DISEÑO EN VERILOG DE UNA ARQUITECTURA SEGMENTADA DE MICROPROCESADOR RISC

Luis Carlos Rosales Alpízar
Johan Bermúdez
Ruddy Céspedes
Eddy Morales Rodríguez
Roberto Pereira Arroyo
Alfonso Chacón Rodríguez

Escuela de Ingeniería Electrónica. Instituto Tecnológico de Costa Rica.

luis.carlos.rosales@gmail.com
johanbermudez@hotmail.com
ruddycc@yahoo.com
rpereira@itcr.ac.cr
achacon@itcr.ac.cr

ABSTRACT

Se propone una arquitectura segmentada de microprocesador RISC de 32 bits, implementada el nivel de RTL en un lenguaje de descripción de hardware, y de código compatible con un microprocesador MIPS®.

Se han usado las herramientas de desarrollo de la empresa Xilinx®. El código ha sido testeado en un sistema de desarrollo de dispositivos programables de Digilent Inc., mediante la ejecución en un sistema mínimo de diferentes programas escritos en lenguaje ensamblador de la familia MIPS®. Se han realizado pruebas de síntesis sobre una tecnología de 0.35 µM, y se han integrado estructuras de testeo al código mediante herramientas Design for Test.

1. INTRODUCCION

El proyecto Tesla tiene como meta final implementar en silicio un microprocesador de arquitectura RISC de 32 bits con un juego de instrucciones compatible con el microprocesador MIPS. Esta es el resultado de un proyecto de investigación y desarrollo de la Escuela de Ingeniería en Electrónica del Instituto Tecnológico de Costa Rica para desarrollar el currículum de pregrado en las áreas de microelectrónica y arquitectura de microprocesadores.

Luego de tres períodos académicos, se cuenta con el diseño en lenguaje Verilog del microprocesador y su implementación y síntesis utilizando herramientas de desarrollo de dispositivos programables. Además, se ha obtenido una síntesis para una tecnología de 0.35µm, a la que se han insertado estructuras de testeo post-fabricación. Además del modelo básico del microprocesador, se provee de soporte de una pila de memoria y de una interfaz de bus para control de memoria asíncrona y dispositivos periféricos.

Los resultados han sido verificados mediante la implementación del microprocesador y un sistema mínimo utilizando el ISE de Xilinx®, y cargando ambos en una tarjeta Spartan 3 de Digilent Inc. El sistema mínimo utiliza como dispositivo periférico de entrada un teclado PS2 y como periférico de salida un display LCD, e integra la memoria asociada para el programa y la memoria asociada para datos.

La sección 2 describe la arquitectura digital del sistema. La sección 3 detalla la implementación del sistema en Verilog y sus pruebas en un sistema programable. La sección 4 muestras los avances hechos en el diseño final del chip, que se encuentra en la actualidad en la etapa de layout.

.

2. ARQUITECTURA DEL MICROPROCESADOR TESLA

Tesla cuenta con un modelo de arquitectura segmentada en 4 etapas que permiten ejecutar varias tareas de manera simultánea, es decir, cuenta con una estructura de segmentación *(pipeline)* de 4 etapas de profundidad.

Las etapas de Tesla son: Instruction Fetch (IF), Instruction Decode (ID), Execution (EX) y Memory_WriteBack (MEM_WB), las cuales realizan las tareas de Fetch o búsqueda de instrucciones, decodificación, ejecución y escritura/lectura de datos en memoria o retroalimentación de los mismos respectivamente (Figura 1).

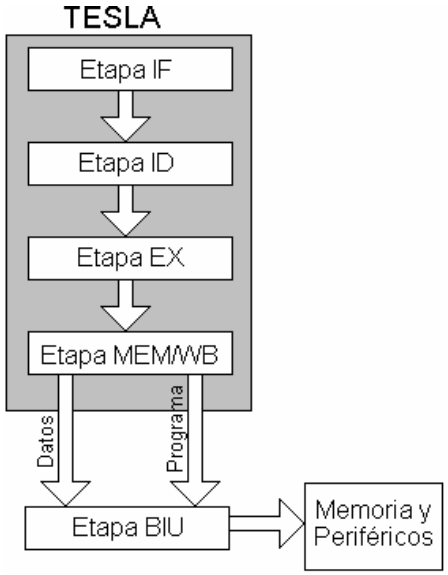

Figura 1 Diagrama general de bloques de Tesla y su interfaz de conexión

El modelo de arquitectura de Tesla es del tipo Harvard, es decir, el microprocesador utiliza 2 bancos de memoria, uno para el almacenamiento del programa de la cual se leen las instrucciones que deberán ser ejecutadas y el otro banco de memoria se utiliza para la lectura/escritura de datos. La Figura 2 resume la estructura interna básica de la que partió el diseño [1]. Esta recibió modificaciones a lo largo del proyecto. Para lograr que Tesla pudiera comunicarse con ambos bancos de memoria, así como con los dispositivos periféricos, se creó una unidad de interfaz con el bus de memoria y periféricos que se llama BIU (Bus Interface Unit).

2.1 Unidad de búsqueda de instrucciones (Instruction Fetch - IF)

Esta unidad es la encargada de realizar una lectura de la memoria de instrucciones para determinar la siguiente operación a ser ejecutada. El acceso a la memoria de instrucciones se controla por un contador de programa o PC (Program Counter) que determina la dirección de la memoria de instrucciones donde se deberá leer la siguiente instrucción. En condiciones normales, el PC cambia su valor mediante incrementos de 4, sin embargo, si alguna de las instrucciones requiere que el valor del PC cambie a una dirección específica se puede lograr mediante la realimentación o el write back. Para esto, el PC cuenta con un multiplexor que controla su valor y permite seleccionar si se carga el valor realimentado o el valor de incremento normal.

De acuerdo con el valor del PC la memoria de instrucciones entrega un código de operación que será luego decodificado. Este código de operación se almacena en un registro de interfaz que mantiene el dato de salida de IF hasta que la siguiente unidad de Tesla (ID) lo pueda utilizar sin correr el riesgo de perder el dato o de generar resultados erróneos.

2.2 Unidad de decodificación de instrucciones (Instruction Decode - ID)

Esta unidad recibe el código de operación. Este código lleva información necesaria para ejecutar las operaciones, y la decodificación de la operación consiste básicamente en ubicar los valores de los operandos que se deberán extraer de un banco de 32 registros del microprocesador para pasarlos a la unidad de ejecución y cálculo.

Dentro de la información de entrada a la unidad, se cuenta con un campo de 16 bits para un operando inmediato, que es modificado mediante una extensión de signo hasta 32 bits. También se leen las direcciones de los registros que se deberán leer/escribir en las operaciones realizadas. Hay sistemas de realimentación y de una unidad de detección de riesgos para el reenvío de datos, para casos en que una instrucción en esta etapa requiera de datos todavía no almacenados en los registros y que están disponibles ya en una etapa siguiente. Si el dato no está todavía disponible, esta unidad es la responsable de introducir burbujas de espera en el flujo (NOP), para asegurar que dicha disponibilidad al pasar a la unidad de ejecución. En esta unidad puede ejecutarse también la definición del siguiente estado del PC para operaciones que requieran de una modificación de este, como en bifurcaciones a partir de decisiones sobre determinados datos. Para realizar este cálculo se utiliza el valor actual del PC y el valor del operando inmediato, y los valores de salida del banco de registros o los provistos por la unidad de reenvío, dependiendo de la etapa en que se encuentren los datos sobre los que debe decidirse. El resultado se retroalimenta a la unidad de búsqueda de instrucciones (IF). Realizar el cálculo de saltos aquí ahorra una ranura en la ejecución con respecto a si se hiciera en la unidad EX. Es por lo anterior que, técnicamente, esta unidad puede también llamarse de pre-ejecución.

2.3 Unidad de ejecución de instrucciones (Execution Unit - EX)

Aquí se realizan los cálculos asociados con la ejecución de la operación. Hay una que ejecuta operaciones con los parámetros de entrada en función del código de operación. Los parámetros de entrada son los que se leyeron del banco de registros en la unidad anterior y uno de los

parámetros de entrada puede ser el operando inmediato que también proviene de la unidad ID. Los resultados de los procesos de cálculo y modificación del PC se almacenan en un registro de salida que se denomina EX/MEM.

2.4 Unidad de manejo de memoria y retroalimentación (MEM_WB)

Esta unidad unifica el manejo de la interfaz con memoria y las retroalimentaciones a las unidades anteriores. En cuanto al manejo de memoria, como se indicó en la unidad de IF, la memoria de instrucciones se maneja de manera independiente de la memoria de datos. La unidad de MEM_WB se hace cargo de la memoria de datos, y para definir las ubicaciones de esta memoria que serán leídas o escritas se utiliza el resultado de la ALU de la unidad de ejecución. Para el caso de las operaciones en que se requiere de datos leídos de la memoria se procede a utilizar un registro de WB, que toma los datos obtenidos de la memoria o directamente del resultado calculado por la ALU en la unidad de EX.

2.5 Unidad de interfaz con el bus (BIU)

Aunque esta unidad no es parte integral de Tesla es un componente externo de suma importancia dado que permite la utilización del microprocesador en un sistema mínimo para sus pruebas.

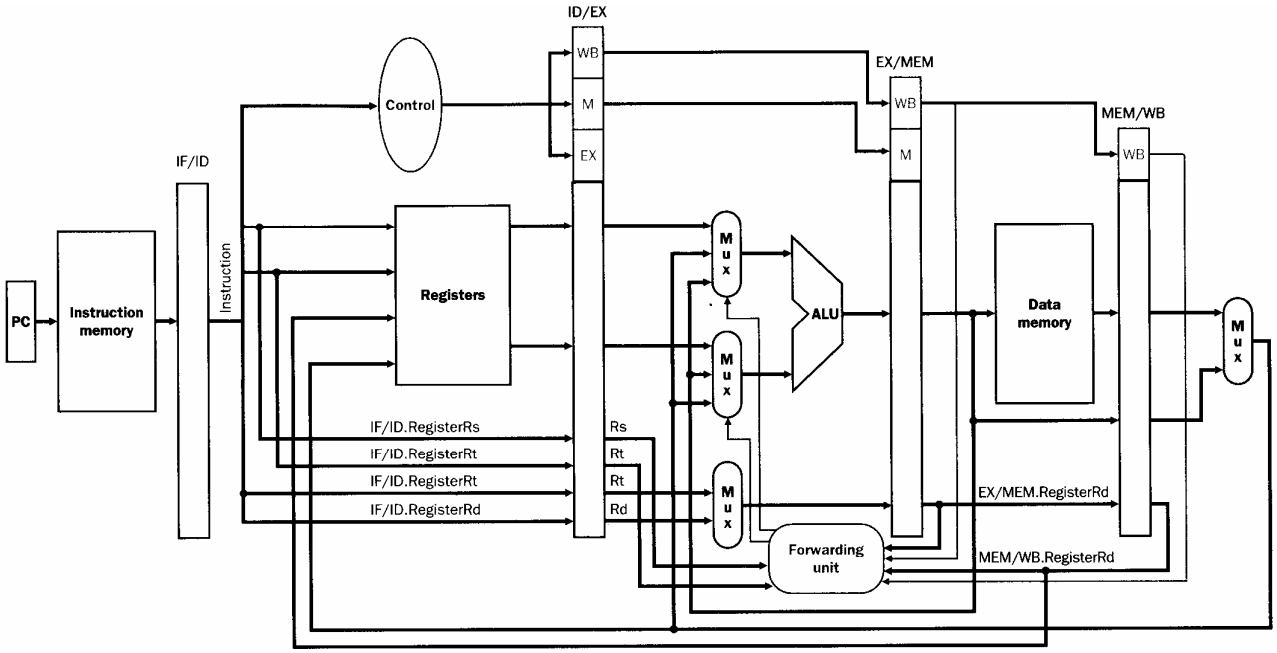

Figura 2. Modelo básico original de la arquitectura de Tesla implementada (tomado de [1]). En la implementación final, la unidad de reenvío (Forwarding unit) se trasladó a la etapa de ID. La decisión y cálculo de saltos también se trasladaron a ID, para ahorrar una ranura de ejecución.

La BIU recibe de Tesla los buses de datos y direcciones destinados a las memorias de datos y de programa, luego unifica ambos buses y genera un solo bus de direcciones y un solo bus de datos que se interconectan con ambos bancos de memoria, con los periféricos y con un decodificador que se encarga de activar el segmento de memoria a utilizar.

Esta unidad debe además controlar los tiempos para interfazar dispositivos lentos como las EEPROM al bus de datos de microprocesador, sin penalizar la frecuencia de operación del sistema general.

2.6 Clasificación de instrucciones

Tesla cuenta con un juego de instrucciones compatible con el del microprocesador comercial MIPS®; adicionalmente se han incluido una serie de instrucciones para enriquecer y optimizar ciertas operaciones. De acuerdo con esto, cada una de las instrucciones se compone por 32 bits y se agrupan en tipos según las acciones que realiza la operación.

2.6.1 Tipo R – Operaciones de registro a registro

Son operaciones que toman como parámetros de entrada los valores almacenados en el banco de registros y el

resultado se maneja nuevamente como parte del banco de registros. Organiza sus 32 bits de la siguiente manera:

- OP: Bits del 31 al 26 que se usan para almacenar el código de operación
- RS: Bits del 25 al 21 que almacenan la dirección del registro fuente del primer operando
- RT: Bits del 20 al 16 que indican la dirección del registro fuente del segundo operando.
- RD: Bits del 15 al 11 para indicar la dirección del registro que almacenará el resultado de la operación
- Shamt: Bits del 10 al 6 utilizado como indicador de funciones de corrimiento (Shift Ammount)
- Funct: Bits del 5 al 0 que indican la variante de la función que será ejecutada

2.6.2 Tipo I – Operaciones de registro a operando inmediato

Son operaciones que toman parámetros de entrada valores del banco de registros que luego son almacenados como operando inmediato. Se organizan los 32 bits de la siguiente manera:

- OP: Bits del 31 al 26 para indicar el código de operación.
- RS: Bits del 25 al 21 que indican la dirección del registro fuente del primer operando.
- RT: Bits del 20 al 16 que indican la dirección del registro fuente del segundo operando.
- Inmediate: Bits del 15 al 0 que representan el valor del operando inmediato.

2.6.2 Tipo J – Operaciones de salto

Son operaciones que se utilizan para definir una dirección dentro de la memoria de programa a la cual se salta para continuar su ejecución, por lo que solamente se utilizan los siguientes 2 campos:

- OP: Bits del 31 al 26 para representar el codigo de operación
- Address: Bits del 25 al 0 para indicar la dirección de memoria donde se ubicará el contador de programa.

La Tabla 1, muestra el detalle de las instrucciones implementadas en el microprocesador.

3. IMPLEMENTACIÓN Y VERIFICACIÓN DEL MODELO

Una vez analizado y definido el diseño de arquitectura, se procedió a trabajar en la implementación de las unidades básicas de Tesla, así como las unidades de interfaz y adicionales utilizando Verilog según la especificación de 1993. La segmentación de la arquitectura por etapas permitió que se concentrara el trabajo en cada etapa sin recurrir a las demás etapas, siguiendo una estructura lógica de trabajo recomendada para las etapas de diseño o front-end dentro del trabajo de la industria de los circuitos integrados.

3.1 Estructuras lógicas de las unidades básicas del microprocesador

A partir de la arquitectura descrita en la etapa de diseño, se procedió a implementar una estructura lógica para cada etapa del microprocesador a nivel de RTL.

3.1.1 Unidad de búsqueda de instrucción (Instruction Fetch)

Aquí se retira la instrucción de la memoria de programa y se actualiza el contador de programa o PC en función de la instrucción que se está ejecutando. La estructura lógica que se maneja es una lista de casos que indica los posibles valores del PC. Las variables de entrada que regulan el valor del PC, provienen de la etapa de WB o de ID del microprocesador, esto porque el cambio en el valor del PC estará definido por los resultados de la instrucción que se encuentra en la fase final de ejecución, o por una decisión de salto en ID.

En caso de que todas las variables de control de cambio del PC se mantengan pasivas, se aplica un incremento normal al contador, que se aumenta en valores decimales de 4, y en caso de que se aplique una condición de reset, la unidad IF envía al PC a su dirección de inicio (0x00400020).

Hay además una estructura de registros al final de la etapa, las cuales almacenan los valores de salida de las diferentes señales. Estos registros son parte de la estructura de control y detección de riesgos de datos, propia del modelo descrito en el diseño.

3.1.2 Unidad de decodificación de instrucción (Instruction Decode)

La unidad de decodificación de instrucciones se estructura como una serie ordenada de casos, empezando por un caso principal donde se debe definir el tipo de instrucción, que puede ser R,I o J.

El valor del código de operación de 32 bits se descompone para leer los segmentos de interés según el tipo de operación, como por ejemplo los registros de destino y fuente a utilizar en cada operación. Asimismo, se realiza una evaluación del valor que se debe asignar al PC en función de cada instrucción ejecutada, lo cual aplica en instrucciones de tipo condicional y en instrucciones tipo J. Se cierra la etapa con una estructura de actualización de registros de segmentación para la protección por riesgo de pérdida de datos.

3.1.3 Unidad de ejecución (Execution Unit)

Esta unidad recibe como parámetro principal de entrada el código de operación. La definición lógica de las operaciones se maneja a partir de una tabla de

mnemónicos para cada una de las instrucciones. Se implementaron 31 instrucciones básicas.

Los parámetros que se ingresan a las operaciones ejecutadas por la ALU, son tomados del banco de registros definido o del código mismo, para operandos inmediatos. La Tabla 1 resume las instrucciones construidas para esta versión del microprocesador.

Tabla 1. Instrucciones implementadas en Tesla

Instrucción	Significado
sll	rd = rt << shamt
srl	rd = rt >> shamt
sra	rd = rt >> shamt
jr	PC= rs
add	rd <= rs + rt
sub	rd <= rs - rt
and	rd <= rs and rt
or	rd <= rs or rt
xor	rd <= rs xor rt
slt	si rs > rt => rd <= 1 si no, rd<=0
sltu	Igual que slt, pero sin signo
addi	rd <= rs + inmediato
j	PC <= destino
jal	PC <= destino ra <= PC+4
beq	(rs=rt)?:[PC=destino] [PC=PC+4]
bne	(rs<>rt)?:[PC=destino] [PC=PC+4]
lw	rd <= mem [rs]
sw	mem[rs] <= rd
nop	No se ejecuta nada
lui	$rt << imm&0000h
lbu	Rt = {[24'b0],[byte]}
lb	Rt = {[24'byte(7)],[byte]}
lh	Rt = {[16'b0],[half]}
lhu	Rt = {[15'half(15)],[half]}
addu	Rd = rs + rt
subu	Rd = rs - rt
bgtz	(rs>0)?:[PC=Fuente] [PC=PC+4]
bltz	(rs<0)?:[PC=Fuente] [PC=PC+4]
bgez	(rs≥0)?:[PC=Fuente] [PC=PC+4]

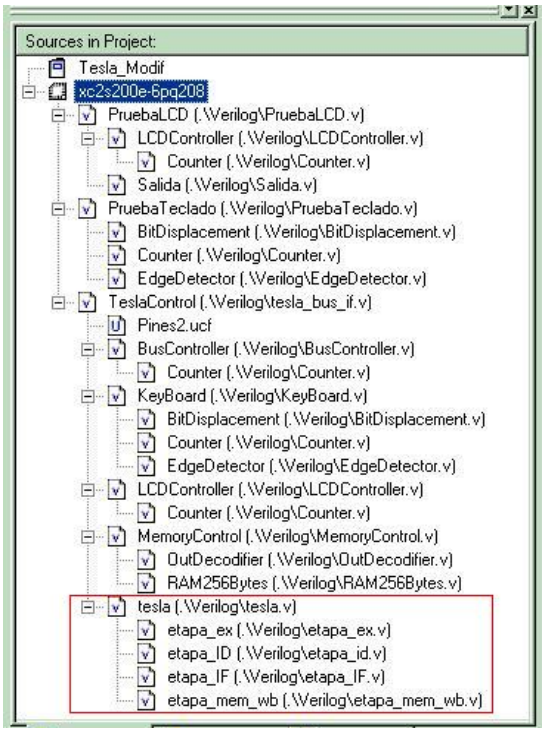

Figura 3 Jerarquía completa de archivos en Verilog. Tomada del ISE® de Xilinx.

3.1.4 Unidad de memoria y realimentación (Mem Write Back)

Para que esta unidad se pueda interconectar con una memoria, fue necesario implementar un registro de lectura y escritura, lo que implica un registro con variables de tercer estado, manejables en Verilog con señales de tipo INOUT. Asimismo, en esta etapa se trabajó en la generación de las unidades extensoras de signo, las cuales se usan para almacenar los datos de manera correcta en la memoria. Es necesario proveer de señales de control de lectura escritura, y controlar un bus bidireccional de datos. Se incorporaron señales que permiten leer palabras de formatos variables y con dirección impar.

3.1.5 Elementos adicionales para pruebas del sistema

Se implementaron las unidades de interfaz con la memoria, así como interfaces con periféricos que permitieran probar el microprocesador en un sistema computacional mínimo. Estas unidades también fueron programadas en Verilog, como se puede apreciar en la jerarquía de archivos de la Figura 3. La Figura 4 muestra el sistema mínimo final. Como punto de prueba inicial para el sistema, se hicieron varias simulaciones en ModelSim® de Mentor Graphics. Dentro de las pruebas de simulación se verificó paso a paso la interconexión de cada una de las etapas dentro del

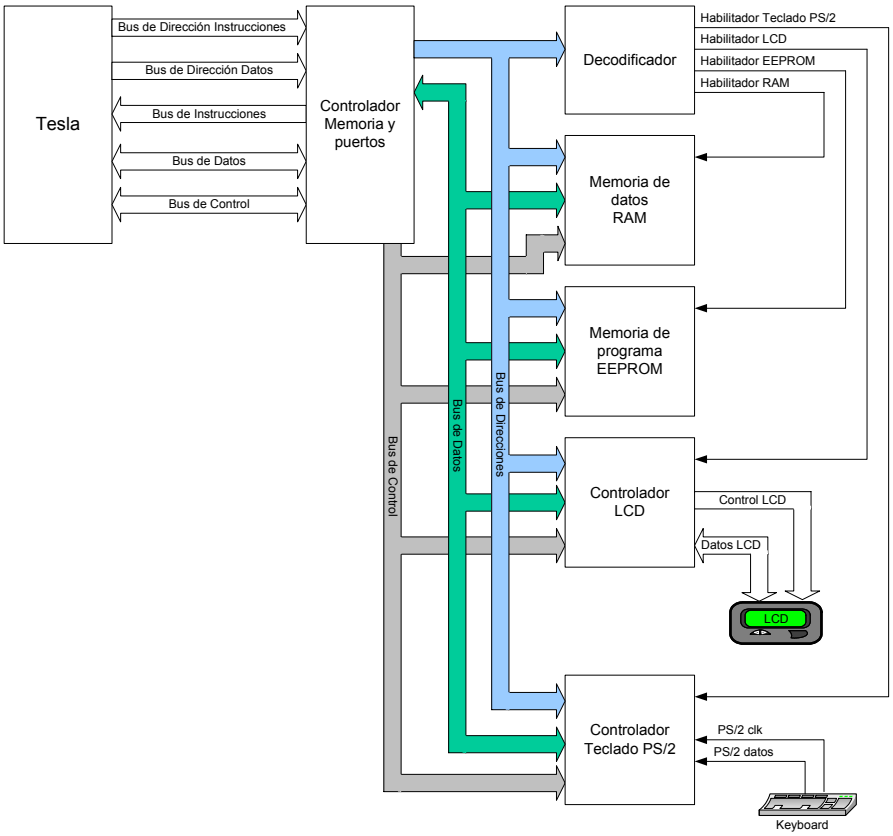

Figura 4. Diagrama de bloques de estructura final de pruebas.

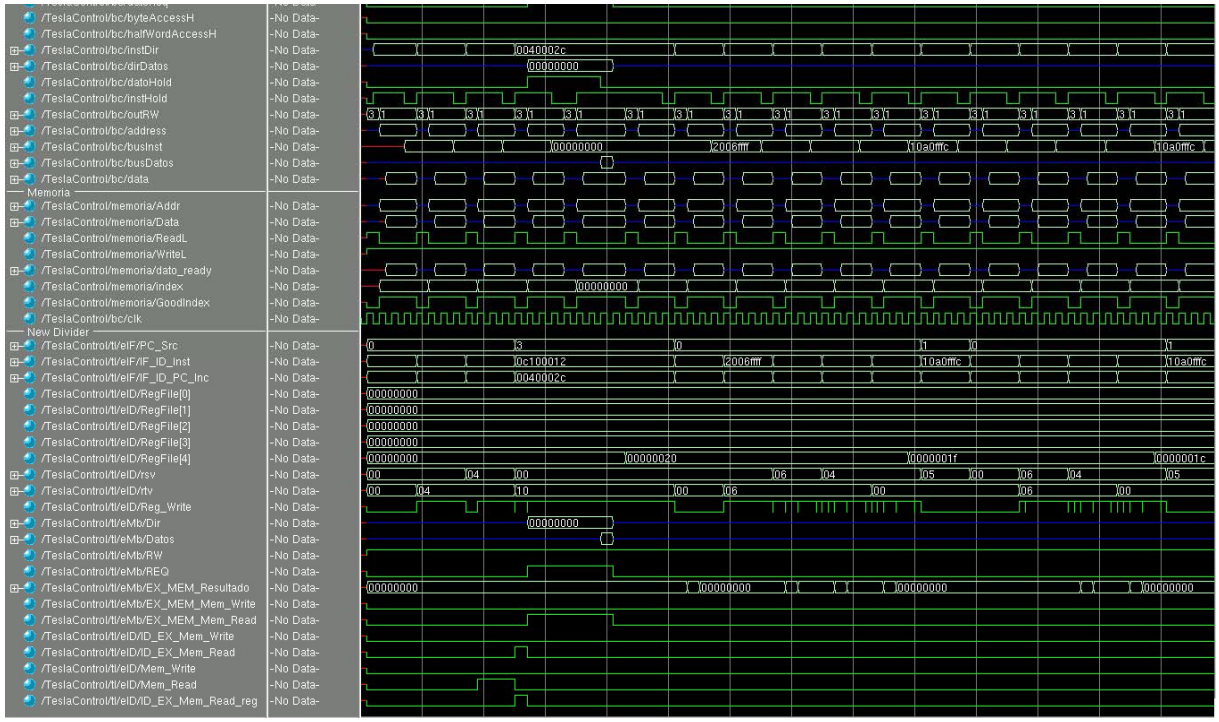

Figura 5. Simulación de un ciclo de ejecución de Tesla por medio del ModelSim® de Mentor Graphics. Se aprecia la segmentación en funcionamiento, la unidad de interfaz de bus y los datos en la RAM implementada dentro de la FPGA.

microprocesador para culminar el proceso de ensamble y prueba. Luego se montó el sistema mínimo con varios bancos de prueba, para así observar la ejecución paso a paso de las instrucciones, el flujo de las operaciones a través de las etapas de la arquitectura y la efectividad de la operación de los sistemas para detección y prevención de riesgos de datos. La Figura 5 muestra un ejemplo de simulación de un ciclo de ejecución de Tesla. El algoritmo implementado en ensamblador es una aproximación entera de una raíz cuadrada. Los lazos en el algoritmo permiten comprobar el correcto funcionamiento tanto de las operaciones como de la detección de riesgos y reenvío de datos. La verificación final se hizo sobre una Spartan 2E. La figura 6 detalla una simulación a nivel de ensamblador del algoritmo probado (utilizando XSPIM), cuyos resultados fueron cotejados entre la simulación funcional RTL y el sistema computacional de pruebas implementado.

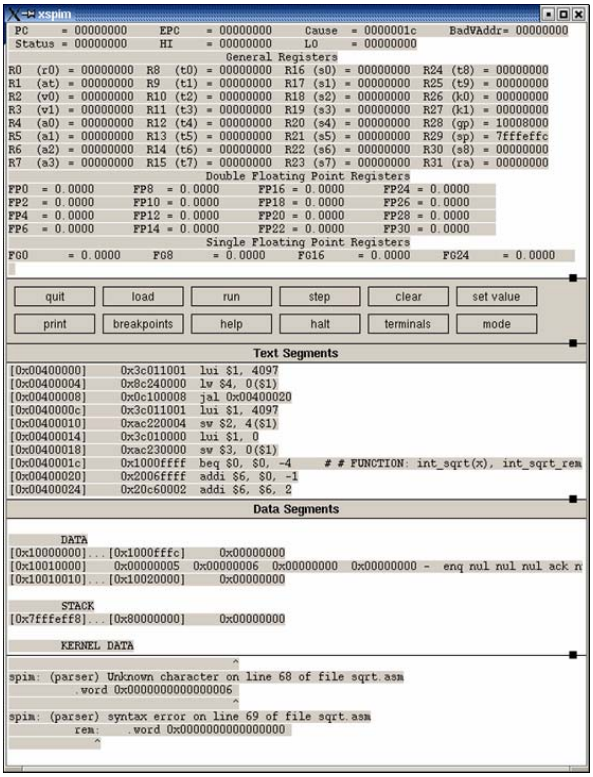

Figura 5. Pruebas en XSPIM del código en ensamblador

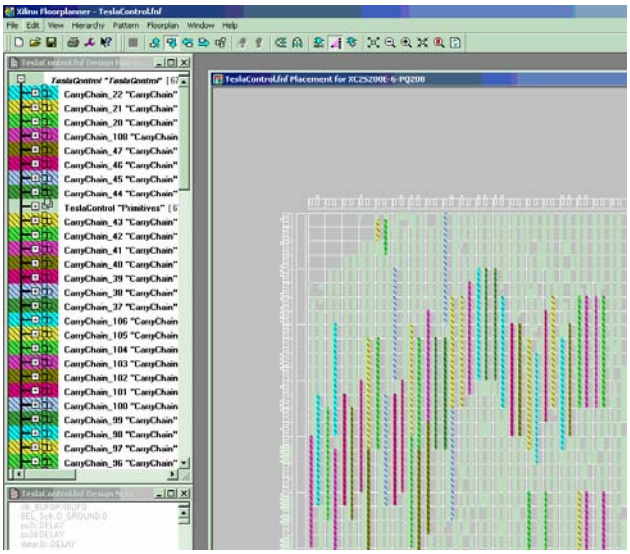

Figura 6. Detalle del plano de piso final en la FPGA

Se utilizó el display LCD de una Digilent Inc. IO2, un teclado PS2 como periférico de entrada y dentro de la FPGA se instanció una memoria RAM. Dentro de las pruebas iniciales, se verificó el despliegue correcto de caracteres del teclado en la pantalla, y luego se cargaron en las EEPROM los programas de prueba, que incluían, aparte del algoritmo para estimar la raíz cuadrada de un número, rutinas de escritura y lectura de LCD y teclado, compilados con herramientas libres para desarrollo de sistemas MIPS®. Todo este sistema fue descargado sobre la tarjeta Spartan2e de Digilent Inc. Los resultados fueron verificados tanto en el LCD como mediante uso de analizador lógico. Se obtuvieron velocidades de 50MHz para la frecuencia del reloj principal, con memorias de programa de 100 ns. En la figura 6 se aprecia un detalle mapeo de piso final de la FPGA. El informe de Place&Route del ISE® reportó los siguientes recursos consumidos:

```
---------------------------
Device utilization summary:
---------------------------
Selected Device : 2s200epq208-6

Number of Slices: 2340 out of 2352 99%
Number of Slice Flip Flops: 1514 out of 4704 32%
Number of 4 input LUTs: 3352 out of 4704 71%
Number of bonded IOBs: 66 out of 146 45%
Number of TBUFs: 224 out of 2352 9%
Number of GCLKs: 1 out of 4 25%
```

4. INSERCIÓN DE ESTRUCTURAS DE TESTEO E INICIO DE BACK-END

Una vez comprobada la funcionalidad de todo el sistema como tal, se procedió a introducir estructuras de prueba en el microprocesador (la intención es construir solo este

como IC, el resto del sistema se dejará sobre la FPGA para pruebas).

Se utilizaron herramientas de DFT sobre el código Verilog sintetizado en Leonardo® para una tecnología comercial de 0.35µm, siguiendo el modelo de cadenas de *scan* recomendados por Mentor Graphics [5]. En la Figura 7 se ejemplifica parte del proceso de verificación de la inserción correcta de las estructuras. Los FFs D estándar son sustituidos por FFs especiales que permiten su conexión en cadena. Se descubrieron errores en las herramientas ATPG que obligaron a obtener soporte de Mentor Graphics [4] y que retrasaron varios meses este trabajo [5].

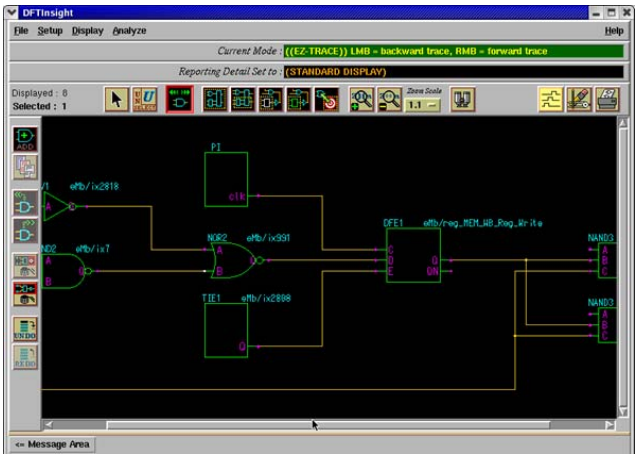

Figura 7. Pruebas de verificación sobre estructuras de DFT insertadas en el código Verilog

Al final, se obtuvo un código Verilog de bajo nivel, mapeado sobre la tecnología adecuada y listo para ser trasladado a las herramientas de back-end, junto con un archivo de vectores para testeo post-fabricación. A partir del código se generó el circuito final por medio de celdas estándar, a través del mapeo del código desde el ICStation® de Mentor Graphics. La Figura 8 muestra parte del inicio del proceso de colocación de los módulos en la herramienta de *layout* Los hilos amarillos definen las conexiones eléctricas por enrutar. Los pads de salida ya han sido colocados. En la actualidad, se está trabajando en el enrutamiento de los bloques externos por medio de herramientas semiautomáticas provistas.

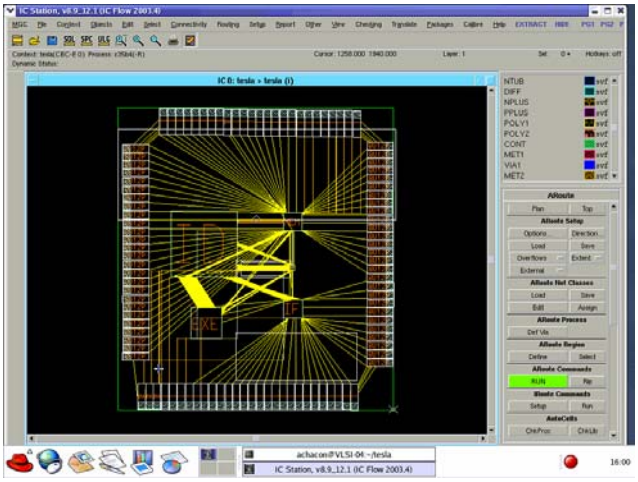

Figura 8. Proceso de colocación de estructuras importadas desde Verilog en la herramienta de layout ICStation de Mentor Graphics. Nótense los pads ya colocados.

Dentro de las pruebas que restan por hacer se encuentran las evaluaciones de tiempos de respuesta, niveles de potencia consumida así como poner a prueba algoritmos y programas cada vez más complejos en el sistema mínimo..

5.CONCLUSIONES

Se ha mostrado el proceso de diseño de una arquitectura RISC de microprocesador y su implementación tanto en dispositivo FPGA como en circuito integrado. Todo el trabajo ha sido realizado por estudiantes de grado de la carrera de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica a lo largo de tres semestres académicos, como parte de sus trabajos de investigación asociados a diferentes cursos y a sus proyectos de graduación. Todo esto forma parte de la intención de esta institución de introducir ya a este nivel la formación en microelectrónica. Se ha concluido que parte del proceso se dificulta por la gran variedad de herramientas disponibles que los estudiantes deben aprender en un tiempo limitado generalmente a un semestre, y las dificultados para migrar entre cada una de ellas en cada una de las etapas.

5. AGRADECIMIENTOS

Deseamos agradecer a Intel y su filial en Costa Rica, Componentes Intel de Costa Rica, por el apoyo económico a esta iniciativa. Este proyecto ha sido financiado también por la Vicerrectoría de Investigación del Instituto Tecnológico de Costa Rica, y la Fundación Tecnológica del ITCR.

6. REFERENCIAS

[1] Hennessy, D. A., y Patterson, J. L. Computer Organization and Design. *The Hardware Software Interface*. 2 ed. Morgan Kauffman, San Francisco, 1998.

[2] R. Pereira, A. Chacón. Informe de avance de proyecto de investigación: "Diseño del primer circuito integrado en el ITCR", octubre, 2003.

[3] J. Bermudez, R. Calvo. Informe. "Depuración e implementación física del microprocesador RISC Tesla", enero, 2004.

[4] Mentor Graphics Documentation. "Scan and ATPG Process Guide", 2003

[5] E. Morales Rodríguez, Informe de proyecto de graduación: "Diseño de las estructuras de prueba para un microprocesador RISC de 32 bit", febrero, 2005.