IMPLEMENTACIÓN MIPS 32-bits PIPELINE CON BUS AXI

Sebastián Matías Segura, Nehuén Berón López, Luis Pablo Seva, Leandro Tozzi
INTI Centro de Micro y Nanoelectrónica del Bicentenario
ssegura@inti.gob.ar

Introducción

En el área de diseño de circuitos integrados del Centro de Micro y Nanoelectrónica del Bicentenario (CMNB), se desarrollan bloques de propiedad intelectual en distintas líneas de trabajo, siendo las aplicaciones para control una de ellas.

Partiendo del diseño previo de un MIPS (*Microprocessor without Interlocked Pipeline Stages*) *Single-cycle* de 32-bit con cinco instrucciones [1] y, ante la necesidad de obtener mayor rendimiento, resultó de interés realizar una mejora para futuras aplicaciones integradas. El mismo se complementó con el desarrollo de una interfaz de comunicación AMBA AXI4 que es un protocolo de bus desarrollado por ARM.

Objetivo

- Diseño de un un bloque de propiedad intelectual (IP) de un microprocesador *MIPS32* de arquitectura *Pipeline* basándose en un diseño previo *Single Cycle.*
- Realizar un bloque de propiedad intelectual que gestione una interfaz AMBA AXI4.
- Desarrollar un bloque de propiedad intelectual de verificación (VIP) para la especificación AMBA AXI4.

Descripción

I) MIPS: Diseño

La arquitectura MIPS Pipeline consiste en dividir el *datapath* en etapas. De esta manera, se pueden procesar partes de distintas instrucciones en simultáneo, a diferencia del *Single-cycle*, el cual solamente puede ejecutar una instrucción entera por ciclo de reloj. Por consiguiente, la latencia no se ve altamente afectada pero si mejora considerablemente el rendimiento del microprocesador, medido en CPI *(Cycles per Instruction*), que es el número de ciclos de reloj necesarios para ejecutar una instrucción en promedio [2]. El período del reloj (Tc) se determina por el camino crítico dado por la lógica del microprocesador, que al dividirla en etapas mediante registros (fig. 1), permite lograr una disminución del mismo y aumentar la frecuencia de funcionamiento.

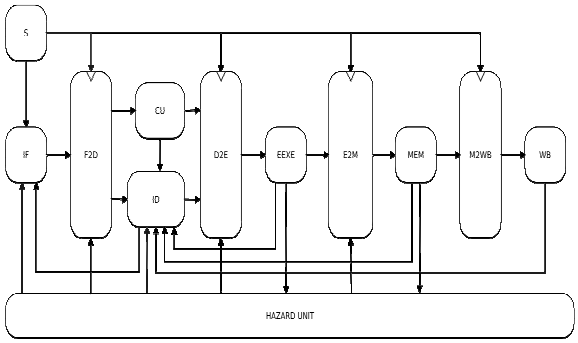

Figura 1: Diagrama de bloques.

El pipeline se divide en cinco etapas:

- *IF (Instruction Fetch) - Búsqueda de la instrucción*. La instrucción se lee de la memoria, usando la dirección determinada por el *Program Counter* (PC) y se almacena en el registro F2D. Luego, al siguiente ciclo de reloj, el PC decide la próxima instrucción a leer.
- *ID (Instruction Decode)- Decodificación de la instrucción*. En esta etapa se interpreta la instrucción, y se realiza la escritura/lectura de la información necesaria del banco de registros junto con operaciones complementarias, como lo es la extensión del signo.
- *EXE (Execution) - Ejecución de la instrucción.* Las operaciones de la ALU, ya sean aritméticas o lógicas, se realizan en esta etapa. El resultado de la operacion se envía al registro E2M.
- *MEM (Memory) - Memoria.* El objetivo de esta etapa es leer y escribir en la memoria de datos. Para eso se obtiene el resultado del registro E2M que se utiliza como dirección. Una señal de control indica si se lee o escribe en memoria. En caso de leer, se graba el dato y la dirección en el registro M2WB.
- *WB (WriteBack) - Escritura.* Esta es la última etapa, en la cual se lee el registro M2WB y se envía a la etapa ID para que los datos sean almacenados en el banco de registros.

Se realizó el trabajo de portar el *diseño Single-Cycle* a uno *Pipeline*, mejorando la perfomance del microprocesador y abarcando el set completo de instrucciones.

II) AXI4: Diseño

La especificación AMBA es un estándar para la comunicación e interconexión entre los bloques funcionales en un *System on a Chip (SoC)*. El mismo facilita el diseño de sistemas con múltiples procesadores y periféricos.

AXI es la tercera generación de AMBA, orientada a sistemas de alta perfomance.

Partiendo del análisis y la interpretación de la especificación, se resolvió que el mejor camino por optar era realizar primeramente el diseño de la interfaz *AXI4 lite*, la cual es una versión simplificada que se adecúa a periféricos que no tengan un gran flujo de datos. Una vez que se obtuvo el mismo correctamente verificado, se procedió a modificarlo para llevarlo a una interfaz AXI4 que acepte el modo transferencia de datos en modo ráfaga.

III) MIPS: Verificación

Se escribieron en *SystemVerilog* tests de unidad para cada componente, a fin de eliminar la mayor cantidad de errores posibles en las primeras etapas del diseño y obtener una métrica de cobertura de código testeado.

Debido a la realización del set completo de instrucciones, se utilizó el simulador MARS MIPS [3], desarrollado por la universidad de Missoury, para escribir programas en assembler a fin de verificar la totalidad del microprocesador. Éstos se llevaron a código máquina y se guardaron en un archivo de texto y luego, mediante un modelo que se comporta como una memoria de instrucciones, se envió el contenido del archivo al microprocesador en forma sincronizada. Finalmente, en el *testbench* de *SystemVerilog*, se verificó el correcto funcionamiento de cada programa.

Para toda la verificación se utilizó la herramienta *QuestaSim.*

IV) AXI4: Verificación

Para el desarrollo de una *VIP* para la interfaz AXI4, se comenzó realizando un *Protocol Checker*, el cual consistió en un módulo que monitorea y verifica el correcto comportamiento de las señales. Luego, mediante el uso del lenguaje *SystemVerilog* se generan estímulos aleatorios que se le aplican al diseño, logrando cuantificar métricas de cobertura de verificación. Se logró una cobertura funcional del cien por ciento.

Resultados

En la figura 2 se pueden apreciar resultados de síntesis, utilizando la herramienta Leonardo Spectrum y la tecnología XH-035 de 350 nm del fabricante XFAB.

	Single-Cycle	*Pipeline*
Frecuencia Maxima	39.2MHz	65.0MHz
Numero De Compuertas	737263	1068605

Figura 2: Resultados de la síntesis

Conclusiones

Se obtuvo una implementación de un microprocesador MIPS32 *Pipeline*, mejorando considerablemente un MIPS32 *Single-cycle* preexistente y cubriendo el set de instrucciones MIPS32 excepto las instrucciones de multiplicación y división. Se logró aumentar la frecuencia de funcionamiento por un factor de 1.65, que dista del factor teórico de 5 ya que la resolución de riesgos del *Pipeline* introduce una penalidad en la métrica CPI, puesto que debe descartar ciertas instrucciones que se están ejecutando. Además, a nivel de compuertas lógicas, las restricciones temporales que implican los registros físicos, tales como el tiempo de propagación de la salida *clkQ* y el tiempo de *setup* intrínseco, no sólo deben cuantificarse en una etapa, sino en todas. Esto aumenta el período del camino crítico. También se destaca que la etapa ID es sustancialmente más lenta respecto del *Single-cycle* debido a que la comparación del salto a realizar y la lectura del banco de registros deben suceder en medio ciclo de reloj.

Respecto al área resultante, la arquitectura *Pipeline* por su naturaleza aumenta el número de registros y además, la cantidad de multiplexores y lógica de control para realizar la correcta prevención de conflictos.

Se pudieron resolver todos los riesgos producidos en la arquitectura *Pipeline* de una manera simple en términos de lógica adicional.

Resultó muy útil la utilización de herramientas como el *MARS MIPS*, la cual aceleró el proceso de verificación de manera considerable, complementando los tests unitarios desarrollados en *SystemVerilog*.

En cuanto a la interfaz de comunicación AMBA AXI4 ahora ya es parte del repositorio de IPs del centro y está lista para ser utilizada en futuros desarrollos, así como también la suite de verificación del protocolo que permite validar la correcta implementación del bus AXI4.

Bibliografía

[1] D. A. P. . J. L. Hannessy, Computer Organization and Design. 500 Sansome Street, San Francisco, CA 94111, USA: Morgan Kaufmann, 2005.

[2] D. M. H. . S. L. Harris, Digital Design and Computer architecture. 225
Wyman Street, Waltham, MA 02451, USA: Morgan Kaufmann, 2013.

[3] M. S. University. Mars (mips assembler and runtime simulator).[Online].Available:http://courses.missouristate.edu/kenvollmar/mars/