Facultad de Informática

Diseño de un entorno para la inserción y detección de errores en sistemas basados en FPGAs

Profesor director: Hortensia Mecha

Autores: Carlos Sánchez-Vellisco Sánchez
Antonio José García Martínez
David Fernández Maiquez

Curso académico: 2009-2010

Índice

Resumen del proyecto

El objeto de este proyecto es el desarrollo de un entorno para inyectar errores en una FPGA, en este caso del modelo Virtex II Pro- y así evaluar el posible impacto que pueden tener las alteraciones en la memoria de configuración de un dispositivo reconfigurable sobre un determinado diseño. Estas alteraciones emulan el efecto que una partícula cósmica pudiera tener sobre una celda RAM de la memoria de configuración. Para ello, cargaremos un circuito específico en la FPGA, conectándolo previamente a un módulo de entrada/salida que nos permita enviar y recibir datos de la placa. Una vez cargado, aplicaremos una serie de entradas al circuito y obtendremos sus salidas. Tras ello iremos insertando errores en el circuito -mediante la modificación sucesiva de los bits de configuración de la FPGA- y veremos si la salida es o no la esperada.

Summary of the project

The goal of this project is to develop an environment for injecting faults into a FPGA device (Virtex-II Pro model) and therefore to evaluate which consequences may those changes have in the configuration memory of a reconfigurable device using any design. These changes emulate the same effect as if a cosmic particle hit on a memory configuration RAM cell. In order to do this, we will load a specific circuit on the FPGA device, which will be previously connected to an output/intput module so that we can send and receive data from the device. Once the circuit is loaded we will apply several inputs to the circuit and we will receive its outputs. After that we will inject faults into the circuit modifying the configuration bits of the FPGA device and we will see if we obtain the expected output.

1 Marco de trabajo

Las FPGA's (Field Programmable Gate Array) son dispositivos que contienen bloques de lógica cuya interconexión y funcionalidad es programable. En ellas se puede programar desde una simple puerta lógica hasta un complejo circuito secuencial.

Existen otras formas de implementar circuitos digitales, tales como ASIC (Aplication Specific Integrated Circuit), microcontroladores (con un conjunto fijo de instrucciones), CPLDs [9] (basado en memorias no reconfigurables, pero más lentos y densos que una FPGA).

Las ventajas que tienen las FPGA's, [3] y [5] , es que son dispositivos reconfigurables, tienen un costo menor con respecto a los ASIC y sus circuitos se ejecutan más rápido que en otros dispositivos reprogramables. Además al tratarse de una solución basada en hardware, su ejecución es "en paralelo", cosa que no ocurre en un microcontrolador en el que las instrucciones se ejecutan de forma secuencial. La reconfiguración permite además la ejecución de múltiples tareas, multiplexadas en el tiempo, lo que nos permite pensar en una verdadera multitarea hardware.

Nosotros trabajaremos con una placa basada en una FPGA del modelo Virtex-II Pro, [1], [8] y [10], como la que se muestra en la Figura 1.1.

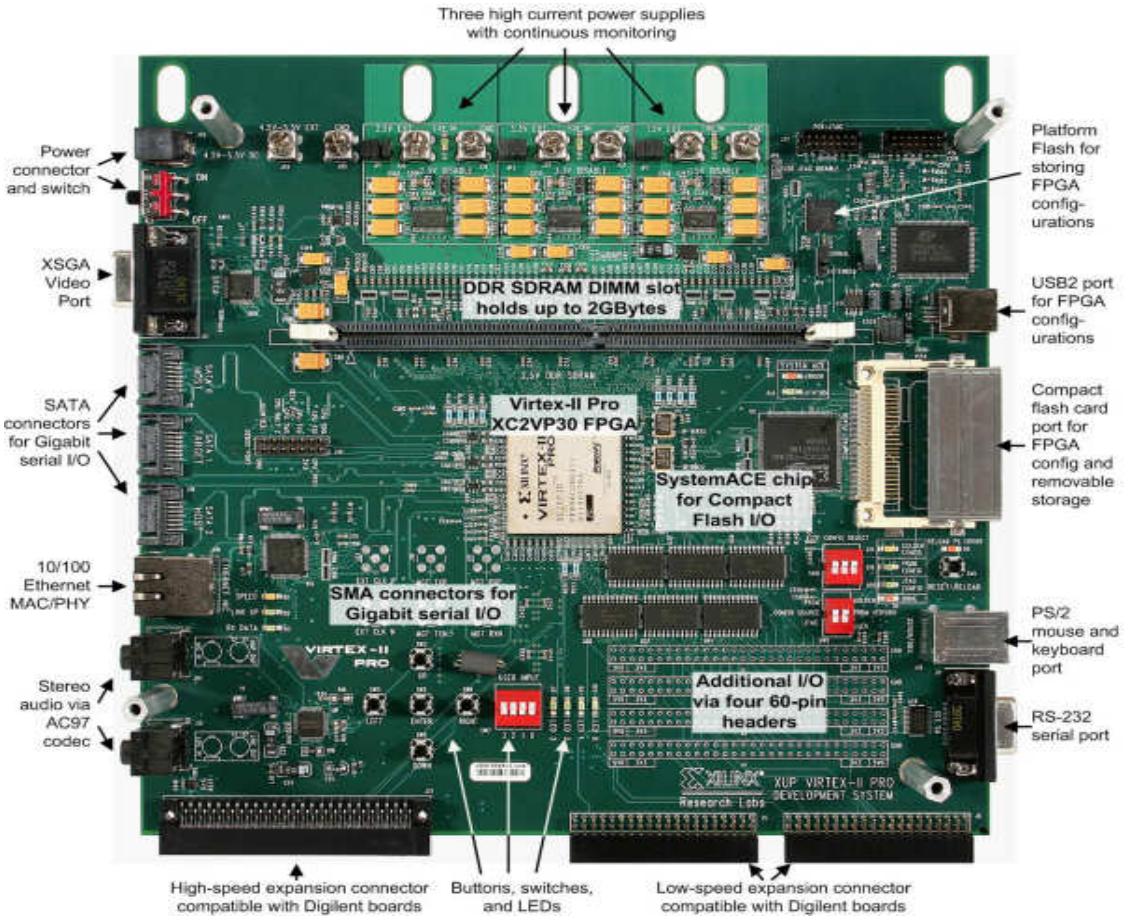


Figura 1.1 Fotografía de la placa XUP Virtex-II Pro Development System

Debido a su bajo coste, a su capacidad de reprogramación y el poco espacio que ocupan sus circuitos, se está estudiando la posibilidad de utilizar las FPGAs en tecnología aeroespacial. Una de las aplicaciones implicaría el envío de estos dispositivos al espacio exterior. Aquí encontramos un problema real. En el espacio exterior no hay protección atmosférica tal y como ocurre en la Tierra. Este hecho tiene como consecuencia que cualquier partícula solar, al chocar con la superficie de la FPGA podría modificar su contenido, alterando por tanto su comportamiento esperado. En concreto podría modificar un bit concreto de una LUT, una interconexión o el estado del biestable [6] y [7].

Nuestra tarea es, por consiguiente, emular el choque de una partícula, modificando para ello un bit concreto, y ver cómo reacciona el circuito a dicha modificación. Nosotros conoceremos a priori cuál es la salida esperada de dicho circuito, por lo que si se produce cualquier comportamiento anómalo, sabremos detectarlo fácilmente.

2 Plataforma de inyección de errores

A efectos de llevar a cabo los propósitos comentados anteriormente deberemos diseñar una plataforma de inyección de errores a través de la cual podamos desarrollar todas las funcionalidades requeridas para poder emular la inyección de errores en una FPGA.

Dicha plataforma se dividirá en distintos componentes tal y como se muestra en la Figura 2.1:

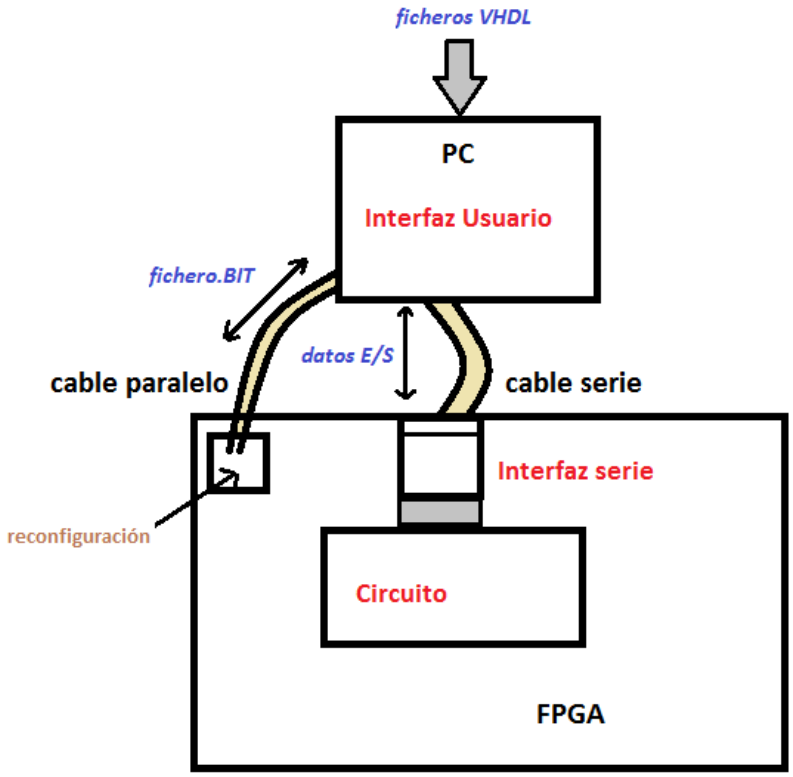


Figura 2.1 Esquema de la plataforma de inyección de errores

Se observan dos bloques principales correspondientes al PC y a la FPGA. Dentro del PC tendremos el Interfaz de usuario. En la FPGA estarán cargados el interfaz serie y el circuito que queremos probar. Las conexiones con la placa se realizarán mediante el cable serie para la entrada/salida de datos y mediante el cable paralelo para configurar la FPGA y alterar los bits de la memoria de configuración.

La idea inicial para el interfaz de usuario, es que éste no dependa de la utilización de otras aplicaciones. Es decir, que el usuario no tenga que abrir por sí mismo otras aplicaciones necesarias para las distintas tareas a realizar, tales como la generación de un fichero de configuración (.BIT) a partir de varios ficheros VHDL, la carga de dicho fichero en la FPGA, o la entrada/salida de datos mediante un puerto serie. Queremos que todo esto sea lo más transparente posible para el usuario de tal forma que no se tenga que preocupar por ciertos aspectos, y así facilitar más las cosas.

Mediante el **interfaz de usuario**, un usuario será capaz de introducir uno o más ficheros VHDL a partir de los cuales se generará un **fichero .BIT** (archivo de configuración de la FPGA). Para ello se generará automáticamente otro fichero VHDL que describa la conexión del interfaz serie al módulo superior (top) de los archivos VHDL introducidos. Tras juntar los ficheros VHDL introducidos por el usuario, más el generado automáticamente, más los ficheros de descripción del interfaz serie, se generará el fichero .BIT. La generación de este fichero será posible gracias a una serie de scripts que ejecutarán ciertos comandos involucrados en el proceso de generación de un .BIT

Este fichero .BIT será posible cargarlo también utilizando el interfaz de usuario a través del **cable paralelo**. Para ello usaremos internamente la herramienta Impact de Xilinx a la que le indicaremos con un script cómo cargar cualquier fichero de configuración .BIT a través del puerto paralelo.

Además, en el interfaz de usuario también se podrá probar la ejecución de un circuito previamente cargado en la FPGA. Para ello leeremos de un fichero las entradas de datos que se le quieren aplicar al circuito activo, y el interfaz de usuario se encargará de enviar esas entradas a través del puerto serie y recibir las salidas del circuito por la misma vía. Las entradas serán procesadas para que sean enviadas correctamente al circuito. Del mismo modo las salidas recibidas serán traducidas para poder ser visualizadas de forma legible por parte del usuario. Cada uno de los grupos de entradas/salidas que se reciben/envían representan a los distintos valores que toman en cada ciclo de reloj.

Desde el interfaz de usuario también se puede realizar la **inyección de errores en la FPGA**. Para ello combinaremos toda la funcionalidad comentada. En primero lugar generaremos un fichero .BIT a partir de una entidad concreta. Después lo cargaremos en la FPGA y tras ello elegiremos un banco de pruebas para ejecutar el modelo. Durante la ejecución del circuito con el banco de pruebas se generará una salida llamada **salida golden**, que será con la que se comparará el resto de las ejecuciones. Procesaremos el fichero de configuración para modificar un bit de la memoria de configuración de la FPGA, mediante **reconfiguración parcial**. Haremos este proceso un número determinado de veces, volviendo a ejecutar el circuito para cada bit modificado y veremos cómo ha influido la alteración en las salidas del circuito comparando con la salida golden.

El **interfaz serie** será el encargado de comunicar el PC con la placa. Para ello utilizaremos el protocolo RS232 del puerto serie. El interfaz consta de un emisor, un receptor, y una ampliación para poder enviar y recibir 32 bits que además conecta los pines de entrada y salida con la entrada y la salida del puerto serie. Como se puede observar en la Figura2.1, el interfaz serie estará cargado en la FPGA junto con el circuito principal, siendo el intermediario entre el PC y la FPGA.

2.1. Interfaz serie

Como en todo sistema, necesitábamos poder leer y escribir datos en nuestra FPGA. Teníamos que conseguir una forma de comunicarnos con la placa. En cursos anteriores, habíamos trabajado con FPGA's –en este caso de una placa basada en el modelo Spartan III- para las cuales introducíamos las entradas utilizando los switches y los botones de los que viene provisto dicha placa, y visualizábamos las salidas tanto a través de los leds como del display de 7 segmentos que tenía aquel modelo de placa.

Esa metodología nos es insuficiente para nuestro propósito por varios motivos. En primer lugar, nos ofrece muy poca versatilidad y margen de maniobra para ejecutar y probar circuitos que tengan muchas entradas o salidas. En segundo lugar, es incómodo y poco eficaz estar utilizando en todo momento switches y botones si queremos, por ejemplo, introducir un gran número de instrucciones en la FPGA (simular muchos ciclos de reloj). Además no tendríamos forma de registrar las salidas obtenidas si no es apuntándolas con papel y lápiz. Por último, se hace inviable el uso de la entrada/ salida con este método.

Ha quedado claro que hay que comunicarse con la FPGA de otra forma. Una de las formas de comunicarse con la placa es mediante el puerto serie RS232. Por tanto tendremos que diseñar un módulo que nos permita comunicar nuestra placa con en puerto serie del PC. En la Figura 2.2 se muestra un esquema de la comunicación mediante el puerto serie.

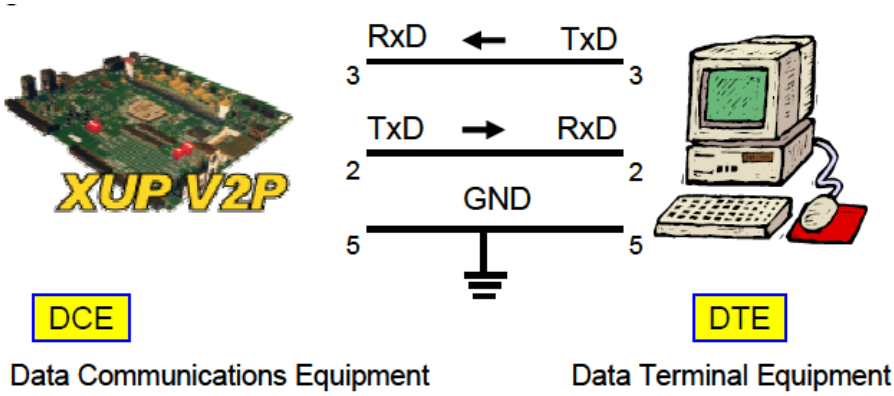


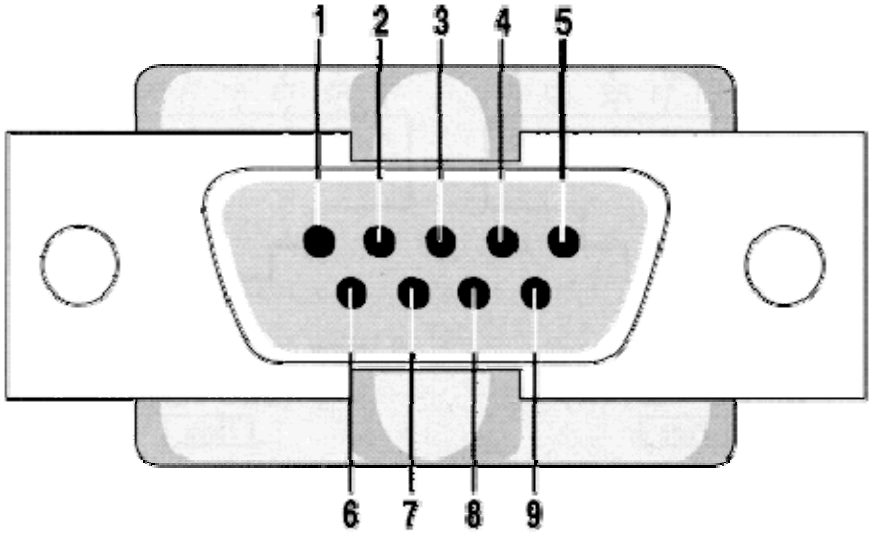
Figura 2.2 Comunicación placa-PC

A continuación explicaremos el funcionamiento de este protocolo de comunicaciones.

En la norma se definen dos tipos de terminales: DTE y DCE, donde DTE es el equipo terminal de datos (en nuestro caso el PC) y DCE es el de comunicación de datos (en nuestro caso la placa).

El puerto serie se compone de 9 pines tal y como se muestra en la Figura 2.3.

Para nosotros, sólo serán imprescindibles tres señales: la línea de transmisión de datos (Transmitted Data, **TxD**), la línea de recepción de datos (**RxD**) y la toma de tierra (Signal Ground, **GND**).



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Figura 2.3 Pines de un cable serie

Para la conexión se utiliza un cable con conector db9 como el de la figura de arriba. Los pines de la FPGA que están conectados con el puerto RS232 se muestran en la Tabla 2.1:

Nombre pin	Tipo	Nombre en FPGA (ucf)
RS232_TX_DATA	**Salida**	**AE7**
RS232_RX_DATA	**Entrada**	**AJ8**
RS232_DSR_OUT	**Salida**	**AD10**
RS232_CTS_OUT	**Salida**	**AE8**
RS232_RTS_IN	**Entrada**	**AK8**

Tabla 2.1 Pines de la FPGA que se conectan al Puerto serie

Existen distintas velocidades de transmisión de datos, que se definen en bits por segundo (bps) o baudios (921600, 460800, 230400, 115200, 57600, 38400, 19200, 9600, 4800, etc). En nuestro caso enviaremos datos a una velocidad de **9600 bps**.

La línea serie permanece a nivel alto (‘1’) mientras no se envíen datos. Cuando el transmisor va a empezar la transmisión, lo hace enviando un bit de inicio que está a nivel bajo (‘0’). Posteriormente se envían consecutivamente (en serie) los bits del dato empezando por el menos significativo. Después del último bit de dato, se envía un bit a ‘1’ para indicar el final de la transmisión (el llamado ‘bit de parada’). Este protocolo puede variar en la forma en la que se envían los datos. Existe también la posibilidad de enviar un bit de paridad y/o dos de parada. En la Fig. 2.4 se muestra el cronograma de un envío RS232 con 8 datos. En este caso se usa un bit de paridad y otro de parada.

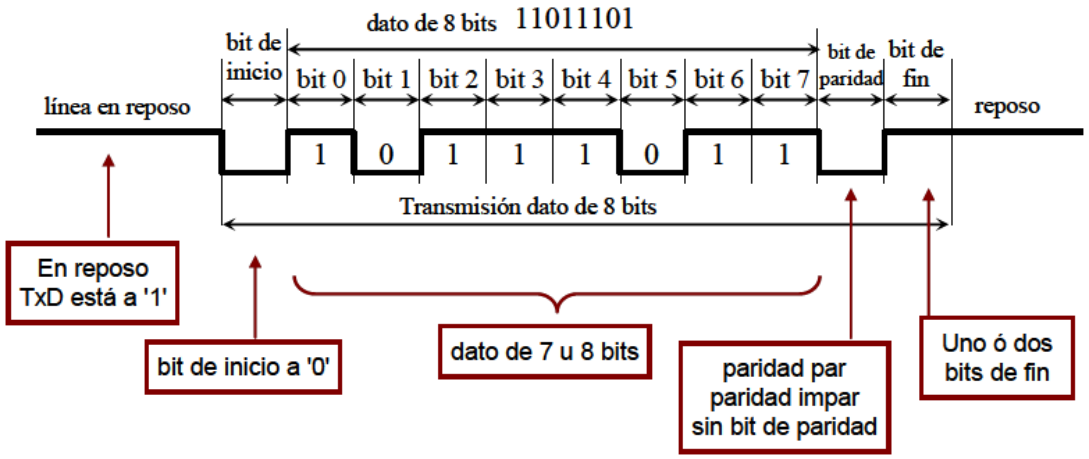


Figura 2.4. Protocolo de comunicación RS232

Una vez conocido el funcionamiento del protocolo de comunicaciones que vamos a utilizar para enviar y recibir datos de la placa, pasamos a su diseño.

2.1.1. Diseño del transmisor serie

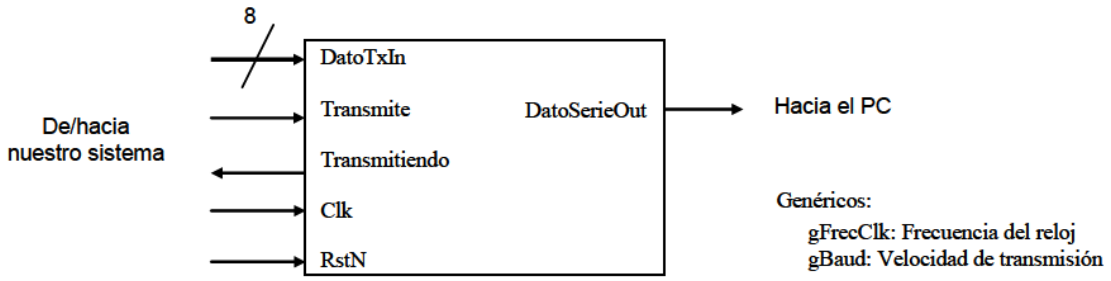


Figura 2.5 Diseño del transmisor serie

En la Figura 2.5 se muestra un esquema del diseño del transmisor serie. Los puertos de la izquierda son los que se relacionan con nuestro circuito (el que hayamos implementado en la FPGA) y el puerto de la derecha (**DatoSerieOut**), envía el dato serie al PC. Vamos a usar también dos genéricos que harán que el circuito se pueda adaptar fácilmente a distintas frecuencias del reloj, y así poder implementar el diseño en otra placa, con otras velocidades de transmisión.

Las especificaciones de los puertos se muestran en la siguiente tabla (Tabla 2.2)

Señal	Nº bits	Tipo	Descripción
RstN	**1**	**Entrada**	**Señal de reset asíncrono (activo a '0')**
Clk	**1**	**Entrada**	**Señal de reloj de la placa, en principio de 100 MHz, pero configurable por gFrecClk**
Transmite	**1**	**Entrada**	**Señal del sistema que ordena al módulo la transmisión del dato que se encuentra en DatoTxIn.**
DatoTxIn	**8**	**Entrada**	**El dato a enviar. Se proporciona de manera simultánea cuando Transmite = '1'**
Transmitiendo	**1**	**Salida**	**Señal del sistema que indica que en ese instante se está transmitiendo un dato.**
DatoSerieOut	**1**	**Salida**	**Trama de datos que se envía al PC y que sigue el protocolo RS232**

Tabla 2.2 Especificaciones de los puertos del transmisor serie

Con estas especificaciones tenemos información suficiente para hacer el transmisor:
Lo dividiremos en cuatro bloques principales, tal y como se muestra en la Figura 2.6:

- **DivFrec**: Un divisor de frecuencia. Dividirá la frecuencia de reloj tantas veces como indique gFrecClk.
- **Control**: Una máquina de estados finitos.
- **Carga_desplaz**: Un registro de carga en paralelo y salida serie.
- **Selección**: Un multiplexor que selecciona la señal de salida según el estado actual. Este multiplexor termina en un biestable para evitar pulsos no deseados, ya que su salida (DatoSerieOut) es la salida del circuito.

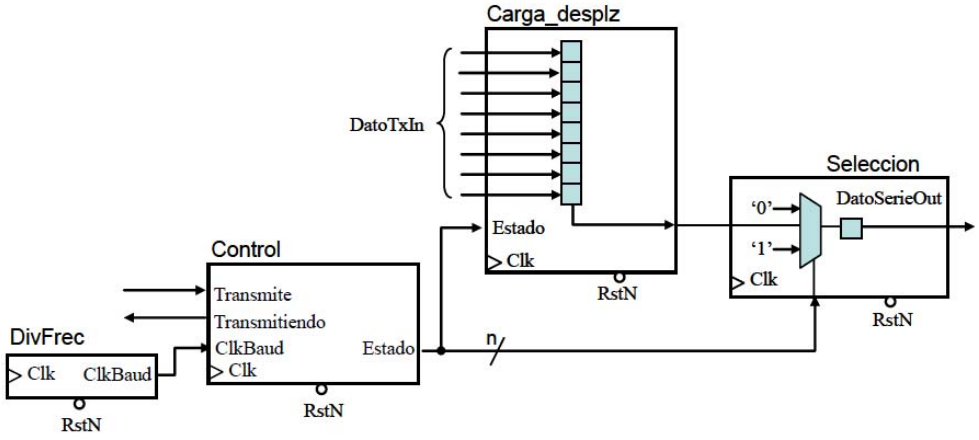


Figura 2.6 Diagrama de bloques del transmisor serie

El divisor de frecuencia es genérico, es decir, que la frecuencia de salida del divisor, se determina mediante una o varias constantes.

En nuestro diseño usaremos dos genéricos:

- gFrecClk: indica la frecuencia de reloj de la placa. Usando este genérico podremos implementar nuestro módulo de entrada/salida en otra placa que tenga un reloj diferente. En nuestro caso, la frecuencia de reloj de la Virtex II Pro es de 100 MHz.

- gBaud: indica la velocidad de transmisión de la unidad de entrada/salida. En nuestro caso, como ya dijimos anteriormente, usaremos 9600 bps.

A partir del reloj de la placa de 100 MHz (Clk), queremos proporcionar una señal con frecuencia de 9600 Hz (**ClkBaud**). Este reloj tendrá por tanto un periodo de 104,167 µs, y estará a '1' durante un solo ciclo de reloj, estando el resto de tiempo a '0'. En la Figura 2.7 se muestra cómo se divide la frecuencia de reloj.

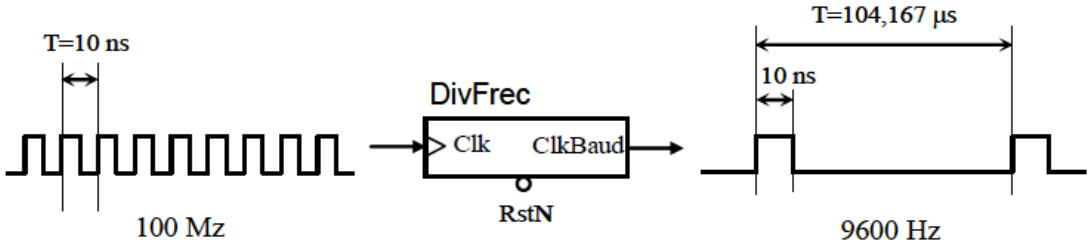


Figura 2.7 Divisor de frecuencia

Para diseñar el divisor de frecuencia se divide la frecuencia de entrada entre la frecuencia de salida (100 MHz/9.6 KHz = 10416,67 Hz) y el número resultante nos dará el contador que se necesitará para obtener la frecuencia de salida.

El circuito de control es el encargado de dirigir al resto de bloques. Si el circuito no se encuentra enviando ningún dato, cuando se recibe la indicación de transmitir (Transmite = '1'), el circuito comienza la secuencia de envío. Esta secuencia de envío será gestionada mediante la máquina de estados (FSM) que se muestra en la Figura 2.8.

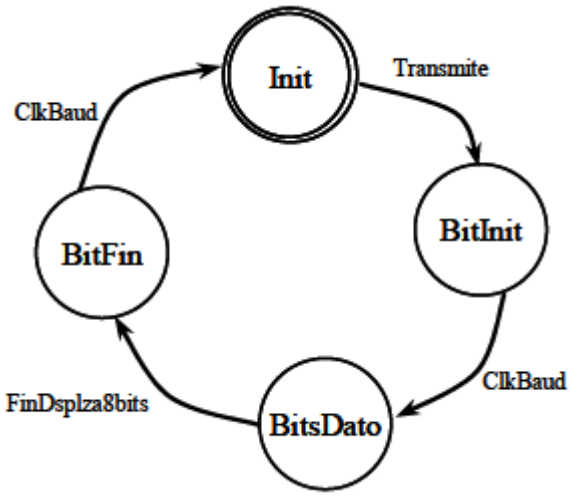


Figura 2.8 Máquina de estados del circuito de control para transmisor serie

- **Init**: Es el estado inicial. El sistema está en reposo esperando la orden de transmitir. Cuando la señal Transmite se ponga a '1', se pasará a enviar el bit de inicio, pasando para ello al siguiente estado (BitInit). En ese momento se dará la orden de cargar el

dato (DatoTxIn) en el registro de desplazamiento. También tendremos que sincronizar el contador del divisor de frecuencia; para esto tenemos dos opciones: o inicializar el contador a 0 en este estado, o hacer que en el estado inicial no cuente, y en el resto se habilite el contador. Nosotros implementaremos esta última opción.

- **BitInit**: En este estado se está enviando el bit de inicio. Se saldrá de este estado al recibir un pulso de ClkBaud, que nos dirá que debemos pasar a enviar los bits de dato. El siguiente estado es BitsDato.

- **BitsDato**: Este estado se encarga de enviar los 8 bits del dato. Utilizando un contador, llevaremos la cuenta del número de bits que se han enviado. Cuando se hayan enviado los 8 bits –es decir, cuando hayan llegado 8 pulsos de Clkbaud- se activará la señal FinDsplza8bits que hará que cambiemos al siguiente estado (BitFin).

- **BitFin**: Este estado envía el bit de fin. Al llegar el siguiente pulso de ClkBaud, cambiaremos al estado inicial Init.

Si incluimos las entradas y las salidas en la máquina de estados, nuestro diagrama quedará de la siguiente manera, tal y como muestra la Figura 2.1.8.

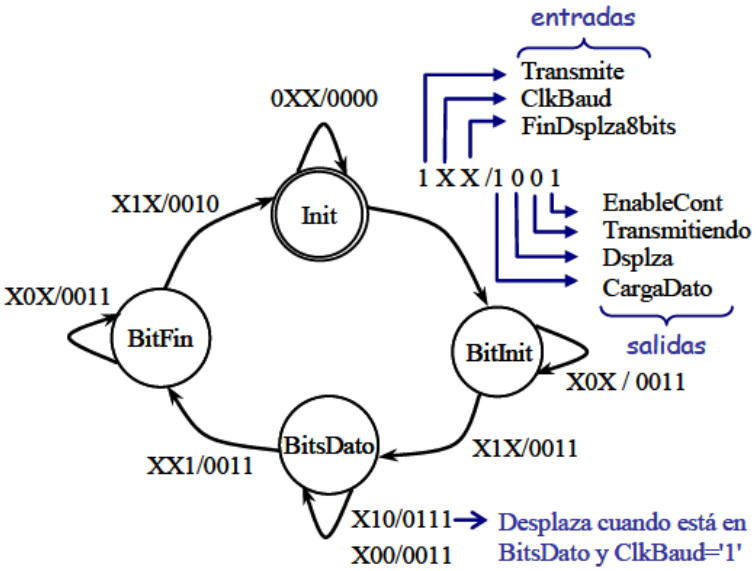


Figura 2.9 Máquina de estados con entradas y salidas

Vemos que se trata de una **máquina de Mealy**, pues las salidas no dependen solamente del estado sino también de las entradas.

Si incluimos las señales de la unidad de control en nuestro diagrama de bloques, tendremos el diagrama que se muestra en la figura 2.10.

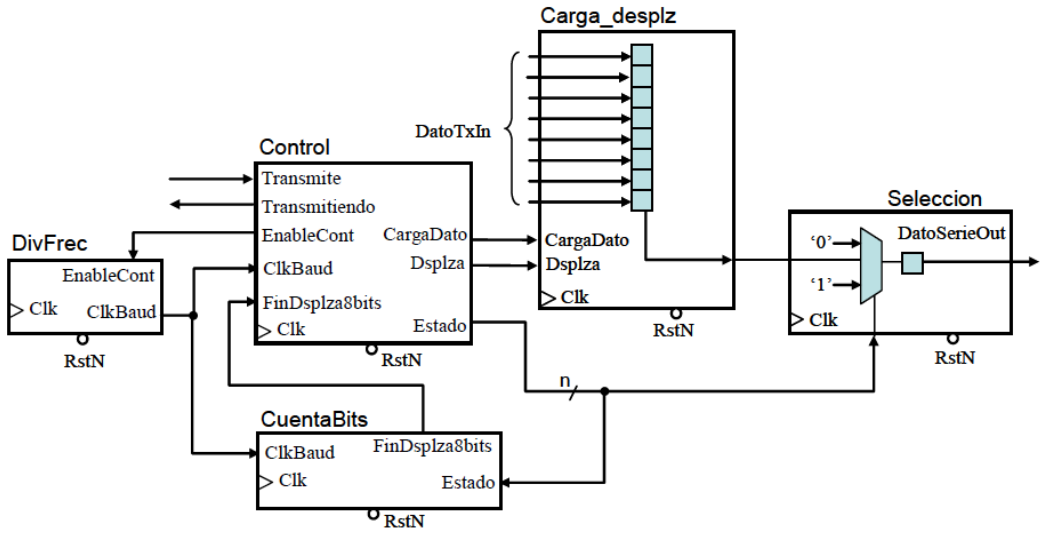


Figura 2.10 Diagrama de bloques con señales de control

2.1.2. Diseño del receptor serie

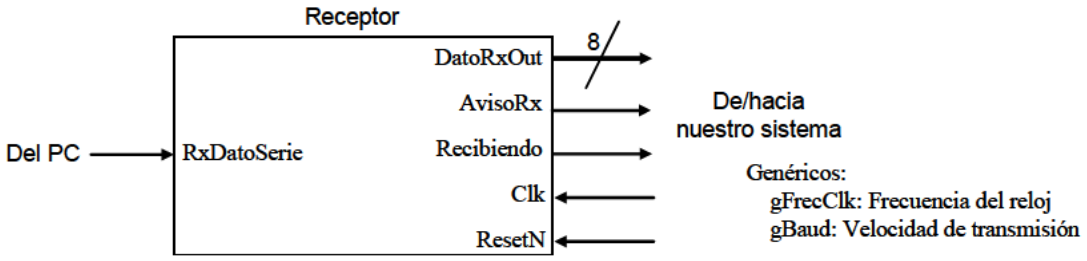


Figura 2.11 Esquema del receptor serie

En la Figura 2.11 se muestra un esquema del receptor serie. El puerto de la izquierda (RxDatoserie) es el que se encarga de recibir el dato del PC. Los puertos de la derecha se relacionan con el sistema de la FPGA. Análogamente al transmisor serie, vamos a definir los genéricos **gFrecClk** y **gBaud**. Las especificaciones de los puertos se describen en la Tabla 2.2:

Señal	Nº bits	Tipo	Descripción
RstN	**1**	**Entrada**	**Señal de reset asíncrono (activo a '0')**
Clk	**1**	**Entrada**	**Señal de reloj de la placa, en principio de 100 MHz, 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 sólo es válido desde que AvisoRx vale '1' y mientras recibiendo sea '0'.**
AvisoRx	**1**	**Salida**	**Aviso de que se ha recibido un nuevo dato y que está disponible en DatoRxOut. El aviso se dará poniendo la señal 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 válido.**

Tabla 2.2 Especificaciones de los puertos del receptor serie

En la figura 2.12 se muestra un ejemplo de cronograma del final en la recepción de un dato.

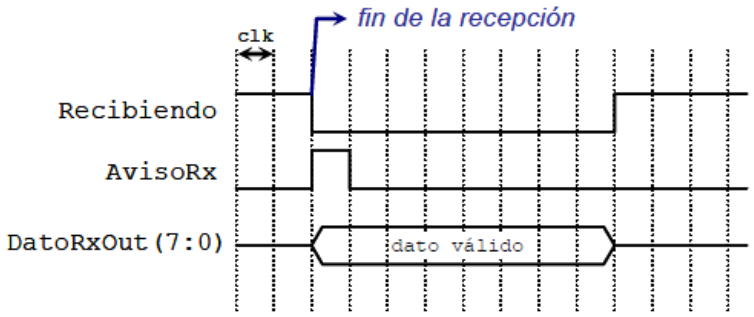


Figura 2.12 Cronograma de la recepción de un dato

El diseño del receptor se puede hacer de manera muy parecida al del transmisor serie, como se muestra en la figura 2.13

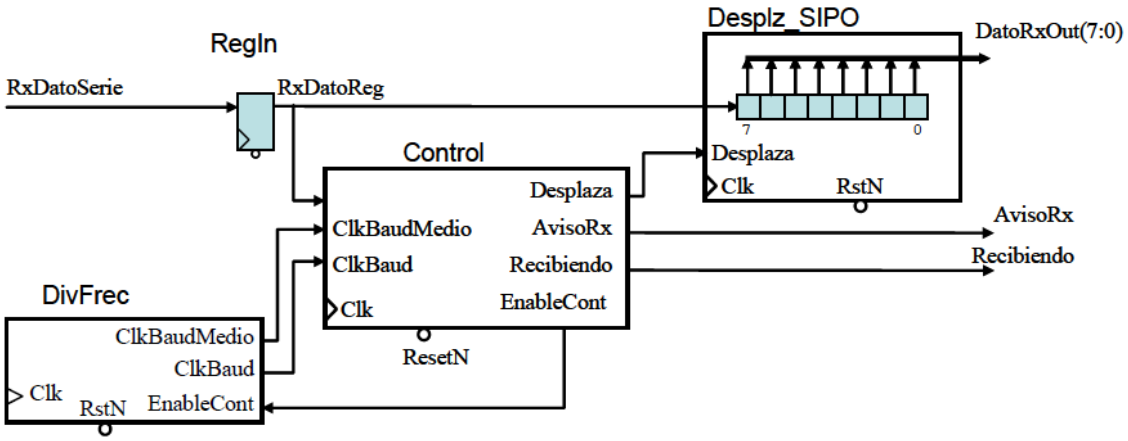


Figura 2.13 Diagrama de bloques del receptor serie

Se puede observar en la Figura 2.13 ver que también tenemos un diseño dividido en bloques.

Para el receptor, el divisor de frecuencia debe sincronizarse con la trama que se recibe porque el receptor no es el que dirige la transmisión. El divisor de frecuencia se mantiene inactivo y a '0' hasta que **EnableCont** se pone a '1'. El divisor de frecuencia tiene dos salidas: **Clkbaud** y **ClkBaudMedio**. ClkBaud marca la transición de estado mientras que ClkBaudMedio marca el punto medio de cada bit de la transmisión. Con esto nos aseguramos que el dato está estable en este punto ya que es el punto intermedio y por tanto el bit recibido es el correcto, evitando evaluar el bit cerca de las transiciones donde se puede tomar el valor del bit anterior o siguiente. En la figura 2.14 vemos cómo se activa ClkBaudMedio en la mitad del bit de recepción.

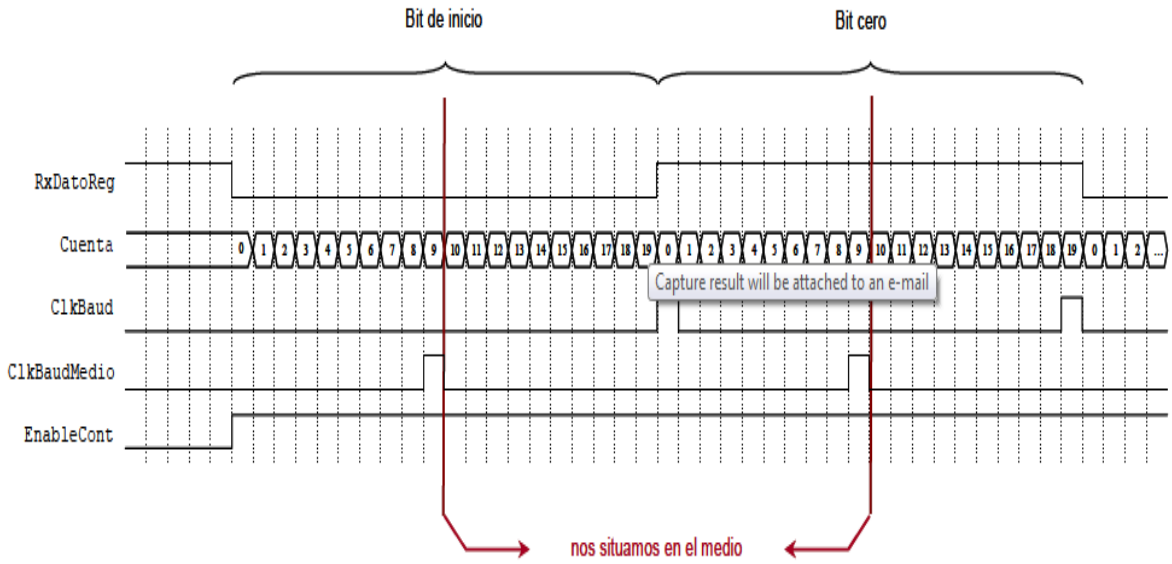


Figura 2.14 Cronograma de la recepción serie usando ClkBaudMedio

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. Es justo el registro opuesto al que teníamos en el transmisor (en el que cargábamos en paralelo y leíamos uno a uno). Como el primer bit que se recibe es el 0, si la carga serie se hace por el bit más significativo del registro y se hacen desplazar los bits hacia la derecha, en el último desplazamiento el bit 0 recibido estará en el bit menos significativo del registro, estando así todos los bits ordenados. La carga y el desplazamiento se realizan bajo la orden de la señal Desplaza que sale del bloque de control.

Como la entrada de la comunicación serie **RxDatoserie** es asíncrona, se almacena en un registro **RxDatoReg** para evitar pulsos y entradas no deseadas.

El circuito de control es similar al explicado para el transmisor serie.

Una vez diseñados el emisor y el receptor serie ya podíamos enviar y recibir datos de la FPGA. Lo primero que hicimos fue diseñar un contador muy simple (Figura 2.15) para probar nuestro módulo de entrada/salida. Se trataba de un contador módulo 16 con carga en paralelo que funcionaba a 1 Hz, es decir contaba cada 1 s. Para ello utilizamos una señal Cuenta que se iría incrementando en una unidad en cada pulso de reloj. Al llegar a 100.000 (número de ciclos que se ejecutan en un segundo) la salida se incrementaría en una unidad. Cuando la salida fuera 15 (todo 1's) en el siguiente ciclo se resetearía a 0. El contador también tendría una entrada de enable y otra de reset asíncrona. En la figura 2.15 se muestra un diagrama del contador inicial.

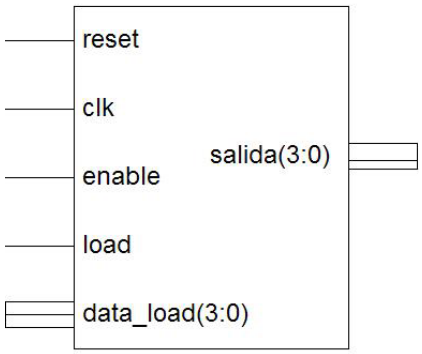


Figura 2.15 Circuito inicial (Contador)

Nuestra idea inicial fue la de codificar y decodificar los datos de entrada/salida de tal forma que cada combinación distinta de entradas se interpretaba de forma distinta para las entradas de nuestro circuito, en este caso el contador. La interpretación se hacía como se muestra en la tabla 2.3:

		Entradas al contador			
Byte recibido	Carácter ASCII	Enable	Reset	Load	Data_Load
0110 0101	e	1	0	0	xxxx
0111 0010	r	0	1	0	xxxx
0011 0000	0	0	0	1	0000
0011 0001	1	0	0	1	0001
…….	….	…	..	..	..
0011 1111	?	0	0	1	1111

Tabla 2.3 Codificación datos de entrada

Pero esto tiene un claro problema. Varias entradas del contador no pueden estar simultáneamente a '1'. En este caso, con cada una de las combinaciones sólo está a '1' una de las entradas de un bit. La forma de arreglar esto es haciendo corresponder a cada bit de entradas un bit de cada byte recibido en la FPGA. Así por ejemplo haríamos corresponder al enable el bit cero (el menos significativo) de la transmisión, al reset el bit uno y así sucesivamente. En caso de existir menos de 8 entradas el resto de bits simplemente se ignorarían. Un ejemplo de este circuito se muestra en la figura 2.16.

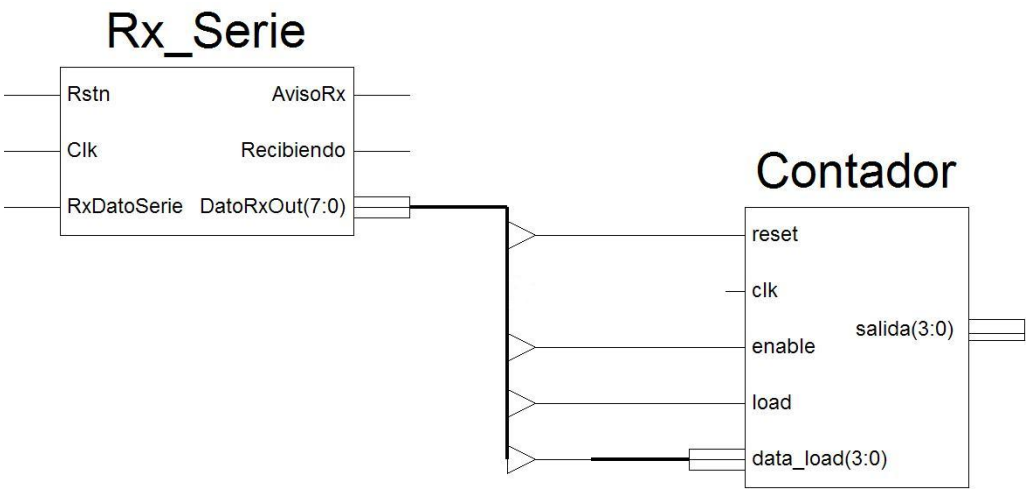


Figura 2.16: Receptor – Contador

De forma análoga, conectaríamos las salidas del contador al transmisor serie de tal forma que las pudiéramos leer en el PC, tal y como muestra la figura 2.17:

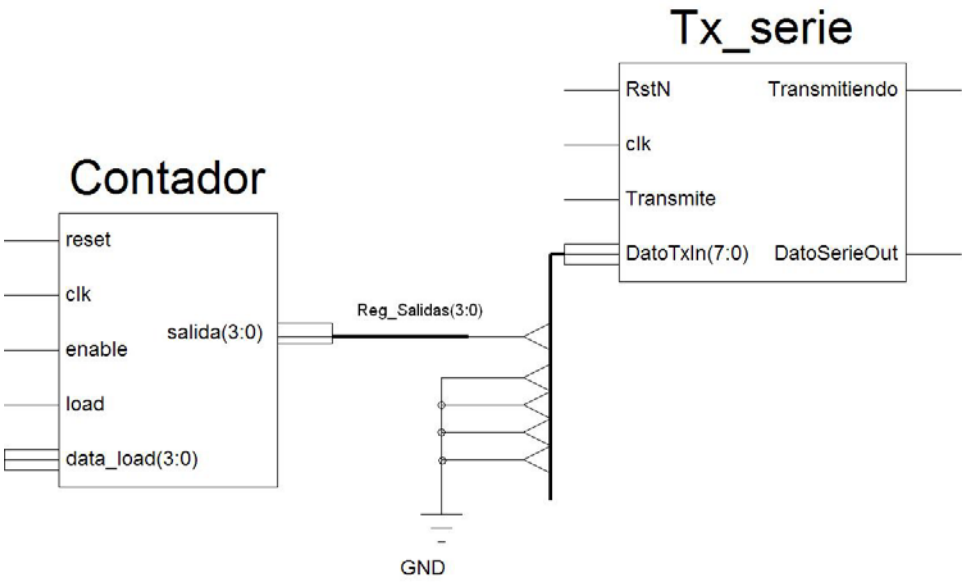


Figura 2.17: Contador - Transmisor

Nuestro circuito completo quedaría de la siguiente manera (Figura 2.18).

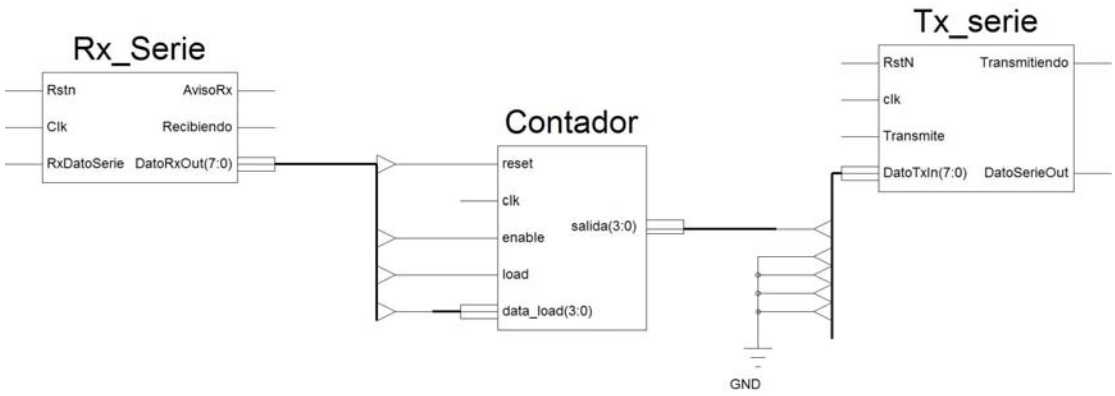


Figura 2.18: Receptor – Contador – Transmisor

Nótese que la única entrada del circuito que no se hace corresponder con ninguno de los bits de recepción es la entrada de reloj. Esta entrada como podíamos imaginar es especial y le asignaremos la propia entrada de reloj de la placa.

Ahora no tendríamos más que generar un .bit con este circuito y cargarlo en la FPGA. Pero, ¿qué ocurre si queremos conectar cualquier otro circuito a nuestro módulo de entrada/salida? ¿Tenemos que volver a reescribir el código vhdl que haga que se conecten las entradas y las salidas con el módulo? ¿No se podría hacer esto de forma automática? La respuesta es sí y lo detallaremos en el apartado 2.2.1.

2.1.3. Ampliación a 32 bits

Anteriormente hemos visto que podíamos leer y escribir datos fácilmente mediante el puerto serie RS232 utilizando como ejemplo un contador muy simple. Éste tenía 7 entradas (sin contar la entrada clk), y 4 salidas. Nos sobraban pues, una entrada y cinco salidas para asignar al módulo de entrada/salida. Sin embargo esto es muy poco escalable. A la hora de querer probar otros circuitos, vimos que estábamos muy limitados por el número de entradas y salidas. Por tanto tomamos la decisión de ampliar la posibilidad de leer y escribir hasta 32 bits.

Como vimos anteriormente, el protocolo RS232 sólo es capaz de mandar 8 bits cada vez que envía o recibe. Por tanto vimos que no podíamos modificar el módulo de entrada/salida que habíamos diseñado. Tendríamos que adaptar nuestro módulo top que se generaba automáticamente a este nuevo modelo de 32 bits.

Para poder recibir 32 bits deberemos utilizar una máquina de estados, en este caso de cuatro estados, en cada uno de los cuales se reciba un byte. En la figura 2.19 se muestra el diagrama de estados.

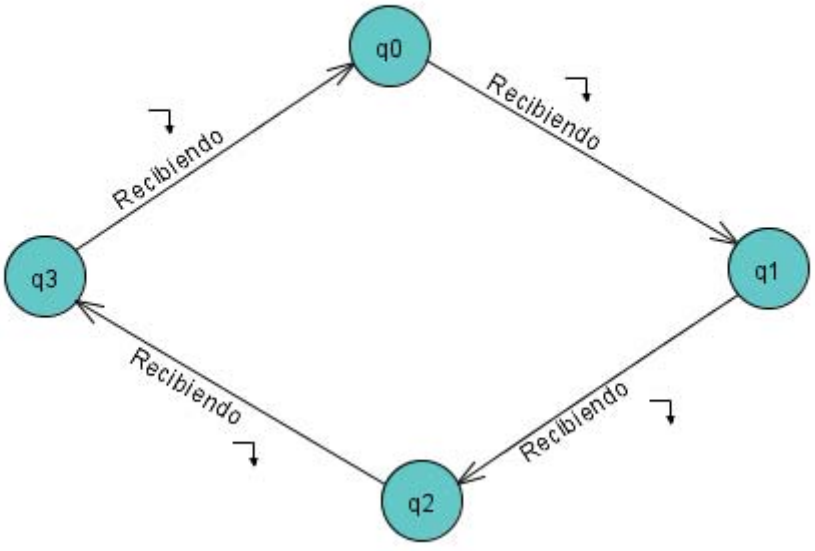


Figura 2.19 Diagrama de estados para la recepción serie de 32 bits

Cada vez que haya un flanco de bajada de la señal Recibiendo significará que se acaba de recibir un dato satisfactoriamente. Cuando esto suceda y, dependiendo del estado en el que nos encontremos se asignará el dato leído a las entradas correspondientes de un registro al que llamaremos Reg_entradas y que será el encargado de ofrecer las entradas al circuito. A continuación (Figura 2.20) se muestra un diagrama del circuito resultante utilizando como circuito general un contador con carga en paralelo de 16 bits y salida de 32 bits.

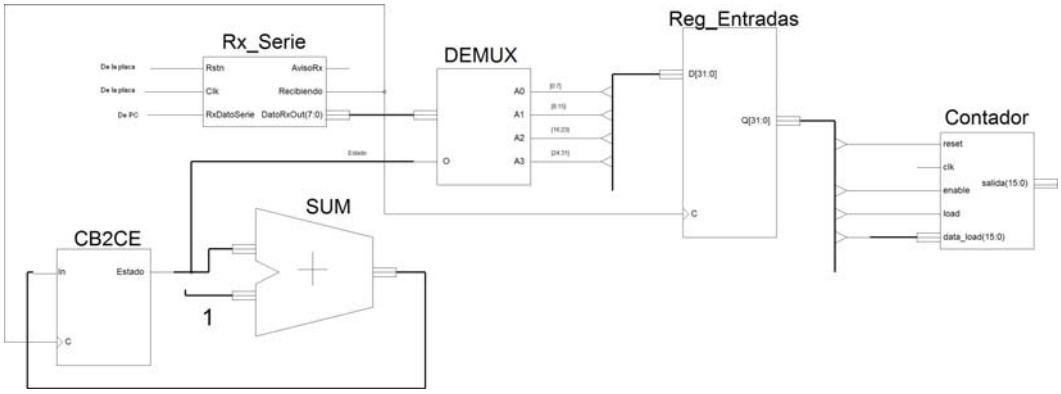


Figura 2.20 Diagrama del circuito con E/S de 32 bits usando un contador como ejemplo

En el cuarto ciclo de lectura (cuando se hayan recibido cuatro pulsos de bajada de Recibiendo) se activará una señal llamada **fin_recepcion** para indicar que la recepción ha concluido. Esta señal tiene una doble función. En primer lugar servirá como reloj del circuito general. La motivación de esto es que no queremos que se ejecute un ciclo de reloj si las entradas aún no están listas. Por otro lado, generará un flanco que durará un solo ciclo y que hará que se active la señal de transmisión.

Para la transmisión serie de 32 bits también hemos utilizado una máquina de estados. Esta máquina está compuesta también por cuatro estados. En este caso hacemos el mismo proceso pero a la inversa. Vamos a tener que prestar especial atención a algo que no se producía en la recepción de datos. Al recibir sabemos por el bit de inicio que la recepción ha comenzado. Esto nos da tiempo para hacer la asignación correcta de las entradas correspondientes para que estén listas justo cuando la recepción de cada byte termina. Ahora tenemos que tener lista la asignación de salidas antes de que se produzca la transmisión. Por esto introduciremos dos procesos para el caso de la transmisión de datos. Uno que se encargue de cambiar de estado a la máquina de estados de la transmisión de 32 bits y otro que, dependiendo del estado y del flanco positivo del reloj (para asegurar que está preparado antes de que la transmisión comience), haga las asignaciones oportunas.

En el siguiente cronograma (Figura 2.21) se muestra un ejemplo del funcionamiento.

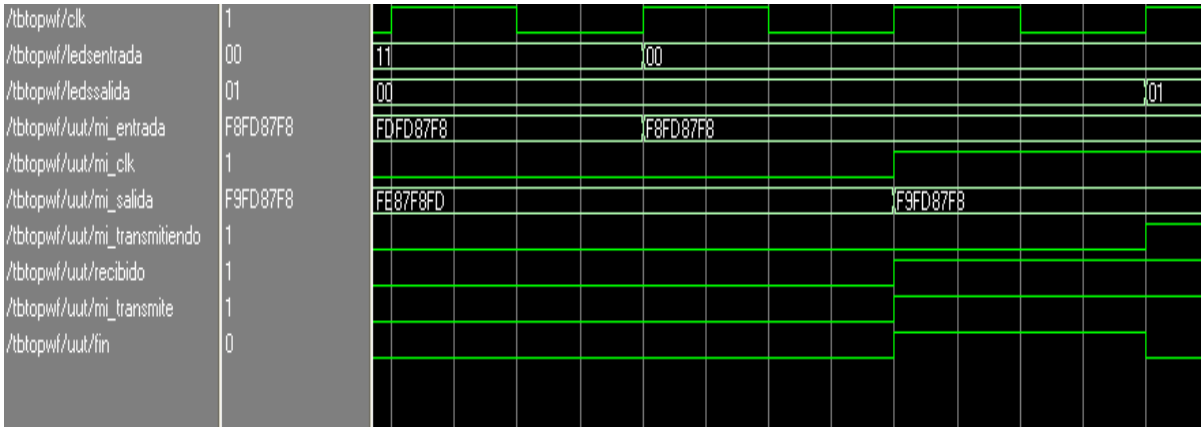


Figura 2.21 Cronograma de entrada/salida con 32 bits para un circuito que suma 01 00 00 00 a los datos de entrada

En este cronograma se puede observar cómo se realiza la entrada/salida. En este caso se ha simulado para un circuito muy simple, el cual suma 01 00 00 00 a los datos de entrada cuando recibe un flanco positivo de reloj.

La secuencia es la siguiente. El receptor termina de recibir el cuarto byte que será asignado a los 8 bits más significativos de la señal **mi_entrada** (la entrada del circuito sumador). En el caso del cronograma el valor del byte recibido es F8. Vemos cómo pasa del estado 11 al 00 (**ledsEntrada**) y la entrada se queda como F8 FD 87 F8. A continuación en el siguiente flanco positivo de reloj, la señal **recibido** se activa, para indicar que ha terminado la recepción de datos y que las entradas del circuito están listas. Esta señal **recibido** actuará además como señal de reloj del circuito sumador (**recibido** y **mi_clk** son la misma señal) y por tanto hará que cambie su salida. Se puede observar que la salida pasa a valer F9 FD 87 F8 que es precisamente la suma de los datos de entrada más 01 00 00. Además, a la misma vez que se termina la recepción, se envía un flanco de **fin** que hace que la transmisión comience.

Los datos generados por la salida del circuito sumador serán recogidos por el transmisor serie hasta que se envíen sus 4 bytes, comenzando siempre por los menos significativos. Vemos cómo el estado del receptor va cambiando (**ledsSalida**) a medida que se van transmitiendo cada uno de los bytes. En este caso se muestra la transición del estado 00 al estado 01. Cuando la transmisión de ese dato de salida termine, se generará otro flanco de 'fin' que provocará que cese la transmisión de datos poniendo la señal **mi_transmite** a '0'. De esta forma conseguimos que se transmitan 4 bytes y sólo 4.

2.2. Interfaz de usuario

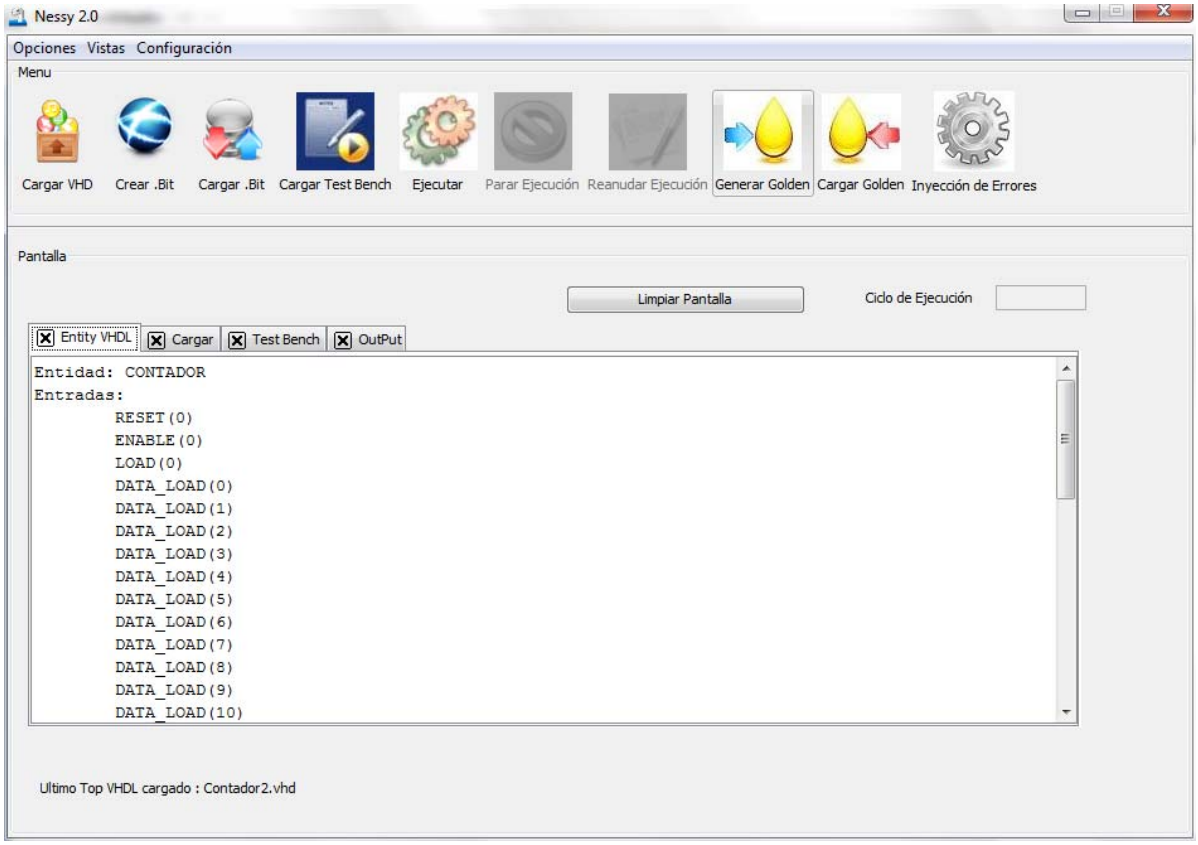


Figura 2.22 Pantalla principal de la interfaz de usuario

Como hemos comentado anteriormente, uno de los componentes de la plataforma de inyección de errores es el interfaz de usuario. Éste interfaz proporciona al usuario la funcionalidad necesaria tanto para cargar un fichero de configuración en la FPGA como para ejecutar el circuito cargado. Éstas son a grandes rasgos las principales funciones que nos permite realizar la interfaz de usuario. A partir de ellas podremos desarrollar el resto de funcionalidad.

A continuación, desglosaremos cada una de las posibilidades que ofrece según aparecen en el interfaz de usuario.

Las diferentes acciones que podemos realizar desde el interfaz de usuario son:

1. **Cargar VHD:** Este es el primer paso que debemos efectuar antes de poder tener acceso al resto de la funcionalidad. Lo primero que debe saber el interfaz de usuario es con qué entidad se va a comunicar. Al pulsar este botón podremos cargar uno o varios ficheros VHDL. De todos ellos deberemos elegir un **fichero top**. Al cargarlo, se generará internamente un nuevo fichero VHDL llamado **Circuito_FPGA.vhd**, el cual se encargará de conectar el top con el interfaz serie. Para hacer esto se leerá la entidad principal y se cargará en memoria de tal forma que tengamos acceso a sus características (entradas y salidas) en todo momento. Esto se realizará mediante técnicas de procesamiento de lenguajes, como veremos en el apartado 2.2.1.

Las rutas de todos los ficheros VHDL que se han elegido serán incluidos en un **fichero .prj** a partir del cual se podrá generar posteriormente un fichero .BIT en caso de ser necesario.

2. **Crear .BIT:** Este paso es opcional. Puede ocurrir que ya tengamos creado un fichero .BIT correspondiente a la entidad que queremos ejecutar. Teniendo en cuenta que el proceso de creación de un fichero .BIT puede durar varios minutos, si sabemos que este fichero ya está generado –por ejemplo en una ejecución anterior-, no tendremos que crearlo de nuevo. Simplemente conociendo su ubicación no tendremos más que volverlo a cargar en la FPGA usando el botón Cargar .BIT.

   Se usará una serie de scripts para generar el fichero de configuración. Es aquí donde se usará el **fichero .prj** para indicar todos los ficheros que se usan. Además de los introducidos por el usuario, se incluirán interfaz serie y el generado automáticamente (Circuito_FPGA.vhd).

   Los scripts (ficheros.bat que han de estar situados en la carpeta **comandosXilinx**) ejecutarán distintos comandos de Xilinx que harán que se genere el **bitstream.** La ejecución de estos scripts se podrá visualizar mediante la pantalla de comandos de Windows, de tal forma que podamos visualizar los posibles mensajes de error que se puedan producir durante los distintos pasos de la generación del fichero de configuración. Al terminar el proceso, habrá que cerrar esta ventana manualmente.

3. **Cargar .BIT:** Al pulsar sobre este botón, el usuario podrá cargar en la FPGA el fichero de configuración que elija.

   Para cargarlo se utilizará la herramienta IMPACT de Xilinx a la cual se le pasará un fichero de **batch** que le indique cómo cargar el fichero que elija el usuario. El interfaz de usuario redireccionará los mensajes de salida que ofrezca Impact para poder visualizar el estado de la carga. La forma que tendremos de saber si se ha cargado exitosamente es si alguna de las cadenas de salida de Impact se corresponde con “Programmed succesfully”. En caso de fallo de carga, se intentará una segunda vez por si ha ocurrido algún error inesperado (el puerto paralelo está desconectado, la placa está apagada, etc). Si tras este segundo intento tampoco se ha podido cargar aparecerá un mensaje indicando que ha sido imposible cargarlo.

4. **Cargar Test Bench:** El test bench (banco de pruebas) está representado por los metadatos que enviaremos a la FPGA. Estos metadatos estarán representados mediante cadenas de 0’s y 1’s. Cada cadena representará los datos de un ciclo de reloj del circuito que estamos ejecutando.

   A la hora de cargar un banco de pruebas tenemos dos opciones. O bien cargar desde un fichero y ejecutar directamente con su contenido, o bien cargar en la pantalla el contenido del fichero antes de ejecutar. En ese caso, se podrá modificar el banco de pruebas en el propio interfaz de tal forma que se lea lo que hay en la pantalla. Cuando el banco de pruebas sea muy grande (más de 2.000 ciclos de ejecución), se recomienda la ejecución directa desde fichero, a efectos de no saturar los recursos de memoria.

   Tanto si se carga directamente como si se lee de pantalla, se realizará antes de ejecutar una comprobación del formato de las cadenas. Dichas cadenas deberán estar formadas por 0’s y 1’s y coincidir en longitud con el número de bits de entrada de la entidad cargada en ese momento.

5. **Ejecutar:** Ejecutar implica enviar los datos del test bench al circuito y recibir las salidas que produzca como consecuencia de esas entradas. Para ello se traducirán los datos en forma de cadenas de 0's y 1's a los bytes que se desea enviar. Para recibir haremos el mismo proceso a la inversa de tal forma que de una serie de bytes, obtengamos una representación en forma de cadenas de 0's y 1's.

   Mientras se ejecuta se comparará la salida obtenida con una salida golden (ubicada en **salidas/Golden.txt**)

6. **Generar/Cargar Golden:** Como hemos explicado anteriormente, la salida golden es aquella con la que vamos a comparar el resto de salidas de una ejecución determinada. Digamos que es una salida que suponemos correcta para unas entradas concretas. Esta salida puede ser cargada bien mediante un fichero, bien generada por una ejecución determinada. En este último caso, el efecto será el mismo que el de una ejecución simple, con la única diferencia de que no se comparará con ninguna golden anterior, sino que en este caso sólo se generará la salida golden. Por tanto al final no obtendremos ningún mensaje.

7. **Inyectar errores:** En este caso combinaremos todas las funcionalidades explicadas anteriormente. Cargaremos un fichero de configuración, a continuación cargaremos un banco de pruebas, y se pedirá un número de iteraciones. Las iteraciones indicarán el número de veces que se va a inyectar un error en la memoria de configuración. Se generará una salida golden con la configuración inicial cargada. A partir de ahí, en cada una de las iteraciones se modificará **aleatoriamente** un solo bit de la memoria de configuración y se volverá a ejecutar comparando con la salida golden. Al finalizar se guardará en un fichero los resultados de la inyección de errores.

2.2.1. Generación automática de VHDL

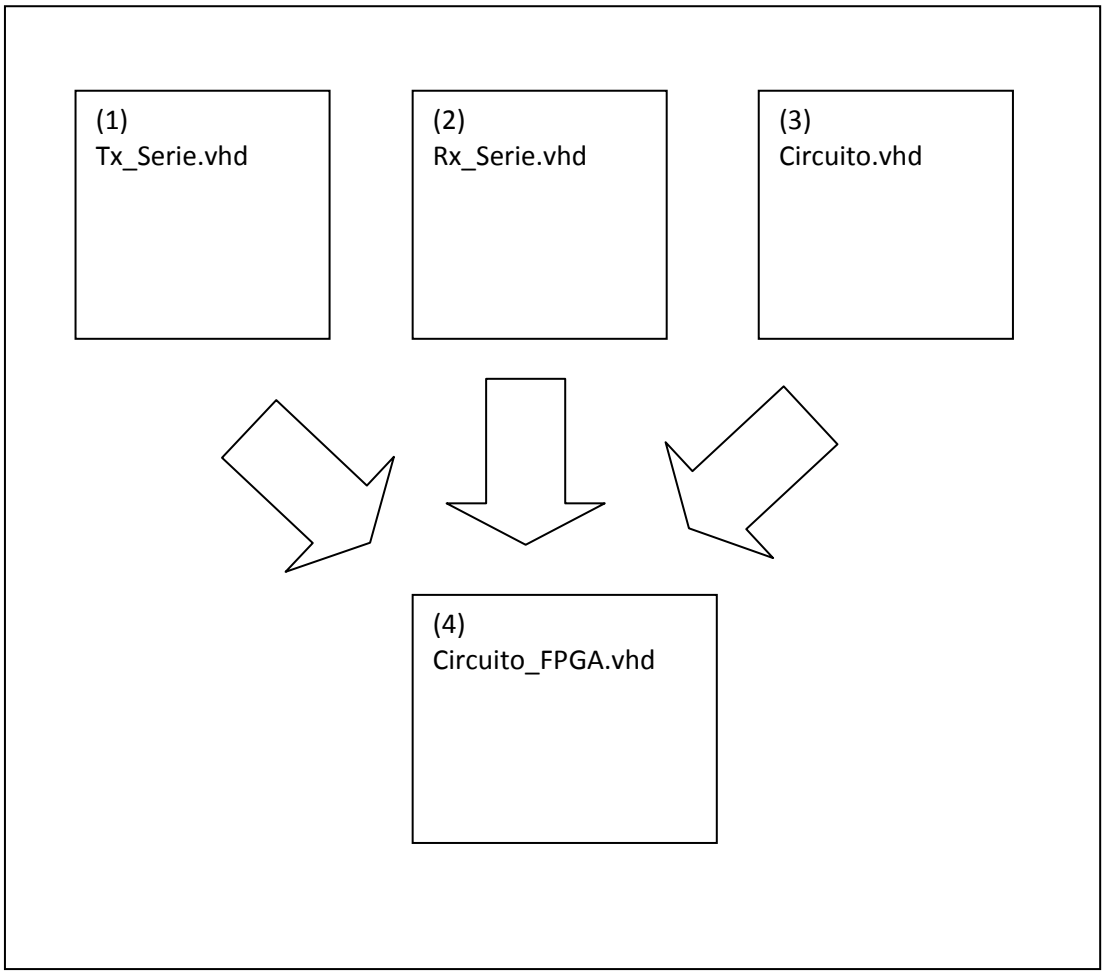


Figura 2.23 Esquema de la generación automática del circuito vhd

En la figura 2.23 se muestra un esquema de la generación automática de un fichero VHDL a partir de otros. Los ficheros implicados son:

(1) Fichero vhd de transmisión serie

(2) Fichero vhd de recepción serie

(3) Circuito introducido por el usuario (en el ejemplo anterior, un contador). Luego explicaremos que también pueden ser varios ficheros. Este es el fichero cuya entidad se procesará.

La salida es el:

(4) Circuito vhd del módulo top generado automáticamente.

La idea es poder leer cualquier circuito, descrito mediante una entidad en vhdl, y conectarlo automáticamente a nuestro módulo de entrada/salida. Para ello deberemos leer la parte del fichero que nos interese (en este caso sólo nos interesa la parte de la entidad) y cargarlo en

memoria, para a partir de esa entidad crear automáticamente un fichero vhdl –que será el módulo top del circuito- en el que estén conectadas las entradas al receptor y las salidas al emisor.

Tendremos que poner en marcha técnicas de procesadores de lenguajes que nos permitan leer de un fichero la entidad, y comprobar que está correctamente definida, tanto léxica como sintácticamente.

Es importante remarcar que en este proceso no se analiza el funcionamiento de un circuito descrito en vhdl sino únicamente sus entradas y salidas a efectos de elaborar un modelo a partir de esa entidad que nos permita posteriormente ejecutar el circuito, ofreciéndole las entradas necesarias y leyendo las salidas pertinentes.

En primer lugar debemos analizar léxicamente el fichero que contiene la entidad. En esta fase nos interesa leer cada uno de los pequeños elementos de los que se compone el fichero vhdl. Desde palabras reservadas hasta elementos tales como un punto y coma. Hay que saber separarlos para ofrecérselos al analizador sintáctico (que explicaremos más adelante). Este analizador está regido por un autómata finito, el cual se caracteriza por el diagrama de la figura 2.24.

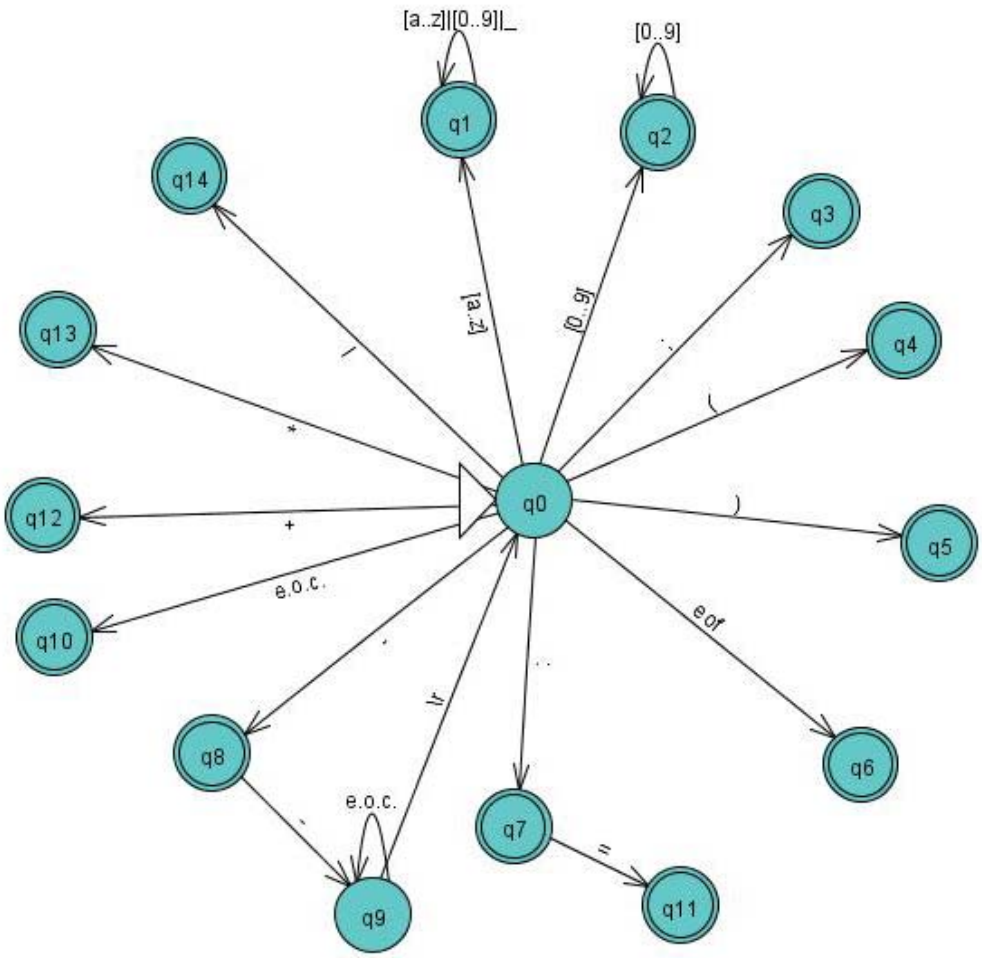


Figura 2.24: Autómata Analizador Léxico

En el siguiente cuadro (Cuadro 2.1) se muestra la gramática que se ha usado para leer la entidad y poder reconocerla correctamente:

Entidad ::= Cabecera **entity** identificador **is** Generic Puertos **end** identificador
Generic ::= **generic (**Variables**);**
Variables ::= Variable RVariables**;**
Variable ::= identificador **:** InOut **integer :=** entero
RVariables ::= **;** Variable Rvariables
RVariables ::= λ
Puertos ::= **port(**Señales**)**;
Señales = Señal RSeñales
Señal = identificador **:** InOut Tipo
RSeñales = **;** Señal RSeñales
RSeñales = λ
InOut = **in**
InOut = **out**
Tipo = **std_logic**
Tipo = **std_logic_vector(**Exp **downto** Exp**)**
Exp = Term RExp
RExp = **+** Exp | **-** Exp | λ
Term = Fac RTem
RTerm = ***** Term | **/** Term | λ
Fac = **(** E **)** | nat | identificador
Cabecera = *

Cuadro 2.1 Gramática para la lectura de una entidad

Los valores en negrita indican elementos terminales. Al comienzo del fichero, estaremos leyendo sin procesar hasta encontrar la palabra 'ENTITY', tal y como indica la producción Cabecera, donde el asterisco indica cualquier cosa excepto la palabra 'Entity'. Una vez encontrada la palabra, se empezará a procesar el contenido de la entidad. Este analizador es también capaz de reconocer genéricos, evaluarlos y sustituir sus valores donde corresponda.

Una vez procesado el fichero, tendremos en memoria cargada una entidad. De esta forma podremos hacer una asignación de cada uno de los bits recibidos en la FPGA a cada una de las entradas de la entidad automáticamente. Sin embargo, como dijimos anteriormente, hay una entrada especial que es la de reloj la cual no queremos que sea asignada a ningún bit de entrada serie, sino que queremos asignarla a la propia entrada de reloj de la placa. Por eso, cuando nuestro lector de ficheros encuentre una entrada que contenga 'clk' o 'reloj', la marcará con un atributo especial que indique que se trata de la entrada de reloj.

Puede ocurrir que la entidad no esté bien definida, en cuyo caso se indicará mediante un mensaje indicativo la línea en la que se ha producido el error, con una breve explicación del mismo. También puede ser que la entidad definida tenga un número de entradas mayor de las permitidas para este protocolo de comunicaciones (en este caso 8), por lo que también se indicará mediante un mensaje, no permitiendo la generación del fichero top vhdl.

Más adelante, en el manual de la aplicación, se explicará la forma en la que se muestran estos mensajes y la manera en la que se presenta al usuario la entidad leída.

2.2.2. Generación de fichero .BIT

Como hemos comentado anteriormente, creamos automáticamente un fichero vhd que sirve como módulo top para nuestro diseño que conecta cualquier circuito de (como máximo) 32 entradas a nuestro módulo de entrada/salida.

La motivación principal de todo esto es que el usuario seleccione un fichero vhd que quiera probar en la FPGA, y de forma transparente se genere un fichero .bit con la entrada/salida ya conectadas de tal forma que sólo tenga que indicar el conjunto de entradas que quiera introducir al circuito. Si solamente se generaran los ficheros vhd y el usuario los tuviera que introducir manualmente en la herramienta Xilinx para generar el fichero .bit, sería todo mucho más incómodo, si bien también se puede utilizar esta metodología de trabajo. La idea es automatizar también este proceso.

Para ellos deberemos utilizar los procesos que tiene Xilinx para sintetizar, traducir, mapear, *"Place and Route"* y generar .bit. Intentaremos integrarlos en la medida de lo posible con nuestra aplicación.

Pasos a seguir para la generación de un fichero .bit

En primer lugar hay que generar un fichero de proyecto. El **fichero de proyecto (.prj)** lo utiliza Xilinx para saber dónde están todos los ficheros vhd que van a ser incluidos en la generación del .bit (en concreto es el fichero de entrada al proceso de síntesis). En nuestro caso tendremos que introducir los ya conocidos del módulo de entrada/salida serie más los introducidos por el usuario más el fichero top generado automáticamente. En el cuadro 2.2 se muestra un ejemplo del fichero prj que se generará:

```
vhdl work "Tx_serie.vhd"
vhdl work "Rx_serie.vhd"
vhdl work "Circuito_FPGA.vhd"
vhdl work "E:\Universidad 2009-2010\PFC\Nessy2.0\test\FFE_FPGA.vhd"
vhdl work "E:\Universidad 2009-2010\PFC\Nessy2.0\test\FFE_TOP.vhd"
```

Cuadro 2.2 Fichero PRJ necesario para la generación de .BIT

Los tres primeros que se escriben son fijos y correspondientes a la entrada/salida y los dos últimos son los introducidos por el usuario.

A efectos de evitar conflictos de ficheros de proyecto, antes de generar cualquier otro fichero, borraremos los anteriores existentes excepto los vhd correspondientes a entrada/salida.

A continuación se ejecutarán todos los comandos necesarios para la creación del .bit. Esto se ha implementado con ficheros por lotes de comandos (.bat). Cada uno de los ficheros bat, hará una llamada al proceso correspondiente de Xilinx basándose en la configuración de fichero para determinar la ruta donde se encuentra instalado el ISE Xilinx. Todos los ficheros bat se encuentran en una carpeta llamada comandosXilinx.

Al ejecutar estos comandos se abrirá una pantalla del símbolo del sistema (Windows) la cual, al finalizar el proceso habrá que cerrar manualmente. La razón de esto es que si algún error ha ocurrido, éste pueda ser visualizado correctamente por el usuario.

Las llamadas a los comandos se harán a través de un fichero bat general llamado **COMPILAR.bat** que será el encargado de gestionar todas las llamadas y hacer los cambios de ruta oportunos entre comando y comando. En COMPILAR.bat se harán distintas llamadas a otros scripts que detallaremos a continuación. Al comienzo del script se guardará la ruta actual desde la que se arranca el comando general. De esta forma, aunque para llamar a los comandos se cambie de unidad de disco, siempre se volverá al directorio original para hacer la siguiente llamada al comando de Xilinx (ubicada siempre en la carpeta */comandosXilinx*).

De ahí que entre comando y comando se observen las siguientes líneas, como muestra el Script 1.

```
cd %pwd%
%pwd:~0,2%
```

Script 1: Órdenes entre comandos para cambiar de unidad en COMPILAR.Bat

La primera línea cambia al directorio que esté especificado en la variable 'pwd'. En caso de que dicho directorio se encontrara en una unidad distinta de disco, la segunda línea se encargaría de cambiar de unidad, ya que lo que hace exactamente es tomar los dos primeros caracteres de la variable (en este caso 'pwd').

Ejemplo: pwd= "D:\PFC\Nessy2.0"

La ejecución haría (en negrita se muestra el estado de la ventana de comandos, en este caso se arranca en C:\).

C:\Xilinx10.1
cd D:\PFC\Nessy2.0 --esto no cambia el directorio si está en otra unidad
C:\Xilinx10.1
D:
D:\ PFC\Nessy2.0

Para la síntesis se utiliza el comando **XST** de Xilinx. Durante la síntesis se analiza el código HDL y se intentan inferir bloques de diseño específicos tales como multiplexores, RAMs, sumadores y restadores para los cuales pueda crear implementaciones eficientes. También se reconocen las posibles máquinas de estados (FSM).

Utilizaremos un fichero script como guía para la síntesis. Este fichero será el que contenga realmente las opciones del comando.

De esta manera escribiremos en un fichero llamado sint.txt lo siguiente:
(donde %cd% es el directorio actual). Este script (Script 2) se encuentra en **sintetizarXST.bat.**

```
echo run  %cd%\..\IOSerie\Circuito_FPGA.prj -ifmt mixed -top Circuito_FPGA -ofn
%cd%\..\IOSerie\Circuito_FPGA.ngc -ofmt NGC -p xc2vp30-7ff896 -opt_mode Speed -
opt_level 1 > sint.txt
```

Script 2: sintetizarXST.Bat (1) Escritura en sint.txt de los comandos necesarios para la síntesis.

A continuación (Script 3) haremos la llamada al comando XST pasándole el fichero que acabamos de generar.

```
C:\Xilinx10.1\ISE\bin\nt\xst.exe -ifn "%cd%\sint.txt"
```

Script 3: sintetizarXST.bat (2). Llamada al comando xst.exe con el fichero de comandos generado como parámetro

Donde (C:\Xilinx10.1) sea la ruta que está determinada en el fichero de configuración para Home Xilinx.

La explicación del commando se presenta en la tabla 2.4:

ifmt	Indica el formato de los ficheros de entrada. Para permitir varios tipos de ficheros nosotros permitiremos entrada mezclada (mixed) aunque podríamos poner solamente vhdl.
top	Especifica el del módulo top. En nuestro caso siempre va a ser Circuito_FPGA
ofn	Especifica el nombre del fichero de salida.
ofmt	Indica el formato del fichero de salida. En este caso queremos que sea NGC.
p	Dispositivo de destino. Introducimos los parámetros para nuestra Virtex II-Pro.
opt_mode	Define la estrategia de optimización en la síntesis. Al poner Speed como en este caso, lo que hacemos es intentar reducir el número de niveles lógicos y así incrementar la frecuencia. También podríamos definir que se optimice el área de tal forma que los circuitos se ajusten mejor al área disponible.
opt_level	Define el nivel de optimización de la síntesis. Nosotros utilizamos el nivel 1 que es nivel por defecto, para que no se consuma demasiado tiempo en el proceso de síntesis, puesto que nuestros circuitos no van a ser nunca especialmente complejos.

Tabla 2.4: Opciones de comando XST

Ficheros de entrada: Circuito_FPGA.prj
Ficheros de salida: Circuito_PGA.ngc

El proceso de traducción se lleva a cabo con el comando **NGDBUILD**. Este comando realiza todos los pasos necesarios para leer un fichero en formato **NGC** (el obtenido mediante la síntesis con XST) en el que se encuentra la **netlist del proyecto**, y crear un fichero **NGD** (Native Generic Database) que describe el diseño lógico en términos de puertas lógicas, decodificadores, flip-flops y RAMs. El fichero NGD puede ser mapeado a la familia de FPGAs que se elija.

El comando NGDBUILD necesita que haya creada una **variable de entorno** llamada XILINX (el nombre es obligado) en la que se encuentra una librería dinámica que necesita NGDBuild para ejecutarse. Por ello, lo primero que haremos será crear esta variable de entorno (Script 4):

```
set XILINX=C:\Xilinx10.1\ISE\
```

Script 4: Creación de variable de entrono Xilinx

Tras ello, haremos la llamada al comando de la siguiente manera (Script 5):

```
C:\Xilinx10.1\ISE\bin\nt\unwrapped\ngdbuild.exe -uc %ioserie%\constraints.ucf -intstyle
xflow -dd _ngo -nt timestamp -p xc2vp30-ff896-7 %ioserie%\Circuito_FPGA.ngc
%ioserie%\Circuito_FPGA.ngd
```

Script 5: NGDBuild.bat

La explicación del comando se presenta en la tabla 2.5.

uc	Indicamos que estamos introduciendo un fichero de restricciones. Este fichero lo especificaremos más adelante.
dd	Indica el directorio en el que ubicaremos los ficheros de salida.
nt	Indica la forma en la que se tratan los timestamps al llamar al comando NGDBUILD. Un timestamp es la información relativa a la fecha y la hora en la que un archivo fue creado. Las opciones pueden ser timestamp|on|off. En nuestro caso está en la opción timestamp, de tal forma que se ordena al comando que chequee de forma normal el timestamp y actualice los ficheros ngo de acuerdo con esta información.
p	Especifica la parte en la cual se implementa el diseño. Puede especificar sólo una arquitectura, una parte completa (dispositivo, paquete y velocidad) o una especificación

Tabla 2.5: Opciones del comando NGDBUILD

A continuación se indican los ficheros de entrada y de salida.

Ficheros de entrada: Circuito_PGA.ngc, constraints.ucf
Ficheros de salida: Circuito_PGA.ngd

Después de la traducción se ejecuta el proceso de **mapeo** sobre el diseño. Este proceso utiliza el fichero NGD creado en la traducción y mapea el diseño lógico a la FPGA tras comprobar unas reglas de diseño determinadas. Los resultados se escriben en un fichero **NCD** (Native Circuit Description) que será usado más adelante para el proceso de Place and Route.

Para el proceso de mapeo se hace la llamada al comando **MAP** (Script 6):

```
C:\Xilinx10.1\ISE\bin\nt\map -intstyle xflow -p xc2vp30-ff896-7 -cm area -pr off -k 4 -c 100 -
tx off -o %ioserie%/Circuito_FPGA_map.ncd %ioserie%/Circuito_FPGA.ngd
%ioserie%/Circuito_FPGA.pcf
```

Script 6: Map.bat

La explicación de este comando se presenta en la tabla 2.6:

p	Para indicar dispositivo, paquete y velocidad
cm	Especifica el criterio utilizado en la fase de de cobertura del mapeo. En esta fase, el proceso asigna la lógica a los generadores de funciones de los CLBs, es decir a los LUTs. Se pueden usar las opciones *area*, *speed* y *balanced*. Nosotros utilizamos *area* para que dé prioridad a la reducción del número de LUTs y por tanto también del número de CLBs.
pr	Especifica que los flip-flops o latches pueden estar empaquetados en registros de entrada salida. En nuestro caso esta opción está desactivada.
c	Pack Slices. Esta opción determina el grado en el cual las slices utilizan empaquetado no relativo cuando el diseño está mapeado. El valor por defecto es 100 que es el que utilizaremos.
o	Especifica el nombre del fichero de salida para el diseño.

Tabla 2.6: Opciones del comando MAP

Ficheros entrada: Circuito_FPGA.ngd
Ficheros salida: Circuito_FPGA.ncd, Circuito_FPGA_map.ncd, Circuito_FPGA.pcf

El siguiente proceso a ejecutar es el llamado **Place and Route (PAR)**. Este proceso es el encargado de colocar y rutar lo que hay en el fichero NCD generado mediante el proceso de mapeo. Al hacerlo, genera un fichero de salida NCD que será el que lea el proceso encargado de generar el fichero BIT (proceso BitGen). El proceso de colocación y rutado puede ser dirigido por temporización (timing driven) o por tablas de coste (cost-based). Nosotros lo haremos de esta segunda manera; de esta forma se realiza utilizando varias tablas de coste (cost tables) que asignan pesos a factores relevantes tales como restricciones, longitud de conexión y recursos disponibles.

La llamada al comando PAR se hace como muestra el Script 7:

```
C:\Xilinx10.1\ISE\bin\nt\par -w -intstyle ise -ol std -t 1 %ioserie%/Circuito_FPGA_map.ncd
%ioserie%/Circuito_FPGA.ncd %ioserie%/Circuito_FPGA.pcf
```

Script 7: PAR.bat

La explicación del comando se presenta en la tabla 2.7:

w	Esta opción sirve para sobrescribir el fichero NCD en caso de que éste existiera.
ol	Overall Effort Level. Para hacer que el proceso sea más o menos rápido. Nosotros ponemos la opción std que es la opción por defecto (la más rápida).
t	Especifica la tabla de costes. Usamos el valor por defecto que es 1.

Tabla 2.7: Opciones del comando PAR

Por último tendremos que crear físicamente el fichero de configuración. Una vez que el diseño está completamente rutado tras ejecutar el proceso de colocación y rutado (PAR), configuramos el dispositivo mediante el fichero generado mediante **BITGEN** (fichero BIT). El fichero BIT contendrá la información de configuración del fichero NCD. El fichero NCD define la lógica interna y las interconexiones de la FPGA. Este fichero se descargará en las celdas de memoria de la FPGA de la forma que explicaremos más adelante.

La llamada al comando la haremos tal y como se muestra en el Script 8:

```
C:\Xilinx10.1\ISE\bin\nt\unwrapped\bitgen.exe -intstyle ise -w -g ActiveReconfig:Yes -g
Persist:yes -g DebugBitstream:No -g Binary:no -g CRC:Enable -g ConfigRate:4 -g
CclkPin:PullUp -g M0Pin:PullUp -g M1Pin:PullUp -g M2Pin:PullUp -g ProgPin:PullUp -g
DonePin:PullUp -g PowerdownPin:PullUp -g TckPin:PullUp -g TdiPin:PullUp -g
TdoPin:PullNone -g TmsPin:PullUp -g UnusedPin:PullDown -g UserID:0xFFFFFFFF -g
DCMShutdown:Disable -g DisableBandgap:No -g DCIUpdateMode:AsRequired -g
StartUpClk:JtagClk -g DONE_cycle:4 -g GTS_cycle:5 -g GWE_cycle:6 -g LCK_cycle:NoWait -g
Match_cycle:Auto -g Security:None -g DonePipe:No -g DriveDone:No -g Encrypt:No
%ioserie%/Circuito_FPGA.ncd
```

Script 8: BITGEN.bat

Como vemos tiene una gran cantidad de opciones. De todas ellas, destacaremos una que será importante a la hora de reconfigurar nuestro fichero .bit, y que explicaremos en la tabla 2.8.

-g ActiveReconfig:Yes	Se requiere para reconfiguración parcial, lo cual significa que el dispositivo se mantiene operativo mientras el nuevo bitstream parcial está siendo cargado. De esta forma sólo se cargaran en la FPGA las frames que difieran entre el archivo que se está cargando y el que está en la FPGA ya cargado.

Tabla 2.8: Opciones BITGEN

Una vez tengamos el fichero BIT generado, la idea es también poder cargarlo a través de nuestra propia aplicación. Esta funcionalidad nos la ofrece el programa **IMPACT**, una de las utilidades del Xilinx ISE. Trataremos pues de integrar este programa en nuestra aplicación en la medida de lo posible.

Sabemos que la aplicación IMPACT se basa en la llamada a comandos. No tendremos más que conocer qué comandos se usan cuando se carga cualquier fichero BIT.

Nota: los siguientes comandos están adaptados para una Virtex-II Pro cargada mediante Paralell Cable IV.

Lo que haremos será crear un fichero llamado **carga.txt** que tenga todos los comandos que queramos ejecutar con el IMPACT. Después ejecutaremos el IMPACT en modo batch de manera que ejecute los comandos que le pasamos por parámetro:
C:\Xilinx82\bin\nt\impact.exe –batch carga.txt

Donde carga.txt tendrá como contenido el que se muestra en el Script 9. Donde la ruta "D:\PFC\Nessy2.0\IOSerie\circuito_fpga.bit" será la ruta que introduzca el fichero bit que quiere cargar.

```
setMode -bs
setCable -port auto
Identify
identifyMPM
assignFile -p 3 -file "D:\PFC\Nessy2.0\IOSerie\circuito_fpga.bit"
Program -p 3
exit
```

Script 9: Comandos para carga mediante Impact

2.2.3. Inyección de errores

Hasta ahora hemos hablado de aspectos que eran fundamentales para el desarrollo de la aplicación y sin los cuales la funcionalidad se vería muy afectada. Al comienzo de esta lectura hablábamos de la posibilidad de insertar errores en la FPGA de forma que emulásemos así la posibilidad de que una partícula solar incidiera en la superficie de la FPGA modificando su contenido. En concreto, modificando la memoria de configuración.

Lo que tendremos que hacer entonces para inyectar un error es modificar el contenido de la memoria de configuración de la FPGA. Cada vez que reconfiguremos el dispositivo, querremos modificar un solo bit de la memoria de configuración.

La memoria de configuración de la Virtex-II Pro está organizada en frames verticales que tienen un bit de anchura y abarcan todo el alto de la FPGA. Estos frames son los segmentos más pequeños que se pueden direccionar en el espacio de memoria de la Virtex-II Pro. La longitud de una frame de la Virtex-II Pro depende del tamaño del dispositivo, tal y como se muestra en la Tabla 2.9.

Device	Frames	Frame Length (32-bit Words)	Configuration Bits (1)	Default Bitstream Size (2)
XC2VP2	884	46	1,301,248	1,305,376
XC2VP4	884	106	2,998,528	3,006,496
XC2VP7	1,320	106	4,477,440	4,485,408
XC2VP20	1,756	146	8,204,032	8,214,560
XC2VPX20	1,756	146	8,204,032	8,214,560
XC2VP30	1,756	206	11,575,552	11,589,920
XC2VP40	2,192	226	15,852,544	15,868,192
XC2VP50	2,628	226	19,005,696	19,021,344
XC2VP70	3,064	266	26,080,768	26,098,976
XC2VPX70	3,064	266	26,080,868	26,098,976
XC2VP100	3,500	306	34,272,000	34,292,768

Tabla 2.9: Número de frames, longitud, bits de configuración y tamaño de bitstream en Virtex-II Pro

En nuestro caso estamos trabajando con el modelo XC2VP30, por lo que tendremos 1756 frames y 206 bits por cada frame.

La tecnología permite diferentes formas de cambiar la configuración de los bloques y conexiones.

- **Reconfiguración total, en tiempo de compilación o estática**: Cada vez que se realiza una nueva configuración, toda la FPGA se actualiza. Se tiene que detener la ejecución de la FPGA, configurarlo todo y volver a ponerlo en marcha.

- **Reconfiguración parcial**: Se puede modificar una parte de de la configuración mientras la otra sigue realizando la computación de forma no interrumpida. En el fichero de configuración se especifican las direcciones (de las frames) para que se reconfigure. Con esto se puede cargar nuevas configuraciones en áreas de la FPGA sin necesidad de cambiar el contexto.

En este punto tenemos que coordinarnos con otro compañero que nos ofrecerá una aplicación capaz de modificar un solo bit de un fichero de configuración (fichero .BIT). Esta aplicación solo necesita conocer el frame que se desea modificar y el bit que se desea modificar dentro de ese frame.

La aplicación generará dos nuevos ficheros de configuración a partir del original. El primero habrá modificado un bit concreto de un frame concreto (ambos pasados como parámetro). Ese fichero de configuración se cargará modificando, por tanto, tan solo un frame de la FPGA, lo cual será equivalente a haber modificado tan solo un bit. El segundo fichero que se genera es un fichero de configuración para restaurar el frame modificado. De esta forma al cargarlo dejaremos el circuito como estaba originalmente.

El proceso que ejecutará la aplicación para la emulación de inyección de errores será el siguiente:

Se hará para un número determinado de iteraciones que introducirá el usuario. En primer lugar se cargará una entidad seguida de su fichero .BIT correspondiente. A continuación se pedirá al usuario un fichero para cargar el banco de pruebas. Tras cargarlo, y sabiendo que en la FPGA está cargado el fichero .BIT correcto (sin modificar), se generará una salida especial, llamada **salida Golden**, la cual será con la que comparemos en el resto de ejecuciones. Este paso sólo lo realizaremos una vez, al comienzo del proceso.

A continuación se ejecutarán todas las iteraciones. En cada una de ellas se ejecutará la aplicación de reconfiguración aplicada siempre al mismo fichero .BIT original (introducido por el usuario) en el que iremos cambiando los parámetros de frame y bit **de forma aleatoria**. Cada vez que se ejecute la aplicación se generarán dos ficheros de configuración: **fichero_modif.bit** y **fichero_modifRestore.bit**. Cargaremos el primer fichero en la FPGA, lo que es equivalente a que una partícula solar modificara el valor de un bit aleatorio. Una vez inyectado el error (cargado el fichero de configuración) volveremos a ejecutar el circuito en la FPGA, con las mismas entradas con las que habíamos generado la salida Golden.

Compararemos las nuevas salidas obtenidas con la salida Golden para ver si el error ha incidido en la ejecución del circuito.

Ahora tenemos que devolver a la FPGA a su “estado anterior”. Es decir, tenemos que restaurar el bit que acabamos de modificar. Para ello ordenaremos la carga del segundo fichero que se había generado: fichero_modifRestore.bit.

Con el circuito restaurado, ya podemos volver a ejecutar una iteración más del proceso.

2.2.4. FPGA

Fichero de restricciones de usuario (User Constraints File)

Un fichero de restricciones de usuario (UCF) es un fichero que contiene restricciones de localización y temporización. Este fichero es leído por el proceso NGDBUILD (detallado más adelante) durante el proceso de traducción. La finalidad de este fichero es asignar nuestras entradas y salidas del circuito top a localizaciones físicas de pines.

Las entradas de nuestro circuito principal van a ser:

Clk
Reset
Entrada_Serie

Las salidas serán:

Salida_serie

Además para visualizar los estados de la máquina de estados del módulo de entrada/salida de 32 bits, añadimos (de forma opcional) la visualización en los leds de dichos estados.

Nuestro fichero constraints.ucf tendrá el siguiente contenido:

```
NET "clk" LOC = "AJ15";          # Reloj del Sistema
NET "salida_serie" LOC = "AE7";   #Datos puerto serie RS232_TX_DATA
NET "reset" loc="AG5";           #Botón enter el del centro
NET "entrada_serie" LOC = "AJ8";  #Datos entrada puerto serie RS232_RX_DATA
NET "ledsEntrada<0>" LOC = "AC4";
NET "ledsEntrada<1>" LOC = "AC3";
NET "ledsSalida<0>" LOC = "AA6";
NET "ledsSalida<1>" LOC = "AA5";
```

De esta forma hemos conectado la entrada de reloj (**clk**) al reloj generado por la placa. La entrada **entrada_serie** al pin de Rx del puerto serie y la **salida_serie** al pin Tx del puerto serie. El reset al botón central de la FPGA (sólo resetea el módulo del puerto serie). Los leds de la derecha en la placa serán los correspondientes al estado de la Entrada Serie y los de la izquierda los correspondientes al estado de la Salida Serie.

La única forma que tenemos de interactuar con el módulo de entrada/salida es mediante los mecanismos que nos ofrece la placa. Por tanto, la forma en la que podemos resetear el módulo es pulsando el **botón central (ENTER)** de los botones de la placa. En principio no necesitaremos pulsar ese botón. Para ver que la entrada/salida está funcionando correctamente es suficiente con fijarse en si todos los leds están encendidos. Esto indicará que el módulo está sincronizado. En caso de desincronizarse puede volver al estado correcto pulsando reset. Los leds y el reset están marcados en la Figura 2.25.

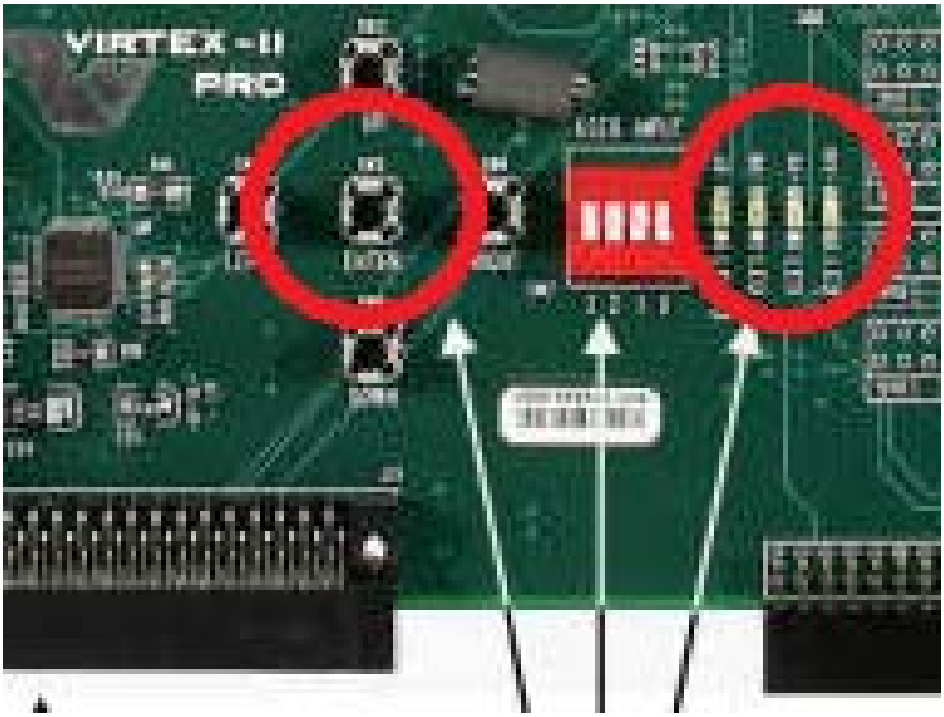


Figura 2.25: Reset y LEDs de estado de E/S

2.2.5. Datos de entrada/salida de un circuito

Una vez conocidos los pasos previos, la ejecución de un circuito es simple. Para ejecutar un circuito tendremos que ofrecer al mismo, una serie de entradas en forma de cadenas de 0's y 1's que coincidan en tamaño con el número de bits de entrada que tenga la entidad que queramos ejecutar.

Estas entradas pueden ofrecerse o bien mediante fichero, o bien escribiéndolas directamente en la aplicación (en el manual se explicará el lugar exacto). Si hemos comprendido el diseño y comportamiento de la entrada/salida, será fácil ver que cada una de las distintas entradas se corresponderá exactamente a un ciclo de ejecución.

Ejemplo: Imaginemos la siguiente entidad:

```
component CONTADOR
        Port(
                RESET: in STD_LOGIC;
                CLK: in STD_LOGIC;
                ENABLE: in STD_LOGIC;
                LOAD: in STD_LOGIC;
                DATA_LOAD: in STD_LOGIC_VECTOR(15 downto 0);
                SALIDA: out STD_LOGIC_VECTOR(15 downto 0)
        );
```

Vemos que tiene 20 bits de entrada y 15 de salida. De los 20 bits de entrada debemos descartar la entrada de reloj ya que esa no se transmitirá por el puerto serie sino que se conectará a la señal de fin de recepción (tal y como explicamos anteriormente). Por tanto

tenemos que tener en cuenta 19 bits de entrada y 15 de salida de datos. Pues bien, la entrada menos significativa se asignará a la primera señal de entrada declarada de la entidad y así sucesivamente. Lo mismo para las salidas.

La entrada (19 bits):

0000 0000 0000 0000 0 1 0

Se interpreta como:

RESET <= 0
ENABLE <= 1
LOAD <= 0
DATA_LOAD(0) <= 0
DATA_LOAD(1) <= 0
:
:
:
DATA_LOAD(15) <= 0
Si la salida (16 bits) fuera:
0001 0100 000 1011
Es sencillo intuir que corresponderá a:
SALIDA(0) <= 1
SALIDA(1) <= 1
SALIDA(2) <= 0
:
:
:
SALIDA(15) <= 0

Antes de cada ejecución y para asegurar que el circuito arranca en un estado correcto, enviaremos una señal de reset al circuito.

3 Manual de Usuario

En esta sección, buscamos dar una guía al usuario para facilitar el manejo de la aplicación y su correcto uso. Comenzaremos hablando de los requisitos del sistema necesarios para que se pueda ejecutar con éxito.

Los requisitos del sistema son:

Requisitos software

- Windows XP (la aplicación no se ejecuta para sistemas basados en 64 bits)
- La aplicación Xilinx ISE debe estar instalada
- La aplicación Impact debe estar instalada

Requisitos hardware

- Máquina con puerto serie (RS232) habilitado
- Máquina con puerto paralelo (Parallel Cable IV) habilitado
- Máquina con puerto PS2
- FPGA Virtex-II Pro (REV 04)

Además, Para ejecutar nuestra aplicación es necesaria la estructura de carpetas y archivos que se detalla en la Figura 3.1.

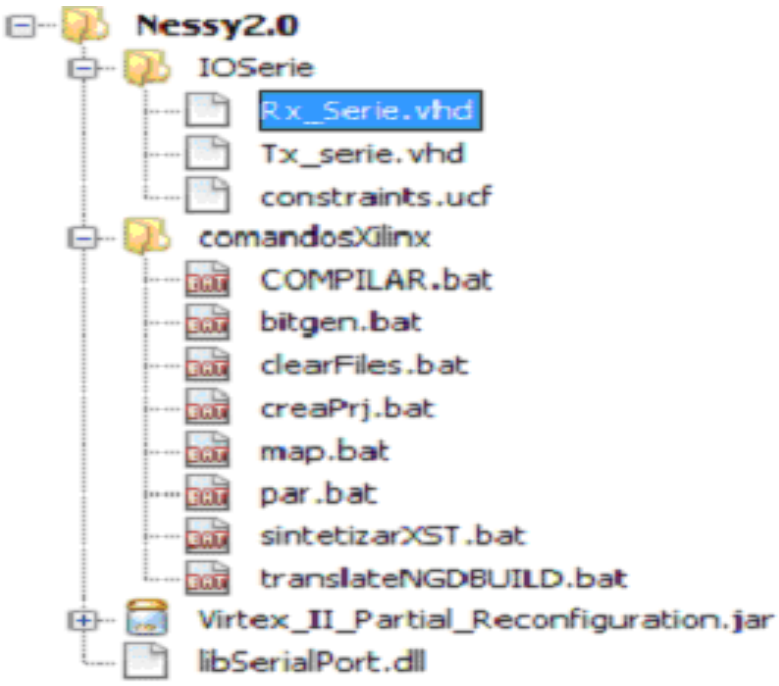


Figura 3.1: Directorios necesarios para la ejecución de la aplicación.

Ventana Principal.

Es la ventana principal de la aplicación (ver Figura 3.2). Desde ella se pueden realizar todas las acciones permitidas en cada momento por la aplicación. También se visualizan distintas salidas de nuestra ejecución en cada una de las pestañas disponibles.

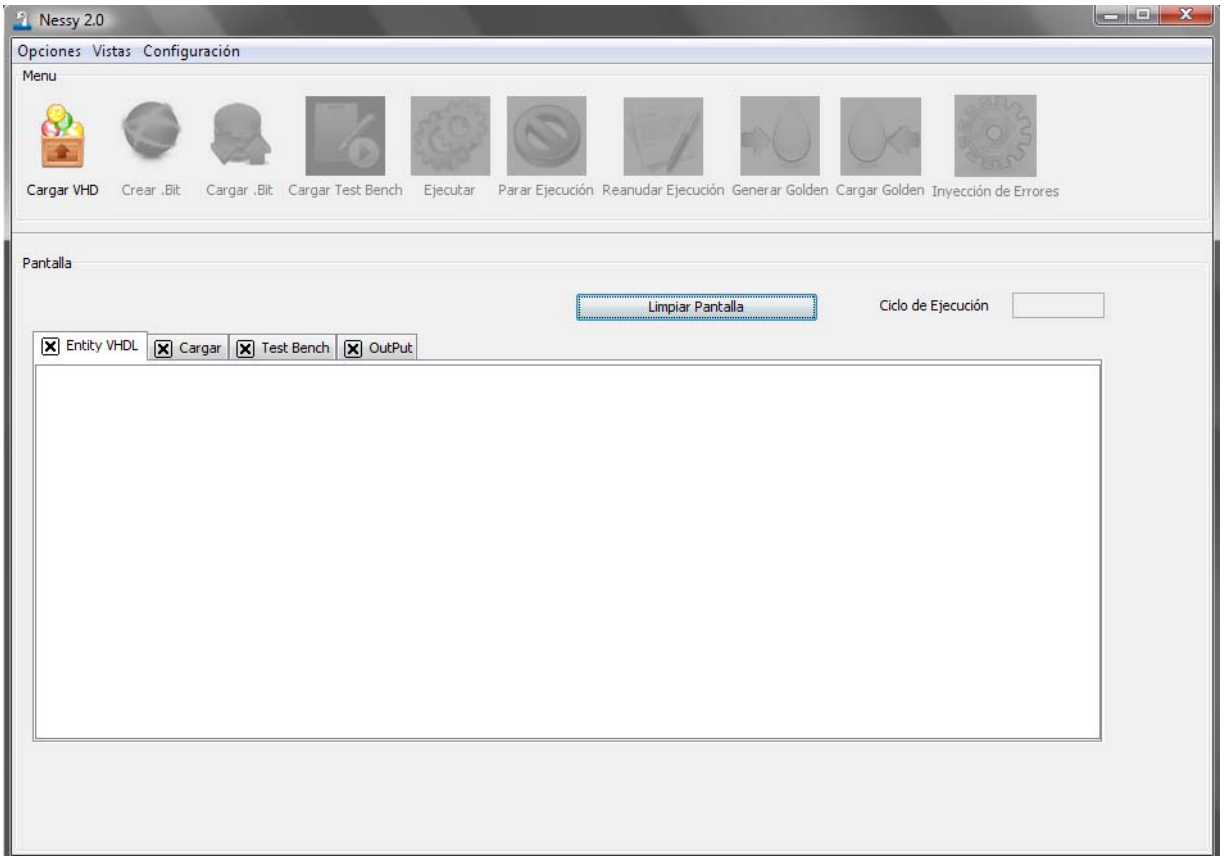


Figura 3.2: Ventana principal de la aplicación.

Barra de menús

Existen tres elementos desplegables en el menú de la parte superior que son el desplegable opciones, el desplegable vistas y el desplegable configuración.

Desplegable Opciones:

Se muestra un menú con que contiene las mismas opciones que hay en la botonera (ver Figura 3.3). Si un elemento de la barra de botones se encuentra deshabilitado, en este menú aparece de forma análoga.

El efecto de pulsar cualquier acción desde el menú de opciones o desde la barra de botones tiene el mismo comportamiento.

Las opciones que se permiten realizar en este desplegable son cargar VHD, crear .bit, cargar.bit, cargar Test Bench, ejecutar, parar ejecución, reanudar ejecución, generar golden, cargar golden e inyección de errores. Más tarde, cuando expliquemos en detalle la barra de botones, detallaremos el efecto de cada elemento de este desplegable.

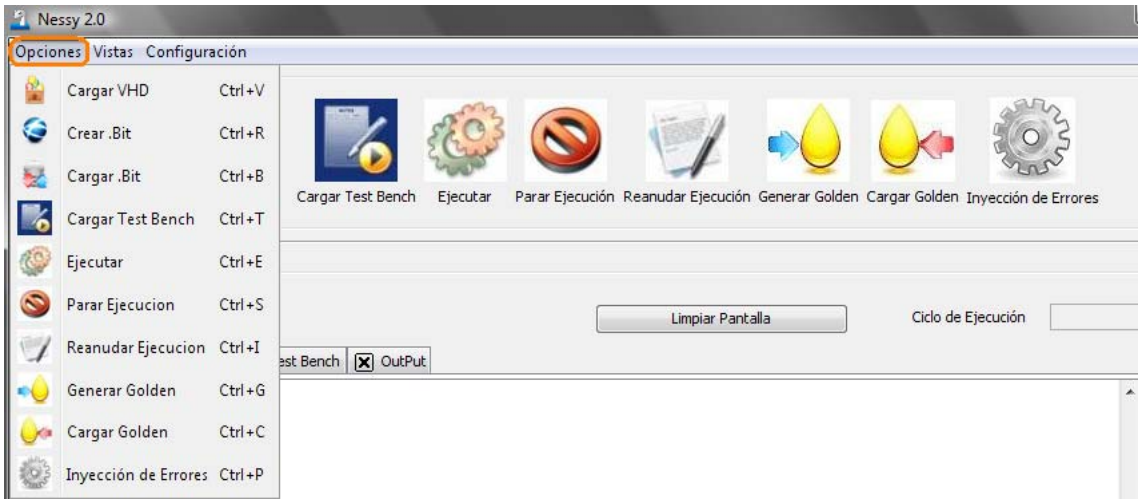


Figura 3.3: Desplegable Opciones

Desplegable Vistas:

Desde el desplegable vistas, ilustradas en la Figura 3.4 se puede seleccionar de entre las vistas disponibles cuál es la que deseamos visualizar. Si la vista se encontrara cerrada se abriría y se seleccionaría. Si por el contrario se encuentra abierta, simplemente se activa como la vista seleccionada, con su pestaña correspondiente.

Tenemos 4 vistas disponibles en la ventana principal:

1. **Entity VHDL:** En ella mostramos la entidad VHDL del archivo top con el que estamos trabajando.
2. **Cargar** : Mostramos la salida generada al cargar un archivo .bit en nuestra aplicación
3. **Test Bench :** Mostramos el banco de pruebas, de forma editable, para que el usuario pueda modificarlo. Se ejecutará este banco de pruebas si previamente seleccionamos la opción de Cargar Test Bench en Pantalla.
4. **Output :** Se muestra la salida generada por la última ejecución.

Ejemplo:

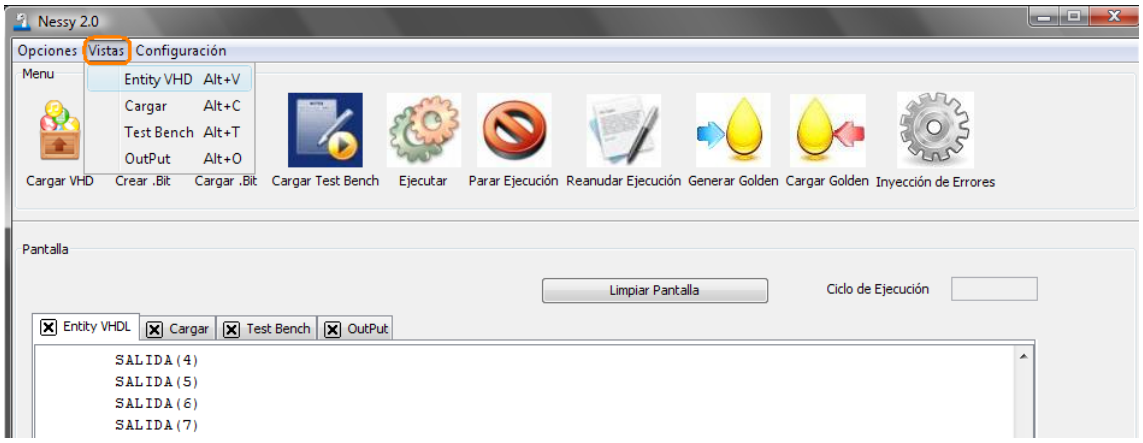


Figura 3.4: Desplegable Vistas.

Al pulsar en Output aparece la nueva vista, como se puede apreciar en la Figura 3.5. En este caso no había ninguna vista abierta.

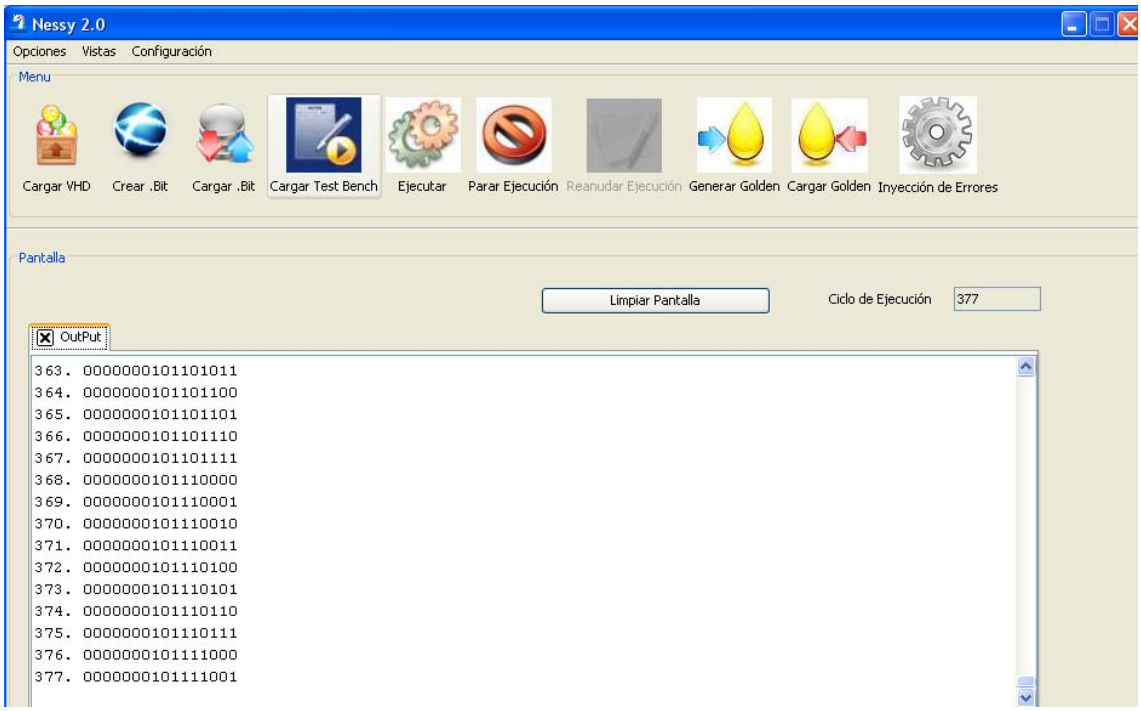


Figura 3.5: Vista Output.

El desplegable Configuración:

En este menú podremos establecer la carpeta donde se encuentra instalada la aplicación Xilinx. Es importante que al seleccionarla sea la carpeta raíz de la herramienta Xilinx porque si no, no funcionará correctamente la aplicación. Tenemos dos opciones de configuración, que son configurar Nessy y cargar fichero de ejecución, como se ve en la Figura 3.6.

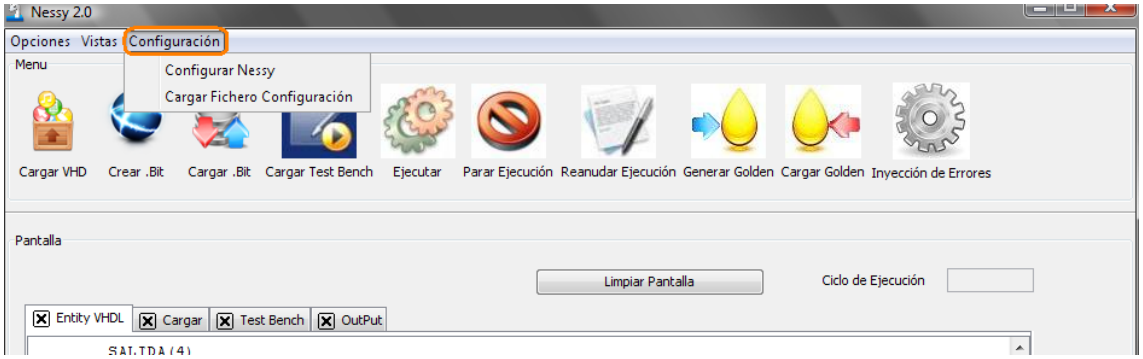


Figura 3.6: Desplegable configuración

Configurar Nessy :

Se abre una ventana emergente (ver Figura 3.7), en cual podremos seleccionar el fichero raíz donde encuentra la aplicación Xilinx instalada. Deberemos seleccionarla dando al botón de selección y pulsando sobre la carpeta elegida.

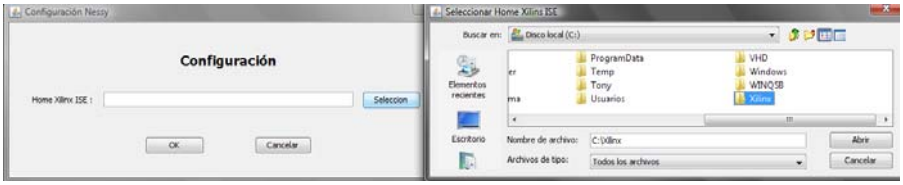


Figura 3.7: Configuración de Nessy

Si tuviéramos ya una dirección de Xilinx establecida, ésta aparecería escrita al abrir la ventana. Una vez elegida la nueva dirección bastaría con pulsar OK para guardarla.

Cargar Fichero de Ejecución:

Se abre una ventana para seleccionar el fichero .properties para establecer las propiedades de nuestra aplicación (ver Figura 3.8). Si se seleccionara un fichero que no tuviera el campo HomeXilinx daría el aviso que se muestra en la Figura 3.9.

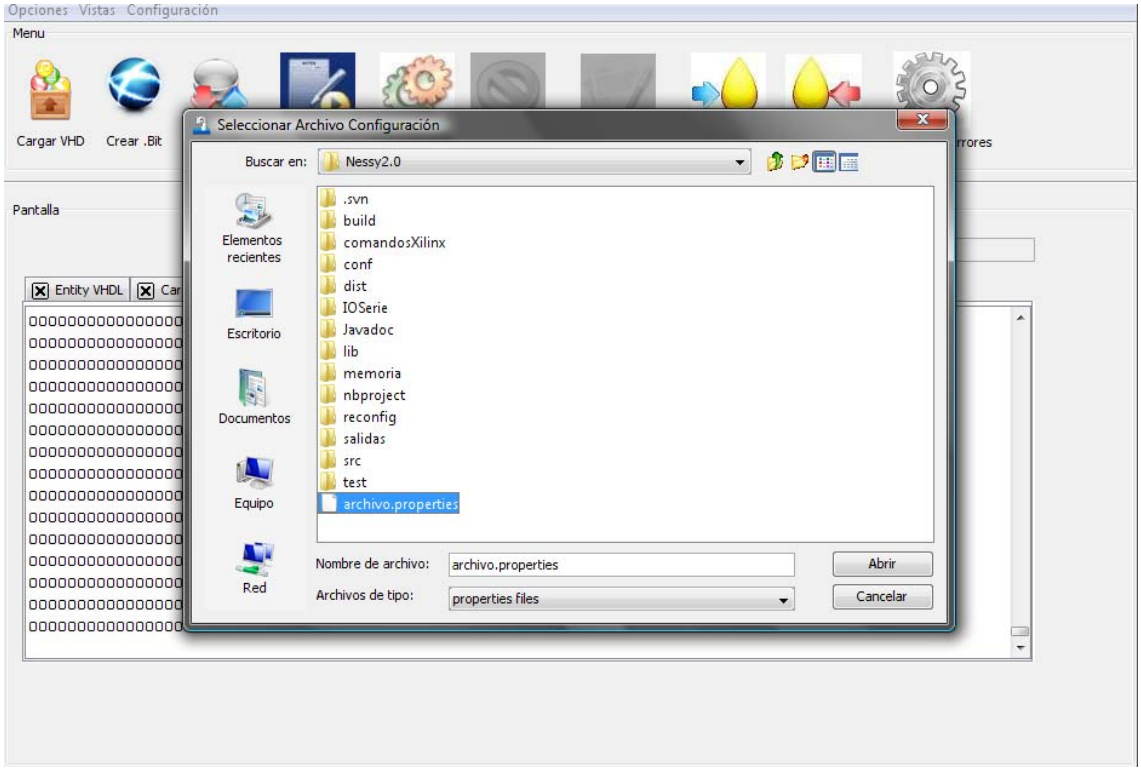


Figura 3.8: Cargar fichero de configuración

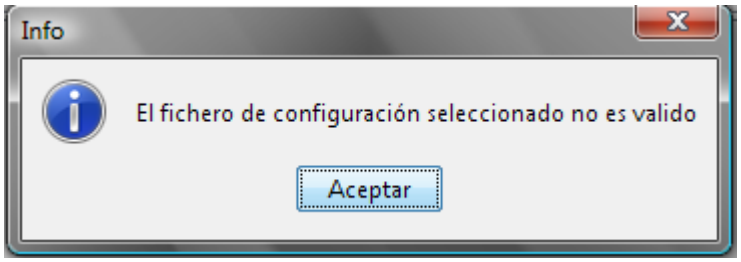


Figura 3.9: Error al cargar configuración desde archivo .properties

Barra de botones:

Desde ella, como se ve en la Figura 3.10, se controlan los procesos más importantes de la herramienta. Al comenzar la aplicación sólo estará activo el primer botón (cargar VHD), porque para trabajar correctamente es necesario tener bien definida la entidad con la que deseamos comunicarnos.

Figura 3.10: Botonera.

Cargar VHDL: 

Desde aquí cargamos los ficheros de tipo VHDL del proyecto con el que queramos trabajar. Al seleccionar esta opción se abrirá una ventana emergente, como se muestra en la Figura 3.11. Si nuestro circuito sólo contiene un archivo VHDL deberemos seleccionar Cargar un VHDL y este será el archivo Top.

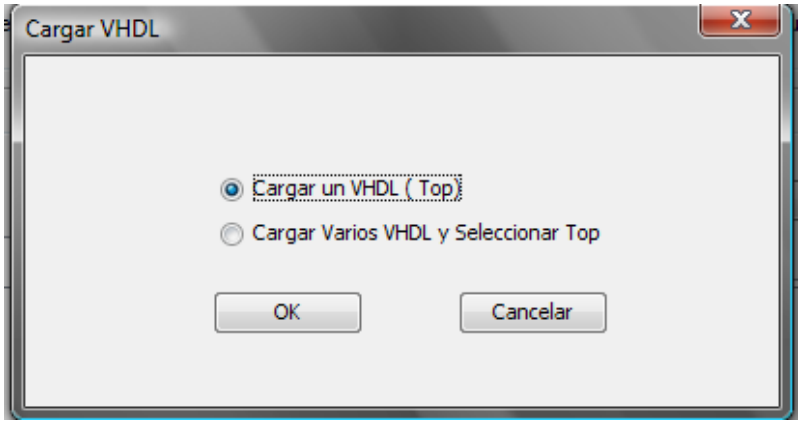


Figura 3.11: Opciones de Carga de VHDL

Si por el contrario, queremos seleccionar varios archivos utilizaremos la segunda opción. En ella podremos añadir todos los ficheros VHDL que deseemos, tanto de la misma carpeta como de carpetas distintas. Podemos añadir cuantas veces deseemos. Es importante tener seleccionado un archivo como Top, y solamente uno. En caso de no elegir ninguno, no podremos continuar ya que el proyecto sólo puede tener un fichero Top, tal y como como se ve en la Figura 3.12.

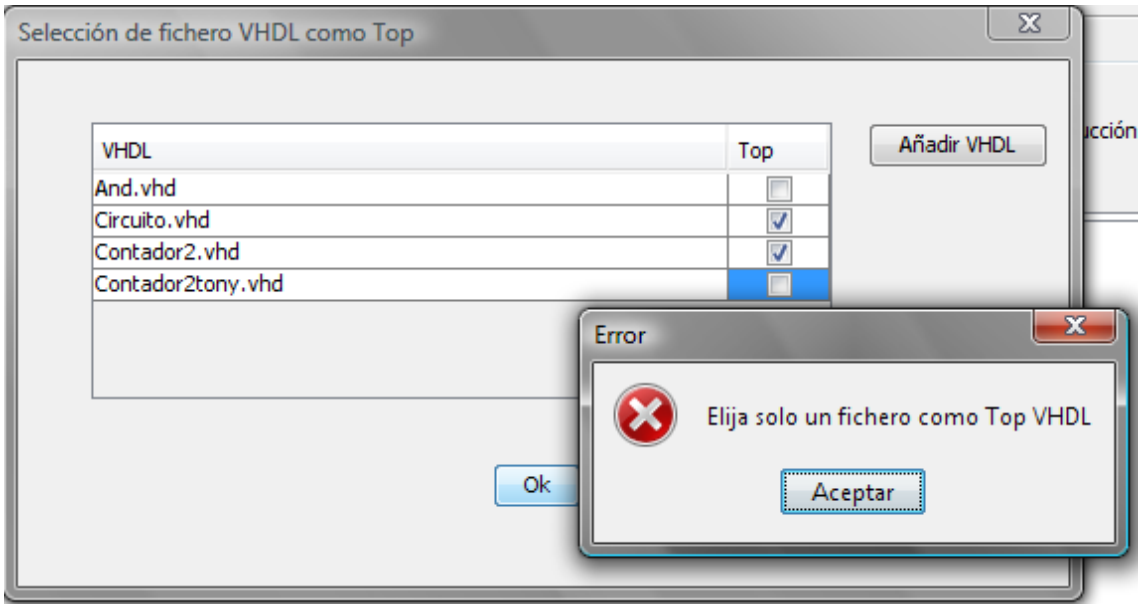


Figura 3.12: Error en la selección del arhivo TOP.

Una vez seleccionados los ficheros necesarios se cargará el archivo Top en la Vista Entity VHDL, para la cómoda visualización de la entidad principal.

El último archivo Top cargado correctamente se mostrará en la parte inferior de la ventana (ver Figura 3.13).

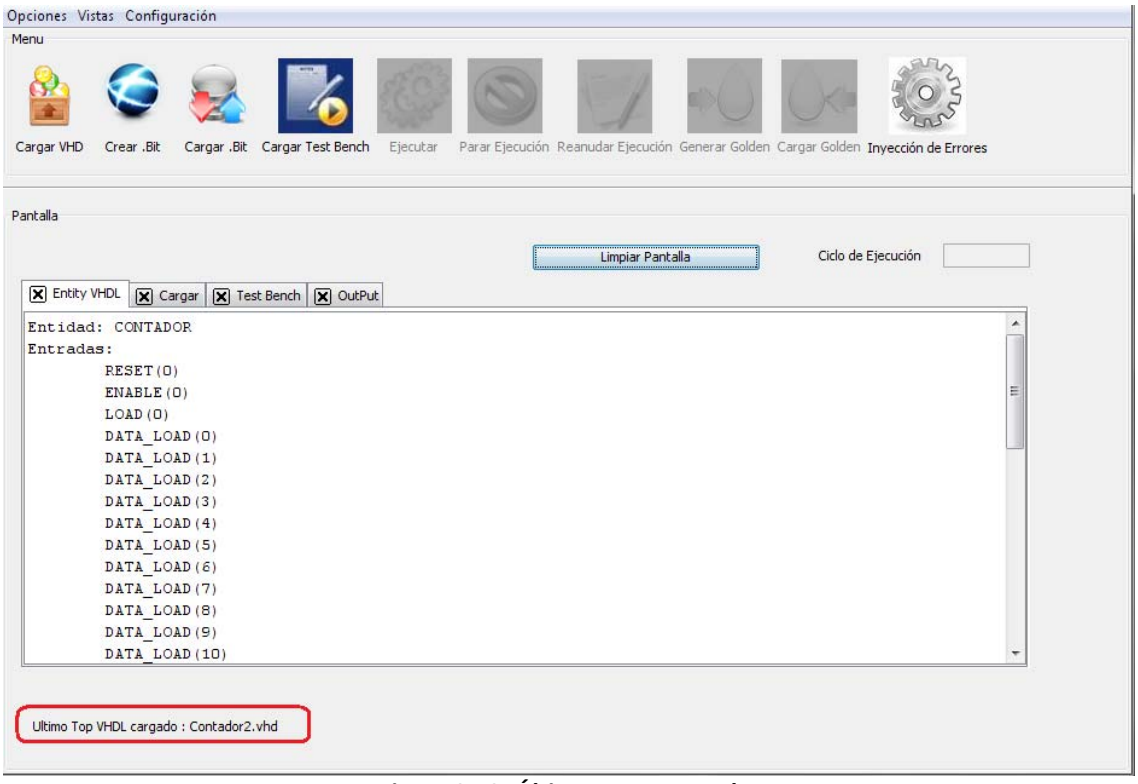


Figura 3.13: Último VHDL cargado.

Crear .Bit: 

Este botón aparece deshabilitado hasta que no hemos pulsado Cargar VHDL y hemos definido la entidad correctamente.

El primer paso al pulsar este botón es escoger dónde deseamos guardar el fichero .Bit y el nombre que tendrá el fichero, como se indica en la Figura 3.14. El fichero .Bit se generará a basándose en la entidad que hemos cargado y siempre que no aparezca ningún error a la hora de generarlo.

Este proceso es lento porque realiza una acción similar a la de crear el .bit desde Xilinx. Se abrirá una consola de comandos (Figura 3.15) que no se cerrará por si se desea consultar alguna etapa de la creación.

Internamente generamos un .bit creando un proyecto como haríamos en Xilinx, añandiéndole además los archivos Rx.vhd, Tx.vhd y un archivo que será TOP de todos. Con estos nuevos archivos proporcionamos una comunicación correcta con la FPGA siguiendo el protocolo del puerto serie. No importa el circuito que se declare como Top en nuestra herramienta siempre que no sobrepase las 32 entradas y 32 salidas.

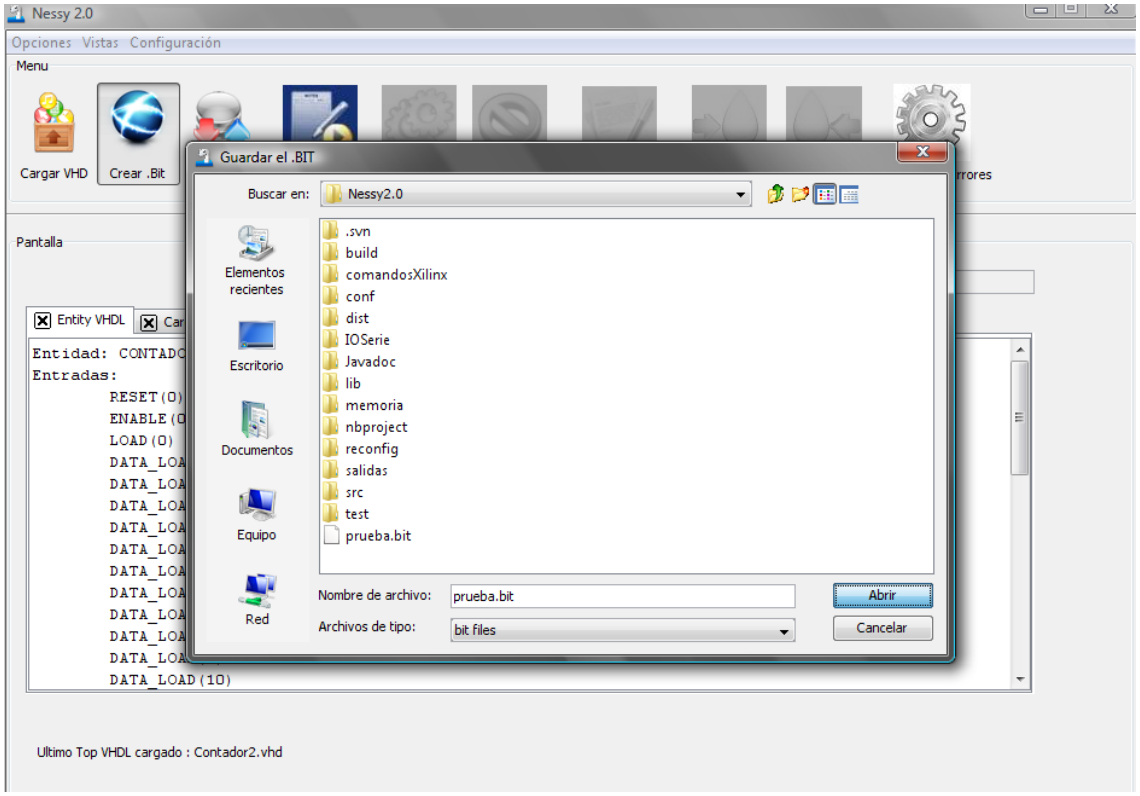


Figura 3.14: Guardar el archivo .bit que vamos a crear.

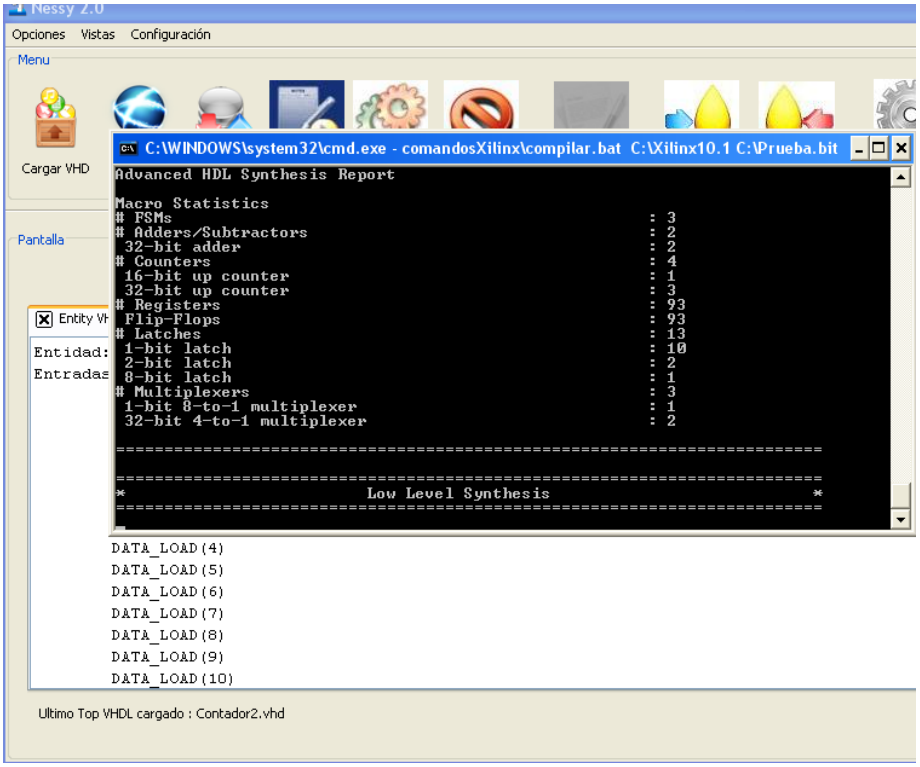


Figura 3.15: Creación del archivo .bit

Cargar .Bit: 

Desde esta opción podemos cargar el archivo .bit que elijamos. Para ello se abrirá una ventana en la que elegiremos el fichero (Figura 3.16). Esta opción está disponible desde que tenemos cargados correctamente los ficheros VHDL, porque podría ocurrir que ya estuviera generado el .BIT que queremos generar, y por tanto no tendríamos necesidad de generarlo de nuevo.

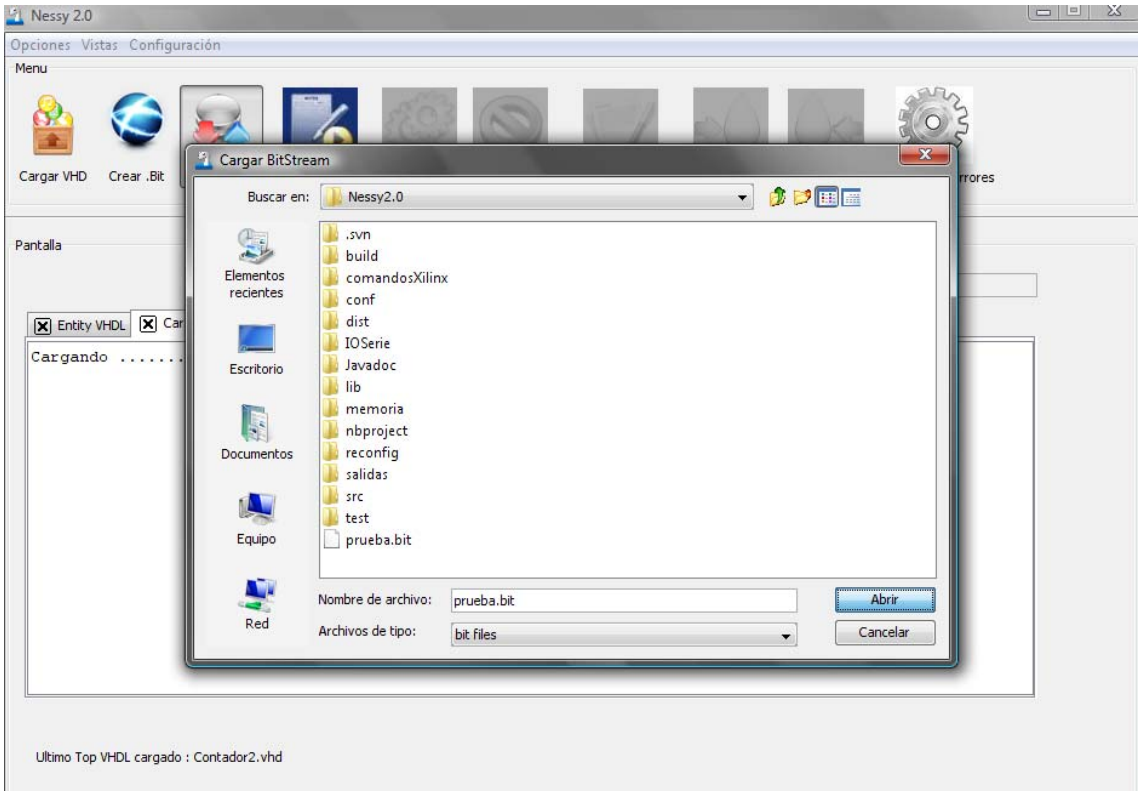


Figura 3.16: Carga de un archivo .bit

La salida que genere este proceso se mostrará por la vista Cargar y podremos ver si hemos tenido éxito en la carga del fichero en la FPGA, como en el caso de la Figura 3.17.

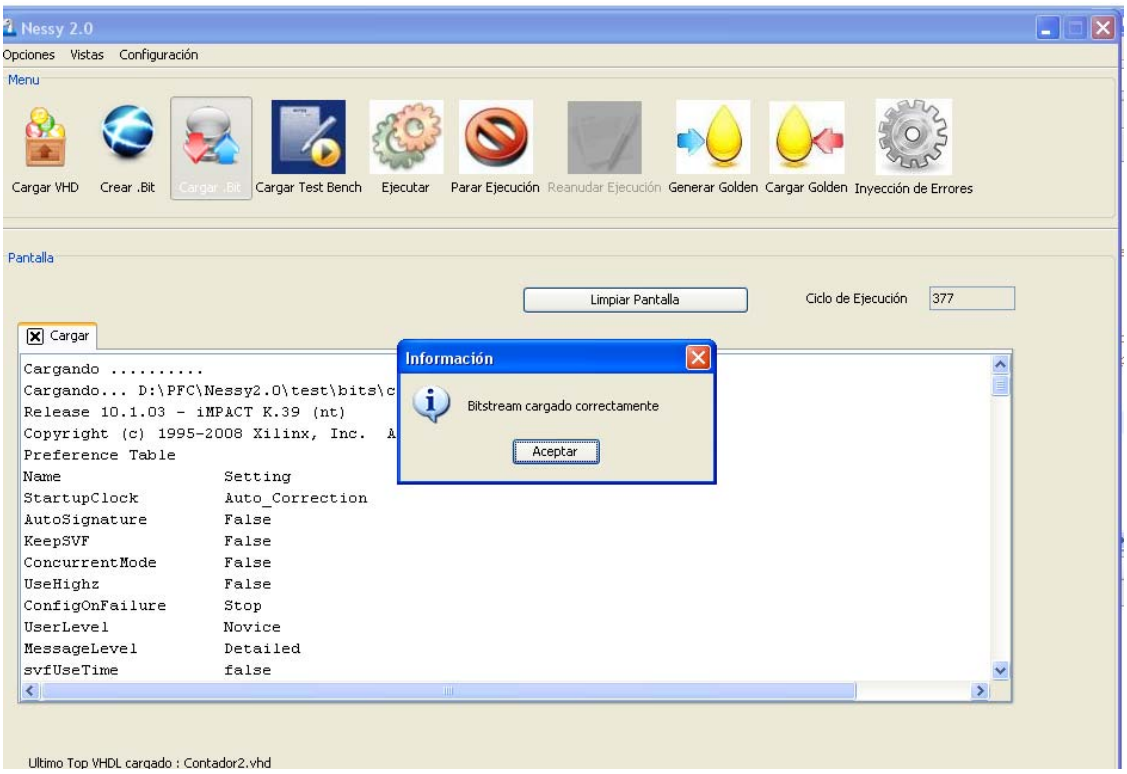


Figura 3.17: Carga correcta de un archivo .bit

En la Figura 3.18 podemos observar, que en la parte inferior de la ventana principal se encuentra el último archivo .bit que hemos cargado en la aplicación.

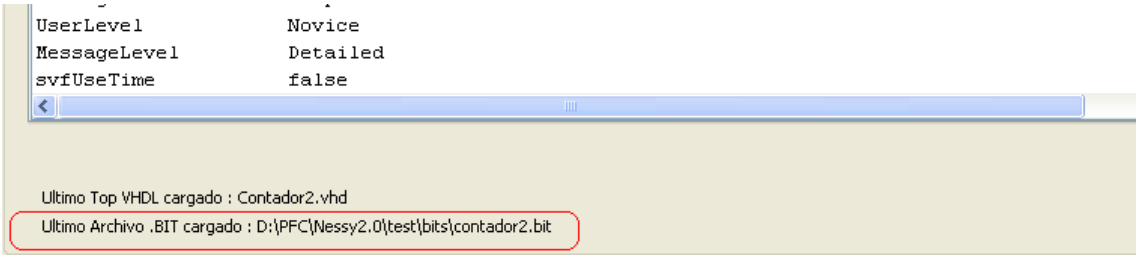


Figura 3.18: Detalle del último archivo .bit cargado.

Importante: el fichero .BIT que se cargue tiene que cumplir dos restricciones **para la correcta ejecución posterior**. La primera es que dicho fichero haya sido generado mediante el propio interfaz de usuario (con el botón Crear .bit). La segunda es que el fichero .BIT se corresponda a la entidad que se tiene cargada en ese momento, ya que si no, obtendremos resultados inesperados a la hora de ejecutar. Para ello, al generarlos, recomendamos nombrar a los ficheros .BIT con un nombre parecido al de la entidad a la cual se corresponden. Otro tipo de ficheros .Bit podrán ser cargados pero producirán resultados de ejecución inesperados.

Cargar Test Bench: 

En este caso, también esta opción de trabajo, esta habilitada desde que tenemos cargados correctamente los ficheros VHDL, porque puede que ya tengamos cargado en la FPGA el fichero .bit con el que deseamos trabajar.
Al pulsar el botón, se nos ofrecen dos opciones de trabajo, la carga del Test Bench en Pantalla, o desde Fichero y que comience la ejecución (ver Figura 3.19):

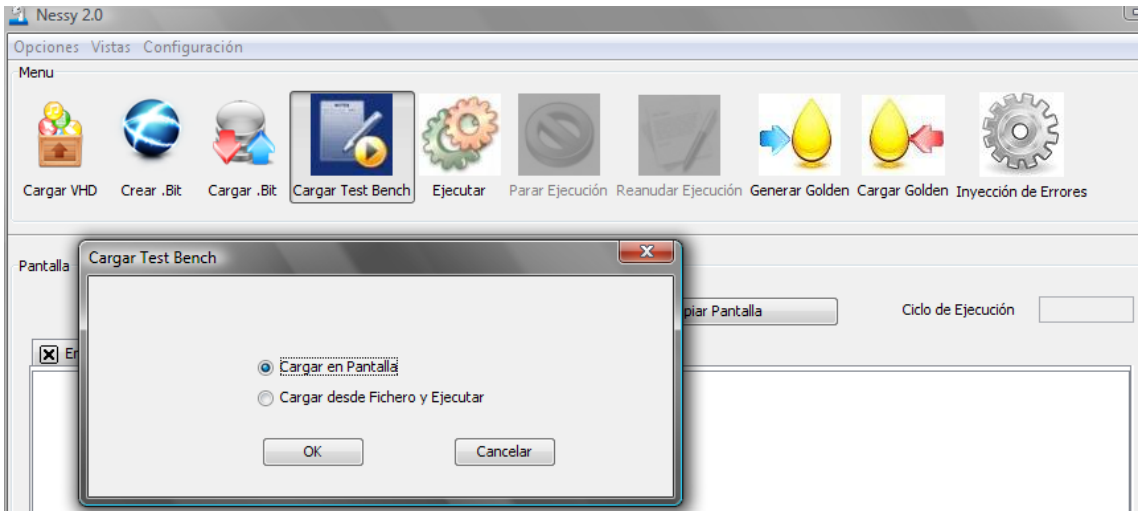


Figura 3.19: Opciones de la carga de un Test Bench.

Cargar en Pantalla: Al pulsar la opción Cargar en Pantalla, tendremos que seleccionar el archivo .txt con el repertorio de datos que deseamos enviar a la FPGA, y se nos cargará temporalmente en la vista Test Bench. Esta vista es editable por lo que se puede modificar los datos cuantas veces deseemos. Si tenemos éxito en la carga aparecerá un mensaje como el de la Figura 3.20.

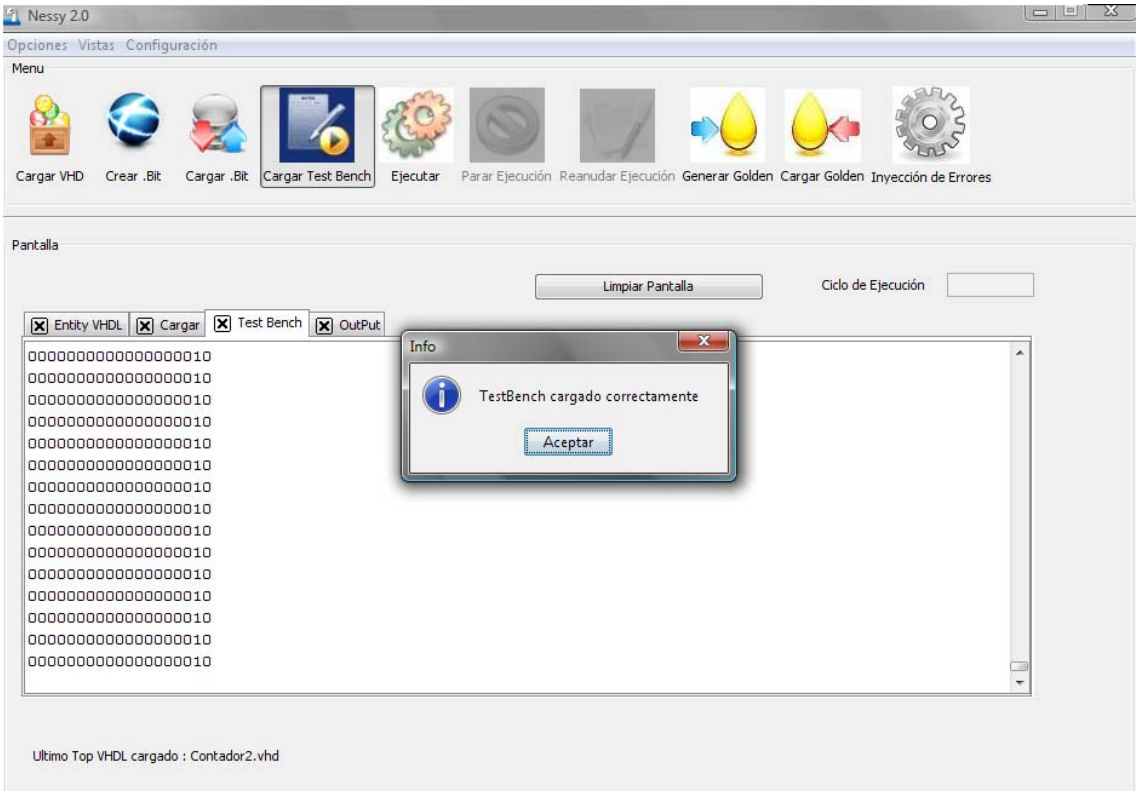


Figura 3.20: Cargar Test Bench en Pantalla con éxito.

Si el número de datos a enviar supera las 250.000 líneas, no se podrá ejecutar esta opción, por lo que deberemos seleccionar Cargar desde Fichero y Ejecutar, y nos aparecerá el error de la Figura 3.21.

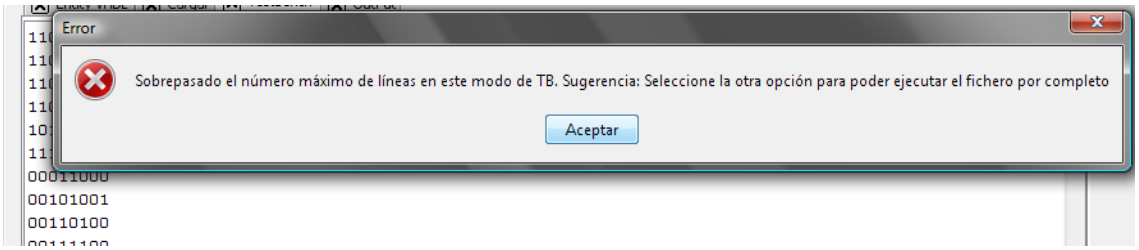


Figura 3.21: Error al cargar en Pantalla por sobrepasar el límite de instrucciones.

Cargar desde Fichero y Ejecutar: Con esta opción al seleccionar el archivo de banco de pruebas elegido, comenzará la ejecución siempre que el formato de los datos que contenga ese fichero esté en concordancia con la entidad Top que hayamos elegido, en caso contrario mostrará el error de la Figura 3.22.

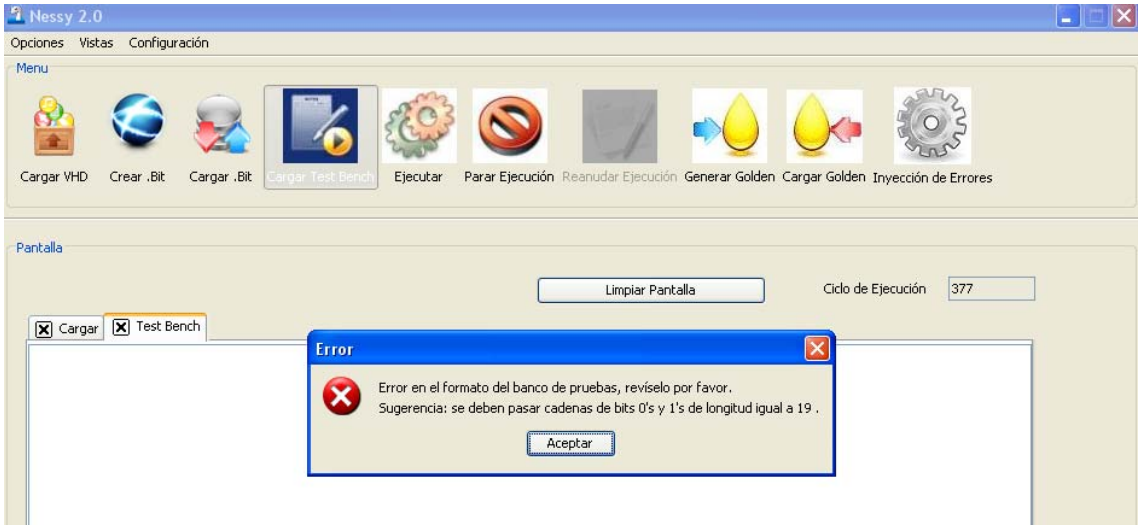


Figura 3.22: Error porque el formato del fichero no se corresponde con la entidad.

Para la generación correcta de un TestBench, ya sea cargado en pantalla o desde fichero, se ha de respetar el formato consistente en cadenas binarias por linea, cada cual se corresponde con un dato y tiene que tener la misma longitud que el número de entradas que tengamos definidas en el archivo VHDL que hemos cargado como TOP.
La última entrada declarada en la entidad TOP se corresponde con el bit más significativo, el siguiente bit se correspondería con la penúltima entrada declarada, y así sucesivamente hasta que llegamos a la primera entrada declarada que sería el bit menos significativo.
Ejemplo :

```
entity MiEntidadInventada is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          c : in  STD_LOGIC;
          z : out STD_LOGIC);
end MiEntidadInventada;
```

Aquí solo tenemos tres entradas, deberíamos generar cadenas de longitud 3.

Para a = 0, b = 1, c = 1 deberíamos enviar 110

Para a = 1, b = 1, c = 0 deberíamos enviar 011

Ejecutar: 

Esta opción esta habilitada, en el momento que hemos definido el banco del pruebas con el que queremos trabajar. Al pulsarlo se comienzan a enviar los datos que tengamos en la vista del Test Bench si hemos elegido cargar los datos en pantalla, o los cogerá del fichero que le hallamos definido, y se empezarán a visualizar en la vista Output como se aprecia en la Figura 24.

En el caso de elegir cargar los datos en pantalla si el formato no es el correcto nos aparece el mensaje de la Figura 3.23 y deberemos modificar el Test Bench que tenemos en la solapa y volver a ejecutar.

Es importante que el formato de los datos sean cadenas de números binarios, con una longitud igual a la suma de las entradas del archivo Top que hemos definido al cargar la Entidad.
La primera entrada que tenemos definida se corresponde con el bit menos significativo de la cadena y la última entrada con el bit más significativo de la entrada. Esto es importante tenerlo en cuenta cuando definamos los datos que queremos mandar.
Si alguna de las cadenas contiene un carácter que no se un número binario o la cadena tenga una longitud distintas del número de entradas, saltará un aviso y no se ejecutará nada (Figura 3.23).

La salida que se genera se compara con la última salida Golden que se guarda en un fichero de texto. Tanto si la salida de la ejecución actual coincide con la Golden como si no se mostrará un mensaje de información. La salida que se genera al pulsar este botón se guarda en un archivo de texto llamado Salida.txt dentro de la carpeta de salidas.

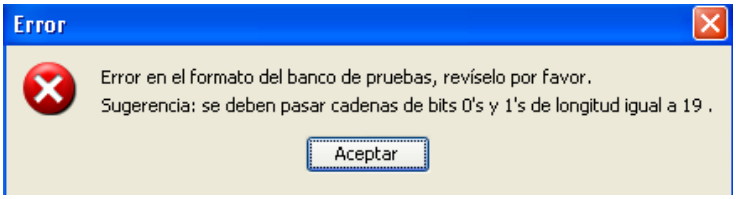


Figura 3.23: Error al ejecutar con Test Bench cargado en Pantalla.

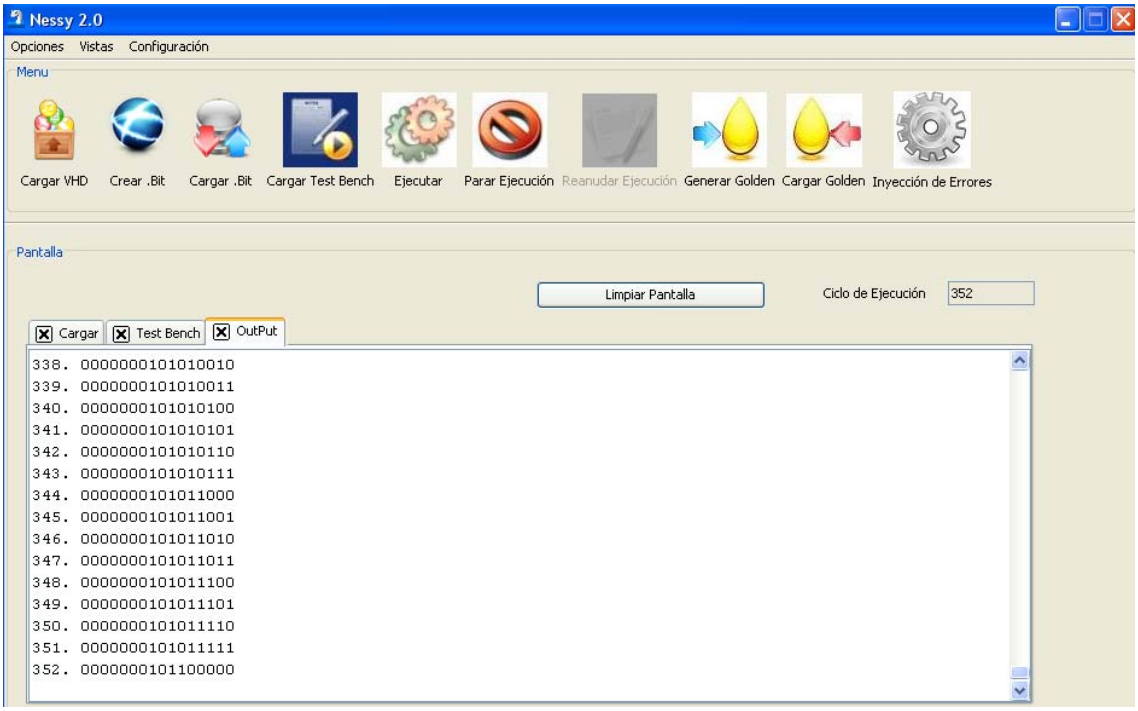


Figura 3.24: Ejecución correcta

Durante la ejecución se puede observar el número de datos enviados por el que va el proceso. Al pulsar ejecutar, y mientras dure la ejecución se habilita el botón Parar Ejecución para detener el proceso.

Parar Ejecución: 

Este botón se habilita mientras dura la ejecución de un banco de pruebas. Se detiene el envío de datos y se deja de recibir. Podemos observar en el dato que nos hemos quedado.
El parar la ejecución no afecta en absoluto a la salida que se produzca, al menos que se modifique el archivo .bit cargado en la placa o el fichero con los datos que se están enviando a la FPGA.

Si la salida actual que se está generando no coincide con la salida Golden al pulsar este botón nos alertaría ya de este hecho (como ocurre en la Figura 3.25), indicándonos también el dato que causó la diferencia.

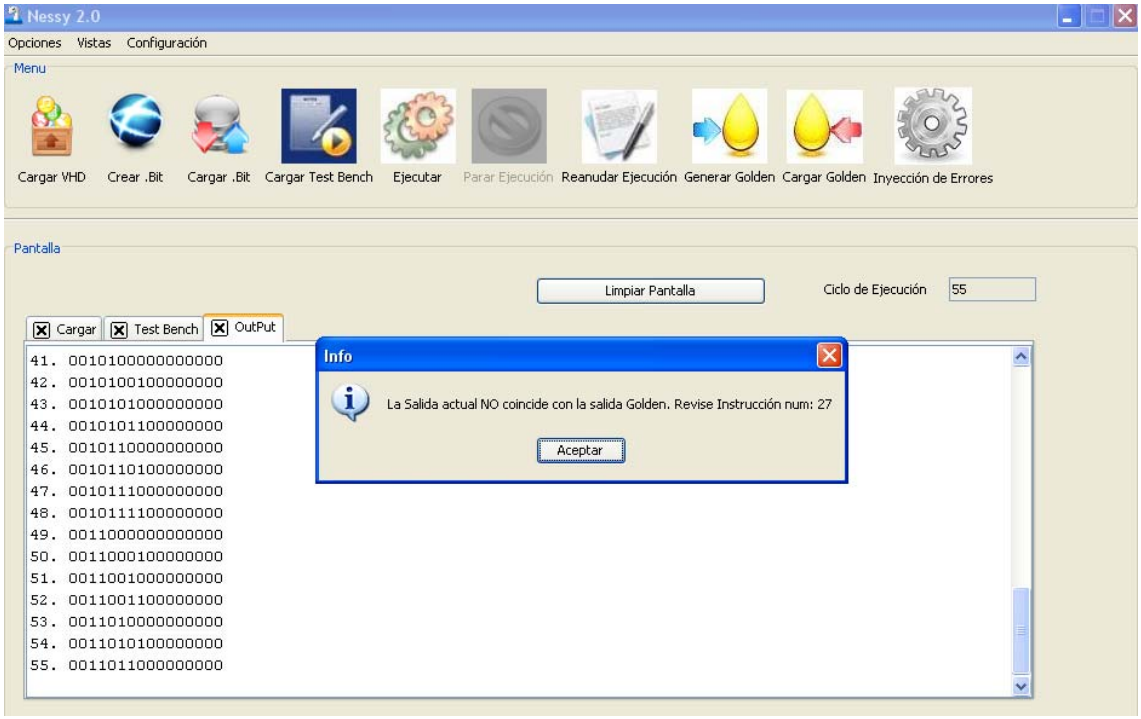


Figura 3.25: Parar la ejecución, con distinta salida.

Al pulsar este botón se deshabilita el propio botón y habilita el botón de reanudar ejecución. Vuelve estar habilitado al volver al pulsar ejecución, o al pulsar reanudar ejecución.

Reanudar Ejecución: 

Este botón esta habilitado cuando hemos parado la ejecución, y permite continuar la ejecución en el punto donde se estaba antes de pulsar parar ejecución.

Este botón solo esta habilitado mientras dura la ejecución y después de haber parado la ejecución. El efecto al pulsarlo es que habilita parar ejecución y prosigue el envío de datos a la FPGA, como ocurre, en la Figura 3.26.

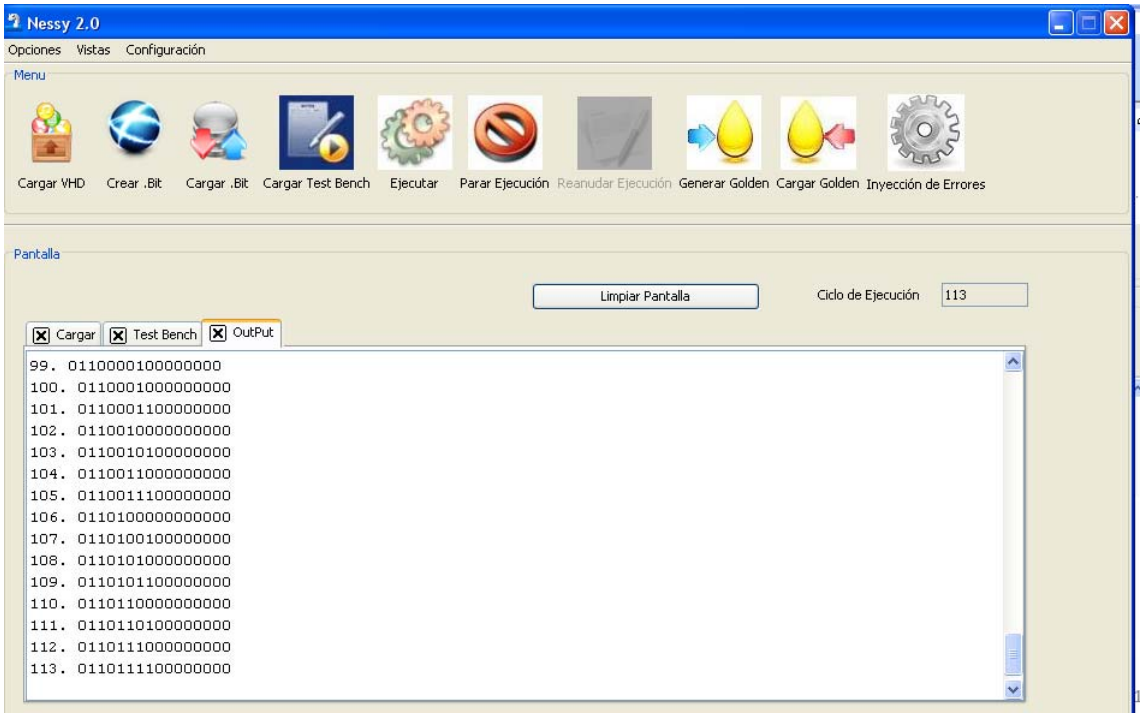


Figura 3.26: Reanudar Ejecución.

Generar Golden: 

Este botón queda habilitado después de que tengamos definido el conjunto de datos del banco de pruebas que deseamos enviar a la placa. Su funcionalidad es la de una ejecución pero guarda la salida en Golden.txt y será el fichero de referencia para comparar el resto de ejecuciones y ver si ha habido algún error entre las distintas ejecuciones.

Cargar Golden: 

Se encuentra habilitado al definir el cargar el banco de pruebas. Al pulsar este botón se abrirá una ventana para elegir el fichero que a partir de ese momento será nuestra salida Golden con la que comparemos todas las ejecuciones.

Inyección de Errores: 

Este botón se encuentra habilitado al tener definida la entidad con la que vamos a trabajar. La funcionalidad es ir modificando bit a bit el archivo .bit que elijamos al principio para ir viendo como afecta la modificación de ese bit en la salida.

Al pulsar el botón, lo primero será indicar el número de iteraciones que deseamos que haga (Ver Figura 3.27).

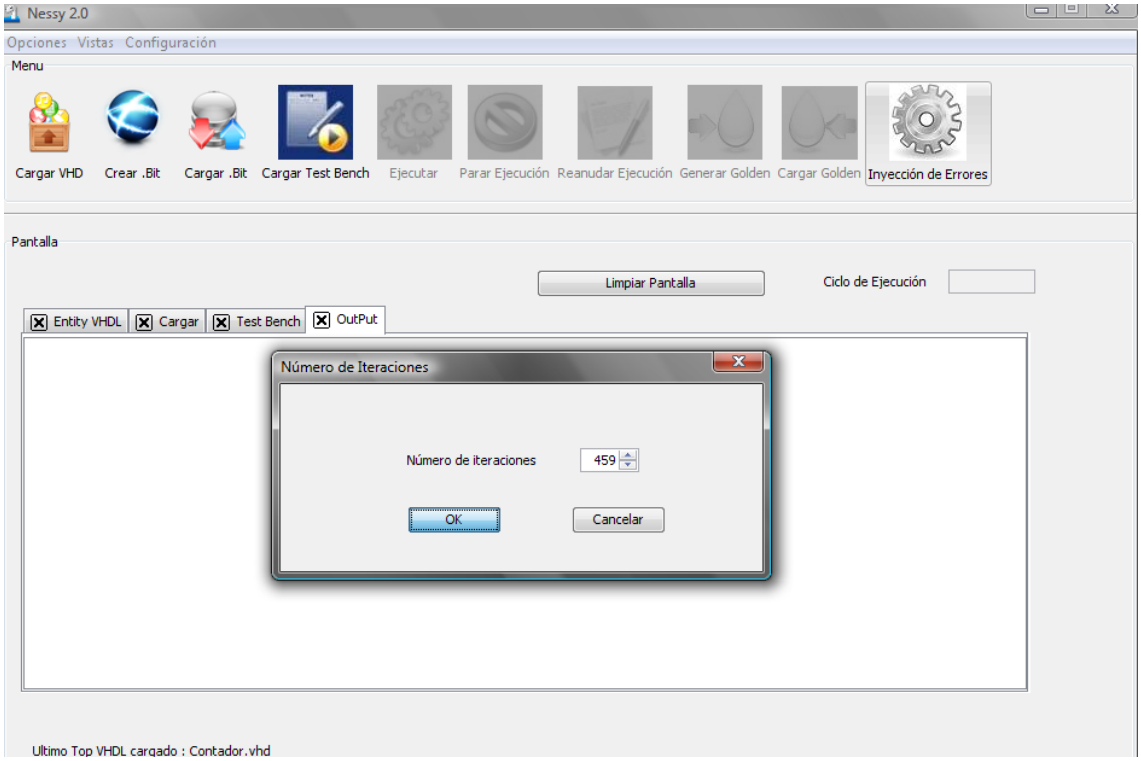


Figura 3.27: Selección del número de iteraciones.

Lo siguiente que tenemos que elegir es el archivo .bit (Figura 3.28) que vamos a tomar como referencia y sobre el que haremos las modificaciones. Posteriormente tendremos que seleccionar el fichero de banco de pruebas (Figura 3.29) con el repertorio conjunto de datos que se van a enviar.

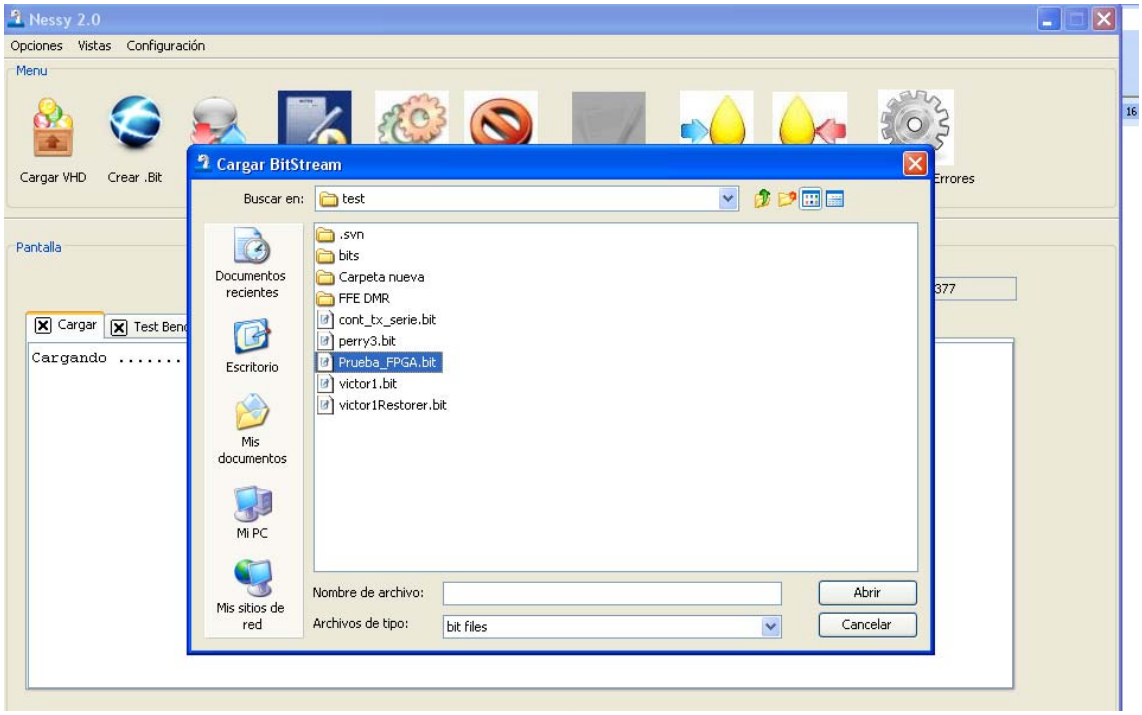


Figura 3.28: Inyección de Errores. Cargar .bit de trabajo.

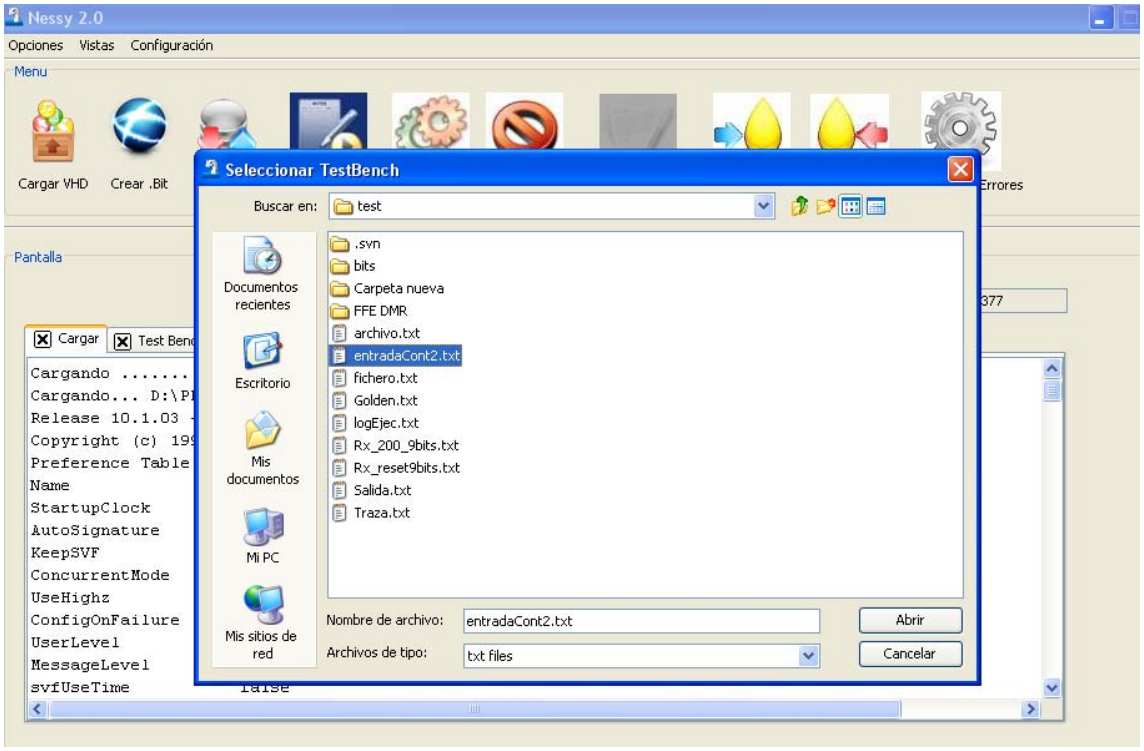


Figura 3.29: Inyección de Errores. Cargar Test Bench.

En un primer momento se hará una ejecución sin modificar el archivo para generar la salida Golden, y así, cuando vayamos modificando el archivo cargado en la FPGA ver como ha afectado a la ejecución.

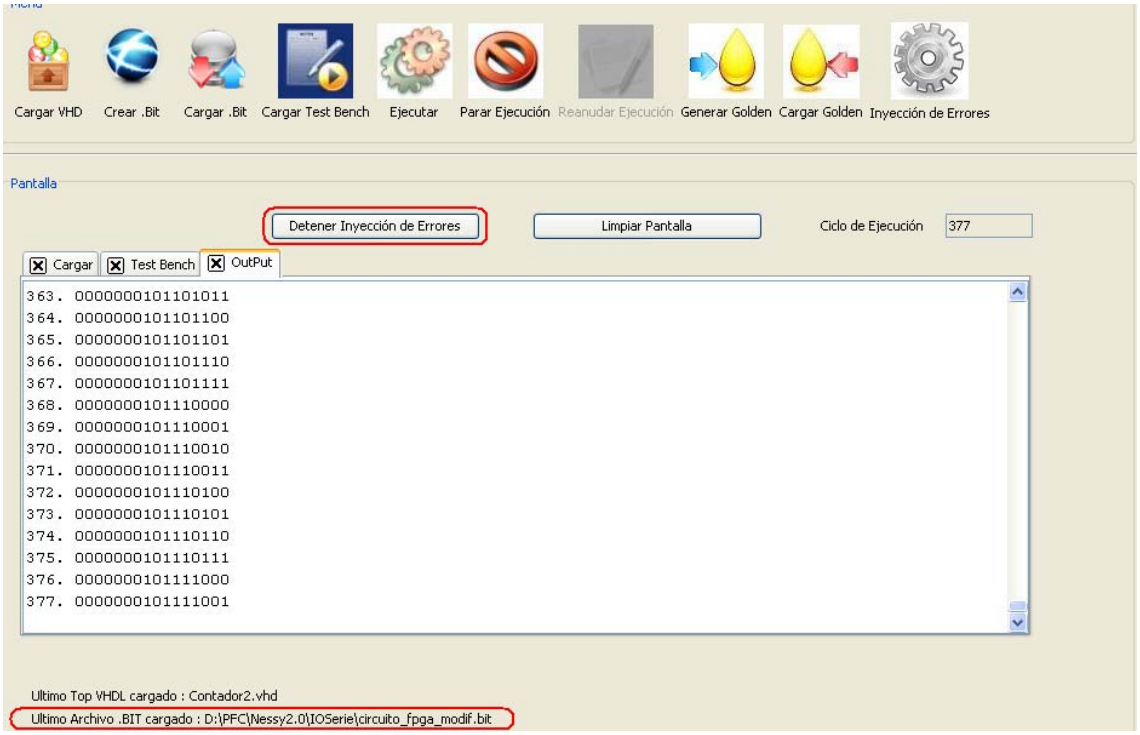


Figura 3.30: Inyección de Errores.

La salida, que genera este proceso se guarda en un fichero logEjec.txt, donde se puede revisar en que punto la ejecución fue distinta porque el bit modificado afecto a la salida generada.

Es importante tener en cuenta que como es proceso largo no se muestran ventanas emergentes con el resultado de la comparación entre la salida Golden y la última ejecución que se ha producido por comodidad para el usuario (Figura 3.30).

Al pulsar este botón se habilita un botón (Detener Inyección de Errores) debajo de la barra de botones que permite abortar la inyección de errores, por si el usuario decide terminar el proceso.

Toda la salida y diferencias generadas por la inyección de errores se guardan en un fichero de texto, para saber como ha afectado la modificación del mapa de bits a la salida. Dicho fichero se guarda en *salidas/logEjec.txt.* Un ejemplo de dicho fichero se muestra en la Figura 3.31.

```
0000000101101111
0000000101110000
0000000101110001
0000000101110010
0000000101110011
0000000101110100
0000000101110101
0000000101110110
0000000101110111
0000000101111000
0000000101111001
La Salida actual coincide con la Golden

Cargando BIT: D:\PFC\Nessy2.0\IOSerie\circuito_fpga_modifRestorer.bit


Modificando FRAME: 18601 BIT: 4
Ejecutando: cmd.exe /K java -jar Virtex_II_Partial_Reconfiguration.jar -i D:\PFC\Nessy2.0\test\Prueba_FPGA.bit -o D:\PFC\Nessy2.0\
Cargando BIT: D:\PFC\Nessy2.0\IOSerie\circuito_fpga_modif.bit
Ejecutando...
0000000000000001
0000000000000010
0000000000000011
0000000000000100
0000000000000101
```

Figura 3.31 Ejemplo de fichero log de inyección de errores

Más detalles a tener en cuenta:

En la primera ejecución de la aplicación se abrirá la ventana para definir la ruta HomeXilinx y hasta que no la definamos no podremos acceder a la ventana principal.
El botón de Limpiar Pantalla situado debajo de la barra de botones sirve para limpiar el contenido de la vista que actual.
El último .bit y el último archivo Top utilizados durante el uso de la aplicación se detallan en la parte inferior de la ventana principal, como se puede ver en la Figura 3.32.

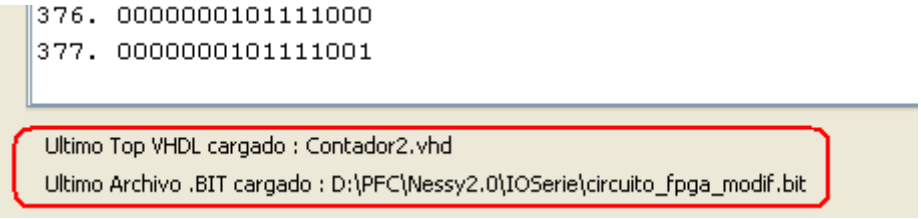


Figura 3.32: Detalle del último .bit y VHDL (top) cargardos.

Log de la aplicación

Durante la ejecución de la aplicación, se irá generando un fichero log en el que se indique todas las acciones que el usuario ha ido realizando, así como posibles errores que hayan podido producirse tanto en la ejecución actual como en ejecuciones anteriores. De esta manera si se ha experimentado un resultado anómalo o inesperado, se podrá consultar este fichero para ver qué ha podido suceder. Dicho archivo se crea en la ruta -> ***C:\aplicacion.log***.

Teclas Rápidas:

Para facilitar la interfaz con el usuario, existe la siguiente combinación de teclas rápidas:

Opciones:

- Cargar VHD → Control + V
- Crear .Bit → Control + R
- Cargar .Bit → Control + B
- Cargar Test Bench → Control + T
- Ejecutar → Control + E
- Parar Ejecución → Control + S
- Reanudar Ejecución → Control + I
- Generar Golden → Control + G
- Cargar Golden → Control + C
- Inyección de Errores → Control + P

Vistas:

- Entity VHD → Alt + V
- Cargar → Alt + C
- Test Bench → Alt + T
- Output → Alt + O

4 Código Fuente

4.1. Ficheros VHDL

TX_SERIE

```
----------------------------------------------------------------------------
-- Company:       UCM Facultad de Informática
-- Engineer:      Carlos Sánchez-Vellisco Sánchez
--                Antonio José García Martínez
--                David Fernández 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. Dividirá la
--                  frecuencia de reloj tantas veces como indique
--                  gFrecClk.
--                  ->Control: Una máquina de estados finitos.
--                  ->Carga_desplaz: Un registro de carga en paralelo.
--                  ->Selección: Un multiplexor que selecciona la señal
--                  de salida según 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;

----------------------------------------------------------------------------
--    PUERTO      NºBits            INFO
--
--    RstN           1  Entrada     Señal de reset asíncrono
--                                  (activo a '0')
--    Clk            1  Entrada     Señal de reloj de la placa,
--                                  en principio de 100MHz, pero
--                                  configurable por gFrecClk
--    Transmite      1  Entrada     Señal del sistema que ordena al
--                                  módulo la transmisión del dato que
--                                  se encuentra en DatoTxIn.
--    DatoTxIn       8  Entrada     El dato a enviar. Se proporciona de
--                                  manera simultánea cuando
--                                  Transmite = '1'
--    Transmitiendo  1  Salida      Señal del sistema que indica que en
--                                  ese instante se está transmitiendo
--                                  un dato.
--    DatoSerieOut   1  Salida      Trama de datos que se envía al PC y
--                                  que sigue el protocolo RS232
--
```

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


entity Tx_serie is
 generic (
                        gFrecClk : integer := 100000000; --100MHz
                        gBaud : integer := 9600 --9600bps
                    );
    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;

      -- Señales 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
-- señal con frecuencia de 9600 Hz (ClkBaud). Este reloj tendrá por
-- tanto un
-- periodo de 104,167 µs, y estará a ‘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 señales 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 máquina de estados
--eInit:     Es el estado inicial. El sistema está en reposo esperando
--           la orden de transmitir. Cuando la señal Transmite se ponga
--           a '1', se pasará a enviar el bit de inicio, pasando para
--           ello al siguiente estado (eBitInit). En ese momento dará
--           la orden de cargar el dato (DatoTxIn) en el registro de
--           desplazamiento.También 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 está enviando el bit de inicio. Se
--           saldrá de este estado al recibir un pulso de ClkBaud, que
--           nos dirá que 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 número de
--           bits que se han enviado. Cuando se hayan enviado los 8
--           bits ,es decir, cuando hayan llegado 8 pulsos de Clkbaud,
--           se activará la señal FinDsplza8bits que hará que cambiemos
--           al siguiente estado (eBitFin).
-- eBitFin:  Este estado envía 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 función del 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;
```

RX_SERIE

```
----------------------------------------------------------------------
-- Company:             UCM Facultad de Informática
-- Engineer:            Carlos Sánchez-Vellisco Sánchez
--                      Antonio José García Martínez
--                      David Fernández 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 diseño en 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 más
--                      significativo del registro y se hacen desplazar
--                      los bits hacia la derecha, en el último
--                      desplazamiento el bit 0 recibido estará en el
--                      bit menos significativo del registro, estando
--                      así todos los bits ordenados. La carga y el
--                      desplazamiento se realizan bajo la orden de la
--                      señal Desplaza que sale del bloque de control.

--                      ->Registro de entrada: Como la entrada de la
--                      comunicación serie RxDatoserie es asíncrona, se
--                      almacena en un registro RxDatoReg 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;

----------------------------------------------------------------------
-- PUERTO         NºBits          INFO
--
-- RstN           1     Entrada   Señal de reset asíncrono (activo a
--                                ‘0’)
-- Clk            1     Entrada   Señal de reloj de la placa,
--                                enprincipio 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 sólo es válido desde que
--                                AvisoRx vale ‘1’ y mientras
--                                recibiendo sea ‘0’.
-- AvisoRx        1     Salida    Aviso de que se ha recibido un
--                                nuevo dato y que está disponible en
--                                DatoRxOut. El aviso se dará
--                                poniendo la señal 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 válido.
----------------------------------------------------------------------
entity Rx_Serie is
       generic (
                        gFrecClk : integer :=100000000; --100MHz
                        gBaud : integer :=9600--3 --9600bps

                        );
    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 ;

      -- Señales 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 señal con frecuencia de 9600 Hz (ClkBaud). Este
-- reloj tendrá por tanto un periodo de 104,167 µs, y estará a '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 señales 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 máquina de estados
--eInit:     Es el estado inicial. El sistema está en reposo esperando
--           la orden de transmitir. Cuando la señal Transmite se ponga
--           a '1', se pasará a enviar el bit de inicio, pasando para
--           ello al siguiente estado (eBitInit). En ese momento dará
--           la orden de cargar el dato (DatoTxIn) en el registro de
--           desplazamiento.También 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 está enviando el bit de inicio. Se
--           saldrá de este estado al recibir un pulso de ClkBaud, que
--           nos dirá que 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 número de
--           bits que se han enviado. Cuando se hayan enviado los 8
--           bits ,es decir, cuando hayan llegado 8 pulsos de Clkbaud,
--           se activará la señal FinDsplza8bits que hará que cambiemos
--           al siguiente estado (eBitFin).
-- eBitFin:  Este estado envía 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 señal de entrada
-- RxDatoReg seleccionando un 1 cuando no hay nada en la línea.

      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;
```

Contador Inicial de Prueba

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


-- Entidad del contador Ppal. Aqui se genera el pulso de transmite que
-- usa
-- Tx_serie para transmitir un nuevo dato.

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

entity Contador is
      Generic(a : in integer := 3;
                  b : in integer := 5);
    Port ( reset : in  STD_LOGIC;
           clk : in  STD_LOGIC;
                    enable : in STD_LOGIC;
                    load : in STD_LOGIC;
                    data_load : in  STD_LOGIC_VECTOR (3 downto 0);
                    cambiando : out std_logic;
           salida : out  STD_LOGIC_VECTOR (3 downto 0));
end Contador;
----------------------------------------------------------------------------


architecture Behavioral of Contador is


   -- Señales auxiliares --

      signal mienable, mireset, micambiando,miload: std_logic;
      signal aux: integer;
      signal misalida,midata_load: std_logic_vector(3 downto 0);

begin

      process (clk,reset,enable,load)
      begin
      if miload='1' then
                  misalida <= midata_load;
                  micambiando <= '1';
                  aux<=0;
      else
                  if mienable='1' then
                        if mireset='1' then
                              aux<=0;
                              misalida <= "0000";
                              micambiando <= '1';
                        elsif(clk'event and clk='1')
                                    if(aux=100000000)
                                    aux<=0;
                                    micambiando <= '1';
                                    if misalida = "1111" then
                                          misalida <= "0000";
                                    else
                                          misalida <= misalida +1;
                                    end if;
```

```
                        else
                              aux<=aux+1
                                    micambiando <= '0';

                        end if;
                  end if;
            end if;
      end if;

      end process;


      -- Asignación de señales --

      -- Como en la placa los botones están negados, negamos el reset.
      salida <= misalida;
      mireset <= not reset;
      mienable <= enable;
      cambiando <= micambiando;
      miload <= load;
      midata_load<=data_load;

end Behavioral;

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

Circuito_FPGA.vhd (Generado automáticamente)

```
----------------------------------------------------------------------------
----------
--Descripción:
--    Este fichero ha sido generado automáticamente por la aplicación
Nessy2.0
--    Se trata del fichero que describe la entidad top generada para
cualquier circuto que quiera ser ejecutado en la FPGA
--
--
--Especificaciones:
--    Circuito a ejecutar:
--          Num. Entradas: 19
--          Num. Salidas: 16
--Autor:
--    Carlos Sanchez-Vellisco Sanchez
--    Facultad de Informatica. Universidad Complutense de Madrid
--Fecha:
--    Thu Jun 10 23:02:35 CEST 2010
----------------------------------------------------------------------------
----------

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 Circuito_FPGA is
Port ( clk : in  STD_LOGIC;
reset : in  STD_LOGIC;
salida_serie : out  STD_LOGIC;
entrada_serie : in  STD_LOGIC;
ledsEntrada : out std_logic_vector(1 downto 0);
ledsSalida : out std_logic_vector(1 downto 0));
end Circuito_FPGA;


architecture Behavioral of Circuito_FPGA is


--Transmisor serie
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;


--Receptor Serie
component Rx_Serie
      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 component ;


--Entidad que se quiere ejecutar
component CONTADOR
      Port(
            RESET: in STD_LOGIC;
            CLK: in STD_LOGIC;
            ENABLE: in STD_LOGIC;
            LOAD: in STD_LOGIC;
            DATA_LOAD: in STD_LOGIC_VECTOR(15 downto 0);

            SALIDA: out STD_LOGIC_VECTOR(15 downto 0)
);
end component;


--Señales del circuito principal

--Entradas
signal mi_RESET: STD_LOGIC;
signal mi_CLK: STD_LOGIC;
signal mi_ENABLE: STD_LOGIC;
signal mi_LOAD: STD_LOGIC;
signal mi_DATA_LOAD: STD_LOGIC_VECTOR(15 downto 0);

--Salidas
signal mi_SALIDA: STD_LOGIC_VECTOR(15 downto 0);


--Señales para la Recepción Serie
signal mi_resetserie:std_logic;
signal mi_transmite:std_logic;
signal mi_datotxin:std_logic_vector(7 downto 0);
signal mi_datorxout:std_logic_vector(7 downto 0);
signal mi_avisoRx:std_logic;
signal mi_recibiendo:std_logic;


--Señales para la Transmisión Serie
signal mi_transmitiendo:std_logic;
signal mi_datoserieout:std_logic;
signal mi_rxdatoserie: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(31 downto 0);
signal Reg_salidas: std_logic_vector(31 downto 0);


--Señales para los estados del emisor/receptor de 32 bits
signal estadoEnt: integer;
signal estadoSal: integer;

signal ledsEnt:std_logic_vector(1 downto 0);
signal ledsSal:std_logic_vector(1 downto 0);
```

```
--Señales necesarias para la correcta entrada/salida con 32 bits
signal fin_recepcion : std_logic;
signal recibido,transmitido,biest_recibido, biest_transmitido:
std_logic;
signal frecibido,ftransmitido,fin: std_logic;  --flancos de fin


begin


--Asignación de señales a los componentes de la entrada/salida
f: 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);


--Asignación de señales al componente del circuito principal
U: CONTADOR port
map(mi_RESET,mi_CLK,mi_ENABLE,mi_LOAD,mi_DATA_LOAD,mi_SALIDA);


--Proceso encargado de la correcta recepción de datos (32 bits)
--Cada vez que se reciba un byte, se irá asignando a las entradas
desde las menos significativas a las más significatias
process(mi_recibiendo,mi_resetserie)
begin
      if mi_resetserie = '0' then
            estadoEnt <= 0;
            fin_recepcion <= '0';
      elsif mi_recibiendo'event and mi_recibiendo = '0' then
            fin_recepcion <= '0';
            if estadoEnt = 0 then
                  Reg_entradas(7 downto 0) <= mi_datorxout;
                  estadoEnt <= 1;
            elsif estadoEnt = 1 then
                  Reg_entradas(15 downto 8) <= mi_datorxout;
                  estadoEnt <= 2;
            elsif estadoEnt = 2 then
                  Reg_entradas(23 downto 16) <= mi_datorxout;
                  estadoEnt <= 3;
            elsif estadoEnt = 3 then
                  Reg_entradas(31 downto 24) <= mi_datorxout;
                  estadoEnt <= 0;
                  fin_recepcion <= '1'; --fin de la recepción
            else
                  Reg_entradas(7 downto 0) <= mi_datorxout;
                  estadoEnt <= 1;
            end if;
      end if;
end process;


--Proceso encargado de que la salida correspondiente del circuito esté
conectada a la salida serie antes de que la transmisión se produzca
process(estadoSal, clk, mi_resetserie)
begin
      if mi_resetserie = '0' then
            mi_datotxin <= Reg_salidas(7 downto 0);
```

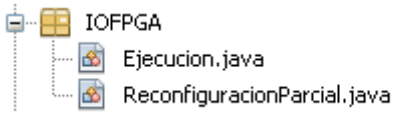
```
      elsif clk'event and clk = '1' then
            if estadoSal = 0 then
                  mi_datotxin <= Reg_salidas(7 downto 0);
            elsif estadoSal = 1 then
                  mi_datotxin <= Reg_salidas(15 downto 8);
            elsif estadoSal = 2 then
                  mi_datotxin <= Reg_salidas(23 downto 16);
            elsif estadoSal = 3 then
                  mi_datotxin <= Reg_salidas(31 downto 24);
            end if;
      end if;
end process;


--Proceso encargado de cambiar de estado cada vez que se comienza a
transmitir un byte
process(mi_transmitiendo, mi_resetserie)
begin
      if mi_resetserie = '0' then
            estadoSal<= 0;
            transmitido <= '0';
      elsif mi_transmitiendo'event and mi_transmitiendo = '1' then
            if estadoSal = 3 then
                  transmitido <= '1';
                  estadoSal <= 0;
            else
                  estadoSal <= estadoSal+1;
                  transmitido <= '0';
            end if;
      end if;
end process;


--Proceso encargado de registrar que la recepción ha terminado. La
salida del biestable ('recibido') se usará como reloj del circuito
principal
process(clk,fin_recepcion)
begin
      if clk'event and clk='1' then
            if fin_recepcion = '1' then
                  recibido <= '1'; --flanco positivo que hará funcionar
el circuito principal
            else
                  recibido <= '0';
            end if;
      end if;
      end process;


--Proceso encargado de registrar el fin de la recepción y de la
transmisión
process(clk)
begin
      if clk'event and clk='1' then
            biest_recibido <= recibido;
            biest_transmitido <= transmitido;
      end if;
end process;
```

```
--Proceso que indica que, o bien ha terminado una recepción de datos,
o bien ha terminado una transmisión
--Cuando se detecte un flanco positivo, se negará la señal que hace
que se transmita, de tal forma que si se estaba transitiendo,
--se deje de transmitir y si no se estaba transmitiendo se comience
una nueva transmisión
process(fin)
begin
      if fin'event and fin = '1' then
            if mi_transmite = '0' then
                  mi_transmite <= '1';
            else
                  mi_transmite <= '0';
            end if;
      end if;
end process;


--Este proceso es prescindible. Sólo a efectos de visualizar estado de
entrada
process(estadoEnt)
begin
      if estadoEnt = 0 then
            ledsEnt <= "00";
      elsif estadoEnt = 1 then
            ledsEnt <= "01";
      elsif estadoEnt = 2 then
            ledsEnt <= "10";
elsif estadoEnt = 3 then
            ledsEnt <= "11";
      end if;
end process;


--Este proceso es prescindible. Sólo a efectos de visualizar estado de
salida
process(estadoSal)
begin
      if estadoSal = 0 then
            ledsSal <= "00";
      elsif estadoSal = 1 then
            ledsSal <= "01";
      elsif estadoSal = 2 then
            ledsSal <= "10";
elsif estadoSal = 3 then
            ledsSal <= "11";
      end if;
end process;


ledsEntrada <= ledsEnt;
ledsSalida <= ledsSal;


--El reloj del circuito principal será el flanco que indique el fin de
la recepción
mi_clk <= recibido;

--Asignación de las señales del circuito general
mi_resetserie <= reset;
salida_serie <= mi_datoserieout;
```

```
mi_rxdatoserie <= entrada_serie;


--Asignación de las señales necesarias para la transmisión correcta
frecibido <= not biest_recibido and recibido; --flanco que indica fin
de recepcion
ftransmitido <= not biest_transmitido and transmitido; --flanco que
indica fin de transmision
fin <= frecibido or ftransmitido; --flanco que indica fin de envio o
transmision


--Asignación de las señales intermedias de entrada/salida a las del
circuito principal
mi_RESET <= Reg_entradas(0);
mi_ENABLE <= Reg_entradas(1);
mi_LOAD <= Reg_entradas(2);
mi_DATA_LOAD(0) <= Reg_entradas(3);
mi_DATA_LOAD(1) <= Reg_entradas(4);
mi_DATA_LOAD(2) <= Reg_entradas(5);
mi_DATA_LOAD(3) <= Reg_entradas(6);
mi_DATA_LOAD(4) <= Reg_entradas(7);
mi_DATA_LOAD(5) <= Reg_entradas(8);
mi_DATA_LOAD(6) <= Reg_entradas(9);
mi_DATA_LOAD(7) <= Reg_entradas(10);
mi_DATA_LOAD(8) <= Reg_entradas(11);
mi_DATA_LOAD(9) <= Reg_entradas(12);
mi_DATA_LOAD(10) <= Reg_entradas(13);
mi_DATA_LOAD(11) <= Reg_entradas(14);
mi_DATA_LOAD(12) <= Reg_entradas(15);
mi_DATA_LOAD(13) <= Reg_entradas(16);
mi_DATA_LOAD(14) <= Reg_entradas(17);
mi_DATA_LOAD(15) <= Reg_entradas(18);


Reg_salidas(0) <= mi_SALIDA(0);
Reg_salidas(1) <= mi_SALIDA(1);
Reg_salidas(2) <= mi_SALIDA(2);
Reg_salidas(3) <= mi_SALIDA(3);
Reg_salidas(4) <= mi_SALIDA(4);
Reg_salidas(5) <= mi_SALIDA(5);
Reg_salidas(6) <= mi_SALIDA(6);
Reg_salidas(7) <= mi_SALIDA(7);
Reg_salidas(8) <= mi_SALIDA(8);
Reg_salidas(9) <= mi_SALIDA(9);
Reg_salidas(10) <= mi_SALIDA(10);
Reg_salidas(11) <= mi_SALIDA(11);
Reg_salidas(12) <= mi_SALIDA(12);
Reg_salidas(13) <= mi_SALIDA(13);
Reg_salidas(14) <= mi_SALIDA(14);
Reg_salidas(15) <= mi_SALIDA(15);


end Behavioral;
```

4.2. Ficheros código java

Package IOFPGA



Ejecucion.java

```
package IOFPGA;

import app.Com;
import compiladorEntidad.Entidad;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.StringTokenizer;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

/**
 * Clase encargada de hacer que se ejecute correctamente el circuito
cargado
 * en la FPGA. Contiene los métodos necesarios para el envío y la
recepción de
 * datos cn dicho dispositivo. La ejecución se hará a través de un
Test Bench o
 * Banco de Pruebas que podrá leerse bien desde fichero, o bien desde
la pantalla
 * de la aplicación (pestaña TestBench).
 * Hereda de la clase Thread puesto que en ocasiones queremos que la
ejecución
 * se realice en paralelo con la escritura en la interfaz gráfica,
operaciones que
 * no pueden ser realizadas a la vez si no es en hilos diferentes.
 *
 * @author Carlos
 *
 */
public class Ejecucion extends Thread {

    /**
     * Estructura para almacenar las cadenas del banco de pruebas
cuando éste se
     * ha introducido por pantalla.
```

```
     */
    private ArrayList<String> cadenaaEnviar;

    /**
     * Indica que el hilo de ejecución se encuentra activo cuando está
a true
     */
    private boolean ejecutando;

    /**
     * Objeto para controlar al puerto serie RS232
     */
    private Com com1;

    /**
     * Almacena en un solo String todas las cadenas leidas de la
pantalla de
     * texto del Test Bench
     */
    private String ls_cadenaaejecutar;

    /**
     * Indica el número de bits de entrada que tiene la entidad sobre
la que se va a
     * ejecutar
     */
    private int li_bits_entrada;

    /**
     * Indica el número de bits de salida que tiene la entidad sobre
la que se va a
     * ejecutar
     */
    private int li_bits_salida;

    /**
     * Elemento gráfico sobre el que se desea escribir el número de
instruccion
     * mientras se ejecuta el  hilo de ejecución
     */
    private final JTextField ljtfield;

    /**
     * Indica si se ha ordenado al hilo correspondiente a esta
ejecución
     * que espere.
     */
    private boolean setwait;

    /**
     * Elemento gráfico sobre el que se desea escribir la salida de la
ejecución
     * mientras se ejecuta este hilo
     */
    private JTextArea ata_textarea;

    /**
     * Indica a la ejecución que el banco de pruebas hay que leerlo
desde fichero
     */
    private boolean entraDesdeFichero;
```

```
    /**
     * Lectura de ficheros
     */
    private BufferedReader bfr;

    /**
     * Ruta donde serán almacenados los ficheros correspondientes a la
ejecución
     * tales como la salida de la última ejecución y la salida golden.
     */
    static String rutaficherosSalida =
System.getProperties().getProperty("user.dir") + "\\salidas";

    /**
     * Fichero donde se volcará la salida
     */
    private final File fichero_salida;

    /**
     * Fichero con el cual se comparará la ejecución
     */
    private final File fichero_compararTraza;

    /**
     * Fichero para la escritura de lo que va sucediendo
     * durante la ejecución. Esto será útil a la hora
     * de visualizar lo sucedido en la reconfiguración
     * parcial.
     */
    private FileWriter ficheroLogEjec;

    /**
     * Indica si la ejecución coincide con la traza almacenada
     */
    private boolean coincideTraza;

    /**
     * Indica si hay que mostrar mensajes con pantallas modales al
ejecutar ya que
     * en el caso de reconfigurar varios ficheros, no querremos
ninguna pantalla
     * que se nos muestre entre ejecuciones.
     */
    private boolean mostrarMensaje;

    /**
     * Almacena la primera instrucción en la que no coincide la traza
     */
    private int NumInstrNoCoincideTraza;

    /**
     * Indica si en la ejecuición hay que comparar o no con la traza.
Por ejemplo,
     * cuando se está generando la salida golden no hay que hacer
ninguna comparación.
     */
    private boolean comparar;

    /**
```

```
     * Indica si se trata de una ejecución especial, que se da cuando
estamos
     * realizando ejecuciones sucesivas cambiando un solo bit de la
FPGA. Para esos
     * casos no querremos que la ejecución sea un hilo diferente ni
que ninguna
     * pantalla se nos muestre. Simplemente querremos volcar la salida
a un fichero
     * y guardar el resultado del método en un fichero de log.
     */
    private boolean reconfiguracionParcial;

    /**
     * Indica la posición en la que se encuentra el bit de reset en la
entidad que
     * se está ejecutando. La razón de almacenar esto es la de tener
la posibilidad
     * de resetear el circuito al comenzar cada ejecución. Para ello
mandaremos una
     * cadena de todo 0's excepto un 1 en la posición del reset
     */
    private int posicionResetEntidad;

    /**
     * Crea un objeto de la clase Ejecución y escribe el resultado en
un JTextField . Este constructor
     * está definido para el caso en el que el test bench es leido
desde la pantalla de la GUI
     * @param lj_jtf JTextField en el que se escribirá la salida que
reciba de la FPGA al ejecutar las instrucciones.
     * @param e Entidad (top) sobre la que vamos a ejecutar.
     * @param ac_com Puerto con el que nos comunicamos con la FPGA.
     * @param ata_textarea TextArea de la aplicación del cual se
recogerán las instrucciones a ejecutar.
     * @param comparar Booleano que indica si hay que comparar los
ficheros de traza y el de salida, para detectar diferencias.
     * @param nombreSalida Fichero en el que se guardará el resultado
de la ejecución.
     * @param nombreTraza Fichero de traza con el que se podrá
comparar la salida que está generando este objeto.
     * @param ab_reconfiguracionParcial Boolean que indica si estamos
ejecutando con la opción de reconfiguración parcial.
     *  Si es falso se ejecutará el hilo directamente y no mostrará
mensajes emergentes.
     */
    public Ejecucion(JTextField lj_jtf, Entidad e, Com ac_com,
JTextArea ata_textarea, boolean comparar, String nombreSalida, String
nombreTraza, boolean ab_reconfiguracionParcial) {
        fichero_salida = new File(rutaficherosSalida, nombreSalida);
        fichero_compararTraza = new File(rutaficherosSalida,
nombreTraza);
        this.ljtfield = lj_jtf;
        this.ejecutando = true;
        this.com1 = ac_com;
        this.li_bits_entrada = e.getBitsEntrada();
        this.li_bits_salida = e.getBitsSalida();
        cadenaaEnviar = new ArrayList();
        setwait = false;
        this.ata_textarea = ata_textarea;
        entraDesdeFichero = false;
        coincideTraza = true;
```

```
        mostrarMensaje = true;
        NumInstrNoCoincideTraza = 0;
        this.comparar = comparar;
        reconfiguracionParcial = ab_reconfiguracionParcial;
        this.posicionResetEntidad = e.getPosicionReset();
    }

    /**
     * Crea un objeto de la clase Ejecución y escribe el resultado en
un JTextField. Este constructor está definido
     * para el caso en el que el test bench es leido desde fichero
     * @param lj_jtf JTextField en el que se escribirá la salida que
reciba de la FPGA al ejecutar las instrucciones.
     * @param e Entidad top sobre la que vamos a ejecutar.
     * @param ac_com Puerto con el que nos comunicamos con la FPGA.
     * @param ata_textarea TextArea de la aplicación del cual se
podrían haber recogido las instrucciones (en este caso no se utiliza).
     * @param l_br (BufferedReader)del que se cogerán las
instrucciones a ejecutar.
     * @param comparar Booleano que indica si hay que comparar los
ficheros de traza y el de salida, para detectar diferencias.
     * @param nombreSalida Fichero en el que se guardará el resultado
de la ejecución.
     * @param nombreTraza Fichero de traza con el que se podrá
comparar la salida que está generando este objeto.
     * @param ab_reconfiguracionParcial Boolean que indica si estamos
ejecutando con la opción de reconfiguración parcial.
     *  Si es falso se ejecutará el hilo directamente y no mostrará
mensajes emergentes.
     */
    public Ejecucion(JTextField lj_jtf, Entidad e, Com ac_com,
JTextArea ata_textarea, BufferedReader l_br, boolean comparar, String
nombreSalida, String nombreTraza, boolean ab_reconfiguracionParcial) {

        fichero_salida = new File(rutaficherosSalida, nombreSalida);
        fichero_compararTraza = new File(rutaficherosSalida,
nombreTraza);
        this.ljtfield = lj_jtf;
        this.ejecutando = true;
        this.com1 = ac_com;
        this.li_bits_entrada = e.getBitsEntrada();
        this.li_bits_salida = e.getBitsSalida();
        cadenaaEnviar = new ArrayList();
        setwait = false;
        this.ata_textarea = ata_textarea;
        bfr = l_br;
        entraDesdeFichero = true;
        coincideTraza = true;
        mostrarMensaje = true;
        NumInstrNoCoincideTraza = 0;
        this.comparar = comparar;
        reconfiguracionParcial = ab_reconfiguracionParcial;
        this.posicionResetEntidad = e.getPosicionReset();
    }

    /**
     * Asigna la cadena que se quiere ejecutar para el caso de
ejecución desde
     * la pantalla de la GUI. Todas las cadenas a ejecutar estarán
contenidas
     * en as_cadenaaejecutar
```

```
     * @param as_cadenaajecutar Cadena a ejecutar.
     */
    public void setCadena(String as_cadenaajecutar) {
        ls_cadenaaejecutar = as_cadenaajecutar;
    }

    /**
     * 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;
    }

    /**
     * Indica si el hilo que se está ejecutando tiene que pararse o
reanudarse, según el argumento que recibe.
     * @param setwait boolean, valor true que indica que el hilo tiene
que continuar, false para pararlo.
     */
    public void setSetwait(boolean setwait) {
        this.setwait = setwait;
    }

    /**
     * Establece el fichero de escritura para el log de ejecucion
     * @param wr El fichero de escritura
     */
    public void setFileLogEjec(FileWriter wr){
        this.ficheroLogEjec = wr;
    }

    /**
     * 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.
     */
    private void enviarBinaria(String s) {
        String cad3, cad2, cad1, cad0;
        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
        }
```

```
            cad3 = s.substring(0, 8);
            cad2 = s.substring(8, 16);
            cad1 = s.substring(16, 24);
            cad0 = s.substring(24);
            try {
                com1.sendSingleData(traduceString(cad0));
                com1.sendSingleData(traduceString(cad1));
                com1.sendSingleData(traduceString(cad2));
                com1.sendSingleData(traduceString(cad3));
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }

        /**
         * Comprueba que una serie de cadenas tiene el formato correcto,
es decir,
         * tiene el número de bits adecuado y sólo está compuesta por 0's
y 1's
         * @return boolean si todo es correcto.
         */
        public boolean convierteCadenas() {
            StringTokenizer st;
            st = new StringTokenizer(this.ls_cadenaaejecutar, "\n\r");
            int numBits = this.li_bits_entrada;
            boolean correcto = true;
            int numCadenas = st.countTokens();
            cadenaaEnviar = new ArrayList<String>();
            int i = 0;
            correcto = numCadenas > 0;
            while (st.hasMoreTokens() && correcto) {
                String cadena = st.nextToken();
                if (cadena.length() == numBits) {
                    this.cadenaaEnviar.add(cadena);
                    correcto = traduceString(cadena) >= 0;
                    i++;
                } else {
                    correcto = false;
                }
            }
            return correcto;
        }

        /**
         * Método que comienza a ejecutar el hilo.
         *
         */
        @Override
        public void run() {
            synchronized (this) {
                ejecuta();
            }
        }

        /**
         * Método que consulta el estado de la ejecucución del hilo.
         * @return Boolean estado de la ejecución. Cierto si está
ejecutando, falso en caso contrario.
         */
        public boolean getejecutando() {
            return ejecutando;
```

```
    }

    /**
     * Método que termina la ejecución de un hilo.
     */
    public void pararrecepcionfpga() {
        this.ejecutando = false;
        if (!coincideTraza && mostrarMensaje &&
!reconfiguracionParcial) {
            JOptionPane.showMessageDialog(this.ata_textarea, "La
Salida actual NO coincide con la salida generada por la última
ejecución. Revise Instrucción num: " + NumInstrNoCoincideTraza,
"Info", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    /**
     * Procedimiento que lee de la FPGA un entero y lo transforma a la
cadena de bits.
     * @return Cadena de bits recibida.
     */
    private String recibirBinaria(int numBitsSalida) throws Exception
{
        int num;
        String cadenaRecibida = "";
        for (int i = 0; i < 4; i++) {
            num = com1.receiveSingleDataInt();
            cadenaRecibida = this.convertirByteBinario(num) +
cadenaRecibida;
        }
        return cadenaRecibida.substring(cadenaRecibida.length() -
numBitsSalida);
    }

    /**
     * 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;
    }

    /**
     * Procedimiento que ejecuta el hilo.
     */
```

```
    public void ejecuta() {
        try {

            File dirsalida = new File(rutaficherosSalida);
            if(!dirsalida.exists()){
                dirsalida.mkdir();
            }

            if(!fichero_compararTraza.exists()){
                fichero_compararTraza.createNewFile();
            }
            FileReader fr = new FileReader(fichero_compararTraza);
            BufferedReader rw = new BufferedReader(fr);
            FileWriter file_wr;
            String linea_traza = "";
            int instruccion = 0;
            String datoaenviar = null;
            fichero_salida.createNewFile();
            boolean seguir;
            if (entraDesdeFichero) {
                datoaenviar = bfr.readLine();
                seguir = datoaenviar != null;
            } else {
                seguir = instruccion < this.cadenaaEnviar.size();
            }
            file_wr = new FileWriter(fichero_salida, false);
            //Enviamos un reset antes de la ejecución.
            this.enviaReset();
            this.recibirBinaria(li_bits_salida);
            while (ejecutando && seguir) {
                if (this.setwait) {
                    System.out.println("Ejecución antes");
                    file_wr.close();
                    if (comparar && !coincideTraza && mostrarMensaje)
{
                        if (!reconfiguracionParcial) {

JOptionPane.showMessageDialog(this.ata_textarea, "La Salida actual NO
coincide con la salida Golden. Revise Instrucción num: " +
NumInstrNoCoincideTraza, "Info", JOptionPane.INFORMATION_MESSAGE);
                        }
                        this.escribeEnLog("La salida no coincide con
Salida Golden en la instruccion: " + NumInstrNoCoincideTraza+"\n\n");
                        System.out.println("LA EJECUCION HA SIDO
MODIFICADA. Ver log al finalizar");
                        mostrarMensaje = false;
                    }
                    this.wait();
                    file_wr = new FileWriter(fichero_salida, true);
                    System.out.println("Ejecución despues");
                    this.setwait = false;
                } else {
                    if (!entraDesdeFichero) {
                        datoaenviar =
this.cadenaaEnviar.get(instruccion);
                    }
                    this.enviarBinaria(datoaenviar);//TODO divido 32
                    String c =
this.recibirBinaria(this.li_bits_salida);
                    if (instruccion < 250000) {
```

```
                        this.ata_textarea.append((instruccion + 1) +
". " + c + "\n");
                    }
                    file_wr.write(c + "\n");
                    this.escribeEnLog(c);

                    if (comparar && coincideTraza && linea_traza !=
null) {
                        linea_traza = rw.readLine();
                        this.escribeEnLog("\n");
                        if (linea_traza == null ||
linea_traza.compareTo(c) != 0) {
                            coincideTraza = false;
                            NumInstrNoCoincideTraza = instruccion + 1;
                            this.escribeEnLog("<----- falla aqui\n");

                        }
                    }
                }

this.ata_textarea.setCaretPosition(this.ata_textarea.getText().length(
));
                this.ljtfield.setText(Integer.toString(instruccion +
1));
                instruccion++;
                if (entraDesdeFichero) {
                    datoaenviar = bfr.readLine();
                    seguir = datoaenviar != null;
                } else {
                    seguir = instruccion < this.cadenaaEnviar.size();
                }

            }
            file_wr.close();
            rw.close();
            if (comparar && coincideTraza) {
                if (!reconfiguracionParcial) {
                    JOptionPane.showMessageDialog(this.ata_textarea,
"La Salida actual coincide con la Golden", "Info",
JOptionPane.INFORMATION_MESSAGE);
                }
                this.escribeEnLog("La Salida actual coincide con la
Golden\n\n");
                System.out.println("Ejecucion correcta");
            } else {
                if (comparar && mostrarMensaje) {
                    if (!reconfiguracionParcial) {

JOptionPane.showMessageDialog(this.ata_textarea, "La Salida actual NO
coincide con la salida generada por la última ejecución. Revise
Instrucción num: " + NumInstrNoCoincideTraza, "Info",
JOptionPane.INFORMATION_MESSAGE);
                    }
                    this.escribeEnLog("La Salida actual NO coincide
con la salida Golden. Revise Instrucción num: " +
NumInstrNoCoincideTraza+"\n\n");
                    System.out.println("LA EJECUCION HA SIDO
MODIFICADA. Ver log al finalizar");

                    mostrarMensaje = false;
                }
```

```
            }
        } catch (InterruptedException ex) {
Logger.getLogger(Ejecucion.class.getName()).log(Level.SEVERE, null,
ex);
        } catch (Exception ex) {
Logger.getLogger(Ejecucion.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    /**
     * Función para copiar un fichero en otro fichero. En nuestro caso
copiamos el
     *  archivo de escritura en el de comparar con traza.
     * @throws IOException
     */
    public void CopiarSalida() throws IOException {

        InputStream in = new FileInputStream(fichero_salida);
        OutputStream out = new
FileOutputStream(fichero_compararTraza);
        byte[] buf = new byte[1024];
        int len;
        while ((len = in.read(buf)) > 0) {
            out.write(buf, 0, len);
        }
        in.close();
        out.close();
    }

    /**
     * Comprueba si un fichero de Test Bench tiene el formato
correcto, es decir,
     * si sus cadenas tienen el tamaño correcto correspondiente con
las entradas
     * de la entidad a ejecutar, y si se trata sólamente de cadenas de
0's y 1's
     * @param ficheroTB Fichero de Test Bench a analizar
     * @return true si el formato es correcto y false en caso
contrario
     */
    public boolean formatoCorrectoFicheroTB(String ficheroTB) {
        String linea = null;
        boolean correcto = true;
        try {
            BufferedReader bf;
            bf = new BufferedReader(new FileReader(ficheroTB));
            do {
                linea = bf.readLine();
                if (linea != null) {
                    if (linea.length() == this.li_bits_entrada) {
                        int i = 0;
                        while (i < linea.length() && correcto) {
                            correcto = linea.charAt(i) == '0' ||
linea.charAt(i) == '1';
                            i++;
                        }
                    } else {
                        correcto = false;
```

```
                    }
                }
            } while (linea != null && correcto);
            bf.close();
        } catch (IOException ex) {
            correcto = false;
        }
        return correcto;
    }

    /**
     * Realiza el envío de reset a la entidad, mandando para ello una
cadena
     * con todo ceros excepto un uno en la posición correspondiente al
reset
     */
    public void enviaReset() {
        String cadenaReset = "";
        for (int i = 0; i < 32; i++) {
            if (i == this.posicionResetEntidad) {
                cadenaReset = "1" + cadenaReset;
            } else {
                cadenaReset = "0" + cadenaReset;
            }
        }
        this.enviarBinaria(cadenaReset);
    }

    /**
     * Escribe la cadena s en el fichero de log en el que se
introducen todos
     * los detalles de ejecución para el caso del proceso de
reconfiguración.
     * Se controla que el fichero esté inicializado para el caso de
ejecuciones
     * en las que no se escribe en el log (ejecuciones simples).
     * @param s La cadena que se desea escribir en el fichero de log.
     */
    public void escribeEnLog(String s){
        if (this.ficheroLogEjec != null){
            try {
                this.ficheroLogEjec.write(s);
            } catch (IOException ex) {

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

ReconfiguracionParcial.java

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

import java.io.File;
import java.io.FileNotFoundException;
```

```
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;
import nessy20.GUIPrincipal;
import nessy20.SeleccionNumIter;
import org.apache.log4j.Logger;

/**
 * Clase que contiene el proceso en el que se incluye el algoritmo
para la
 * reconfiguración parcial de la FPGA. Necesita ser creado en una
nueva clase
 * que herede de un hilo, de tal forma que la ejecución pueda ser
parada en
 * cualquier momento desde la interfaz gráfica de usuario.
 *
 * @author Carlos, Tony, David
 */
public class ReconfiguracionParcial extends Thread {


    private static Logger log =
Logger.getLogger(ReconfiguracionParcial.class);
    /**
     * Necesitamos tener la interfaz gráfica ya que contiene algunos
     * métodos de los que vamos a utilizar.
     */
    private GUIPrincipal gui;
    /**
     * Indica si se está ejecutando la reconfiguración para poder
parar
     * en cualquier momento el proceso
     */
    private boolean ejecutandoReconfiguracion;
    /**
     * Panel que será activado mientras se ejecuta el proceso
     */
    private javax.swing.JPanel panelOutPut;
    /**
     * La ruta del fichero .BIT que se va a modificar
     */
    private String rutaBit;
    /**
     * Número de iteraciones que se quiere ejecutar la reconfiguración
parcial
     */
    private int numIteraciones;

    /**
     * Inicia el hilo de la reconfiguración parcial
     * @param in_gui GUI de entrada a copiar
     * @param rutaBit Ruta del .BIT a modificar
     */
    public ReconfiguracionParcial(GUIPrincipal in_gui, String rutaBit)
{
        this.rutaBit = rutaBit;
        this.gui = in_gui;
        this.numIteraciones = 10;
    }

    /**
```

```
     * Método que contiene el algoritmo que ejecuta la reconfiguración
parcial
     * sucesiva.En primer lugar se cargará una entidad seguida de su
fichero
     * .BIT correspondiente. A continuación se pedirá al usuario un
fichero
     * para cargar el banco de pruebas. Tras cargarlo, y sabiendo que
en la
     * FPGA está cargado el fichero .BIT correcto (sin modificar), se
generará
     * una salida especial, llamada salida Golden, la cual será con la
que
     * comparemos en el resto de ejecuciones. Este paso sólo lo
realizaremos
     * una vez, al comienzo del proceso.
     * A continuación entraremos en un bucle, en el cual se irá
iterando para cada bit
     * y para cada frame distinto. En cada una de las iteraciones se
ejecutará
     * la aplicación de reconfiguración aplicada siempre al mismo
fichero .BIT
     * original (introducido por el usuario) en el que iremos
cambiando los
     * parámetros de frame y bit. Cada vez que se ejecute la
aplicación se
     * generarán dos ficheros de configuración: fichero_modif.bit y
fichero_
     * modifRestore.bit. Cargaremos el primer fichero en la FPGA, lo
que es
     * equivalente a que una partícula solar modificara el valor del
bit de la
     * LUT que estamos modificando. Una vez inyectado el error
(cargado el
     * fichero de configuración) volveremos a ejecutar el circuito en
la FPGA,
     * con las mismas entradas con las que habíamos generado la salida
Golden.
     * Compararemos las nuevas salidas obtenidas con la salida Golden
para ver
     * si el error ha incidido en la ejecución del circuito.
     * Después tendremos que devolver a la FPGA a su “estado
anterior”.
     * Es decir, tenemos que restaurar el bit que acabamos de
modificar.
     * Para ello ordenaremos la carga del segundo fichero que se había
generado:
     * fichero_modifRestore.bit.

     * @return true si la ejecución ha sido correcta y false en caso
contrario
     */
    public boolean reconfiguracionParcial() {
        int numBits = 32;
        int numFrames = 36194;

        boolean b = false;
        ejecutandoReconfiguracion = true;
        if (gui.cargarBitConChooser() &&
gui.SeleccionTBModifFichero()) {
            gui.setEnabledBtnDetenerInyeccion(true);
            gui.seleccionaPanel(panelOutPut);
```

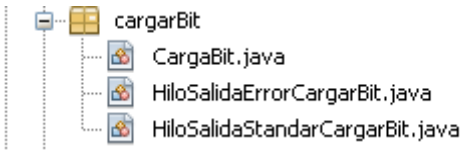
```
            int frame = 0;
            int bit = 0;
            try {

                FileWriter fw = new FileWriter(new
File("salidas//logEjec.txt"));
                String fichero = gui.getFichero_bit();
                gui.setFwLog(fw);
                while (frame < numFrames && ejecutandoReconfiguracion)
{
                    bit = 0;
                    while (bit < numBits && ejecutandoReconfiguracion)
{
                        fw.write("\n\nModificando FRAME: " + frame + "
BIT: " + bit + "\n");
                        log.info("\n\n********** Modificando FRAME: "
+ frame + " BIT: " + bit + " ************");

                        String coms = "cmd.exe /K java -jar
Virtex_II_Partial_Reconfiguration.jar -i " + fichero + " -o " +
rutaBit + "\\circuito_fpga_modif -f " + frame + " -b " + bit;
                        fw.write("Ejecutando: " + coms + "\n");
                        log.info("Modificando .bit");
                        Process p = Runtime.getRuntime().exec(coms);
                        log.info("Cargando BIT: " + rutaBit +
"\\circuito_fpga_modif.bit");
                        fw.write("Cargando BIT: " + rutaBit +
"\\circuito_fpga_modif.bit\n");
                        if (this.gui.getCom1() == null) {
                            gui.inicializarPuertoSerie();
                        }
                        gui.cargarBit(rutaBit +
"\\circuito_fpga_modif.bit", false);
                        if (this.gui.getCom1() == null) {
                            fw.write("Ejecutando...\n");
                            if (gui.inicializarPuertoSerie()) {
                                gui.ejec(true);
                            }
                        } else {
                            gui.ejec(true);
                        }
                        log.info("Cargando BIT: " + rutaBit +
"\\circuito_fpga_modifRestorer.bit");
                        fw.write("Cargando BIT: " + rutaBit +
"\\circuito_fpga_modifRestorer.bit\n");
                        if (this.gui.getCom1() == null) {
                            gui.inicializarPuertoSerie();
                        }
                        b = gui.cargarBit(rutaBit +
"\\circuito_fpga_modifRestorer.bit", false);

                        bit++;
                    }
                    frame++;
                }
                fw.close();
            } catch (FileNotFoundException ex) {
                //
Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
                return false;
```

```
            } catch (IOException ex) {
                //
Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
                return false;
            }/* catch (InterruptedException ex) {
            return false;
            }*/
        }

        ejecutandoReconfiguracion = false;
        return b;
    }


/**
 * Se hará para un número determinado de iteraciones que introducirá
el usuario.
 * En primer lugar se cargará una entidad seguida de su fichero .BIT
 * correspondiente. A continuación se pedirá al usuario un fichero
para cargar
 * el banco de pruebas. Tras cargarlo, y sabiendo que en la FPGA está
cargado el
 * fichero .BIT correcto (sin modificar), se generará una salida
especial,
 * llamada salida Golden, la cual será con la que comparemos en el
resto de
 * ejecuciones. Este paso sólo lo realizaremos una vez, al comienzo
del proceso.
 * A continuación se ejecutarán todas las iteraciones. En cada una de
ellas se
 * ejecutará la aplicación de reconfiguración aplicada siempre al
mismo fichero
 * .BIT original (introducido por el usuario) en el que iremos
cambiando los
 * parámetros de frame y bit de forma aleatoria. Cada vez que se
ejecute la
 * aplicación se generarán dos ficheros de configuración:
fichero_modif.bit y
 * fichero_modifRestore.bit. Cargaremos el primer fichero en la FPGA,
lo que es
 * equivalente a que una partícula solar modificara el valor del bit.
Una vez
 * inyectado el error (cargado el fichero de configuración) volveremos
a
 * ejecutar el circuito en la FPGA, con las mismas entradas con las
que habíamos
 * generado la salida Golden.
 * Compararemos las nuevas salidas obtenidas con la salida Golden para
ver si el
 * error ha incidido en la ejecución del circuito.
 * Ahora tenemos que devolver a la FPGA a su “estado anterior”. Es
decir, tenemos
 * que restaurar el bit que acabamos de modificar. Para ello
ordenaremos la
 * carga del segundo fichero que se había generado:
fichero_modifRestore.bit.
 * Con el circuito restaurado, ya podemos volver a ejecutar una
iteración más
 * del bucle.
 * @return true si la ejecucion ha sido correcta
```

```
 */
    public boolean reconfiguracionParcialAleatoria() {
        int numBits = 32;
        int numFrames = 36194;
        int numIteracion = 0;
        boolean errorCarga = false;
        ejecutandoReconfiguracion = true;
        int iter=gui.dameNumIter();
        if (iter > 0) {
            this.numIteraciones = iter;
            errorCarga = !gui.cargarBitConChooser();
            if (!errorCarga && gui.SeleccionTBModifFichero()) {


                gui.seleccionaPanel(panelOutPut);
                //  gui.setEnabledBtnDetenerInyeccion(true);
                int frame = 0;
                int bit = 0;
                try {
                    FileWriter fw = new FileWriter(new
File("salidas//logEjec.txt"));
                    String fichero = gui.getFichero_bit();
                    gui.setFwLog(fw);
                    while (numIteracion < this.numIteraciones &&
ejecutandoReconfiguracion && !errorCarga) {
                        bit = (int) (Math.random() * numBits);
                        frame = (int) (Math.random() * numFrames);

                        fw.write("\n\nModificando FRAME: " + frame + "
BIT: " + bit + "\n");
                        log.info("\n\nIteracion: " + numIteracion +
"\n********** Modificando FRAME: " + frame + " BIT: " + bit + "
**********");

                        String coms = "cmd.exe /K java -jar
Virtex_II_Partial_Reconfiguration.jar -i " + fichero + " -o " +
rutaBit + "\\circuito_fpga_modif -f " + frame + " -b " + bit;
                        fw.write("Ejecutando: " + coms + "\n");
                        log.info("Modificando .bit");
                        Process p = Runtime.getRuntime().exec(coms);
                        log.info("Cargando BIT: " + rutaBit +
"\\circuito_fpga_modif.bit");
                        fw.write("Cargando BIT: " + rutaBit +
"\\circuito_fpga_modif.bit\n");
                        if (this.gui.getCom1() == null) {
                            gui.inicializarPuertoSerie();
                        }
                        errorCarga = errorCarga ||
!gui.cargarBit(rutaBit + "\\circuito_fpga_modif.bit", false);
                        if (this.gui.getCom1() == null) {
                            fw.write("Ejecutando...\n");
                            if (gui.inicializarPuertoSerie()) {
                                gui.ejec(true);
                            }
                        } else {
                            gui.ejec(true);
                        }
                        log.info("Cargando BIT: " + rutaBit +
"\\circuito_fpga_modifRestorer.bit");
                        fw.write("Cargando BIT: " + rutaBit +
"\\circuito_fpga_modifRestorer.bit\n");
```

```
                        if (this.gui.getCom1() == null) {
                            gui.inicializarPuertoSerie();
                        }
                        errorCarga = errorCarga ||
!gui.cargarBit(rutaBit + "\\circuito_fpga_modifRestorer.bit", false);
                        numIteracion++;
                    }
                    if (errorCarga){
                        JOptionPane.showMessageDialog(gui, "Error
durante el proceso de inyección de errores. No ha finalizado
correctamente", "Información", JOptionPane.ERROR_MESSAGE);
                    }
                    fw.close();
                    gui.setEnabledBtnDetenerInyeccion(false);
                } catch (FileNotFoundException ex) {
                    gui.setInyeccErr(false);
                    //
Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
                } catch (IOException ex) {
                     gui.setInyeccErr(false);
                    //
Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
                }
            }
        }
        gui.setInyeccErr(false);
        return !errorCarga;
    }


@Override
    public void

run() {
        synchronized (this) {
            this.reconfiguracionParcialAleatoria();
        }

}

    /**
     * Detiene el proceso de reconfiguración parcial sucesiva
     */
    public void

pararreconfiguracionparcial() {
        ejecutandoReconfiguracion = false;
    }
}
```

Package cargarBit



CargaBit.java

```
package cargarBit;

import java.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import nessy20.GUIPrincipal;
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 * Clase para que contiene la funcionalidad necesaria para
 * cargar un archivo .BIT con la aplicación IMPACT
 * @author User
 */
public class CargaBit {

    //private final static String RUTA_IMPACT =
"C://Xilinx82//bin//nt//impact.exe";
    private String ficheroBit;
    GUIPrincipal interfaz;
    String rutaImpact;
    private boolean SalidaCargaBit;

    /**
     * Constructor de la clase.
     * @param interfaz Interfaz sobre la que estamos ejecutando
     * @param fich String con la ruta del fichero .bit que deseamos
cargar.
     */
    public CargaBit(GUIPrincipal interfaz, String fich, String
rutaImpact) {
        this.ficheroBit = fich;
        this.interfaz = interfaz;
        this.rutaImpact = rutaImpact;


    }

    /**
     * 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();
```

```
    }

    /**
     * Procedimiento que carga un archivo .bit en la FPGA.
     * Ejecuta el script de carga el archivp .bit y lanza dos hilos
para
     * leer la salida que ha generado esta ejecución. Uno lee la
salida de error
     * y otro la salida estándar.
     * @param escribirEnPantalla Para no escribir la salida en la
pantalla
     * en caso de que estemos en el proceso de reconfiguración.
     * @return Cierto si se ha conseguido cargar correctamente, falso
en caso contrario.
     * @throws java.io.FileNotFoundException
     */
    public boolean cargar(boolean escribirEnPantalla) throws
FileNotFoundException {
       if (existeFichero(rutaImpact)) {
            try {
                SalidaCargaBit = false;
                interfaz.escribirEnPantalla("Cargando... " +
ficheroBit);
                FileOutputStream os = new
FileOutputStream("carga.txt");
                BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(os));
                /*String coms = "setMode -ss \nsetMode -sm \n" +
                "setMode -sm \nsetMode -hw140 \nsetMode -spi\nsetMode
-acecf\nsetMode -acempm\nsetMode -pff\n" +
                "setMode -bs\nsetMode -bs\nsetCable -port
auto\nIdentify\naddDevice -p 3 -file \""+ ficheroBit+"\"\n" +
                "deleteDevice -p 4\nProgram -p 3 -defaultVersion
0\nexit";*/
                String coms = "setMode -bs \n" +
                        "setCable -port auto\n" +
                        "Identify\n" +
                        "identifyMPM\n" +
                        "assignFile -p 3 -file \"" + ficheroBit +
"\"\n" +
                        "Program -p 3\n" +
                        "exit";
                bw.write(coms);
                bw.close();
                //rutaImpact = "C://Xilinx82//bin//nt//impact.exe";
                //rutaImpact =
"C://Xilinx10.1//ISE//bin//nt//impact.exe";
                Process p;

                p = Runtime.getRuntime().exec(rutaImpact + " -batch
carga.txt");

                HiloSalidaStandarCargarBit hilo = new
HiloSalidaStandarCargarBit(interfaz, escribirEnPantalla, p, this);
                hilo.start();
                HiloSalidaErrorCargarBit hilo2 = new
HiloSalidaErrorCargarBit(interfaz, escribirEnPantalla, p);
                hilo2.start();
                try {
                    hilo.join();
                    hilo2.join();
```

```
                } catch (InterruptedException ex) {
Logger.getLogger(CargaBit.class.getName()).log(Level.SEVERE, null,
ex);
                }

                if (SalidaCargaBit) {
                    System.out.println("Programada correctamente");
                } else {
                    System.out.println("Error al cargar la FPGA");
                }


            } catch (IOException e) {
                SalidaCargaBit = false;
            }
        } else {
            SalidaCargaBit = false;
        }
        return SalidaCargaBit;
    }

    /**
     * Parámetro que establece si la carga del archivo .bit ha sido
correcta.
     * @param SalidaCargaBit Boolean con el nuevo valor que indica si
la
     * carga del archivo .bit ha sido correcta
     */
    public void setSalidaCargaBit(boolean SalidaCargaBit) {
        this.SalidaCargaBit = SalidaCargaBit;
    }
}
```

HiloSalidaErrorCargarBit.java

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

package cargarBit;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;
import nessy20.GUIPrincipal;

/**
 *
 * Hio encargado de leer la salida de error generada por el proceso
que carga
 * el archivo .bit en la FPGA.
 */


public class HiloSalidaErrorCargarBit extends Thread {
```

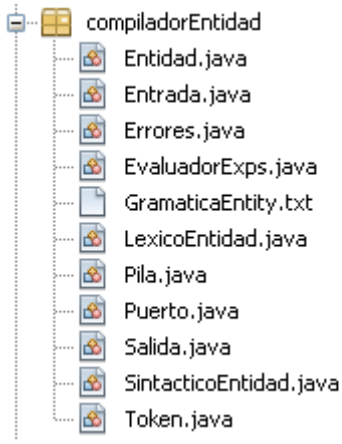
```
    private GUIPrincipal interfaz;
    private boolean escribirEnPantalla;
    private Process proc ;


    /**
     * Contructora de la clase HiloSalidaErrorCargarBit
     * @param in_interfaz GUIPrincipal de la aplicación.
     * @param in_escribirEnPantalla Boolean que indica si la aplicacón
tiene que escribir
     * el resultado en la pantalla da la aplicación.
     * @param f Process que se ha encargado de intentar cargar el
archivo .bit.
     */
    public HiloSalidaErrorCargarBit(GUIPrincipal in_interfaz, boolean
in_escribirEnPantalla, Process f) {
        interfaz = in_interfaz;
        escribirEnPantalla = in_escribirEnPantalla;
        proc = f;
    }

   /**
     * Proceso encargado de leer la salida de error generada por el
proceso de
     * cargar el archivo .bit.
     */
    private void cargabit(){
    boolean correcto;
    try {
        InputStream is = proc.getErrorStream();
        BufferedReader br = new BufferedReader(new
InputStreamReader(is));

        String s;
        s = br.readLine();
        boolean errorCarga = true;
        while(s!=null && errorCarga){
            if(escribirEnPantalla)
                interfaz.escribirEnPantalla(s);
            if (s.contains("Programmed successfully")){
                errorCarga = false;
            }
            s=br.readLine();
        }
        correcto = !errorCarga;
    } catch (IOException ex) {

Logger.getLogger(HiloSalidaErrorCargarBit.class.getName()).log(Level.S
EVERE, null, ex);
    }
}
    @Override
    public void run() {
        this.cargabit();

    }
}
```

HiloSalidaStandarCargarBit.java

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

package cargarBit;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;
import nessy20.GUIPrincipal;

/**
 *
 * Hio encargado de leer la salida estándar generada por el proceso
que carga
 * el archivo .bit en la FPGA.
 */


public class HiloSalidaStandarCargarBit extends Thread {

    private GUIPrincipal interfaz;
    private boolean escribirEnPantalla;
    private Process proc ;
    private CargaBit cargabit;

    /**
     * Contructora de la clase HiloSalidaStandarCargarBit
     * @param in_interfaz GUIPrincipal de la aplicación.
     * @param in_escribirEnPantalla Boolean que indica si la aplicacón
tiene que escribir
     * el resultado en la pantalla da la aplicación.
     * @param f Process que se ha encargado de intentar cargar el
archivo .bit.
     * @param in_carga Objeto de la clase Carga_bit que ha llamado al
hilo
     */
    public HiloSalidaStandarCargarBit(GUIPrincipal in_interfaz,
boolean in_escribirEnPantalla, Process f,CargaBit in_carga) {
        interfaz = in_interfaz;
        escribirEnPantalla = in_escribirEnPantalla;
        proc = f;
        cargabit = in_carga;
    }
    /**
     * Proceso encargado de leer la salida estandar generada por el
proceso de
     * cargar el archivo .bit y establecer dependiendo la salida, en
la clase
     * CargaBit si ha habido algún error o no.
     */
    private void cargabit(){
    boolean correcto ;
    try {
        InputStream is = proc.getInputStream();
        BufferedReader br = new BufferedReader(new
InputStreamReader(is));
```

```
            String s;
            s = br.readLine();
            boolean errorCarga = true;
            while(s!=null && errorCarga){
                if(escribirEnPantalla)
                    interfaz.escribirEnPantalla(s);
                if (s.contains("Programmed successfully")){
                    errorCarga = false;
                }
                s=br.readLine();
            }
            correcto = !errorCarga;
            cargabit.setSalidaCargaBit(correcto);
        } catch (IOException ex) {

Logger.getLogger(HiloSalidaStandarCargarBit.class.getName()).log(Level
.SEVERE, null, ex);
        }
}
    @Override
    public void run() {
        this.cargabit();
    }
}
```

Package compiladorEntidad



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;
    }
}
```

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;
```

```
    }

}
```

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;
    }

}
```

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;
            }
        }

}
```

LexicoEntidad.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 lÃ-nea
                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;
    }
}
```

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;
    }

}
```

Puerto.java

```
package compiladorEntidad;



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

/**
 * 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;
    }




}
```

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{

}
```

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() +
"\".");
    }



}
```

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.
```

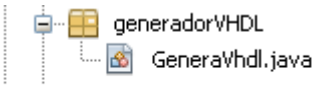
```
     * @return El entero con la línea donde se encuentra el Token.
     */
    public int getNumLinea() {
        return numLinea;
    }
}
```

Además contiene el fichero GramaticaEntity.txt

GramaticaEntity.txt

```
Entidad = Cabecera entity identificador is Generic Puertos end
identificador
Generic = generic (Variables);
Variables = Variable RVariables;
Variable = identificador : InOut integer := entero
RVariables = ; Variable Rvariables
RVariables = lambda
Puertos = port(Señales);
Señales = Señal RSeñales
Señal = identificador : InOut Tipo
RSeñales = ; Señal RSeñales
RSeñales = lambda
InOut = in
InOut = out
Tipo = std_logic
Tipo = std_logic_vector(Exp downto Exp)
Exp = Term RExp
RExp = + Exp | - Exp | lambda
Term = Fac RTem
RTerm = * Term | / Term | lambda
Fac = ( E ) | nat | identificador
Cabecera = *
```

Package generadorVHDL



GeneraVhdl.java

```
/*
 * 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;

/**
 * Fichero encargado de generar automáticamente el VHDL de circuito
top
 * (Circuito_FPGA.vhd) que será el que conecte el circuito que
queramos ejecutar
 * en la FPGA con nuestro módulo de entrada/salida. Además también
contendrá
 * los procesos necesarios para leer y escribir 32 bits desde/hacia la
FPGA.
 * esta clase está estructurada en diferentes métodos cada uno de los
cuales
 * escriben un bloque concreto del fichero.
 *
 * @author Carlos, Tony y David.
 */
public class GeneraVhdl {

    /**
     * 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
     * @param fichero Sobre el que vamos a generar el nuevo vhdl.
     * @param entidad Entidad con la que estamos trabajando y a partir
de la cual se genera el vhdl.
     * @param errores Objeto de la clase Errores para ir almacenando
los errores encontrados.
     * @throws java.io.IOException
     */
    public GeneraVhdl(String fichero, Entidad entidad, Errores
errores) throws IOException {
        this.fichero = fichero;
        this.entidad = entidad;
        this.errores = errores;
        nomEntidadGeneral = "Circuito_FPGA"; //de momento lo llamamos
asi
    }

    /**
     * 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;
    }

    /**
     * Hace la distinción entre is es un puerto con varios bits o
solamente uno.
     * En cada caso escribirá STD_LOGIC o STD_LOGIC_VECTOR.
     * @param p Puerto del que se desea saber su tipo
     * @return String con el tipo de puerto
     */
    private String tipoPuerto(Puerto p) {
        int numBits = p.getNumBits();
        if (numBits > 1) {
            return "STD_LOGIC_VECTOR(" + (numBits - 1) + " downto " +
"0)";
        }
        return "STD_LOGIC";
    }

    /**
     * Devuelve la lista de señales la entidad concatenadas con el
prefijo
     * "mi_" y separadas por comas. Se utilizará para hacer la
asignación
     * de puertos al declarar el componente.
     * @return Cadena con lista de señales de la entidad separadas por
comas
     */
    private String listaSenales() {
        String s = "(";
        for (int i = 0; i < entidad.getNumEntradas(); i++) {
            Entrada e = entidad.getEntrada(i);
            s += "mi_" + e.getNombre() + ",";
        }
        for (int i = 0; i < entidad.getNumSalidas() - 1; i++) {
            Salida sal = entidad.getSalida(i);
            s += "mi_" + sal.getNombre() + ",";
        }//la entidad tiene que tener al menos alguna salida
        s += "mi_" + entidad.getSalida(entidad.getNumSalidas() -
1).getNombre() + ")";
        return s;
    }

    /**
     * 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 del fichero que describe la
entidad top generada para cualquier circuto que quiera ser ejecutado
en la FPGA");
        escribirLinea("--");
        escribirLinea("--");
        escribirLinea("--Especificaciones:");
        escribirLinea("--\tCircuito a ejecutar:");
        escribirLinea("--\t\tNum. Entradas: "+
entidad.getBitsEntrada());
        escribirLinea("--\t\tNum. Salidas: " +
entidad.getBitsSalida());
        escribirLinea("--Autor:");
        escribirLinea("--\tCarlos Sanchez-Vellisco Sanchez");
        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("use IEEE.STD_LOGIC_ARITH.ALL;");
        escribirLinea("use IEEE.STD_LOGIC_UNSIGNED.ALL;");
        escribirLinea("");
        escribirLinea("");
    }

    /**
     * Descripción de la entidad
     */
    private void entidadGeneral() {
        escribirLinea("");
        escribirLinea("--Descripcion de la entidad");
        escribirLinea("entity " + nomEntidadGeneral + " is");
        escribirLinea("Port ( clk : in  STD_LOGIC;");
        escribirLinea("reset : in  STD_LOGIC;");
        escribirLinea("salida_serie : out  STD_LOGIC;");
        escribirLinea("entrada_serie : in  STD_LOGIC;");
        escribirLinea("ledsEntrada : out std_logic_vector(1 downto
0);");
        escribirLinea("ledsSalida : out std_logic_vector(1 downto
0));");
        escribirLinea("end " + nomEntidadGeneral + ";");
        escribirLinea("");
    }

    /**
     * Inicio de la arquitectura
     */
    private void inicioArquitectura() {
        escribirLinea("");
        escribirLinea("architecture Behavioral of " +
nomEntidadGeneral + " is");
```

```
        escribirLinea("");
    }

    /**
     * Declara el componente del transmisor
     */
    private void compSalidaSerie() {
        escribirLinea("");
        escribirLinea("--Transmisor serie");
        escribirLinea("component Tx_serie");
        escribirLinea("\tPort ( RstN : in  STD_LOGIC;");
        escribirLinea("\tclk : in  STD_LOGIC;");
        escribirLinea("\tTransmite : in  STD_LOGIC;");
        escribirLinea("\tDatoTxIn : in  STD_LOGIC_VECTOR (7 downto
0);");
        escribirLinea("\tTransmitiendo : out  STD_LOGIC;");
        escribirLinea("\tDatoSerieOut : out  STD_LOGIC);");
        escribirLinea("end component;");
        escribirLinea("");
    }

    /**
     * Declara el componente del receptor
     */
    private void compEntradaSerie() {
        escribirLinea("");
        escribirLinea("--Receptor Serie");
        escribirLinea("component Rx_Serie");
        escribirLinea("\tPort ( Rstn : in  STD_LOGIC;");
        escribirLinea("\tClk : in  STD_LOGIC;");
        escribirLinea("\tRxDatoSerie : in  STD_LOGIC;");
        escribirLinea("\tDatoRxOut : out  STD_LOGIC_VECTOR (7 downto
0);");
        escribirLinea("\tAvisoRx : out  STD_LOGIC;");
        escribirLinea("\tRecibiendo : out  STD_LOGIC);");
        escribirLinea("end component ;");
        escribirLinea("");
    }

    /**
     * Declara el componente del circuito principal
     */
    private void componentePrincipal() {
        escribirLinea("");
        escribirLinea("--Entidad que se quiere ejecutar");
        escribirLinea("component " + entidad.getNombre());
        escribirLinea("\tPort(");
        //Escribir las entradas
        for (int i = 0; i < entidad.getNumEntradas(); i++) {
            escribirLinea("\t\t" + entidad.getEntrada(i).getNombre() +
": in " + tipoPuerto(entidad.getEntrada(i)) + ";");
        }
        escribirLinea("");
        //Escribir las salidas
        for (int i = 0; i < entidad.getNumSalidas() - 1; i++) {
            escribirLinea("\t\t" + entidad.getSalida(i).getNombre() +
": out " + tipoPuerto(entidad.getSalida(i)) + ";");
        }
        escribirLinea("\t\t" +
entidad.getSalida(entidad.getNumSalidas() - 1).getNombre() + ": out "
+ tipoPuerto(entidad.getSalida(entidad.getNumSalidas() - 1)));
```

```
        escribirLinea(");");
        escribirLinea("end component;");
        escribirLinea("");
    }

    /**
     * Declara las señales del circuiti principal. Concatena el nombre
original
     * de la entrada o salida con el prefijo "mi_" de tal forma que
tengamos
     * señales intermedias
     */
    private void SenalesCircuitoPrincipal() {
        escribirLinea("");
        escribirLinea("--Señales del circuito principal");
        escribirLinea("");
        escribirLinea("--Entradas");
        for (int i = 0; i < entidad.getNumEntradas(); i++) {
            Entrada e = entidad.getEntrada(i);
            escribirLinea("signal mi_" + e.getNombre() + ": " +
tipoPuerto(e) + ";");
        }
        escribirLinea("");
        escribirLinea("--Salidas");
        for (int i = 0; i < entidad.getNumSalidas(); i++) {
            Salida s = entidad.getSalida(i);
            escribirLinea("signal mi_" + s.getNombre() + ": " +
tipoPuerto(s) + ";");
        }
        escribirLinea("");
    }

    /**
     * Declara señales intermedias para la entrada serie
     */
    private void SenalesEntradaSerie() {
        escribirLinea("");
        escribirLinea("--Señales para la Recepción Serie");
        escribirLinea("signal mi_resetserie:std_logic;");
        escribirLinea("signal mi_transmite:std_logic;");
        escribirLinea("signal mi_datotxin:std_logic_vector(7 downto
0);");
        escribirLinea("signal mi_datorxout:std_logic_vector(7 downto
0);");
        escribirLinea("signal mi_avisoRx:std_logic;");
        escribirLinea("signal mi_recibiendo:std_logic;");
        escribirLinea("");
    }

    /**
     * Declara señales intermedias para la salida serie
     */
    private void SenalesSalidaSerie() {
        escribirLinea("");
        escribirLinea("--Señales para la Transmisión Serie");
        escribirLinea("signal mi_transmitiendo:std_logic;");
        escribirLinea("signal mi_datoserieout:std_logic;");
        escribirLinea("signal mi_rxdatoserie:std_logic;");
        escribirLinea("");
    }
```

```
    /**
     * Declara las señales intermedias para conectar las entradas y
las salidas
     * Éstas se conectarán a su vez a los módulos de entrada/salida y
al
     * circuito principal
     */
    private void regsEntradaSalida() {
        escribirLinea("");
        escribirLinea("--Señales intermedias para la entrada y la
salida. Se conectarán a las entradas y las salidas del circuito
principal");
        escribirLinea("signal Reg_entradas: std_logic_vector(31 downto
0);");
        escribirLinea("signal Reg_salidas: std_logic_vector(31 downto
0);");
        escribirLinea("");
    }

    /**
     * Señales para visualizar los estados
     */
    private void SenalesEstados(){
        escribirLinea("");
        escribirLinea("--Señales para los estados del emisor/receptor
de 32 bits");
        escribirLinea("signal estadoEnt: integer;");
        escribirLinea("signal estadoSal: integer;");
        escribirLinea("");
        escribirLinea("signal ledsEnt:std_logic_vector(1 downto 0);");
        escribirLinea("signal ledsSal:std_logic_vector(1 downto 0);");
        escribirLinea("");
    }

    /**
     * Señales necesarias para la correcta entrada/salida con 32 bits
     */
    private void SenalesIO(){
        escribirLinea("");
        escribirLinea("--Señales necesarias para la correcta
entrada/salida con 32 bits");
        escribirLinea("signal fin_recepcion : std_logic;");
        escribirLinea("signal recibido,transmitido,biest_recibido,
biest_transmitido: std_logic;");
        escribirLinea("signal frecibido,ftransmitido,fin: std_logic;
--flancos de fin");
        escribirLinea("");

    }

    /**
     * Comienzo de la descripcion
     */
    private void begin() {
        escribirLinea("");
        escribirLinea("begin");
        escribirLinea("");
    }

    /**
     * Se asignan los puertos de los componentes de entrada/salida
```

```
     */
    private void asigSenalesCompsSerie() {
        escribirLinea("");
        escribirLinea("--Asignación de señales a los componentes de la
entrada/salida");
        escribirLinea("f: Tx_serie port
map(mi_resetserie,clk,mi_transmite,mi_datotxin,mi_transmitiendo,mi_dat
oserieout);");
        escribirLinea("R: Rx_serie port
map(mi_resetserie,clk,mi_rxdatoserie,mi_datorxout,mi_avisorx,mi_recibi
endo);");
        escribirLinea("");
    }

    /**
     * Se asignan las señales al componente del cricuito principal.
Utilizaremos
     * la lista de señales cobtenida más
     */
    private void asigSenalesCompPrinc() {
        escribirLinea("");
        escribirLinea("--Asignación de señales al componente del
circuito principal");
        escribirLinea("U: " + entidad.getNombre() + " port map" +
listaSenales() + ";");
        escribirLinea("");
    }

    /**
     * Proceso encargado de la correcta recepción de datos con 32 bits
     */
    private void procesoEntradas() {
        escribirLinea("");
        escribirLinea("--Proceso encargado de la correcta recepción de
datos (32 bits)");
        escribirLinea("--Cada vez que se reciba un byte, se irá
asignando a las entradas desde las menos significativas a las más
significatias");
        escribirLinea("process(mi_recibiendo,mi_resetserie)");
        escribirLinea("begin");
        escribirLinea("\tif mi_resetserie = '0' then");
        escribirLinea("\t\testadoEnt <= 0;");
        escribirLinea("\t\tfin_recepcion <= '0';");
        escribirLinea("\telsif mi_recibiendo'event and mi_recibiendo =
'0' then");
        escribirLinea("\t\tfin_recepcion <= '0';");
        escribirLinea("\t\tif estadoEnt = 0 then");
        escribirLinea("\t\t\tReg_entradas(7 downto 0) <=
mi_datorxout;");
        escribirLinea("\t\t\testadoEnt <= 1;");
        escribirLinea("\t\telsif estadoEnt = 1 then");
        escribirLinea("\t\t\tReg_entradas(15 downto 8) <=
mi_datorxout;");
        escribirLinea("\t\t\testadoEnt <= 2;");
        escribirLinea("\t\telsif estadoEnt = 2 then");
        escribirLinea("\t\t\tReg_entradas(23 downto 16) <=
mi_datorxout;");
        escribirLinea("\t\t\testadoEnt <= 3;");
        escribirLinea("\t\telsif estadoEnt = 3 then");
        escribirLinea("\t\t\tReg_entradas(31 downto 24) <=
mi_datorxout;");
```

```
        escribirLinea("\t\t\testadoEnt <= 0;");
        escribirLinea("\t\t\tfin_recepcion <= '1'; --fin de la
recepción");
        escribirLinea("\t\telse");
        escribirLinea("\t\t\tReg_entradas(7 downto 0) <=
mi_datorxout;");
        escribirLinea("\t\t\testadoEnt <= 1;");
        escribirLinea("\t\tend if;");
        escribirLinea("\tend if;");
        escribirLinea("end process;");
        escribirLinea("");

    }

    /**
     * Proceso encargado de que la salida correspondiente del circuito
esté conectada
     * a la salida serie antes de que la transmisión se produzca
     */
    private void procesoSalidas(){
        escribirLinea("");
        escribirLinea("--Proceso encargado de que la salida
correspondiente del circuito esté conectada a la salida serie antes de
que la transmisión se produzca");
        escribirLinea("process(estadoSal, clk, mi_resetserie)");
        escribirLinea("begin");
        escribirLinea("\tif mi_resetserie = '0' then");
        escribirLinea("\t\tmi_datotxin <= Reg_salidas(7 downto 0);");
        escribirLinea("\telsif clk'event and clk = '1' then");
        escribirLinea("\t\tif estadoSal = 0 then");
        escribirLinea("\t\t\tmi_datotxin <= Reg_salidas(7 downto
0);");
        escribirLinea("\t\telsif estadoSal = 1 then");
        escribirLinea("\t\t\tmi_datotxin <= Reg_salidas(15 downto
8);");
        escribirLinea("\t\telsif estadoSal = 2 then");
        escribirLinea("\t\t\tmi_datotxin <= Reg_salidas(23 downto
16);");
        escribirLinea("\t\telsif estadoSal = 3 then");
        escribirLinea("\t\t\tmi_datotxin <= Reg_salidas(31 downto
24);");
        escribirLinea("\t\tend if;");
        escribirLinea("\tend if;");
        escribirLinea("end process;");
        escribirLinea("");
    }

    private void procesoEstadoSalidas(){
        escribirLinea("");
        escribirLinea("--Proceso encargado de cambiar de estado cada
vez que se comienza a transmitir un byte");
        escribirLinea("process(mi_transmitiendo, mi_resetserie)");
        escribirLinea("begin");
        escribirLinea("\tif mi_resetserie = '0' then");
        escribirLinea("\t\testadoSal<= 0;");
        escribirLinea("\t\ttransmitido <= '0';");
        escribirLinea("\telsif mi_transmitiendo'event and
mi_transmitiendo = '1' then");
        escribirLinea("\t\tif estadoSal = 3 then");
        escribirLinea("\t\t\ttransmitido <= '1';");
        escribirLinea("\t\t\testadoSal <= 0;");
```

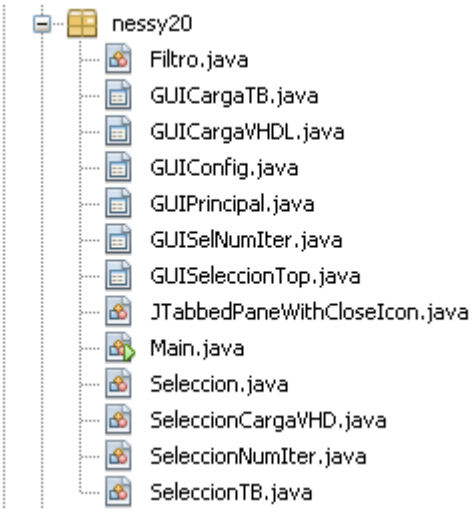
```
        escribirLinea("\t\telse");
        escribirLinea("\t\t\testadoSal <= estadoSal+1;");
        escribirLinea("\t\t\ttransmitido <= '0';");
        escribirLinea("\t\tend if;");
        escribirLinea("\tend if;");
        escribirLinea("end process;");
        escribirLinea("");
    }

    private void procesofin_recepcion(){
        escribirLinea("");
        escribirLinea("--Proceso encargado de registrar que la
recepción ha terminado. La salida del biestable ('recibido') se usará
como reloj del circuito principal");
        escribirLinea("process(clk,fin_recepcion)");
        escribirLinea("begin");
        escribirLinea("\tif clk'event and clk='1' then");
        escribirLinea("\t\tif fin_recepcion = '1' then");
        escribirLinea("\t\t\trecibido <= '1'; --flanco positivo que
hará funcionar el circuito principal");
        escribirLinea("\t\telse");
        escribirLinea("\t\t\trecibido <= '0';");
        escribirLinea("\t\tend if;");
        escribirLinea("\tend if;");
        escribirLinea("\tend process;");
        escribirLinea("");

    }

    private void procesoBiest(){
        escribirLinea("");
        escribirLinea("--Proceso encargado de registrar el fin de la
recepción y de la transmisión");
        escribirLinea("process(clk)");
        escribirLinea("begin");
        escribirLinea("\tif clk'event and clk='1' then");
        escribirLinea("\t\tbiest_recibido <= recibido;");
        escribirLinea("\t\tbiest_transmitido <= transmitido;");
        escribirLinea("\tend if;");
        escribirLinea("end process;");
        escribirLinea("");
    }

    private void procesoFin(){
        escribirLinea("");
        escribirLinea("--Proceso que indica que, o bien ha terminado
una recepción de datos, o bien ha terminado una transmisión");
        escribirLinea("--Cuando se detecte un flanco positivo, se
negará la señal que hace que se transmita, de tal forma que si se
estaba transitiendo,");
        escribirLinea("--se deje de transmitir y si no se estaba
transmitiendo se comience una nueva transmisión");
        escribirLinea("process(fin)");
        escribirLinea("begin");
        escribirLinea("\tif fin'event and fin = '1' then");
        escribirLinea("\t\tif mi_transmite = '0' then");
        escribirLinea("\t\t\tmi_transmite <= '1';");
        escribirLinea("\t\telse");
        escribirLinea("\t\t\tmi_transmite <= '0';");
        escribirLinea("\t\tend if;");
        escribirLinea("\tend if;");
```

```
        escribirLinea("end process;");
        escribirLinea("");
    }



    private void procesoLedsEnt(){
        escribirLinea("");
        escribirLinea("--Este proceso es prescindible. Sólo a efectos
de visualizar estado de entrada");
        escribirLinea("process(estadoEnt)");
        escribirLinea("begin");
        escribirLinea("\tif estadoEnt = 0 then");
        escribirLinea("\t\tledsEnt <= \"00\";");
        escribirLinea("\telsif estadoEnt = 1 then");
        escribirLinea("\t\tledsEnt <= \"01\";");
        escribirLinea("\telsif estadoEnt = 2 then");
        escribirLinea("\t\tledsEnt <= \"10\";");
        escribirLinea("elsif estadoEnt = 3 then");
        escribirLinea("\t\tledsEnt <= \"11\";");
        escribirLinea("\tend if;");
        escribirLinea("end process;");
        escribirLinea("");
    }

    private void procesoLedsSal(){
        escribirLinea("");
        escribirLinea("--Este proceso es prescindible. Sólo a efectos
de visualizar estado de salida");
        escribirLinea("process(estadoSal)");
        escribirLinea("begin");
        escribirLinea("\tif estadoSal = 0 then");
        escribirLinea("\t\tledsSal <= \"00\";");
        escribirLinea("\telsif estadoSal = 1 then");
        escribirLinea("\t\tledsSal <= \"01\";");
        escribirLinea("\telsif estadoSal = 2 then");
        escribirLinea("\t\tledsSal <= \"10\";");
        escribirLinea("elsif estadoSal = 3 then");
        escribirLinea("\t\tledsSal <= \"11\";");
        escribirLinea("\tend if;");
        escribirLinea("end process;");
        escribirLinea("");
    }

    private void asigLeds(){
        escribirLinea("");
        escribirLinea("ledsEntrada <= ledsEnt;");
        escribirLinea("ledsSalida <= ledsSal;");
        escribirLinea("");
    }

    private void entSalPuertoSerie() {
        escribirLinea("");
        if (this.hayReloj()){
            escribirLinea("--El reloj del circuito principal será el
flanco que indique el fin de la recepción");

escribirLinea("mi_"+entidad.getEntrada(entidad.getPosicionClk()).getNo
mbre()+ " <= recibido;");
        }
        escribirLinea("");
```

```
        escribirLinea("--Asignación de las señales del circuito
general");
        escribirLinea("mi_resetserie <= reset;");
        escribirLinea("salida_serie <= mi_datoserieout;");
        escribirLinea("mi_rxdatoserie <= entrada_serie;");
        escribirLinea("");
    }

    private void asigSenalesTransmision(){
        escribirLinea("");
        escribirLinea("--Asignación de las señales necesarias para la
transmisión correcta");
        escribirLinea("frecibido <= not biest_recibido and recibido; -
-flanco que indica fin de recepcion");
        escribirLinea("ftransmitido <= not biest_transmitido and
transmitido; --flanco que indica fin de transmision");
        escribirLinea("fin <= frecibido or ftransmitido; --flanco que
indica fin de envio o transmision");
        escribirLinea("");
    }

    private void asigEntradasCircuitoPrinc() {
        escribirLinea("");
        escribirLinea("--Asignación de las señales intermedias de
entrada/salida a las del circuito principal");
        int numEntrada = 0;
        for (int i = 0; i < entidad.getNumEntradas(); i++) {
            Entrada e = entidad.getEntrada(i);
            if (!e.getEsReloj()) {
                if (e.getNumBits() > 1) {
                    for (int j = 0; j < e.getNumBits(); j++) {
                        escribirLinea("mi_" + e.getNombre() + "(" + j
+ ") <= Reg_entradas(" + numEntrada++ + ");");
                    }

                } else {
                    escribirLinea("mi_" + e.getNombre() + " <=
Reg_entradas(" + numEntrada++ + ");");
                }
            }
        }
        escribirLinea("");
    }

    private void asigSalidasCircuitoPrinc() {
        escribirLinea("");
        int numSalida = 0;
        for (int i = 0; i < entidad.getNumSalidas(); i++) {
            Salida s = entidad.getSalida(i);
            if (s.getNumBits() > 1) {
                for (int j = 0; j < s.getNumBits(); j++) {
                    escribirLinea("Reg_salidas(" + numSalida++ + ") <=
mi_" + s.getNombre() + "(" + j + ");");
                }
            } else {
                escribirLinea("Reg_salidas(" + numSalida++ + ") <=
mi_" + s.getNombre() + ";");
            }
        }
        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();
        compSalidaSerie();
        compEntradaSerie();
        componentePrincipal();
        SenalesCircuitoPrincipal();
        SenalesEntradaSerie();
        SenalesSalidaSerie();
        regsEntradaSalida();
        SenalesEstados();
        SenalesIO();
        begin();
        asigSenalesCompsSerie();
        asigSenalesCompPrinc();
        procesoEntradas();
        procesoSalidas();
        procesoEstadoSalidas();
        procesofin_recepcion();
        procesoBiest();
        procesoFin();
        procesoLedsEnt();
        procesoLedsSal();
        asigLeds();
        entSalPuertoSerie();
        asigSenalesTransmision();
        asigEntradasCircuitoPrinc();
        asigSalidasCircuitoPrinc();
        end();
    }
}
```

Package Nessy20



Filtro.java

```
package nessy20;

/**
 *
 * @author Tony,David y Carlos.
 */

import javax.swing.filechooser.*;
import java.io.File;

/**
 * Clase que sirve para crear un filtro de extensiones a la hora de
mostrar
 * un selector de ficheros
 * @author David
 */
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; }
}
```

GUICargaTB.java

```
/*
 * GUICargaVHDL.java
 *
 * Created on 10 de mayo de 2010, 15:35
 */

package nessy20;

import javax.swing.JFrame;

/**
 * Interfaz gráfica para elegir de dónde se carga un test bench
 *
 * @author  David, Tony y Carlos
 */
public class GUICargaTB extends javax.swing.JDialog {

    /**
     * Tipo de seleccion que se hará. En este caso de TB
     */
    private Seleccion sel;

    /** Constructor del form GUICargaTB.
     * @param jf  Frame padre encargado de la llamada.
     * @param sel Tipo de Seleccion a elegir.
     * @param bol Indica si es modal o no.
     */
    public GUICargaTB(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();
        _btnCompilar = new javax.swing.JRadioButton();
        _btnCompModoTraza = new javax.swing.JRadioButton();
        _btnOK = new javax.swing.JButton();
        _btnCancelar = new javax.swing.JButton();

        setTitle("Cargar Test Bench");
        setIconImage(null);
        setResizable(false);

        _btnCompilar.setSelected(true);
        _btnCompilar.setText("Cargar en Pantalla");
        buttonGroup1.add(_btnCompilar);
        _btnCompilar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCompilarActionPerformed(evt);
            }
        });

        _btnCompModoTraza.setText("Cargar desde Fichero y Ejecutar");
        buttonGroup1.add(_btnCompModoTraza);

        _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(131, Short.MAX_VALUE)
                .addComponent(_btnOK,
javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGap(67, 67, 67)
                .addComponent(_btnCancelar)
                .addGap(88, 88, 88))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(124, 124, 124)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                    .addComponent(_btnCompilar)
                    .addComponent(_btnCompModoTraza))
                .addContainerGap(137, Short.MAX_VALUE))
        );

        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(_btnCompilar)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(_btnCompModoTraza)
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.BASELINE)
                    .addComponent(_btnCancelar)
                    .addComponent(_btnOK))
                .addContainerGap(22, 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)
            .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 _btnCompilarActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btnCompilarActionPerformed
```

```
// TODO add your handling code here:
}//GEN-LAST:event__btnCompilarActionPerformed

private void _btnOKActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnOKActionPerformed

    if(_btnCompilar.isSelected())
    {
        sel.selTB=SeleccionTB.CARGA_PANTALLA;
    }
    else
    {
        if(_btnCompModoTraza.isSelected())
        {
            sel.selTB=SeleccionTB.CARGA_FICHERO;
        }
        else
            sel.selTB=SeleccionTB.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.JRadioButton _btnCompModoTraza;
    private javax.swing.JRadioButton _btnCompilar;
    private javax.swing.JButton _btnOK;
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration//GEN-END:variables

}
```

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");
        setIconImage(null);
        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(97, 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(37, 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)
            .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 _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

}
```

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 rutaXilinx;
    private boolean cierre;
    /** Creates new form GUIConfig */
    public GUIConfig(java.awt.Frame parent, boolean modal,String ruta)
{
        super(parent, modal);
        rutaXilinx=ruta;
        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 = new javax.swing.JLabel();
        _txt_HomeXilinx = new javax.swing.JTextField();
        _btn_HomeXilinx = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        _btnOK = new javax.swing.JButton();
        _btnCancelar = 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.setText("Home Xilinx ISE :");

        _txt_HomeXilinx.setText(rutaXilinx);

        _btn_HomeXilinx.setText("Seleccion");
        _btn_HomeXilinx.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btn_HomeXilinxActionPerformed(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);
            }
        });

        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(31, 31, 31)
                .addComponent(_lbl_HomeXilinx,
javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(_txt_HomeXilinx,
javax.swing.GroupLayout.PREFERRED_SIZE, 330,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGap(12, 12, 12)
                .addComponent(_btn_HomeXilinx)
                .addContainerGap(32, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(229, Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addGap(220, 220, 220))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(182, Short.MAX_VALUE)
                .addComponent(_btnOK,
javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(67, 67, 67)
                .addComponent(_btnCancelar)
                .addGap(174, 174, 174))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(39, 39, 39)
                .addComponent(jLabel1)
                .addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                    .addComponent(_lbl_HomeXilinx)
                    .addComponent(_btn_HomeXilinx)
                    .addComponent(_txt_HomeXilinx,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
37, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                    .addComponent(_btnCancelar)
                    .addComponent(_btnOK))
                .addGap(31, 31, 31))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void
_btn_HomeXilinxActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btn_HomeXilinxActionPerformed
        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) {
            rutaXilinx=chooser.getSelectedFile().getAbsolutePath();
            _txt_HomeXilinx.setText(rutaXilinx);
```

```
        } else {
            //log.info("Seleccion no llevada a cabo");
        }

    }//GEN-LAST:event__btn_HomeXilinxActionPerformed

    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("HomeXilinx", rutaXilinx);
            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



    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton _btnCancelar;
    private javax.swing.JButton _btnOK;
    private javax.swing.JButton _btn_HomeXilinx;
    private javax.swing.JLabel _lbl_HomeXilinx;
    private javax.swing.JTextField _txt_HomeXilinx;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables

    boolean getCierre() {
      return cierre;

    }

}
```

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 cargarBit.CargaBit;
import compiladorEntidad.Errores;
import compiladorEntidad.SintacticoEntidad;



import IOFPGA.Ejecucion;
import IOFPGA.ReconfiguracionParcial;
import app.*;
//import com.sun.org.apache.bcel.internal.util.ClassPath;
import compiladorEntidad.Entidad;
import core.SerialPort;
import generadorVHDL.GeneraVhdl;
import java.io.BufferedReader;
import java.io.BufferedWriter;
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.OutputStream;
import java.util.ArrayList;
import java.util.Properties;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import org.apache.log4j.*;

/**
 * Frame principal de la aplicación
 * @author Tony, David y Carlos.
 */
public class GUIPrincipal extends javax.swing.JFrame {
    //private EstadoContador estado_cont;

    /**
     * Fichero de log de la aplicación
     */
    private static Logger log = Logger.getLogger(GUIPrincipal.class);
    /**
     * Ruta en la que se encuentran los ficheros necesarios para la
entrada/salida
     */
    private final static String RUTA_IOSERIE =
System.getProperties().getProperty("user.dir") + "\\IOSerie";

    /*
     * Ruta en la que se encuentra el Xilinx. Será introducida por el
usuario
     * o cargada mediante el fichero de configuración
```

```
     */
    private String RUTA_XILINX;
    /**
     * Parametros de configuración que se introducen al puerto serie
     */
    private Parameters param;
    /**
     * Interfaz para el manejo del puerto RS232
     */
    private Com com1;
    /**
     * Clase encargada de la ejecución de circuitos
     */
    private Ejecucion ejec;
    /**
     * Fichero VHD en el que está definida la entidad top que se desea
ejecutar.
     */
    private String ficheroEntidadVHD;
    /**
     * Representación interna de la entidad que se quiere ejecutar
     */
    private Entidad entidad;
    /**
     * Número de fichero que representa el top dentro del conjunto de
ficheros
     */
    private int top;
    /**
     * Conjunto de todos los ficheros VHD que pueden formar parte de
un proyecto
     * a la hora de generar un .BIT
     */
    private ArrayList<File> ficherosVHD;
    /**
     * Indica que el menú de selección top se ha cerrado
     */
    private boolean cerradoTop;

    /**
     * Indica la ruta fichero de banco de pruebas
     */
    String fichero_tb;

    /**
     * Indica la ruta del fichero .bit que se desea cargar
     */
    String fichero_bit;

    /**
     * Indica si, para ejecutar, la selección de un banco de pruebas
se obtiene
     * desde la pantalla o directamente desde un fichero.
     */
    private boolean SeleccionTBFich;

    /**
     * Lector de fichero para el banco de pruebas
     */
    private BufferedReader brTB;
    /**
```

```
     * Escritor de fichero para el log del proceso de reconfiguración
     */
    private FileWriter fwLog;

    /**
     * Proceso de reconfiguración
     */
    private ReconfiguracionParcial reconfiguracion;

    /**
     * Nos permite saber si estamos ejecutando inyeccion de errores o
no
     */
    private boolean inyeccErr;
     /**
     * Metodo accesor
     */
    public boolean isInyeccErr()
    {
        return inyeccErr;
    }
     /**
     * Metodo mutador
     */
    public void setInyeccErr(boolean inyeccErr)
    {
        this.inyeccErr = inyeccErr;
    }



    /** Constructor de la clase.
     *  Los botones reanudar y parar ejecución se ponen a no visibles.
     */
    public GUIPrincipal() {

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


PropertyConfigurator.configure("src/recursos/log4j.properties");

        config("conf/Config.properties", false);
        initComponentsAux();
        initComponents();
        this._btnReanudar.setEnabled(false);
        this._btnPararEjecucion.setEnabled(false);
        this.menuOpcionesReanudarEjec.setEnabled(false);
        this.menuOpcionesPararEjec.setEnabled(false);
        this.ficherosVHD = new ArrayList<File>();


log.info("=====================================================");
        log.info("Inicializado Nessy 2.0");
        _btnPararReconf.setEnabled(false);
        _btnPararReconf.setVisible(false);


    }
```

```
    /**
     * 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 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;
    }

    /**
     * Genera el archivo Golden.txt que será el fichero con el que
compararemos nuestras salidas.
     * @return Cierto si todo ha sido correcto, falso si ha habido
algún error.
     */
    public boolean generarGolden() {
        boolean correcto = true;
        if (this.ejec != null) {
            ejec.pararrecepcionfpga();
            this._TextSalida.setText("");
        }
        seleccionaPanel(panelOutPut);

        if (this.entidad != null) {//si la entidad está definida
            //System.out.println("Generando salida golden");
            log.info("Generando salida golden");
            if (SeleccionTBFich) {
                try {
                    brTB = new BufferedReader(new
FileReader(fichero_tb));
                } catch (FileNotFoundException ex) {
                    correcto = false;
                }
                this.ejec = new Ejecucion(this._lblnInst,
this.entidad, this.com1, this._TextSalida, brTB, false, "Golden.txt",
"Traza.txt", false);
```

```
                this.ejec.setCadena("");
                ejec.start();
                this._btnReanudar.setEnabled(false);
                this._btnPararEjecucion.setEnabled(true);
                this.menuOpcionesReanudarEjec.setEnabled(false);
                this.menuOpcionesPararEjec.setEnabled(true);
            } else {
                String ls_cadenaaejecutar = this._txtTB.getText();
                this.ejec = new Ejecucion(this._lblnInst,
this.entidad, this.com1, this._TextSalida, false, "Golden.txt",
"Traza.txt", false);
                this.ejec.setCadena(ls_cadenaaejecutar);
                if (ejec.convierteCadenas()) {
                    ejec.start();
                    //this.jTabbedPane1.setSelectedIndex(3);
                    this._btnReanudar.setEnabled(false);
                    this._btnPararEjecucion.setEnabled(true);
                    this.menuOpcionesReanudarEjec.setEnabled(false);
                    this.menuOpcionesPararEjec.setEnabled(true);
                } else {
                    JOptionPane.showMessageDialog(this, "Error en el
formato del banco de pruebas, revíselo por favor.\n" + "Sugerencia: se
deben pasar cadenas de bits 0's y 1's de longitud igual a " +
Integer.toString(this.getEntidad().getBitsEntrada()) + " .", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        } else {
            JOptionPane.showMessageDialog(this, "La entidad no está
definida", "Error", JOptionPane.ERROR_MESSAGE);
            correcto = false;
        }
        return correcto;
    }

    /**
     * Función para cargar el archivo .bit. Abre un JFileChooser para
elegir el
     * archivo a cargar. Tras la elección intenta cargar el fichero de
     * configuración en la FPGA.
     * @return Cierto si todo ha sido correcto, falso si ha habido
algún error.
     */
    public boolean cargarBitConChooser() {
        this._TextCargarbit.setText("Cargando ........." + "\n");
        boolean error = false;
        JFileChooser chooser;
        chooser = new JFileChooser();
        Filtro filter = new Filtro("bit");
        chooser.addChoosableFileFilter(filter);
        chooser.setCurrentDirectory(new java.io.File("."));
        chooser.setDialogTitle("Cargar BitStream");
        chooser.setAcceptAllFileFilterUsed(false);
        seleccionaPanel(panelCargar);
        if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
            if(inyeccErr)
                setEnabledBtnDetenerInyeccion(true);
            fichero_bit = chooser.getSelectedFile().getAbsolutePath();
            error = !this.cargarBit(fichero_bit, true);
        } else {
```

```
            setEnabledBtnDetenerInyeccion(false);
            System.out.println("Selecc ");
            this._TextCargarbit.setText("No ha seleccionado el .bit,
puede que si no lo ha cargado con anterioridad la aplicación no
funcione.");
            error = true;
        }
        return !error;
    }

    /**
     * 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;
    }

    /**
     * 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;
        SintacticoEntidad compilador = null;
        Errores errores = new Errores();
        GeneraVhdl generador;

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

            boolean error = compilador.Entidad();
            if (!error) {
                compilador.getEntidad().muestra();
                this.entidad = compilador.getEntidad();
                generador = new
GeneraVhdl("IOSerie//Circuito_FPGA.vhd", compilador.getEntidad(),
errores);
                if (generador.abrir()) {
                    generador.crearFichero();
                    generador.cerrar();
                    log.info("Fichero vhdl creado correctamente");
                } else {
                    this.muestraErroresConsola(errores);
                    correcto = false;
                }
            } 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;
    }

    /**
     * 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("COM1");
            param.setBaudRate("9600");
            SerialPort puerto = new SerialPort();
            if (puerto.getStateSerialPortC("COM1").equals("free")) {
                com1 = new Com(param);
            } else {
                JOptionPane.showMessageDialog(this, "El puerto COM1 no
se encuentra libre o " + "el PC no posee puerto COM1", "Info",
JOptionPane.INFORMATION_MESSAGE);
                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;
    }

    /**
     * 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 {
            log.error("Error al cargar VHDL");
```

```
            JOptionPane.showMessageDialog(this, "Error al cargar el
fichero de la entity", "Error", JOptionPane.ERROR_MESSAGE);
        }

    }

    /**
     * Carga un fichero VHD como top
     */
    private void cargaTopVHDL() {
        JFileChooser chooser;
        this._TxtEntityVHD.setText("");
        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) {
            deshabilitarBtnYmenu();
            ficherosVHD.add(chooser.getSelectedFile());
            ficheroEntidadVHD = ficherosVHD.get(0).getAbsolutePath();
            _lbl_VHDLCargado.setText("Ultimo Top VHDL cargado : "
                    + chooser.getSelectedFile().getName());
            this.cargarVHDL();
        } else {
            log.info("Seleccion no llevada a cabo");
        }
    }

    /**
     * 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) {
            deshabilitarBtnYmenu();
            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.setVisible(true);
            if (!cerradoTop) {
                ficheroEntidadVHD =
ficherosVHD.get(top).getAbsolutePath(); //el fichero es el absoluto
                _lbl_VHDLCargado.setText("Ultimo Top VHDL cargado : "
                        + ficherosVHD.get(top).getName());
                this.cargarVHDL();
            }

        }
    }

    /**
     * Copia un fichero en otro
     * @param fich_lectura Fichero a copiar
     * @param fich_escritura Fichero a crear
     */
    private void copiaArchivo(String fich_lectura, String
fich_escritura) {

        try {
            InputStream in;
            OutputStream out = new FileOutputStream(fich_escritura);

            byte[] buf = new byte[1024];
            int len;

            in = new FileInputStream(fich_lectura);

            while ((len = in.read(buf)) > 0) {
                out.write(buf, 0, len);
            }
            in.close();
            out.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    /**
     * Crea el fichero (.PRJ)necesario para generar un .BIT con los
comandos de
     * Xilinx. En concreto este fichero es necesario para el proceso
de síntesis
     * implementado en XST de Xilinx.
     */
    private void creaPrj() {
        BufferedWriter bw = null;
        try {
            bw = new BufferedWriter(new
FileWriter("IOSerie//Circuito_FPGA.prj"));
            bw.write("vhdl work \"Tx_serie.vhd\"\n");
            bw.write("vhdl work \"Rx_serie.vhd\"\n");
            for (int i = 0; i < ficherosVHD.size(); i++) {
                bw.write("vhdl work \"" +
ficherosVHD.get(i).getAbsolutePath() + "\"\n");
            }
            bw.write("vhdl work \"Circuito_FPGA.vhd\"");
            bw.close();
        } catch (IOException ex) {
            ex.printStackTrace();
```

```
        }
    }

    /**
     * Realiza la ejecución de un circuito. Para ello lee las entradas
del banco
     * de pruebas, las cuales pueden ser ofrecidas desde la pantalla
de la
     * aplicación o directamente desde fichero. A continuación
comprueba que
     * el formato de las entradas es correcto (están compuestas por
0's y 1's
     * de la misma longitud que el número de entradas que la entidad
cargada).
     * Si la ejecución actual no pertence a uno de los pasos de la
reconfiguración
     * parcial, entonces por cada ejecución se lanzará un nuevo hilo,
de tal forma
     * que podamos visualizar la salida a medida que se va
produciendo. Además
     * este hilo podrá ser detenido en cualquier momento y reanudado
más adelante.
     * Si por el contrario sí pertenece a la reconfiguración parcial,
no queremos
     * que nada más ocurra mientras se está ejecutando, para evitar
así que por
     * ejemplo se cargue el próximo .BIT mientras se está aun
ejecutando con el
     * anterior. Por lo tanto no se lanzará un nuevo hilo sino que se
hará de
     * forma secuencial.
     * @param lb_reconfiguracionParcial Indica si pertence a un paso
de la
     * reconfiguración parcial
     */
    public boolean ejec(boolean lb_reconfiguracionParcial) {

        if (this.ejec != null) {
            ejec.pararrecepcionfpga();
            this._TextSalida.setText("");
        }
        //Selecciona panel
        seleccionaPanel(panelOutPut);

        if (this.entidad != null) {//si la entidad está definida
            if (SeleccionTBFich) {
                try {
                    brTB = new BufferedReader(new
FileReader(fichero_tb));

                } catch (FileNotFoundException ex) {
                }
                this.ejec = new Ejecucion(this._lblnInst,
this.entidad, this.com1, this._TextSalida, brTB, true, "Salida.txt",
"Golden.txt", lb_reconfiguracionParcial);
                if (ejec.formatoCorrectoFicheroTB(fichero_tb)) {
                    this.ejec.setCadena("");
                    if (!lb_reconfiguracionParcial) {
                        ejec.start();
                    } else {
                        ejec.setFileLogEjec(fwLog);
```

```
                        //ejec.ejecuta();
                        ejec.start();
                        try {
                            ejec.join();
                        } catch (InterruptedException ex) {

java.util.logging.Logger.getLogger(GUIPrincipal.class.getName()).log(j
ava.util.logging.Level.SEVERE, null, ex);
                        }
                    }
                    this._btnReanudar.setEnabled(false);
                    this._btnPararEjecucion.setEnabled(true);
                    this.menuOpcionesReanudarEjec.setEnabled(false);
                    this.menuOpcionesPararEjec.setEnabled(true);
                    return true;

                } else {
                    JOptionPane.showMessageDialog(this, "Error en el
formato del banco de pruebas, revíselo por favor.\n" + "Sugerencia: se
deben pasar cadenas de bits 0's y 1's de longitud igual a " +
Integer.toString(this.getEntidad().getBitsEntrada()) + " .", "Error",
JOptionPane.ERROR_MESSAGE);
                    return false;
                }
            } else {
                String ls_cadenaaejecutar = this._txtTB.getText();
                this.ejec = new Ejecucion(this._lblnInst,
this.entidad, this.com1, this._TextSalida, true, "Salida.txt",
"Golden.txt", lb_reconfiguracionParcial);
                this.ejec.setCadena(ls_cadenaaejecutar);
                if (ejec.convierteCadenas()) {
                    if (!lb_reconfiguracionParcial) {
                        ejec.start();
                    }
                    this._btnReanudar.setEnabled(false);
                    this._btnPararEjecucion.setEnabled(true);
                    this.menuOpcionesReanudarEjec.setEnabled(false);
                    this.menuOpcionesPararEjec.setEnabled(true);
                    return true;
                } else {
                    JOptionPane.showMessageDialog(this, "Error en el
formato del banco de pruebas, revíselo por favor.\n" + "Sugerencia: se
deben pasar cadenas de bits 0's y 1's de longitud igual a " +
Integer.toString(this.getEntidad().getBitsEntrada()) + " .", "Error",
JOptionPane.ERROR_MESSAGE);
                    return false;
                }
            }
        } else {
            JOptionPane.showMessageDialog(this, "La entidad no está
definida", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

    }

    /**
     * Establece el fichero de escritura para el fichero de log del
proceso de
     * reconfiguracion
     * @param fw
```

```
     */
    public void setFwLog(FileWriter fw) {
        this.fwLog = fw;
    }

    private void initComponentsAux() {
        jTabbedPane1 = new JTabbedPaneWithCloseIcon();
    }


    /**
     * Habilita o deshabilita el boton Detener Inyeccion de errores.
     */
    public void setEnabledBtnDetenerInyeccion(boolean b)
    {
            _btnPararReconf.setEnabled(b);
            _btnPararReconf.setVisible(b);
    }

    /**
     * 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");
        }
    }


    /**
     * Escribe una cadena que se le pasa como argumento en el text
Area correspondiente al proceso
     * de cargar un archivo .bit en la FPGA
     * @param str Cadena a escribir.
     */
    public void escribirEnPantalla(String str) {
        this._TextCargarbit.append(str + "\n");
    }

    /**
     * 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, boolean
ab_mostrar_mensajes) {
        boolean error = false;
        int intentos = 2;
        CargaBit cargaBit = new CargaBit(this, fichero_bit,
this.RUTA_XILINX + "\\ISE\\bin\\nt\\impact.exe");
        try {
            do {//si hay un error lo vuelve a intentar
                error = !cargaBit.cargar(ab_mostrar_mensajes);
                if (!error) {
                    if (ab_mostrar_mensajes) {
                        JOptionPane.showMessageDialog(this, "Bitstream
cargado correctamente", "Información",
JOptionPane.INFORMATION_MESSAGE);
                    }
                    if (com1 != null) {
                        com1.close();
                        com1 = null;
                    }
                } else {
                    intentos--;
                }
            } while (error && intentos > 0);//intenta cargar 6 veces
el .bit
        } catch (Exception e) {
            error = true;
        }
        if (!error) {
            _lbl_BitCargado.setText("Ultimo Archivo .BIT cargado : "
                    + fichero_bit);
        }
        return !error;
    }

    /** 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() {

        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();
        _btnEjecutar = new javax.swing.JButton();
        _btnPararEjecucion = new javax.swing.JButton();
        _btnReanudar = new javax.swing.JButton();
        _btnGenerarGolden = new javax.swing.JButton();
        _btnCargarGolden = new javax.swing.JButton();
        _btnCargBitReconfParcial = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        _btnClear = new javax.swing.JButton();
        _lblnInst = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jTabbedPane1 = new JTabbedPaneWithCloseIcon();
```

```
        panelVHD = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        _TxtEntityVHD = new javax.swing.JTextArea();
        panelCargar = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        _TextCargarbit = new javax.swing.JTextArea();
        panelTB = new javax.swing.JPanel();
        jScrollPane3 = new javax.swing.JScrollPane();
        _txtTB = new javax.swing.JTextArea();
        panelOutPut = new javax.swing.JPanel();
        jScrollPane4 = new javax.swing.JScrollPane();
        _TextSalida = new javax.swing.JTextArea();
        _lbl_BitCargado = new javax.swing.JLabel();
        _lbl_VHDLCargado = new javax.swing.JLabel();
        _btnPararReconf = new javax.swing.JButton();
        jSeparator1 = new javax.swing.JSeparator();
        jMenuBar1 = new javax.swing.JMenuBar();
        menuOpciones = new javax.swing.JMenu();
        menuOpcionesCargarVHD = new javax.swing.JMenuItem();
        menuOpcionesCrearBit = new javax.swing.JMenuItem();
        menuOpcionesCargarBit = new javax.swing.JMenuItem();
        menuOpcionesCargarTB = new javax.swing.JMenuItem();
        menuOpcionesEjec = new javax.swing.JMenuItem();
        menuOpcionesPararEjec = new javax.swing.JMenuItem();
        menuOpcionesReanudarEjec = new javax.swing.JMenuItem();
        menuOpcionesGeneraGolden = new javax.swing.JMenuItem();
        menuOpcionesCargarGolden = new javax.swing.JMenuItem();
        menuOpcionesReconfParcial = new javax.swing.JMenuItem();
        menuVistas = new javax.swing.JMenu();
        menuVistasEntityVHD = new javax.swing.JMenuItem();
        menuVistasCargar = new javax.swing.JMenuItem();
        menuVistasTB = new javax.swing.JMenuItem();
        menuVistasOutPut = new javax.swing.JMenuItem();
        menuConfig = new javax.swing.JMenu();
        menuConfigNessy = new javax.swing.JMenuItem();
        menuConfigFichConf = new javax.swing.JMenuItem();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Nessy 2.0");
        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("Menu")
);
        jPanel2.setOpaque(false);

        jToolBar1.setBorder(null);
        jToolBar1.setRollover(true);
```

```
        _btnCargarVhd.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnCargarVhdl.
png"))); // NOI18N
        _btnCargarVhd.setText("Cargar VHD");
        _btnCargarVhd.setFocusable(false);

_btnCargarVhd.setHorizontalTextPosition(javax.swing.SwingConstants.CEN
TER);

_btnCargarVhd.setVerticalTextPosition(javax.swing.SwingConstants.BOTTO
M);
        _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 .Bit");
        _btnCrearBit.setEnabled(false);
        _btnCrearBit.setFocusable(false);

_btnCrearBit.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);

_btnCrearBit.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
);
        _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 .Bit");
        _btnCargarBit.setEnabled(false);
        _btnCargarBit.setFocusable(false);

_btnCargarBit.setHorizontalTextPosition(javax.swing.SwingConstants.CEN
TER);

_btnCargarBit.setVerticalTextPosition(javax.swing.SwingConstants.BOTTO
M);
        _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 Test Bench");
        _btnCargarTB.setEnabled(false);
        _btnCargarTB.setFocusable(false);

_btnCargarTB.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);

_btnCargarTB.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
);
        _btnCargarTB.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCargarTBActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnCargarTB);

        _btnEjecutar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnEjec.png"))
); // NOI18N
        _btnEjecutar.setText("Ejecutar");
        _btnEjecutar.setContentAreaFilled(false);
        _btnEjecutar.setEnabled(false);
        _btnEjecutar.setFocusable(false);

_btnEjecutar.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);

_btnEjecutar.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
);
        _btnEjecutar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnEjecutarActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnEjecutar);

        _btnPararEjecucion.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnPararEjec.p
ng"))); // NOI18N
        _btnPararEjecucion.setText("Parar Ejecución");
        _btnPararEjecucion.setEnabled(false);
        _btnPararEjecucion.setFocusable(false);

_btnPararEjecucion.setHorizontalTextPosition(javax.swing.SwingConstant
s.CENTER);

_btnPararEjecucion.setVerticalTextPosition(javax.swing.SwingConstants.
BOTTOM);
        _btnPararEjecucion.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnPararEjecucionActionPerformed(evt);
```

```
            }
        });
        jToolBar1.add(_btnPararEjecucion);

        _btnReanudar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnReanudarEje
c.png"))); // NOI18N
        _btnReanudar.setText("Reanudar Ejecución");
        _btnReanudar.setEnabled(false);
        _btnReanudar.setFocusable(false);

_btnReanudar.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);

_btnReanudar.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
);
        _btnReanudar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnReanudarActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnReanudar);

        _btnGenerarGolden.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/btnGeneraGolde
n.png"))); // NOI18N
        _btnGenerarGolden.setText("Generar Golden");
        _btnGenerarGolden.setEnabled(false);
        _btnGenerarGolden.setFocusable(false);

_btnGenerarGolden.setHorizontalTextPosition(javax.swing.SwingConstants
.CENTER);

_btnGenerarGolden.setVerticalTextPosition(javax.swing.SwingConstants.B
OTTOM);
        _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");
        _btnCargarGolden.setEnabled(false);
        _btnCargarGolden.setFocusable(false);

_btnCargarGolden.setHorizontalTextPosition(javax.swing.SwingConstants.
CENTER);

_btnCargarGolden.setVerticalTextPosition(javax.swing.SwingConstants.BO
TTOM);
        _btnCargarGolden.addActionListener(new
java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCargarGoldenActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnCargarGolden);

        _btnCargBitReconfParcial.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/reconParc.JPG"
))); // NOI18N
        _btnCargBitReconfParcial.setText("Inyección de Errores");
        _btnCargBitReconfParcial.setEnabled(false);
        _btnCargBitReconfParcial.setFocusable(false);

_btnCargBitReconfParcial.setHorizontalTextPosition(javax.swing.SwingCo
nstants.CENTER);

_btnCargBitReconfParcial.setVerticalTextPosition(javax.swing.SwingCons
tants.BOTTOM);
        _btnCargBitReconfParcial.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnCargBitReconfParcialActionPerformed(evt);
            }
        });
        jToolBar1.add(_btnCargBitReconfParcial);

        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, 862,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(62, Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
                .addComponent(jToolBar1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(8, 8, 8))
        );


jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Pantal
la"));
        jPanel1.setOpaque(false);

        _btnClear.setText("Limpiar Pantalla");
        _btnClear.setAutoscrolls(true);
```

```
        _btnClear.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                _btnClearActionPerformed(evt);
            }
        });

        _lblnInst.setEditable(false);

        jLabel1.setText("Ciclo de Ejecución");
        jLabel1.setAutoscrolls(true);

        _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.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 831, Short.MAX_VALUE)
        );
        panelVHDLayout.setVerticalGroup(

panelVHDLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 294, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Entity VHDL", panelVHD);

        _TextCargarbit.setColumns(20);
        _TextCargarbit.setRows(5);
        _TextCargarbit.setMaximumSize(getMaximumSize());
        jScrollPane2.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(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 831, Short.MAX_VALUE)
        );
        panelCargarLayout.setVerticalGroup(

panelCargarLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 294, Short.MAX_VALUE)
```

```
        );

        jTabbedPane1.addTab("Cargar", panelCargar);

        _txtTB.setColumns(20);
        _txtTB.setRows(5);
        jScrollPane3.setViewportView(_txtTB);

        javax.swing.GroupLayout panelTBLayout = new
javax.swing.GroupLayout(panelTB);
        panelTB.setLayout(panelTBLayout);
        panelTBLayout.setHorizontalGroup(

panelTBLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 831, Short.MAX_VALUE)
        );
        panelTBLayout.setVerticalGroup(

panelTBLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 294, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("Test Bench", panelTB);

        _TextSalida.setColumns(20);
        _TextSalida.setEditable(false);
        _TextSalida.setRows(5);
        jScrollPane4.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(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 831, Short.MAX_VALUE)
        );
        panelOutPutLayout.setVerticalGroup(

panelOutPutLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addComponent(jScrollPane4,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 294, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab("OutPut", panelOutPut);

        _btnPararReconf.setText("Detener Inyección de Errores");
        _btnPararReconf.setEnabled(false);
        _btnPararReconf.setVisible(false);
        _btnPararReconf.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
```

```
                _btnPararReconfActionPerformed(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()
                .addGap(20, 20, 20)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                    .addComponent(_lbl_BitCargado,
javax.swing.GroupLayout.DEFAULT_SIZE, 435, Short.MAX_VALUE)
                    .addComponent(_lbl_VHDLCargado,
javax.swing.GroupLayout.PREFERRED_SIZE, 212,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(469, 469, 469))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGap(217, 217, 217)
                .addComponent(_btnPararReconf)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
281, Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addGap(18, 18, 18)
                .addComponent(_lblnInst,
javax.swing.GroupLayout.PREFERRED_SIZE, 72,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(78, 78, 78))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 836,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap())
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                    .addContainerGap(433, Short.MAX_VALUE)
                    .addComponent(_btnClear,
javax.swing.GroupLayout.PREFERRED_SIZE, 190,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(301, 301, 301)))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap()
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.BASELINE)
                    .addComponent(_lblnInst,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1)
                    .addComponent(_btnParar Reconf))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
373, Short.MAX_VALUE)
                .addComponent(_lbl_VHDLCargado,
javax.swing.GroupLayout.PREFERRED_SIZE, 13,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(_lbl_BitCargado,
javax.swing.GroupLayout.PREFERRED_SIZE, 13,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(6, 6, 6))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(_btnClear)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 322,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap()))
        );

        jTabbedPane1.getAccessibleContext().setAccessibleName("");

        menuOpciones.setText("Opciones");


menuOpcionesCargarVHD.setAccelerator(javax.swing.KeyStroke.getKeyStrok
e(java.awt.event.KeyEvent.VK_V, java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesCargarVHD.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuCargarVhdl
.png"))); // NOI18N
        menuOpcionesCargarVHD.setText("Cargar VHD");
        menuOpcionesCargarVHD.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesCargarVHDActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesCargarVHD);


menuOpcionesCrearBit.setAccelerator(javax.swing.KeyStroke.getKeyStroke
(java.awt.event.KeyEvent.VK_R, java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesCrearBit.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuCrearBit.p
ng"))); // NOI18N
```

```
        menuOpcionesCrearBit.setText("Crear .Bit");
        menuOpcionesCrearBit.setEnabled(false);
        menuOpcionesCrearBit.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesCrearBitActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesCrearBit);


menuOpcionesCargarBit.setAccelerator(javax.swing.KeyStroke.getKeyStrok
e(java.awt.event.KeyEvent.VK_B, java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesCargarBit.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuCargarBit.
png"))); // NOI18N
        menuOpcionesCargarBit.setText("Cargar .Bit");
        menuOpcionesCargarBit.setEnabled(false);
        menuOpcionesCargarBit.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesCargarBitActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesCargarBit);


menuOpcionesCargarTB.setAccelerator(javax.swing.KeyStroke.getKeyStroke
(java.awt.event.KeyEvent.VK_T, java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesCargarTB.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuCargarTB.p
ng"))); // NOI18N
        menuOpcionesCargarTB.setText("Cargar Test Bench");
        menuOpcionesCargarTB.setEnabled(false);
        menuOpcionesCargarTB.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesCargarTBActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesCargarTB);


menuOpcionesEjec.setAccelerator(javax.swing.KeyStroke.getKeyStroke(jav
a.awt.event.KeyEvent.VK_E, java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesEjec.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuEjec.png")
)); // NOI18N
        menuOpcionesEjec.setText("Ejecutar");
        menuOpcionesEjec.setEnabled(false);
        menuOpcionesEjec.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesEjecActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesEjec);
```

```
menuOpcionesPararEjec.setAccelerator(javax.swing.KeyStroke.getKeyStrok
e(java.awt.event.KeyEvent.VK_S, java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesPararEjec.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuPararEjec.
png"))); // NOI18N
        menuOpcionesPararEjec.setText("Parar Ejecucion");
        menuOpcionesPararEjec.setEnabled(false);
        menuOpcionesPararEjec.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesPararEjecActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesPararEjec);


menuOpcionesReanudarEjec.setAccelerator(javax.swing.KeyStroke.getKeySt
roke(java.awt.event.KeyEvent.VK_I,
java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesReanudarEjec.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuReanudarEj
ec.png"))); // NOI18N
        menuOpcionesReanudarEjec.setText("Reanudar Ejecucion");
        menuOpcionesReanudarEjec.setEnabled(false);
        menuOpcionesReanudarEjec.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesReanudarEjecActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesReanudarEjec);


menuOpcionesGeneraGolden.setAccelerator(javax.swing.KeyStroke.getKeySt
roke(java.awt.event.KeyEvent.VK_G,
java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesGeneraGolden.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuGeneraGold
en.png"))); // NOI18N
        menuOpcionesGeneraGolden.setText("Generar Golden");
        menuOpcionesGeneraGolden.setEnabled(false);
        menuOpcionesGeneraGolden.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesGeneraGoldenActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesGeneraGolden);


menuOpcionesCargarGolden.setAccelerator(javax.swing.KeyStroke.getKeySt
roke(java.awt.event.KeyEvent.VK_C,
java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesCargarGolden.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menuCargaGolde
n.jpg"))); // NOI18N
```

```
        menuOpcionesCargarGolden.setText("Cargar Golden");
        menuOpcionesCargarGolden.setEnabled(false);
        menuOpcionesCargarGolden.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesCargarGoldenActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesCargarGolden);


menuOpcionesReconfParcial.setAccelerator(javax.swing.KeyStroke.getKeyS
troke(java.awt.event.KeyEvent.VK_P,
java.awt.event.InputEvent.CTRL_MASK));
        menuOpcionesReconfParcial.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/recursos/menureconParc.
JPG"))); // NOI18N
        menuOpcionesReconfParcial.setText("Inyección de Errores");
        menuOpcionesReconfParcial.setEnabled(false);
        menuOpcionesReconfParcial.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuOpcionesReconfParcialActionPerformed(evt);
            }
        });
        menuOpciones.add(menuOpcionesReconfParcial);

        jMenuBar1.add(menuOpciones);

        menuVistas.setText("Vistas");


menuVistasEntityVHD.setAccelerator(javax.swing.KeyStroke.getKeyStroke(
java.awt.event.KeyEvent.VK_V, java.awt.event.InputEvent.ALT_MASK));
        menuVistasEntityVHD.setText("Entity VHD");
        menuVistasEntityVHD.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuVistasEntityVHDActionPerformed(evt);
            }
        });
        menuVistas.add(menuVistasEntityVHD);


menuVistasCargar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(jav
a.awt.event.KeyEvent.VK_C, java.awt.event.InputEvent.ALT_MASK));
        menuVistasCargar.setText("Cargar");
        menuVistasCargar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuVistasCargarActionPerformed(evt);
            }
        });
        menuVistas.add(menuVistasCargar);
```

```
menuVistasTB.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.aw
t.event.KeyEvent.VK_T, java.awt.event.InputEvent.ALT_MASK));
        menuVistasTB.setText("Test Bench");
        menuVistasTB.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuVistasTBActionPerformed(evt);
            }
        });
        menuVistas.add(menuVistasTB);


menuVistasOutPut.setAccelerator(javax.swing.KeyStroke.getKeyStroke(jav
a.awt.event.KeyEvent.VK_O, java.awt.event.InputEvent.ALT_MASK));
        menuVistasOutPut.setText("OutPut");
        menuVistasOutPut.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuVistasOutPutActionPerformed(evt);
            }
        });
        menuVistas.add(menuVistasOutPut);

        jMenuBar1.add(menuVistas);

        menuConfig.setText("Configuración");

        menuConfigNessy.setText("Configurar Nessy");
        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");
        menuConfigFichConf.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                menuConfigFichConfActionPerformed(evt);
            }
        });
        menuConfig.add(menuConfigFichConf);

        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)
```

```
            .addComponent(jPanel2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jSeparator1,
javax.swing.GroupLayout.DEFAULT_SIZE, 940, Short.MAX_VALUE)
            .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void
_btnCargarVhdActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnCargarVhdActionPerformed


        log.info("Cargando archivo VDHL...");
        Seleccion sel = new Seleccion();
        new GUICargaVHDL(this, true, sel).setVisible(true);
        if
(sel.seleccion.equals(SeleccionCargaVHD.SELECCION_VHDL_TOP)) {
            cargaTopVHDL();

            _btnCrearBit.setEnabled(true);
            _btnCargarBit.setEnabled(true);
            _btnCargarTB.setEnabled(true);
            _btnCargBitReconfParcial.setEnabled(true);
            this.menuOpcionesCrearBit.setEnabled(true);
            this.menuOpcionesCargarBit.setEnabled(true);
            this.menuOpcionesCargarTB.setEnabled(true);
            this.menuOpcionesReconfParcial.setEnabled(true);

        } else {
            if
(sel.seleccion.equals(SeleccionCargaVHD.SELECCION_VARIOS_VHDL)) {
                cargaVariosVHDL();
                _btnCrearBit.setEnabled(true);
                _btnCargarBit.setEnabled(true);
                _btnCargarTB.setEnabled(true);
                _btnCargBitReconfParcial.setEnabled(true);
```

```
                this.menuOpcionesCrearBit.setEnabled(true);
                this.menuOpcionesCargarBit.setEnabled(true);
                this.menuOpcionesCargarTB.setEnabled(true);
                this.menuOpcionesReconfParcial.setEnabled(true);
            }

            else
            {
                if (sel.seleccion.equals(SeleccionCargaVHD.NADA))
                {}

            }
        }

    }//GEN-LAST:event__btnCargarVhdActionPerformed

    private void
_btnCrearBitActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnCrearBitActionPerformed
        try {
            JFileChooser chooser;
            chooser = new JFileChooser();
            Filtro filter = new Filtro("bit");
            chooser.addChoosableFileFilter(filter);
            chooser.setCurrentDirectory(new java.io.File("."));
            chooser.setDialogTitle("Guardar el .BIT");
            chooser.setAcceptAllFileFilterUsed(false);
            String rutaDestino;
            if (chooser.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
                rutaDestino =
chooser.getSelectedFile().getAbsolutePath();
                if (rutaDestino.lastIndexOf(".bit") + 4 !=
rutaDestino.length()) {
                    rutaDestino = rutaDestino + ".bit";
                }
                //creamos el prj para poder crear el .bit
                this.creaPrj();
                //compilación y creación del .bit
                Process p = Runtime.getRuntime().exec("cmd.exe /C
start comandosXilinx\\compilar.bat " + this.RUTA_XILINX + " " +
rutaDestino);
                  log.info("Se ha creado .bit");
                //Process copiar = Runtime.getRuntime().exec("cmd.exe
/C start comandosXilinx\\copiararchivo.bat " + this.RUTA_XILINX);
            }
        } catch (IOException ex) {
            log.error("Error creando .BIT", ex);
        }

    }//GEN-LAST:event__btnCrearBitActionPerformed

    private void
_btnCargarBitActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnCargarBitActionPerformed

        //POsibilidad de ver si se carga con exito
        log.info("Cargando .BIT...");
        if (this.cargarBitConChooser()) {
            log.info("Cargar .BIT : Cargado archivo .bit
correctamente");
```

```
        } else {

            log.warn("Cargar .BIT : No se ha podido Cargar el archivo
.bit correctamente");
            JOptionPane.showMessageDialog(this, "No se ha podido
Cargar el archivo" +
                    " .bit correctamente", "Error",
JOptionPane.ERROR_MESSAGE);
        }

    }//GEN-LAST:event__btnCargarBitActionPerformed

    private void
_btnEjecutarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnEjecutarActionPerformed
        log.info("Ejecutando ...");
        if (this.com1 == null) {
            if (this.inicializarPuertoSerie()) {
                ejec(false);
            }
        } else {
            log.error("Puerto COM1 no conectado");
            ejec(false);
        }
    }//GEN-LAST:event__btnEjecutarActionPerformed

    private void
_btnPararEjecucionActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnPararEjecucionActionPerformed

        seleccionaPanel(panelOutPut);
        log.info("Ejecucion Parada");
        int longitud = this.entidad.getBitsEntrada();
        System.out.println("PARANDO EL HILO..");
        this.ejec.setSetwait(true);
        this._btnReanudar.setEnabled(true);
        this._btnPararEjecucion.setEnabled(false);
        this.menuOpcionesReanudarEjec.setEnabled(true);
        this.menuOpcionesPararEjec.setEnabled(false);

    }//GEN-LAST:event__btnPararEjecucionActionPerformed

    private void
_btnReanudarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnReanudarActionPerformed

        seleccionaPanel(panelOutPut);
        log.info("Reanudando Ejecución");
        synchronized (this.ejec) {
            this.ejec.notify();
        }
        this._btnReanudar.setEnabled(false);
        this._btnPararEjecucion.setEnabled(true);
        this.menuOpcionesReanudarEjec.setEnabled(false);
        this.menuOpcionesPararEjec.setEnabled(true);

    }//GEN-LAST:event__btnReanudarActionPerformed

    private void
_btnCargarTBActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event__btnCargarTBActionPerformed
```

```
        log.info("Cargando archivo TestBench...");
        Seleccion sel = new Seleccion();
        new GUICargaTB(this, true, sel).setVisible(true);
        if (sel.selTB.equals(SeleccionTB.CARGA_FICHERO)) {
            SeleccionTBFich = true;
            cargaFicheroTB();
            if (this.com1 == null) {
                if (this.inicializarPuertoSerie()) {
                    if(ejec(false))
                        habilitarBtnsCargarTB();
                }
            } else {
                if(ejec(false))
                    habilitarBtnsCargarTB();
            }


        } else {
            if (sel.selTB.equals(SeleccionTB.CARGA_PANTALLA)) {
                SeleccionTBFich = false;
                cargarTextArea();
                habilitarBtnsCargarTB();
            }
        }


    }//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
        ejec.pararrecepcionfpga();

    }//GEN-LAST:event_formWindowClosed

    private void
menuOpcionesCargarVHDActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuOpcionesCargarVHDActionPerformed
        _btnCargarVhdActionPerformed(evt);
    }//GEN-LAST:event_menuOpcionesCargarVHDActionPerformed

    private void
menuOpcionesCrearBitActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuOpcionesCrearBitActionPerformed
        _btnCrearBitActionPerformed(evt);
```

```
    }//GEN-LAST:event_menuOpcionesCrearBitActionPerformed

    private void
menuOpcionesCargarBitActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuOpcionesCargarBitActionPerformed
        _btnCargarBitActionPerformed(evt);
    }//GEN-LAST:event_menuOpcionesCargarBitActionPerformed

    private void
menuOpcionesCargarTBActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuOpcionesCargarTBActionPerformed
        _btnCargarTBActionPerformed(evt);
    }//GEN-LAST:event_menuOpcionesCargarTBActionPerformed

    private void
menuOpcionesEjecActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuOpcionesEjecActionPerformed
        _btnEjecutarActionPerformed(evt);
    }//GEN-LAST:event_menuOpcionesEjecActionPerformed

    private void
menuOpcionesPararEjecActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuOpcionesPararEjecActionPerformed
        _btnPararEjecucionActionPerformed(evt);
    }//GEN-LAST:event_menuOpcionesPararEjecActionPerformed

    private void
menuOpcionesReanudarEjecActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuOpcionesReanudarEjecActionPerformed
        _btnReanudarActionPerformed(evt);
    }//GEN-LAST:event_menuOpcionesReanudarEjecActionPerformed

    private void
menuVistasEntityVHDActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuVistasEntityVHDActionPerformed
        try {
            jTabbedPane1.setSelectedComponent(panelVHD);
        } catch (IllegalArgumentException ex) {

            _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);
        }
```

```
    }//GEN-LAST:event_menuVistasEntityVHDActionPerformed

    private void
menuVistasCargarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuVistasCargarActionPerformed
        try {
            jTabbedPane1.setSelectedComponent(panelCargar);
        } catch (IllegalArgumentException ex) {

            _TextCargarbit.setColumns(20);
            _TextCargarbit.setRows(5);
            _TextCargarbit.setMaximumSize(getMaximumSize());
            jScrollPane2.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(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", panelCargar);
            jTabbedPane1.setSelectedComponent(panelCargar);

        }
    }//GEN-LAST:event_menuVistasCargarActionPerformed

    private void
menuVistasTBActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_menuVistasTBActionPerformed


        try {
            jTabbedPane1.setSelectedComponent(panelTB);
        } catch (IllegalArgumentException ex) {
            _txtTB.setColumns(20);
            _txtTB.setRows(5);
            jScrollPane3.setViewportView(_txtTB);

            javax.swing.GroupLayout panelTBLayout = new
javax.swing.GroupLayout(panelTB);
            panelTB.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", panelTB);
```

```
            jTabbedPane1.setSelectedComponent(panelTB);

        }
    }//GEN-LAST:event_menuVistasTBActionPerformed

    private void
menuVistasOutPutActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_menuVistasOutPutActionPerformed
        try {
            jTabbedPane1.setSelectedComponent(panelOutPut);
        } catch (IllegalArgumentException ex) {

            _TextSalida.setColumns(20);
            _TextSalida.setEditable(false);
            _TextSalida.setRows(5);
            jScrollPane4.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(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", panelOutPut);
            jTabbedPane1.setSelectedComponent(panelOutPut);
        }
    }//GEN-LAST:event_menuVistasOutPutActionPerformed

private void
_btnGenerarGoldenActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnGenerarGoldenActionPerformed
    if (this.com1 == null) {
        if (this.inicializarPuertoSerie()) {
            if (generarGolden()) {
                log.info("Generar Golden : Generado correctamente el
archivo golden");
            } else {
                log.warn("Generar Golden : No se ha podido generar
correctamente el archivo golden");
            }
        }
    } else {
        if (generarGolden()) {
            log.info("Generar Golden : Generado correctamente el
archivo golden");
        } else {
            log.warn("Generar Golden : No se ha podido generar
correctamente el archivo golden");
        }
    }
}//GEN-LAST:event__btnGenerarGoldenActionPerformed
```

```
private void
_btnCargarGoldenActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnCargarGoldenActionPerformed

    log.info("Cargando archivo Golden...");
    JFileChooser chooser;
    this._TxtEntityVHD.setText("");
    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) {

        copiaArchivo(chooser.getSelectedFile().getAbsolutePath(),
"salidas//Golden.txt");

        log.info("Cargar Golden : Se ha cargado archivo golden
correctamente");
    } else {
        log.info("Cargar Golden : No se ha seleccionado archivo");
    }

}//GEN-LAST:event__btnCargarGoldenActionPerformed

private void menuConfigNessyActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuConfigNessyActionPerformed
    String ruta = "";
    if (RUTA_XILINX != null) {
        ruta = RUTA_XILINX;
    }
    GUIConfig config = new GUIConfig(this, true, ruta);
    config.setVisible(true);

}//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
_btnCargBitReconfParcialActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btnCargBitReconfParcialActionPerformed
```

```
    /* if(this.reconfiguracion != null){
    this.reconfiguracion.pararreconfiguracionparcial();
    }*/
    inyeccErr=true;
    log.info("Comienza ejecucion Inyeccion de errores");
    seleccionaPanel(panelOutPut);
    reconfiguracion = new ReconfiguracionParcial(this, RUTA_IOSERIE);
    reconfiguracion.start();
    //setEnabledBtnDetenerInyeccion(false);
    /* if(procesoModificarFicheros())
    log.info("Reconfiguración Parcial : Ejecutado Reconfiguración
Parcial");
    else
    log.warn("Reconfiguración Parcial :No se ha podido ejecutar " +
    "correctamente Reconfiguración Parcial");*/

}//GEN-LAST:event__btnCargBitReconfParcialActionPerformed

private void _btnPararReconfActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btnPararReconfActionPerformed

    log.info("Inyeccion de errores parada");
    this.reconfiguracion.pararreconfiguracionparcial();
    this._btnPararReconf.setEnabled(false);
    this._btnPararReconf.setVisible(false);
    //this.ejecutandoReconfiguracion = false;
}//GEN-LAST:event__btnPararReconfActionPerformed

private void formWindowClosing(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_formWindowClosing

    log.info("Finalizado Nessy 2.0");

log.info("=====================================================\n");
}//GEN-LAST:event_formWindowClosing

private void
menuOpcionesGeneraGoldenActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuOpcionesGeneraGoldenActionPerformed
    _btnGenerarGoldenActionPerformed(evt);
}//GEN-LAST:event_menuOpcionesGeneraGoldenActionPerformed

private void
menuOpcionesCargarGoldenActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuOpcionesCargarGoldenActionPerformed
    _btnCargarGoldenActionPerformed(evt);
}//GEN-LAST:event_menuOpcionesCargarGoldenActionPerformed

private void
menuOpcionesReconfParcialActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_menuOpcionesReconfParcialActionPerformed
    _btnCargBitReconfParcialActionPerformed(evt);
}//GEN-LAST:event_menuOpcionesReconfParcialActionPerformed
    /**
     * Actualiza el numero de instrucción que se está ejecutando.
     * @param inst Número de instruccion actual.
     */
    public void setNumeroInst(int inst) {
        this._lblnInst.setText(Integer.toString(inst));
    }
```

```
/**
 *
 */
// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JTextArea _TextCargarbit;
private javax.swing.JTextArea _TextSalida;
private javax.swing.JTextArea _TxtEntityVHD;
private javax.swing.JButton _btnCargBitReconfParcial;
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 _btnPararEjecucion;
private javax.swing.JButton _btnPararReconf;
private javax.swing.JButton _btnReanudar;
private javax.swing.JLabel _lbl_BitCargado;
private javax.swing.JLabel _lbl_VHDLCargado;
private javax.swing.JTextField _lblnInst;
private javax.swing.JTextArea _txtTB;
private javax.swing.JLabel jLabel1;
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.JSeparator jSeparator1;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JToolBar jToolBar1;
private javax.swing.JMenu menuConfig;
private javax.swing.JMenuItem menuConfigFichConf;
private javax.swing.JMenuItem menuConfigNessy;
private javax.swing.JMenu menuOpciones;
private javax.swing.JMenuItem menuOpcionesCargarBit;
private javax.swing.JMenuItem menuOpcionesCargarGolden;
private javax.swing.JMenuItem menuOpcionesCargarTB;
private javax.swing.JMenuItem menuOpcionesCargarVHD;
private javax.swing.JMenuItem menuOpcionesCrearBit;
private javax.swing.JMenuItem menuOpcionesEjec;
private javax.swing.JMenuItem menuOpcionesGeneraGolden;
private javax.swing.JMenuItem menuOpcionesPararEjec;
private javax.swing.JMenuItem menuOpcionesReanudarEjec;
private javax.swing.JMenuItem menuOpcionesReconfParcial;
private javax.swing.JMenu menuVistas;
private javax.swing.JMenuItem menuVistasCargar;
private javax.swing.JMenuItem menuVistasEntityVHD;
private javax.swing.JMenuItem menuVistasOutPut;
private javax.swing.JMenuItem menuVistasTB;
private javax.swing.JPanel panelCargar;
private javax.swing.JPanel panelOutPut;
private javax.swing.JPanel panelTB;
private javax.swing.JPanel panelVHD;
// End of variables declaration//GEN-END:variables

private void cargarTextArea() {
    boolean error = false;
```

```
        JFileChooser chooser;

        this._txtTB.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) {
            try {
                fichero_tb =
chooser.getSelectedFile().getAbsolutePath();

                FileReader fr = new FileReader(fichero_tb);
                BufferedReader br = new BufferedReader(fr);
                String linea = br.readLine();

                int num_linea = 1;

                //Selecciona panel
                seleccionaPanel(panelTB);

                while (linea != null) {
                    if (num_linea == 280000) {
                        JOptionPane.showMessageDialog(this,
"Sobrepasado el número máximo de líneas en este modo de TB.
Sugerencia: Seleccione la otra opción para poder ejecutar el fichero
por completo", "Error", JOptionPane.ERROR_MESSAGE);
                        br.close();
                        return;
                    }
                    this._txtTB.append(linea + "\n");
                    linea = br.readLine();
                    num_linea++;
                }

                br.close();
            } catch (IOException ex) {
                error = true;
//
Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);


                this._txtTB.append(
                        "Error al cargar el banco de pruebas");
            }
            if (!error) {
                JOptionPane.showMessageDialog(this, "TestBench cargado
correctamente", "Info", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "Error al cargar
el fichero de test", "Error", JOptionPane.ERROR_MESSAGE);
            }

        } else {
            System.out.println("No Selection ");
```

```
        }
    }

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



        this._txtTB.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();

            //Selecciona panel
            seleccionaPanel(panelTB);

        } else {
            log.info("Seleccion TB no realizada ");
            error = true;
        }
        return !error;
    }

    /**
     * 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);
                    panelOutPut.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", panelOutPut);
                    jTabbedPane1.setSelectedComponent(panelOutPut);
                } else if (panel.getName().equals(panelCargar)) {
                    _TextCargarbit.setColumns(20);
                    _TextCargarbit.setRows(5);
                    _TextCargarbit.setMaximumSize(getMaximumSize());
                    jScrollPane2.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(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", panelCargar);
                    jTabbedPane1.setSelectedComponent(panelCargar);
                } else if (panel.getName().equals(panelTB)) {
                    _txtTB.setColumns(20);
                    _txtTB.setRows(5);
                    jScrollPane3.setViewportView(_txtTB);

                    javax.swing.GroupLayout panelTBLayout = new
javax.swing.GroupLayout(panelTB);
                    panelTB.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", panelTB);
                    jTabbedPane1.setSelectedComponent(panelTB);
                } 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) {
            log.error("Selecciona Panel() : " + ex);
//
Logger.getLogger(GUIPrincipal.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    /**
     * Abre un selector de ficheros para seleccionar un test bench y a
continuación
     * genera una salida golden a partir de él. Utilizado para el
comienzo
     * del proceso de reconfiguración parcial
     * @return true si ha sido correcto y false en caso contrario
     */
    public boolean SeleccionTBModifFichero() {
        boolean correcto = false;
        SeleccionTBFich = true;
        if (cargaFicheroTB()) {
            if (this.com1 == null) {
                if (this.inicializarPuertoSerie()) {
                    return generarGolden();
                }
            } else {
                return generarGolden();
            }
        }
        return correcto;
    }

    /**
     * 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_XILINX = prop.getProperty("HomeXilinx");
            if (cargaFich) {
```

```
                if (RUTA_XILINX == null || RUTA_XILINX.equals("")) {
                    JOptionPane.showMessageDialog(this, ""
                            + "El fichero de configuración
seleccionado "
                            + "no es valido", "Info",
JOptionPane.INFORMATION_MESSAGE);
                }


            } else {
                boolean cierre=false;
                while ((RUTA_XILINX == null ||
RUTA_XILINX.equals(""))&& !cierre) {
                    GUIConfig config = new GUIConfig(this, true, "");
                    config.setVisible(true);
                    is = new FileInputStream(fichConf);
                    prop.load(is);
                    RUTA_XILINX = prop.getProperty("HomeXilinx");
                    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);
        }

    }
    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 deshabilitarBtnYmenu() {
        _btnCrearBit.setEnabled(false);
        _btnCargarBit.setEnabled(false);
        _btnEjecutar.setEnabled(false);
        _btnPararEjecucion.setEnabled(false);
        _btnReanudar.setEnabled(false);
        _btnGenerarGolden.setEnabled(false);
        _btnCargarGolden.setEnabled(false);
        _btnCargarTB.setEnabled(false);
        _btnCargBitReconfParcial.setEnabled(false);

        menuOpcionesCrearBit.setEnabled(false);
        menuOpcionesCargarBit.setEnabled(false);
        menuOpcionesEjec.setEnabled(false);
        menuOpcionesPararEjec.setEnabled(false);
        menuOpcionesReanudarEjec.setEnabled(false);
        menuOpcionesGeneraGolden.setEnabled(false);
        menuOpcionesCargarGolden.setEnabled(false);
        menuOpcionesCargarTB.setEnabled(false);
        menuOpcionesReconfParcial.setEnabled(false);

    }

    /**
     * Obtiene el interfaz de puerto serie
     * @return El valor del atributo com1
     */
    public Com getCom1() {
        return com1;
    }

    private void habilitarBtnsCargarTB() {
        _btnEjecutar.setEnabled(true);
        _btnCargarGolden.setEnabled(true);
        _btnGenerarGolden.setEnabled(true);
        this.menuOpcionesEjec.setEnabled(true);
        this.menuOpcionesCargarGolden.setEnabled(true);
        this.menuOpcionesGeneraGolden.setEnabled(true);
    }

    public int dameNumIter() {
        Seleccion sel = new Seleccion();
        new GUISelNumIter(this, true, sel).setVisible(true);
        if (sel.selIter.equals(SeleccionNumIter.CANCELAR)) {
            return 0;
        } else if (sel.selIter.equals(SeleccionNumIter.OK)) {
            return sel.numIter;
    }
        return 0;
    }
}
```

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

}
```

GUISelNumIter.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 GUISelNumIter 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 GUISelNumIter(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();
        _btnOK = new javax.swing.JButton();
        _btnCancelar = new javax.swing.JButton();
        _spn_NumIter = new javax.swing.JSpinner();
        _lbl_NumIter = new javax.swing.JLabel();

        setTitle("Número de Iteraciones");
        setIconImage(null);
        setResizable(false);

        _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);
            }
        });

        _spn_NumIter.setModel(new
javax.swing.SpinnerNumberModel(Integer.valueOf(1), Integer.valueOf(1),
null, Integer.valueOf(1)));

        _lbl_NumIter.setText("Número de iteraciones");
```

```
        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()
                .addContainerGap(105, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                    .addComponent(_lbl_NumIter,
javax.swing.GroupLayout.PREFERRED_SIZE, 133,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(_btnOK,
javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(3, 3, 3)
                        .addComponent(_btnCancelar))
                    .addGroup(jPanel1Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(_spn_NumIter,
javax.swing.GroupLayout.PREFERRED_SIZE, 49,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(80, 80, 80))
        );

        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(53, 53, 53)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.BASELINE)
                    .addComponent(_lbl_NumIter)
                    .addComponent(_spn_NumIter,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(28, 28, 28)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.BASELINE)
                    .addComponent(_btnOK)
                    .addComponent(_btnCancelar))
                .addContainerGap(31, 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)
            .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 _btnOKActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event__btnOKActionPerformed


    sel.selIter=SeleccionNumIter.OK;
    sel.numIter=((Integer) _spn_NumIter.getValue());
    this.dispose();
}//GEN-LAST:event__btnOKActionPerformed

private void _btnCancelarActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event__btnCancelarActionPerformed

    sel.selIter=SeleccionNumIter.CANCELAR;
    this.dispose();
}//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.JLabel _lbl_NumIter;
    private javax.swing.JSpinner _spn_NumIter;
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration//GEN-END:variables

}
```

JTabbedPaneWithCloseIcon.java

```
package nessy20;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.HashMap;
```

```
/**
 * A JTabbedPane which has a close ('X') icon on each tab.
 *
 * To add a tab, use the method addTab(String, Component)
 *
 * To have an extra icon on each tab (e.g. like in JBuilder, showing
the file type) use
 * the method addTab(String, Component, Icon). Only clicking the 'X'
closes the tab.
 */
public class JTabbedPaneWithCloseIcon extends JTabbedPane implements
MouseListener {

    private HashMap tablaPaneles;

    /**
     *
     * @return
     */
    public HashMap getTablaPaneles() {
        return tablaPaneles;
    }

    /**
     *
     * @param tablaPaneles
     */
    public void setTablaPaneles(HashMap tablaPaneles) {
        this.tablaPaneles = tablaPaneles;
    }

    /**
     *
     */
    public JTabbedPaneWithCloseIcon() {
    super();
    tablaPaneles = new HashMap();
    addMouseListener(this);
  }

    @Override
  public void addTab(String title, Component component) {

    tablaPaneles.put(component, true);
    this.addTab(title, component, null);
  }

  /**
   *
   * @param title
   * @param component
   * @param extraIcon
   */
  public void addTab(String title, Component component, Icon
extraIcon) {
    super.addTab(title, new CloseTabIcon(extraIcon), component);
  }

  public void mouseClicked(MouseEvent e) {
    int tabNumber=getUI().tabForCoordinate(this, e.getX(), e.getY());
```

```
    if (tabNumber < 0) return;
    Rectangle rect=((CloseTabIcon)getIconAt(tabNumber)).getBounds();
    if (rect.contains(e.getX(), e.getY())) {
      //the tab is being closed
      tablaPaneles.put(getComponent(tabNumber),false);
      this.removeTabAt(tabNumber);

    }
  }

  public void mouseEntered(MouseEvent e) {}
  public void mouseExited(MouseEvent e) {}
  public void mousePressed(MouseEvent e) {}
  public void mouseReleased(MouseEvent e) {}
}
class CloseTabIcon implements Icon {
  private int x_pos;
  private int y_pos;
  private int width;
  private int height;
  private Icon fileIcon;

  public CloseTabIcon(Icon fileIcon) {
    this.fileIcon=fileIcon;
    width=16;
    height=16;
  }

  public void paintIcon(Component c, Graphics g, int x, int y) {
    this.x_pos=x;
    this.y_pos=y;

    Color col=g.getColor();

    g.setColor(Color.black);
    int y_p=y+2;
    g.drawLine(x+1, y_p, x+12, y_p);
    g.drawLine(x+1, y_p+13, x+12, y_p+13);
    g.drawLine(x, y_p+1, x, y_p+12);
    g.drawLine(x+13, y_p+1, x+13, y_p+12);
    g.drawLine(x+3, y_p+3, x+10, y_p+10);
    g.drawLine(x+3, y_p+4, x+9, y_p+10);
    g.drawLine(x+4, y_p+3, x+10, y_p+9);
    g.drawLine(x+10, y_p+3, x+3, y_p+10);
    g.drawLine(x+10, y_p+4, x+4, y_p+10);
    g.drawLine(x+9, y_p+3, x+3, y_p+9);
    g.setColor(col);
    if (fileIcon != null) {
      fileIcon.paintIcon(c, g, x+width, y_p);
    }
  }

  public int getIconWidth() {
    return width + (fileIcon != null? fileIcon.getIconWidth() : 0);
  }

  public int getIconHeight() {
    return height;
  }

  public Rectangle getBounds() {
```

```
    return new Rectangle(x_pos, y_pos, width, height);
  }
}
```

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.setVisible(true);

    }
}
```

Selección.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;
    }
}
```

SeleccionCargaVHD.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

}
```

SeleccionNumIter.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

}
```

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
}
```

Paquete que contiene las imágenes utilizadas en la aplicación además del fichero de propiedades del Log, donde se indica el nivel de traza y el destino del fichero aplicación.log

Log4j.properties

```
#################################################################
### Configuracion Nivel de Trazas y Controlador              ###
#################################################################
 log4j.rootCategory=INFO, LOGFILE


#################################################################
### Configuracion LOGFILE
###
#################################################################
 log4j.appender.LOGFILE=org.apache.log4j.DailyRollingFileAppender
 log4j.appender.LOGFILE.file=C:\aplicacion.log
 log4j.appender.LOGFILE.append=true
 log4j.appender.LOGFILE.DatePattern='.'yyyy-MM-dd

 log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout
 log4j.appender.LOGFILE.layout.ConversionPattern=%d{dd-MMM HH:mm:ss}
%m%n
```

5 Conclusiones

5.1. Sobre los conocimientos adquiridos y requeridos

Al comenzar este curso, conocíamos unas pocas cosas sobre las FPGAs. Habíamos usado en cursos anteriores Spartan para simular circuitos secuenciales muy sencillos, para los cuales utilizábamos los botones, los switches y los leds para comunicarnos con ella. A medida que hemos ido profundizando en el desarrollo del proyecto hemos ido viendo todas las posibilidades que ofrecen este tipo de dispositivos.

El manejo de la entrada/salida para la comunicación con la FPGA nos ha permitido aplicar muchos de los conceptos aprendidos durante toda la carrera en relación con el hardware. En muchas ocasiones estos conceptos se quedan en la mera teoría, y sin embargo en esta ocasión hemos podido experimentar el verdadero uso y una aplicación concreta.

También hemos adquirido un profundo conocimiento de las aplicaciones de Xilinx. Anteriormente simplemente generábamos –por ejemplo- un bitfile sin saber lo que estaba haciendo internamente el programa. También hacíamos lo mismo a la hora de cargar un bitstream a la placa FPGA. Hemos tenido que aprender cómo funcionan todos estos procedimientos para poder llevar a cabo una aplicación más sencilla de ser utilizada.

Y no solamente hemos tenido que desarrollar conocimientos relacionados con el apartado hardware de la informática. Los conceptos adquiridos durante la carrera, relacionados con el software también nos han sido imprescindibles a la hora de desarrollar esta aplicación. Así por ejemplo hemos utilizado todos los conceptos relacionados con la programación orientada a objetos para toda la división de clases de la aplicación. También hemos tenido que utilizar técnicas de procesamiento de lenguajes para poder leer ficheros con un formato determinado (en este caso formato vhdl). Asimismo, también el uso de estructuras de datos para almacenar representaciones de información y evaluar, entre otras cosas, expresiones aritméticas.

5.2. Sobre la inserción de errores

Se ha llevado a cabo todo lo necesario para poder insertar errores en una FPGA. Para ello se carga un fichero en la FPGA, se ejecuta, se modifica un bit de su configuración y se compara. Esto ha de hacerse una gran cantidad de veces. Cada iteración no es baladí; tarda bastante tiempo ya que debe cargar 2 veces una frame y ejecutar. Además debe guardar los resultados en algún tipo de soporte (en este caso un fichero) para poder analizar los resultados posteriormente. Esto aumenta mucho el tiempo que requiere para ejecutar todo. Si tenemos en cuenta que esto se debe hacer 36194 frames x 32 bits = 1.158.208 veces, veremos claramente que es imposible ejecutar esto en un tiempo razonable de tal forma que podamos obtener unos resultados adecuados.

6 Bibliografía

[1] Data Sheet Virtex-II Pro and Virtex-II Pro X FPGA User Guide
http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf

[2] Digital integrated circuits. A design perspective, Jan M. Rabaey

[3] FPGA-Based System Design, Wayne Wolf

[4] Gyovinet. http://www.giovynet.com/serialport.html

[5] Introducción a las FPGA, Alejandro Ehrenfeld G.

[6] J. F. Ziegler and W. A. Lanford, "The effect of sea level cosmic rays on electronic devices", J. Appl. Phys., vol. 52, pp. 4305–4318, 1981.

[7] M.Alderighi, F.Casini, S.D'Angelo, M. Mancini, S. Pastore, G. R. Sechi, R. Weigand, "Evaluation of Single Event Upset Mitigation Schemes for SRAM Based FPGAs Using the FLIPPER Fault Injection Platform", Defect and Fault-Tolerance in VLSI Systems, (DFT '07), 105-113, 2007

[8] Virtex-II Pro and Virtex-II Pro X FPGA User Guide
http://www.xilinx.com/support/documentation/user_guides/ug012.pdf

[9] XC95288 In-System Programmable CPLD
http://www.xilinx.com/support/documentation/data_sheets/ds069.pdf

[10] XST Guide User Guide, Xilinx
http://www.xilinx.com/itp/xilinx7/books/docs/xst/xst.pdf

[11] Development System Reference Guide 10.1
http://www.xilinx.com/itp/xilinx10/books/docs/dev/dev.pdf