Sistemas Informáticos 2005-2006:

Estudio de técnicas para multitarea HW sobre FPGAs Tridimensionales (FPGA3D)

Tutor:
D. Julio Septién del Castillo
Alumnos:
D. Jorge Díaz Pascual
D. Ángel Arbeteta Hernández
Dña. Cristina Martínez Martínez

ÍNDICE

Resumen

Este texto documenta el trabajo realizado por los autores para diseñar e implementar un simulador de FPGA 3D (Field Programmable Gate Array).

Tras una introducción sobre aspectos generales del proyecto, se da una introducción sobre circuitos lógicos programables y en concreto sobre FPGA. Después se explica el motivo de interés en este proyecto.

La segunda parte de la memoria contiene el diseño y la implementación propuestas para el simulado y detalla el trabajo realizado durante la creación del simulador.

Finalmente se concluye el proyecto dando algunas posibles modificaciones para futuros trabajos.

Palabras clave

FPGA, 3D

Abstract

This report documents the work done by the authors to design and implement a 3D graphics system on an FPGA (Field Programmable Gate Array).

After an introduction about general points of the project, an introduction about programmable circuits and in particular about FPGA is given. Then we explain the reason of the interest of this project.

The second part of this report contains the proposed graphics system design for FPGA implementation and details the work made during de creation of the 3D graphics system.

Finally, chapter six concludes the project with some suggestions for future work.

Keywords

FPGA, 3D

1 INTRODUCCIÓN

En la realización de este estudio, se aborda la multitarea hardware en FPGAs tridimensionales desde un punto teórico, nunca práctico, teniendo como objetivo desarrollar una herramienta que permita simular dicha tecnología, entendiendo primero en qué consisten las FPGA, la reconfiguración y la multitarea hardware.

Para ello, se hace una introducción de cada uno de los conceptos por separado (definiciones, usos, modificaciones sufridas en el tiempo, tecnologías, ejemplos comerciales) para entender las motivaciones del estudio de las FPGAs de tres dimensiones.

Una vez *justificado* su estudio, se presenta una herramienta software que las *simula* , quedando la gestión reducida a cálculos de geometría, como se verá.

Por último, se exponen, además de otros anexos, puntos que no se tienen en cuenta en dicha herramienta y que pueden servir de punto de partida en futuros proyectos de la asignatura de Sistemas Informáticos.

2 Hardware reconfigurable

2.1 Definición

Existen dos alternativas hardware para ejecución de programas. La primera es diseñar hardware específico para la resolución de un problema, es lo que se conoce como ASIC (*Applicaction Specific Integrated Circuit*), donde un componente o combinación de varios se unen para realizar operaciones en hardware. El segundo método, es usar un hardware que sea capaz, mediante configuración, de comportarse de diferentes modos, de tal forma que pueda resolver distintos problemas; esto es lo que se conoce como **hardware reconfigurable.**

Este tipo de hardware tiene múltiples ventajas con respecto a los ASIC:

- velocidades comparables
- funcionalidad determinada mediante bits de configuración **y modificable**
- al fabricarse gran tirada de circuitos integrados, resultan más baratos
- una vez distribuido, no hace falta, en el momento de realizar algún cambio, volver a fabricar otro CI, bastaría con reconfigurarlo

Tienen varias entradas y salidas, veamos un ejemplo muy sencillo con el siguiente circuito:

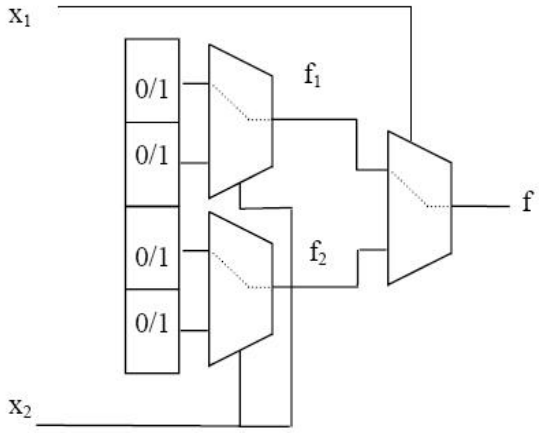
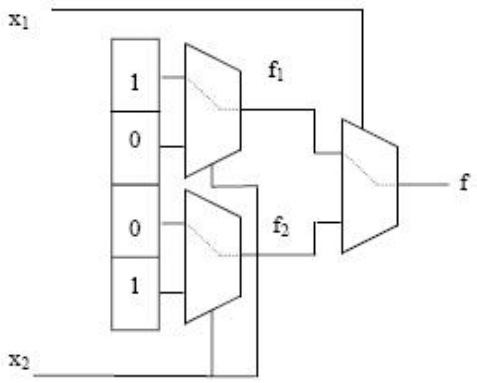


Ilustración 1: Circuito inicial

Para implementar la función $f = \bar{x}_1\bar{x}_2 + x_1x_2$ con el circuito de la ilustración 1, habría que configurar las entradas de la siguiente forma:



x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	1

$f = \bar{x}_1\bar{x}_2 + x_1x_2$

Ilustración 2: Circuito implementando una función

2.2 Características generales

A continuación, se describen las principales características de este tipo de hardware.

2.2.1 Tecnologías de configuración

Existen dos modelos de configuración:

- al arrancar: configuración estática
- durante la ejecución de un programa: **reconfiguración dinámica**

Con la primera opción, las posibilidades de aceleración están limitadas por la capacidad de los circuitos integrados programables. El flujo de estados es el siguiente:

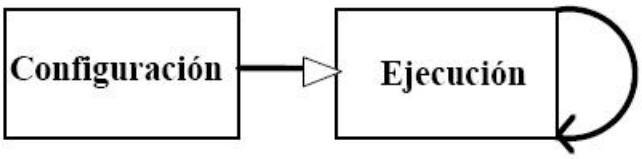


Ilustración 3: Configuración estática

En el modelo de reconfiguración dinámica o RTR (*Run Time Reconfiguration)*, el flujo es el siguiente:

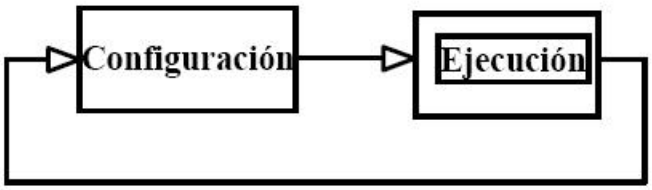


Ilustración 4: Reconfiguración dinámica

- ⇨ se incrementa la capacidad de aceleración ya que se pueden cargar varias configuraciones durante la ejecución de un programa

El tiempo empleado en cargar una configuración se convierte en una limitación de la posible aceleración del sistema reconfigurable.

Actualmente, existen tres tecnologías de configuración:

- mediante memoria volátil estática (SRAM)
- mediante antifusibles
- mediante EPROM, EEPROM o FLASH

2.2.2 Reconfiguración parcial y dinámica

Una vez que el hardware está arrancado y está en ejecución, es posible reconfigurarlo.

Además, no es necesario reconfigurarlo por completo, si no que es posible únicamente reconfigurar una sección sin que el resto se vea afectada, de tal manera que mientras se está reconfigurando una parte, la otra puede estar en ejecución.

Mediante la reconfiguración, se aumenta notablemente el rendimiento de un circuito, ya que para empezar a ejecutar tareas en él, no es necesario esperar a que terminen todas las posibles tareas cargadas en un momento dado, sólo habría que esperar a que hubiera suficientes recursos disponibles.

2.3 Conclusiones

El hardware reconfigurable permite obtener distintos resultados de ejecución bajo un mismo hardware y mismas entradas, modificando su configuración. Esto supone que ante posibles modificaciones futuras de comportamiento no sea necesario fabricar un circuito nuevo específico para la nueva funcionalidad, si no que simplemente habría que **reconfigurarlo**, esto es un beneficio en coste.

Además, con un mismo hardware, podemos obtener un alto rendimiento ejecutando simultáneamente varias tareas, pudiendo reconfigurar el hardware *on-air*, mientras se encuentra en ejecución. De esta forma se consigue la **multitarea hardware**, como veremos en el siguiente punto.

3 FPGA

3.1 Qué son

Las FPGAs se incluyen dentro del grupo de hardware reconfigurable de *grano fino*. Su estudio se remonta a mediados de los años 80s.

Las siglas responden a *Field Programmable Gate Arrays*: Campos de Matrices de Puertas Programables.

Representa uno de los últimos avances en tecnología de dispositivos lógicos programables. Es importante señalar que una FPGA realmente se reconfigura con un programa, a diferencia de lo que normalmente se conoce como sistema programado (microcontrolador, microprocesador, etc) en donde un hardware fijo es capaz de interpretar y ejecutar un programa especificado como un conjunto de instrucciones por el programador, en las FPGA lo que se tiene es un hardware que se configura realizando conexiones físicas que son especificadas por un programa o cadena de configuración.

Es importante notar que al realizar un diseño con FPGA se presentan los mismos inconvenientes que al realizar un sistema con componentes discretos, es decir toman relevancia los fenómenos de retardo de propagación y los relacionados con las señales de reloj. (jitter, etc).

Su estructura no está compuesta por compuertas AND/OR , en su lugar contienen *bloques lógicos* para implementar las funciones requeridas. La estructura general de una FPGA es la siguiente:

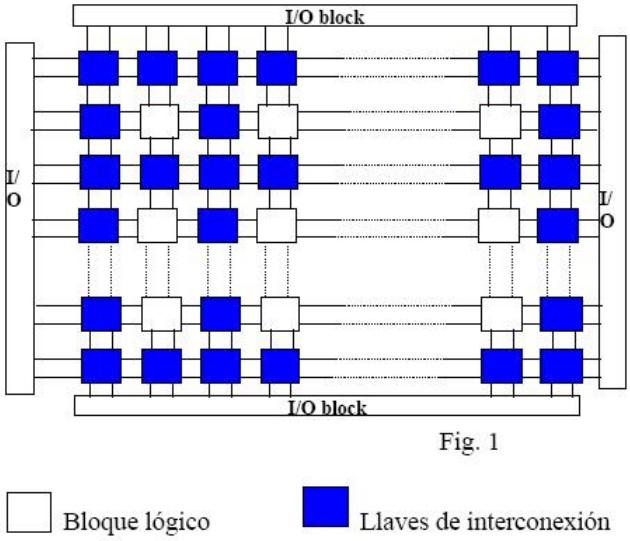


Ilustración 5: Estructura general de una FPGA

Cada bloque lógico se denomina CLB. Los bloques de entrada/salida se denominan IOB.

La tarea del programador es definir la función lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos todos.

3.2 Para qué se usan

Las características de las FPGA son su flexibilidad, capacidad de procesado en paralelo y velocidad. Esto les convierte en dispositivos idóneos para:

- Simulación y depuración en el diseño de microprocesadores.
- Simulación y depuración en el diseño de ASICs.
- Procesamiento de señal digital, por ejemplo vídeo.
- Sistemas aeronáuticos y militares.

3.3 Reconfigurables en 1D y 2D

En un modelo de 1D de FPGA, la reconfiguración se hace por filas o columnas completas de CLBs. Supongamos, por ejemplo, que tenemos las tareas T1, T2 y T3, una posible configuración sería la siguiente:

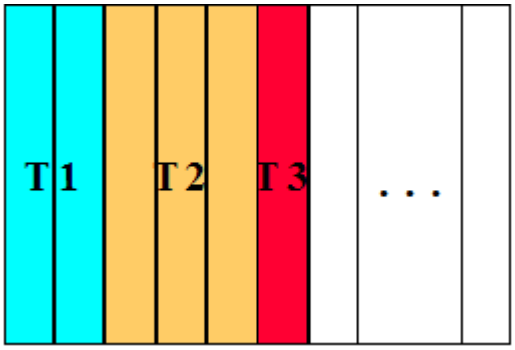


Ilustración 6: Ejemplo de configuración en 1D

Cuando se trabaja en 2D, la reconfiguración se hace por bloques de CLBs, bajo demanda del usuario. Para la anterior lista de tareas, una posible configuración sería la siguiente:

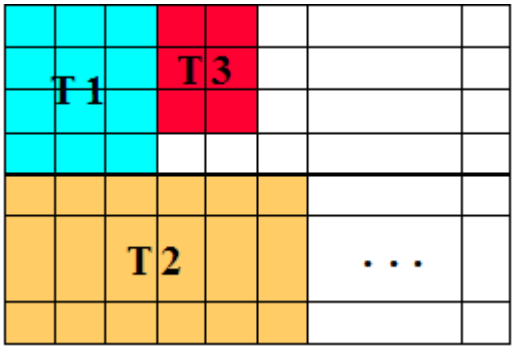


Ilustración 7: Ejemplo de configuración en 2D

Para gestionar los CLBs, se trabaja con particiones, habiendo dos técnicas:

- tamaño fijo
- tamaño arbitrario

Cada una de ellas tiene sendas ventajas y inconvenientes, como veremos a continuación.

Al realizar particiones de tamaño fijo, en cada reconfiguración es probable que queden CLBs libres en la partición donde se está ubicando la tarea, degradándose el rendimiento pero siendo su gestión muy simple.

Si se realizan particiones de tamaño arbitrario, su tamaño se ajusta directamente al tamaño de las tareas, lo que le da mayor flexibilidad, pero su gestión se vuelve más complicada y aparece la fragmentación externa, frecuente en 2D y lógico en 1D y 2D al salir las tareas y dejar los huecos.

Veamos ejemplos de ambas técnicas de particionado:

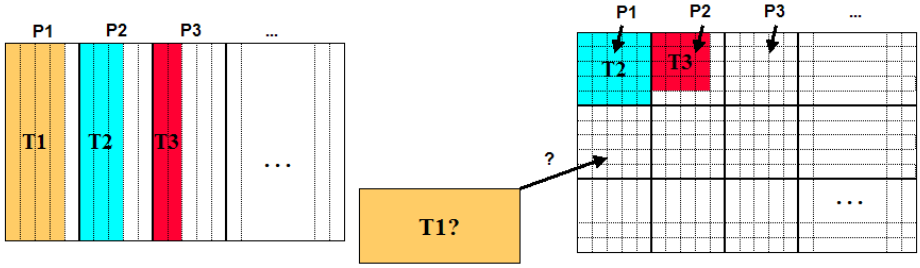


Ilustración 8: Ejemplo de particiones de tamaño fijo en 1D y 2D

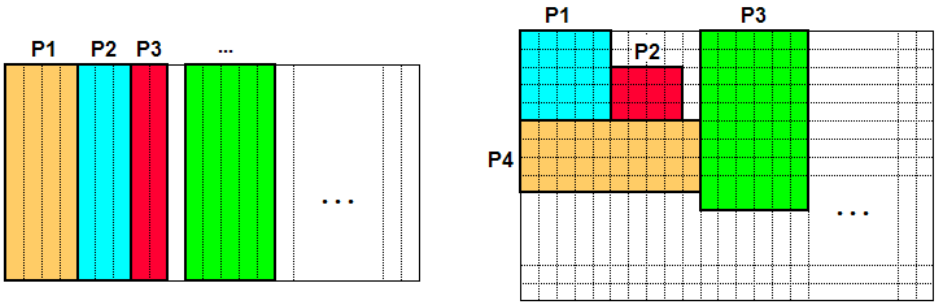


Ilustración 9: Ejemplo de particiones arbitrarias en 1D y 2D

En la ilustración 8, se puede ver cómo al ubicar tareas en las particiones, quedan huecos libres. En la 9, sin embargo, las tareas se ubican en particiones que tienen su mismo tamaño.

Nótese, que una misma tarea puede tener distintas *formas*, es decir, con diferentes configuraciones se puede obtener el mismo resultado. Se puede dar el caso de que sean iguales pero distinto *aspecto*, ocupando el mismo área y teniendo el mismo tiempo de ejecución, o también que ocupen diversas áreas y tengan distintos tiempos de ejecución (mayor paralelismo).

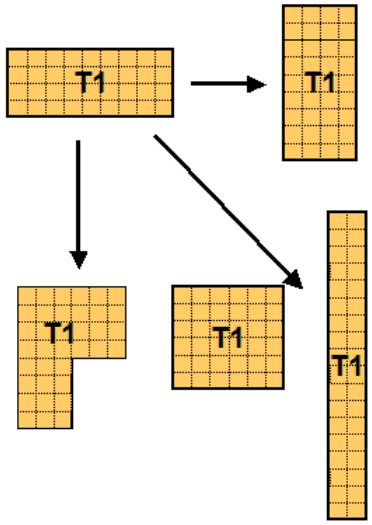


Ilustración 10: Distintas configuraciones de una misma tarea

También es posible que una aplicación se pueda dividir en varias tareas, se denominan aplicaciones jerárquicas.

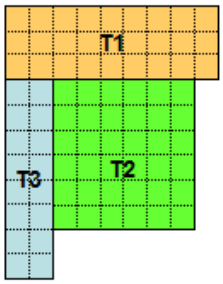


Ilustración 11: Aplicación jerárquica

3.4 Gestión de Multitarea Hardware con FPGAs

Las características de las FPGA hacen que pueda haber multitarea *real,* de tal forma que puede haber más de una tarea ejecutándose en distintas *zonas* de la FPGA, como hemos visto. Además, según van terminando tareas, van dejando libres las CLBs que estaban usando y es posible ubicar nuevas tareas.

Todo este trasiego de tareas necesita una gestión:

- tareas pendientes de ejecutarse
- tareas en ejecución
- CLBs libres
- CLBs ocupadas
- algoritmos de ubicación de tareas
- etc

3.4.1 Características de las tareas

Los datos necesarios de gestionar de las tareas son:

- *h*: ancho
- *w*: alto
- *t_ex*: tiempo de ejecución
- *t_arr*: instante de llegada
- *t_max*: instante máximo permitido para que finalice
- *t_conf*: tiempo de configuración, función de h y w
- ⇨ si se admiten varias formas, cada una tendrá h, w y t_ex diferentes

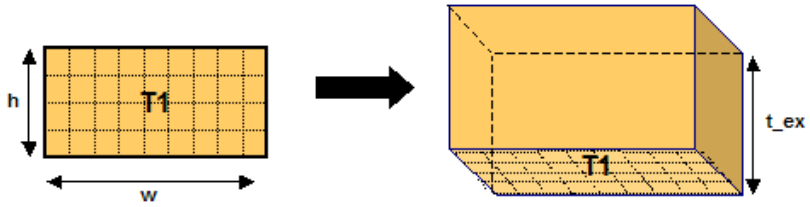


Ilustración 12: Características de una tarea

3.4.2 Planificación de las tareas

Además, para poder planificar la ejecución de las tareas, hay que tener en cuenta lo siguiente:

- si t_ex < t_max - t_arr - t_conf , entonces es posible demorar la planificación de la tarea, en el intervalo desde t_start = t_arr + t_conf hasta t_start = t_max - t_ex
- razones de la demora: no hay sitio, interesa dejar paso a otra tarea más urgente (se aproxima su t_max)
- si se supera el valor máximo de t_start ya no es posible ejecutar la tarea: descarte => pérdida de rendimiento
- se dispone de t_marg para realizar distintas operativas: preemption, reubicación, defragmentación, etc

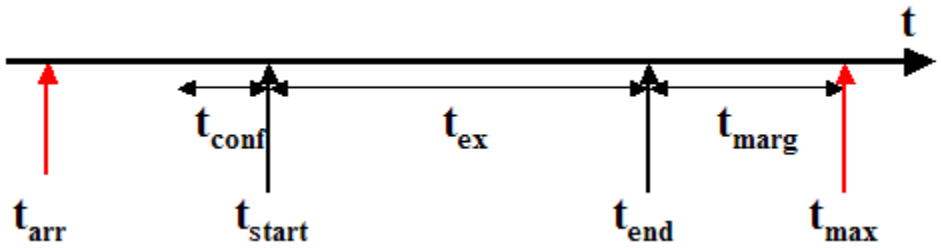


Ilustración 13: Gestión de la planificación

3.4.3 Ubicación de las tareas

La gestión de la ubicación de las tareas en la FPGA varía según se hagan particiones de igual o distinto tamaño.

Cuando las particiones son fijas, se trabaja con posiciones fijas. Las principales características son:

- la información necesaria sobre el espacio libre es un único bit asociado a cada partición
- los algoritmos de asignación son simples: por ejemplo, primera partición libre, o partición más aproximada en tamaño

- la ubicación dentro de la partición es irrelevante: por ejemplo, Bottom-Left, esquina inferior izquierda

En el caso de particiones variables, las posiciones se asocian a posiciones arbitrarias. Sus principales características son:

- mantenimiento de información sobre el espacio libre en estructuras complejas
- cualquier CLB puede ser candidato a ubicar la tarea (si está libre, obviamente)
- algoritmos de gestión del área libre complejos, con el fin de reducir el tiempo de búsqueda del espacio libre para ubicación de la tarea

3.4.4 Aceleración de la carga de las tareas

Existen varias técnicas para reducir el tiempo de carga de la configuración de las tareas en la FPGA. A continuación se nombran algunas:

- prebúsqueda y precarga: el mapa de bits de la tarea planificada se busca y carga con antelación al instante de inicio
- compresión de configuraciones: los mapas de bits se cargan comprimidos y la FPGA tiene un HW de descompresión
- cachés de configuraciones: pueden almacenarse configuraciones por si vuelven a ejecutarse más adelante
- múltiples contextos con reconfiguración parcial: carga en un contexto mientras se ejecuta otro; el cambio de contexto se hace en un ciclo

3.4.5 Entrada/Salida

Aquí nos referimos a la comunicación entre tareas. Obviamente, el tiempo necesario para comunicarse dos CLBs adyacentes es menor que entre dos CLBs más distantes, debido al retardo de propagación de las señales.

Para esta comunicación, se diseñan buses, cuando las particiones de la FPGA son fijas; y redes de interconexión estándar cuando las particiones son variables.

3.4.6 Fragmentación de la FPGA

Veamos la siguiente FPGA:

Ilustración 14: FPGA fragmentada

Con el tiempo, las tareas han ido entrando y saliendo, adoptando formas irregulares y dejando CLBs libres. Supongamos ahora que hay que planificar una tarea que requiere 4 x 3 CLBs (12). En esta situación, la FPGA tiene 12 CLBs libres, sin embargo, no es posible planificarla porque *no cabe*, no hay 12 CLBs formando un hueco para dicha tarea. Este problema se conoce con el nombre de **fragmentación**.

Para controlar estas situaciones se utilizan métricas de fragmentación, para obtener valores numéricos cuantificables. Por ejemplo:

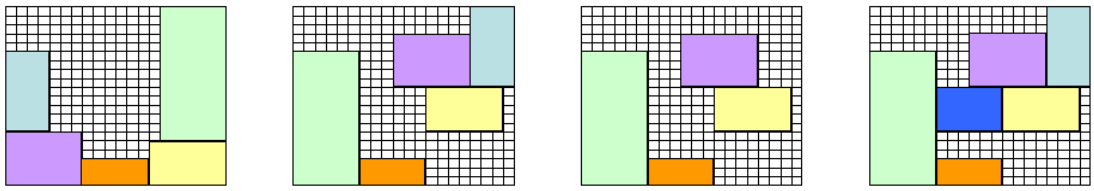


Ilustración 15: De izquierda a derecha, el grado de fragmentación es mayor

De esta forma, se puede orientar el proceso de ubicación de las tareas y desencadenar operativas de **defragmentación** cuando se llegue a un umbral. Esta técnica consiste en reubicar las tareas, pero sólo si se satisfacen las restricciones de planificación (t_marg de cada tarea que se va a reubicar). Veamos el resultado de defragmentar la FPGA de la ilustración 14:

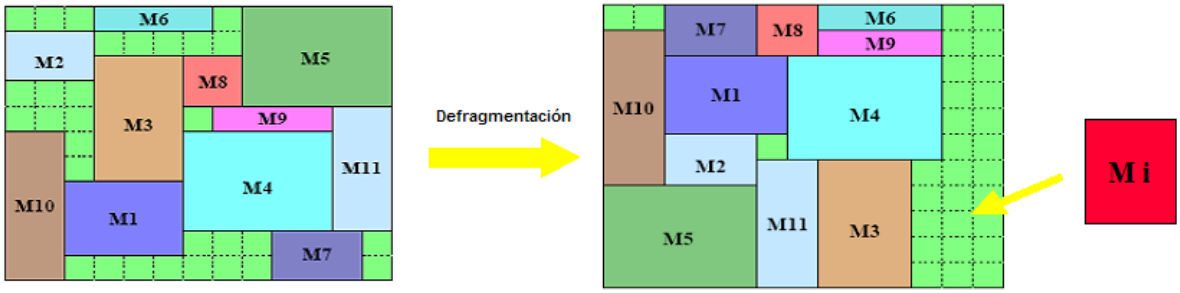


Ilustración 16: Resultado de la defragmentación

En esta situación, ya sí es posible ubicar una tarea de 3 x 4.

3.5 Ejemplo comercial: Virtex de Xilinx

Xilinx es el mayor fabricante de FPGAs del mundo. A continuación, la Virtex:

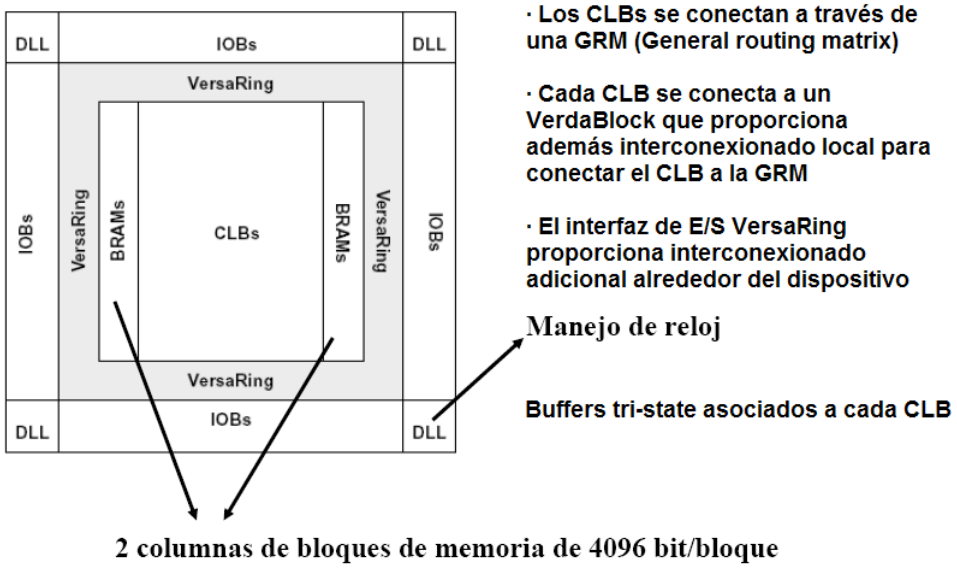


Ilustración 17: Estructura básica de la Virtex de Xilinx

4 Circuitos Integrados 3D

En la actualidad existen algunos circuitos integrados en 3D. En los últimos años el interconexionado ha llegado a ser el mayor cuello de botella de los circuitos integrados 2D por lo que los CI's 3D ofrecen una atractiva alternativa a este problema ya que las mejoras en los circuitos integrados 2D están llegando a sus límites físicos.

Aunque durante varias décadas han coexistido varios tipos de tecnologías 3D, sólo en los últimos años se ha conseguido desarrollar tecnologías de diseños integrados 3D con menor coste.

Los procesos de fabricación obtenidos en el Instituto Tecnológico de Massachussets (MIT), IBM y el Laboratorio Lincoln del MIT, han llegado a producir circuitos integrados con una separación entre capas de unos diez micrómetros.

En las arquitecturas tridimensionales, se consiguen mejoras de hasta el 20% en el retardo del circuito y del 30% en longitud de interconexionado. También disminuye el número de transistores, por lo que el área del circuito es menor y se mejora la congestión existente en los CI's 2D.

En contraposición, al haber una alta concentración de componentes en espacio reducido, aumenta la cantidad de energía disipada => calentamiento; por lo que requiere una gestión térmica, distribuyendo uniformemente las secciones de alta potencia para repartir la carga por todo el diseño.

Otra desventaja es la falta de herramientas CAD para gestionar la complejidad de las tres dimensiones, más adelante presentaremos una.

5 Multitarea Hardware en 3D: FPGAs tridimensionales

Hasta ahora, hemos mostrado las principales características del hardware reconfigurable en 1D y 2D, con el fin de estar en condiciones de presentar un paso más, el hardware en 3D, concretamente las FPGAs tridimensionales.

5.1 Motivación

Como hemos visto hasta ahora, las FPGAs en 1D y 2D son capaces de implementar lógica arbitraria, dando un alto rendimiento y un bajo coste comparado con un ASIC, ya que tienen un ciclo de vida más largo.

Esta flexibilidad se ve penalizada por un cuello de botella: los retardos de propagación de las señales a la hora de comunicarse unos CLBs con otros, siendo menor cuanto más próximos están.

En un circuito en 1D o 2D, un CLB puede tener, en el mejor de los casos, 4 CLBs *vecinos*¸ con los que comparte adyacencia, con los que el retardo de propagación será mínimo. Cuando las tareas ocupan muchos CLBs o cuando las tareas tienen que comunicarse entre sí, el tiempo necesario para esta comunicación degrada el rendimiento de la FPGA.

Si pensamos ahora en una estructura en tres dimensiones un CLB podría tener 9 vecinos con los que el tiempo de retardo sería mínimo. De esta forma aumentaría el rendimiento del hardware:

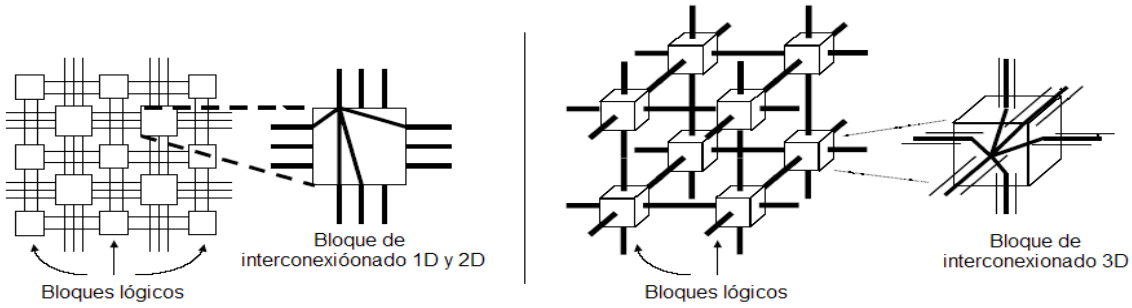


Ilustración 18: Diferencia de interconexionado entre 1D/2D y 3D

Al igual que en arquitecturas 1D y 2D, es posible conseguir multitarea hardware. Igualmente, habrá que gestionar lo visto en el punto 3.4:

- características de las tareas
- planificación de las tareas
- ubicación de las tareas
- entrada/salida
- fragmentación

5.2 Arquitecturas 3D en FPGAs

Una posible implementación de FPGAs tridimensionales es el apilado de FPGAs de dos dimensiones unidas mediante conexiones verticales denominadas “solder bumps”, como se muestra en la siguiente ilustración:

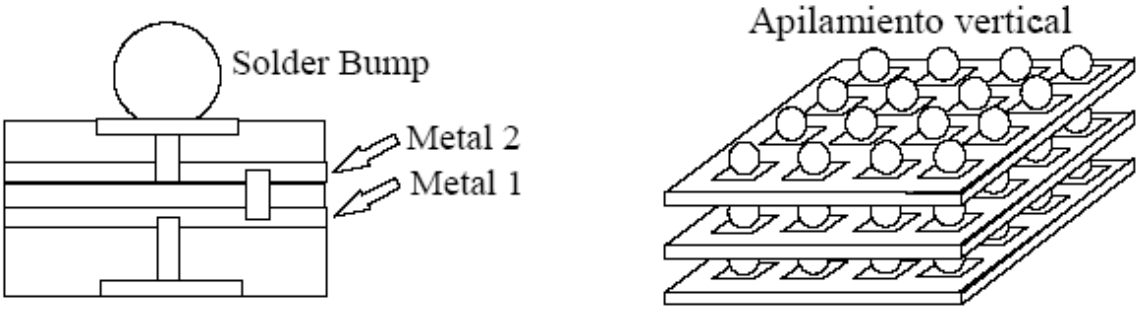


Ilustración 19: Apilado de FPGAs para formar una FPGA tridimensional

6 Simulador software de FPGAs tridimensionales

En esta parte de la memoria pasamos a describir las tareas llevadas a cabo para la implementación del simulador software de FPGAs tridimensionales.

6.1 Objetivo: simulación de multitarea HW sobre FPGA 3D

Debido a las mejoras que supone la implementación 3D de las FPGA:

- Interconexiones más cortas
- Mejor rendimiento
- Menos transiciones
- Menor área

Y ya que está previsto que en un futuro no muy lejano se empiecen a comercializar circuitos integrados 3D, el objetivo de este proyecto es la creación de un simulador de multitarea HW sobre FPGA 3D.

Se ha diseñado e implementado una herramienta de visualización y control que permite simular la gestión de una FPGA tridimensional con reconfiguración parcial, que permite la ejecución simultánea de múltiples tareas HW 3D.

Más concretamente, se han tenido en cuenta los siguientes subproblemas:

- Se ha utilizado un modelo de arquitectura FPGA3D ya propuesto a nivel académico, analizado previamente con cierto detalle.
- Se propone un modelo de tarea HW tridimensional, partiendo de la hipótesis de que ya existen herramientas de diseño que generan este tipo de circuitos sobre modelos de FPGA3D (ya existen a nivel académico).
- Se ha programado un entorno de gestión que admite la entrada de tareas HW 3D, capaces de cargarse y ejecutarse sobre el modelo de FPGA3D, que realiza una planificación sencilla de las tareas (decide cuándo van a ejecutarse) y les asigna una ubicación para su ejecución dentro del volumen de la FPGA3D.
- Se ha diseñado una utilidad de visualización que permite seguir de forma gráfica la ejecución de las tareas sobre la FPGA3D.
- Se ha estudiado la repercusión del problema de la fragmentación del volumen disponible, a medida que las tareas finalizan su ejecución, y se considera la posibilidad de aplicar técnicas de defragmentación.

6.2 Definiciones

En este punto especificamos las abstracciones realizadas necesarias para la implementación del simulador.

6.2.1 Abstracción de las tareas a cargar en la FPGA

Las características principales de las tareas de la FPGA son:

- **dimensión** de la tarea sobre el eje **x**
- **dimensión** de la tarea sobre el eje **y**
- **dimensión** de la tarea sobre el eje **z**

Estas dimensiones anteriores definen el tamaño de las tareas que vamos a intentar ubicar en la FPGA durante la ejecución de la simulación.

- **duración** de la tarea, representa el número de ciclos que estará cargada la tarea en la FPGA
- **time out** de la tarea, es el instante absoluto antes del cual debe haber sido ejecutada la tarea. Es decir representa el tiempo máximo absoluto que determina el valor máximo de ejecución de la tarea en cuestión. Si una tarea no ha finalizado su ejecución antes de su time out, se descarta.

En realidad, se tiene en cuenta el time out de una tarea antes de cargarla, es decir, si en el momento en el que vamos a cargar una tarea en la FPGA el tiempo actual de la simulación más la duración de la tarea a cargar es mayor o igual a su time out, la tarea no se carga sino que se descarta.

- **identificador** asociado a cada tarea, que la identifica de manera única. Cuando una tarea se ubica, los CLB's en los que se encuentra pasan a tener el identificador de la tarea que contienen.
- **Colores** que representan la tarea, cada tarea tiene asociados unos colores (R, G, B) que la distinguen del resto de las tareas, para que el usuario del simulador pueda diferenciar a simple vista las tareas que están cargadas en cada instante en la FPGA.
- **Inicio** de la tarea, es el instante de tiempo en el que se ha cargado la tarea en la FPGA o instante de inicio de ejecución.
- **Tiempo de llegada** de la tarea, o instante en el cual una tarea llega para ser cargada en la FPGA, solo podemos intentar ubicar tareas cuyo tiempo de llegada sea menor o igual al tiempo actual de la ejecución.

Las tareas se almacenan en dos listas:

- **Lista de tareas pendientes**, inicialmente todas las tareas estarán en esta lista, ya que hemos de ubicarlas, es decir, están pendientes de ubicar.
- **Lista de tareas cargadas**, en esta lista se añaden las tareas en el momento en que se ubican en la FPGA, o son descartadas porque ha pasado su time out o tiempo máximo de ejecución. De esta manera, al insertar una tarea en la lista de tareas cargadas hemos de eliminarla de la lista de tareas pendientes.

La lista de tareas pendientes esta ordenada ascendentemente respecto al tiempo de llegada de las tareas que contiene.

La lista de tareas cargadas no esta ordenada, cuando se carga una tarea se añade al final de esta lista.

6.2.2 Abstracción de la FPGA 3D

Para poder simular el funcionamiento del circuito integrado correspondiente a una FPGA3D se ha desarrollado el simulador basándonos en una matriz tridimensional que representa la FPGA3D, en la que las coordenadas de la matriz se corresponden con los CLB de la FPGA.

La FPGA con la que trabajamos se genera de manera aleatoria en base a los valores introducidos por el usuario (o los valores por defecto) de la pestaña de configuración.

Los posibles valores a configurar son los siguientes:

- **Tamaño x**, se puede configurar el tamaño de la dimensión sobre el eje x de la FPGA
- **Tamaño y**, se puede configurar el tamaño de la dimensión sobre el eje y de la FPGA
- **Tamaño z**, se puede configurar el tamaño de la dimensión sobre el eje z de la FPGA

Estos tamaños se corresponden con las dimensiones de la matriz que representa a la FPGA3D

6.2.3 Algoritmos utilizados

En el simulador se han implementado dos algoritmos de ubicación de tareas:

- Algoritmo **heurístico**
- Algoritmo que recorre la FPGA3D desde los **extremos**

Ambos algoritmos utilizan una heurística que pasamos a describir.

6.2.4 Heurística empleada

La heurística que hemos tenido en cuenta para implementar los algoritmos de ubicación de las tareas dentro de la FPGA consiste en calcular para cada CLB de la FPGA3D la distancia a las *paredes* más cercanas al CLB para sus 3 dimensiones. A este valor se le suma el número de CLB's adyacentes que están libres. El objetivo al ubicar una tarea es minimizar el valor resultante de sumar la heurística de todos los CLB's que ocuparía la tarea que se intenta ubicar.

Esta heurística nos asegura **que las tareas se ubicaran lo mas cerca posible de los extremos de la FGPA3D o de CLB's ya ocupados**, de manera que se intenta minimizar la fragmentación al ubicar tareas.

6.2.5 Algoritmo heurístico

Este algoritmo consiste en recorrer cada CLB de la FPGA3D. En este caso no nos importa el orden en el que se recorren los CLB's de la FPGA3D ya que se van a tener en cuenta todos.

Para cada CLB de la FPGA3D se intenta ubicar la tarea en 6 posiciones diferentes, las correspondientes a las posibles permutaciones positivas de los tamaños de la tarea dentro de la FPGA.

Comprobamos que los CLB's que ocuparía la tarea desde esta posición inicial y en la dirección actual estén libres.

Se calcula el coste de ubicar la tarea. Entendemos por coste el sumatorio de las heurísticas de cada CLB de la tarea en la posición actual. El objetivo es minimizar el coste, por lo que solo almacenaremos los datos correspondientes a la posición actual si el coste en este momento mejora el coste anterior.

Tras probar como origen de ubicación todos los CLB's de la FPGA3D, se comprueba si se ha podido ubicar la tarea y se almacena en los CLB's correspondientes a la ubicación que minimiza el coste de entre las posibles (que pasaran a estar ocupados).

6.2.6 Algoritmo que recorre la FPGA3D desde los extremos

Este algoritmo se basa en minimizar los huecos que se producen al ubicar las tareas en la FPGA. La idea surgió al pensar en un ejemplo sencillo:

Intentamos empaquetar pequeñas cajitas (que representan las tareas) en una caja grande (que representa a la FPGA3D). Lo más normal es situar inicialmente las cajitas en los extremos de la caja, para evitar que se generen huecos.

Siguiendo esta idea, lo que hace este algoritmo es recorrer la matriz desde los extremos hacia el centro y parar en el momento en el que se encuentra una ubicación para la tarea, ya que suponemos que esta será la ubicación ideal de la misma.

Para cada CLB que tomamos como origen en el recorrido de la FPGA desde los extremos intentamos ubicar la tarea en sus diferentes posiciones (o permutaciones de las dimensiones x, y, z en sentido positivo o negativo), para cada una de estas posiciones se calcula el coste de ubicar la tarea desde el origen y en la posición actual y en caso de poder ubicarla se elige la posición que minimiza el coste, entendiendo como coste el sumatorio de la heurística de cada CLB que ocuparía la tarea a ubicar.

La búsqueda de la ubicación de la tarea finaliza en el momento en el que hemos encontrado una posición en la que ubicar la tarea y se ocupan los CLB's.

6.3 Defragmentación de la FPGA3D

En el simulador se incluye la posibilidad de defragmentar la FPGA cuando finalice alguna tarea que ya está cargada en la misma.

Esta opción es útil si pensamos en que las tareas, cuando finalizan, dejan huecos en la FPGA que pueden causar el retraso de otras tareas pendientes de cargar, y que podrían ser cargadas si estos huecos formaran un hueco grande, situación que se daría si las tareas que permanecen en ejecución, es decir todavía cargadas en la FPGA3D, estuvieran ubicadas en la posición ideal dentro de la FPGA3D. Esta posición ideal se supone que se consigue al ubicar de nuevo todas las tareas que se encuentran dentro, por lo tanto en la opción de defragmentar, se realiza una reubicación de las tareas que están dentro de la matriz con los algoritmos anteriormente descritos cada vez que salen las tareas que han finalizado y siempre que la opción defragmentar este activada.

Si la opción de defragmentar no esta activada, la finalización de las tareas no produce reubicación de las tareas cargadas en la FPGA3D que continúan su ejecución.

Hemos decidido ofrecer la posibilidad de defragmentar la FPGA para que sea el usuario el que decida si quiere realizar defragmentación o no en la ejecución de la simulación ya que en la realidad el coste de aplicar la defragmentación en un chip de las características que tiene en cuenta nuestro simulador de FPGA3D es muy elevado, por lo que no se aplicara normalmente. Por lo tanto para poder realizar simulaciones que se ajusten en mayor medida al caso real la defragmentación es opcional.

De este modo permitimos ver el distinto comportamiento de la simulación teniendo en cuenta si se aplica o no la defragmentación.

6.4 Entorno de simulación

Tras haber detallado los puntos principales de la implementación del simulador, nos disponemos en este punto a describir el entorno de simulación creado detallando las características esenciales del mismo.

El simulador se ha implementado en C++, más concretamente hemos utilizado la herramienta Builder C++ 5.

6.4.1 Visualización del estado de la FPGA

El estado de la FPGA3D en cada momento puede verse en la imagen de la parte central de la ventana del simulador, que es siempre visible:

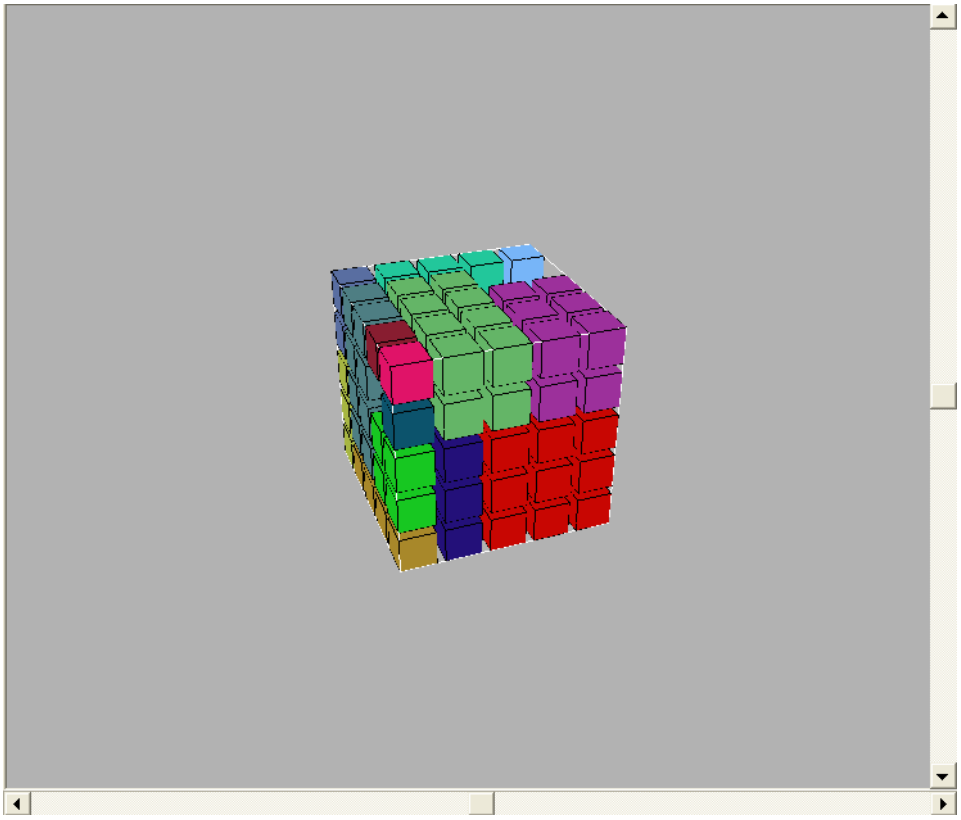
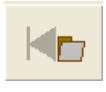
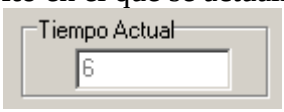


Ilustración 20: Ejemplo de visualización de matriz

Lo que veremos fundamentalmente es el dibujo de la matriz que representa la FPGA y dentro de esta se visualizan las casillas (o CLB's) de diferentes colores en caso de que haya cargada una tarea o con el color de fondo en caso de que la casilla correspondiente se encuentre vacía, es decir, que el CLB este libre.

También son siempre visibles los botones principales para el funcionamiento del simulador, que son:

-  Botón **parar**, para la simulación y destruye los valores generados en la ejecución anterior.
-  Botón **iniciar**, comienza una nueva simulación, creando desde el principio tanto la FPGA como las listas de tareas. Este botón comienza la ejecución automática del simulador.
-  Botón **pausar**, para la ejecución automática del simulador.
-  Botón **anterior**, reproduce el estado de la FPGA para el instante de tiempo anterior al actual
-  Botón **siguiente**, realiza las operaciones correspondientes para pasar al estado de tiempo siguiente al actual.
-
- Otros botones siempre visibles, correspondientes a una funcionalidad añadida al simulador son:
-  Botón **siguiente importar**, pasa al instante de tiempo siguiente al actual de una simulación realizada en un simulador externo. Los datos son cargados de un fichero generado en otra aplicación.
-  Botón **cargar fichero externo**, carga la configuración general de la FPGA de una simulación generada en otra aplicación.
-  Botón **anterior importar**, pasa al instante de tiempo anterior al actual de una simulación generada en una aplicación externa. Los datos se cargan de un fichero (generado por otra aplicación).

Además, hay un **campo de texto** en el que se actualiza el instante de **tiempo actual** de la ejecución de la simulación. 

6.4.2 Pestaña Tareas

En la pestaña de tareas podemos ver el estado correspondiente a las listas de tareas. En cada instante se muestra el contenido de la lista de tareas cargadas (que como hemos dicho anteriormente son las tareas que ya han sido cargadas en la FPGA o que han sido rechazadas por haberse pasado su time out) y el contenido de la lista de tareas pendientes (que son las tareas que todavía vamos a intentar cargar y no han sido cargadas aun porque su instante de llegada es menor al instante de tiempo actual o porque aun teniendo un instante de llegada menor o igual al instante de tiempo actual no caben en la FPGA).

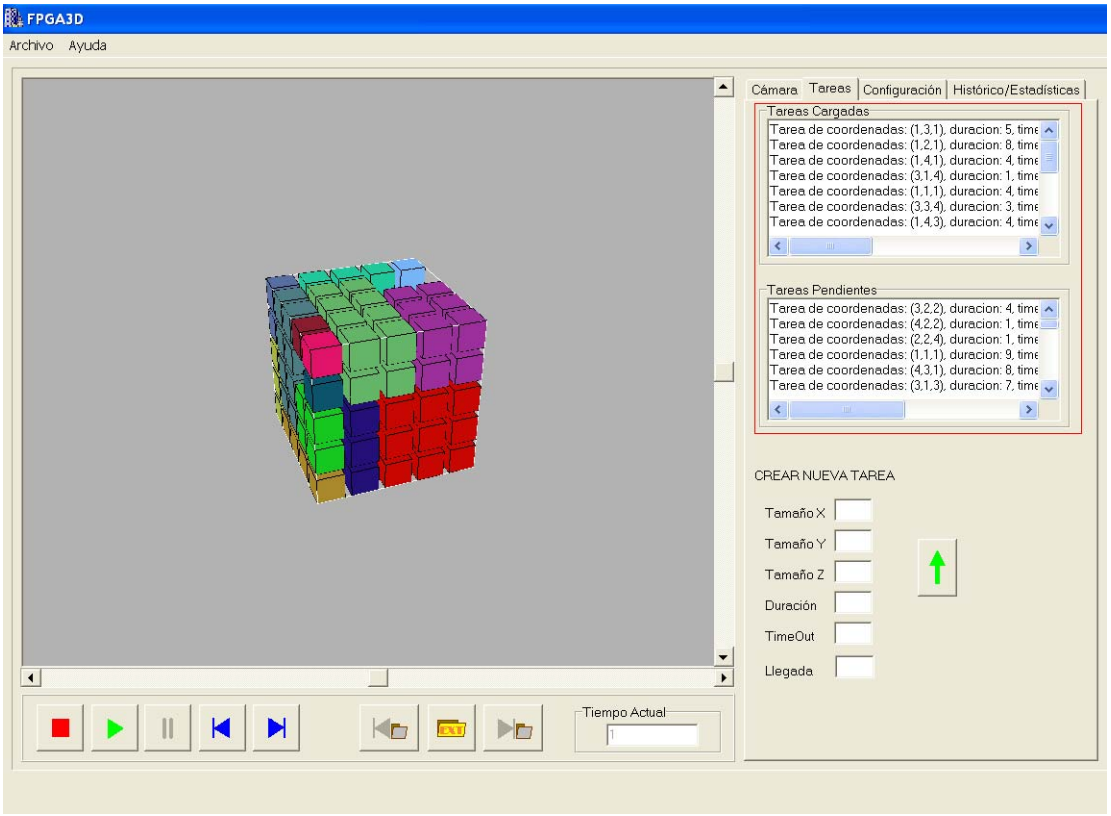


Ilustración 21: Situación de las áreas de texto correspondientes a las listas de tareas

En las listas se visualizan las tareas mediante un mensaje descriptivo de cada una de ellas y en el caso de la lista de tareas cargadas el mensaje indica si la tarea se ha

cargado o rechazado y en caso de haber sido cargada se indica el algoritmo utilizado para ubicarla (los algoritmos disponibles se explican en el apartado de algoritmos).

Esta pestaña (tareas), también nos ofrece la posibilidad de añadir mas tareas a la lista de tareas pendientes, de manera que tras introducir los valores correspondientes en los campos de texto y pulsar el botón para añadir, pasaremos a visualizar la nueva tarea en la lista de tareas pendientes, y se tendrá en cuenta esta tarea en el siguiente instante de tiempo.

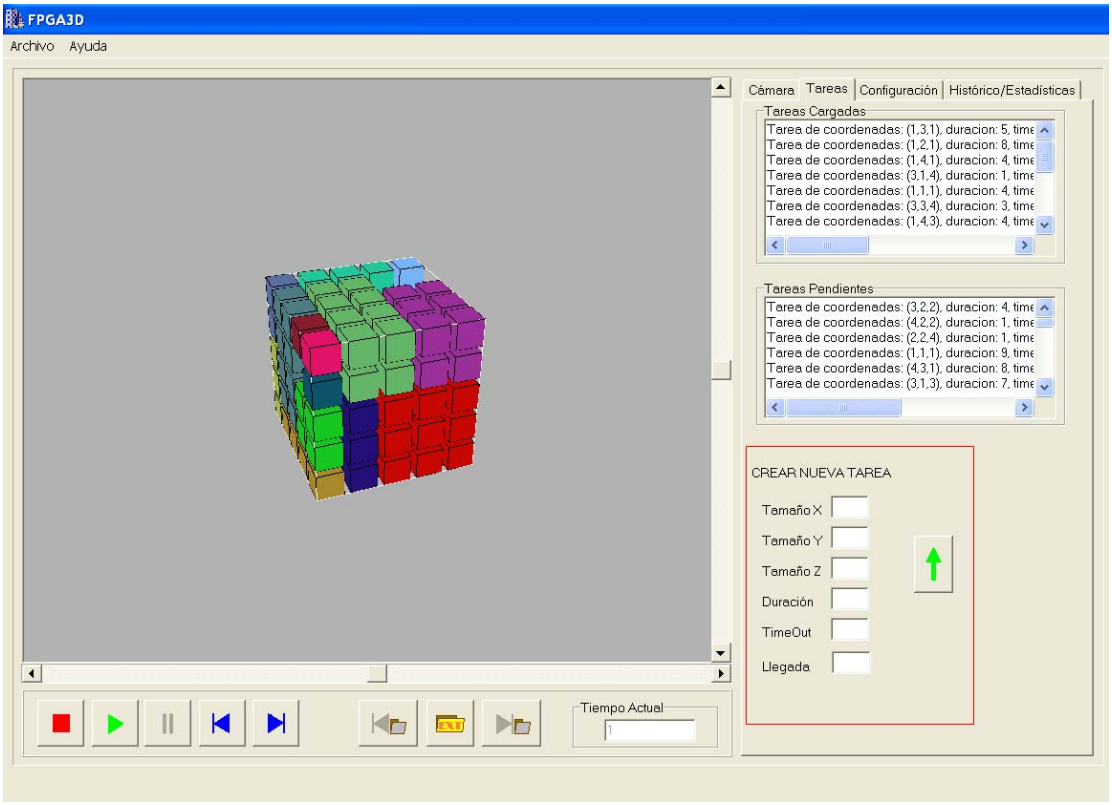


Ilustración 22: Situación de la opción crear nueva tarea

6.4.3 Pestaña Cámara

En esta pestaña tenemos diversas opciones correspondientes a las características de la imagen que se representa en la parte central de la ventana del simulador (estado actual de la FPGA3D).

Las opciones son:

- Rotación x
- Rotación y
- Rotación z
- Zoom

Estas opciones permiten rotar la matriz que representa a la FPGA3D y ampliar o reducir su tamaño dentro de la imagen, de manera que se mejore la visualización de la matriz y se permita ver todas las casillas o CLB's de la FPGA para comprobar su estado.

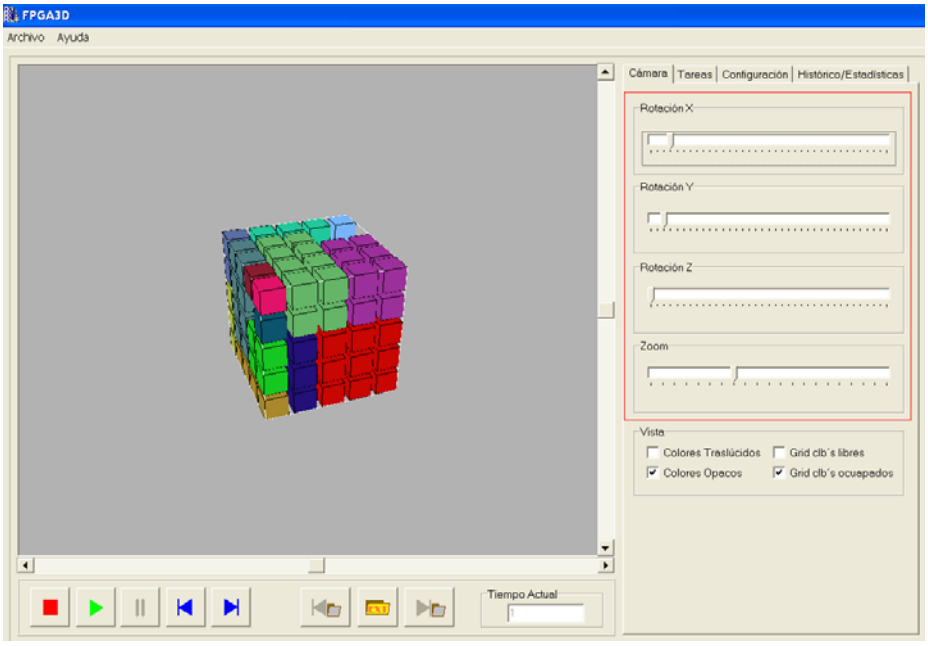


Ilustración 23: Situación de la pestaña Cámara dentro de la ventana del simulador

Además también podemos rotar y ampliar/reducir el tamaño de la FPGA3D visualizada en la imagen del simulador desde el teclado con las teclas a, w, s, d y g, p respectivamente

6.4.4 Pestaña Configuración

Esta pestaña nos ofrece distintas opciones para configurar el tamaño de la FPGA, las tareas a cargar en la FPGA y otras características de la simulación como los algoritmos con los que ubicar las tareas y otras opciones de visualización de la simulación.

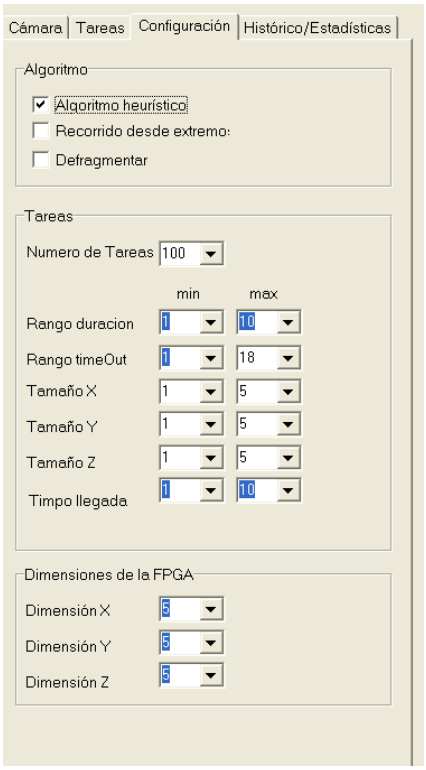


Ilustración 24: Pestaña de configuración

La primera de las opciones que nos ofrece esta pestaña es la configuración del algoritmo que se va a aplicar en la ubicación de las tareas y la posibilidad de aplicar defragmentación o no tras la finalización de alguna tarea.

Las características de los algoritmos se describen en el apartado de algoritmos, pero aquí podemos decir básicamente que nuestro simulador tiene implementados dos algoritmos, que son los que se ofrecen al usuario en esta pestaña:

- Algoritmo heurístico
- Algoritmo de recorrido desde los extremos

La opción de defragmentar se explica también en un apartado aparte. Pero se basa principalmente en reubicar las tareas cargadas en la FPGA en el instante en que alguna de las tareas finaliza su ejecución para evitar la fragmentación de la FPGA y mejorar la ubicación de las tareas pendientes de ubicar.

La generación de las tareas que se intentaran ubicar en la FPGA es aleatoria. Para esta generación de tareas se tienen en cuenta los valores introducidos por el usuario en esta pestaña de configuración del simulador, de manera que el usuario puede seleccionar valores máximos y mínimos que caractericen a las tareas a generar.

Los valores que se pueden configurar y que determinan la generación de las tareas son:

- **Número de tareas**, podemos introducir el número de tareas que queremos intentar ubicar en la FPGA.
- Rango de **duración** de las tareas, podemos introducir valores máximo y mínimo para la duración de las tareas, la duración de cada tarea a generar tomara un valor aleatorio dentro del rango de valores introducido.
- Rango de **time out**, el usuario puede seleccionar valores máximo y mínimo para el time out de cada tarea, este valor se generara aleatoriamente dentro de este rango y se ajustara de manera automática en el caso de que el rango no sea realista, es decir, en una situación real una tarea tendrá un valor de time out superior a la suma del instante de llegada y la duración de la tarea. Por ejemplo, no tiene sentido generar una tarea cuyo time out sea menor que el instante de llegada de la misma.
- **Tamaño** de la tarea en **x**, se pueden introducir valores máximo y mínimo para el tamaño de la tarea en el eje x, estos valores deben ser mayores de 0 y menores que el tamaño x de la FPGA (la tarea a generar debe ser menor o igual a la FPGA para poder ubicarla).
- **Tamaño** de la tarea en **y**, se pueden introducir valores máximo y mínimo para el tamaño de la tarea en el eje y, estos valores deben ser mayores de 0 y menores que el tamaño y de la FPGA (la tarea a generar debe ser menor o igual a la FPGA para poder ubicarla).
- **Tamaño** de la tarea en **z**, se pueden introducir valores máximo y mínimo para el tamaño de la tarea en el eje z, estos valores deben ser mayores de 0 y menores que el tamaño z de la FPGA (la tarea a generar debe ser menor o igual a la FPGA para poder ubicarla).
- Instante de **llegada**, el usuario puede seleccionar los valores máximo y mínimo que determinaran el rango de los instantes de llegada de las tareas a generar.

También podemos configurar el tamaño de la FPGA con la que vamos a trabajar, los valores que podemos introducir son:

- Dimensión X
- Dimensión Y
- Dimensión Z

Por ultimo, en esta pestaña también encontramos algunas opciones de configuración para el aspecto de la imagen que representa el estado de la FPGA en cada instante, son los siguientes:

- **Colores traslúcidos**, los colores de las tareas cargadas en la FPGA son transparentes.

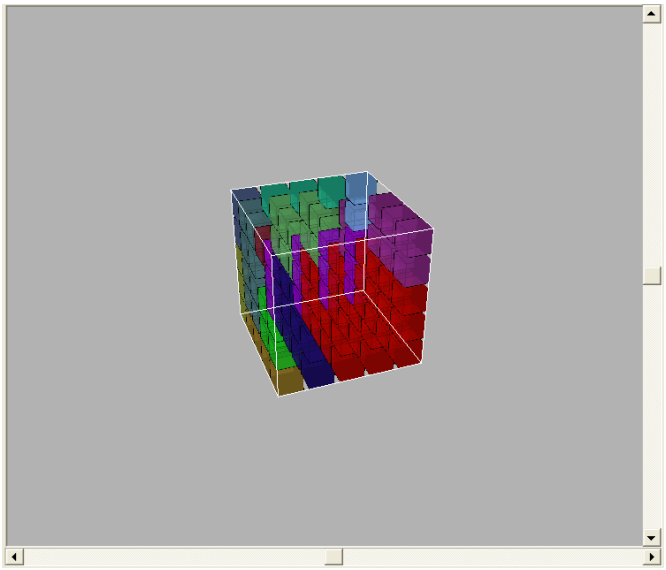


Ilustración 25: Resultado obtenido tras seleccionar la opción de colores traslucidos

- **Colores opacos**, los colores de las tareas cargadas en la FPGA no son transparentes. Esta opción y la anterior son mutuamente excluyentes.

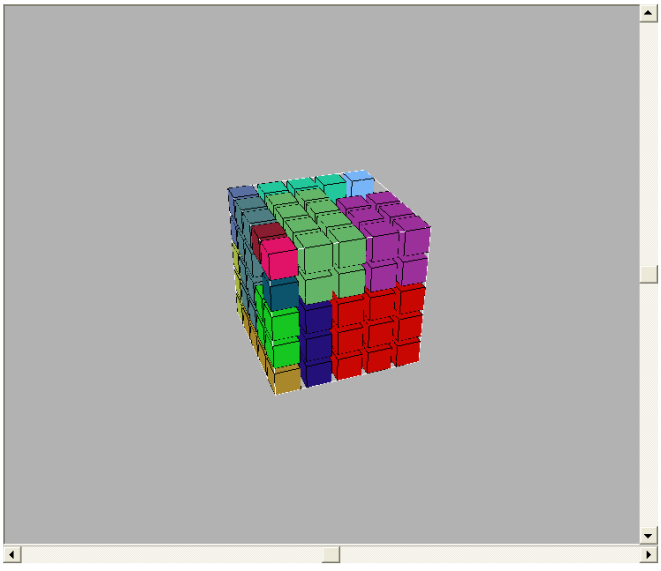


Ilustración 26: Resultado obtenido tras seleccionar la opción de colores opacos

- **Grid clb's libres**, si esta opción esta activada, la imagen que representa el estado de la FPGA3D, tiene una rejilla de color blanco que representa el perímetro de los CLB's (o casillas de la matriz tridimensional) que están libres.

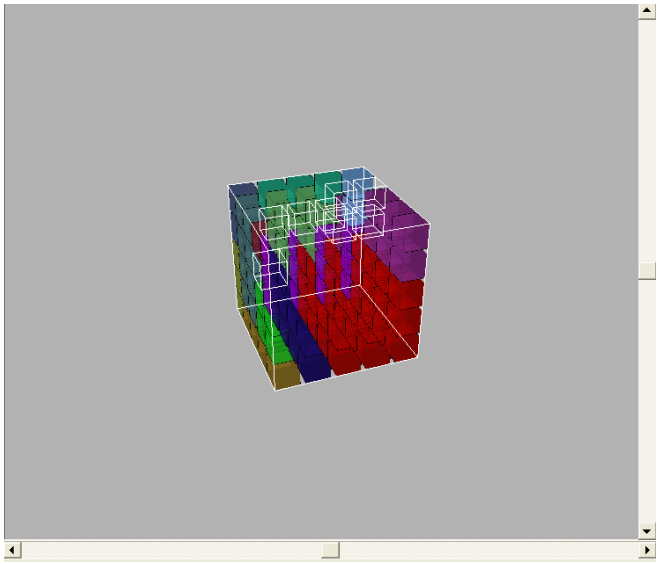


Ilustración 27: Resultado obtenido tras seleccionar la opción de grid clb's libres

- **Grid clb's ocupados**, si esta opción esta activada la imagen en la que se representa el estado de la FPGA3D contiene una rejilla de color negro que representa el perímetro de los CLB's que se encuentran ocupados (el resultado se puede ver también en la imagen anterior).

6.4.5 Pestaña Histórico/Estadísticas

Esta pestaña esta dedicada a almacenar la información correspondiente a los eventos u operaciones que se realizan o tienen lugar durante la ejecución de la simulación.

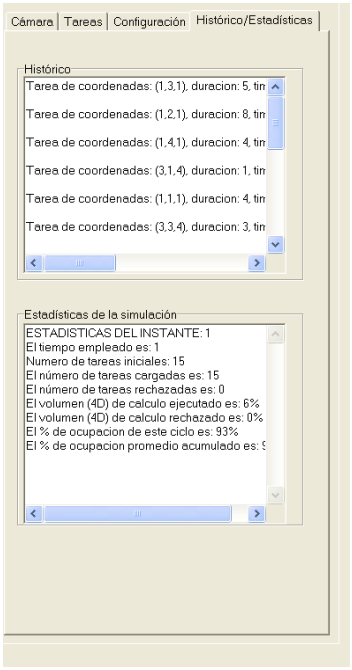


Ilustración 28: Pestaña de histórico/estadísticas

La pestaña se divide en:

- Histórico, área de texto en la que se almacenan los eventos u operaciones principales que tienen lugar en cada instante durante la simulación. Los comentarios que se van añadiendo en el histórico tienen carácter informativo, y están destinados a ayudar al usuario cuando quiere saber qué ha pasado en cada instante de la simulación.
- Estadísticas, área de texto en la que se van mostrando las estadísticas de la ejecución en cada instante de tiempo y se muestran las estadísticas de la ejecución al fin de la misma cuando ya no hay mas tareas en la lista de tareas pendientes y ha finalizado la ejecución de todas las tareas que se habían cargado en la FPGA. De este modo en se muestran estadísticas parciales y finales.

Los datos que se muestran para las estadísticas parciales son:

- **Instante de tiempo**, ciclo actual en la ejecución de la simulación
- **Número de tareas iniciales**, número de tareas que hemos intentado ubicar hasta el instante de tiempo actual en el que se encuentra la ejecución

- **Número de tareas cargadas**, número de tareas que hemos cargado en la FPGA3D hasta el instante de tiempo actual de la ejecución.
- **Número de tareas rechazadas**, número de tareas que han sido rechazadas por haberse pasado su time out o tiempo de ejecución final, es decir tareas que de ser cargadas no pueden finalizar su ejecución en un instante de tiempo anterior a su time out.
- **Volumen 4D de calculo ejecutado**, es la relación entre el volumen 4D de las tareas que se han ejecutado hasta el momento con respecto al volumen 4D del total de las tareas existentes en las dos listas de tareas (cargadas y pendientes)
- **Volumen 4D de calculo rechazado**, es la relación entre el volumen 4D de las tareas que se han rechazado durante la ejecución de la simulación hasta el instante actual con respecto al volumen 4D del total de las tareas existentes en la lista de tareas pendientes y la lista de tareas cargadas
- **% de ocupación en el ciclo**, volumen de tareas que se han cargado en la FPGA3D en el ciclo actual con respecto al volumen total de la FPGA3D
- **% de ocupación promedio acumulado**, es el resultado de ir acumulando el % de ocupación de cada ciclo.

Los datos que se muestran para las estadísticas finales son:

- **Tiempo empleado**, tiempo empleado hasta el fin de la ejecución
- **Número de tareas**, número total de tareas que se han intentado ubicar, son las existentes en este momento en la lista de tareas cargadas
- **Número de tareas cargadas**, numero de tareas que hemos conseguido ubicar durante la ejecución
- **Número de tareas rechazadas**, número de tareas que hemos tenido que rechazar durante la ejecución
- **Volumen 4D de calculo ejecutado**, es la relación entre el volumen 4D de las tareas que se han ejecutado con respecto al volumen 4D del total de las tareas existentes en las dos listas de tareas (cargadas y pendientes)
- **Volumen 4D de calculo rechazado**, es la relación entre el volumen 4D de las tareas que se han rechazado durante la ejecución de la simulación con respecto al volumen 4D del total de las tareas existentes en la lista de tareas pendientes y la lista de tareas cargadas
- **% de ocupación promedio**, es la suma de los % de ocupación de cada ciclo

A continuación pasamos a describir como se ha realizado el cálculo de las estadísticas:

- Tiempo empleado: es el instante de tiempo en el que nos encontramos en el momento de generar la estadística.
- Número de tareas: representa el número de tareas que ya hemos intentado ubicar
- Número de tareas cargadas: total de tareas cargadas hasta el momento
- Numero de tareas rechazadas: total de tareas rechazadas hasta el momento.
- Volumen 4D de cálculo ejecutado: entendemos por volumen 4D de cálculo ejecutado el resultado de aplicar a la simulación en el momento en el que se solicita la estadística, la siguiente formula:

  (tx * ty * tz * duración ejecutadas) / (tx * ty * tz * duración total tareas)

  donde tx, ty y tz son los tamaños de los tamaños de las tareas en los ejes x, y y z respectivamente.
- Volumen 4D de calculo rechazado: entendemos por volumen 4D de calculo rechazado el resultado de aplicar a la simulación en el momento en el que se solicita la estadística, la siguiente formula:

  (tx * ty * tz * duración rechazadas) / (tx * ty * tz * duración total tareas)

  donde tx, ty y tz son los tamaños de los tamaños de las tareas en los ejes x, y y z respectivamente.
- La suma del volumen 4D de cálculo ejecutado y el volumen 4D de cálculo rechazado será 100%
- **% de ocupación en el ciclo:** el % de ocupación para un ciclo determinado será el resultado de calcular:

  (tx * ty * tz * duración ejecutadas) / (tx FPGA* ty FPGA * tz FPGA* tiempo actual)

  donde tx, ty y tz son los tamaños de los tamaños de las tareas en los ejes x, y y z respectivamente y tx FPGA, ty FPGA y tz FPGA son las dimensiones de la FPGA.

  Es el cálculo del volumen 4D de las tareas ejecutadas hasta el momento con respecto al volumen 4D de la FPGA que hemos tenido disponible hasta el momento.
- **% de ocupación promedio:** es la suma acumulada del % de promedio en cada ciclo

Además de las funcionalidades anteriormente descritas, desde el menú de la aplicación se pueden seleccionar las opciones para guardar una simulación en un archivo o cargar una simulación, generada anteriormente, desde un archivo.

En este caso la simulación a guardar / cargar es una simulación generada por nuestra aplicación.

6.5 Integración con terceras herramientas

Ya que una de las funciones principales de nuestro proyecto es la simulación del resultado obtenido, o estado de la FPGA tras aplicar los algoritmos de ubicación de las tareas en cada ciclo, añadimos la funcionalidad de abrir ficheros generados en otras aplicaciones que tengan algoritmos de ubicación para que nuestro proyecto pueda servir de simulador a estas aplicaciones.

Mediante los botones  podemos cargar un fichero generado en una aplicación externa y ver el resultado de la simulación en cada momento.

En este caso solo se tiene en cuenta el estado de la FPGA3D, es decir solo generamos para ficheros externos la visualización de la imagen principal de la ventana que representa a la FPGA3D.

El fichero que importamos debe tener el siguiente formato

- Carácter de inicio de fichero
- Dimensión x de la FPGA
- Dimensión y de la FPGA
- Dimensión z de la FPGA

Estas dimensiones indican el tamaño que tiene la FPGA3D.

Para cada instante de tiempo de la simulación tendremos:

- Carácter delimitador para el tiempo, que indica que después vienen los datos para el tiempo
- Instante actual

Para cada CLB de la FPGA:

- Color R del CLB
- Color G del CLB
- Color B del CLB

Los colores R, G, B son los que utilizamos para la visualización del color de la tarea que esta ubicada en el CLB correspondiente si no esta libre.

- Estado del CLB, que indica si esta libre u ocupado (1 ó 0 respectivamente)
- Carácter de fin de fichero

Lo que hacemos es recorrer el archivo indicado y crear una matriz con las dimensiones indicadas en dicho archivo. Cuando encontremos el instante de tiempo que se corresponde con el actual de la simulación, modificaremos los CLB's de la FPGA3D para que se ajusten a los valores que hay en el archivo.

De esta manera, otra aplicación que tenga implementados algoritmos de ubicación de tareas en una FPGA3D, puede generar un archivo con el formato anteriormente indicado para poder visualizar la simulación de los resultados obtenidos por esos algoritmos desde nuestra aplicación.

7 Anexos

7.1 Simulador Software de FPGA tridimensional

7.1.1 Manual de usuario

Este apartado está destinado a detallar la forma correcta de uso del simulador de FPGA3D mediante un ejemplo.

Al lanzar la aplicación lo primero que hacemos es ir a la pestaña de Configuración para pasar al simulador los valores iniciales para las tareas y la matriz que representa la FPGA.

Los valores para la configuración inicial que hemos introducido para el ejemplo que vamos a describir son los siguientes:

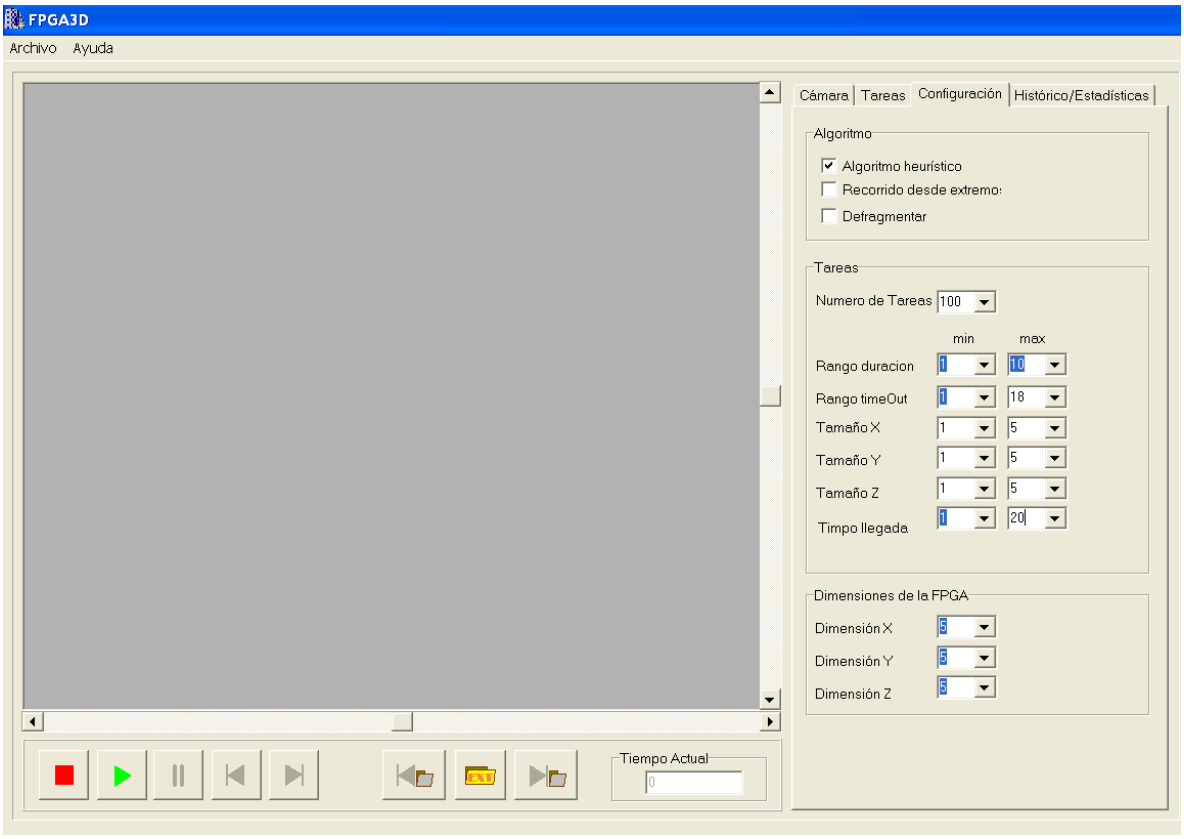


Ilustración 29: Vista inicial

Tras introducir los valores deseados en la pestaña de configuración, se pulsa el botón iniciar  . Tras esta operación se generan las tareas a ubicar en la FPGA, con

valores contenidos en el rango indicado en la configuración y también se crea la FPGA del tamaño indicado.

Además, la ejecución de la simulación comienza automáticamente tras pulsar el botón iniciar. Pero podemos pausarla pulsando el botón pausar  :

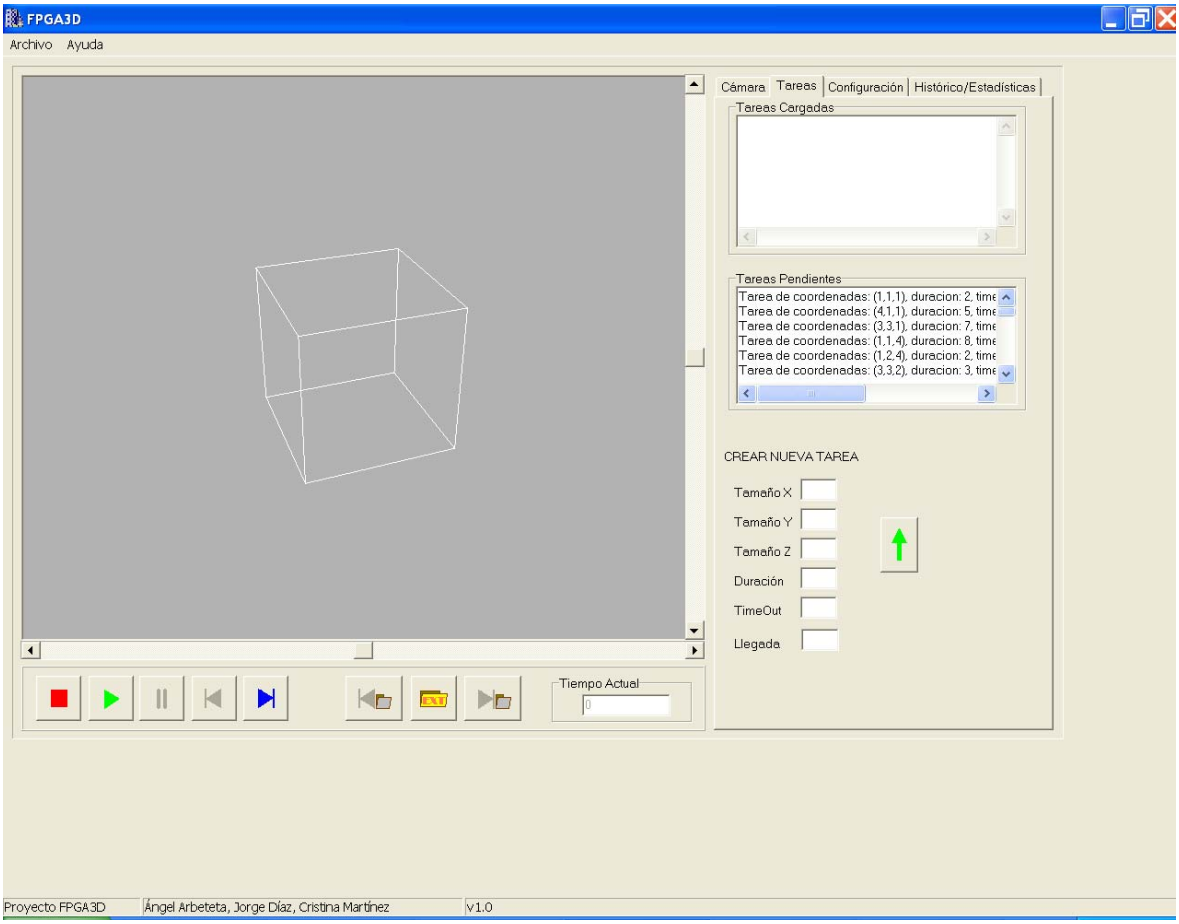


Ilustración 30: Aspecto inicial de la matriz

En la imagen anterior, que es la correspondiente a pausar la simulación tras pulsar iniciar, podemos comprobar que se han creado las tareas con los valores que indica su descripción y se han añadido a la lista de tareas pendientes.

En este punto podemos continuar la simulación pulsando el botón siguiente  , que avanzará un ciclo la ejecución de la simulación o pulsando de nuevo el botón iniciar, con lo que continuara la ejecución automática de la simulación.

Nosotros pulsaremos el botón siguiente:

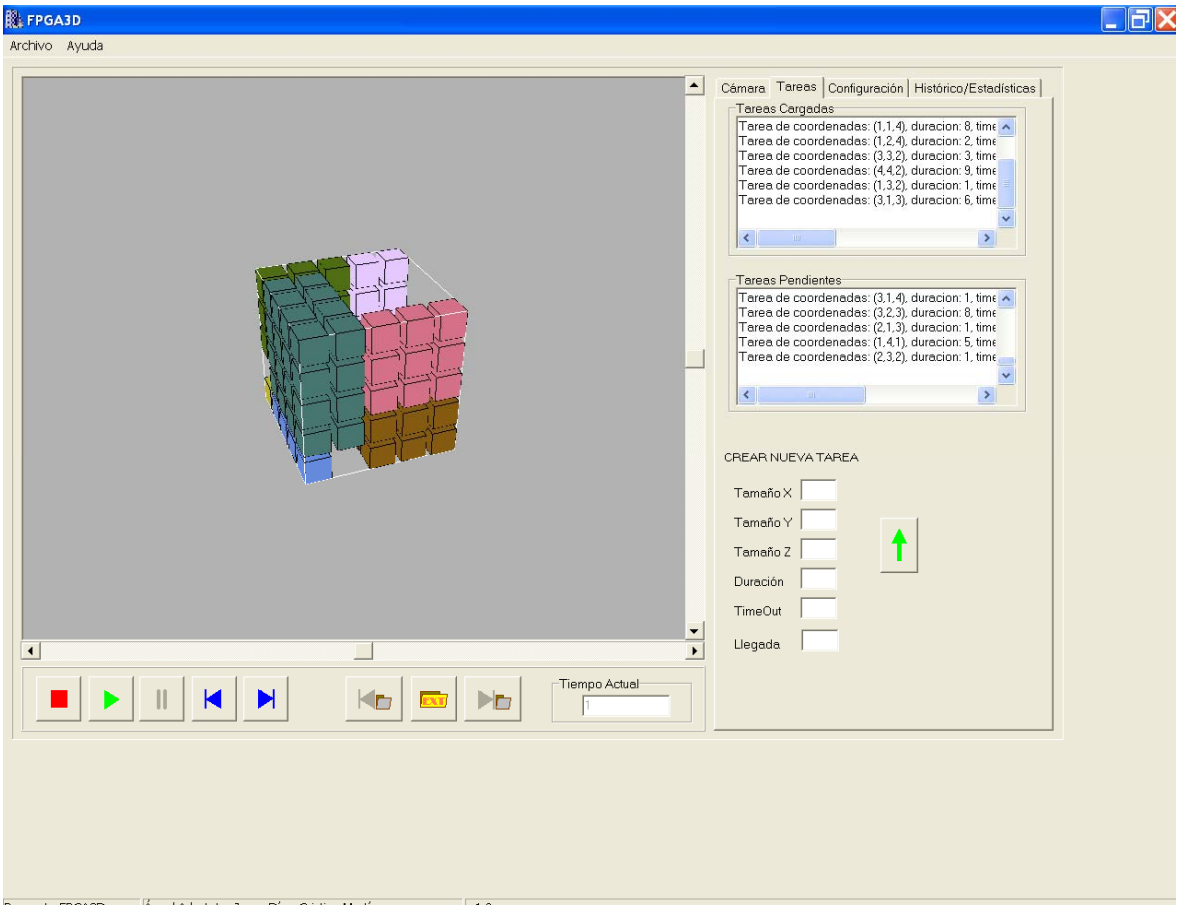


Ilustración 31: Matriz con tareas cargadas

La ventana anterior muestra el estado en el que se encuentra la ejecución de la simulación en el instante 1. Ahora podemos ver que hay tareas cargadas en la matriz y que dichas tareas han pasado de la lista de tareas pendientes a la lista de tareas cargadas. También hay que añadir que se han cargado con el algoritmo heurístico, que es la opción que tenemos seleccionada actualmente, ya que no hemos modificado esta opción en la pestaña de configuración.

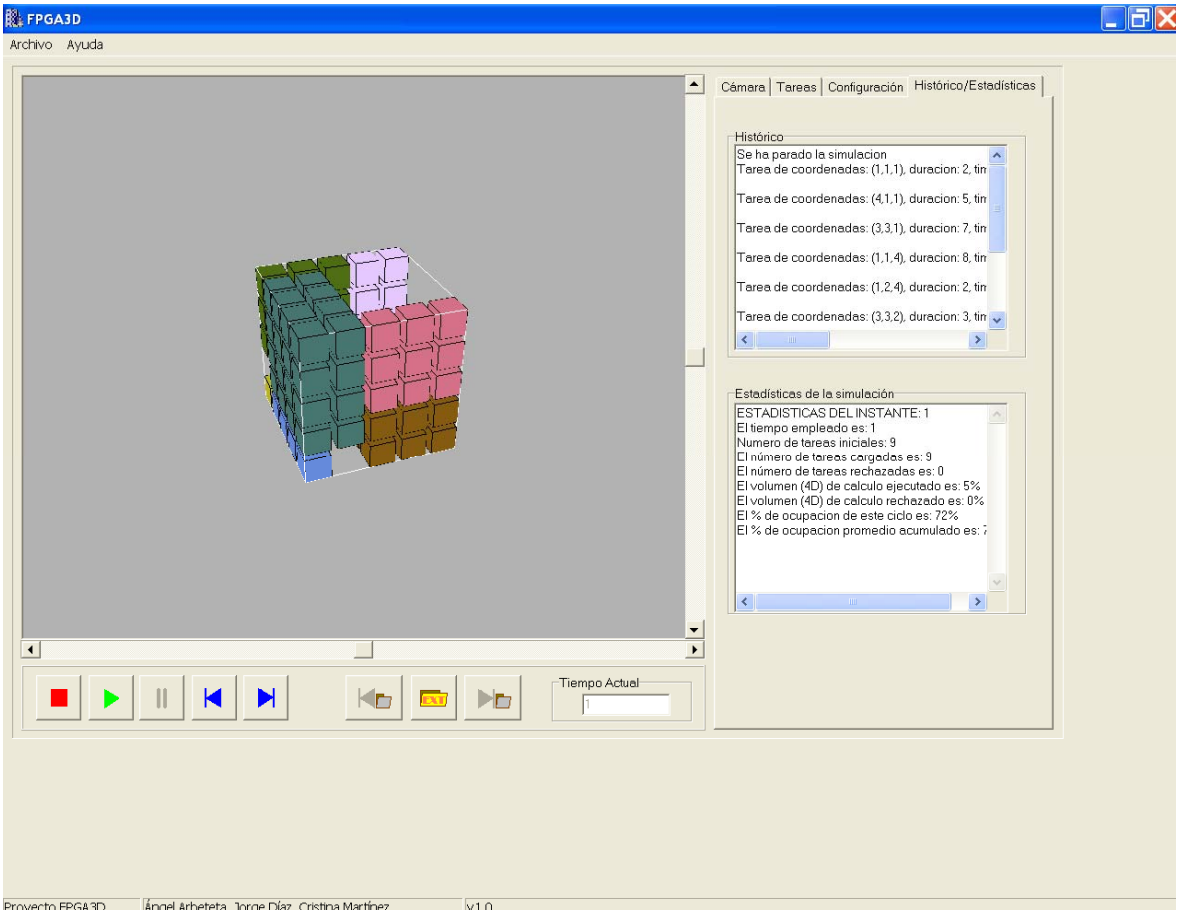


Ilustración 32: Aspecto según van saliendo tareas

En la pestaña de Histórico/Estadísticas de la ventana anterior podemos ver que en las áreas de texto se van añadiendo los datos de la simulación. En el caso del histórico aparecen los datos de las tareas que se han cargado en la FPGA3D. En el área de texto destinada a las estadísticas se han generado las estadísticas del ciclo 1 de ejecución.

En este momento podemos volver al instante anterior pulsando el botón anterior  .

En nuestro caso no pulsaremos este botón ya que volvemos al instante 0 y en ese momento no hay nada cargado en la FPGA3D.

Volvemos a pulsar el botón iniciar para que la ejecución avance automáticamente unos cuantos ciclos:

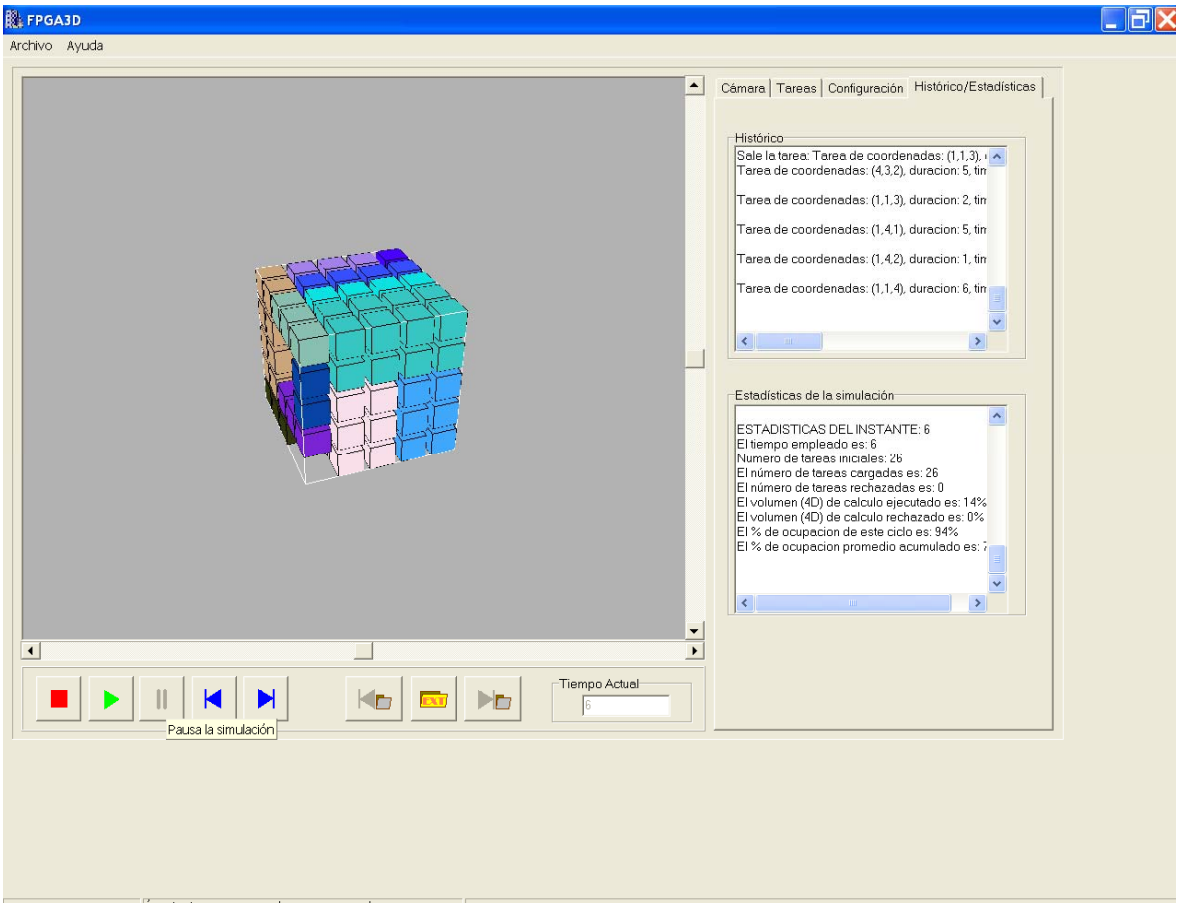


Ilustración 33: Avance automático

Como vemos en la ilustración de arriba, hemos pausado la ejecución en el instante 6. Las tareas que están cargadas en este momento han variado. Los datos de la simulación se siguen cargando en el histórico y podemos ver las estadísticas generadas para cada ciclo.

Ahora vamos a pulsar el botón anterior, para ver el estado de la FPGA3D en el instante 5. El resultado de efectuar esta operación es:

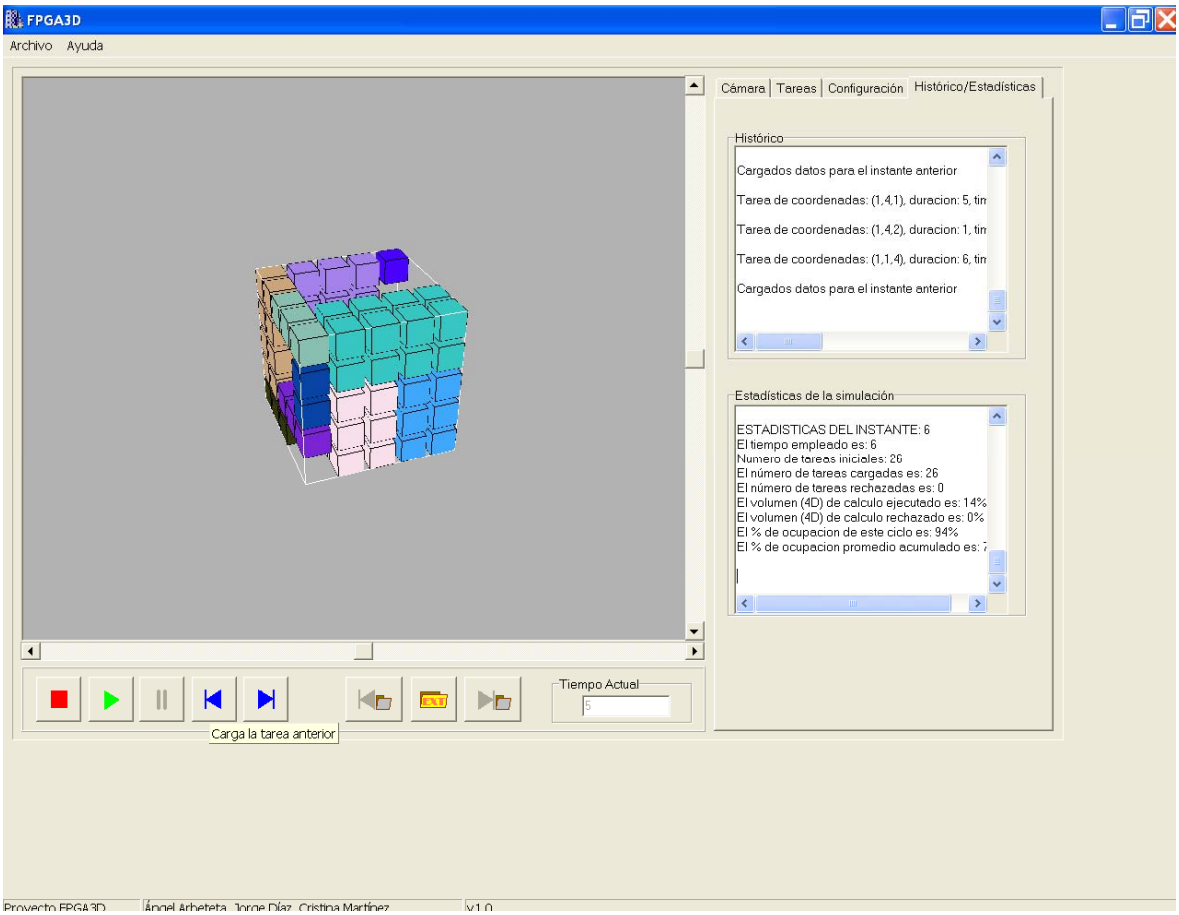


Ilustración 34: Estado anterior

A medida que avancemos en el tiempo de la ejecución, ya sea de forma manual o automática, se irán cargando las tareas que queden en la lista de tareas pendientes o se rechazaran si llega su time out, en ambos casos la tarea correspondiente pasara de la lista de tareas pendientes a la lista de tareas cargadas, con una descripción de la tarea y del caso que ha hecho que pasara de pendientes a cargadas.

Cuando ya no queden tareas en la lista de tareas pendientes y además hayan finalizado todas las tareas que fueron cargadas en la FPGA3D finaliza la ejecución.

Vamos a producir el fin de la ejecución mediante la simulación automática, para ello pulsaremos de nuevo el botón iniciar, obteniendo el resultado siguiente:

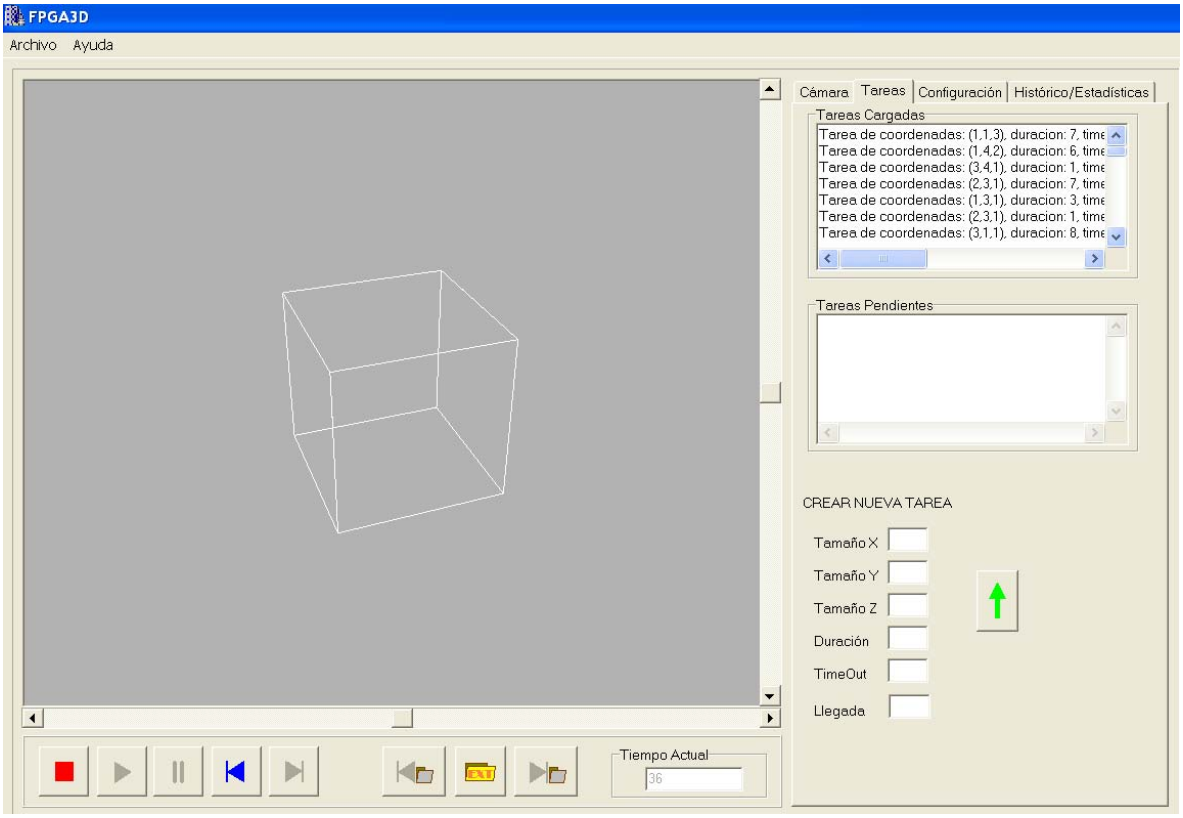


Ilustración 35: Fin de ejecución

La ejecución de nuestro ejemplo ha finalizado en el instante 36. Podemos comprobar que la FPGA3D se encuentra vacía y todas las tareas han pasado de la lista de tareas pendientes a la lista de tareas cargadas.

Además, en la pestaña de histórico/estadísticas se indica el fin de la ejecución en el histórico y el cálculo de las estadísticas finales:

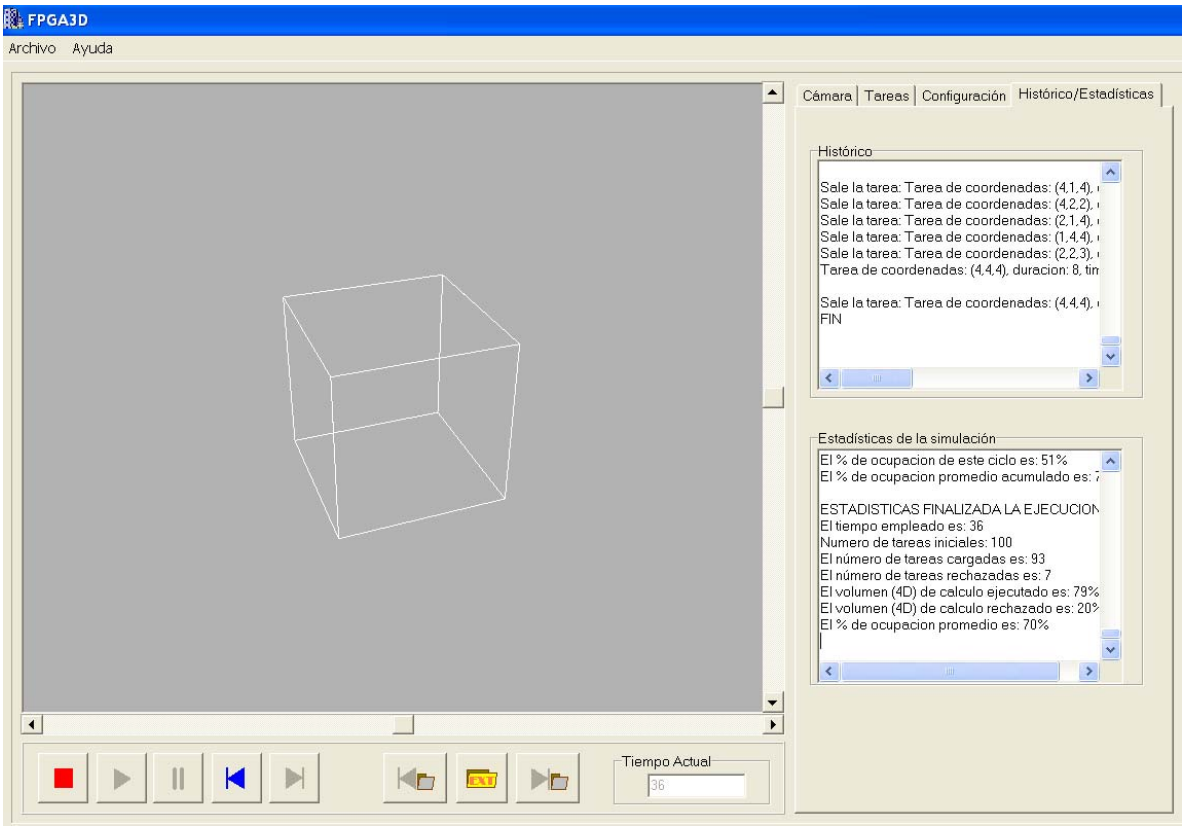


Ilustración 36: Estadísticas finales

Del mismo modo que antes, podemos pulsar el botón anterior tantas veces como queramos para ver el estado de la FPGA3D en los instantes posteriores al fin de la ejecución y así ver paso a paso lo que ha ido ocurriendo en cada ciclo.

Otra de las opciones del simulador es añadir tareas nuevas en la lista de tareas pendientes, esta operación se puede realizar desde la pestaña de tareas. Vamos a añadir una tarea, introducimos los valores que caracterizan a la nueva tarea:

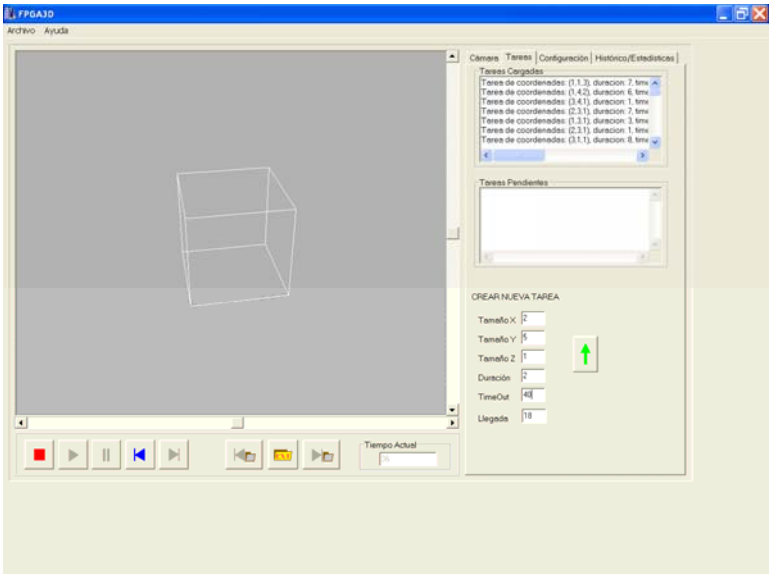


Ilustración 37: Antes de añadir tarea

Tras pulsar el botón añadir  obtenemos los siguientes resultados:

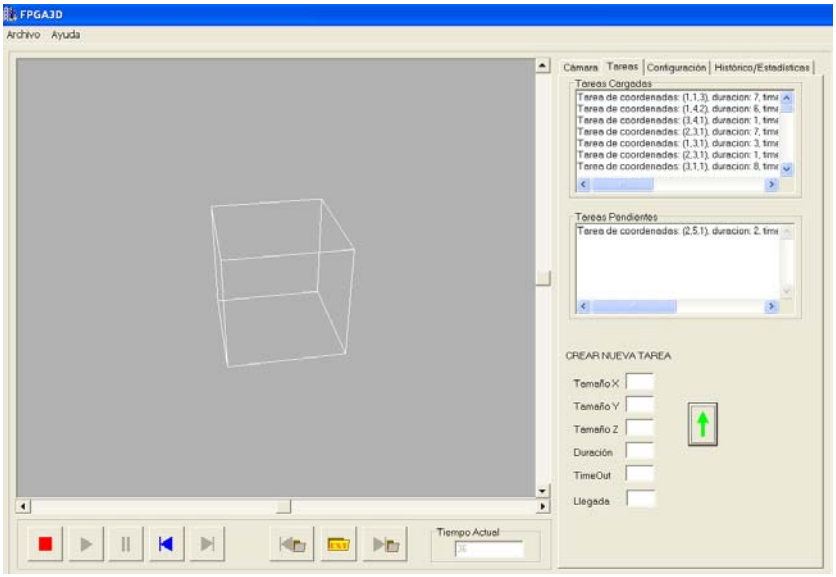


Ilustración 38: Después de añadir la tarea manualmente

Podemos comprobar que se ha dado de alta la tarea con los valores indicados y se ha insertado en la lista de tareas pendientes. Se intentara ubicar en el siguiente ciclo al pulsar el botón siguiente o iniciar pasando de la lista de tareas pendientes a la lista de tareas cargadas en el momento en el que se ubique o se descarte.

A continuación vamos a probar otras funcionalidades que no hemos probado en nuestro ejemplo, pero que también pueden ser útiles en las ejecuciones del simulador.

La rotación puede ser útil en instantes de la ejecución en los que haya varias tareas cargadas en la FPGA3D y queramos ver con detalle en que posiciones están cargadas, volviendo al instante 6 en el que había varias tareas y tras rotar la matriz y ajustar su tamaño se ve lo siguiente:

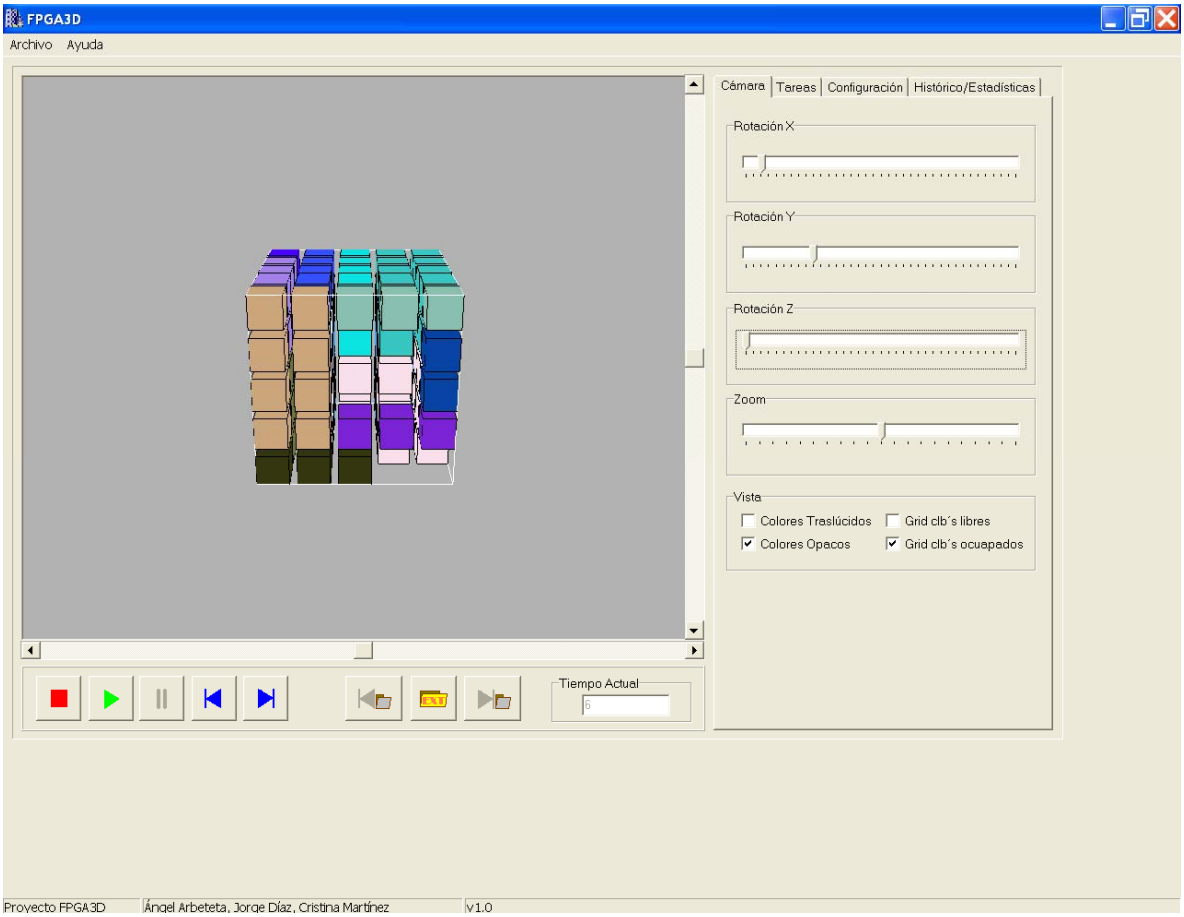


Ilustración 39: Efecto de rotación

Para comenzar una nueva ejecución desde 0 debemos pulsar previamente el botón parar  . Volveremos al estado de la simulación con el que empezamos (se borran todas las tareas y la FPGA3D de la anterior simulación).

Otras opciones del simulador son:

- distintas opciones de visualización de los perímetros de los clb's ocupados/libres y elección de colores traslúcidos/opacos
- ubicación de las tareas con el algoritmo que recorre desde los extremos
- defragmentación de la FPGA3D cuando finalice alguna de las tareas que se encuentran cargadas en la FPGA
- guardar o abrir ficheros que almacenan simulaciones anteriores
- realizar simulaciones de ejecuciones de aplicaciones externas, es decir abrir ficheros que se han generado en otra aplicación para comprobar el funcionamiento de sus algoritmos. Esto se realiza con los botones  .

Estas opciones no las probamos en este ejemplo porque su funcionamiento es similar al descrito anteriormente.

7.2 Aplicaciones futuras del simulador

Una de las posibles aplicaciones futuras del simulador creado puede ser la aplicación del mismo para comprobar el funcionamiento de distintos algoritmos de ubicación de tareas en una FPGA 3D implementados en otras aplicaciones.

7.3 Posibles modificaciones

Una de las posibles modificaciones del simulador puede ser la introducción de nuevos algoritmos de ubicación de tareas dentro de la FPGA.

También se puede ampliar la funcionalidad dirigida al funcionamiento del simulador a partir de 'ficheros de carga externa', es decir ubicaciones de tareas generadas en otras aplicaciones.

7.4 Bibliografía

- Presentaciones del curso de tercer ciclo “Hardware dinámicamente reconfigurable”, *Julio Septién del Castillo y Hortensia Mecha López (2005-2006)*
- "Guest Editor's Introduction: New Dimensions in 3D Integration", *IEEE Design & Test of Computers*
- Introducción a las FPGA, *Cavallero, Rodolfo Antonio. rcavallero@scdt.frc.utn.edu.ar, Gutiérrez, Francisco Guillermo. fgutierrez@scdt.frc.utn.edu.ar*
- Tecnología 3D FPGA: Hardware dinámicamente reconfigurable, *Oscar Ruano*
- Dispositivos lógicos programables, *Miguel Ángel Montejo Ráez (http://www.redeya.com/electronica/tutoriales/pld/pld.htm)*

Autorización

Los alumnos autores de este proyecto autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales, tanto la memoria como el código, la documentación y/o el prototipo desarrollado.

Ángel Arbeteta Hernández

Jorge Díaz Pascual

Cristina Martínez Martínez