

# Una plataforma reconfigurable para la enseñanza de sistemas lógicos

Héctor Fabio RESTREPO<sup>1</sup>    Francisco J. GÓMEZ<sup>2</sup>    Christof TEUSCHER<sup>1</sup>  
Jacques-Olivier HAENNI<sup>1</sup>    Eduardo SÁNCHEZ<sup>1</sup>

<sup>1</sup>Laboratoire de Systèmes Logiques, Ecole Polytechnique Fédérale de Lausanne  
CH – 1015 Lausanne, Suiza  
E-mail: {name.surname}@di.epfl.ch

<sup>2</sup> Escuela Técnica Superior de Informática, Universidad Autónoma de Madrid  
E – 28049 Madrid, Spain  
E-mail: francisco.gomez@ii.uam.es

## Resumen

*En este artículo se describen las características de la tarjeta Labomat 3 y un conjunto de programas de aplicación que muestran su utilidad para la enseñanza en tres dominios de interés: diseño de sistemas lógicos, arquitectura de ordenadores y codiseño.*

*Labomat 3 es una plataforma reconfigurable desarrollada en el Laboratorio de Sistemas Lógicos de la EPFL. La tarjeta consta básicamente de un microprocesador conectado a dos FPGAs de tamaño medio. Su comunicación con el exterior está garantizada mediante una conexión Ethernet 10Base-T y una conexión serie RS232. Un sistema operativo tiempo real, de dominio público, proporciona la base sobre el que se puede ejecutar una máquina virtual Java, un servidor TCP-IP que permite el acceso remoto mediante sockets, o bien directamente programas de usuario compilados para este microprocesador. Se dispone también de aplicaciones para PC's, que permiten el prototipaje rápido de sistemas lógicos. Estas herramientas hacen uso del acceso remoto a la tarjeta mediante un protocolo de paso de mensajes sobre TCP/IP.*

*Todas estas características hacen de Labomat 3 una plataforma completa, única en el ámbito de la enseñanza. Además, sus grandes posibilidades de comunicación, abren un prometedor campo de investigación en sistemas reconfigurables paralelos .*

## 1 Introducción

La reconfigurabilidad y la densidad creciente de los circuitos FPGA (Field Programmable Gate Arrays) [7] han logrado una verdadera revolución en el campo de circuitos digitales. Inicialmente estaban exclusivamente reservados a tareas de prototipaje, pero gracias a su evolución se han visto involucrados rápidamente en un gran número de aplicaciones.

Naturalmente, la enseñanza ha aprovechado todas estas posibilidades y como consecuencia de ello se disponen de numerosas plataformas con este fin. En algunos casos son los propios fabricantes de circuitos FPGA quien las proporcionan. [1] [5] Sin embargo, estos sistemas de desarrollo no consideran uno de los campos más prometedores: el codiseño, donde se plantea la posibilidad de elegir que partes de una aplicación se van a realizar en hardware y cuáles en software [2].

En la enseñanza universitaria tradicional existe una clara distinción entre los campos relativos al diseño de hardware por un lado, y todo lo relacionado con aplicaciones y desarrollo de software por otro. Los primeros son objeto de estudio en ingeniería electrónica, mientras que los segundos se enseñan en facultades de informática. Con el codiseño esta frontera desaparece, lo cual implica también un cambio fundamental en la formación que debe tener un ingeniero informático y por tanto, un nuevo planteamiento de qué conocimientos deben formar parte de su curriculum universitario.

La tarjeta presentada en este artículo, Labomat 3, ha sido concebida para ser utilizada por estudiantes en todo tipo de cursos de diseño de hardware, desde diseño elemental de circuitos digitales al codiseño, pasando por arquitectura de ordenadores. Para su utilización en estos campos, Labomat 3 presenta unas características únicas:

1. Un software específico, llamado RVS6200, que ha sido desarrollado para la enseñanza de sistemas lógicos. Su característica principal es el aprovechamiento de la reconfiguración dinámica parcial de la familia de FPGA XC6200. La idea es realizar simultáneamente el proceso de emplazamiento y conexionado, a medida que se diseña el esquema del circuito. Cuando el usuario realiza una conexión o coloca un componente en el editor de esquemas, se determina de manera biunívoca

la configuración de la celda correspondiente en la FPGA. De esta manera, el esquema diseñado por el alumno es una imagen del circuito configurado en la FPGA.

La verificación de la funcionalidad del diseño en la FPGA puede comprobarse en tiempo real, sin necesidad de un simulador, ya que es posible leer en la FPGA el estado de la salida de todos los componentes presentes en el esquema, y visualizarlos en la pantalla. RVS6200 combina así las ventajas de simulación y emulación.

2. Para un curso de arquitectura de ordenadores, se dispone de una FPGA XC4013E relativamente compleja. En ella, por ejemplo, los estudiantes pueden realizar un procesador completo, con un conjunto de instrucciones simple, pero implementando cinco etapas de segmentación (pipeline) con detección de riesgos y corrección de dependencias.
3. En un curso de codiseño se puede emplear el microprocesador de la familia 68000 de motorola y las dos FPGAs disponibles en la tarjeta. Además se dispone de un sistema operativo (RTEMS) y un conjunto de herramientas, como por ejemplo una máquina Java y un servidor de comunicaciones para TCP/IP.
4. Es posible la configuración remota de la tarjeta por gracias a su conexión Ethernet, utilizando el protocolo TCP/IP.

La combinación de todas estas características hacen de Labomat 3 una herramienta única de enseñanza. Además, sus grandes capacidades de comunicación abren un nuevo campo de investigación muy prometedor: los sistemas reconfigurables paralelos.

El resto de este artículo tiene la siguiente organización: En el primer apartado se describe el hardware de la Labomat 3, el sistema operativo y las posibilidades de comunicación. En la segunda parte se presentan tres campos de aplicación: diseño de sistemas lógicos, arquitectura de ordenadores y codiseño. El artículo termina presentando algunas consideraciones sobre los trabajos futuros.

## Parte I

### Descripción de Labomat 3

La tarjeta Labomat 3 es un versión simplificada y modificada de la tarjeta RENCO (Reconfigurable Network Computer) [6, 3] desarrollada igualmente en nuestro laboratorio. El objetivo principal es proporcionar a los estudiantes una plataforma de gran potencia, pero a la vez, fácil de comprender y de utilizar.

## 2 Descripción Hardware.

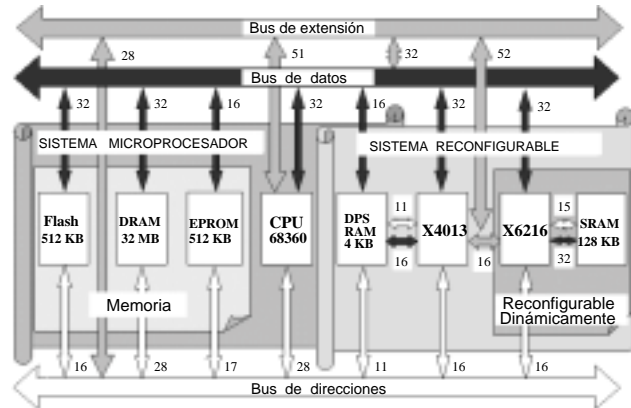


Figura 1: Arquitectura de Labomat 3

La tarjeta consta de dos partes. (Fig. 1):

1. **La parte del procesador** está construida en torno a un procesador Motorola MC68EN360 [8], conectado a 512 KB EPROM (para el arranque de la tarjeta) y a un máximo de 32 MB de DRAM (SIMM). Dispone de una memoria flash adicional de 512 KB para guardar configuraciones de las FPGA o cualquier otra información no volátil. Una memoria de doble acceso de 4 KB permite compartir los datos entre el procesador y la parte reconfigurable. La comunicación con el mundo exterior se realiza mediante interfase RS232 y Ethernet 10Base-T.
- El procesador MC68360 se ha elegido por sus capacidades de comunicación, así como la disposición, para este procesador, de un sistema operativo de dominio público.
2. **La parte reconfigurable** está construida alrededor de dos circuitos FPGA, el XC4013E y el XC6216 de Xilinx[11]. El XC6216 esta conectado a 128 KB de SRAM que no son accesibles directamente desde el procesador. El XC4013E está conectado a la memoria de doble acceso, que a su vez está conectada al procesador.

El procesador puede acceder como periféricos a los circuitos FPGAs por el bus de direcciones y por el bus de datos. Un bus adicional de 80 líneas conecta ambas FPGA. También es posible la conexión de la tarjeta con dispositivos externos, como puede verse en la figura 2, mediante múltiples conectores de 10 contactos que son un estándar en nuestro laboratorio. A través de estos conectores se dispone de acceso a un subconjunto de 52 señales de las 80 que conectan las FPGA. Un bus de extensión incluye el bus de datos, de

direcciones y de control. Otros interfaces disponibles son BDM para la programación del controlador de Bus y JTAG que permite la depuración a bajo nivel del hardware. El control del bus y de las interrupciones se realiza mediante un circuito programable (MAX7128). Con ello se ha conseguido ocultar las complicadas especificaciones de tiempos que hay que garantizar para el acceso a los buses y la configuración de las FPGAs. El controlador de Bus también dispone de registros para control y estado de la tarjeta, como por ejemplo control de interrupciones y selección de reloj.

Cada FPGA recibe dos señales de reloj, y cada una de ellas puede provenir de diferentes fuentes: un reloj programable que proporciona un amplio rango de frecuencias, o bien, una señal de reloj que puede ser generada por el mismo procesador. También se dispone de un pulsador que permite la ejecución paso a paso. La frecuencia base del sistema es de 25 MHz. El código de arranque se ejecuta desde la EPROM. Este código carga, mediante la conexión de red, el sistema operativo completo y las aplicaciones (ver sección 4); así no es necesaria una reprogramación de la EPROM si se modifica el sistema operativo o las aplicaciones.



Figura 2: La plataforma Labomat 3

La tarjeta Labomat 3 está realizada en una placa de circuito impreso con 6 capas (Fig. 2). El XC6216 y el microprocesador están insertados en la cara posterior de la tarjeta, no mostrada en la figura.

### 3 Descripción del Software

Para conseguir mayor versatilidad, se decidió incluir en Labomat3 un sistema operativo completo que

permita crear un entorno de comunicación más apropiado para el usuario y no sólo las posibilidades limitadas de un monitor.

Después de estudiar muchas posibilidades, se eligió el sistema operativo en tiempo real RTEMS [9], adecuado por sus modestas necesidades de memoria. Dispone de los drivers para Ethernet, que junto a una librería TCP/IP, permite el uso de protocolos estándar basados en TCP/IP. No es necesario programar al nivel físico de Ethernet. Además ya ha sido adaptado para el procesador 68360 y su código fuente es de dominio público. En cuanto al código específico para la tarjeta, la librería - Custom Hardware Library (CHL)- incluye un conjunto de funciones que permiten acceder a los recursos internos de la tarjeta.

Por encima del sistema operativo se ha instalado una máquina virtual JAVA. Las ventajas que proporciona Java son, entre otras, su simplicidad de utilización, un API estándar, y sus capacidades de comunicación por la red. Se ha elegido Kaffe [10] por ser accesible su código fuente y estar ya disponible para el procesador 68000. Además del API Java estándar, el usuario tiene a su disposición un API específico de la tarjeta, que proporciona clases y métodos para el acceso a los recursos de la tarjeta. La figura 3 muestra las diferentes capas de acceso usadas en la Labomat 3.

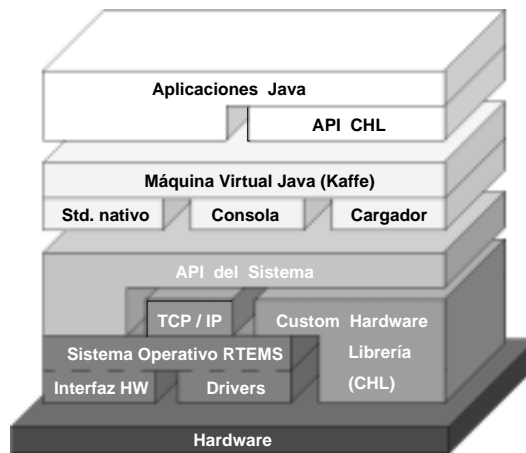


Figura 3: Estructura de capas que permiten el acceso a Labomat 3

### 4 Interfaz de comunicación

La plataforma Labomat 3 ofrece un variado conjunto de posibilidades de comunicación para configurar y controlar la tarjeta. El control puede hacerse a partir de un programa C, un programa Java o bien interactivamente de tres maneras diferentes (Figura 4):

1. La comunicación con la tarjeta se puede realizar con un programa de emulación empleando la

conexión RS-232. La consola del terminal muestra en un menú todas las opciones disponibles para controlar los recursos de la tarjeta, como por ejemplo, la programación de las FPGAs, selección de frecuencia de reloj, etc...

2. Labomat 3 también dispone de un servidor telnet. Una vez conectado a la tarjeta, el usuario elige los comandos de un menú en modo texto, semejante al que se obtiene con el emulador de terminal. En este caso, todos los comandos pasan ahora por la conexión Ethernet de Labomat 3, y el control de la tarjeta puede hacerse de manera remota.
3. En el Futuro, está previsto implementar un servidor Web sobre Labomat 3, para poder controlar la tarjeta a partir de un navegador Web. La idea es que una tarjeta conectada a Internet pueda ser programada y configurada desde cualquier máquina conectada a Internet, a partir de un programa integrado en una página Web. Los estudiantes podrán entonces trabajar en su casa con un navegador Web y comunicarse con una tarjeta instalada en el laboratorio. Actualmente, la conexión remota solo es posible via telnet, o mediante un servidor TCP/IP que se ejecuta como una tarea más en el sistema operativo RTEMS.

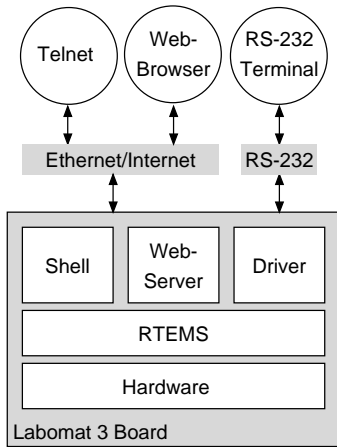


Figura 4: Posibilidades de comunicación

La estructura de comunicaciones, mostrada en la figura 4, permite desarrollar fácilmente otras aplicaciones a partir de los componentes básicos. Por ejemplo, ya se dispone de la máquina Virtual Java y de un servidor de comunicaciones, que usa un protocolo de paso de mensajes basado en sockets. La librería CHL incluye funciones C específicas que permiten acceder y controlar todos los recursos de la tarjeta. Esta capa adicional de software, permite a los estudiantes familiarizarse con la tarjeta en un corto periodo de tiempo.

Los ficheros de configuración de las FPGA admiten formatos Xilinx estándar (.hex y .cal) y son, en la implementación actual, leídos desde un servidor TFTP (Trivial File Transfer Protocol). La lectura remota de un fichero de configuración se realiza a partir de cualquiera de las distintas maneras de comunicación mencionados anteriormente, o bien directamente desde código fuente en C o en JAVA.

Por último, pensando en conectar por red un conjunto de tarjetas Labomat, se dispone en cada Tarjeta Labomat 3 de una dirección hardware (dirección MAC) que puede hacerse corresponder con cualquier dirección IP.

## Parte II

### Aplicaciones

#### 5 Diseño Lógico: Una Herramienta de diseño y verificación de circuitos lógicos

##### 5.1 Introducción

El desarrollo de un sistema digital basado en un circuito FPGA, puede descomponerse en diferentes etapas (ver Fig. 5): primero se diseña el circuito con un editor de esquemas; después se realiza su simulación. Cuando la simulación es satisfactoria, se procede a la implementación material sobre la FPGA. Esta etapa consta a su vez de diferentes pasos: en primer lugar se realiza una translación de los elementos lógicos a las celdas básicas que componen la FPGA (Mapping). Una vez que las celdas están definidas, se procede a su colocación para su posterior interconexión (Place and route). Estos procesos pueden optimizarse para mejorar el tamaño o la velocidad de funcionamiento del circuito a implementar sobre la FPGA.

La familia de circuitos FPGA Xilinx XC6200 proporciona la posibilidad de configurar independientemente cada celda básica, así como de leer su estado actual y eventualmente modificarlo. Esto abre un nuevo campo de aplicaciones basadas en la configuración del hardware en tiempo real, empleando la reconfiguración dinámica.

Nuestro objetivo es desarrollar un sistema de prototipaje **R**ápido para la **V**erificación de **S**istemas lógicos

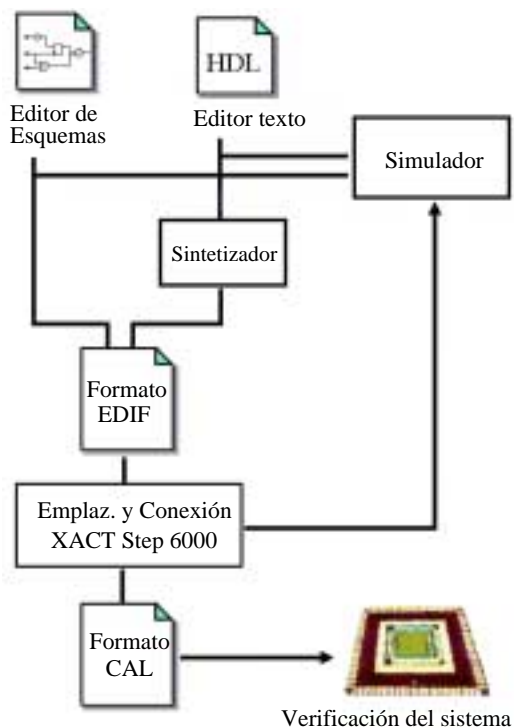


Figura 5: Etapas en el diseño de un circuito lógico sobre FPGA con herramientas estándar

usando la reconfiguración dinámica del circuito FPGA XC6200 (RVS6200). Esta aproximación difiere de las herramientas que se encuentran comercialmente disponibles. Xilinx y otros fabricantes dispone de productos comerciales que permiten todo el proceso de edición, simulación y síntesis, siguiendo las etapas anteriormente indicados en la figura 5. Lo novedoso es integrar (ver Fig. 6) en una sola herramienta todas las etapas: el editor de esquemas, el proceso de emplazamiento y conexionado, consiguiendo que sobre la FPGA tengamos una imagen exacta elemento a elemento del circuito diseñado, y el test del circuito, siendo posible verificar directamente el funcionamiento sin necesidad de realizar ninguna simulación.

## 5.2 Descripción del Circuito FPGA XC6216

El circuito XC6216 es una FPGA de grano fino que contiene una matriz bidimensional de 64X64 celdas. En una celda se puede implementar una función lógica de dos entradas, o un multiplexor y contiene además un registro. Cada celda dispone de recursos de conexión con sus vecinos y con conexiones más lejanas que se repiten periódicamente en las fronteras de cada 4 o 16 celdas. Las celdas de un XC6200 se configuran a partir de un control de memoria SRAM muy estable, con lo que no hay limitación en el número de veces



Figura 6: Diseño lógico, configuración y verificación simultánea en tiempo real

que se puede configurar, y se realiza de una manera muy rápida. Este control de memoria SRAM es directamente accesible desde el espacio de direcciones del microprocesador MC68360 de la tarjeta Labomat 3.

## 5.3 Descripción de RVS6200

La interfaz gráfica de usuario (GUI) es similar a la de cualquier herramienta de captura de esquemas comercialmente disponible. Su modo de empleo es intuitivo y fácil de aprender. Dispone de las funciones más usuales presentes en este tipo de GUI, como: cortar, copiar, pegar, aumento, disminución y los desplazamientos de ventana.

Una visión general de la aplicación se muestra en la Fig. 7. En la figura se observa cómo el fondo de la ventana tiene una retícula que representa la matriz de 64x64 celdas de la FPGA XC6216.

El editor de esquemas presenta tres modos básicos de operación, que coinciden con las operaciones a realizar para diseñar un esquema.

El modo **añadir componentes** permite la inserción de un componente desde una librería. La selección del componente se realiza desde un cuadro de diálogo. El emplazamiento del componente se realiza mediante el desplazamiento del mismo a la posición deseada de la retícula. Para facilitar esta operación se visualiza con un rectángulo el número de celdas que

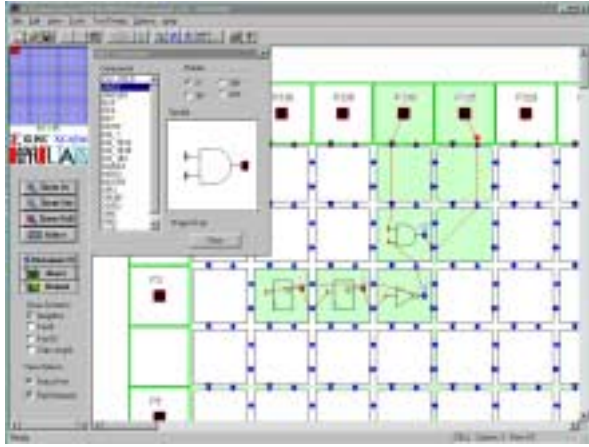


Figura 7: Aspecto general del editor de esquemas

ocupa el componente seleccionado.

El modo de **conexión** se emplea a continuación para conectar los diferentes componentes. Las conexiones deben ser realizadas en los puntos de contacto definidos en cada componente.

Además se dispone del modo **normal** como modo por defecto. En este modo se permite, seleccionar, cortar, copiar, pegar y borrar componentes. También es el modo desde donde se inicia la comunicación para configurar la FPGA y la posterior verificación del funcionamiento del circuito realizado.

También se contempla la posibilidad de diseño jerárquico, es decir, que un nuevo componente se pueda crear a partir de los elementos básicos que lo forman, incluyendo las conexiones entre elementos. A los nuevos componentes creados se les asocia un símbolo y pasan a estar disponibles como un elemento más de la librería.

Una vez realizado el diseño ya se dispone de toda la información de emplazamiento y conexionado a transmitir a las celdas de la FPGA. A partir del esquema se genera un fichero de configuración en formato CAL, para que puede ser enviado a la FPGA. La translación del diseño es inmediata como consecuencia de que el esquema es ya una imagen unívoca del circuito que se va a configurar en la FPGA, y por tanto el proceso de emplazamiento y conexionado consiste básicamente en un cambio de formato.

Otro aspecto interesante del programa RVS6200 es el hecho de que, una vez diseñado el esquema del circuito, no es necesario ninguna herramienta de simulación. Los valores de las salidas de cada componente en el diseño son verificados y mostrados sobre el esquema.

Para realizar el test del circuito configurado en la FPGA, se controla internamente el conexionado de la línea de reloj que accede a todas las celdas de la FPGA. Esta línea de reloj se ha conectado a la celda 0,0

de la FPGA y con solo cambiar el estado de esa celda se generan los sucesivos pulsos de reloj que llegaran a todos los elementos del circuito. La verificación del componente en la FPGA se realiza mediante la lectura del registro incluido en cada celda. Es posible también asignar valores como estímulos mediante la escritura de una celda por parte del usuario. Una vez verificado el funcionamiento del circuito, la línea de reloj puede conectarse de nuevo a su pin de entrada específico. A partir de entonces nuestro circuito funcionará con la frecuencia de reloj seleccionada para la tarjeta.

El programa RVS6200 hace uso del servidor TCP/IP con una comunicación basada en sockets mediante un protocolo de paso de mensajes. De manera remota se envía a la tarjeta el fichero de configuración y la generación de los pulsos de reloj y también se lee el estado actual de cada celda en la FPGA. Por tanto esta herramienta permite controlar Labomat 3 de manera remota, siendo posible ejecutar el programa desde cualquier máquina con acceso a internet. El único parámetro que se necesita es la dirección IP asociada a la tarjeta Labomat 3 que se pretenda utilizar.

#### 5.4 Ejemplo de aplicación: un multiplicador de 4 bits

A modo de ejemplo, se va a diseñar un multiplicador segmentado de 4 bits. Su estructura básica es una unidad de multiplicación de 1 bit repetida de una manera regular, y los registros necesarios para permitir la introducción de nuevos valores cada ciclo de reloj.

El primer paso es la definición de un multiplicador de un bit y su integración como un elemento más de la librería. Su esquema consta simplemente de una puerta AND y un sumador completo. Una vez que se han colocado los componentes se realizan las conexiones mostradas en la figura 8. El proceso para crear un nuevo componente comienza con la selección de todas las celdas que lo componen. Una opción de Menú permite crear el nuevo componente y asociarle un nombre. Automáticamente el grupo de celdas seleccionadas forman el componente, y las entradas y salidas que no tengan conexión son las que se asignan como entradas y salidas del nuevo componente. Posteriormente se asocia un símbolo al grupo de celdas mediante un editor que permite realizar símbolos simples como una colección de líneas, rectángulos, círculos y curvas Bezier. Por último se añade el elemento a una librería para que posteriormente pueda seleccionarse como un elemento más de la misma. Para generar un multiplicador de 4 bits, basta con conectar el multiplicador 1x1 bit que acabamos de crear de la manera indicada en la figura 8.

Una vez finalizado el diseño del esquema, se establece una conexión con el servidor TCP/IP que está ejecu-

tandose en la tarjeta Labomat 3. Establecido el enlace se habilitan los comandos de configuración y test. Un nuevo comando (Send Data to Xilinx) genera y envía a la FPGA el fichero de configuración. Este fichero es tipo texto en formato CAL estandar de Xilinx. Este formato es un conjunto de parejas de números en hexadecimal. El primer número es una dirección para indicar la celda concreta que queremos configurar y el segundo es el dato que permite configurar los multiplexores de cada celda. Este formato es de dominio público. El envío del fichero se realiza mediante sockets que comunican RVS6200 como cliente del servidor que está ejecutándose en Labomat3. Configurada la FPGA, se pueden enviar sucesivos pulsos de reloj a la FPGA. En cada pulso de reloj, se muestra sobre el esquema en pantalla el valor lógico que corresponde a la salida de cada celda. La actualización de las señales en cada ciclo de reloj permite la verificación del funcionamiento y la detección de errores de una manera inmediata.

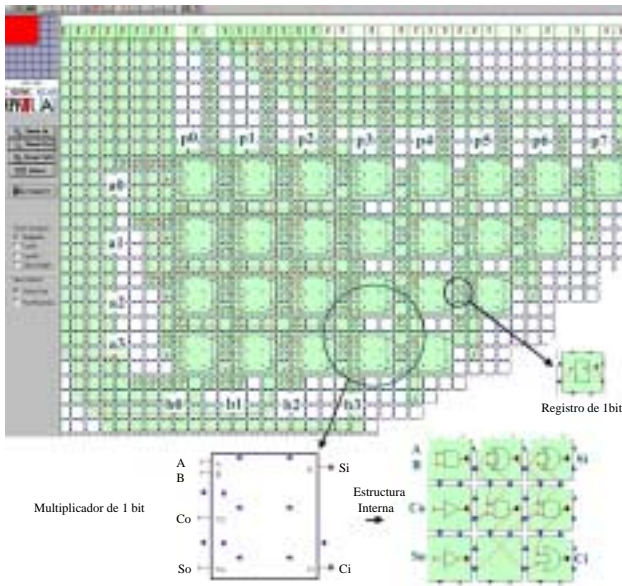


Figura 8: Multiplicador segmentado de 4 bits

## 6 Arquitectura de Ordenadores: Un Procesador Segmentado Sencillo

### 6.1 Introducción

Este apartado presenta la implementación de un procesador segmentado sobre la FPGA XC4013E de Xilinx en nuestra tarjeta. Los estudiantes no solamente pueden simular su procesador, sino que además pueden ir un paso más lejos realizándolo materialmente sobre hardware, y así realmente comprobar su funcionamiento. El objetivo no es realizar un proce-

sador de alto rendimiento, sino más bien examinar ciertas dificultades tales como dependencias y excepciones que aparecen de manera natural en un procesador segmentado. Se trata por tanto claramente de una aplicación académica.

### 6.2 Especificaciones

Las especificaciones de este procesador están en parte inspiradas en la arquitectura DLX [4]: Se trata de una arquitectura load/store 8 bits con cuatro registros de 8 bits (R0,R1,R2 y R3). A diferencia de la arquitectura DLX, el contenido del registro R0 no es siempre cero. No hay registros de coma flotante y el único tipo de dato soportado son enteros de 8 bits.

El juego de instrucciones consta de las siguientes 8 instrucciones:

LOAD	$Rd \leftarrow M[\text{dirección}]$
STORE	$M[\text{dirección}] \leftarrow Rs$
MOVE	$Rd \leftarrow Rs$
SUB	$Rd \leftarrow Rs1 - Rs2$
ADD	$Rd \leftarrow Rs1 + Rs2$
CMP	$flag \leftarrow Rs1 - Rs2$
BRANCH	si flag entonces $PC \leftarrow PC + desplazamiento$
JMP	$PC \leftarrow PC + desplazamiento$

Rd, Rs, Rs1 o Rs2 representan uno de los cuatro registros disponibles y  $M[\text{dirección}]$  un acceso a la dirección de memoria *dirección*. Ninguna instrucción genera más de un resultado.

Se ha definido una unidad de segmentación con 5 etapas:

IF	(Instruction Fetch) Captura instrucción
ID	(Instruction Decode) Decodificación
EX	(Execution) Ejecución
MEM	(Memory access) Acceso a Memoria
WB	(Write back) Postescritura

Cada etapa se ejecuta en un ciclo de reloj.

### 6.3 Formato de instrucciones

De la misma manera que en los procesadores RISC actuales, las instrucciones tienen un formato de longitud fija (16 bits). El único modo de direccionamiento disponible es direccionamiento absoluto para las instrucciones load/store y el direccionamiento relativo al PC para las instrucciones de ramificación y salto. Todas las instrucciones tienen un código de operación de 3 bit y el modo de direccionamiento va indicado implícitamente en la instrucción.

Cualquiera de los cuatro registros de propósito general puede utilizarse en las operaciones con memoria (Load/Store). Los registros se especifican mediante dos bits en el formato de la instrucción, quedando 11 bits para indicar el desplazamiento en las instrucciones de salto.

Instr.	OPC															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD	0	0	0	Rd	dirección											
STORE	0	0	1	Rs	dirección											
MOVE	0	1	0	Rs	Rd	0	0	0	0	0	0	0	0	0	0	0
SUB	0	1	1	Rd	Rs1	Rs2	0	0	0	0	0	0	0	0	0	0
ADD	1	0	0	Rd	Rs1	Rs2	0	0	0	0	0	0	0	0	0	0
CMP	1	0	1	0	0	Rs1	Rs2	0	0	0	0	0	0	0	0	0
BRANCH	1	1	0	0	0	desplazamiento										
JMP	1	1	1	0	0	desplazamiento										

Figura 9: Formato de instrucciones

## 6.4 Arquitectura del Procesador

Para evitar paradas de la unidad de segmentación y simplificar el acceso de instrucciones y datos en los puertos de I/O, la memoria de instrucciones y de datos están separadas (Arquitectura Harvard). De esta manera es posible leer una instrucción (Etapa IF) y simultáneamente acceder a un dato en memoria (Etapa MEM). La unidad de detección de riesgos colabora muy estrechamente con la unidad de control. Los cinco bloques funcionales que aparecen en la figura 10 se han implementado como entidades independientes empleando un lenguaje de descripción de Hardware - VHDL.

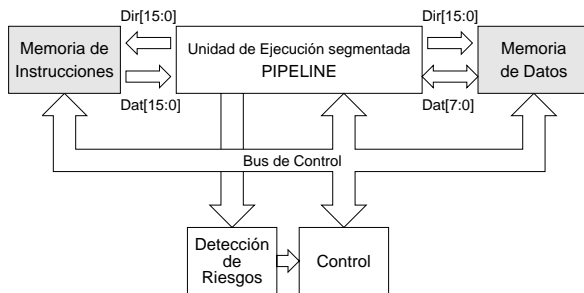


Figura 10: Arquitectura del procesador

## 6.5 Unidad de segmentación

Para alcanzar la ejecución de una instrucción por ciclo de reloj, se tiene que poder ejecutar cualquier combinación de instrucciones. Para evitar riesgos estructurales, la unidad de segmentación mostrada en la figura 11 no tiene conflictos de recursos, es decir, cada instrucción se puede ejecutar simultáneamente y solaparse con cualquier instrucción. El acceso al banco de registros de propósito general puede realizarse con ambos flancos de reloj. En cada una de las salidas del banco de registros (oa,ob,oc) puede seleccionarse, mediante un multiplexor interno, uno de los cuatro registros (R0,R1,R2,R3). El contenido de los registros

se modifica en el flanco de subida del reloj en la etapa de Postescritura (WB), y se leen con el flanco de subida ascendente. Los registros temporales (En color gris oscuro en la figura 11) mantienen los valores hasta el siguiente ciclo de reloj. Estos registros se cargan sincronamente (señales de control LREGIF, LREGID, LREGEX, LREGMEM) con el flanco descendente del reloj si no ocurren bloqueos en la unidad de segmentación.

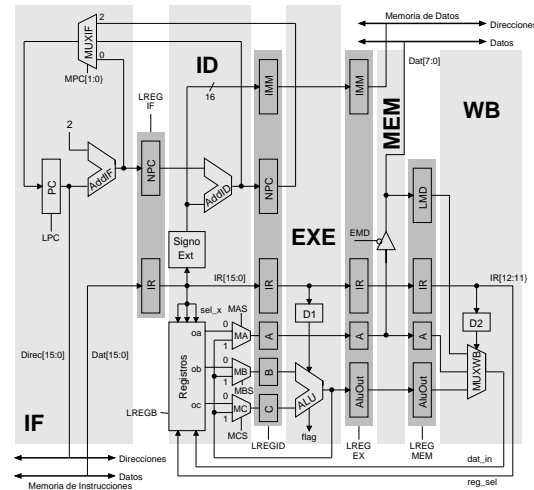


Figura 11: Unidad de segmentación

Como consecuencia de la ejecución de una instrucción por ciclo (CPI = 1), el contador de programa (PC) se incrementa en la etapa de captura de instrucciones. Tres multiplexores (MA,MB,MC) permiten avanzar los datos. Dos decodificadores (D1 y D2) controlan las operaciones de la ALU y del multiplexor de postescritura (MUXWD). Todas estas operaciones dependen de las instrucciones contenidas en el registro IR. El bloque de extensión de signo en la etapa ID permite extender el desplazamiento y la dirección absoluta de los 11 bits codificada en la instrucción hasta 16 bits.

## 6.6 Detección de Riesgos

La unidad de detección de riesgos, sólo tiene como entrada la instrucción capturada. Como no hay riesgos estructurales, sólo hay que detectar riesgos de control y dependencias de datos. Se han tratado los riesgos de control de la manera más fácil: una instrucción de salto incondicional bloquea la unidad de segmentación durante un ciclo de reloj y una instrucción de salto condicional durante dos ciclos, independientemente de si el salto es efectivo o no. También se ha adoptado una solución simple para tratar las dependencias de datos: el número del registro destino (registro que será modificado en la última etapa de la unidad de segmentación) se introduce en la etapa de captura (IF) en una cola FIFO de cuatro elementos.

Por ejemplo, si se considera la siguiente instrucción *move*:

```
MOVE R2 ← R3
```

Esta instrucción modificará el registro de propósito general número 2 en la etapa de postescritura (WB). Por consiguiente el número 2 es introducido en la cola FIFO. Los elementos de la cola se desplazan una posición cada ciclo de reloj. Cada vez que se introduce una nueva instrucción en el registro de captura de instrucciones, se realiza una comprobación para ver si el número de registro usado como fuente (3 en el ejemplo anterior) se encuentra almacenado en la cola FIFO. Si se encuentra presente la unidad de segmentación se bloquea hasta que el número en cuestión desaparece de la FIFO, es decir, hasta que el registro ha sido modificado. Si no es el caso, se puede continuar capturando instrucciones, y no se bloquea la unidad de segmentación. En el peor de los casos, una dependencia de datos genera un bloqueo de cuatro ciclos de reloj como se ilustra con el siguiente ejemplo:

```
MOVE R2 ← R3
MOVE R0 ← R2
```

La segunda instrucción no puede ejecutarse antes de que la primera se haya completado. El número 2 (correspondiente al registro modificado por la primera instrucción) permanecerá en la cola FIFO durante cuatro ciclos de reloj. Claramente, se puede mejorar la eficiencia implementando un mecanismo de avance de datos.

El mecanismo de avance de datos en la figura 11 ya está implementado, pero en la versión actual de la unidad de segmentación no ha sido utilizado en la unidad de control (Fig. 10).

## 6.7 Implementación

Este procesador RISC se ha implementado completamente en VHDL con la ayuda del sintetizador FPGA-Express. Se ha empleado una simulación de la memoria de instrucciones para evitar restricciones de tiempos de acceso, aunque se podría haber utilizado un modelo más realista, respetando los tiempos de acceso, empleando como memoria de instrucciones la SRAM de doble acceso conectada a la FPGA XC4013E. En este caso, el microprocesador 68360 podría emplearse para escribir y cambiar el código de instrucciones a ejecutar por el procesador RISC implementado en la FPGA. La memoria de datos se hace corresponder directamente, con los valores de interruptores y led externos. Para ello se emplean los conectores a los buses de extensión de 8 bit de Labomat3. Empleando el botón de ejecución paso a paso, puede verificarse el funcionamiento o, en su caso, detectar los errores.

La realización física sobre la FPGA XC4013E emplea 296 CLBs de los 576 disponibles. Por lo que

quedan suficientes recursos disponibles para realizar futuras extensiones o mejoras. La frecuencia máxima de funcionamiento del diseño es de 25 MHz.

## 7 Codiseño: Un Coprocesador de Coma-Flotante

Esta aplicación consiste en la realización de un coprocesador de coma flotante (FPU, Floating Point Unit). Los estudiantes deben diseñar un multiplicador de coma flotante a nivel de esquema de puertas lógicas, simular su funcionamiento y posteriormente comprobarlo físicamente sobre la tarjeta Labomat 3 (fig. 12). El trabajo consistirá, por un lado, en la programación de la FPGA con el multiplicador diseñado y también su interfase con el procesador, y por otro lado, la escritura de un programa para que el procesador controle y compruebe el funcionamiento del coprocesador.

El procesador puede acceder a la FPGA como si fuera un periférico, utilizando directamente los buses de direcciones y de datos. El interfaz de comunicaciones a implementar en la FPGA se debe ocupar sólo de la decodificación de la dirección y registrar los datos que llegan por el bus. En la tarjeta Labomat 3 el control de escrituras y lecturas en los buses queda oculto y la FPGA no necesita generar complejas señales de control. De hecho, la FPGA está separada del 68360 por una capa de lógica que se encarga de adaptar las frecuencias de reloj. Por tanto, sólo se necesita generar señales de control para el asentimiento de la recepción de operandos y para indicar cuando la FPU ha terminado el cálculo. Estas señales ya estarán sincronizadas con la frecuencia de reloj utilizada.

El procesador y la FPU se comunican empleando el modo de acceso a periférico del procesador. El protocolo de comunicación comienza con la escritura por parte del procesador de los operandos en direcciones específicas de la FPGA. La FPGA debe asentir la recepción de operandos. El procesador queda en estado de espera hasta que la FPGA termine el cálculo. Esta espera puede controlarse mediante lectura continua de un registro de estado de la FPGA, que indique el estado del proceso de cálculo, o bien mediante interrupciones. Finalizado el cálculo en la FPU, el procesador puede leer el resultado en una dirección prefijada.

El desarrollo de la FPU se lleva a cabo usando el entorno de trabajo WorkView Office para ordenadores personales con Windows-NT. Básicamente está compuesto de lógica combinatorial controlada mediante máquinas de estados, que son las encargadas de gestionar la secuencia de microoperaciones. Pueden distinguirse tres módulos interaccionando entre sí, que corresponden a los cálculos del signo, la mantisa y el exponente.

El programa de test para comprobar el fun-

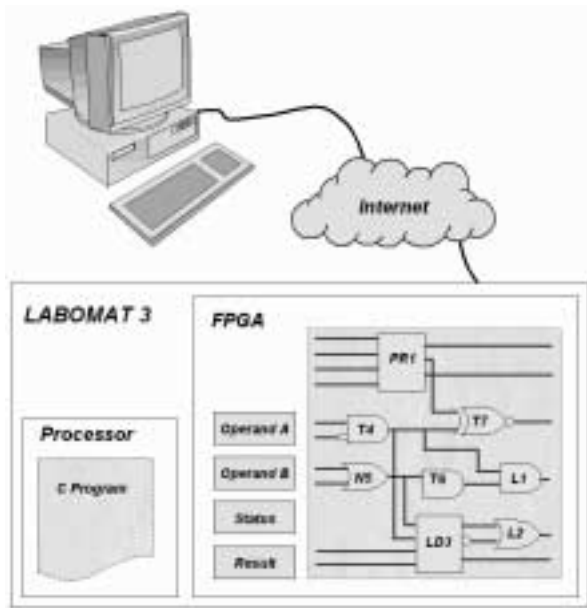


Figura 12: Labomat 3 como herramienta para el codiseño

cionamiento, escrito en C, comienza con una llamada a una rutina predefinida para la configuración de la FPGA. Después el programa entra en un bucle de lectura para la introducción de los dos operandos desde el terminal y, una vez leídos, enviarlos al coprocesador. Una vez que el cálculo está terminado, el resultado es leído y mostrado en el terminal. Gracias a los programas ya disponibles para la tarjeta Labomat3, el usuario no debe preocuparse de la inicialización del procesador ni de otros problemas de bajo nivel.

## 8 Conclusiones

El acceso remoto por red y la combinación de recursos programables con herramientas que permitan trabajar a un alto nivel de especificación, hacen de Labomat 3 una plataforma ideal para la experimentación y la concepción de sistemas que consten de una parte hardware y de otra software. Su utilización es especialmente adecuada cuando se necesite investigar como conseguir la mejor interacción entre ambas partes.

Labomat 3 puede emplearse para desarrollar y validar herramientas automáticas de codiseño. Aprovechando que su arquitectura es conceptualmente simple, el diseñador puede concentrar su esfuerzo en el particionamiento y la síntesis sin tener que estar pendiente de la complejidad de la plataforma de desarrollo.

En cuanto a su utilidad docente, nuestro laboratorio de estudiantes está actualmente equipado con 50 tarjetas Labomat3, que ya han sido utilizadas con éxito para ejercicios de prácticas. Los estudiantes tienen así la posibilidad de ir un paso más lejos de la simu-

lación, comprobando sus sistemas realmente en Hardware y enfrentándose a problemas que no aparecen en la simulación.

En un futuro próximo, dispondremos de herramientas que permitan emplear las 50 tarjetas simultáneamente, con el objetivo de conseguir un gran conjunto reconfigurable (100 FPGA) para realizar experiencias de cálculo de alto rendimiento. La comunicación entre las tarjetas será posible por Ethernet, o por líneas de conexión directa que enlacen las tarjetas. Ya hay previstos experimentos en dominios de redes neuronales artificiales, algoritmos genéticos y de vida artificial.

## Agradecimientos

Queremos expresar nuestro agradecimiento a André Badertscher por la fotografía y por el montaje de la tarjeta. Este trabajo está parcialmente financiado por una beca de la fundación Werner Steiger.

## Referencias

- [1] D. van den Bout. *The practical Xilinx designer lab book*. Prentice Hall. 1998.
- [2] L. Garber and D. Sims. *In pursuit of hardware-software codesign*. IEEE Computer, 31(6):12-14, June 1998.
- [3] J.O. Haenni, J.L. Beuchat, E. Sanchez. *RENCO: A Reconfigurable Network Computer*. Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines. FCCM '98. Napa USA. April 15-17, 1998. pp. 288-289.
- [4] J.L. Hennessy, D.A. Patterson. *Computer Architecture - A Quantitative Approach*. Morgan Kaufmann. Second Edition, 1990.
- [5] Z. Salcic and A. Smailagic. *Digital systems design and prototyping using field programmable logic*. Kluwer. 1997.
- [6] E. Sanchez, M. Sipper, J.O. Haenni, J.L. Beuchat, A. Stauffer, and Andres Perez-Urbe. *Static and Dynamic Configurable Systems*. IEEE Transactions on Computers, Special Issue on Configurable Computing, 48(6):556-564, June 1999.
- [7] J. Villasenor and W. H. Mangione-Smith. *Configurable computing*. Scientific American, 276(6):54-59, June 1997.
- [8] Motorola MC68360. *Quad Integrated Communication Controller*. User's Manual. 1993.
- [9] RTEMS Home Page. <http://www.oarcorp.com>.

- [10] Kaffe, A free virtual machine to run Java code.  
<http://www.kaffe.org>.
- [11] Xilinx. *Inc XC6200 Advanced product specification* V1.10 4/97. The Programmable Logic Data Book 1997. <http://www.xilinx.com>.