ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

PROYECTO FINAL DE CARRERA

Ignacio Traverso Ribón

15 de julio de 2011

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

PROYECTO FINAL DE CARRERA

- Departamento: Ingeniería de Sistemas y Automática, Tecnología Electrónica y Electrónica
- Codirectores del proyecto: Juan Barrientos Villar y Mª Ángeles Cifredo Chacón
- Autor del proyecto: Ignacio Traverso Ribón

Cádiz, 15 de julio de 2011

Fdo: Ignacio Traverso Ribón

Agradecimientos

Me gustaria agradecer y/o dedicar este texto a mi familia por todo el apoyo que me han mostrado durante la realización de mis estudios, a aquellos compañeros que me han hecho más ameno el recorrido y al equipo educativo de la escuela, en especial a Mª de los Ángeles Cifredo Chacón, codirectora de este proyecto.

Agradecimientos también para Raúl Gómez Sánchez, compañero que diseñó desinteresadamente el logotipo de THOr.

Licencia

Notación y formato

Aquí incluiremos los aspectos relevantes a la notación y el formato a lo largo del documento. Para simplificar podemos generar comandos nuevos que nos ayuden a ello, ver `comandos.sty` para más información.

Cuando nos refiramos a un programa en concreto, utilizaremos la notación:
emacs.

Cuando nos refiramos a un comando, o función de un lenguaje, usaremos la notación:
`quicksort`.

Índice general

Indice de figuras

Indice de tablas

Capítulo 1

Introducción

1.1. Antecedentes

El proyecto final de carrera tiene como objetivo completar la formación del alumno en la titulación de Ingeniero Técnico en Informática de Sistemas. De esta manera se ponen en práctica las destrezas y conocimientos adquiridos en las distintas materias, relacionándolos todos e incluso añadiendo otros nuevos que no se han podido adquirir en dichas materias por separado.

Con este proyecto se analiza un problema real y se resuelve de la mejor manera posible ya que, un ingeniero no puede dar una solución, sino que debe dar la mejor. Ésta es una solución eficaz que contemple tanto la eficiencia de la solución como su coste. Este es un coste global y no únicamente económico.

En la realización del proyecto el alumno no se encuentra solo, ya que dispone de los tutores del proyecto, en este caso Dª Mª de los Ángeles Cifredo Chacón y D. Juan Barrientos Villar. Ellos realizan una labor de seguimiento, control, apoyo y guía al alumno en su cometido. De forma que éste tenga alguien a quién recurrir cuando tenga problemas en la realización de su trabajo o le puedan pulir aspectos de su proyecto antes de que éste llegue al tribunal.

Con la culminación exitosa de este proyecto, el alumno alcanza sus motivaciones, metas y objetivos como la preparación para trabajar en una empresa real del campo de la Informática o en un equipo de investigación.

Este proyecto está enmarcado en la línea de trabajo del grupo de investigación "Grupo de Diseño de Circuitos Microelectrónicos" de la Universidad de Cádiz entre cuyos integrantes se encuentran Juan Barrientos Villar y María de los Ángeles Cifredo Chacón, codirectores de este proyecto.

Diversos integrantes del grupo de investigación han detectado la necesidad de un entorno como el que se desarrolla en este proyecto durante la elaboración de sus tesis doctorales, habiendo

desaprovechado mucho tiempo en actividades repetitivas y costosas.

En la búsqueda, por parte del grupo, de soluciones a este problema han encontrado pocas alternativas, teniendo que recurrir a la programación de diversos scripts que resolvieran los problemas más inmediatos que iban surgiendo. Estos scripts tenían diversas limitaciones como problemas de rendimiento, falta de interfaz de usuario o dependencias de software comercial privativo. Esto se debe, fundamentalmente, a la ausencia de un conocimiento más profundo en el campo de la informática por parte de los integrantes del grupo. Otra posibilidad para resolver una parte del problema era *Athena*, un software que permite la evaluación de diseños con FPGAS que tiene el inconveniente de enmarcarse en el campo específico de la criptografía. Esto lo convierte en un software que puede ser utilizado en un ámbito más reducido que el proyecto que se está presentando y por tanto no satisface todas las necesidades del grupo de investigación.

Es por tanto, deber del alumno que realiza este proyecto satisfacer las necesidades de este grupo y solventar los problemas detectados en las soluciones propuestas por los miembros del grupo.

1.1.1. El entorno: THOr

El objetivo principal del entorno es doble. Por un lado se pretende dar servicio a usuarios de FPGAs y diseñadores de circuitos a través de lenguajes formales que quieren analizar qué FPGA se adapta mejor a sus necesidades. Por otro lado, se trata de reducir, aun más, el tiempo de diseño y desarrollo de un circuito en una FPGA mediante el uso de clústers formados por PCs sin grandes prestaciones.

El incremento en el uso de las FPGAs debido a la bajada de su precio en el mercado, el coste que tiene en una empresa el tiempo de prototipado, reducido gracias al uso de FPGAs, o la ausencia de soluciones por parte de los fabricantes a este problema son razones suficientes para justificar la necesidad de este entorno. Además el volumen de modelos de FPGAs en el mercado está en constante crecimiento, habiendo cada vez más niveles de rendimiento, lo que hace inviable un análisis manual de todas las posibles opciones.

Encontramos, entonces, varias partes en el proyecto:

- Una parte que ejecuta el ciclo completo de desarrollo de diversos circuitos en una FPGA ejecutando las fases de síntesis, traducción, mapeado tecnológico, emplazamiento y conexionado y un análisis de tiempo de forma desatendida. Cada fase genera un informe con información relevante para la selección de la FPGA más adecuada. De esta manera se permite la implementación de un circuito sobre una FPGA sin intervención del usuario, lo que ahorra una considerable cantidad de tiempo. Además se reduce el tiempo que se tarda en conocer los datos de los diseños contenidos en los reports
- Otra parte que analiza todos estos ficheros extrayendo los datos relevantes e introduciéndolos en una base de datos para que sean más manejables para usos posteriores.

Fórmula	i	j	k	z	N
$N = i * j * k * z$	16	105	1	5	8400

Tabla 1.1: Número de ficheros según parámetros

- Una última parte que permite la utilización de filtros de búsqueda sobre estos datos para obtener el diseño que proporciona una mejor relación calidad-precio.

Cuando se quieren probar varios circuitos en un conjunto mediano de FPGAs, el tiempo que se tarda en obtener esos datos en forma de ficheros es muy elevado. Hay que tener en cuenta que el número N de ficheros a tratar es $N = i * j * k * z$, siendo i el número de circuitos, j el número de FPGAs, k el número de estrategias empleadas y z el número de fases del ciclo de desarrollo que se ejecutan. Puede verse una ejemplificación de la fórmula en la tabla 1.1

Supóngase que se quieren realizar prototipos de 16 circuitos en 105 FPGAs, con una única estrategia de síntesis y generándose 5 ficheros en cada prototipo. Esto da un total de 8400 ficheros que deben abrirse, analizarse en busca de los parámetros relevantes y cerrarse. Además, algunos de estos parámetros deberán ser convertidos a otras unidades para mantener coherencia entre los datos facilitados por los distintos fabricantes. Como el tiempo es una magnitud que, en el ámbito empresarial, se puede transformar en dinero, la reducción de éste aporta beneficios indudablemente. Además el tiempo de salida al mercado de un producto puede ser crítico en el éxito de éste frente al que ofertan las marcas competidoras. De hecho hay marcas que han conseguido que sus productos sean los líderes del mercado simplemente por haberlo sacado antes que las demás.

Con esta selección se pretende conseguir que el usuario escoja la FPGA que mejor se adapte a su circuito y base su decisión en parámetros y medidas tangibles, con una mayor precisión y probabilidad de éxito en su decisión. Para ello la aplicación ofrecera filtros para que el usuario pueda restringir los resultados y visualizar únicamente el conjunto de implementaciones que le interesan. De esta manera se puede completar el ciclo de desarrollo y la selección de la tecnología y dispositivo pertinentes en el menor tiempo posible y de la forma más económica.

Por otra parte, la extracción de datos de los ficheros generados por las herramientas anteriormente mencionadas y su introducción en una base de datos abre la puerta a los usuarios a crear nuevas aplicaciones que pueda utilizar dichos datos de una manera mucho más sencilla y transparente.

1.2. Objetivos

Con este proyecto se desean alcanzar los siguientes objetivos:

1. Reducir el tiempo necesario para completar el flujo de diseño con FPGAs para un número

determinado de circuitos y tecnologías de forma desatendida utilizando el potencial de PCs que se encuentren inutilizados.

2. Presentar los datos que el proceso anterior arroja de una manera más amigable y cómoda para el usuario, evitándole tener que abrir cada informe y localizar manualmente en ellos los distintos parámetros.

3. Ayudar al usuario a escoger la FPGA que mejor se adapte a su circuito o circuitos al menor coste posible, sabiendo rápidamente cual será aquella que permita una mayor frecuencia de funcionamiento y menor precio a partir de los datos leídos según el punto anterior.

4. Facilitar la creación de nuevas aplicaciones que utilicen los datos que se extraen de las implementaciones.

5. Demostrar que las aulas de informática y laboratorios de las universidades son clústers en potencia que podrían utilizarse cuando no fuera necesario utilizarlos para la docencia. Este uso ahorraría dinero a la universidad y facilitaría las tareas de algunos equipos de investigación, no teniendo que esperar turno para utilizar el clúster corporativo de la universidad en cuestión.

6. Independencia del sistema operativo para la segunda parte del proyecto.

7. Aumentar el valor de la herramienta gratuita *Xilinx ISE Webpack*, la cual no presenta paralelización y es de uso obligatorio cuando se trabaja con FPGAs de Xilinx [1]. Este proyecto permitirá repartir diversas tareas, que deberán ser ejecutadas por este programa, entre varios nodos. De esta manera se utiliza *Xilinx ISE Webpack* de una manera más eficiente.

Además de estos objetivos específicos, existen otros generales que se han cumplido durante la realización de este proyecto como:

- El uso de herramientas de libre distribución: Todas las herramientas, salvo el programa *ISE webpack*, son software libre. *ISE webpack* es gratuita aunque su código no es público.
- La utilización de lenguajes modernos, como Python, y nuevas tecnologías como Doxygen o RMI.
- Es una aplicación facilmente actualizable que podría ser mantenida en el futuro por una comunidad de usuarios.

1.3. Definiciones, acrónimos y abreviaturas

- **VHDL**: Representa la combinación de los acrónimos VHSIC y HDL.

[1] Xilinx es el primer fabricante mundial de FPGAs

- **VHSIC**: Very High Speed Integrated Circuit (Circuito integrado de muy alta velocidad).
- **HDL**: Hardware Description Language (Lenguaje de descripción de hardware).
- **FPGA**: Field Programmable Gate Array. Dispositivo semiconductor que contiene bloques lógicos configurables por el usuario mediante un lenguaje de programación especializado.
- **IEEE**: Institute of Electrical and Electronic Engineers (Instituto de ingenieros eléctricos y electrónicos).
- **ASIC**: Application-Specific Integrated Circuit (Circuito integrado de aplicación específica).
- **NFS**: Network File System (Sistema de ficheros en red).
- **Pyro**: Python Remote Objects (Objetos remotos de python).

1.4. Desarrollo del calendario

Como se observa en la Figura 1.1 el proyecto ha tenido una duración total de 8 meses, comenzando en septiembre de 2010 y acabando a finales de mayo de 2011.

El esfuerzo empleado para la realización de las tareas es directamente proporcional a la duración de las mismas. De esta manera, las tareas que más esfuerzo han requerido han sido la instalación del cluster y las pruebas de las distintas versiones de la aplicación. En cambio fue necesario un menor esfuerzo para la ampliación del cluster o la integración de los scripts con la aplicación. El primer caso es debido a que, gracias al algoritmo elegido para el clúster de alta productividad la inclusión de nodos adicionales se simplifica considerablemente, teniendo que, en el caso de las librerías MPI era necesario escribir una IP en un archivo, instalar unos paquetes desde los repositorios en el nuevo nodo e incluir el nuevo nodo el sistema NFS. En cambio, en los scripts de Python sólo era necesario compartir un sistema de ficheros entre todos los nodos.

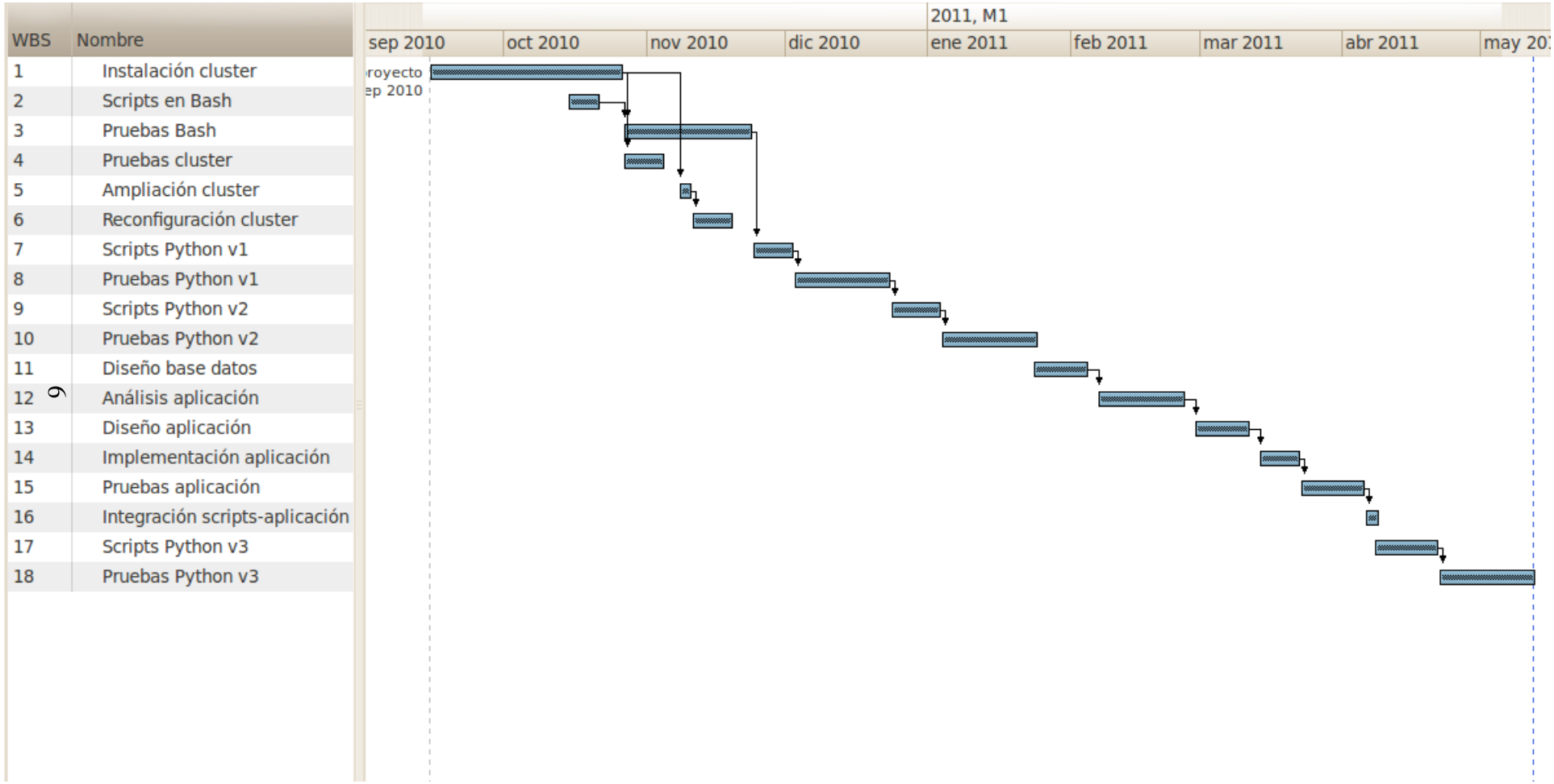

Figura 1.1: Diagrama de Gantt del proyecto

Capítulo 2

Conceptos y tecnologías

2.1. Clústers

2.1.1. ¿Qué es un clúster?

Un clúster es un conjunto de ordenadores cada uno con su hardware y software independientes que pueden actuar como si de uno solo se tratase.

En función de su arquitectura pueden ser:

- Simétricos: Todos los nodos son igualmente importantes y trabajan en paralelo en un determinado trabajo.
- Asimétricos: Un ordenador actúa de maestro controlando al resto de nodos para equilibrar la carga de trabajo.

Estos ordenadores tienen la cualidad de ser totalmente escalables puesto que siempre se pueden añadir más ordenadores. También se han utilizado en sistemas donde la disponibilidad es fundamental ya que la caida de un nodo, siempre y cuando no sea el maestro, no impide el correcto funcionamiento del resto del sistema.

2.1.2. ¿Cómo surgieron?

La aparición de microprocesadores de bajo coste y la necesidad de mayor capacidad de computación hicieron que los usuarios se las ingeniaran para combinarlos de forma que su conjunto actuara como uno solo pero con mayor capacidad de computación.

2.1.3. Clasificación

Existen tres tipos de clúster:

- Alto rendimiento: Su única finalidad es resolver problemas que requieren una gran capacidad de cálculo.
- Alta disponibilidad: Tienen como objetivo ofrecer una alta confiabilidad, seguridad y disponibilidad de sus servicios.
- Alta eficiencia: Son clústers construidos para ejecutar en un período de tiempo el máximo número de tareas posible.

Se debe mencionar que los clústers no tienen por qué estar encasillados en alguna de las tres posibilidades anteriores, sino que pueden ser una combinación de ellas.

En la figura 2.1 puede verse un diagrama de la red formada por el clúster que se montaría.

2.2. FPGAs y HDL

2.2.1. FPGAs

Una FPGA o Field Programmable Gate Array es un circuito integrado que contiene bloques que son capaces de implementar funciones de lógica multinivel y programables por parte del usuario mediante, preferentemente, un lenguaje de descripción de hardaware. Cada uno de esos bloques está conectado mediante interconexiones programables por parte del usuario.

Antecedentes históricos

El comienzo de la industria de las FPGA tiene lugar en la década de los ochenta. Las FPGAs que conocemos hoy día, surgen de los experimentos de Ross Freeman y Bernard Vonderschmitt, ambos fundadores de Xilinx. Ambos desarrollaron en 1985 la primera FPGA de la historia, la XC2064. Las FPGAs surgen como una evolución de la tecnología CPLD y PAL.

Los años noventa fueron el periodo de consagración de la tecnología FPGAs en el mercado y de su evolución tanto en tecnología como en tamaño. Las FPGAs comenzaron a usarse en diversos campos como las telecomunicaciones o la automoción.

En la actualidad las compañias Xilinx y Altera copan el mercado, siendo la primera la más potente con un 51,2 % de las FPGAs del mundo.

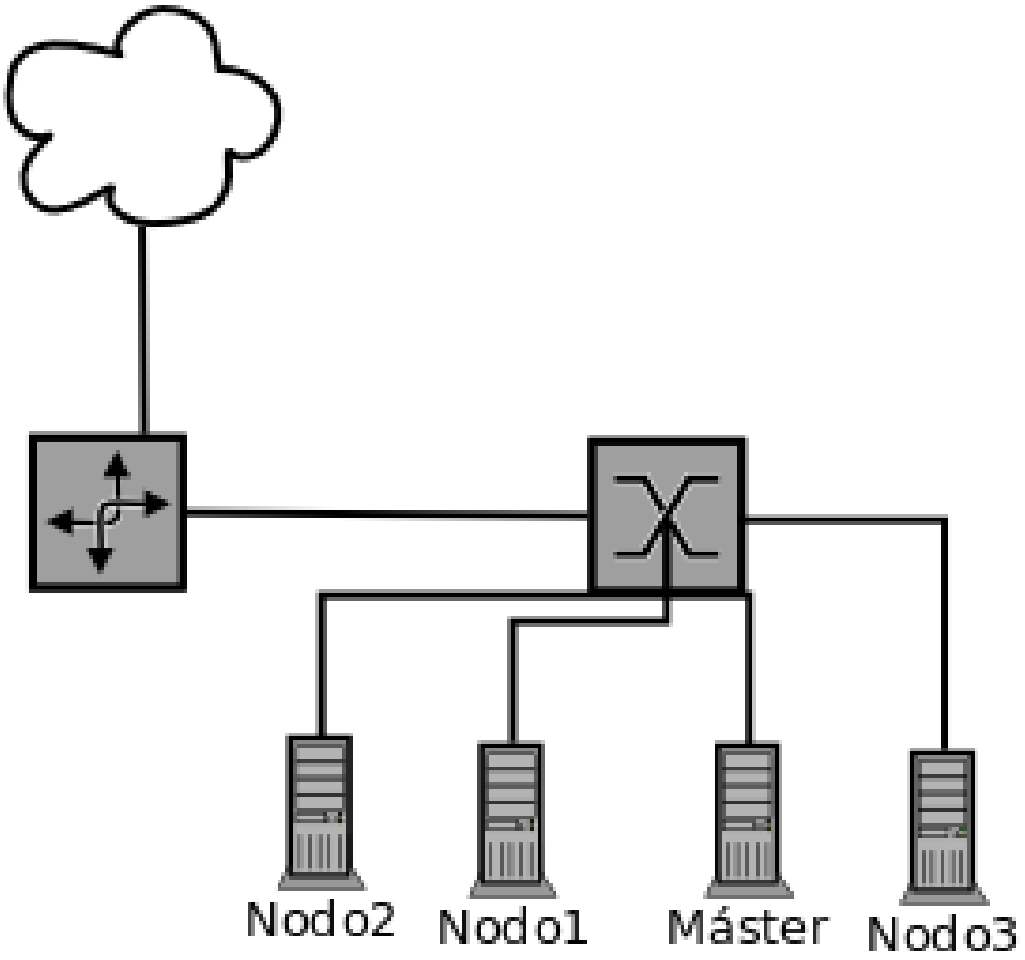

Figura 2.1: Diagrama de la red del clúster

Recursos de una FPGA

Las FPGA están compuestas por tres recursos principales y otros adicionales:

- Bloques de lógica
- Recursos de interconexión
- Bloques de entrada/salida.
- Recuros adicionales (Multiplicadores, memoria embebida o DSP).

Los bloques de lógica son aquellos que pueden ser configurados con una determinada función lógica. Normalmente, en una FPGA, los bloques están compuestos por unos elementos denominados Slices. A su vez, un Slice está compuesto por celdas lógicas, las cuales son definidas a continuación: Las celdas lógicas están compuestas de los siguientes elementos:

- LUT.
- Multiplexores.
- Biestables.

Una LUT es una tabla de valores preasignados. Durante la programación, la LUT almacena los valores de todas las posibles combinaciones de entrada para la función lógica programada, asignándole a cada uno de ellos una dirección.

En cuanto a los biestables, se usan para almacenar las salidas de cada slice para que el dato que se transmita entre ellos sea estable.

Características

La principal característica de una FPGA es la capacidad de ser reprogramable, de modo que permite al programador revisar y mejorar los circuitos electrónicos que se van a implementar. Esto contrasta con los ASIC que sólo pueden ser programados una vez. Los ASIC son confeccionados a partir de una máscara fija que los dota de una funcionalidad exclusica. Cualquier mejora o modificación, por simple que sea, obliga a fabricar de nuevo el ASIC, lo que supone un elevado coste y demora de su salida al mercado.

La programación de una FPGA se realiza a nivel de interconexiones de bloques de lógica.

Programación de las FPGA

La programación de una FPGA puede ser realizada mediante leguajes de descripción de hardware (HDL) o con esquemáticos. Lo normal es usar los lenguajes de descripción de hardware debido a que evitan tener que dibujar todos y cada uno de los componentes del diseño. Además, las herramientas de diseño son capaces de generar visualizaciones de los circuitos a partir de descripciones en HDL. Los lenguajes de descripción de hardware más usados en la actualidad son Verilog y VHDL. A VHDL se le dedicará una sección más adelante.

Los pasos que se siguen en el desarrollo con dispositivos programables FPGA son los que siguen:

- Realizar la descripción del circuito digital mediante un lenguaje de descripción de hardware (HDL).
- Realizar simulaciones funcionales del circuito digital.
- Generar la "netlist" en la que se describen todas las conexiones de la descripción del circuito con los bloques E/S de la FPGA.
- Mediante la herramienta de automatización del fabricante, realizar los siguientes pasos:
  - Síntesis.
  - Traducción.
  - Mapeado tecnológico.
  - Emplazamiento y ruteado.
  - Generar fichero de configuración bit.
- Cargar el fichero de configuración en la FPGA mediante la herramienta que proporciona el fabricante.

Además de todo lo anterior, el programador puede simular su diseño mediante simuladores tanto del fabricante como de terceras compañias para comprobar la correcta funcionalidad de su diseño.

Ventajas e inconvenientes de las FPGA

Las FPGA tienen múltiples ventajas con respecto a los circuitos ASIC.

- Rápida disponibilidad: El prototipado se puede realizar en un corto período de tiempo.
- Herramientas de bajo coste: sólo es necesario el software del fabricante para implementar el circuito y descargarlo en la FPGA.

- Diseños más agresivos: la capacidad de ser reprogramables influye en el desarrollo de diseños más agresivos debido a que no se inutiliza el dispositivo una vez programado el diseño.
- Verificación del diseño más efectiva: las herramientas de automatización del proceso de desarrollo favorecen comprobaciones de la corrección del diseño de manera más rápida y efectiva.
- Procesamiento paralelo.
- Desarrollos más económicos.

Como inconvenientes, presentan los siguientes:

- Mayor tamaño que otras tecnologías.
- Más lentas que otras tecnologías.
- Pueden inducir procesos de desarrollo basados en ensayo-error debido su reprogramabilidad.
- Tradicionalmente han tenido un mayor coste, aunque la situación está cambiando y cada vez son más baratas.

Aplicaciones

Las FPGA están siendo utilizadas en diversos campos entre los cuales podemos destacar:

- Procesamiento digital de señales
- Telecomunicaciones
- Sector aeroespacial
- Sistemas de defensa
- Automoción
- Prototipado de circuitos ASIC
- Reconocimiento del habla
- Criptografía
- Emulación de hardware

Todos los sectores anteriores necesitan gran capacidad de procesamiento paralelo además de capacidad de reconfiguración. Ambas son las principales características de las FPGA y de ahí su auge en estos últimos veinte años.

2.2.2. Lenguaje VHDL

VHDL, acrónimo de VHSIC (Very High Speed Integrated Circuit) y HDL (Hardware Description Language) es un lenguaje de descripción y simulación de hardware definido por IEEE[1] usado para la descripción de circuitos digitales. La gran mayoría de esos circuitos digitales serán luego implementados sobre FPGA aunque el lenguaje también se usa para describir de forma genérica sin que sea necesario luego implementarlo sobre una FPGA.

La principal ventaja del VHDL frente a otros métodos de descripción de circuitos digitales es la sencilla sintaxis y facilidad de diseño jerárquico a diferencia del uso de esquemáticos los cuales son inviables cuando el tamaño del diseño es considerable.

Formas de diseño

En VHDL podemos describir los circuitos digitales de tres formas distintas:

- Comportamiento: describe el comportamiento del circuito digital. Es la forma más parecida a los lenguajes de programación software debido a que se basa en sentencias secuenciales dentro de un proceso. Las sentencias se ejecutan de forma secuencial mientras que los procesos son paralelos entre sí.
- Flujo de datos: se usan sentencias de asignación que se ejecutan concurrentemente.
- Estructural: se usa para describir jerarquías entre los dos tipos anteriores. Es la que permite interconectar bloques entre sí.
- Las tres formas anteriores se pueden combinar entre sí.

Características

El lenguaje VHDL posee similares características a los lenguajes de programación estructurados. Incluye operadores lógicos, asignaciones y nuevas sentencias propias del lenguaje.

Una biblioteca VHDL es un lugar donde se almacena toda la información relacionada con el diseño que se esté realizando. La biblioteca por defecto de un proyecto se denomina “work”.

[1] Estándar IEEE std 1076

Una de las bibliotecas más importantes a usar con VHDL es la de IEEE que normalmente se incluye en los diseños.

En los paquetes se almacena información sobre tipos, objetos que pueden ser usados en los diseños.

Herramientas

Hay disponible una gran cantidad de herramientas para realizar diseños de circuitos digitales con VHDL entre las que destacan:

- Xilinx ISE.
- Altera Quartus II
- Labview
- Matlab
- Simulador GHDL
- Modelsim

2.2.3. Verilog

Verilog es un lenguaje de descripción de hardware utilizado para describir circuitos digitales. Es como VHDL un lenguaje estructurado y está influenciado por el C. De hecho tiene casi las mismas palabras reservadas y una sintáxis muy parecida. La principal diferencia con C es el uso de las palabras begin y end para marcar el inicio y fin de un bloque en lugar de las conocidas llaves.

Al igual que VHDL la ejecución de sentencias en Verilog no es totalmente secuencial. Un diseño en Verilog está compuesto de módulos que tienen sus señales interconectadas. Cada módulo tiene sentencias secuenciales y concurrentes. Un bloque de sentencias secuenciales se enmarca dentro de un bloque begin/end en un determinado módulo y las sentencias concurrentes se encuentran fuera de este marco. Todas las sentencias concurrentes y los módulos begin/end se ejecutan concurrentemente y las sentencias internas a estos bloques se ejecutan secuencialmente.

Verilog es también un estándar del IEEE[2].

[2]Estándar IEEE std 1364

2.3. Lenguajes de programación

2.3.1. Python

Python es un lenguaje de alto nivel multiplataforma que soporta la programación orientada a objetos y la programación imperativa.

Es un lenguaje interpretado, algo que lo hace más lento pero que lo suple con la claridad y limpieza del código que genera. Utiliza tipado dinámico y conteo de referencias, un mecanismo idéntico al *garbage collector* que está implementado en Java.

El lenguaje es administrado por la Python Software Foundation con la licencia Python Software Foundation License, una licencia compatible con la GPL a partir de su versión 2.1.1.

Python fue creado entre finales de los 80 y principios de los 90 en los Países Bajos por Guido Van Rossum como sucesor del lenguaje de programación ABC. El lenguaje está influenciado, además de ABC, por C, Perl, Java o Haskell. En la actualidad es muy utilizado por usuarios de sistemas UNIX e incluso por grandes multinacionales como Google.

2.3.2. BASH

Bash es el lenguaje que se utiliza en la shell Unix que lleva el mismo nombre. Esta shell fue escrita por Brian Fox dentro del proyecto GNU como sustituta de la shell Bourne. Se liberó la primera versión en 1989 y actualmente se encuentra instalada por defecto en los sistemas GNULinux, Mac OS X y Darwin. También se ha exportado para sistemas Windows, aunque no está instalada por defecto.

El nombre no es más que un acrónimo, como la mayoría de los programas de GNU, que significa Bourne-Again SHell refiriéndose a si misma como la sustituta libre de la shell Bourne.

2.4. Base de datos: MySQL

MySQL es un sistema de gestión base de datos relacional, multihilo y multiusuario que, en la actualidad se distribuye bajo la licencia GNU GPL para un uso compatible con ésta o bajo una licencia especial si se va a utilizar en productos privativos. Antiguamente se distribuía bajo GPL sea cual fuera el uso que se le fuera a dar pero la compra de MySQL por parte de Sun y esta a su vez por parte de Oracle cambiaron la forma de distribución.

MySQL está escrito en C/C++ y posee APIs para varios lenguajes entre los que se encuentra Python. Salió al mercado en 1981 por parte de IBM y desde entonces ha crecido más y más

llegando a convertirse en un estándar para las bases de datos relacionales.

La portabilidad de MySQL entre los diferentes sistemas operativos y la rapidez con la que realiza sus operaciones han sido claves en su éxito. Cabe destacar también la gran seguridad que nos ofrece, con una buena gestión de usuarios y claves para mantener siempre un alto nivel de seguridad.

En este proyecto se ha utilizado la versión 5.1.41 de MySQL para procesadores de 64 bits.

2.5. Otros programas o tecnologías utilizados

2.5.1. Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto multiplataformas. Comenzó a programarse en 1995 en lenguaje C y es el servidor web más utilizado en la red desde 1996, alcanzando su máximo en 2005 con un 70 % de los servidores de internet que lo contenían.

Apache es modular y esto lo hace muy configurable por el usuario, que no fácil, ya que no dispone de interfaz gráfica. Podemos dividir los módulos en tres categorías:

- Módulos que contienen las funcionalidades básicas.
- Módulos multiproceso que tienen la responsabilidad de atender las peticiones y unir los puertos de la máquina.
- Módulos adicionales.

Los principales competidores del servidor Apache son Internet Information Services de Microsoft, Sun Java System Web Server, de Sun Microsistems y en algunos casos, otras aplicaciones como Zeus Web Server.

En este proyecto se ha utilizado la versión 2.2.14 de Apache.

2.5.2. Doxygen

Doxygen es un generador de documentación para varios lenguajes de alto nivel como son C, C++, python, Java o PHP. Está escrito en C++ y fue creado por Dimitri van Heesch en 1997. Aunque ha sido desarrollado bajo Linux y Mac OS fue diseñado para ser portable y existen ejecutables disponibles para Windows.

Doxygen crea la documentación a partir de comentarios en el fichero fuente. La salida se puede producir en varios formatos, siendo HTML y LaTeX los más populares. Varios conocidos proyectos como son KDE, ALSA o LyX emplean Doxygen en su proceso de documentación.

En este proyecto se ha utilizado la versión 1.6.3 de Doxygen.

2.5.3. Ganglia

Ganglia es una herramienta de monitorización de sistemas distribuidos como son los clusters y los grids. Está escrito en C, Perl, PHP y Python y fue concebido en la universidad de Berkeley, California, en el año 2000.

Ganglia se distribuye bajo licencia BSD y actualmente es desarrollado por una considerable comunidad de usuarios. A día de hoy es utilizado para monitorizar miles de clusters en todo el mundo.

Ganglia está compuesto de tres módulos:

- Gmond: Es el demonio de monitorización. Se encuentra instalado en cada nodo del cluster y su labor consiste en consultar parámetros relacionados con el rendimiento y enviarlos al nodo central.
- Gmetad: Se encuentra instalado en el nodo central y se encarga de recolectar los datos que le envían el resto de nodos para almacenarlos.
- Web-FrontEnd: Se encuentra instalado en el nodo central y se encarga de tratar los datos que obtiene Gmetad para mostrarlos al usuario mediante una interfaz web. Dicha interfaz tiene gráficos y un historial de los valores que han ido tomando los parámetros desde un año atrás.

2.5.4. Pyro

Pyro es una tecnología que permite el uso de objetos remotos o distribuidos. Es el equivalente al RMI de Java en Python.

Pyro establece una capa intermedia o middleware que permite implementar una estrategia cliente-servidor sin que el usuario tenga que preocuparse de cómo se comunican a través de la red estas entidades. Esta abstracción simplifica la labor del programador y fomenta la creación de un software de mayor calidad y más fácil de mantener.

Además Pyro incluye la posibilidad de utilizar un servidor de nombres, de manera similar a los DNS de internet. De esta manera apenas hay que trabajar con IPs, sino con nombres, con la simplificación que esto conlleva.

Capítulo 3

Descripción general del proyecto

3.1. Perspectiva del producto

Este proyecto ofrece un entorno que trate de resolver las necesidades planteadas en el grupo de investigación y demandadas durante el desarrollo de las tesis por varios de sus integrantes.

El grupo ya había trabajado en una primera versión de la aplicación que estaba programada en Visual Basic y utilizaba el sistema de base de datos Microsoft Access con unos esquemas que no seguían las formas normales que se deben cumplir al diseñar una base de datos.

El proyecto trata de ofrecer un completo entorno que incluya una mejora en rendimiento y funcionalidad de la primera aplicación, además de acortar el tiempo de evaluación y selección de la tecnología soporte. De esta manera aparece la integración de la aplicación con el clúster y se realiza un nuevo esquema para la base de datos. Además se trata de hacer la aplicación compatible tanto para sistemas Windows como para sistemas GNU/Linux. Por este motivo se ha cambiado de sistema de base de datos a MySQL y de lenguaje de programación a Python.

La aplicación interactúa con el programa *Xilinx ISE Webpack*, una utilidad que permite diseñar circuitos digitales en una FPGA y obtener informes con los datos relevantes en la selección de una FPGA concreta. Esta aplicación es gratuita y esta disponible para Windows y Linux, por lo que no impide la realización de ninguno de los objetivos anteriores. El uso de esta aplicación no es una elección sino un requisito de ésta ya que las FPGAs que se pretenden evaluar sólo son compatibles con este software. Existen entornos de terceros y libres para la realización de las fases de síntetis y simulación. Sin embargo la implementación requiere información reservada, por lo que cada fabricante obliga a trabajar con sus propias herramientas cuando se pretende concluir un diseño digital. En la figura 3.1 pueden verse las fases que deben ejecutarse para completar el flujo de diseño de un circuito en una FPGA.

En la tabla 3.1 pueden observarse la relación entre las distintas herramientas de *Xilinx ISE Webpack* y las fases que se deben completar. La elección de la herramienta de este fabricante

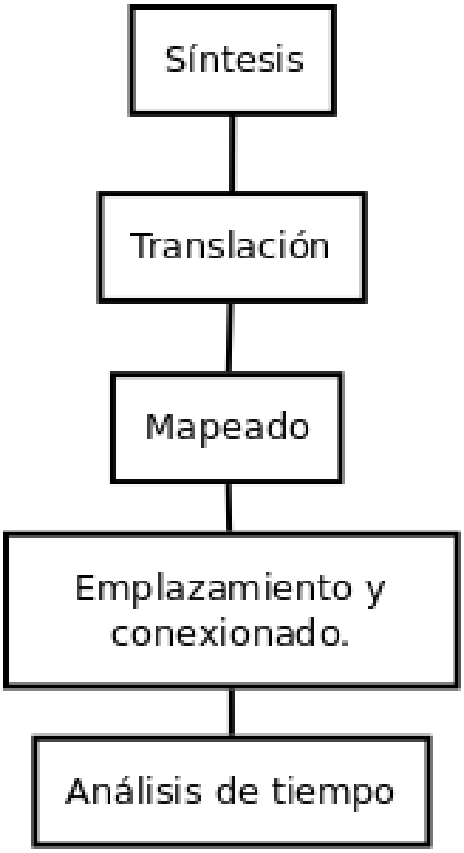

Figura 3.1: Diagrama de flujo de las fases del diseño de circuitos en dispositivos FPGAs

Tarea	***Xilinx ISE Webpack***
Síntesis	XST
Translate	NGDBuild
Mapping	MAP
Place and Route	PAR
Análisis de tiempo	TRACE

Tabla 3.1: Relación herramientas-fases de *Xilinx ISE Webpack*

no es trivial. Xilinx tiene en la actualidad el 51,2 % de la cuota de mercado, siendo Altera la que la sigue con un 35,5 %

Con este entorno un usuario podría, entre otras cosas, tener almacenada la información de todos sus diseños y comparar distintas versiones del mismo. De esta manera podría decidir cual es más óptimo según los parámetros extraídos de los reports (frecuencia máxima, área de ocupación, etc). El significado de estos parámetros serán explicados más adelante.

También podría saber qué diseños fallaron y cual fue el motivo: falló el software durante la implementación o la FPGA escogida no cuenta con suficientes recursos para el circuito determinado. Cuando ocurre esto último se dice que la implementación fue sobredimensionada. Esto puede ser por:

- El número de slices necesarios para implementar el circuito es mayor que el que ofrece la FPGA.
- La FPGA no tiene suficientes pines de entrada/salida para el circuito en cuestión.

En la figura 3.2 puede visualizarse un esquema de ésta con los CLBs, los multiplicadores o los bloques RAM:

Si THOr no existiera el proceso que debería realizar un usuario por cada para circuito-FPGA sería el siguiente:

1. Ejecutar las distintas fases para completar un flujo de diseño desde la interfaz de *Xilinx ISE Webpack*.
2. Abrir cada uno de los informes para leer los resultados y conocer cómo se adapta su circuito a la FPGA escogida.

Si la FPGA no tuviera los suficientes recursos para alojar el circuito sólo se ejecutaría hasta la fase de mapping. Esto no se conoce con antelación por lo que hasta que no ejecute las distintas fases no lo sabrá. Con THOr, en cambio, ese resultado podría estar ya almacenado en la base de datos y por tanto saber de antemano y con certeza qué FPGA es la mejor para su circuito.

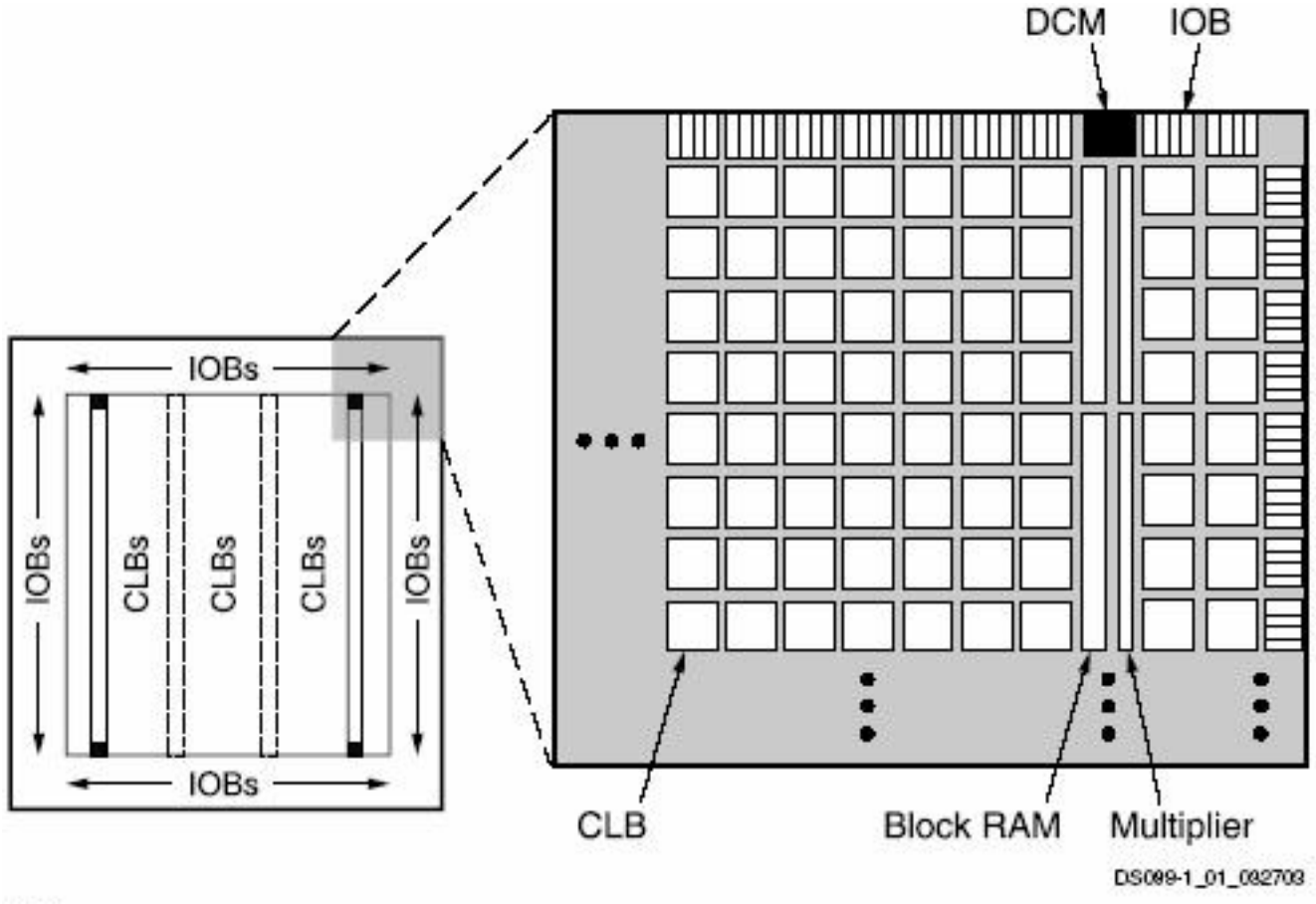

Figura 3.2: Esquema interno de una FPGA. Fuente: Xilinx Inc.

3.2. Limitaciones de memoria

La cantidad de memoria principal que utiliza el entorno es menor que la que necesitan las librerías y aplicaciones auxiliares. Por ejemplo, *ISE Webpack* requiere 3.2GB de espacio en disco y unos 512MB de memoria RAM. Esto no quiere decir que ocupe esta cantidad de memoria, sino que es lo que requiere para ejecutarse en concurrencia con otras aplicaciones. De esta manera el entorno desarrollado deberá poder ejecutarse en concurrencia con *ISE Webpack*, con el intérpetre de Python, el servidor de MySQL, las librerías de GUI WxPython y las librerías Python-MySQLdb para la interacción del intérprete de Python con el servidor de base de datos. Esto quiere decir que en cualquier equipo en el que se pueda ejecutar *ISE*, se debe poder ejecutar *THOr*, ya que este no requiere, apenas, recursos extra.

3.3. Dependencias

La aplicación desarrollada en este proyecto hace uso o depende de:

- Python
- Xilinx ISE Webpack
- MySQL
- Python-MySQLdb
- WxPython

Capítulo 4

Metodología de desarrollo

4.1. Introducción

A principios de los años 70, surgen las metodologías de desarrollo de sistemas informáticos como necesidad para abordar la gran cantidad de problemas que se generaban durante el desarrollo de dichos sistemas.

Las metodologías de desarrollo se han convertido en una parte integral de la Ingeniería del Software cuyo principal objetivo es obtener un software de calidad, esto es legible, ampliable, y correcto.

Una metodología de desarrollo debe ofrecer un marco de referencia para el desarrollo de sistemas software. Éstos sistemas están en permanente evolución y las metodologías no deben ser menos. Esto quiere decir que una metodología de desarrollo no es algo estático, sino que debe ir actualizándose y adaptándose a los nuevos modelos de sistemas software que van apareciendo. Por ejemplo, la última versión de Métrica no está adaptada a la web, algo que deberán corregir en su posterior versión. Entre las metodologías oficialmente adoptadas por la administración pública destacan SSADM, metodología oficial del Reino Unido, Merise, la metodología Francesa, y en España, la MÉTRICA, que es el método de desarrollo de sistemas de información promovido por el Consejo Superior de Informática para ser aplicado por la Administración General del Estado. Al desarrollarse el proyecto en territorio español se escogerá Métrica como la metodología a utilizar.

4.2. Métrica V3

Métrica proporciona a las organizaciones e instituciones que la utilizan un instrumento para la sistematización de actividades que se han de llevar a cabo durante el ciclo de vida de un sistema

software. Los objetivos a alcanzar son los siguientes:

- Proporcionar sistemas software que ayuden a conseguir los fines que tiene la organización o institución en cuestión definiendo un marco estratégico para la definición de los mismos.
- Proporcionar a la organización productos software que satisfagan sus necesidades incidiendo en el análisis de requisitos.
- Hacer los departamentos de Sistemas y Tecnologías de la Información y Comunicaciones más productivos y competitivos a la vez que más adaptables a los cambios y orientándolos a la reutilización, cuando sea posible.
- Facilitar la comunicación y entendimiento entre los distintos participantes en la producción del software a lo largo del ciclo de vida del proyecto
- Facilitar la utilización y el mantenimiento de los productos obtenidos.

La nueva versión de MÉTRICA contempla tanto el desarrollos de los sistemas software como ciertos aspectos de gestión que aseguren el cumplimiento de ciertos objetivos como pueden ser calidad, coste o plazos.

MÉTRICA V3 parte de su versión anterior, de la que conserva su flexibilidad, adaptabilidad, sencillez y estructura de actividades y tareas. Sin embargo se han dividido las fases y módulos en procesos, algo más adecuado para el formato entrada-transformación-salida que se produce en las diferentes divisiones del ciclo de vida del software. Para cada tarea se detallan los participantes que intervienen, los productos de entrada y de salida así como las técnicas y prácticas a emplear para su obtención.

En la elaboración de MÉTRICA V3 se han tenido en cuenta los últimos estándares de la Ingeniería del Software así como los métodos más extendidos. También se han pulido ciertos detalles a partir de la experiencia de los usuarios con la anterior versión.

MÉTRICA V3 cubre en una única estructura los dos paradigmas de programación más extendidos (estructurado y orientado a objetos), facilitando a través de interfaces la realización de procesos de apoyo y organizativos: Gestión de Proyectos, Gestión de Configuración, Aseguramiento de Calidad y Seguridad.

La automatización de las actividades propuestas en MÉTRICA V3 es posible gracias a que sus técnicas están soportadas por una amplia variedad de herramientas de ayuda al desarrollo. Además, para facilitar la utilización de MÉTRICA V3 se ha desarrollado la herramienta software Gestor Metodológico. Una herramienta de ayuda a la aplicación de la metodología todo tipo de proyecto y que permite adaptar la estructura de MÉTRICA V3 de acuerdo a sus características particulares, permitiendo el seguimiento y control de sus actividades y tareas realizadas.

Se ha desarrollado también el software Selector de Herramientas, que ayuda a seleccionar entre las herramientas CASE aquella que mejor se adapta a las necesidades del proyecto concreto. Así mismo se ha elaborado un curso de autoformación que sirve para aprender todos los conceptos y elementos de la metodología MÉTRICA V3 contemplando varios niveles y diversos perfiles. Tanto la metodología como todas estas herramientas están disponibles en la web del Consejo Superior de Informática.

4.2.1. Procesos de Métrica V3

El desarrollo de este proyecto exige una formalidad y una metodología de desarrollo bien definida. Se ha escogido la metodología Métrica V3 al ser necesaria su aplicación en cualquier proyecto dependiente de la Administración Pública Española, ser la más extendida en el territorio dónde se ha desarrollado el proyecto y unir en una misma estructura el paradigma estructurado y el paradigma orientado a objetos, ya que la aplicación desarrollada combina ambos.

Gracias la adaptabilidad de Métrica V3 a cualquier tipo de proyecto, sea cual sea su magnitud, no es obligatorio realizar todos los pasos que se describen, sino sólo los aplicables al proyecto concreto. Hay algunas etapas que serán descartadas debido a que hacen referencia a proyectos de dimensiones más elevadas.

La metodología Métrica V3 se divide en tres procesos fundamentales y sus subprocesos:

- Planificación de Sistemas de Información (PSI)
- Desarrollo de Sistemas de Información
  - Estudio de Viabilidad del Sistema (EVS)
  - Análisis del Sistema de Información (ASI)
  - Diseño del Sistema de Información (DSI)
  - Construcción del Sistema de Información (CSI)
  - Implantación y Adaptación del Sistema (IAS)
- Mantenimiento de Sistemas de información (MSI)

4.3. ¿Por qué una mezcla entre programación estructurada y orientación a objetos?

El desarrollo de un sistema consiste en una serie de iteraciones del tipo división-unión; hay que descomponer (dividir) para comprender y hay que componer (unir) para construir.

El proceso de descomposición se lleva cabo tradicionalmente mediante un criterio funcional. Se identifican las funciones del sistema, luego se descompone en subfunciones, y así sucesivamente hasta obtener elementos simples. La arquitectura de un programa así realizado es un reflejo del sistema. Este método aporta resultados satisfactorios cuando las funciones están bien definidas y son estables en el tiempo. Sin embargo, al existir un acoplamiento grande entre arquitectura y funciones, las evoluciones importantes pueden suponer modificaciones estructurales importantes.

La descomposición orientada a objetos se basa en los objetos que integran la estructura y el comportamiento. Las funciones se representan mediante colaboraciones entre los objetos que componen el sistema. El acoplamiento se hace dinámico, y las evoluciones funcionales no cuestionan la estructura estática del programa.

El empleo de las metodologías orientadas a objetos se justifica con las siguientes razones;

- La orientación a objetos se acerca más a la forma de pensar de las personas, haciéndolo más comprensible y fácil de aplicar.
- Facilita la abstracción, centrándonos en cada momento únicamente en los elementos que nos interesen descartando los demás.
- Aumenta la consistencia interna al tratar los atributos y las operaciones como un todo.
- Permite expresar características comunes sin repetir la información.
- Facilita la reutilización de diseños y código.
- Facilita la revisión y modificación de sistemas desarrollados.
- Origina sistemas más estables y robustos.

Sin embargo, no todos son ventajas. Cada vez que se crea un objeto este permanece en memoria, consumiendo espacio, hasta su destrucción. Por tanto, si necesitamos ejecutar un método de un objeto, éste debe estar creado y ocupando espacio en memoria. Cuando el número de objetos es pequeño no pasa nada.

En el caso de este proyecto por la parte que a base de datos respecta, no sería lógico tener un objeto por cada tupla a insertar en la base de datos. Menos lógico aun sería tener un objeto por cada fichero que leemos. Tampoco sería razonable tener un objeto por cada tupla e ir destruyéndolo conforme fuera insertado.

¿Cuál es la solución entonces? Emplear funciones que no pertenecen a ningún objeto, es decir, programación estructurada. Las funciones sólo ocupan en memoria lo que corresponde a su porción de código y a las variables locales que utiliza, lo cual es un ahorro considerable.

Otra posibilidad sería utilizar bases de datos orientadas a objetos. Sin embargo aun están poco desarrolladas y no hay mucha documentación de ellas. Por ello será la decisión expuesta en el párrafo anterior la que tomemos.

Capítulo 5

Especificación de los requisitos del sistema

5.1. Requisitos de interfaces externas

- El programa debe ser capaz de introducir y extraer datos de la base de datos con la que esté trabajando, así como realizar consultas en ella.
- Será portable entre los sistemas operativos Windows y Linux.
- Utilizará la aplicación Xilinx ISE Webpack a partir de llamadas al sistema, mediante las órdenes que dicho software proporciona para trabajar desde línea de comandos.
- La interfaz de usuario deberá mostrar como pantalla principal tres tablas:

  1. Una tabla con información extraída de los reports que se generan mediante ISE Webpack así como el precio de las FPGAs empleadas en cada implementación. Figura 5.1
  2. Una tabla con información de las implementaciones que no se han podido realizar junto con el nombre de los ficheros que no han sido encontrados.
  3. Una tabla en la que se indica las implementaciones que no se han podido realizar al no disponer la FPGA de suficientes recursos lógicos para un determinado circuito.

  Cada tabla deberá ir acompañada del número de registros que contiene y se debe indicar el directorio en el que se encuentran los reports recién analizados. Así mismo, la interfaz debe permitir visualizar los circuitos implementados (figura 5.2), filtrar los resultados (figura 5.3) que se visualizan en las tablas anteriormente y, mediante un menú, analizar nuevos reports e implementar nuevos circuitos. También existirá un diálogo al inicio para solicitar los datos del usuario (figura 5.4) y diversos diálogos de error.
- El formato de los ficheros que se analizarán se encuentra en la sección 8.8
- El usuario debe ser capaz de introducir, mediante la interfaz gráfica, datos como los circuitos y las FPGAs con los que van a realizarse las fases ya mencionadas a lo largo de este documento.

Figura 5.1: Pantalla principal

Figura 5.2: Ventana auxiliar con circuitos

Figura 5.3: Diálogo de filtros

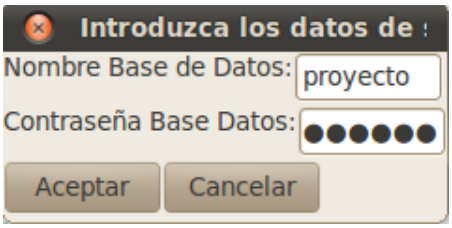

Figura 5.4: Diálogo introducción datos

- La interfaz también deberá mostrar el directorio donde se encontraban los últimos informes que se agregaron a la base de datos.

5.2. Requisitos funcionales

- El sistema deberá ser capaz de ordenar al programa Xilinx ISE Webpack la ejecución de las cinco herramientas implicadas en el proceso de desarrollo y evaluación del dispositivo FPGA de Xilinx.

- El sistema deberá ser capaz de extraer información de cada uno de los informes generados por cada una de esas herramientas y almacenarla en una base de datos.

- El sistema deberá ser capaz de mostrar la información leída en los informes anteriores mendiante la interfaz gráfica.

- El sistema debe ser capaz de ordenar las tablas mostradas según el campo que se le indique.

- El sistema debe ser capaz de mostrar los circuitos que tiene constancia de haber probado y analizado.

- El sistema debe ser capaz de desactivar los filtros que se les hayan aplicado a las diferentes tablas.

5.3. Requisitos de rendimiento

- El sistema debe mejorar el tiempo que tardaba la aplicación previa en leer los informes generados por las cinco herramientas de *Xilinx ISE Webpack*. Recuérdese la fórmula expuesta anteriormente $N = 5 * Tecnologias * Circuitos$, siendo N el número de ficheros a tratar.
- El sistema debe mejorar el tiempo de la aplicación previa en abrir cada informe, localizar los datos, realizar los cálculos o conversiones oportunas y almacenar la información en la base de datos.

5.4. Restricciones de diseño

Se debe dar una mayor prioridad a la optimización temporal frente a la espacial, ya que uno de los principales objetivos de la aplicación es ser más eficiente que su predecesora.

5.5. Atributos del sistema software

El sistema deberá cumplir los siguientes atributos:

- Código mantenible y ampliable.
- Facilidad para la actualización de inclusión de nuevas FPGAs.
- Portabilidad entre sistemas Windows y GNU/Linux.
- Robusto durante la ejecución.
- Seguro en cuanto al manejo de datos del usuario.

5.6. Otros requisitos

El código debe ser sencillo, legible y poco pesado ya que va a ser utilizado y estudiado por personas con más conocimientos en la electrónica que en la informática.

Capítulo 6

Análisis del sistema

En este apartado se procederá al análisis del sistema en notación UML. Se expondrán los modelos de casos de uso, de datos y de comportamiento.

6.1. Modelo de casos de uso

6.1.1. Descripción de los casos de uso

A continuación se pasan a describir los distintos casos de uso del sistema con sus escenarios principales y alternativos según la notación UML.

Caso de uso: Validar usuario

- **Caso de uso**: Validar usuario.
- **Descripción**: Solicita al usuario sus datos para acceder a la base de datos o crearla en su defecto.
- **Actores**: Usuario.
- **Precondiciones**: Ninguna.
- **Postcondiciones**: El sistema se conecta la base de datos,
- **Escenario principal**:
    1. El usuario solicita validarse en el sistema.
    2. El sistema muestra una pantalla donde pueda introducir los datos.

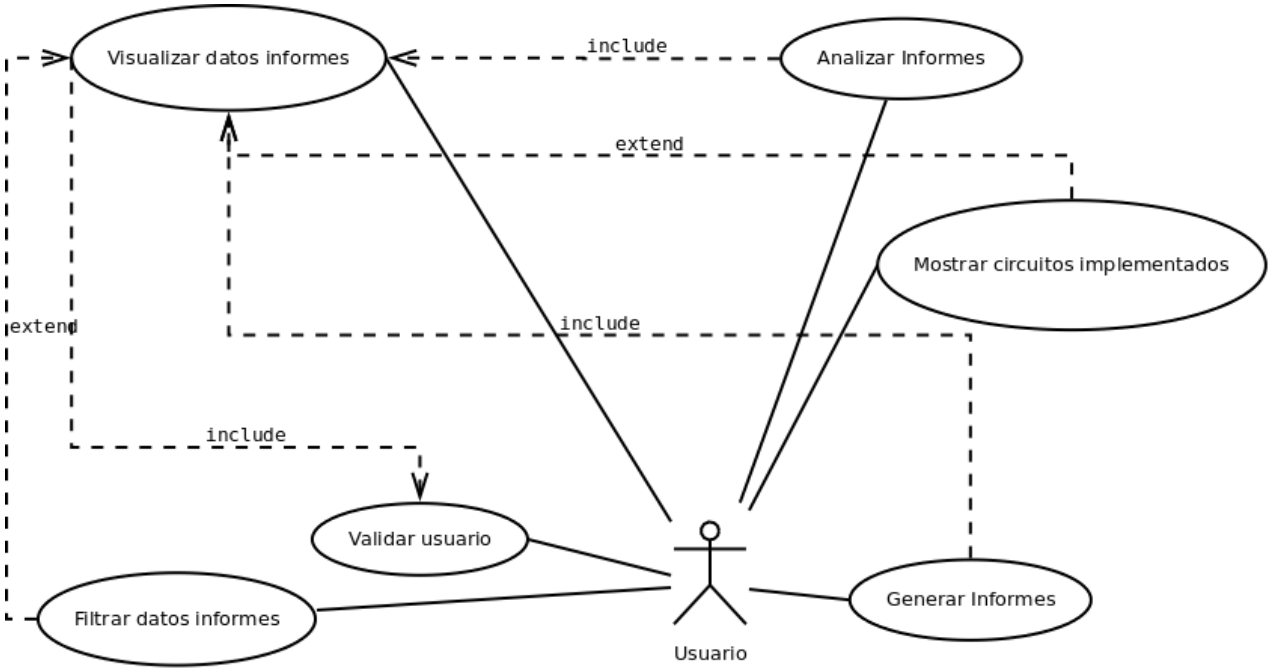

Figura 6.1: Diagrama de casos de uso

3. El usuario introduce su contraseña del sistema de base de datos.
4. El sistema comprueba que es correcta.
5. El usuario introduce el nombre de la base de datos que quiere utilizar.
6. El sistema comprueba que es correcto y se conecta a la base de datos.

- **Extensiones**:

*a El usuario decide no validarse.

1. El sistema se cierra.

4.a La contraseña no es correcta.

1. El sistema solicita de nuevo la contraseña.

6.a La base de datos indicada no existe

1. El sistema crea una base de datos con ese nombre.

6.1.2. Caso de uso: Visualizar datos informes

- **Caso de uso**: Visualizar datos informes.
- **Identificación de los escenarios**:
  - **Escenario principal**: Se visualizan los datos extraídos de los informes correctamente.

- **Escenario alternativo 1**: El usuario decide, además, filtrar los datos visualizados.
- **Escenario alternativo 2**: El usuario decide, además, ver los circuitos implementados.
- **Escenario excepción 1**: El usuario decide no visualizar los datos.

- **Descripción**: Muestra por pantalla los datos de los informes analizados que se encuentran en la base de datos.
- **Puntos de extensión**: Filtrar en 3a y Ver circuitos en 3b.
- **Actores**: Usuario.
- **Precondiciones**: Include Validar Usuario
- **Postcondiciones**: El sistema muestra los datos requeridos por pantalla
- **Escenario principal**:

  1. El usuario quiere visualizar los datos extraidos de los informes.
  2. El sistema obtiene dichos datos de la base de datos indicada y los muestra.
  3. El usuario sale del sistema.

- **Extensiones**:

  *a El usuario decide que no desea visualizar ningún dato.

  1. El sistema se cierra.

  3.a El usuario solicita filtar los resultados visualizados.

  3.b El usuario solicita visualizar los circuitos que han sido implementados.

6.1.3. Caso de uso: Filtrar datos informe

- **Caso de uso**: Filtrar datos informe.
- **Identificación de los escenarios**:

  - **Escenario principal**: Se visualizan los datos filtrados correspondientes a la tabla informes según las condiciones impuestas por el usuario.
  - **Escenario alternativo 1**: Se visualizan los datos correspondientes a la tabla fallidas según las condiciones impuestas por el usuario.
  - **Escenario alternativo 2**: Se visualizan los datos filtrados correspondientes a la tabla sobredimensionadas según las condiciones impuestas por el usuario.
  - **Escenario excepción 1**: El usuario decide no filtrar los datos.
  - **Escenario error 1**: Alguna de las condiciones introducidas no es correcta.

- **Descripción**: Muestra por pantalla los datos de los informes analizados que cumplan una serie de condiciones impuestas por el usuario.
- **Puntos de extensión**: **Filtrar** en Visualizar Datos Informes.
- **Actores**: Usuario.
- **Precondiciones**: El usuario debe estar validado en el sistema.
- **Postcondiciones**: El sistema muestra los datos requeridos por pantalla
- **Escenario principal**:

  1. El usuario quiere aplicar una serie de condiciones a los datos extraidos de los informes que se muestran en la tabla de reports.
  2. El usuario introduce las condiciones.
  3. El sistema comprueba que son correctas.
  4. El sistema muestra por pantalla aquellos datos que cumplen esas condiciones.

- **Extensiones**:

  *a El usuario decide que no desea aplicar ningún filtro.

  1. El sistema muestra los datos tal cual se extrayeron, sin aplicar ningún filtro.

  1.a El usuario quiere aplicar una serie de condiciones a los datos extraidos de los informes que se muestran en la tabla de fallidas.

  1.b El usuario quiere aplicar una serie de condiciones a los datos extraidos de los informes que se muestran en la tabla de sobredimensionadas.

  3.a Las condiciones a aplicar no son válidas.

  1. El sistema muestra un mensaje indicando que se ha producido un error.
  2. El sistema muestra los datos tal cual se extrayeron, sin aplicar ningún filtro.

6.1.4. Caso de uso: Visualizar circuitos implementados

- **Caso de uso**: Visualizar circuitos implementados.
- **Identificación de los escenarios**:

  - **Escenario principal**: Se visualizan los circuitos implementados correctamente.
  - **Escenario excepción 1**: El usuario decide no visualizar dichos circuitos.

- **Descripción**: Muestra por pantalla los circuitos de los que existe al menos una implementación.
- **Puntos de extensión**: **Ver circuitos** en Visualizar Datos Informes.

- **Actores**: Usuario.
- **Precondiciones**: El usuario debe estar validado en el sistema.
- **Postcondiciones**: El sistema muestra los datos requeridos por pantalla
- **Escenario principal**:
    1. El usuario solicita visualizar los circuitos de los que existe, al menos, una implementación.
    2. El sistema muestra por pantalla dichos circuitos.
- **Extensiones**:

    2.a El usuario decide que no desea visualizar los circuitos.
        1. El sistema no modifica la pantalla o la deja en el estado anterior a esta petición.

6.1.5. Caso de uso: Generar informes

- **Caso de uso**: Generar informes.
- **Identificación de los escenarios**:
    - **Escenario principal**: Se generan los informes del directorio indicado en el directorio correctamente.
    - **Escenario error 1**: El directorio indicado no contiene circuitos.
    - **Escenario excepción 1**: El usuario decide no analizar nuevos informes.
- **Descripción**: Se generan reports de las implementaciones indicadas por el usuario.
- **Actores**: Usuario.
- **Precondiciones**: Include Visualizar Datos Informes.
- **Postcondiciones**: El sistema genera los reports requeridos por el usuario.
- **Escenario principal**:
    1. El usuario quiere generar informes de implementaciones circuito-FPGA.
    2. El usuario introduce el directorio donde se encuentran los circuitos.
    3. El usuario introduce el directorio donde quiere que se almacenen los reports.
    4. El sistema muestra por pantalla una lista de FPGAs.
    5. El usuario escoge de ellas un conjunto para las implementaciones.
    6. El sistema genera los informes requeridos en directorio de destino indicado.
- **Extensiones**:

*a El usuario decide que no desea generar nuevos informes.

1. El sistema no modifica la pantalla o la deja en el estado anterior a esta petición.

6.a La jerarquía de carpetas donde se encuentran los circuitos no es correcta.

1. El sistema no generará los informes cuya jerarquía de carpetas no sea correcta.

6.1.6. Caso de uso: Analizar informes

- **Caso de uso**: Analizar informes.
- **Identificación de los escenarios**:
  - **Escenario principal**: Se analizan los informes del directorio indicado correctamente.
  - **Escenario alternativo 1**: Existe alguna implementación que ya está en la base de datos y el usuario decide no analizar los informes correspondientes.
  - **Escenario alternativo 2**: Existe alguna implementación que ya está en la base de datos y el usuario decide analizar los informes correspondientes y almacenarlos como una nueva versión.
  - **Escenario alternativo 3**: Existe alguna implementación que ya está en la base de datos y el usuario decide analizar los informes correspondientes y sobreescribir todos los coincidentes.
  - **Escenario excepción 1**: El usuario decide no analizar nuevos informes.
- **Descripción**: Se extraen datos de los informes generados y se almacenan en una base de datos.
- **Actores**: Usuario.
- **Precondiciones**: Include Visualizar Datos Informes.
- **Postcondiciones**: El sistema extrae los datos de los informes indicados y los almacena en la base de datos.
- **Escenario principal**:
  1. El usuario quiere analizar informes de implementaciones circuito-FPGA.
  2. El usuario introduce el directorio donde se encuentran los informes.
  3. El sistema extrae y almacena los datos en la base de datos.
  4. El sistema muestra los datos extraídos.
- **Extensiones**:

*a El usuario decide que no desea analizar nuevos informes.

1. El sistema no modifica la pantalla o la deja en el estado anterior a esta petición.

3.a Ya existe una implementación entre un determinado circuito y una determinada FPGA.

1. El sistema muestra un mensaje indicando la situación
2. El usuario responde al mensaje.

3.a.2.a El usuario decide descartar el análisis de dicho informe.

1 El sistema no analiza dicho informe.

3.a.2.b El usuario decide analizar el informe y almacenarlo como una nueva versión.

1 El sistema analiza dicho informe y lo almacena en la base de datos.

3.a.2.c El usuario decide analizar dicho informe y sobreescribir los anteriores coincidentes.

1 El sistema analiza dicho informe, lo almacena en la base de datos y elimina los que coincidieran en la base de datos.

3.b El directorio indicado no contiene informes o contiene algo más que informes.

1. El sistema sólo analizará aquellos informes que encuentre.

6.2. Modelo conceptual de datos

Este sistema combina la programación estructurada con la orientada a objetos por lo que no va a ser posible reflejar todo el sistema con este modelo. Sin embargo sí que se puede reflejar una parte y esto es lo que se va a ver a continuación.

6.2.1. Nodos independientes

La ejecución y el reparto de las tareas puede ser enfocado desde varios puntos de vista. En este trabajo de ha enfocado desde dos distintos. En uno de ellos cada nodo será independiente del resto e irá ejecutando los trabajos que aún no se hayan realizado. Nadie controla ni qué ni cuando ejecuta una tarea un determinado nodo.

En el análisis de este sistema se han detectado las siguientes clases:

- **Base de datos**: Será la encargada de insertar y extraer datos de la base de datos, actuando como capa de datos según el modelo de tres capas. Abstrae las peculiaridades del sistema de base de datos que se utilice del resto de clases.
- **Fase**: Representa a alguna tarea que hay que realizar para completar un Trabajo.
- **Report**: Encapsula y obtiene el conjunto de datos de los informes que se han generado.

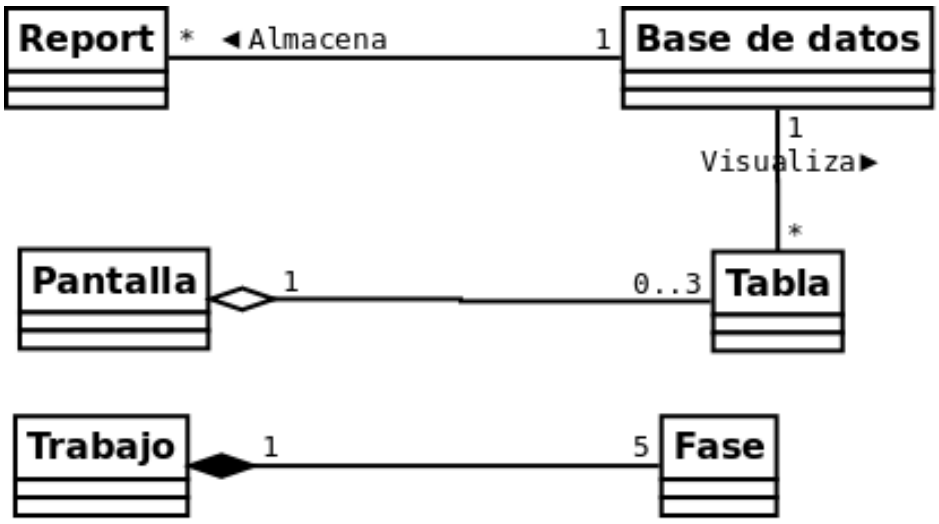

Figura 6.2: Diagrama de clases

- **Trabajo**: Representa la implementación de un circuito en una FPGA determinada. Está compuesto de Fases.

6.2.2. Cliente-Servidor

El otro punto de vista sigue un esquema cliente-servidor, donde un nodo se encargará de proporcionar los trabajos a los nodos según estos los vayan solicitando.

Esta última estrategia implicaría una modificación del modelo de datos, añadiendo las clases:

1. Trabajos: Contiene un conjunto de Trabajo y posee un método ObtenerTrabajo que devolverá un trabajo a ejecutar y lo eliminará del conjunto.

2. Cliente: Se encargará de solicitar objetos del tipo Trabajo a un objeto Servidor y ejecutarlos.

3. Servidor: Proporcionará objetos Trabajo al Cliente mediante un objeto de la clase Trabajos.

De esta manera, el diagrama de datos quedaría como en la figura 6.3

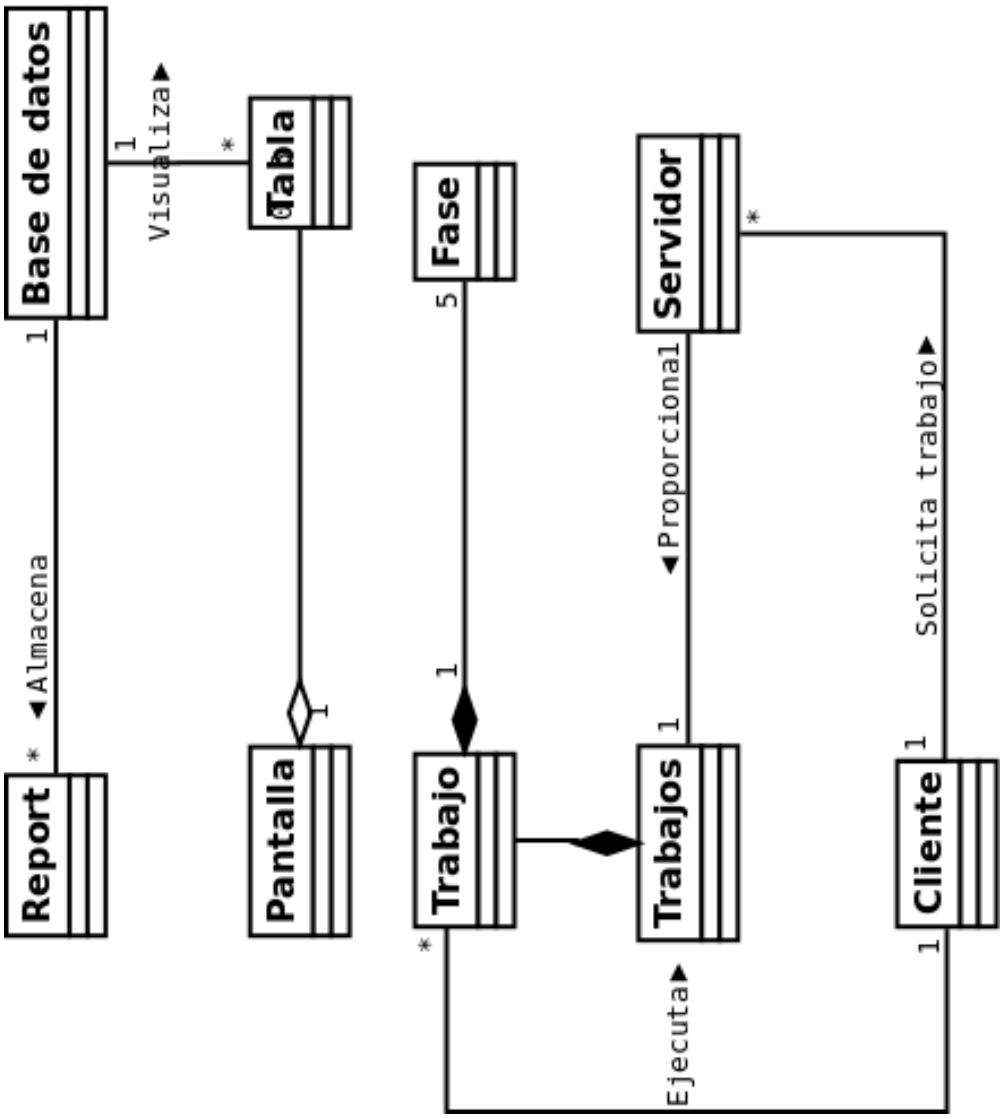

Figura 6.3: Modelo de datos cliente-servidor

6.3. Modelo de comportamiento del sistema

6.3.1. Modelo de comportamiento de Validar Usuario

Contrato de las operaciones

- **Operación: Mostrar_pantalla()**
  - **Responsabilidades:** Muestra una pantalla donde el usuario pueda introducir sus datos.
  - **Referencias cruzadas:** Caso de uso Validar Usuario
  - **Precondiciones:** No tiene.

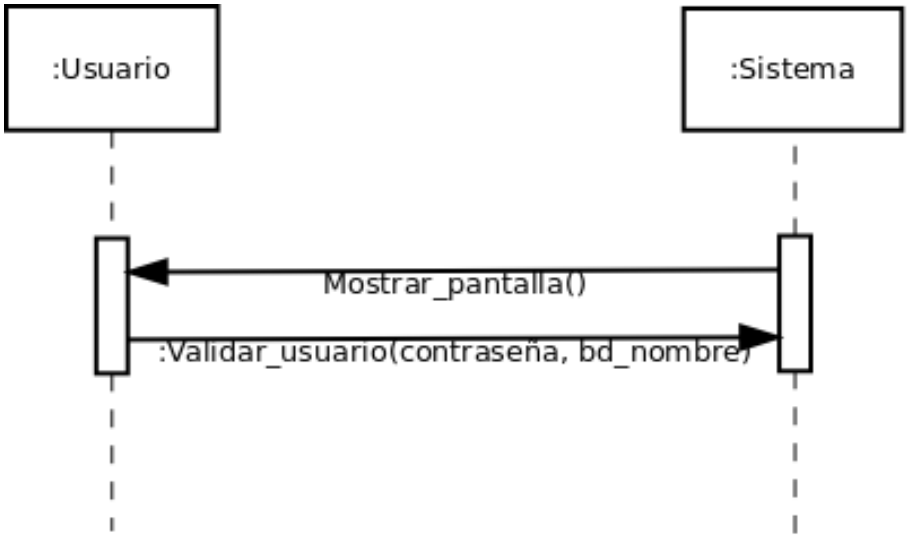

Figura 6.4: Diagrama de secuencia de Validar Usuario

- **Postcondiciones:** El sistema muestra una pantalla donde el usuario debe introducir sus datos.

- **Operación: Validar_usuario(contraseña,bd_nombre)**
  - **Responsabilidades:** Conectar con el sistema de gestión de base de datos con la contraseña dada y a la base de datos indicada.
  - **Referencias cruzadas:** Caso de uso Validar Usuario.
  - **Precondiciones:** No tiene.
  - **Postcondiciones:**
    - Se crea una nueva instancia BD de Base de Datos.
    - Se asigna contraseña a BD.contraseña.
    - Se asigna bd_nombre a BD.bd_nombre.

6.3.2. Modelo de comportamiento de Visualizar Datos Informes

Contrato de las operaciones

- **Operación: Ver_Datos_informes()**
  - **Responsabilidades:** Solicitar la visualización por pantalla los datos que han sido extraídos de los informes.
  - **Referencias cruzadas:** Caso de uso Visualizar Datos Informes
  - **Precondiciones:** Existe un objeto db de la clase Base de Datos.
  - **Postcondiciones:**

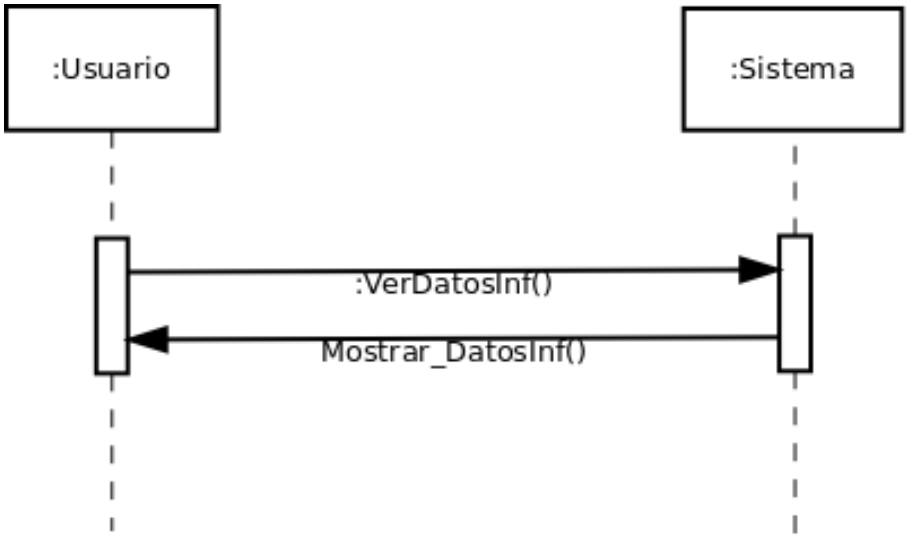

Figura 6.5: Diagrama de secuencia de Visualizar Datos Informes

- Se crean tres objetos, t1, t2, t3, de la clase Tabla.
- Se crea un objeto p de la clase Pantalla.
- Se asocia t1 con db y los atributos de la entidad reports más precio y familia de la entidad tecnología.
- Se asocia t2 con db y la entidad fallidas.
- Se asocia t3 con db y la entidad sobredimensionadas.
- Se asocian t1, t2 y t3 con p.

- **Operación: Mostrar_DatosInf()**
  - **Responsabilidades:** Muestra por pantalla los datos que han sido extraídos de los informes.
  - **Referencias cruzadas:** Caso de uso Visualizar Datos Informes
  - **Precondiciones:** Existen tres objetos de la clase Tabla en curso y un objeto p de Pantalla.
  - **Postcondiciones:**
    - El objeto p muestra las tablas con las que está relacionado.

6.3.3. Modelo de comportamiento de Mostrar circuitos implementados

Contrato de las operaciones

- **Operación: VerCircuitos()**
  - **Responsabilidades:** Solicitar la visualización por pantalla los circuitos que se han implementado en alguna FPGA.

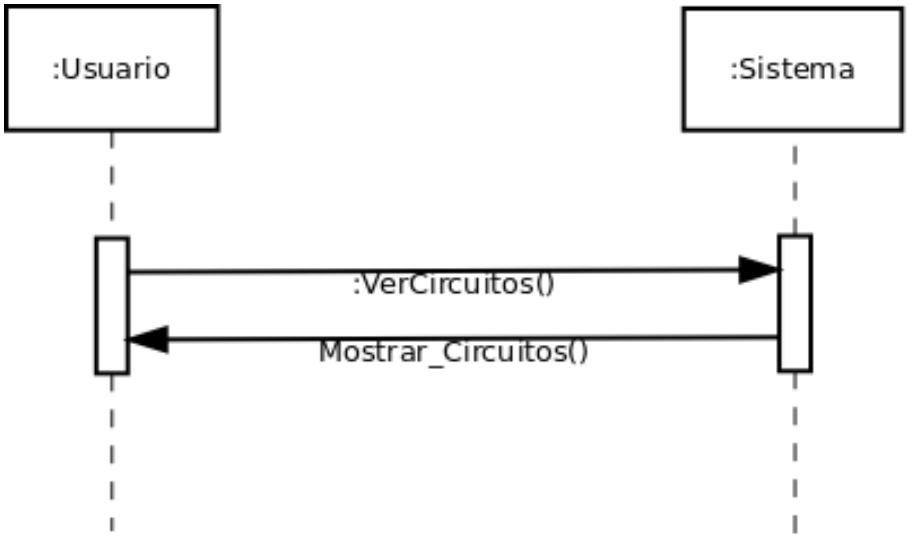

Figura 6.6: Diagrama de secuencia de Mostrar circuitos implementados

- **Referencias cruzadas:** Caso de uso Mostrar circuitos implementados.
- **Precondiciones:** Existe un objeto BD de la clase Base de Datos en curso.
- **Postcondiciones:**
  - Se obtienen los datos solicitados a partir de métodos del objeto BD.
  - Se crea una Pantalla p.
  - Se crea una Tabla t.
  - t obtiene los datos de circuitos del objeto BD.
  - Se relacionan p y t.

- **Operación: Mostrar_Circuitos()**
  - **Responsabilidades:** Muestra por pantalla los datos que han sido extraídos de la base de datos.
  - **Referencias cruzadas:** Caso de uso Mostrar circuitos implementados.
  - **Precondiciones:** Existe un objeto p de la clase Pantalla relacionado con una y sólo una Tabla.
  - **Postcondiciones:**
    - Se muestra la Pantalla p.

6.3.4. Modelo de comportamieno de Filtrar Datos Informes

Contrato de las operaciones

- **Operación: IntroducirCondicion(tabla,condicion)**

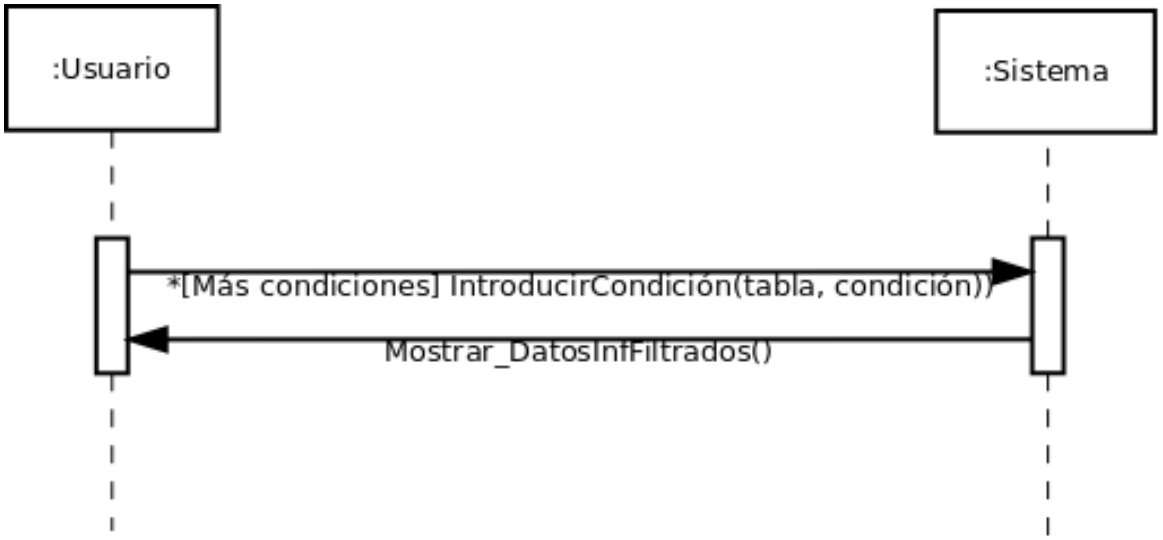

Figura 6.7: Diagrama de secuencia de Filtrar Datos Informes

- **Responsabilidades:** Añadir condiciones a los datos que se muestran en una determinada tabla.
- **Referencias cruzadas:** Caso de uso Filtrar Datos Informes.
- **Precondiciones:** Existe un objeto BD de la clase Base de Datos en curso y un objeto t de la clase Tabla.
- **Postcondiciones:**
  - Se añade la condición *condicion* a la Tabla *tabla*

- **Operación: MostrarDatosFiltrados()**
  - **Responsabilidades:** Mostrar por pantalla los datos que cumplen las condiciones impuestas por el usuario.
  - **Referencias cruzadas:** Caso de uso Filtrar Datos Informes.
  - **Precondiciones:** Existe un objeto p de la clase Pantalla y un objeto t de la clase Tabla.
  - **Postcondiciones:**
    - Se actualiza la relación entre p y t.

6.4. Modelo de comportamiento de Generar Informes

Contrato de las operaciones

- **Operación: IntroducirDirFuenteCir(directorio)**

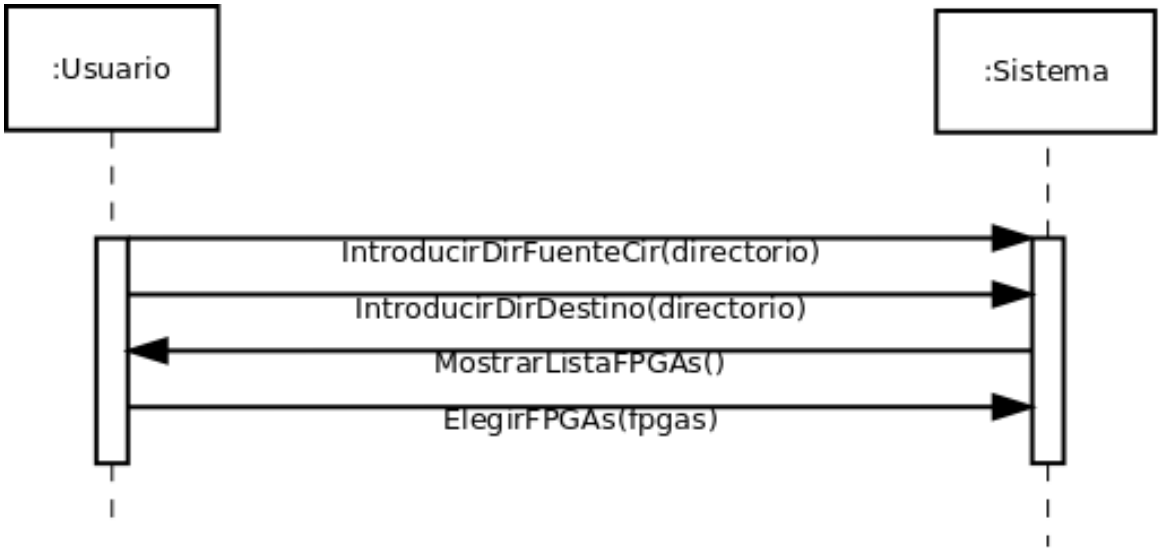

Figura 6.8: Diagrama de secuencia de Generar Informes

- **Responsabilidades:** Introducir el directorio donde se encuentran los circuitos a analizar.
- **Referencias cruzadas:** Caso de uso Generar Informes.
- **Precondiciones:**
- **Postcondiciones:**
  - Se guarda el directorio introducido.

- **Operación: IntroducirDirDestino(directorio)**
  - **Responsabilidades:** Introducir el directorio donde se guardarán los ficheros generados.
  - **Referencias cruzadas:** Caso de uso Generar Informes.
  - **Precondiciones:** Ya se ha guardado el directorio fuente.
  - **Postcondiciones:**
    - Se guarda el directorio introducido.
- **Operación: MostrarListaFPGAs()**
  - **Responsabilidades:** Mostrar al usuario una lista de FPGAS disponibles para utilizar
  - **Referencias cruzadas:** Caso de uso Generar Informes.
  - **Precondiciones:** Existe un objeto BD de la clase Base de datos en curso.
  - **Postcondiciones:**
    - Se muestran las FPGAs disponibles obtenidas de la base de datos gracias al objeto BD.
- **Operación: ElegirFPGAs(fpgas)**

- **Responsabilidades:** Introducir una lista de FPGAS que se van a utilizar.
- **Referencias cruzadas:** Caso de uso Generar Informes.
- **Precondiciones:** Se han guardado los directorios fuente y destino y se ha mostrado la lista de FPGAs disponibles.
- **Postcondiciones:**
  - Se guardan las FPGAs seleccionadas.
  - Se crea un objeto t de la clase Trabajo por cada circuito encontrado en el directorio fuente y cada FPGA seleccionada.
  - Se generan los distintos objetos de la clase Fases de cada Trabajo t.

Capítulo 7

Diseño del sistema

En el diseño de *Thor* se seguirá la arquitectura de tres capas de la ingeniería del software, teniendo:

- Una capa de datos que hará transparente al resto del entorno cómo están almacenados los datos.
- Una capa de dominio que implementará todas las funcionalidades del sistema.
- Una capa de presentación que se encargará de controlar las entradas y salidas del usuario y la interfaz gráfica.

7.1. Comportamiento

A continuación se exponen los diagramas de secuencia obtenidos en la fase de diseño del sistema junto con las descripciones de las operaciones:

7.1.1. Clase ValidarUsuario_Control

- Operación que se ejecuta al hacer new.ValidarUsuario_Control. Invoca a la operación Mostrar_pValidar de la clase Pantalla_ValidarUsuario.

7.1.2. Clase Pantalla_ValidarUsuario

- **Operación Mostrar_pValidar**. Muestra la pantalla correspondiente para que el usuario pueda validarse.

7.1.3. Clase BD (Base de datos)

- **Operación Create(contraseña, bd_nombre)**. Crea una conexión con el sistema de gestión de base de datos pertinente. La conexión se realizará con la base de datos bd_nombre, el usuario que esté utilizando el programa y la contraseña "contraseña".

- **Operación ObtenerDatos(table, atributos,condiciones)**. Solicita los datos contenidos en la tabla table al sistema de gestión de base de datos que cumplan las condiciones dadas. Si no se especifican atributos se proporcionan todos los pertenecientes a la tabla table.

-

7.1.4. Clase Tabla

- **Operación Create(table, atributos)**. Invoca a la operación ObtenerDatos(table,atributos) del objeto de BD en curso.

- **Operación AñadirTabla(table,atributos)**. Invoca a la operación ObtenerDatos(table,atributos) del objeto de BD en curso y se queda con el producto natural de ambas tablas añadiendo a sus condiciones aquellas que igualen los atributos comunes entre las tablas.

- **Operación AñadirCondiciones(condiciones)**. Añade a sus condiciones las recién insertadas.

7.1.5. Clase Pantalla_Ver_Informes

- Operación que se ejecuta al hacer new.Pantalla_Ver_Informes. Invoca a la operación Create(table,atributos) de la clase Tabla en tres ocasiones y a las operaciones AñadirTabla(table,atributos) y AñadirCondiciones(condiciones) de la misma clase.

- **Operación AñadirCondiciones(tabla,condiciones)**. Invoca a la operación AñadirCondiciones(condiciones) el objeto de la clase Tabla tabla.

- **Operación SolicitarFPGAS()**. Solicita al usuario el conjunto de FPGAs con el que desea trabajar y lo devuelve. Esta operación solo es necesaria si se sigue la estrategia cliente-servidor.

7.1.6. Clase MostrarCircuitos_Control

- **Operación VerCirIm()**. Invoca a Create() de la clase Pantalla_VerCircuitos.

7.1.7. Clase Pantalla_Ver_Circuitos

- Operación que se ejecuta al hacer new.Pantalla_Ver_Informes. Invoca a la operación Create(table,atributos) de la clase Tabla siendo table=circuitos y atributos="".
- **Operación RecargaPantalla()**. Refresca la pantalla haciendo que aparezcan los datos de las tablas actualizados.

7.1.8. Clase FiltrarDatosInformes_Control

- **Operación IntroducirCondiciones(tabla,condiciones)**. P es la Pantalla_Ver_Circuitos en curso. Invoca a la operación AñadirCondiciones(tabla,condiciones) del objeto P, siendo tabla una referencia al objeto tabla al que se le quieren aplicar las condiciones dadas.

7.1.9. Clase GenerarInformes_Control

- **Operación IntroducirDirFuente(origen)**. Guarda en una variable propia el directorio origen. Invoca a la operación ObtenerCircuitos(origen), guardando en una variable local el resultado de esta llamada.
- **Operación ObtenerCircuitos(directorio)**. Obtiene una lista de circuitos a partir de los subdirectorios existentes en el directorio dado.
- **Operación IntroducirDirDestino(destino)**. Guarda en una variable propia el directorio destino.
- **Operación IntroducirFPGAs(fpgas)**. Guarda en una variable propia el conjunto de FPGAs fpgas que se va a utilizar. Invoca a la operación Create(circuito,fpga) de la clase Trabajo por cada par circuito-fpga existente. También invoca a la operación Ejecutar() de los objetos Trabajo creados.
- **Operación GenerarDirectorios(circuito,fpga)**. Crea la jerarquía de carpetas para el circuito y fpga dados en el directorio fuente, copiando los archivos necesarios desde origen.
- **Operación IntroducirModo(modo)**. Si el modo no es cliente invocará a la operación SolicitarFPGAs() de la clase Pantalla_Ver_Informes y llamará a la operación Create(FPGAS, origen, destino) de la clase Servidor. Si el modo no fuera servidor crearía un objeto de la clase Cliente invocando al método Create() de dicha clase. Esta clase sólo existirá si se está trabajando con la estrategia cliente-servidor.

7.1.10. Clase Trabajo

- **Operación Create(circuito,fpga)**. Invoca a la operación Generar_Directorios(circuito,fpga) y cinco veces a la operación Create(orden) de las fases, construyendo así la síntesis, la

traducción, el mapeo, el enrutamiento y emplazamiento y el análisis de tiempo.

- **Operación Ejecutar**. Invoca a la operación Ejecutar() de cada una de las fases por las que está compuesto.

7.1.11. Clase Fase

- **Operación Create(orden)**. Ejecuta la asignación orden_ =orden.
- **Operación Ejecutar**. Realiza una llamada a la aplicación *ISE Webpack* mediante la orden dada.

7.1.12. Clase AnalizarInformes_Control

- **AnalizarInformes(directorio)**. Invoca a la operación ObtenerCircuitos(directorio) y una vez por circuito a la operación ObtenerFPGAS(directoriocircuito). Posteriormente invoca una vez por cada pareja circuito-fpga a la operación Create(circuito,fpga) de la clase Report. Por último llama una única vez a la método RecargaPantalla() del objeto Pantalla_Ver_Informes en curso.
- **Operación ObtenerCircuitos(directorio)**. Obtiene una lista de circuitos a partir de los subdirectorios existentes en el directorio dado.
- **Operación ObtenerFPGAS(directorio)**. Obtiene una lista de FPGAs a partir de los subdirectorios existentes en el directorio dado.

7.1.13. Clase Report

- **Create(circuito,fpga)**. Invoca a la operación Create(contraseña, bd_nombre) de la clase BD siendo contraseña y bd_nombre aquellos que se introdujeron en Validar_Usuario. A continuación invoca a las operaciones RegistrarCircuito(circuito) del objeto de la clase BD, AnalizarPAR() de él mismo, RegistrarTecnologia(fpga) del objeto de la clase BD, AnalizarSRP(), AnalizarMAP(), AnalizarTWR() de él mismo y RegistrarImplementacion() del objeto de la clase BD. A continuación según se haya completado la implementación correctamente o no se llamará a RegistrarFallida() o RegistrarExitosa() del objeto de la clase BD y si fue exitosa y no sobredimensionada se invocará a RegistrarReport() del mismo objeto.

7.1.14. Clase Trabajos

- **Create(FPGAs, origen, destino)**. Invoca a la operación ObtenerCircuitos(origen), guardando en una variable local el resultado de esta llamada. También invoca a GenerarDirecto-

rios(circuito,fpga), que es una operación de esta clase.

- **Operación ObtenerCircuitos(directorio)**. Obtiene una lista de circuitos a partir de los subdirectorios existentes en el directorio dado.
- **Operación GenerarDirectorios(circuito,fpga)**. Crea la jerarquía de carpetas para el circuito y fpga dados en el directorio fuente, copiando los archivos necesarios desde origen.
- **Operación TrabajoPop()**. Extrae un trabajo de la lista que posee, lo devuelve y lo elimina de la lista.

La clase Trabajos solo está presente en la estrategia cliente-servidor. Si se siguiera esta estrategia las operaciones ObtenerCircuitos y GenerarDirectorios pasarían a ser propias a esta clase en lugar de a GenerarInformes_Control.

7.1.15. Clase Servidor

- **Create(FPGAs, origen, destino)**. Crea un objeto de tipo Trabajos invocando a la operación Create(FPGAs, origen, destino) de dicha clase.
- **ObtenerTrabajo()**. Invoca a la operación TrabajoPop del objeto Trabajos que tiene asociado y lo devuelve.

7.1.16. Clase Cliente

- **Create()**. Invoca a la operación ObtenerTrabajo() del objeto Servidor asociado y ejecuta el objeto Trabajo que dicha operación devuelve mediante el método Ejecutar() de tal objeto. Estas dos operaciones se ejecutan mientras ObtenerTrabajo() devuelva un Trabajo a realizar.

7.2. Distribución de las clases en capas

Siguiendo la arquitectura de tres capas (presentación, dominio y datos) ésta sería la distribución de las clases anteriores en ellas:

Clase	Presentación	Dominio	Datos
AnalizarInformes_Control		X	
BD			X
Fase		X	
FiltrarDatos_Control		X	
GenerarInformes_Control		X	
MostrarCircuitos_Control		X	
Pantalla_ValidarUsuario	X		
Pantalla_Ver_Circuitos	X		
Pantalla_Ver_Informes	X		
Report		X	
Trabajo		X	
ValidarUsuario_Control		X	

Tabla 7.1: Distribución de las clases en capas

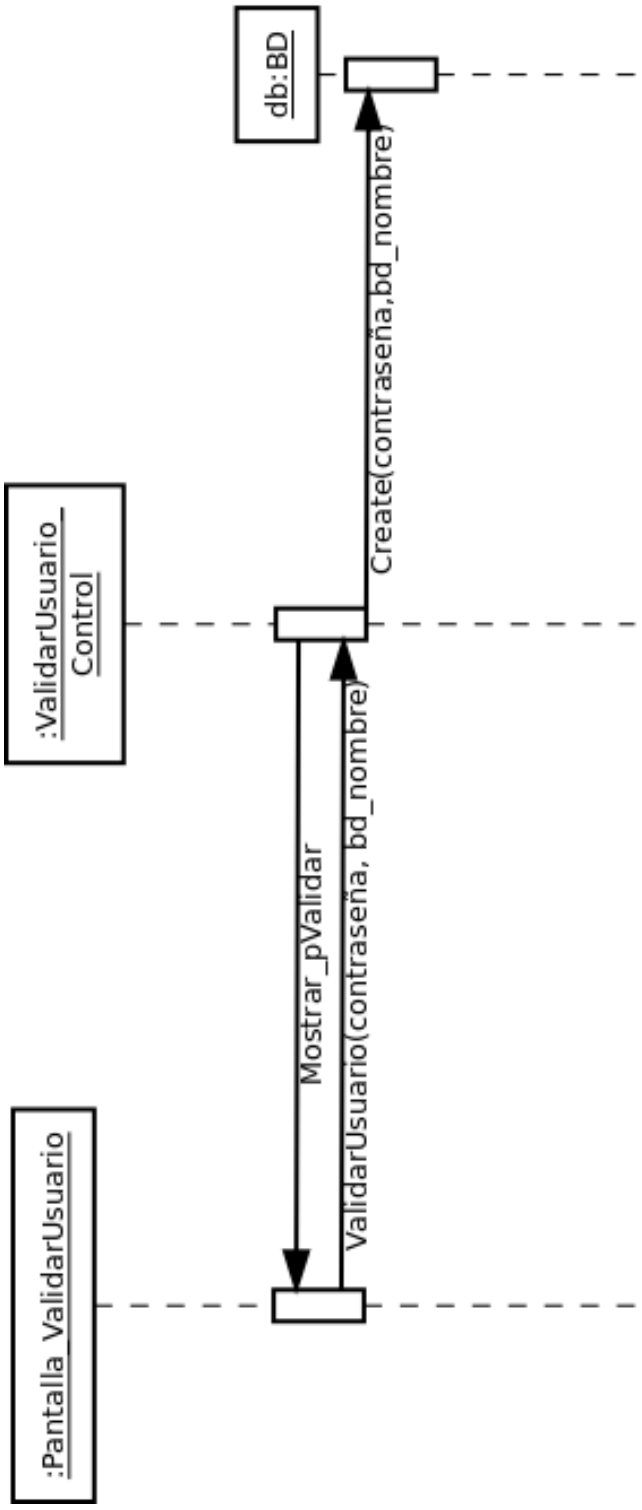

Figura 7.1: Diagrama de secuencia de Validar Usuario

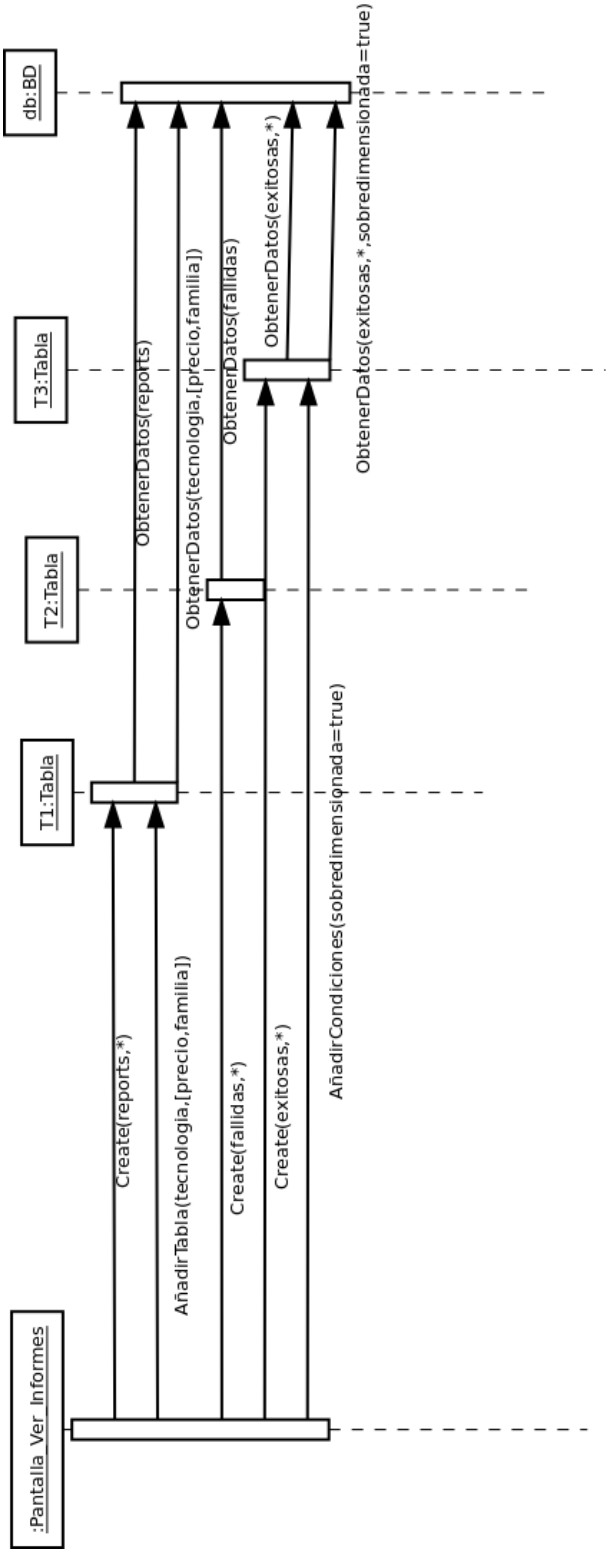

Figura 7.2: Diagrama de secuencia de Ver Datos de Informes

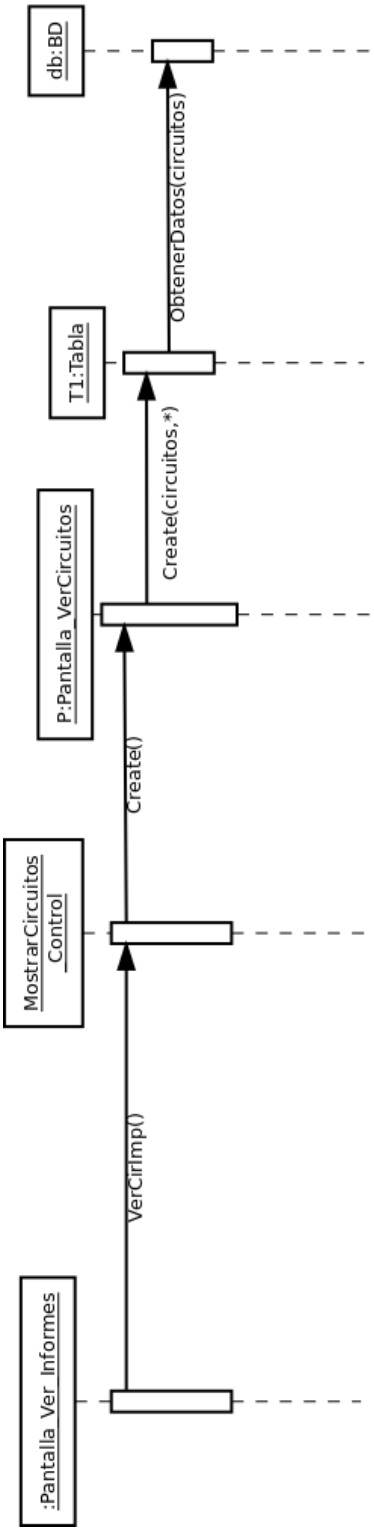

Figura 7.3: Diagrama de secuencia de Ver Circuitos Implementados

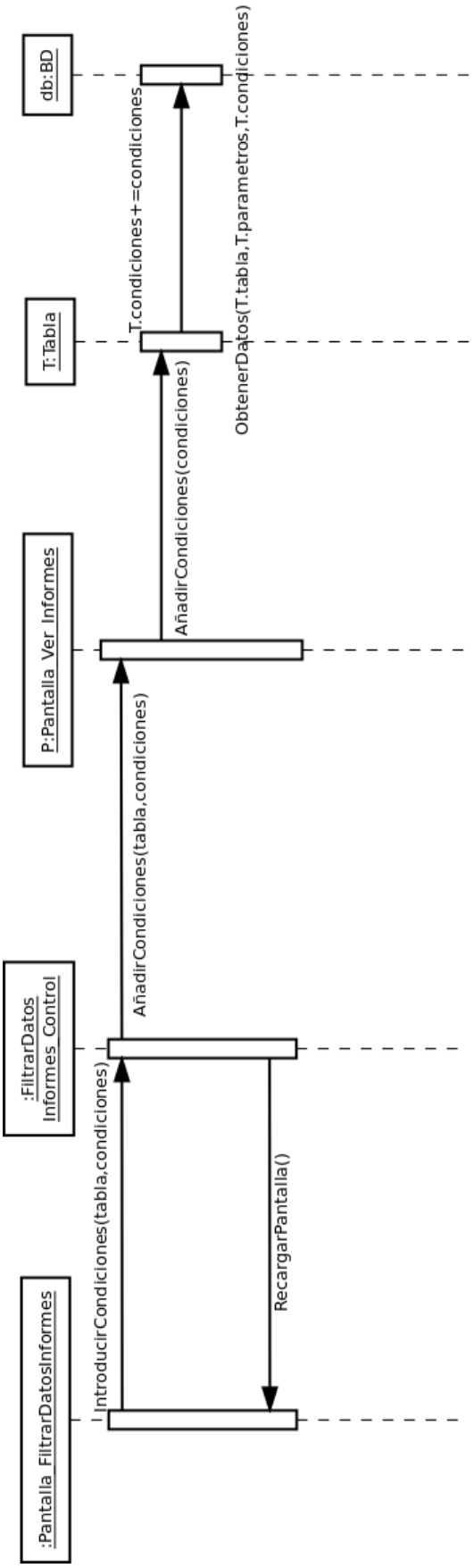

Figura 7.4: Diagrama de secuencia de Filtrar Datos Informes

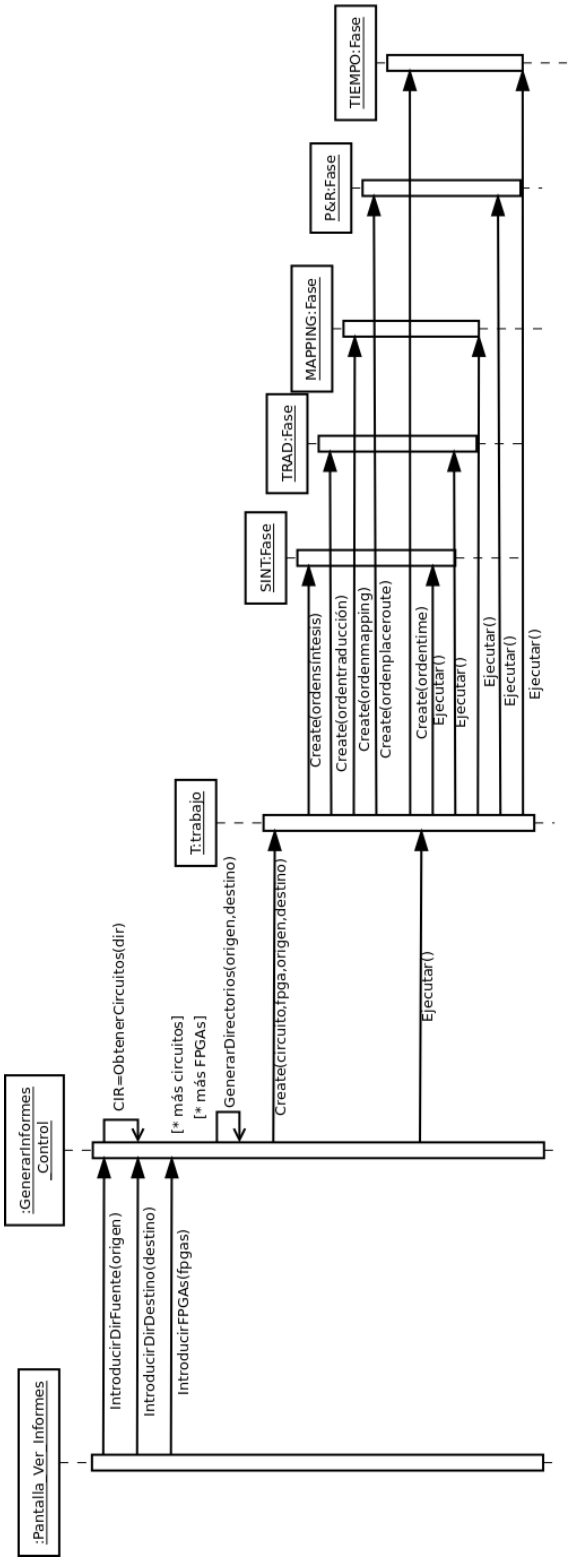

Figura 7.5: Diagrama de secuencia de Generar Informes

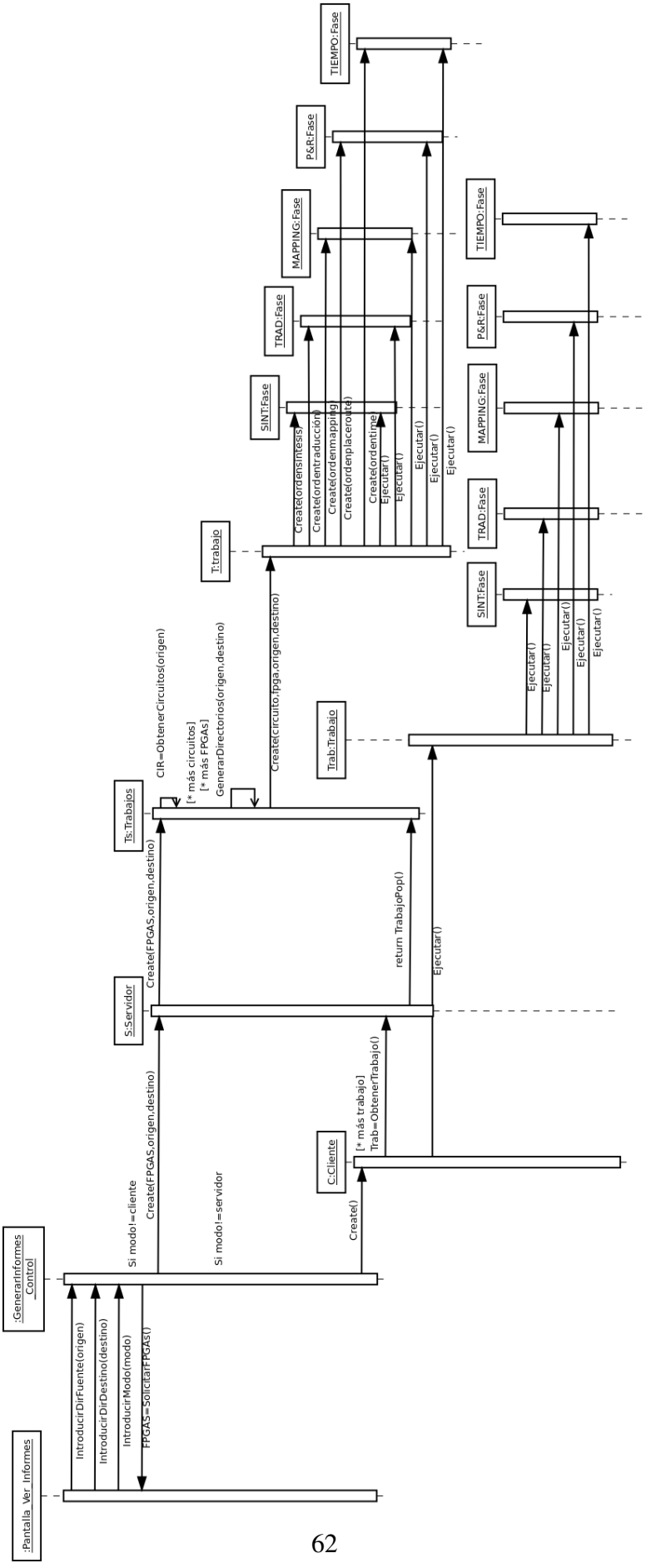

Figura 7.6: Diagrama de secuencia de Generar Informes con la estrategia Cliente-Servidor

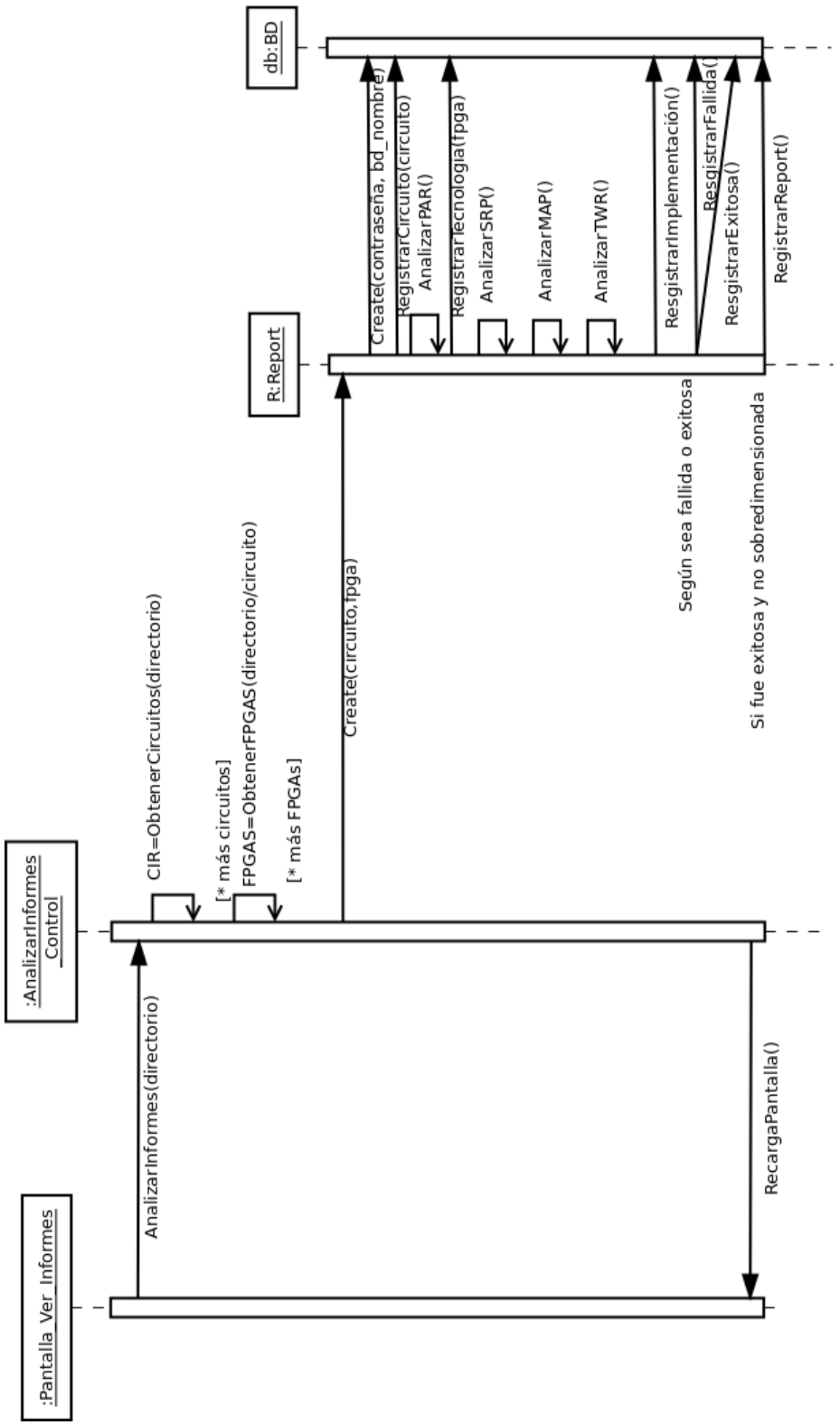

Figura 7.7: Diagrama de secuencia de Analizar Informes

Capítulo 8

Implementación

8.1. Fase de análisis

En esta sección se va a exponer un análisis de las necesidades del usuario para introducir al lector en el problema y entienda los pasos de solución que se irán dando a lo largo del documento.

El usuario tiene tres necesidades claras y bien definidas:

- Ejecutar un gran número de repeticiones de ciclos de desarrollo de circuitos digitales con FPGAs de forma desatendida.
- Lectura de los informes generados durante el ciclo de desarrollo, conversión de unidades y cálculos de magnitudes a partir de otras existentes en los informes y almacenamiento en una base de datos de los parámetros de interés.
- Análisis de los resultados leídos y almacenarlos en la base de datos.

A continuación se desarrollarán las tres necesidades, explicándolas de una maner más exhaustiva.

8.1.1. Ejecución de ciclos de desarrollo

Como se ha mencionado anteriormente una de las necesidades que el usuario presenta es el poder ejecutar ciclos completos de desarrollo de circuitos digitales sobre FPGAs. No obstante este proceso no se puede realizar de cualquier manera.

El usuario pretende poder variar dos parámetros de este ciclo, de manera que se puedan completar varios ciclos de desarrollo de diferentes circuitos sobre diferentes FPGAs de manera iterativa cambiando únicamente dos parámetros:

1. El circuito: El usuario desea evaluar diferentes circuitos de la misma temática, evaluando el rendimiento de cada uno de ellos y decidiendo científicamente cuál es el mejor. El único entorno similar a este es *ATHENa*, que permite evaluar circuitos de una sóla temática, la encriptación. El desarrollador pretende que ésta sea una aplicación más general que pueda ser utilizada con cualquier tipo de temática.

2. La FPGA o tecnología: De la misma manera que el usuario quiere evaluar el rendimiento de los circuitos digitales que tiene a su disposición, pretende evaluar dichos circuitos en diversas FPGAs para conocer cuál es la que mejor se adapta a dichos circuitos al mejor precio. Así, podría ocurrir que el circuito que mejor rendimiento diese necesitara una FPGA demasiado cara para cumplirlo. En cambio, un circuito que, en principio, no daba tan buenos resultados, se adapta muy bien a una FPGA que es más barata y da un rendimiento ligeramente inferior al primero.

El diseño y desarrollo de circuitos digitales en dispositivos FPGAs requiere la ejecución de cinco fases (síntesis, traducción, mapeo, emplazamiento y conexionado y análisis de tiempo) cada una de las cuales se realiza por una herramienta distinta. Teniendo en cuenta que el número de FPGAs en el mercado es del orden de cientos por fabricante, realizar esta tarea manualmente se presenta como un proceso arduo y repetitivo.

8.1.2. Lectura de informes

Los informes generados por las herramientas que llevan a cabo las diferentes fases tiene parámetros de interés para evaluar el rendimiento de un circuito digital en una FPGA. Algunos de éstos son:

- Número de slice: Un slice no es más que un bloque de lógica configurable. Las FPGAs tiene un número determinado de éstos que son utilizados por los circuitos digitales que en ella se implementan. Por tanto, cuantos menos slices utilice el circuito digital, menos espacio de la FPGA ocupará. Esto quiere decir que cabrá en una FPGA más pequeña y por tanto más barata.

- Número de entradas y salidas: Las entradas y salidas en las FPGAs son pines de comunicación con el exterior que éstas tienen soldados en sus placas. Al igual que ocurre con los slices, el número de pines en una FPGA es fijo y está predeterminado. Por tanto, si un circuito digital necesita pocas EntradasSalidas podrá implementarse en una FPGA que tenga menos de éstas y por tanto sea más barata.

- Número de multiplicadores: Los multiplicadores hardware son los encargados de realizar la operación aritmética que lleva el mismo nombre. Cuantos menos utilice un circuito, más barata será la FPGA en la que se pueda implementar.

- Memoria embebida: Algunos circuitos hacen uso de la memoria embebida. Ésta es un recurso interesante y limitado dentro de la FPGA. Gracias a ella se pueden liberar algunos slices que se utilizarán para otros menesteres.

- Frecuencia máxima: Viene determinada por el camino crítico, esto es, el máximo retardo que se produce entre dos registros del circuito. Este parámetro no aparece en los informes y debe ser calculado a partir de otro.

- Tiempo de establecimiento o setup: Cantidad de tiempo que la entrada de un dispositivo síncrono debe mantenerse estable antes del flanco de reloj. Cuanto menor sea esta cantidad, más rápido será el funcionamiento del circuito en la FPGA.

- Tiempo de espera o hold: Cantidad de tiempo que la entrada de un dispositivo síncrono debe mantenerse estable tras el ciclo de reloj. Cuanto menor sea esta cantidad, más rápido será el funcionamiento del circuito en la FPGA. Si este o el anterior valor se infringiera no se asegura el correcto funcionamiento del dispositivo, ya que se pueden tomar valores erróneos de dichas entradas.

La jerarquía de carpetas donde deben estar almacenados los reports puede visualizarse en la figura 8.1

8.1.3. Análisis de datos extraídos

No basta con presentar al usuario una interfaz amigable con los datos extraídos de los informes. El entorno a desarrollar debe ser capaz de ofrecer filtros de búsqueda para que el usuario pueda encontrar de una manera rápida y sencilla la mejor FPGA para sus necesidades. Así el usuario podrá quedarse con las FPGAs que cuesten menos de un determinado precio, que sean de una determinada familia o que el el tiempo de establecimiento sea menor que un determinado valor.

Las búsquedas anteriores es solo un subconjunto de aquellas que se pueden realizar. Además se pueden relacionar hasta cuatro condiciones mediante la conjunción y la disyunción con una sola búsqueda.

Los precios de las FPGAs constituyen una columna de la tabla tecnologías de la base de datos. No son modificables desde la interfaz, sin embargo si el usuario supiera manejarse con *MySQL* podría modificarlos sin problemas. Se considera que es importante mencionar esto porque los precios del mercado son muy cambiantes y la aplicación no está diseñada para durar los años que duren estos precios ni para que el usuario dependa del desarrollador para modificarlos.

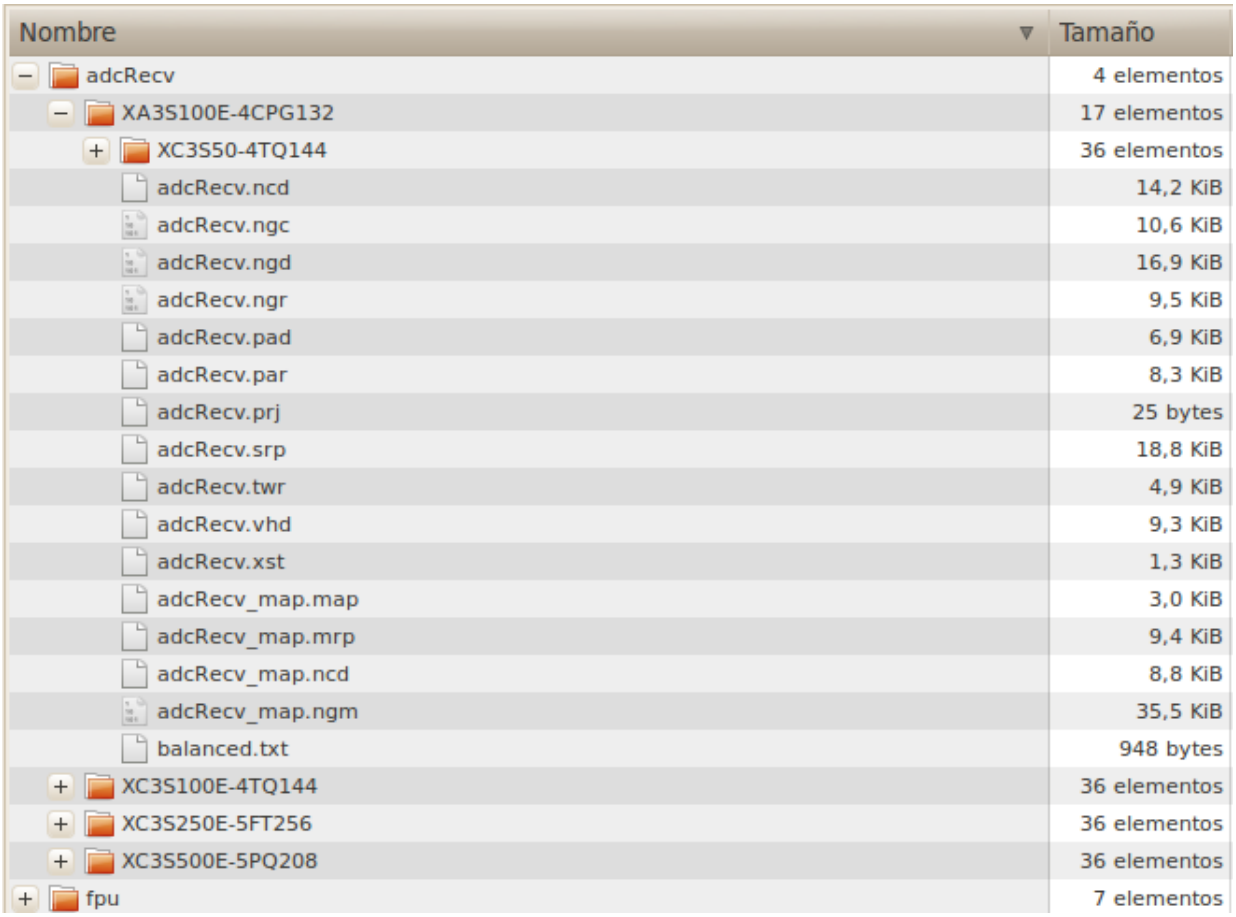

Figura 8.1: Jerarquía de carpetas para el análisis de informes

8.1.4. Otros aspectos

El usuario deberá tener instalada la aplicación *ISE Webpack* en la ruta por defecto que ofrece el instalador de ésta. En el caso de que tenga más de una versión en el equipo, el entorno interactuará con la más antigua.

Las formas de generar los informes y de analizarlos están muy relacionadas con la jerarquía de carpetas.

Para generar los informes se debe tener en el directorio que decida el usuario un conjunto de subdirectorios con el nombre de los circuitos con los que se va a completar el ciclo de diseño y desarrollo. El nombre de éstos viene determinado por el de la entidad de mayor jerarquía en el diseño. Además habrá un fichero llamado balanced.txt que contendrá la estrategia de síntesis que se debe seguir durante esta fase.

Dentro de cada uno de estos subdirectorios habrá, al menos, dos ficheros con el mismo nombre y distinta extensión. Uno será .vhd y el otro .prj. Éstos últimos son el diseño HDL en sí. Es importante mantener una homogeneidad entre el nombre de la carpeta contenedora de los ficheros del circuito, de la entidad de mayor jerarquía y el del fichero de configuración para cada herramienta. Este nombre debe ser el que tenga asignada la entidad superior del circuito. Si no es así, algunas de las herramientas producirán errores durante su ejecución.

8.1.5. ¿Qué había antes?

Antes de que este entorno comenzara a desarrollarse, el grupo de investigación con el que se ha trabajado había desarrollado unos scripts para Windows en batch, una aplicación en VisualBasic y una base de datos que resolvían parte del problema de una manera ruda y lenta. Se realizaron pruebas de estos scripts en un ordenador con procesador Intel QuadCore Q6600, 4GB de memoria RAM y Windows Vista como sistema operativo. Éstas tardaban en generar los informes correspondientes a 106 FPGAS, 16 circuitos y seis fases, las cincos fases citadas anteriormente más una adicional dedicada al análisis de potencia, un total de 89 horas y 10 minutos.

También se realizaron pruebas de la lectura, extracción y almacenamiento de informes en este mismo ordenador. Los resultados fueron poco prometedores. Se tardaba más de 30 minutos en realizar este proceso, el cual dejaba al ordenador en un estado que podía confundir al usuario con un posible bloqueo o cuelgue.

Durante las pruebas se estuvo monitorizando el equipo en el que se estaban ejecutando y se constató que *Xilinx ISE Webpack* sólo utilizaba uno de los cuatro núcleos del ordenador. Es decir, el entorno no estaba paralelizado, sino que era totalmente secuencial.

8.1.6. Primera aproximación

El último párrafo de la sección anterior es la base de la decisión que se toma y se mantiene a partir de este momento. Al no estar paralelizada la aplicación y no poder acceder a su código fuente al ser privativo había que encontrar cómo de alguna manera, externa por supuesto, se podía conseguir que varios ciclos de diseño se ejecutaran a la vez.

La solución a este problema fue tratar de distribuir en una granja de ordenadores (Cluster a partir de ahora), los distintos ciclos de diseño (Trabajos a partir de ahora). Cada trabajo estaría compuesto por las seis fases mencionadas anteriormente y se ejecutarían todas en el mismo equipo para evitar dependencias insatisfechas entre equipos remotos. Esto podría ocurrir cuando un equipo tratara de ejecutar una fase de un trabajo antes de que la anterior se hubiera terminado.

En cuanto se investigó a través de distintas fuentes cómo montar un cluster se observó que la información para sistemas operativos Windows era muy escasa. Por ello y por una apuesta hacia el software libre se decide utilizar un sistema GNU/Linux con una distribución que tuviera una interfaz gráfica fácil de manejar y aprender para usuarios inexpertos.

El lenguaje de programación también fue cambiado. En primer lugar, porque VisualBasic no es compatible con sistemas GNU/Linux y en segundo lugar porque éste es un lenguaje lento, que no saca el mejor partido de la orientación a objetos y que no está aconsejado para aplicaciones de gran evergadura. La intención, desde este momento, es encontrar un lenguaje más eficiente que fuera compatible con ambos sistemas operativos por si en un futuro fuera viable un clúster con Windows como sistema operativo.

Microsoft Access tampoco es compatible con sistemas GNU/Linux y por tanto quedaba descartado para su uso. Además tiene limitaciones en el procesamiento de búsquedas y es poco estable. El paso natural era migrar a una alternativa libre, más robusta y compatible con ambos sistemas operativos pero que no tenía por qué requerir alta disponibilidad.

8.2. Montaje del cluster

El cluster que se va a montar se enmarca en el grupo de los de alta eficiencia ya que su labor consiste en completar el máximo número de ciclos de diseño y desarrollo en el menor simple. No se espera que vaya a tener que atender a un gran número de peticiones a la base de datos ni su caída es crítica. Completar ciclos de diseño y desarrollo puede considerarse una tarea ardua y repetitiva pero en ningún caso requiere cálculos complicados por lo que la potencia de los nodos tampoco supone un problema.

El prototipo consta de tres ordenadores: Todos ellos trabajan con la distribución linux Ubuntu 10.04 como sistema operativo.

	Máster	Nodo1	Nodo2
Procesador	Pentium 4 doble núcleo 3GHz	Athlon XP 2000+	Athlon 3500+
Memoria	489MB	465MB	1002MB

Tabla 8.1: Características de los tres primeros equipos

Físicamente están conectados mediante un switch con conexión ethernet a la velocidad máxima de 100 Mbit, mientras que la paralelización a nivel de software se ha realizado con MPICH2 (una implementación libre y portable de MPI, Message Passing Interface).

El switch le proporciona una IP pública y estática a cada PC. Esta IP estática es necesaria, o al menos una IP reservada en DHCP, puesto que sería imposible la comunicación entre los nodos si cambiaran de dirección en cada conexión.

También ha sido necesario instalar los siguientes paquetes presentes en los repositorios oficiales de Ubuntu:

- libmpich1.0-dev - librerías estáticas de desarrollo mpich
- libmpich-mpd1.0-dev - librerías estáticas y ficheros de desarrollo mpich
- libmpich-shmem1.0-dev - librerías estáticas y ficheros de desarrollo mpich
- mpich2 - Implementación de la estándar MPI
- mpich2-doc - Documentación para MPICH2
- openssh-server - Shell para el acceso remoto seguro
- nfs-kernel-server - Solo en el master para tener ficheros compartidos en la red
- nfs-common - En todos los equipos para acceder al sistema de ficheros compartidos
- libnfsidmap2 - Dependencia de los dos archivos anteriores
- wakeonlan - Permite enceder los nodos remotamente
- ethtool - Herramienta para configurar la tarjeta de red para wakeonlan. Además es necesario configurar la BIOS

Openssh-server además de permitir la comunicación entre los distintos nodos posibilita controlar el cluster desde un ordenador remoto.

8.2.1. Configuración de NFS

Se ha configurado un servidor NFS en el master ya que es necesario que todos los nodos tengan acceso a los programas y ficheros que se utilicen en los distintos momentos. Este servidor reside en el PC maestro del cluster y los demás se conectan automáticamente a él gracias a la configuración de los ficheros /etc/fstab y /etc/exports. Téngase en cuenta que la dirección de la red local es 192.168.1.0 con máscara de subred 255.255.255.0.

Para compartir una carpeta con NFS debe crearse un directorio en el servidor NFS, imagínese que se llama /home/juan/cluster, y añadir la siguiente línea en su fichero /etc/exports:

```
/home/juan/cluster *(rw,sync)
```

De esta manera se exporta a dicha carpeta a cualquier cliente de la red con permisos de lectura y escritura en la carpeta.

Tras esto se reinicia el servidor NFS:

```
/etc/init.d/nfs-kernel-server restart
```

Y se pasa a la configuración de los nodos. Para ello el usuario creará una carpeta vacía la cual contendrá, tras la configuración, la carpeta compartida. Supóngase que esta nueva carpeta es /home/bwana/cluster pues habría que añadir la siguiente línea al fichero /etc/fstab de este nodo:

```
192.168.1.100:/home/juan/cluster /home/bwana/cluster nfs rw,sync,hard,int
0 0
```

8.2.2. Configuración MPICH

MPICH es configurable, al igual que la mayoría de las aplicaciones UNIX, mediante ficheros. En este caso hay que modificar tres ficheros: dos de la aplicación y uno del sistema operativo.

El fichero mpd.conf se encuentra oculto en el directorio casa del equipo. Este fichero únicamente contiene la cadena secretword=tupassword donde tupassword es una claver que requerirá mpd cuando el usuario pretenda correr un programa como root.

El fichero mpd.hosts contiene la ip de cada equipo y el número de procesadores de los que dispone con el siguiente formato:

```
1 192.168.1.100:1
2 192.168.1.101:1
3 192.168.1.102:1
4 192.168.1.103:1
```

Por último, hay que modificar el fichero /etc/hosts para darle un nombre a cada equipo de la red, de manera que no sea necesario dar la ip en cada llamada.

```
#..............
127.0.0.1       localhost
192.168.1.100   cluster-desktop
192.168.1.101   cluster-desktop-1
192.168.1.102   cluster-desktop-2
192.168.1.103   cluster-desktop-3
192.168.1.104   cluster-desktop-4
#.............
```

Únicamente se ha mostrado la parte relevante de este fichero, hay más contenido tras estas líneas.

Los ficheros mpd.hosts y /etc/hosts deben ser compartidos por todos los nodos que colaboren en el cluster.

8.3. Ejecutando programas

El primer programa que se consigue ejecutar paralelamente es cpi, el cual calcula el número pi geométricamente.

```
#include <mpi.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
int main( int argc, char *argv[] )
{
    int n=100, myid, numprocs, i;
    double PI25DT = 3.14159265358979323846264643;
    double mypi, pi, h, sum, x;
    clock_t comienzo;
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    printf("Somos %i procesos\n",numprocs);
    printf("Mi id es %i\n",myid);
    while (1) {
        if (myid == 0) {
            printf("Enter the number of intervals: (0 quits)
                ");
            scanf("%d",&n);
        }
```

Nodos-Intervalos	100.000.000	1.000.000.000
1	1,55s	15,40s
2	0,77s	7,71s
3	1,55s	15,40s

Tabla 8.2: Tiempos de ejecución de cpi.c

```
21          comienzo=clock();
22          MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
23          if (n == 0)
24              break;
25          else {
26              h   = 1.0 / (double) n;
27              sum = 0.0;
28              for (i = myid + 1; i <= n; i += numprocs) {
29                  x = h * ((double)i - 0.5);
30                  sum += (4.0 / (1.0 + x*x));
31              }
32              mypi = h * sum;
33              printf("mypi de momento vale %.16f y soy %i\n",
                    mypi,myid);
34              MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM,
                    0,
35                              MPI_COMM_WORLD);
36              printf("pi despues vale %.16f y soy %i\n",pi,
                    myid);
37              if (myid == 0)
38              {
39                      printf("pi is approximately %.16f, Error
                            is %.16f\n", pi, fabs(pi - PI25DT));
40                  printf ("El tiempo invertido ha sido de %f s
                        \n",(clock()-comienzo)/(double)
                        CLOCKS_PER_SEC);
41              }
42          }
43      }
44      MPI_Finalize();
45      return 0;
46  }
```

Los tiempos obtenidos se pueden observar en la tabla 8.2

También se intentó desencriptar una contraseña MD5 de 128 bits empleando únicamente el master. Sin embargo dicha ejecución debió abortarse tras 22 horas 40 minutos 54 segundos sin ningún resultado. Posteriormente se trató de desencriptar la constraseña utilizando los tres PCs. Dicha prueba no arrojó ningún resultado en 5 días, momento en el cual un fallo eléctrico provocó la caída del clúster.

Nodos-Intervalos	100.000.000	1.000.000.000
1	0,95s	9,52s
2	0,63s	6,33s
3	0,42s	4,22s
4	0,31s	3,18s

Tabla 8.3: Tiempos de ejecución para cpi

8.4. Ampliando el cluster

El switch que se estaba utilizando momentáneamente pertenecía a un aula de la escuela por lo cual tuvo que ser devuelto a su origen, sustituyendo a este un router Cisco wifi con cuatro conexiones por cable para los PCs. Fue necesario desmilitalizar el router de forma que el clúster pudiera ser controlado remotamente por lo que se redireccionó cualquier acceso remoto a la IP interna 192.168.1.100 (el nodo máster). El router asignó direcciones IPs privadas a los nodos por lo que fue necesario cambiar la configuración de los ficheros mpd.conf y /etc/hosts. Hecho por el cual se aprovechó para añadir un nodo más y cambiar el master por otro PC. La configuración del cluster quedaría así:

- Tres ordenadores con un procesador Athlon 3500+ y 1002 MB, uno de ellos actúa como master
- Un ordenador con un procesador Athlon XP 2000+ y 465 MB de memoria RAM

Como prueba del aumento de potencia se exponen los resultados obtenidos con el programa cpi mostrado anteriormente en la tabla 8.3

8.5. Monitorización

A la vista de que acceder a cada nodo para comprobar su carga es algo bastante engorroso se ha buscado una herramienta cómoda y fácil de manejar para la monitorización del cluster.

Ganglia fue la herramienta elegida, pues puede controlar magnitudes individuales de cada nodo, como son el número de procesos ejecutándose, el estado de la memoria o el flujo local de la red; y globales como pueden ser el número de nodos que forman el cluster y el estado de cada uno (en línea, caído, porcentaje de carga). Ganglia es una herramienta libre disponible en los repositorios de Ubuntu y posee una interfaz web que permite poder observar el estado del clúster sin tener que acceder físicamente a éste. Es por esto por lo que se hace necesario instalar un servidor apache en el nodo master. Como contra de este software se podría decir que la interfaz que ofrece es más bien de caracter informativo y que está limitada en unidades de medida y rangos de observación. La configuración se realiza por medio de ficheros de texto,

como la mayoría de las aplicaciones que se ejecutan en sistemas UNIX. Esto hace sea bastante flexible pero incomprensible para un usuario novel. Puede verse una imagen de ganglia en la figura 8.2

8.6. Scripts concurrentes

Una de las motivaciones para montar el cluster fue el poder ofrecer una respuesta en un tiempo razonable a la pregunta ¿Qué FPGA se adapta mejor al circuito X? Existía una aplicación anterior que tenía como componentes unos scripts en batch para el sistema operativo Windows que implementaba una serie de circuitos, en concreto 16, en una lista de 105 FPGAs predefinidas. Este script tardaba 89 horas y 10 minutos al ejecutarse en un intel quad core. Como las herramientas de ISE no están paralelizadas, el tener un quad-core era irrelevante, ya que lo utilizaban como un mono-núcleo.

Al ser privativo el código de estas aplicaciones la única salida posible es repartir las distintas fases que se han de llevar a cabo para implementar el circuito entre distintos procesadores ó núcleos. Para ello se desarrollan varios programas.

8.6.1. Primer intento: Polling

El primero en lenguaje bash trata de paralelizar mediante la herramienta ssh, pero tiene el problema de que para saber cuándo ha acabado un nodo de ejecutar la tarea encomendada es necesario estar constantemente preguntándole. Esto no hace más que sobrecargar al nodo que debe encargarse, tanto de ejecutar la tarea como de responder constantemente al master. Además, la manera de preguntarle al nodo es algo rudimentaria y esto hace que a veces el nodo master se quede algo más relajado que el resto. Se descarta por tanto el script por:

- Colapsar nodos por falta de memoria principal. Figura 8.4
- No aprovechar, en algunos momentos de la ejecución, el máximo de los nodos.
- Nula portabilidad a otros sistemas operativos.

Véase el diagrama de flujo del script que realiza el polling a todos los nodos: Figura 8.5

Mientras esto ocurre el nodo está llevando a cabo las cinco fases del último trabajo que se le ha encargado a la vez que va respondiendo al master sobre su situación de libertad.

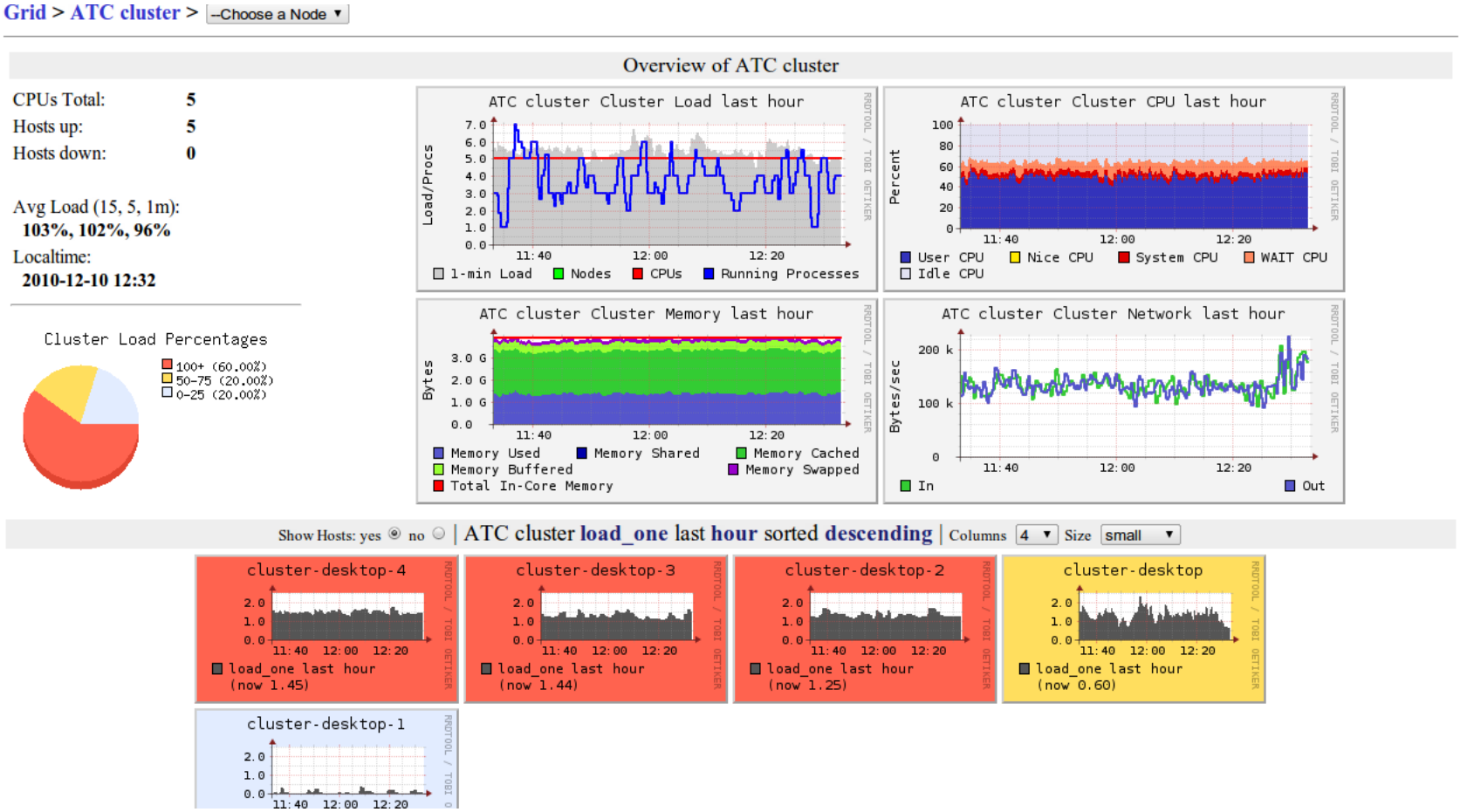

Figura 8.2: Estado del cluster durante la ejecución de scripts en bash

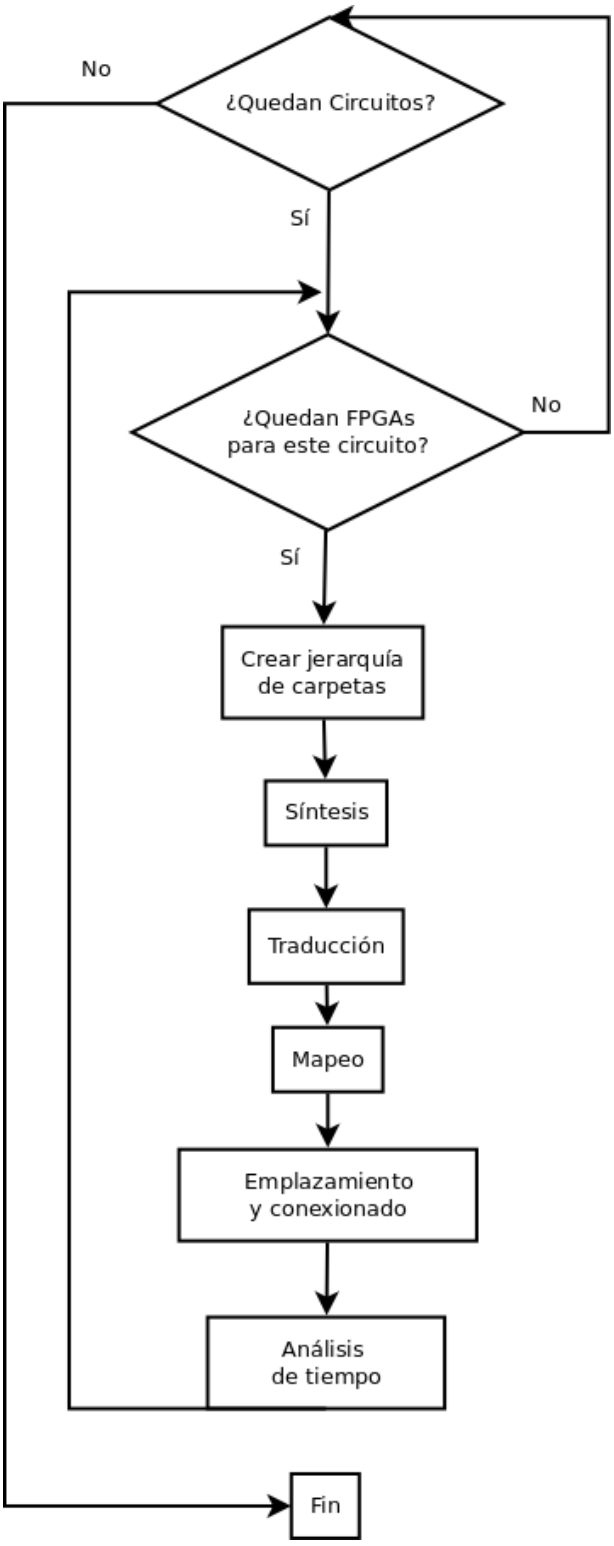

Figura 8.3: Diagrama de flujo de los scripts anteriores

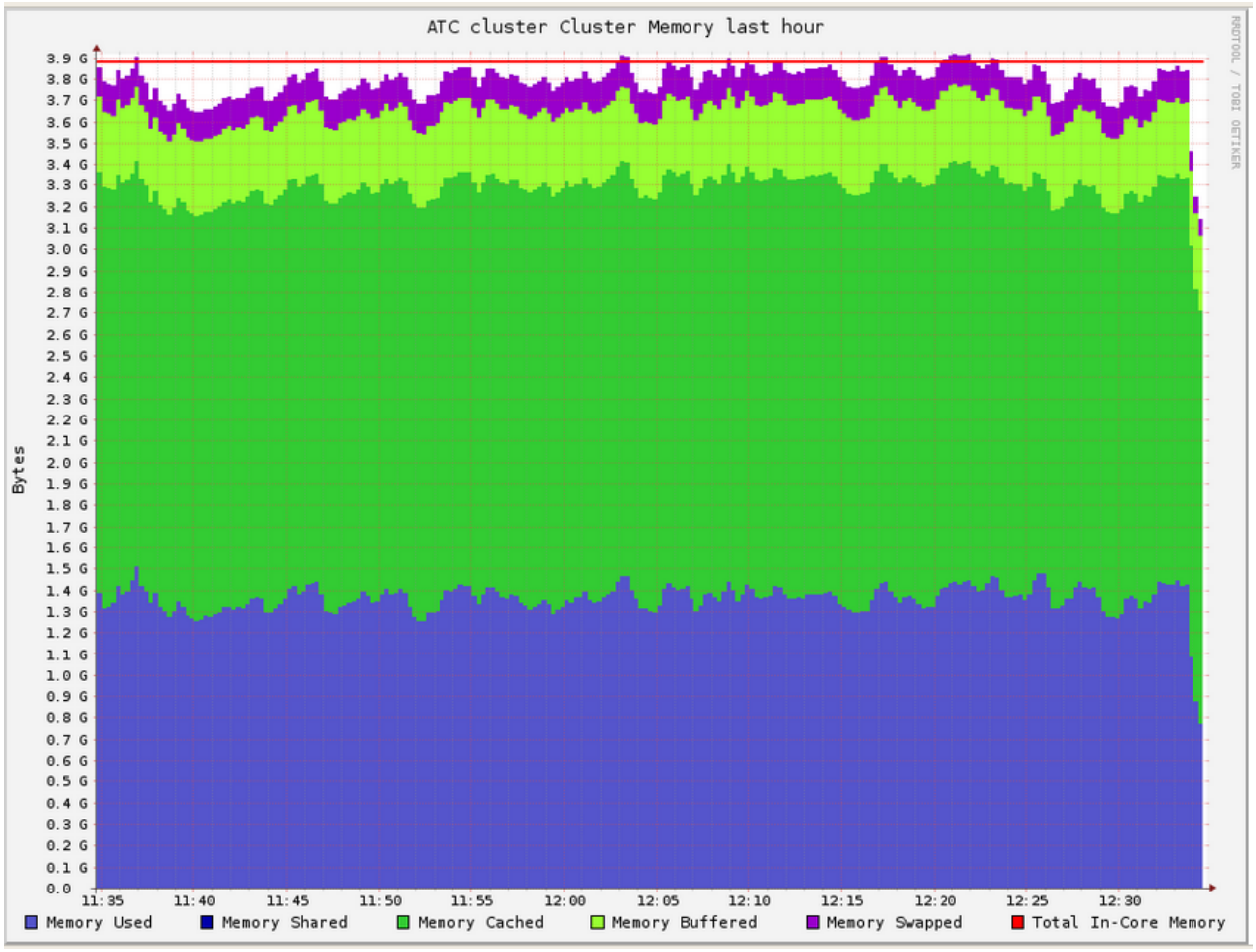

Figura 8.4: Estado de la memoria del cluster durante la ejecución del script mostrado por Ganglia

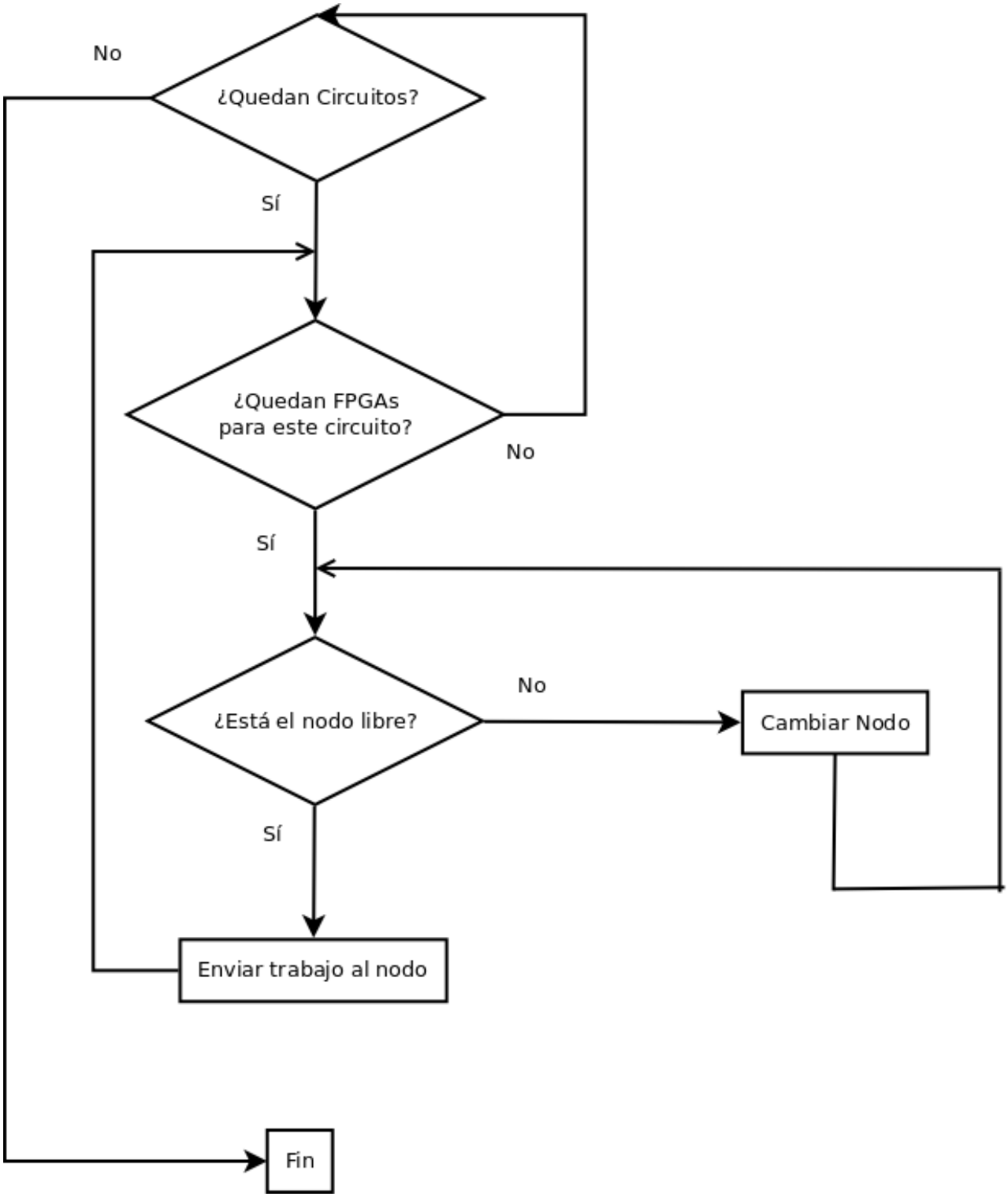

Figura 8.5: Diagrama de flujo de los scripts polling

8.6.2. Segundo intento: Preasignación de tareas

El segundo programa trata de eliminar las carencias del primero utilizando las librerías MPI y python como lenguaje. Atendiendo a ejemplos de utilizaciónde estas librerias, se preasignan trabajos al inicio del programa. En consecuencia se obtiene un master más sobrecargado, con una media de carga por encima del 100 %) y unos nodos entre el 90 % y el 100 % de carga y con una utilización más razonable de la memoria. La preasignación conlleva el problema de que si un nodo acaba antes, se queda ocioso aunque haya trabajos pendientes, ya que éstos no le corresponden. Esta solución en un sistema simétrico ideal, donde todos los ordenadores fueran exactamente iguales funcionaría de manera óptima, puesto que, al ser todos idénticos, todos acabarían las tareas a la vez. En cambio ésta no es una situación real, ya que los sistemas suelen ser heterogéneos.

El diagrama de flujo de este último script es muy parecido al primero, solo que en lugar de realizar todos los trabajos un mismo nodo, cada uno de éstos ejecuta un subconjunto de tamaño similiar. Si hay N nodos y T trabajos cada nodo ejecutará T/N trabajos

8.6.3. Tercer intento: Presencia de ficheros

Como tercera solución, se utilizan los ficheros y directorios que se generan en las distintas fases como forma de saber si un trabajo se ha realizado ya, o no. De esta manera, todos los nodos tratan de reralizar todos los trabajos, pero sólo lo realizará aquel que llegue primero. Arrojó unos resultados de 21 horas 45 minutos. Sin embargo, este tiempo está muy abultado por los accesos constantes al sistema de ficheros compartidos, lo cual, además, genera mucho tráfico en la red.

La idea por tanto es conseguir que cada nodo avise de que ya ha terminado su trabajo para que el máster le asigne uno nuevo. De esta manera, se ahorra el acceso al sistema de ficheros compartido y se evita preguntar constantemente al nodo si ha acabado su tarea.

8.6.4. Cuarto intento: Objetos remotos

Esta es la solución más elegante. Consiste en aprovechar las funcionalidades de Pyro para implementar una estrategia cliente-servidor.

El servidor controlará los trabajos que quedan por realizar y se registrará en un servidor de nombres para que el acceso a él por parte de los clientes sea más sencillo. El servidor es responsable de que dos clientes no obtengan el mismo trabajo.

El cliente será el encargado de solicitar trabajos al servidor y ejecutarlos localmente. De esta manera se evita tener que consultar el sistema de ficheros compartido para comprobar si alguien

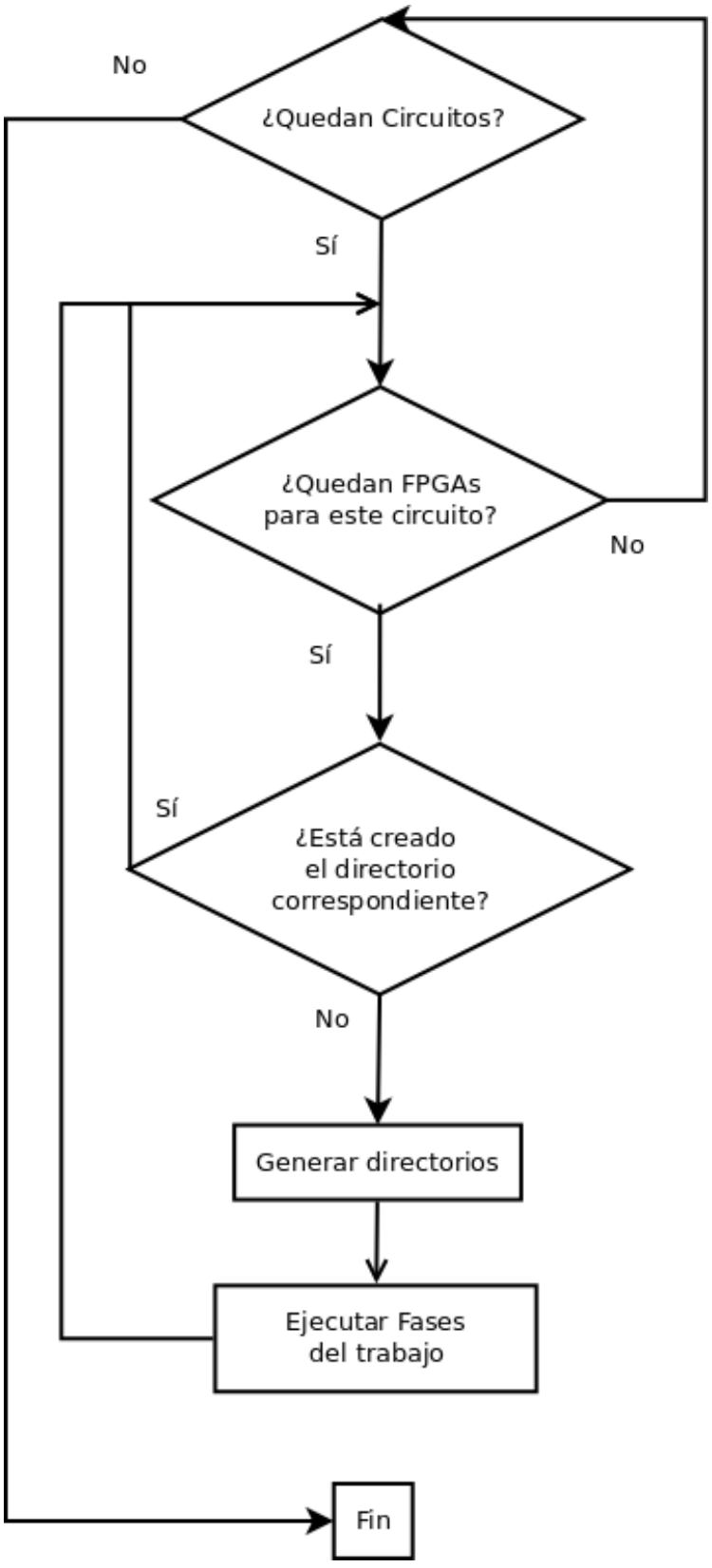

Figura 8.6: Diagrama de flujo scripts ficheros

Algoritmo	Lenguaje	Tiempo	Carga del sistema
Aplicación previa a THOr	Batch	89 horas y 19 minutos	Desconocida
Polling	Bash	34 horas 43 minutos	144 %
Preasignación de tareas	Python	27 horas	90 %
Presencia de ficheros	Python	21 horas 45 minutos	90 %-110 %
Ojetos remotos	Python	21 horas	120 %

Tabla 8.4: Tiempos de ejecución de los scripts

ya ha realizado el trabajo, lo cual supone un importante ahorro en tráfico por la red y una disminución de consultas al sistema de ficheros compartidos.

La dificultad de implementación de esta solución reside en el manejo de la exclusión mutua en un sistema distribuido. Al no compartir memoria no se pueden utilizar semáforos, variables compartidas o monitores y el algoritmo de Ricart-Agrawala no era la mejor solución en este caso.

Los objetos remotos permiten que objetos locales en un nodo sean visibles en equipos externos y llamar a sus métodos. De esta manera el problema de la exclusión mutua pasa a ser local protegiendo el recurso compartido en el servidor mediante un cerrojo o región crítica. Así, es seguro que nunca dos clientes obtendrán el mismo trabajo para realizar.

El entorno está preparado para poder elegir el modo de funcionamiento de cada equipo. De esta manera, un equipo puede ser:

- Cliente: Se encarga de solicitar y ejecutar los trabajos.
- Servidor: Atiende a las peticiones de los clientes manejando la lista de trabajos adecuadamente.
- Cliente y servidor: Atiende a las peticiones de los clientes y participa también en la ejecución de los trabajos.

Los resultados de tiempo que se muestran en este documento corresponden a una configuración de un nodo actuando como cliente y servidor y los tres restantes actuando como clientes.

8.6.5. Resumen de intentos

En la tabla 8.4 puede visualizar los tiempos que tarda cada script en completar el ciclo de desarrollo de 1680 diseños.[1]

Como resumen de cada tipo de script se puede decir que:

[1] Se puede observar que los tres scripts mejoran los tiempos de los anteriores.

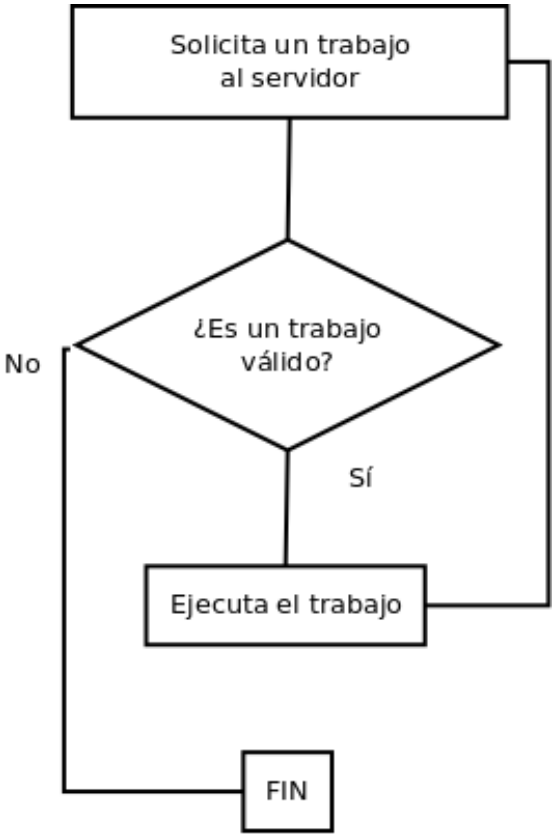

Figura 8.7: Diagrama de flujo del cliente

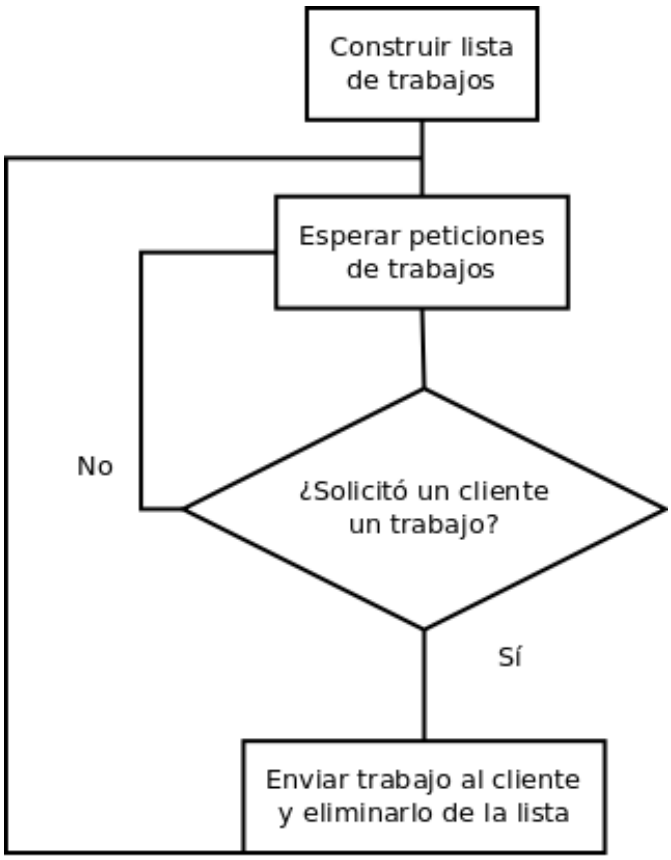

Figura 8.8: Diagrama de flujo del servidor

- La estrategia de polling es muy costosa, innecesaria y sobrecarga a los nodos.
- La segunda estrategia sería óptima si los nodos del cluster fueran homogéneos, lo cual rara vez ocurre.
- La tercera solución es tarda menos tiempo y permite una adición de nodos de manera sencilla. Sin embargo, su tiempo es reducible evitando consultas al sistema de ficheros compartido.
- La cuarta solución conserva las ventajas de la tercera y evita consultas al sistema de fichero. Por contra, todo el trabajo depende del servidor. Si este cayera, los clientes no sabrían qué hacer.

8.7. Base de datos

Dado que ISE produce los resultados de las implementaciones en ficheros de texto, se hace necesario mantenerlos en un formato más manejable que, en este caso, será una base de datos. Este es el diagrama ERE de la base de datos que se implementará en mysql:

La elección se MySQL no es trivial. MySQL es un sistema libre y portable entre distintos sistemas operativos. Además consume pocos recursos, algo crítico en el sistema que se ha montado debido a la poca capacidad de los nodos cuando actúan por sí solos. Por último, las comparativas existentes entre sistemas de gestión de bases de datos dicen que MySQL se integra mejor que otros libres, como postgresql, en sistemas PHP. Esto debe ser tenido en cuenta, ya que existe la posibilidad de que en un futuro la aplicación se sirva vía web.

8.8. Manipulando ficheros

Las bases de datos deben llenarse con datos y éstos datos hay que sacarlos de los ficheros que produce la aplicación ISE.

Se han desarrollado varias funciones para realizar el análisis de los ficheros, siendo las tres principales:

- **buscar(nombrefich, patron)**: Busca en el fichero llamado nombrefich la cadena patron y devuelve el valor que está a su derecha. Si está expresado en varias unidades los unifica en una.
- **buscarlt(nombrefich,patron)**: Busca en el fichero llamado nombrefich la última ocurrencia de la cadena patron y devuelve el valor que está a su derecha. Si está expresado en varias unidades los unifica en una.

- **leer_columna(nombrefich,patron,numlin,sep,col)**: Lee la columna col de fichero llamado nombrefich que se encuentra numlin lineas después de la cadena patron. Las columnas están delimitadas por el carácter sep.

En la tabla tecnologías se almacenarán todos aquellos parámetros característicos de ésta como pueden ser los bits disponibles, los multiplicadores disponibles, los slices disponibles, etcétera.

En la tabla exitosas se almacenarán las unidades lógicas o slices utilizados y los pines utilizados. Si alguna de estas unidades es superior a las que la FPGA tiene disponibles (se pueden consultar en tecnologias) se marca el atributo sobredimensionada a verdadero.

Por último en la tabla reports se almacenarán aquellos parámetros que no hemos tenido en cuenta hasta ahora y que sólo estarán disponibles si se pudo completar el ciclo completo de diseño del circuito en la FPGA. Estos son el tiempo de ejecución de cada fase, la memoria máxima utilizada a lo largo del ciclo de diseño, la frecuencia máxima, etcétera.

8.8.1. Fichero SRP

De este fichero se obtendrán los datos de tiempo de síntesis (T_sint) y el pico de memoria más empleado en esta fase. Dicho pico será comparado con los requeridos en el resto de fase, de forma que en la base de datos sólo se guardará el máximo.

Este fichero contiene aproximadamente un total de 451 líneas de las que interesan las siguientes:

```
.................
 Total CPU time to Xst completion: 44.18 secs
 Total memory usage is 206812 kilobytes
.................
```

8.8.2. Fichero PAR

Del fichero PAR se obtendrán los datos de:

- Pines disponibles y pines usados.
- Multiplicadores disponibles y multiplicadores usados.
- LUTs o slices disponibles y LUTs usadas.

- Bits disponibles y bits usados. En los ficheros no aparecen los bits en sí, sino que contienen la información de los bloques de memoria que se utilizan. Estos bloques de memoria tienen un tamaño de 18KB, por lo que hay que obtener a partir de éstos el total de bits utilizados. Se realiza una conversión a bits debido a que los ficheros que se generan si se trabajara con otros fabricantes presentan este valor en otras unidades y es necesario homogeneizarlas para compararlas con más fidelidad.
- Pico de memoria necesitado en esta fase.
- Tiempo en completar esta fase.

Suele rondar las 250 lineas de las que interesan las aquí abajo especificadas:

```
.............
Device Utilization Summary:
Number of BUFGMUXs   1 out of 8 12%
Number of External IOBs 31 out of 124 25%
Number of LOCed IOBs  0 out of 31 0%
Number of RAMB16s  1 out of 4 25%
Number of Slices  915 out of 768 119%
Number of SLICEMs  169 out of 384 44%
Number of MULT18X18s  2 out of 4 50%


Total CPU time to PAR completion: 1 mins 51 secs
Peak Memory Usage: 288 MB
.............
```

8.8.3. Fichero MAP

Del fichero MAP se obtendrán:

- El tiempo empleado realizar esta fase
- Pico de memoria necesitado en esta fase

De las 80 líneas que ronda el fichero interesan las que aparecen con el formato expresado a continuación:

```
.................
Peak Memory Usage: 288 MB
Total CPU time to MAP completion: 1 mins 51 secs
.................
```

8.8.4. Fichero TWR

Este es el fichero donde se guardan los resultados del análisis de tiempo. Tiene una longitud aproximada de 110 líneas y de él interesan los siguientes datos:

- Frecuencia máxima a la que puede trabajar el circuito. Se calcula como el inverso del parámetro del report denominado "Clock to setup on destination clock".
- Tiempo de establecimiento o setup.
- Tiempo de espera o hold.
- Pico de memoria utilizado en esta fase.

Los tiempos de ejecución de cada herramienta no aparecen expresados en los ficheros con las misma unidades por lo que se convierten todos a segundos antes de ser almaceandos en la BD al objeto de mejorar las comparativas.

Los parámetros anteriores están representados en el siguiente formato:

```
............
Setup/Hold to clock CLK_I
------------+------------+------------+-----------------+--------+
| Setup to |        |Hold to |                    | Clock |
Source | clk (edge) | clk (edge) | Internal Clock(s) | Phase |
------------+------------+------------+-----------------+--------+
CE_I| 4.418(R)| 1.157(R)|CLK_I_BUFGP | 0.000|
DATA_I<0> | -0.422(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
DATA_I<1> | -0.480(R)| 1.378(R)|CLK_I_BUFGP| 0.000|
DATA_I<2> | -0.422(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
DATA_I<3> | -0.480(R)| 1.378(R)|CLK_I_BUFGP| 0.000|
DATA_I<4> | -0.422(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
DATA_I<5> | -0.480(R)| 1.378(R)|CLK_I_BUFGP| 0.000|
DATA_I<6> | -0.422(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
DATA_I<7> | -0.480(R)| 1.378(R)|CLK_I_BUFGP| 0.000|
KEY_I<0> | -0.422(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
KEY_I<1> | -0.480(R)| 1.378(R)|CLK_I_BUFGP| 0.000|
KEY_I<2> | -0.422(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
KEY_I<3> | -0.480(R)| 1.378(R)|CLK_I_BUFGP| 0.000|
KEY_I<4> | -0.422(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
KEY_I<5> | -0.480(R)| 1.378(R)|CLK_I_BUFGP| 0.000|
KEY_I<6> | -0.422(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
KEY_I<7> | -0.480(R)| 1.378(R)|CLK_I_BUFGP| 0.000|
RESET_I | 3.813(R)| 1.157(R)|CLK_I_BUFGP| 0.000|
VALID_DATA_I| 4.680(R)| 1.332(R)|CLK_I_BUFGP| 0.000|
```

```
VALID_KEY_I | 1.419(R)| 1.128(R)|CLK_I_BUFGP| 0.000|
------------+-----------+-----------+-----------------+--------+
Clock CLK_I to Pad
-----------+-----------+-----------------+--------+
| clk (edge) |                        | Clock |
Destination | to PAD |Internal Clock(s) | Phase |
------------+-----------+-----------------+--------+
DATA_O<0> | 9.894(R)|CLK_I_BUFGP| 0.000|
DATA_O<1> | 9.523(R)|CLK_I_BUFGP| 0.000|
DATA_O<2> | 10.048(R)|CLK_I_BUFGP| 0.000|
DATA_O<3> | 9.613(R)|CLK_I_BUFGP| 0.000|
DATA_O<4> | 9.894(R)|CLK_I_BUFGP| 0.000|
DATA_O<5> | 9.961(R)|CLK_I_BUFGP| 0.000|
DATA_O<6> | 9.894(R)|CLK_I_BUFGP| 0.000|
DATA_O<7> | 9.967(R)|CLK_I_BUFGP| 0.000|
KEY_READY_O | 7.809(R)|CLK_I_BUFGP| 0.000|
VALID_O | 9.961(R)|CLK_I_BUFGP| 0.000|
------------+-----------+-----------------+--------+
Clock to Setup on destination clock CLK_I
--------------+---------------+---------+---------+---------+
| Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
-----------------+-----------+---------+---------+---------+
CLK_I | 10.866|       |       |       |
--------------+-----------+---------+---------+---------+


Peak Memory Usage: 180 MB
```

Aunque se leen las columnas completas sólo se almacena un valor de cada una, el que represente el peor caso, es decir el valor máximo.

Capítulo 9

Base de datos

En el desarrollo y diseño de la base de datos se utilizará como modelo lógico el modelo relacional. A continuación se comenzará por describir las relaciones:

9.1. Relaciones

Relación	Descripción	Entidades	Atributos
Se implementa	Un circuito es un componente de una implementación.	Circuito, Implementaciones	Id_cir
Encapsula	Una tecnología encapsula al circuito en una implementación.	Tecnologías, Implementaciones	Id_tecno
Produce	Una implementación exitosa produce un report.	Exitosa, Reports.	Id_cir, Id_tecno, Versión

9.2. Entidades atributos

9.2.1. Circuitos

En la práctica esta tabla podría ser prescindible pero se mantendrá por futuras ampliaciones que se pudieran realizar y porque clarifica la base de datos conceptualmente.

Nombre	Descripción	Tipo de dato	PK
Id_cir	Nombre del circuito	Cadena de caracteres	Sí

9.2.2. Tecnologías

Esta tabla recoge las FPGAs con características propias de ésta, independientes de las implementaciones.

Nombre	Descripción	Tipo de dato	PK
Id_tecno	Nombre de la FPGA	Cadena de caracteres	Sí
Precio	Precio de la FPGA	Real no negativo	No
UL_d	Unidades lógicas disponibles	Natural	No
Bits_d	Número de bits de memoria dispoibles	Natural	No
Mult_d	Número de multiplicadores disponibles	Natural	No
Pines_d	Número de pines disponibles	Natural	No

9.2.3. Implementaciones

Contiene las implementaciones circuito-FPGA que se han llevado a cabo.

Nombre	Descripción	Tipo de dato	PK
Id_cir	Nombre del circuito	Cadena de caracteres	Sí
Id_tecno	Nombre de la FPGA	Cadena de caracteres	Sí
Versión	Número de versión de la implementación	Natural	Sí
Fallida	Indica si una implementación ha sido fallida	Boolean	No

9.2.4. Exitosas

Contiene las implementaciones que no han sido fallidas.

Nombre	Descripción	Tipo de dato	PK
Id_cir	Nombre del circuito	Cadena de caracteres	Sí
Id_tecno	Nombre de la FPGA	Cadena de caracteres	Sí
Versión	Número de versión de la implementación	Natural	Sí
Sobredimensionada	Indica si una implementación no ha podido realizarse al no caber el circuito en la FPGA	Boolean	No
UL_u	Unidades lógicas utilizadas	Natural	No
Pines_u	Pines utilizados en esta implementación	Natural	No

9.2.5. Fallidas

Contiene aquellas implementaciones que fallaron a causa de la falta de uno o varios ficheros y sus nombres.

Nombre	Descripción	Tipo de dato	PK
Id_cir	Nombre del circuito	Cadena de caracteres	Sí
Id_tecno	Nombre de la FPGA	Cadena de caracteres	Sí
Versión	Número de versión de la implementación	Natural	Sí
Fichero	Nombre del fichero que no se encontró	Cadena de caracteres	Sí

9.2.6. Reports

Contiene la información extraida de los ficheros reports que fueron generados mediante la aplicación Xilinx ISE Webpack.

Nombre	Descripción	Tipo de dato	PK
Id_cir	Nombre del circuito	Cadena de caracteres	Sí
Id_tecno	Nombre de la FPGA	Cadena de caracteres	Sí
Versión	Número de versión de la implementación	Natural	Sí
Reg2Reg	Tiempo máximo que tarda una señal en ir de un registro a otro	Real no negativo	No
Mult_u	Multiplicadores utilizados	Natural	No
Bits_u	Bits utilizados	Natural	No
Pico_mem	Cantidad máxima de memoria requerida.	Real no negativo	No
Tsu	Tiempo necesitado para	Real no negativo	No
Th	Tiempo necesitado para	Real no negativo	No
T_sint	Tiempo necesitado para completar la fase de síntesis.	Real no negativo	No
T_imp	Tiempo necesitado para completar la fase de implementación.	Real no negativo	No
T_map	Tiempo necesitado para completar la fase de mapping.	Real no negativo	No

9.3. Diagrama ERE

En el diagrama se puede observar de una manera más clara las entidades y sus relaciones.

En un principio sólo se tenían como claves primarias en la tabla "implementaciones" y sus derivadas el identificador del circuito y el de la tecnología. Fue posteriormente cuando se pensó que podían existir varias versiones de una misma implementación, ya fuera porque se hubiera modificado el circuito o porque la estrategia de síntesis fuera distinta. Esta es la razón de añadir un campo adicional llamado versión que a su vez es clave primaria.

La tabla de implementaciones fallidas se había pensado para que una implementación si era fallida tuviera una y sólo una línea. Finalmente se descartó esta idea ya que si sólo fallaba un fichero se tenían dos campos en blancos ocupando el mismo espacio que si estuvieran llenos, y esto no era óptimo. Por ello se añadió el nombre del fichero a la clave primaria, de forma que una implementación pueda aparecer varias veces en la tabla de fallidas siempre y cuando el fichero por el que han fallado sea distinto.

9.4. Diseño lógico

En este caso las relaciones se corresponden exactamente con las entidades, por lo que se va a proceder a su normalización.

El esquema de relaciones es el siguiente:

- Circuitos (id_circuito)
- Tecnologías (id_tecno, UL_d, Bits_d, Mult_d, Pines_d, precio)
- Implementaciones (id_circuito, id_tecno,versión, fallida)
- Exitosas (id_circuito, id_tecno,versión, UL_u, Pines_u, sobredimensionada)
- Fallidas (id_circuito, id_tecno,versión,fichero)
- Reports (id_circuito, id_tecno,versión, Reg2Reg, Tsu, Th, Mult_u, Bits_u, Pico_mem, T_sint, T_map, T_imp)

9.4.1. Primera Forma Normal 1NF

Una relación está en primera forma normal si todos sus atributos son atómicos, es decir, dada una clave primaria sólo hay un valor para cada atributo que se corresponda con ella.

Se puede observar que todos los atributos son atómicos y por tanto la base de datos está en primera forma normal.

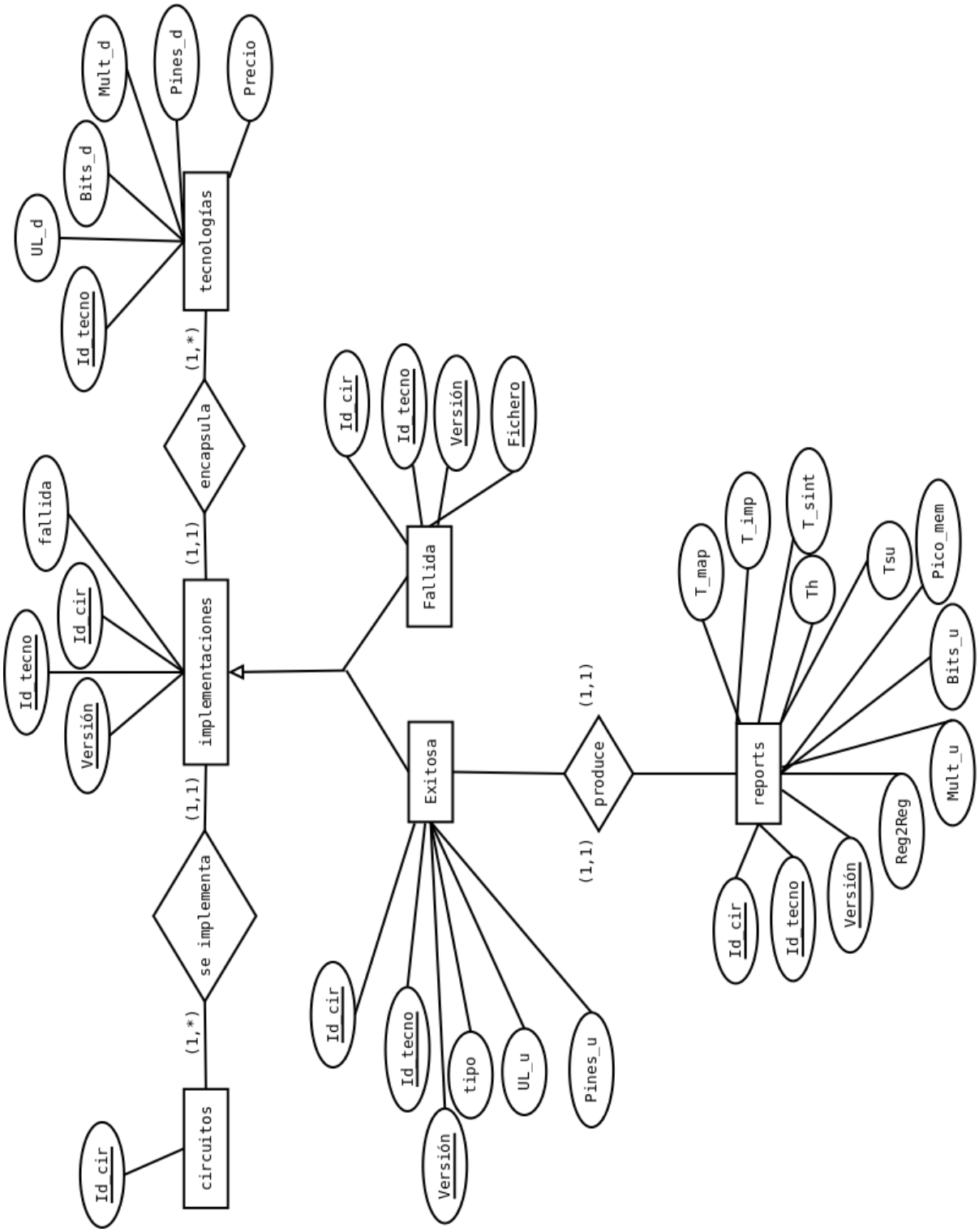

Figura 9.1: Esquema Entidad-Relación-Extendido

9.4.2. Segunda Forma Normal

Una relación está en segunda forma normal si está en primera y además dada una clave candidata y cualquier atributo, éste depende de la clave completa y no sólo de una parte o subconjunto de ella.

Todos los atributos no primarios de las relaciones propuestas dependen de la clave primaria completa y no de un subconjunto de ella. Véase con más detenimiento en aquellas tablas cuyas claves primarias no son simples y tienen atributos no primarios:

- Implementaciones: fallida depende de la clave primaria completa ya que se podrían tener dos versiones de una implementación, siendo una fallida y la otra no.
- Exitosa: Las unidades lógicas utilizadas, los pines utilizados y si la implementación fue sobredimensionada dependen de la clave completa ya que puede haber varias versiones de una misma implementación y tener distintos valores estos atributos.
- Reports: El razonamiento para esta relación es idéntico al que se ha seguido para la anterior.

9.4.3. Tercera Forma Normal

Una relación está en tercera forma normal si está en segunda y todos los atributos no primarios dependen de la clave primaria de manera no transitiva.

Con esta definición se puede afirmar que las relaciones anteriormente expuestas están en tercera forma normal. No obstante analícense detenidamente:

- Tecnologías: Los atributos contemplados dependen exclusivamente de la clave primaria, no se puede conocer el número de bits a partir de los multiplicadores ni nada por el estilo.
- Implementaciones: Sólo tiene un atributo no primario, por lo que es imposible una dependencia transitiva con otro no primario.
- Exitosas: A partir del atributo sobredimensionada podría saberse que UL_u ó Pines_u son mayores que los disponibles, pero en ningún caso podría conocerse el valor concreto. Una implementación es sobredimensionada si UL_u>UL_d o Pines_u>Pines_d. Al no estar los segundos atributos de cada comparación en la tabla exitosas, se puede afirmar que dicha tabla está en tercera forma normal.
- Reports: Para comprender por qué esta tabla está en la tercera forma normal hay que seguir el mismo razonamiento empleado para la tabla Tecnologías.

9.4.4. Forma Normal de Boyce-Codd

Una relación está en la forma normal Boyce-Codd si está en tercera forma normal y sólo existen las dependencias funcionales triviales entre los atributos que conforman la clave primaria.

Compruébese que todas las relaciones están en Boyce-Codd:

- Implementaciones: No existe dependencia entre circuito y tecnología. Tampoco existe dependencia entre circuito y la versión ni entre esta y la tecnología.
- Exitosas: Ídem de la anterior.
- Fallidas: Ídem de la anterior. Además es imposible conocer la ruta del fichero a partir de los otros tres parámetros, en todo caso se podría conocer una dirección relativa, pero ni el nombre concreto ni el resto de la dirección.
- Reports: Ídem de la primera y la segunda.

9.4.5. Cuarta Forma Normal

Una relación está en cuarta forma normal si está en tercera o Boyce-Codd y no existen dependencias multivaluadas no triviales. Esta propiedad también se cumple en todas las relaciones.

Se puede afirmar por tanto que la base de datos está normalizada.

9.5. Implementación

A continuación se expone el script para implementar esta base de datos en un sistema de gestión de base de datos MySQL:

```
DROP TABLE IF EXISTS implementacion;
DROP TABLE IF EXISTS exitosas;
DROP TABLE IF EXISTS circuitos;
DROP TABLE IF EXISTS reports;
DROP TABLE IF EXISTS fallidas;
DROP TABLE IF EXISTS tecnologias;

CREATE TABLE tecnologias (
       id_tecno CHAR(20)
     , UL_d INT
     , bits_d INT
     , mult_d INT
```

```
      , pines_d INT
      , familia CHAR(10)
      , precio FLOAT
      , PRIMARY KEY (id_tecno) #PRIMARY KEY (id_tecno)
);
LOAD DATA LOCAL INFILE '/usr/share/thor/tecno.txt' INTO
   TABLE tecnologias;
CREATE TABLE fallidas (
       #id_imp INT NOT NULL,
        id_circuito CHAR(20) NOT NULL
      , id_tecno CHAR(20) NOT NULL
      , version INT NOT NULL
      , fichero CHAR(90) NOT NULL
      #, f_imp CHAR(90)
      #, f_tiempo CHAR(90)
      #, f_flow CHAR(90)
      #, PRIMARY KEY (id_imp)
       , PRIMARY KEY (id_circuito,id_tecno,version,fichero)
       , CONSTRAINT FK_fallida_1 FOREIGN KEY (id_circuito)
                  REFERENCES circuitos (id_circuito)
       , CONSTRAINT FK_fallida_2 FOREIGN KEY (id_tecno)
                  REFERENCES tecnologias (id_tecno)
);

CREATE TABLE reports (
      #  id_imp INT NOT NULL
        id_circuito CHAR(20) NOT NULL
      , id_tecno CHAR(20) NOT NULL
      , version INT NOT NULL
      , mult_u INT
      , bits_u INT
      , pico_mem FLOAT
      , reg2reg FLOAT
      , Tsu FLOAT
      , Th FLOAT
      , Tco FLOAT
      , T_sint FLOAT
      , T_imp FLOAT
      #, T_map FLOAT
      #, PRIMARY KEY (id_imp)
      , PRIMARY KEY (id_circuito,id_tecno,version)
);

CREATE TABLE circuitos (
        #id_circuito INT NOT NULL AUTO_INCREMENT,
        nombre CHAR(20)
      , PRIMARY KEY (nombre) #PRIMARY KEY (id_circuito)
);

```

```
CREATE TABLE exitosas (
       #id_imp INT NOT NULL
       id_circuito CHAR(20) NOT NULL
     , id_tecno CHAR(20) NOT NULL
     , version INT NOT NULL
     , UL_u INT
     , Pines_u INT
     , sobredimensionada BOOLEAN
     #, PRIMARY KEY (id_imp)
     , PRIMARY KEY (id_circuito,id_tecno,version)
     #, CONSTRAINT FK_exitosas_1 FOREIGN KEY (id_imp)
        REFERENCES reports (id_imp)
     , CONSTRAINT FK_exitosas_1 FOREIGN KEY (id_circuito)
                 REFERENCES circuitos (id_circuito)
     , CONSTRAINT FK_exitosas_2 FOREIGN KEY (id_tecno)
                 REFERENCES tecnologias (id_tecno)
);

CREATE TABLE implementacion (
       version INT NOT NULL AUTO_INCREMENT
     , id_circuito CHAR(20) NOT NULL#INT
     , id_tecno CHAR(20) NOT NULL#INT
     , fallida BOOLEAN
     #, PRIMARY KEY (id_imp)
     , PRIMARY KEY (id_circuito,id_tecno,version)
     , CONSTRAINT FK_implementacion_1 FOREIGN KEY (
        id_circuito)
                 REFERENCES circuitos (id_circuito)
     , CONSTRAINT FK_implementacion_2 FOREIGN KEY (id_tecno)
                 REFERENCES tecnologias (id_tecno)
     , CONSTRAINT FK_implementacion_3 FOREIGN KEY (
        id_circuito,id_tecno,version)
                 REFERENCES fallidas (id_circuito,id_tecno,
                    version)
     , CONSTRAINT FK_implementacion_4 FOREIGN KEY (
        id_circuito,id_tecno,version)
                 REFERENCES exitosas (id_circuito,id_tecno,
                    version)
     #, CONSTRAINT FK_implementacion_3 FOREIGN KEY (id_imp)
     #             REFERENCES DEFAULT_SCHEMA.fallidas (
        id_imp)
     #, CONSTRAINT FK_implementacion_4 FOREIGN KEY (id_imp)
     #             REFERENCES exitosas (id_imp)
);
```

La tabla de tecnologías es inicializada con una serie de FPGAs con sus características y sus precios. En futuras actualizaciones de la aplicación, para añadir nuevas FPGAs bastaría con insertarlas en esta tabla. La aplicación también es capaz de añadir nuevas FPGAs por sí misma,

al analizar los reports. Sin embargo el campo *precio* quedará con el valor NULL, ya que no se puede extraer este atributo de los reports.

Capítulo 10

Pruebas

La realización de pruebas de software, es una de las fases del ciclo del software y permiten la identificación de posibles fallos en la implementación, calidad o usabilidad de la aplicación.

10.1. Plan de pruebas

La arquitectura de Thor obliga a realizar un plan de pruebas algo fuera de lo común. En primer lugar por el entorno de clustering en el que puede trabajar[1] y en segundo por la combinación de programación esctructurada y orientada a objetos que implementa.

Para comprobar el correcto funcionamiento de entorno se desarrolló un plan de pruebas que consistía en:

- **Pruebas de funciones**: Tal como se finalizaba la implementación de una función se probaba que ésta devolvía lo esperado, prestando especial atención a los casos que se darían habitualmente pero sin desatender a los casos extremos.
- **Pruebas de clase**: Tras implementar una clase se probaba el correcto funcionamiento de sus métodos así como de sus constructores.
- **Pruebas de subsistema**: Cuando se completaba un subsistema se comprobaba que funcionaba correctamente. Así ocurrió con los scripts concurrentes, la base de datos o la extracción de datos de los ficheros de texto.
- **Pruebas de sistema**: Cuando se tenían dos o más subsistemas relacionados se probaba que sus interacciones se producían correctamente. De esta manera se probaron los sub-

[1] Trabajar en un entorno de clustering es sólo una posibilidad. THOr está preparado para ejecutarse en un único equipo.

sistemas de base de datos junto con el de extracción de información de los ficheros o la interfaz gráfica con el resto de subsistemas.

Ha habido tres fases de pruebas diferenciadas durante el desarrollo de *Thor*:

- *Durante la implementación*: Como se ha comentado anteriormente, se probaban determinadas unidades tal y como se finalizaba su implementación. La razón de esto es no *programar sobre mojado*, es decir, no programar suponiendo que algo está bien cuando puede no estarlo.
- *Tras la fase de implementación*: Se comprobó que todas las unidades interactuaban bien entre ellas y se manejaban correctamente los errores.
- *Los instaladores*: Al tratarse de un entorno multiplataforma y aunque, teóricamente, todos los programas auxiliares y librerías eran compatibles con ambos sistemas operativos, hubo que probar la integración de *Thor* con ellos y sus peculiaridades.

10.2. Especificación del diseño de pruebas

- *Durante la implementación*: las pruebas se desarrollaron con datos que se adaptaban a los requisitos del módulo o subsistema que se estuviera implementando en ese instante. Estas pruebas consistían en comprobar el tratamiento de la información por parte del módulo, así como la recuperación y almacenamiento de dicha imformación en la base de datos. También se comprobaba que los distintos nodos del clúster hicieran trabajos distintos y no se entorpecieran entre ellos. Con estas pruebas se evitába la propagación de errores básicos a etapas de un nivel superior.
- *Tras la fase de implementación*: Finalizada la codificación era necesario comprobar el correcto funcionamiento del entorno y todas sus funcionalidades.
- *Los instaladores*: Una vez comprobado que el sistema no tenía fallos en una distribución GNU/Linux se crearon instaladores para esta y se procedió a la exportación a Windows. En ambos sistemas hubo problemas con los ficheros que eran necesarios copiar y los permisos necesarios para ello. No obstante, sorprendentemente, fue más sencillo realizar el instalador para la distribución GNU/Linux.

Las pruebas comenzaron por el mismo módulo por el que se empezó la codificación, esto es la parte de clustering. Fueron necesarios varios días hasta conseguir que las diferentes versiones de los scripts funcionaran sin entorpecerse y sin repetir los trabajos. En aquellos momentos no se tenían apenas conocimientos de programación distribuida por lo que la labor fue aún más dura si cabe.

Tras haber probado varias posiblidades se escogió aquella que tardaba menos tiempo en ejecutar los trabajos y era más robusta en cuanto a caida de nodos.

Una vez terminada las pruebas de clustering se procedió a la extracción de información de los ficheros reports y su inserción y consulta de la base de datos. Observando los tiempos que tardaba la aplicación predecesora se llegó a pensar en introducir la potencia del cluster para esta labor. Sin embargo las pruebas en seguida arrojaron resultados excelentes con tiempos no superiores al minuto y medio, por lo que se decidió que no compensaba introducir programación distribuida en este punto.

Finalmente la interfaz gráfica fue integrada y probada con el resto de módulos. Era necesario comprobar el manejo correcto de todos los eventos y que no diera sensación de bloqueo al usuario.

10.3. Especificación de los casos de prueba

El entorno fue probado por dos tipos distintos de usuario:

- *Personas con bajo nivel de conocimientos informáticos*: Estas personas tienen conocimientos informáticos básicos realizando sus tareas habituales con el PC con aplicaciones ofimáticas y con entornos de programación de circuitos digitales. En este grupo tienen cabida algunas de las personas del grupo de investigación con el que se desarrolló el proyecto. No surgieron fallos importantes sino que más bien sugirieron pequeños cambios estéticos orientados a facilitar su trabajo día a día y mejoras del manual de usuario.
- *Personas con nivel avanzado de conocimientos informáticos*: Este grupo es más reducido que el anterior por la dificultad de encontrar individuos que conocieran tanto la parte electrónica como informática. Ellos tuvieron acceso tanto a todo el código así como a la documentación. Gracias a su intervención se detectaron pequeños errores relacionados con el manejo de errores.

10.4. Especificación de los procedimientos de prueba

Como se ha dicho anteriormente, THOr es multiplataforma por lo que debió ser probado en los dos sistemas operativos con los que debía ser compatibles: Windows y Linux. Concretamente se ha probado en la versión Vista 32 bits de Microsoft Windows y en la distribución Ubuntu de GNU/Linux en sus versiones 10.04 y 11.04 de 32 y 64 bits.

Los principales problemas se dieron al exportar la aplicación de Ubuntu, donde fue programada, a Windows. Las especificaciones de las librerías empleadas para implementar la interfaz gráfica y de MySQL exigieron el retoque de varias líneas de código hasta conseguir ejecutar el entorno. También fue necesario cambiar las rutas de llamada a aplicaciones externas y a ficheros, ya que la jerarquía de directorios en Windows es diferente a la de Ubuntu.

10.5. Pruebas de tiempo

A parte de las pruebas realizadas para la detección de errores también se llevaron a cabo otras para escoger el mejor algoritmos e implementación de los scripts concurrentes. A continuación se exponen en una tabla las diferentes configuraciones del cluster junto con la implementación escogida y los tiempos obtenidos:

Algoritmo	**Lenguaje**	**Tiempo**	**Carga del sistema**
Polling	Bash	34 horas 43 minutos	144 %
Preasignación de tareas	Python	27 horas	90 %
Presencia de ficheros	Python	21 horas 45 minutos	90 %-110 %
Objetos remotos	Python	21 horas	120 %

Tabla 10.1: Tiempos de ejecución de los scripts

En la figura 10.1 pueden observarse el estado de los nodos durante la ejecución de las pruebas. En este caso el ordenador llamado cluster-desktop actuaba como cliente y servidor, mientras que el resto eran clientes.

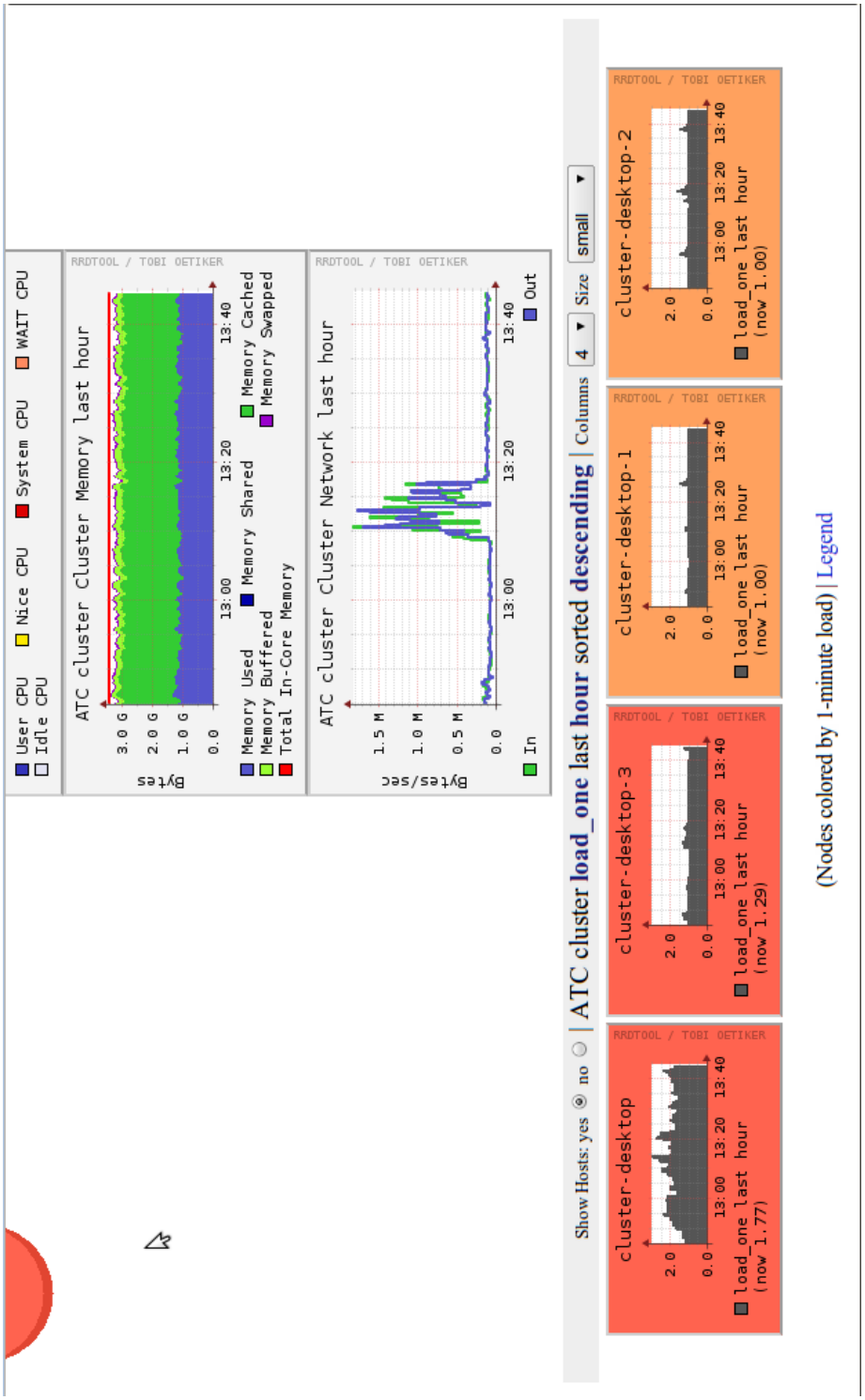

Figura 10.1: Estado del cluster durante la ejecución de los scripts cliente-servidor

Capítulo 11

Conclusiones

En este capítulo se procede a ofrecer una valoración global del proyecto del realizador del mismo así como introducir las posibles ampliaciones futuras.

11.1. Valoración personal

El desarrollo de este entorno de trabajo ha sido el primer proyecto serio que he llevado a cabo. Con él he podido ver como interactúan las distintas partes de la informática que he ido aprendiendo a lo largo de estos años. Además he conseguido terminarlo en el tiempo previsto aunque esto supusiera desarrollarlo al mismo tiempo que tenía lugar el curso académico 2010/2011 en el cual he tenido que cursar las asignaturas previstas en el plan de estudios además de obtener el B1 de inglés e implicarme en actividades del centro como el Proyecto Compañero. Cuando me matriculé en esta carrera me propuse el reto de sacarla en los tres años previstos, aun conociendo que la media estaba en 7 años, y para ello la única posibilidad era hacerlo al mismo tiempo que tercero.

Pienso que es un proyecto bastante completo ya que trato varias partes de la informática, como pueden ser la programación distribuida, las bases de datos, el tratamiento y análisis de ficheros, las interfaces gráficas, y sin dejar de lado al hardware ya que el proyecto está destinado al análisis y evaluación de la adaptación de circuitos digitales en dispositivos FPGAs.

La programación distribuida ha sido, sin lugar a dudas, la parte más dura del proyecto. Es muy difícil encontrar y corregir los errores en este tipo de programación. La asignatura de programación concurrente y distribuida me ha aclarado ideas y me ha mostrado herramientas que me habrían ahorrado mucho tiempo. No obstante, estoy satisfecho con el trabajo realizado e incluso me produce satisfación poder haberlo completado por mí mismo, buscando mis propias soluciones e implementando mis propias ideas, sin ninguna base.

La base de datos sí fue diseñada tras la asignatura que lleva el mismo nombre. Sin embargo en

esta asignatura no se desarrolla ningún programa que las maneje e interactúe con ellas por lo que esto también fue novedoso para mí. Al igual que con la programación distribuida me siento satisfecho de poder haber llevado a cabo esto por mí mismo.

Las interfaces gráficas tampoco se imparten en la carrera, o al menos en las asignaturas que he cursado. Es bastante curioso que cuando entras en la carrera, tu concepto de programa es una interfaz con sus botones, no se piensa en lo que hay detrás, que es lo más importante. Me hizo bastante ilusión ver al entorno con la interfaz gráfica por ser la primera que implemento. Es algo así como darle vida al programa.

También ha sido mi primera experiencia con Python. Me entusiasmaba la idea de aprender un lenguaje por mí mismo, sin ayuda de nadie y creo que más o menos lo he conseguido. Con respecto a los lenguajes más utilizados a lo largo de la carrera (C/C++) resulta mucho más sencillo de utilizar, aunque la corrección de errores es más engorrosa ya que no te percatas del error hasta que el flujo del programa pasa por el punto concreto. Cosas de la interpretación.

Mi idea inicial era que el entorno fuera software libre y construido con herramientas libres. Este objetivo personal también se ha cumplido no habiendo utilizado ninguna aplicación privativa para su desarrollo salvo *Xilinx ISE Webpack*, la cual no había más remedio que utilizar al tratarse de un requisito del sistema.

Algunas herramientas que se han utilizado para desarrollar este proyecto son:

- Geany: Editor de texto para Python, C, C++, LaTeX, etc.
- Gedit: Editor de texto utilizado para Bash y MySQL.
- Dia: Herramienta para dibujar diagramas de flujo, diagramas UML, etc.
- GIMP: Conocido programa para la edición de imágenes. Fue utilizado para retocar algunas de las fotos que aparecen en la memoria.
- Planner: Herramienta para crear y modificar diagramas de Gantt.

El lenguaje LaTeXme ha facilitado la labor de realizar esta memoria. No imagino cuan duro hubiera sido escribirla en un procesador de texto habitual. Este lenguaje lo he aprendido gracias a un taller que tuvo lugar el año pasado en la Semana de la Ciencia, aunque LaTeXes demasiado extenso para aprenderlo por completo en un taller y siempre surgen pequeños problemas que puedes solventar por tí mismo a traves de la documentación existente.

Una gran parte del proyecto se desarrolló antes de que diera Ingeniería del Software por lo que se tuvieron que hacer algunas modificaciones para que se cumplieran algunos requisitos de esta y las metodologías de desarrollo. No obstante creo que es imposible realizar un proyecto siguiendo la ingeniería del software al pie de la letra, entre otros motivos porque al comienzo de éste no se pueden especificar claramente los requisitos o todas las funcionalidades. De hecho hay funcionalidades que en un principio no estaban planteadas y que se añadieron durante el desarrollo de la aplicación.

Por último, el desarrollo del proyecto también me ha ayudado a escuchar opiniones de terceros sobre mi trabajo y aceptar consejos de profesores y compañeros. Uno no puede cerrarse en banda y creer que lo suyo siempre es lo mejor, entre otras cosas porque entonces llegará un momento en el que nadie querrá darte consejos.

11.2. Valoración técnica

Tras dar en la sección anterior un punto de vista personal paso a ser lo más objetivo posible enumerando una lista de conclusiones:

- El documento de especificación de requisitos que se elaboró en la fase de análisis de este proyecto se ha cumplido estrictamente.
- La reducción del tiempo de ejecución de los ciclos de diseño es evidente, habiéndo reducido los tiempos de los scripts anteriores en más de lo esperado. Una simple regla de tres muestra que si con un ordenador se tarda 89 horas y 10 minutos con 4 debería tardarse 22 horas y 17 minutos. Evidentemente se están obviando las características de los ordenadores y el retardo que supone la comunicación a través de la red. Sin embargo el mejor tiempo conseguido hasta el momento es de 21 horas y 45 minutos, lo que indica que, además de haber distribuido correctamente los trabajos, se ha mejorado la implementación haciéndola más eficiente.
- La reducción del tiempo de análisis de ficheros y su inserción en la base de datos también es palpable, pasando de 30 minutos a uno. Es por tanto que la elección del nuevo sistema de gestión de base de datos y la reorganización de las tablas ha sido un éxito.
- Observando los resultados obtenidos con PCs sin grandes prestaciones se hace patente que las aulas de informática y los laboratorios podrían ser utilizados sin demasiado coste como clusters por los grupos de investigación cuando las tareas a realizar fueran demasiado grandes como para realizarlas en un solo equipo. Las aulas están vacías una gran parte del tiempo y el coste eléctrico no se elevaría en exceso. De esta manera no tendrían que esperar turno para utilizar el cluster de la Universidad, ya que tendrían uno disponible en el mismo centro.

11.3. Ampliaciones futuras

Este proyecto tiene varias líneas por donde puede ser ampliado. Mal proyecto sería si se quedara estancado. Pasamos a enumerarlas a continuación:

1. Sería conveniente implementar un script que automatizará el inicio del trabajo de los nodos automáticamente, de forma que el usuario no tenga que ir equipo por equipo incorporándolo a la tarea manejada por el servidor.

2. Habría que estudiar el comportamiento en cuanto a tiempo y recursos de añadir concurrencia al nivel de hilo. Se podría conseguir que cada núcleo de los procesadores ejecutara un hilo que consistiría en la ejecución de un trabajo. Seguramente *THOr* no sería el problema, sino la memoria que consumen las distintas herramientas de *Xilinx ISE Webpack*.

3. El entorno sólo trabaja con FPGAs de la marca Xilinx. Sería conveniente añadir funciones para el análisis de los reports de las FPGAs de la marca Altera, la segunda en cuanto a cuota de mercado con un 35 %. Si además funcionara con FPGAs de la marca Actel y Lattice cubriría a los cuatro fabricantes más importantes de FPGAs.

4. En un futuro podría servirse la aplicación via Internet, de forma que el usuario utilizara un cluster remoto para completar los ciclos de desarrollo de sus circuitos digitales. El único inconveniente es que el usuario debe estar dispuesto a enviar sus diseños a un agente externo, confiando en que éste no los guardará ni utilizará para otro fin que el que le ha sido autorizado.

Apéndice A

Manual de usuario e instalación

A.1. Instrucciones para sistemas GNU/Linux

A.1.1. Paquetes DEB

Se suministran dos paquetes para instalar THOr en sistemas basados en Debian (Ubuntu, Guadalinex, etcétera). Uno para 64 bits y otro para 32. Escoja la versión que corresponda a su sistema operativo y descargue el fichero en algún directorio local.

Una vez descargado haga doble click en él y pulse el botón “Instalar el paquete” que se encuentra en la esquina superior derecha. Automáticamente se satisfarán las dependencias en caso de que el instalador las encuentre en su repositorio. En caso de que no las encuentre deberá instalarlas usted mismo para ejecutar este instalador posteriormente a haberlas satisfecho.

Para configurar el sistema de gestión de base de datos vaya a la sección A.3

A.1.2. Código fuente

Puede satisfacer manualmente las dependencias especificadas en la sección **??** y ejecutar el script llamado "version1.py" mediante un intérpetre de python. En Ubuntu, por ejemplo, se haría así:

```
python version1.py
```

De igual manera debe haber configurado MySQL según las instrucciones que se indican en la sección A.3.

A.2. Instrucciones para sistemas Windows

Existe un instalador de la aplicación que ha sido probado en Windows Vista 32 bits pero que, según las garantías de retrocompatibilidad de Microsoft, debe ser válido para Windows 7 también.

Para instalar la aplicación sólo debe clicar dos veces en el instalador y seguir las instrucciones de éste. A diferencia de otros instaladores, éste no crea ningún acceso directo. Así deja al usuario la libertad de hacerlo manualmente, si así lo desea, del ejecutable que crea el instalador. Este ejecutable se encontrará en el directorio de instalación que el usuario haya escogido y se denominará "version1.exe".

En Windows sólo tendrá que satisfacer la dependencia de MySQL. Este software es libre y gratuito y puede descargarlo desde la página oficial. Deberá configurarlo de igual manera que para el resto de sistemas operativos, siguiendo las instrucciones de la sección A.3 salvo por una diferencia. El usuario que utiliza Windows para ejecutar las aplicaciones no es el mismo que aparece en la pantalla de inicio y que usted selecciona para entrar en el sistema. Windows utiliza un usuario llamado ODCB y por tanto, a éste será al que usted deba poner a la hora de dar privilegios en MySQL como se indica en la sección A.3.

A.2.1. Instalación MySQL en Windows

A diferencia de la instalación en sistemas GNULinux, la instalación de *MySQL* en Windows requiere la toma de varias decisiones. A continuación explicaremos las que se deben tomar si únicamente la va a utilizar para THOr.

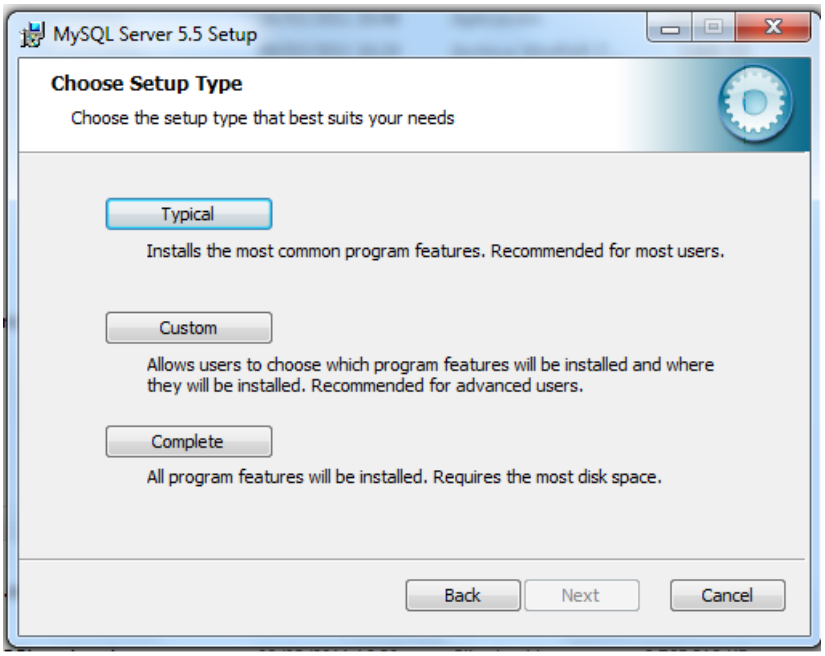

Figura A.1: Tipo instalación MySQL

En primer lugar deberá escoger la instalación típica que instalará los módulos comunes y algunos de los más habituales. Véase figura A.1

A continuación escoja la configuración detallada para que MySQL se adapte lo mejor posible a su equipo. Véase figura A.2.

Escoja ahora una máquina de desarrollador ya que no se trata de una máquina dedicada a la base de datos y en principio sólo THOr utilizará el sistema. Véase figura A.3.

Elija el tipo de base de datos multifuncional, opción que está orientada a bases de datos de propósito general. Véase figura A.4.

El resto de opciones se dejan por defecto salvo las correspondientes a la figura A.5 y a la A.6 donde deberá escoger la codificación UTF8 para una máxima compatibilidad con todos los idiomas e incluiremos el directorio de *MySQL* en el PATH de Windows para poder ejecutarlo desde la terminal.

A.3. Configuración de MySQL

Entre las dependencias se encuentra el sistema de gestión de base de datos MySQL. Para que la aplicación funcione correctamente, el mismo usuario que ejecute la aplicación deberá tener permisos de creación y modificación de tablas y bases de datos en MySQL. Para ello, entre como root en MySQL desde la terminal mediante la siguiente sentencia:

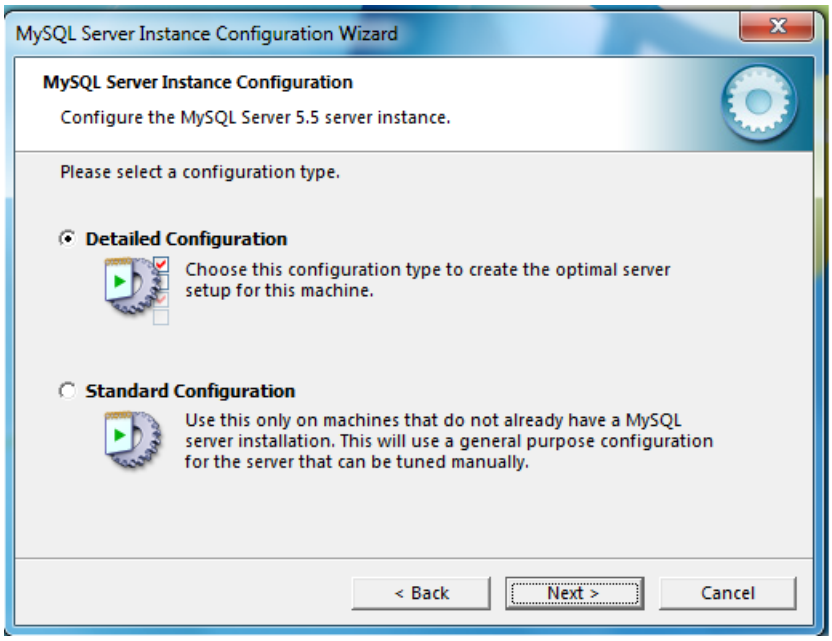

Figura A.2: Tipo configuración MySQL

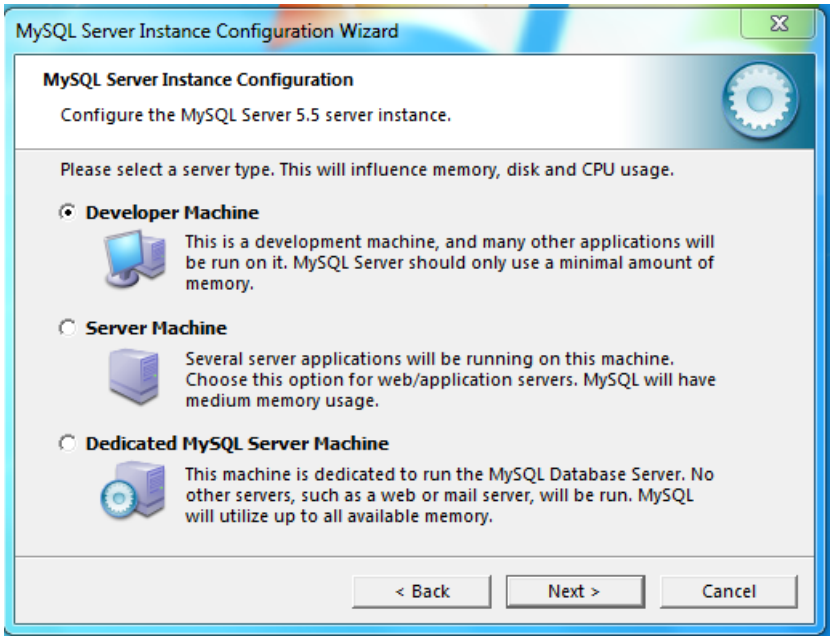

Figura A.3: Tipo de máquina

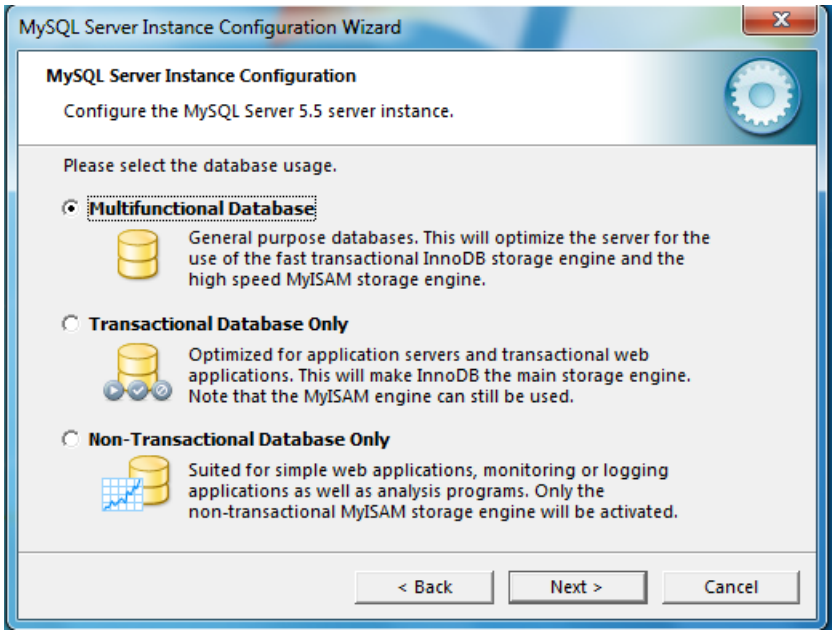

Figura A.4: Tipo de máquina

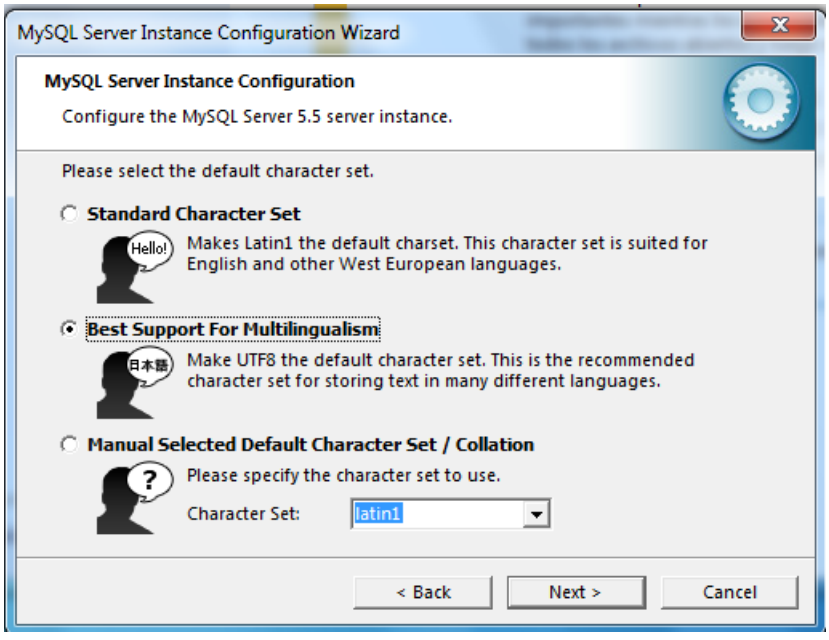

Figura A.5: Codificación

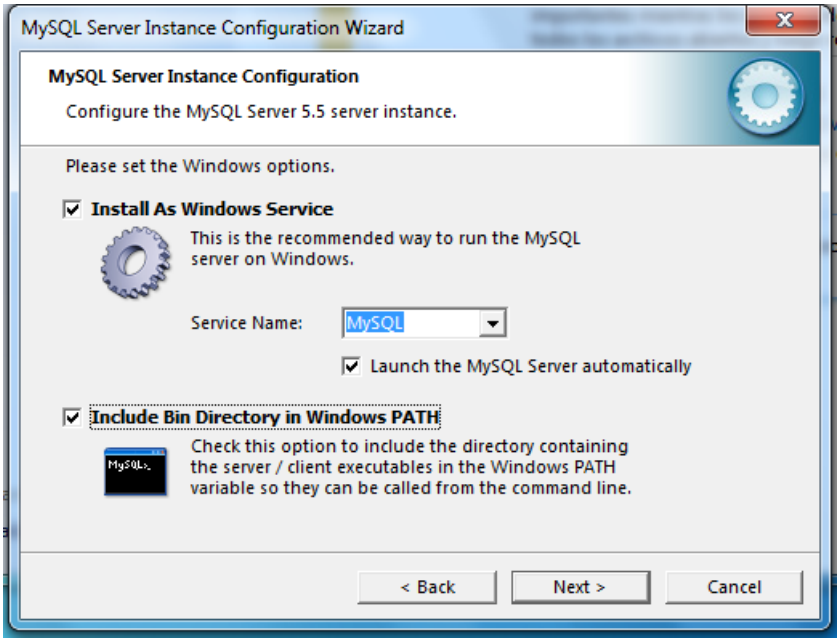

Figura A.6: Configuración Windows PATH

```
mysql -u root -p
```

Se le solicitará la contraseña que introdujo para el usuario root cuando se instaló MySQL.

A continuación ejecute la siguiente instrucción:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'tuusuario'@'localhost
    '
    ->     IDENTIFIED BY 'tupassword' WITH GRANT OPTION;
mysql> quit
```

Para comprobar que todo ha salido correctamente ejecute esta instrucción desde la terminal e introduzca la contraseña que introdujo en la instrucción anterior:

```
mysql -p
```

A.4. Uso de THOr

El uso de THOr es muy intuitivo gracias a la cómoda interfaz que ofrece al usuario. No obstante existen ciertos aspectos que deben ser comentados, sobre todo en cuanto a la jerarquía de ficheros se refiere.

A.4.1. Generar reports

Cuando se utiliza THOr para generar reports de circuitos que se implementan en alguna FPGA la jerarquía de directorios y ficheros no puede ser cualquiera.

Supóngase un directorio, o carpeta, RAÍZ que está vacío. En el interior de este directorio se crearán tantas carpetas como circuitos se quieran analizar, las cuales deberán tener el mismo nombre que el fichero .vhd (sin la extensión) que contiene el circuito a implementar. Además habrá un fichero llamado balanced.txt que contendrá en su interior la estrategia de síntesis que se desea seguir. El nombre de este fichero no implica que la estrategia deba ser balanceada ya que en su interior el fichero puede tener las órdenes que el usuario desee. Por último, dentro de la carpeta de cada circuito estarán los fichero .vhd correspondientes a dicho circuito y un fichero .prj, que contiene la configuración de las distintas herramientas, con el mismo nombre que el primero.

Una vez esté lista la jerarquía de carpetas, el usuario puede abrir la aplicación y seleccionar "Nuevas implementaciones". En ese momento se abrirá un diálogo donde deberá seleccionar el directorio RAÍZ que creó anteriormente. Tras este diálogo se abrirá otro donde podrá seleccionar donde quiere que se guarden los reports y oto más donde seleccionará en qué FPGAs desea implementar el conjunto de circuitos.

A.4.2. Analizar reports

La jerarquía de carpetas que genera la acción anterior es la correcta para analizar los reports. Supóngase de nuevo una carpeta RAÍZ. Dentro de esta habrá una carpeta por cada circuito implementado. Dentro de la carpeta correspondiente a cada circuito habrá una carpeta por cada FPGA con la que se haya implementado dicho circuito. Por último, dentro de estas carpetas estarán los reports generados por el programa Xilinx ISE webpack. Concretamente, la aplicación utilizará los .xst, .par, .srp, _xst.log, _map.map y .twr.

Puede verse una captura del diálogo de elección de directorio en la figura A.7

A.4.3. Filtrar resultados

Mediante esta funcionalidad se pueden reducir los resultados que se visualizan en las tablas de la interfaz a aquellos que cumplan una serie de condiciones. Para ello en el menú se pulsa en Filtrar y se escoge qué tabla se desea filtrar.

Una vez escogida la tabla aparecerá un diálogo donde podrá escoger qué condiciones desea exigir, pudiendo requerir hasta cuatro condiciones. Puede ver este diálogo en la figura A.8

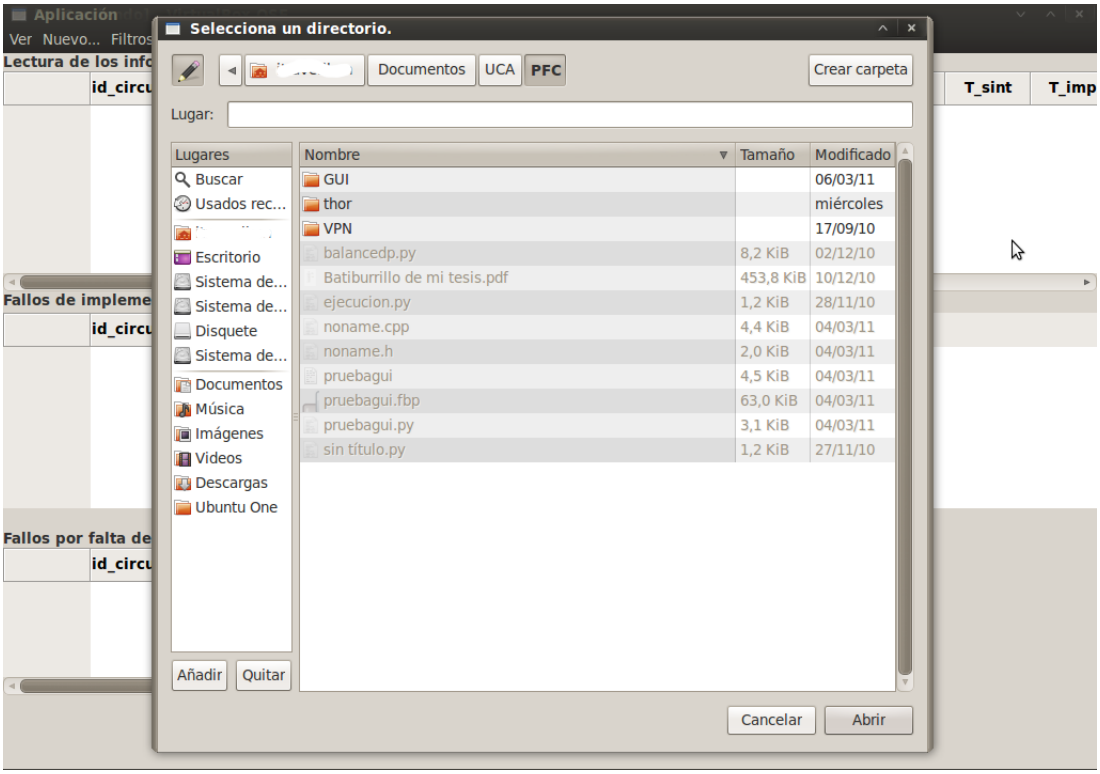

Figura A.7: Diálogo para elección de directorio

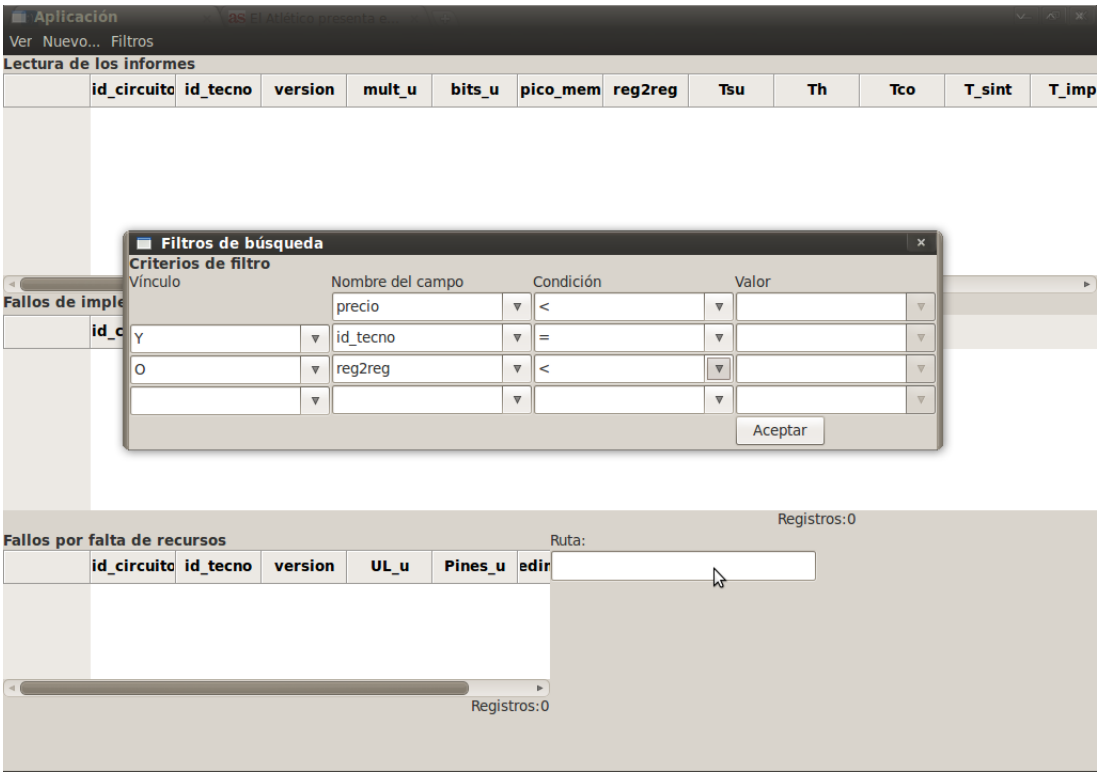

Figura A.8: Diálogo para el filtrado

Apéndice B

Licencia

GNU Free Documentation License

Version 1.3, 3 November 2008

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "freeïn the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of çopyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

B.1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ƶou". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version.ºf the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Sectionïs a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The Ïnvariant Sections.ªre certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The Çover Texts.ªre certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparentçopy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparentïs called .ºpaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or

XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page"means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page"means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher"means any person or entity that distributes copies of the Document to the public.

A section .Entitled XYZ"means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as .Acknowledgements", "Dedications", .Endorsements", or "History".) To "Preserve the Title.of such a section when you modify the Document means that it remains a section .Entitled XYZ.according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

B.2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

B.3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts:

Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

B.4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

1. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
2. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
3. State on the Title page the name of the publisher of the Modified Version, as the publisher.

4. Preserve all the copyright notices of the Document.
5. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
6. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
7. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
8. Include an unaltered copy of this License.
9. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "Historyïn the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
10. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History"section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
11. For any section Entitled .Acknowledgements.or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
12. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
13. Delete any section Entitled .Endorsements". Such a section may not be included in the Modified Version.
14. Do not retitle any existing section to be Entitled .Endorsements.or to conflict in title with any Invariant Section.
15. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled .Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

B.5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "Historyïn the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled .[A]cknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled .[En]dorsements".

B.6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

B.7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an .aggregateïf the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

B.8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled .Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

B.9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally

terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

B.10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License .ºr any later version.ªpplies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

B.11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration"(or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Bibliografía

[1] Raúl González Duque. Python para todos.

[2] Francisco Callejo Giménez. Inmersión en Python. 2001

[3] Documentación MPI4Py http://mpi4py.scipy.org/docs/apiref/index.html

[4] Documentación de Pyro http://packages.python.org/Pyro/

[5] Ramesh Natarajan. Linux 101 Hacks. 2009.

[6] Pétur Ingi Egilsson. John the Ripper on a Ubuntu 10.04 MPI Cluster. 2010.

[7] Ben-Ari, M. Concurrent and Distributed Programming. Prentice Hall, 1990.

[8] Tomeu, A. Apuntes de Programación Concurrente. 2003.