reloj se mapea a la
señal especial "senal_clk" */
                    escribirLinea("\t"+e.getNombre()+" =>
dato_listo,");
                }else{
                    escribirLinea("\t"+e.getNombre()+" =>
entrada_circuito("+bit_usados+"),");
                    bit_usados++;
                }
            }else{
                /* CASO STD_LOGIC_VECTOR */
                escribirLinea("\t"+e.getNombre()+"(0 to
"+(tam_entrada-1)+") => entrada_circuito("+bit_usados+" to
"+(bit_usados+tam_entrada-1)+"),");
                bit_usados = bit_usados + tam_entrada;
            }

        }
        for(int i=0;i<entidad.getNumSalidas();i++){
            Salida s = entidad.getSalida(i);
            if(entidad.getNumSalidas()-1==i){
                //EN EL CASO DE QUE SEA LA ÚLTIMA SALIDA NO SE PONE LA
"," AL FINAL
                escribirLinea("\t"+s.getNombre()+" =>
SENAL_"+s.getNombre());
            }else{
                escribirLinea("\t"+s.getNombre()+" =>
SENAL_"+s.getNombre()+",");
            }
        }
        escribirLinea("\t);");
        escribirLinea("\t------------------------------------------
------------------");
        escribirLinea("");
    }

    private void end() {
```

```
        escribirLinea("");
        escribirLinea("end Behavioral;");
        escribirLinea("");
    }

    /**
     * Crea el nuevo fichero VHDL a partir de la entidad que tiene la
clase.
     */
    public void crearFichero() {
        comentariosCabecera();
        librerias();
        entidadGeneral();
        inicioArquitectura();
        senalesCircuito();
        componenteCircuito();
        beginArquitectura();
        asignacionSenalesCircuito();
        asignacionComponenteCircuito();
        end();
    }
}
```

C.5. Paquete CompiladorEntidad

C.5.1. Entidad.java

```
package compiladorEntidad;

import java.util.ArrayList;

/**
 * Clase que representa una entidad vhdl.
 *
 * @author Carlos, David y Tony
 *
 */
public class Entidad {

    /**
     * Nombre de la entidad
     */
    private String nombre;

    /**
     * Número de bits de entrada
     */
    private int bitsEntrada;

    /**
     * Número de bits de salida
     */
    private int bitsSalida;

    /**
```

```
     * Entradas a la entidad
     */
    private ArrayList<Entrada> entradas;

    /**
     * Salidas de la entidad
     */
    private ArrayList<Salida> salidas;

    /**
     * Indica la posición que ocupa el CLK dentro de la entidad.
     * Si no hay CLK valdrá por defecto -1
     */
    private int posicionClk;

    /**
     * Indica la posición que ocupa el RESET dentro de la entidad.
     * Si no hay RESET valdrá por defecto -1
     */
    private int posicionReset;



    /**
     * Crea una entidad vacía.
     *
     */
    public Entidad() {
        entradas = new ArrayList<Entrada>();
        salidas = new ArrayList<Salida>();
        bitsEntrada = bitsSalida = 0;
        posicionClk = -1;
        posicionReset = -1;
    }

    /**
     *
     * @return Devuelve el nombre de la entidad.
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * Cambia el nombre de la entidad.
     * @param nombre Nuevo nombre de la entidad
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    /**
     * Getter para consultar el número de entradas de la Entidad.
     * @return El número de Entradas de la entidad.
     */
    public int getNumEntradas() {
        return entradas.size();
```

```
    }

    /**
     * Getter para consultar el número de salidas de la Entidad.
     * @return El número de Salidas de la entidad.
     */
    public int getNumSalidas() {
        return salidas.size();
    }

    /**
     * Getter para consultar el número de bits de entrada de la
Entidad
     * @return El número de bits de entrada de la entidad.
     */
    public int getBitsEntrada() {
        return bitsEntrada;
    }

    /**
     * Setter para establecer el número de bits de entrada de la
Entidad.
     * @param bitsEntrada
     */
    public void setBitsEntrada(int bitsEntrada) {
        this.bitsEntrada = bitsEntrada;
    }

    /**
     * Getter para consultar el número de bits de salida de la
Entidad.
     * @return El número de bits de salida de la entidad.
     */
    public int getBitsSalida() {
        return bitsSalida;
    }

    /**
     * Setter para establecer el número de bits de salida de la
Entidad.
     * @param bitsSalida Número de bits de salida
     */
    public void setBitsSalida(int bitsSalida) {
        this.bitsSalida = bitsSalida;
    }

    /**
     * Getter que devuelve la entrada i del ArrayList de las entradas.
     * @param i Posición de la Entrada que queremos devolver.
     * @return La entrada i del ArrayList de Entradas.
     */
    public Entrada getEntrada(int i) {
        return entradas.get(i);
    }

    /**
     * Getter que devuelve la salida i del ArrayList de las entradas.
```

```
     * @param i Posición de la Salida que queremos devolver.
     * @return La entrada i del ArrayList de salidas.
     */
    public Salida getSalida(int i) {
        return salidas.get(i);
    }

    /**
     * Devuelve la posición relativa de las entradas en la que se
encuentra
     * el reset. Esto será utulizdo para poder enviar al circuito la
señal de reset
     * aislándola de todas las demás
     * @return Posisicón que ocupa el reset dentro de las entradas
     */
    public int getPosicionReset(){
        return this.posicionReset;
    }

    /**
     * Devuelve la posición relativa de las entradas en la que se
encuentra
     * el clk. Esto será utilizado para pder hacer la asignación
adecuada, por
     * tratarse esta de una entrada especial.
     * @return Posisicón que ocupa el clk dentro de las entradas
     */
    public int getPosicionClk(){
        return this.posicionClk;
    }

    /**
     * Añade el argumento e de la clase Entrada al array de entradas
de la entidad.
     * Diferencia entre dos tipos de entradas especiales. Si el nombre
de la
     * entrada es CLK, CLOCK O RELOJ, la entrada será marcada como
entrada de reloj.
     * De la misma forma, si el nombre de la entrada es RST o RESET se
marcará como
     * una entrada especial de reset.
     * @param e Entrada a añadir a la entidad
     */
    public void anadeEntrada(Entrada e) {
        this.entradas.add(e);
        if (e.getNombre().equals("CLK") ||
e.getNombre().equals("CLOCK") ||
            e.getNombre().equals("RELOJ")) {
            e.ponerComoReloj();
            this.posicionClk = entradas.size()-1;
        }else if(e.getNombre().equals("RST") ||
e.getNombre().equals("RESET")){
            e.ponerComoReset(true);
            bitsEntrada += e.getNumBits();
            this.posicionReset = entradas.size()-1;
        } else {
            bitsEntrada += e.getNumBits();
```

```
        }
    }

    /**
     * Añade el argumento s de la clase Salida al array de salidas de
la entidad.
     * @param s Salida a añadir a la entidad
     */
    public void anadeSalida(Salida s) {
        this.salidas.add(s);
        bitsSalida += s.getNumBits();
    }

    /**
     *Para mostrar por pantalla. Sólo de prueba
     */
    public void muestra() {
        System.out.println("Entidad: " + nombre);
        System.out.println("Entradas:");
        for (int i = 0; i < entradas.size(); i++) {
            Entrada e = entradas.get(i);
            for (int j = 0; j < e.getNumBits(); j++) {
                System.out.println("\t" + e.getNombre() + "(" + j +
")");
            }
        }
        System.out.println("Salidas:");
        for (int i = 0; i < salidas.size(); i++) {
            Salida s = salidas.get(i);
            for (int j = 0; j < s.getNumBits(); j++) {
                System.out.println("\t" + s.getNombre() + "(" + j +
")");
            }
        }
        System.out.println("\nNum_entradas: " + bitsEntrada);
        System.out.println("Num_salidas: " + bitsSalida);
    }

    /**
     * Método para consultar el nombre de la entrada de reset.
     * @return Devuelve el Nombre de la Entrada de reset.
     */
    public String getNombreReset(){
        int i = 0;
        String reset = null;
        while (i < this.getNumEntradas() && reset == null){
            if (this.getEntrada(i).getEsReset()){
                reset = this.getEntrada(i).getNombre();
            }
            i++;
        }
        return reset;
    }

    /**
     * Devuelve una cadena con la descripción de las entradas y
salidas de la entidad.
```

```
     * @return Cadena con las entradas y salidas de la Entidad.
     */
    @Override
    public String toString() {
        String s = "";
        s += ("Entidad: " + nombre) + "\n";
        s += ("Entradas:") + "\n";
        for (int i = 0; i < entradas.size(); i++) {
            Entrada e = entradas.get(i);
            if (!e.getEsReloj()) {
                for (int j = 0; j < e.getNumBits(); j++) {
                    s += ("\t" + e.getNombre() + "(" + j + ")") +
"\n";
                }
            }
        }
        s += "Salidas:" + "\n";
        for (int i = 0; i < salidas.size(); i++) {
            Salida sal = salidas.get(i);
            for (int j = 0; j < sal.getNumBits(); j++) {
                s += ("\t" + sal.getNombre() + "(" + j + ")") + "\n";
            }
        }
        s += ("\nNum_entradas: " + bitsEntrada) + "\n";
        s += ("Num_salidas: " + bitsSalida) + "\n";
        return s;
    }
}
```

C.5.2. Entrada.java

```
package compiladorEntidad;


/**
 * Clase que representa la entrada de una entidad
 *
 * @author Carlos, David y Tony
 */
public class Entrada extends Puerto{


    /**
     * Indica si es una entrada de reloj
     */
    private boolean esReloj;

    /**
     * Indica si es una entrada de reset
     */
    private boolean esReset;

    /**
     * Método de consulta para saber si la Entrada es el reloj
     * @return Cierto o Falso dependiendo si se trata o no del reloj.
     */
```

```
    public boolean getEsReloj() {
        return esReloj;
    }

    /**
     * Método que establece la Entrada como reloj.
     */
    public void ponerComoReloj() {
        this.esReloj = true;
    }

    /**
     * Método para quitar a la Entrada la propiedad de que es el
reloj.
     */
    public void quitarComoReloj(){
        this.esReloj = false;
    }

    /**
     * Getter de para saber si la entrada se trata del reset.
     * @return Boolean si es cierto que la entrada es un reset.
     */
    public boolean getEsReset(){
        return this.esReset;
    }

    /**
     * Establece la entrada como Reset o le quita la propiedad.
     * @param valor Nuevo valor para el parámetro esReset.
     */
    public void ponerComoReset(boolean valor){
        this.esReset = valor;
    }

}
```

D.5.3. Errores.java

```
package compiladorEntidad;

import java.util.ArrayList;

/**
 *
 * Clase para almacenar los errores de compilación que se puedan ir
 * produciendo
 * @author Carlos,Tony y David
 */
public class Errores {


    /**
     * Estructura que almacena los errores en forma de cadena
     */
    private ArrayList<String> errores;
```

```
    /**
     * Constructor de la Clase. Inicializa el atributo errores.
     */
    public Errores(){
        errores = new ArrayList<String>();
    }

    /**
     * Añade un nuevo error al arrayList errores, donde se encuentran
todos los fallos.
     * @param error Nuevo error a insertar en la clase.
     */
    public void error(String error){
        errores.add(error);
    }

    /**
     * Devuelve todos los errores que tiene la clase.
     * @return ArrayList con todos los errores de la clase.
     */
    public ArrayList<String> getErrores() {
        return errores;
    }

}
```

C.5.4. EvaluadorExps.java

```
package compiladorEntidad;

/**
 * Clase que contiene métodos para evaluar una expresión aritmética
 *
 * @author Carlos, David y Tony.
 */
public class EvaluadorExps {

    /**
     * Método que devuelve un booleano dependiendo si la entrada es un
Operador
     * @param t Entero a consultar.
     * @return Booleano que indica si la entrada es un Operador o no.
     */
    public static boolean esOperador(int t){
        return t == (LexicoEntidad.SUMA) || t == (LexicoEntidad.RESTA)
||
                t == (LexicoEntidad.MULT) || t == (LexicoEntidad.DIV);
    }

    /**
     * Método que devuelve un booleano dependiendo si la entrada es un
Operando
     * @param t Entero a consultar.
     * @return Booleano que indica si la entrada es un Operando o no.
     */
    public static boolean esOperando(int t){
```

```
        return t == LexicoEntidad.ENTERO || t ==
LexicoEntidad.IDENTIFICADOR;
    }

    /**
     * Método de consulta de preferencia de operadores.
     * @param t1 Primer Operador.
     * @param t2 Segundo Operador.
     * @return Bololean Devuelve true si el primer operador tiene
menor o igual preferencia
     * que el segundo operador.
     */
    public static boolean esMenorIg(int t1, int t2){
        return t1 == t2 ||
                ((t1 == LexicoEntidad.SUMA || t1 ==
LexicoEntidad.RESTA) && (t1 == LexicoEntidad.SUMA || t1 ==
LexicoEntidad.RESTA)) ||
                (t2 == LexicoEntidad.MULT ||t2 == LexicoEntidad.DIV);
    }

    /**
     * Reduce una Expresión y devuelve el resultado
     * @param x Operador a aplicar.
     * @param op1 Primer operando.
     * @param op2 Segundo operando.
     * @return Resultado de la operación.
     */
    public static int aplicar(int x, int op1, int op2){
        switch(x){
            case LexicoEntidad.SUMA:
                return op1 + op2;
            case LexicoEntidad.RESTA:
                return op1 - op2;
            case LexicoEntidad.MULT:
                return op1 * op2;
            case LexicoEntidad.DIV:
                return op1 / op2;
            default:
                return -1;
        }
    }

}
```

C.5.5. LexidoEntidad.java

```
package compiladorEntidad;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;

/**
 * Clase que analiza léxicamente un fichero en el que está contenida
```

```
 * una entidad en lenguaje VHDL
 *
 * @author Carlos, David y Tony
 */
public class LexicoEntidad {

    /*Código de las palabras reservadas */
    /**
     * Código de la palabra reservada Entidad  -> 0
     */
    public static final int ENTITY = 0;

    /**
     * Código de la palabra reservada IS  -> 1
     */

    public static final int IS = 1;

    /**
     * Código de la palabra reservada Port  -> 2
     */
    public static final int PORT = 2;

    /**
     * Código de la palabra reservada STD_LOGIC  -> 3
     */
    public static final int STD_LOGIC = 3;

    /**
     * Código de la palabra reservada STD_LOGIC_VECTOR  -> 4
     */
    public static final int STD_LOGIC_VECTOR = 4;

    /**
     * Código de la palabra reservada IN  -> 5
     */
    public static final int IN = 5;

    /**
     * Código de la palabra reservada OUT  -> 6
     */
    public static final int OUT = 6;

    /**
     * Código de la palabra reservada DOWNTO  -> 7
     */
    public static final int DOWNTO = 7;

    /**
     * Código de la palabra reservada END  -> 8
     */
    public static final int END = 8;

    /**
     * Código del elemento PUNTO_Y_COMA  -> 9
     */
    public static final int PUNTO_Y_COMA = 9;
```

```
/**
 * Código del elemento DOS_PUNTOS  -> 10
 */
public static final int DOS_PUNTOS = 10;

/**
 * Código del elemento ABRE_PARENTESIS  -> 11
 */
public static final int ABRE_PARENTESIS = 11;

/**
 * Código del elemento CIERRA_PARENTESIS  -> 12
 */
public static final int CIERRA_PARENTESIS = 12;

/**
 * Código del elemento ENTERO  -> 13
 */
public static final int ENTERO = 13;

/**
 * Código del elemento IDENTIFICADOR  -> 14
 */
public static final int IDENTIFICADOR = 14;

/**
 * Código del elemento EOF  -> 15
 */
public static final int EOF = 15;

/**
 * Código del elemento OTRO  -> 16
 */
public static final int OTRO = 16;

/**
 * Código del elemento GENERIC  -> 17
 */
public static final int GENERIC = 17;

/**
 * Código del elemento INTEGER  -> 18
 */
public static final int INTEGER = 18;

/**
 * Código del elemento ASIG_GENERIC  -> 19
 */
public static final int ASIG_GENERIC = 19;

/**
 * Código del elemento SUMA  -> 20
 */
public static final int SUMA = 20;

/**
```

```
     * Código del elemento RESTA  -> 21
     */
    public static final int RESTA = 21;

    /**
     * Código del elemento MULT  -> 22
     */
    public static final int MULT = 22;

    /**
     * Código del elemento DIV  -> 23
     */
    public static final int DIV = 23;

    /**
     * Lector de fichero
     */
    private BufferedReader reader;

    /**
     * Posibles errores
     */
    private Errores errores;

    /**
     * Almacena el último caracter leído del fichero
     */
    private Character ultimoCharLeido;

    /**
     * Almacena la cadena leída hasta el momento
     */
    private String cadena;

    /**
     * Número de linea que se está leyendo
     */
    private int numLinea;
    private int estado;

    /**
     * Getter que devuelve el número de línea por la que va el
Analizador.
     * @return Integer Número de línea.
     */
    public int getNumLinea() {
        return this.numLinea;
    }

    /**
     * Estructura que almacena el conjunto de palabras reservadas
     */
    private static final HashMap<String, Integer> palabrasReservadas =
new HashMap<String, Integer>(25);

    static {
        palabrasReservadas.put("ENTITY", ENTITY);
```

```
        palabrasReservadas.put("IS", IS);
        palabrasReservadas.put("PORT", PORT);
        palabrasReservadas.put("STD_LOGIC", STD_LOGIC);
        palabrasReservadas.put("STD_LOGIC_VECTOR", STD_LOGIC_VECTOR);
        palabrasReservadas.put("IN", IN);
        palabrasReservadas.put("OUT", OUT);
        palabrasReservadas.put("DOWNTO", DOWNTO);
        palabrasReservadas.put("END", END);
        palabrasReservadas.put("GENERIC", GENERIC);
        palabrasReservadas.put("INTEGER", INTEGER);
    }
    /**
     * Constructor de la clase.
     * @param fichero Código a analizar.
     * @param errores Errores que se han detectado
     * @throws FileNotFoundException
     * @throws IOException
     */
    public LexicoEntidad(String fichero, Errores errores) throws
FileNotFoundException, IOException {
        reader = new BufferedReader(new FileReader(fichero));
        numLinea = 1;
        this.errores = errores;
        leerCaracter();
    }

    /**
     * Inicia el análisis, llama a la función sigToken()
     * @return El primer Token.
     * @throws IOException
     */
    public Token iniciar() throws IOException {
        return sigToken();
    }

    /**
     * Cierra el Buffer de lectura del fichero.
     * @throws IOException
     */
    public void cerrar() throws IOException {
        this.reader.close();
    }

    /**
     * Para saber si un caracter determinado es una letra
     * @param ch El caracter del que se quiere saber si es una letra
     * @return true si ch es una letra y false en caso contrario
     */
    private boolean esLetra(Character ch) {
        return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');
    }

    /**
     * Para saber si un caracter determinado es un dígito
     * @param ch El caracter del que se quiere saber si es un dígito
     * @return true si ch es un dígito y false en caso contrario
     */
```

```
    private boolean esDigito(Character ch) {
        return ch >= '0' && ch <= '9';
    }

    /**
     * Lee un caracter del fichero y lo devuelve
     * @return El caracter leído o null en caso de que no se haya
podido leer
     * @throws IOException
     */
    private Character leerCaracter() throws IOException {
        if (reader == null) {
            errores.error("No se asigno ningún reader");
            throw new IOException("No se asigno ningún reader");
        }
        if (reader.ready()) {
            int ch = reader.read();
            if (ch != -1) //si no es fin de fichero
            {
                ultimoCharLeido = new Character((char) ch);
            } else {
                ultimoCharLeido = null;
            }

            if (ultimoCharLeido.charValue() == '\n') { //Si es un
salto de lÃnea
                numLinea++; //incrementa la linea
            }
            return ultimoCharLeido;
        } else {
            return null;
        }
    }

    /**
     * Método que gestiona el autómata finito que ocntrola el
Analizador Léxico
     * @return El siguiente Token a leer.
     * @throws IOException
     */
    public Token sigToken() throws IOException {
        estado = 0;
        cadena = "";
        while (true) {
            char caracterLeido = 0;
            if (ultimoCharLeido != null) {
                caracterLeido =
Character.toLowerCase(ultimoCharLeido);
            }
            switch (estado) {
                case 0:
                    if (this.ultimoCharLeido == null) {
                        transita(6);

                    } else if ((ultimoCharLeido.charValue() == ' ') ||
(ultimoCharLeido.charValue() == '\n') || (ultimoCharLeido.charValue()
== '\r') || (ultimoCharLeido.charValue() == '\t')) {
```

```
                        transita(0); //salta los blancos
                        cadena = "";
                    } else if (esLetra(caracterLeido)) {
                        transita(1);
                    } else if (esDigito(caracterLeido)) {
                        transita(2);
                    } else if (caracterLeido == ';') {
                        transita(3);
                    } else if (caracterLeido == '(') {
                        transita(4);
                    } else if (caracterLeido == ')') {
                        transita(5);
                    } else if (caracterLeido == ':') {
                        transita(7);
                    } else if (caracterLeido == '-') {//posible
comentario
                        transita(8);
                    } else if (caracterLeido == '+') {
                        transita(12);
                    } else if (caracterLeido == '*') {
                        transita(13);
                    }else if (caracterLeido == '/') {
                        transita(14);
                    } else {//caracter desconocido
                        transita(10);
                    }
                    break;
                case 1:
                    if (esLetra(caracterLeido) ||
esDigito(caracterLeido) || caracterLeido == '_') {
                        transita(1);
                    } else {
                        Integer codigo =
palabrasReservadas.get(cadena.toUpperCase());
                        if (codigo != null) {//si es una palabra
reservada
                            return new Token(codigo, cadena,
numLinea);
                        } else {//si es el nombre de una
entrada/salida/entidad
                            return new Token(IDENTIFICADOR, cadena,
numLinea);
                        }
                    }
                    break;
                case 2:
                    if (esDigito(caracterLeido)) {//continua el número
                        transita(2);
                    } else {//guardar el numero
                        return new Token(ENTERO, cadena, numLinea);
                    }
                    break;
                case 3:
                    return new Token(PUNTO_Y_COMA, cadena, numLinea);
                case 4:
                    return new Token(ABRE_PARENTESIS, cadena,
numLinea);
```

```
                case 5:
                    return new Token(CIERRA_PARENTESIS, cadena,
numLinea);
                case 6:
                    return new Token(EOF, cadena, numLinea);
                case 7:
                    if (caracterLeido == '=') {
                        transita(11);
                    } else {
                        return new Token(DOS_PUNTOS, cadena,
numLinea);
                    }
                    break;
                case 8:
                    if (caracterLeido == '-') {
                        transita(9);
                    } else {
                        return new Token(RESTA, cadena, numLinea);
                    }
                    break;
                case 9://comentarios
                    if (caracterLeido == '\n' || caracterLeido ==
'\r') {
                        transita(0);
                    } else {
                        transita(9);
                    }
                    break;
                case 10://desconocido
                    return new Token(OTRO, cadena, numLinea);
                case 11:
                    return new Token(ASIG_GENERIC, cadena, numLinea);
                case 12:
                    return new Token(SUMA, cadena, numLinea);
                case 13:
                    return new Token(MULT, cadena, numLinea);
                case 14:
                    return new Token(DIV, cadena, numLinea);
            }
        }
    }

    /**
     * Cambia el estado actual del analizador léxico por el que se le
pasa por
     * parámetro. Además lee el siguiente caracter para dejarlo
preparado para
     * ser analizado.
     * @param sigEstado Siguiente estado al que va el analizador.
     * @throws IOException
     */
    public void transita(int sigEstado) throws IOException {
        estado = sigEstado;
        if (ultimoCharLeido != null) {
            cadena = cadena.concat(ultimoCharLeido.toString());
            ultimoCharLeido = leerCaracter();
        }
```

```
    }

    /**
     * Método de Consulta sobre el último carácter leído.
     * @return El último carácter leído.
     */
    public Character getUltimoCharLeido() {
        return ultimoCharLeido;
    }
}
```

C.5.6. Pila.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package compiladorEntidad;

import java.util.ArrayList;

/**
 * Representa una pila de elementos genéricos.
 * Utilizada para la evaluación de expresiones.
 *
 *
 * @param <T>
 * @author User
 */
public class Pila<T> {

    /**
     * La posición el la que se encuentra la cima de la pila
     */
    private int cima;

    /**
     * Estructura para almacenar el contenido de la pila
     */
    private ArrayList<T> pila;

    /**
     * Método constructor de la clase. Crea una pila vacía.
     */
    public Pila(){
        pila = new ArrayList<T>();
        cima = -1;
    }

    /**
     * Método para saber si es pila vacía.
     * @return Boolean, true si es pila vacía, false en caso
contrario.
     */
    public boolean esVacia(){
```

```
        return cima == -1;
    }

    /**
     * Apilar un nuevo elemento en la pila.
     * @param t Elemento a apilar.
     */
    public void apilar(T t){
        pila.add(t);
        cima++;
    }

    /**
     * Desapila un elemento de la pila
     * @return Devuelve el elemento desapilado.
     */
    public T desapilar(){
        if (!esVacia()){
            return pila.remove(cima--);
        }
        return null;
    }

    /**
     * Consulta la cima de la pila
     * @return El elemento de la cima de la Pila
     */
    public T getCima(){
        if (!esVacia()){
            return pila.get(cima);
        }
        return null;
    }

    /**
     * Método que consulta el número de elementos de la pila.
     * @return El número de elementos de la Pila.
     */
    public int numElems(){
        return cima+1;
    }

}
```

C.5.7. Puerto.java

```
package compiladorEntidad;



/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

```java
/**
 * Representa el puerto de una entidad el cual puede ser tanto
 * una entrada como una salida
 * @author Carlos, David y Tony
 */
public abstract class Puerto {

    /**
     * Número de bits del puerto (entrada o salida)
     */
    private int numBits;

    /**
     * Nombre del puerto
     */
    private String nombre;

    /**
     * Método que obtiene el nombre del Puerto.
     * @return El nombre del Puerto.
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * Establece el nombre del Puerto.
     * @param nombre Nuevo nombre del puerto.
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }


    /**
     * Consulta el número de bits que utiliza el puerto.
     * @return Número de bits del puerto.
     */
    public int getNumBits() {
        return numBits;
    }

    /**
     * Establece el número de bits del Puerto
     * @param numBits Nuevo número de bits del Puerto.
     */
    public void setNumBits(int numBits) {
        this.numBits = numBits;
    }

}
```

C.5.8. Salida.java

```
package compiladorEntidad;


/**
 * Representa la salida de una entidad. Hereda de la clase Puerto.
 * @author Carlos, David y Tony
 */
public class Salida extends Puerto{

}
```

C.5.9. SintacticoEntidad.java

```
package compiladorEntidad;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

/**
 * Clase que analiza sintácticamente una entidad definida en VHDL.
Además
 * mientras la analiza, almacena los datos leídos en nuestra
estructura de datos
 * de la entidad. La forma en la que funciona este analizador
sintáctico está
 * determinada en la gramática que se especifica en la memoria.
 *
 * @author Carlos,David y Tony
 */
public class SintacticoEntidad {

    /**
     * Número de entradas máximo. En nuestra Aplicación son 32
     */
    public static final int MAX_ENTRADAS = 32;
    /**
     * Número de salidas máximo. En nuestra Aplicación son 32
     */
    public static final int MAX_SALIDAS = 32;

    /**
     * Token actual que se está analizando
     */
    private Token token;

    /**
     * Entidad donde se va a almacenar lo leido del fichero
     */
    private Entidad entidad;

    /**
     * Analizador léxico que nos va a proveer de los tokens
     */
    private LexicoEntidad lector;
```

```
    /**
     * Posibles errores que se puedan producir
     */
    private Errores errores;

    /**
     * Tabla de símbolos necesaria para almacenar las variables
     * de los posibles generics
     */
    private HashMap<String,Integer> tablaSimbolos;

    /**
     * Getter para consultar la Entidad.
     * @return Devuelve el Objeto de la clase Entidad
     */
    public Entidad getEntidad() {
        return entidad;
    }

    /**
     * Constructor de la clase
     * @param fichero Ruta del fichero que vamos a analizar
sintácticamente.
     * @param errores ArrayList de errores encontrados
     * @throws IOException
     */
    public SintacticoEntidad(String fichero, Errores errores) throws
IOException {
        lector = new LexicoEntidad(fichero, errores);
        this.errores = errores;
        entidad = new Entidad();//creamos una entidad vacía
        tablaSimbolos = new HashMap<String,Integer>();

     }

   /**
    * Inicia el análisis sintáctico.
    * @throws IOException
    */
   public void inicia() throws IOException {
        token = lector.iniciar();
    }

   /**
    * Función para cerrar el fichero que estamos analizando.
    */
   public void cerrar() {
        try {
            lector.cerrar();
        } catch (IOException ex) {
            errores.error("Error al leer el fichero de entity");
        }
    }

   /**
```

```
    * Lee del fichero mientras no encuentra la palabra 'Entity'. Con
esto
    * descartamos todos los comentarios iniciales además de la
incluisión de
    * librerías además de cualquier otra cosa que no nos interese del
fichero.
    *
    * @throws IOException
    */
   public void Cabecera() throws IOException{
        while(token != null && token.getCodigo() !=
LexicoEntidad.ENTITY
                && token.getCodigo() != LexicoEntidad.GENERIC){
            token = lector.sigToken();
        }
    }

    /**
     * Analiza la entidad del fichero.
     * @return Devuelve True si ha habido algún error, si no devuelve
Falso.
     * @throws Exception
     */
    public boolean Entidad() throws Exception{
        Cabecera();
        boolean error = false;
        empareja(LexicoEntidad.ENTITY);
        String nomEntidadInicio = token.getLexema();
        empareja(LexicoEntidad.IDENTIFICADOR);
        empareja(LexicoEntidad.IS);
        if (token.getCodigo() == LexicoEntidad.GENERIC){
            error = error | Generic();
        }
        error = error | Puertos();
        empareja(LexicoEntidad.END);
        String nomEntidadEnd = token.getLexema();
        empareja(LexicoEntidad.IDENTIFICADOR);
        if (nomEntidadInicio.equals(nomEntidadEnd)){
            entidad.setNombre(nomEntidadInicio);//daría igula uno que
otro
        }else{
            errores.error("El nombre de la entidad no coincide");
            error = true;
        }
        return error;
    }

    /**
     * Analiza la parte de un componente genérico.
     * @return Cierto si ha habido algún error o falso si está
correcto.
     * @throws Exception
     */
    public boolean Generic() throws Exception{
        empareja(LexicoEntidad.GENERIC);
        empareja(LexicoEntidad.ABRE_PARENTESIS);
        boolean error = Variables();
```

```
        empareja(LexicoEntidad.CIERRA_PARENTESIS);
        empareja(LexicoEntidad.PUNTO_Y_COMA);
        return error;
    }

    /**
     * Analiza la parte de declaración de variables dentro de un
genérico.
     * @return Cierto si ha habido algún error o falso si está
correcto.
     * @throws Exception
     */
    public boolean Variables() throws Exception{
        boolean error = Variable();
        error = error | RVariables();
        return error;
    }

    /**
     * Analiza la declaración de una variable y la inserta en la tabla
de símbolos.
     * @return Cierto si ha habido algún error o falso si está
correcto.
     * @throws Exception
     */
    public boolean Variable() throws Exception{
        String var = token.getLexema();
        empareja(LexicoEntidad.IDENTIFICADOR);
        empareja(LexicoEntidad.DOS_PUNTOS);
        empareja(LexicoEntidad.IN);
        empareja(LexicoEntidad.INTEGER);
        empareja(LexicoEntidad.ASIG_GENERIC);
        String valor = token.getLexema();
        empareja(LexicoEntidad.ENTERO);
        tablaSimbolos.put(var,new Integer(valor));
        return false;
    }

    /**
     * Analiza el resto de variables.
     * @return Cierto si ha habido algún error o falso si está
correcto.
     * @throws Exception
     */
    public boolean RVariables() throws Exception{
        boolean error = false;
        if (token.getCodigo() == LexicoEntidad.PUNTO_Y_COMA){
            empareja(LexicoEntidad.PUNTO_Y_COMA);
            error = Variable() | RVariables();
        }
        return error;
    }

    /**
     * Analiza la parte correspondiente a los puertos de entrada y
salida
```

```
     * @return Cierto si ha habido algún error o falso si está
correcto.
     * @throws Exception
     */
    public boolean Puertos() throws Exception{
        empareja(LexicoEntidad.PORT);
        empareja(LexicoEntidad.ABRE_PARENTESIS);
        boolean error = Senales();
        empareja(LexicoEntidad.CIERRA_PARENTESIS);
        empareja(LexicoEntidad.PUNTO_Y_COMA);
        return error;
    }

    /**
     * Analiza la parte donde se declaran todas las señales.
     * @return Cierto si ha habido algún error o falso si está
correcto.
     * @throws Exception
     */
    public boolean Senales() throws Exception{
        boolean error = Senal();
        error = error | RSenales();
        return error;
    }

    /**
     * Analiza si está bien declarada una señal. Además la añade
     * a la entidad. Si el número de entradas o de salidas es excedido
     * se mostrará el error al final del análisis.
     * @return Cierto si ha habido algún error o falso si está
correcto.
     * @throws Exception
     */
    public boolean Senal() throws Exception{
        String ident = token.getLexema();
        boolean error = false;
        int entradaSalida;
        empareja(LexicoEntidad.IDENTIFICADOR);
        empareja(LexicoEntidad.DOS_PUNTOS);
        entradaSalida = token.getCodigo();
        if (entradaSalida == LexicoEntidad.IN || entradaSalida==
LexicoEntidad.OUT){
            empareja(entradaSalida);
        }else{
            throw new Exception("Error sintactico en la fila " +
token.getNumLinea() +  ". No se esperaba \"" + token.getLexema() +
"\".");
        }
        int tamano = Tipo();
        if (tamano > 0){
            Puerto es = null;
            if (entradaSalida == LexicoEntidad.IN){//si es una entrada
                es = new Entrada();
                es.setNombre(ident);
                es.setNumBits(tamano);
                entidad.anadeEntrada((Entrada)es);
                if (entidad.getBitsEntrada() > MAX_ENTRADAS){
```

```
                    error = true;
                    errores.error("Error al añadir la entrada: " +
es.getNombre()+". No se permite un número de entradas mayor de " +
MAX_ENTRADAS);
                }
            }else if(entradaSalida == LexicoEntidad.OUT){//si es una
salida
                es = new Salida();
                es.setNombre(ident);
                es.setNumBits(tamano);
                entidad.anadeSalida((Salida)es);
                if (entidad.getBitsSalida() > MAX_SALIDAS){
                    error = true;
                    errores.error("Error al añadir la salida: " +
es.getNombre()+". No se permite un número de salidas mayor de " +
MAX_SALIDAS);
                }
            }


        }else{
            errores.error("Linea: " + token.getNumLinea() + ". Tamaño
incorrecto");//es un error que permite continuar
            error = true;
        }
            tamano = 0;
        return error;
    }

    /**
     * Analiza el resto de señales.
     * @return Cierto si ha habido algún error o falso si está
correcto.
     * @throws Exception
     */
    public boolean RSenales() throws Exception{
        boolean error = false;
        if (token.getCodigo() == LexicoEntidad.PUNTO_Y_COMA){
            empareja(LexicoEntidad.PUNTO_Y_COMA);
            error = Senal() | RSenales();
        }
        return error;
    }

    /**
     * Analiza la parte correspondiente al tipo de una señal. Ésta
puede ser
     * STD_LOGIC o STD_LOGIC_VECTOR. En función de ello, devuelve el
tamaño
     * de esa entrada o salida. Si el tamaño es negativo indicará un
error.
     * @return devuelve el tamaño de una entrada o salida.
     * @throws Exception
     */
    public int Tipo() throws Exception{
        int tamano = 0;
        if (token.getCodigo() == LexicoEntidad.STD_LOGIC){
```

```
            empareja(LexicoEntidad.STD_LOGIC);
            tamano = 1;
        }else if(token.getCodigo() == LexicoEntidad.STD_LOGIC_VECTOR){
            empareja(LexicoEntidad.STD_LOGIC_VECTOR);
            empareja(LexicoEntidad.ABRE_PARENTESIS);
            int inicio = Exp();
            empareja(LexicoEntidad.DOWNTO);
            int fin = 0;//TODO
            empareja(LexicoEntidad.ENTERO);
            if (inicio >= 0 && fin >= 0)
                tamano = inicio - fin +1;
            empareja(LexicoEntidad.CIERRA_PARENTESIS);
        }else{
            throw new Exception("Error sintactico en la fila " +
token.getNumLinea() + ". No se esperaba \"" + token.getLexema() +
"\".");
        }
        return tamano;
    }

    /**
     * Analiza una expresión aritmética.
     * @return Devuelve el valor de la expresión.
     * @throws Exception
     */
    public int Exp() throws Exception{
        return evaluar();
    }

    /**
     * Transforma una expresión a PostFija para poder evaluarla.
     * @return Devuelve el ArrayList con la expresión transformada a
PostFija.
     * @throws Exception
     */
    public ArrayList<Token> pasarAPostFija() throws Exception{
        Pila<Token> pila = new Pila<Token>();
        boolean finExp = false;
        ArrayList<Token> post = new ArrayList<Token>();
        Token t;
        while (!finExp){
            t = new Token(token);//hace una copia del token
            if (EvaluadorExps.esOperando(t.getCodigo())){
                post.add(t);
                empareja(t.getCodigo());
            }else if(t.getCodigo() == LexicoEntidad.ABRE_PARENTESIS){
                pila.apilar(t);
                empareja(LexicoEntidad.ABRE_PARENTESIS);
            }else if(t.getCodigo() ==
LexicoEntidad.CIERRA_PARENTESIS){
                while (pila.getCima().getCodigo() !=
LexicoEntidad.ABRE_PARENTESIS){
                    post.add(pila.desapilar());
                }
                pila.desapilar();
                empareja(LexicoEntidad.CIERRA_PARENTESIS);
            }else if(EvaluadorExps.esOperador(t.getCodigo())){
```

```
                while(!pila.esVacia() &&
EvaluadorExps.esMenorIg(t.getCodigo(),pila.getCima().getCodigo())){
                    post.add(pila.desapilar());
                }
                pila.apilar(t);
                empareja(t.getCodigo());
            }else{
                finExp = true;
            }
        }
        while (!pila.esVacia()){
            post.add(pila.desapilar());
        }
        return post;
    }

    /**
     * Evalúa una expresión y devuelve el resultado.
     * @return El resultado de la expresión.
     * @throws Exception
     */
    public int evaluar() throws Exception{
        int valor = -1;
        Pila<Integer> pila = new Pila<Integer>();
        ArrayList<Token> post = pasarAPostFija();
        int i = 0;
        Token t;
        while (i < post.size()){
            t = post.get(i);
            i++;
            if (t.getCodigo() == LexicoEntidad.IDENTIFICADOR){
                if (tablaSimbolos.get(t.getLexema()) != null){
                    int v = tablaSimbolos.get(t.getLexema()); //el
valor de la variable
                    pila.apilar(v);
                }else{
                    errores.error("La variable " + t.getLexema() + "
no está definida");
                    return -1;
                }
            }else if(t.getCodigo() == LexicoEntidad.ENTERO){
                pila.apilar(Integer.parseInt(t.getLexema()));//sabemos
seguro que es un entero
            }else{ //operador
                int op2 = pila.desapilar();
                int op1 = pila.desapilar();
                valor = EvaluadorExps.aplicar(t.getCodigo(),op1,op2);
                pila.apilar(valor);
            }
        }
        return pila.desapilar();
    }



    /**
```

```
     * Intenta emparejar el token actual del analizador con el del
fichero. Además
     * actualiza el siguiente token que se debe leer, pidiéndoselo al
analizador
     * léxico.
     * @param tk Token a emparejar
     * @throws Exception
     */
    public void empareja(int tk) throws Exception {
        if (token != null)//si no ha habido error lexico
        {
            if (tk == token.getCodigo()) {
                token = lector.sigToken();
            } else {
                errorSint();
            }
        }else{
            throw new Exception("Fila " + lector.getNumLinea() + ":
Error, caracter " + lector.getUltimoCharLeido() + " desconocido");
        }
        return;
    }


    /**
     * Añade el error al atributo errores y lanza la excepción
indicando el error.
     * @throws Exception
     */
    public void errorSint() throws Exception{
       errores.error("Error sintactico en la fila " +
token.getNumLinea() + ". No se esperaba \"" + token.getLexema() +
"\".");
        throw new Exception("Error sintactico en la fila " +
token.getNumLinea() + ". No se esperaba \"" + token.getLexema() +
"\".");
    }

}
```

C.5.10. Token.java

```
package compiladorEntidad;

/**
 * Clase para representar un token del fichero que contiene una
entidad
 * descrita en VHDL
 *
 * @author Carlos, Tony y David
 */
public class Token {

    /**
     * Literal del token
     */
```

```
    private String lexema;

    /**
     * Código interno asociado a ese token
     */
    private int codigo;

    /**
     * Número de línea en el que se encuentra el token
     */
    private int numLinea;

    /**
     * Constructor de la clase.
     * @param codigo Código del Token.
     * @param lexema String del Token
     * @param numLinea Entero que indica la línea donde se encuentra
el token dentro del fichero.
     */
    public Token(int codigo, String lexema, int numLinea) {

        this.codigo = codigo;
        this.lexema = lexema.toUpperCase();
        this.numLinea = numLinea;

//System.out.println("Token("+codigo+","+lexema+","+numLinea+","+numCo
lumna+")");

    }

    /**
     * Contructora que crea una copia de un Token a partir de otro.
     * @param otro Token a copiar.
     */
    public Token(Token otro){
        this.codigo = otro.codigo;
        this.lexema = otro.lexema;
        this.numLinea = otro.numLinea;
    }

    /**
     * Obtiene el código de un Token
     * @return Entero que codifica el Token.
     */
    public int getCodigo() {
        return codigo;
    }

    /**
     * Obtiene la cadena del String.
     * @return El String del Token asociado.
     */
    public String getLexema() {
        return lexema;
    }

    /**
```

```
     * Obtiene el número de línea donde se encuentra el Token.
     * @return El entero con la línea donde se encuentra el Token.
     */
    public int getNumLinea() {
        return numLinea;
    }
}
```

Bibliografía

[BJKM09] “A self-reconfiguring platform”. Brandon Blodget, Philip James-Roxby, Eric Keller, Scott McMillan y Prasanna Sundararajan. Xilinx.

[CGHM05] “Auto-Reconfiguración sobre FPGAs” J. Castillo, I. González, P. Huerta, J.I. Martínez, V Jornadas de Computación Reconfigurable y Aplicaciones, JCRA 2005.

[KBSao6] “Dynamic Partial Reconfiguration of a Field Programmable Gate Array”. Michael Kristan, Brian Loveland y Robert Sazanowicz. Worcester Polytechnic Institute.

[SGGa10] “Diseño de un entorno para la inserción y detección de errores en sistemas basados en FPGAs”. Carlos Sanchez Velisco, Antonio José García Martinez y David García Maiquez. Universidad Complutense de Madrid, Facultad de Informática. curso 2009-2010.

[XilA10] “Virtex-5 FPGA configuration user guide”. Xilinx. Agosto 2010.

[Xili04] “OPB_Hwicap”. Xilinx. 2004.

[Xili05] “Xilinx University Program Virtex-II Pro Development System”. Xilinx. Marzo 2005.

[Xili07] “Virtex-II pro and Virtex-II pro X FPGA User guide”. Xilinx. 2007.

[XilM10] “PlanAhead Software Tutorial”. Xilinx. Mayo 2010.