Arquitectura de Computadoras

Ingeniería en Computación FCEFyN - UNC
Trabajo Práctico 3 - BIP

Aguerreberry Matthew. Mat.: 93739112
Maero Facundo. Mat.: 38479441

6 de diciembre de 2017

1. Introducción

Para el tercer trabajo práctico de la materia se realizó la implementación en Verilog de un procesador BIP I. Se desarrolló para la placa Nexys 4 DDR, utilizando el software Vivado 2017.2. El procesador permite la ejecución de instrucciones en un ciclo, y se conecta con la UART desarrollada en el práctico número 2, para comunicarse con una PC. Desde esta, es posible visualizar el resultado de las instrucciones: ciclos de clock empleados, y resultado final.

2. Desarrollo

Se definió un módulo Top, que contiene:

- Módulo CPU
- Módulo RAM de Instrucciones
- Módulo RAM de Datos
- Módulo UART
- Módulo Interfaz CPU - UART
- Módulo Clock Wizard

2.1. CPU

El módulo CPU contiene una implementación del procesador BIP I en Verilog. Se descompuso en los siguientes bloques funcionales: control y datapath.

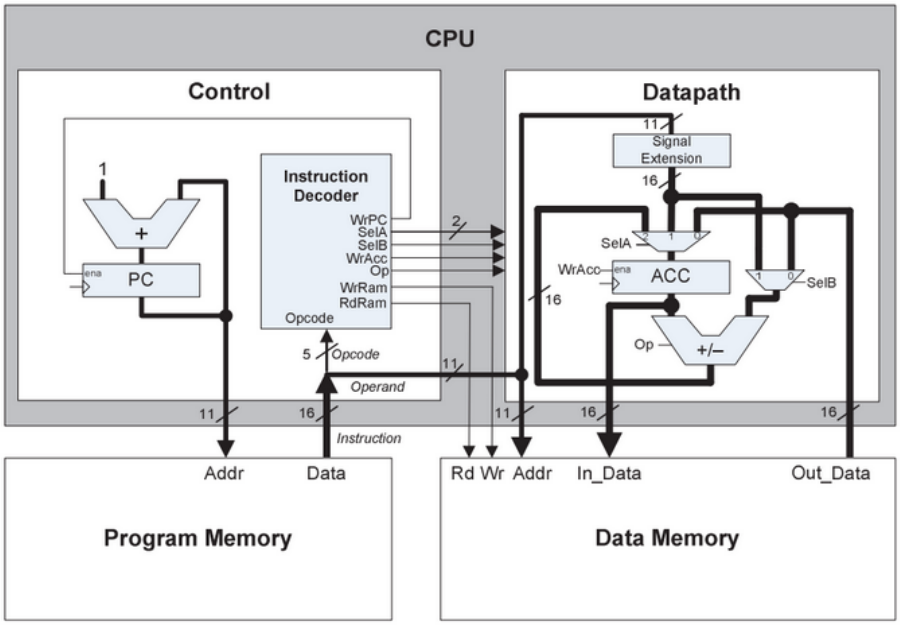

Figura 1: Diagrama de bloques del procesador BIP I

2.1.1. Control

El bloque de control del BIP se encarga de obtener la siguiente instrucción de la memoria de programa, indicada por el valor del PC. Además, posee un circuito combinacional que controla mediante diversas señales, el comportamiento del bloque de datapath. El único elemento controlado por el clock en este bloque es el registro PC.

2.1.2. Datapath

Este bloque maneja el cálculo del resultado de la instrucción en curso. Trabaja accediendo a la memoria de datos según sea necesario, para almacenar un resultado, o leerlo y utilizarlo en una operación. Posee un único registro acumulador, una simple unidad aritmética, que permite realizar sumas y restas, y diversos multiplexores para seleccionar el dato con el que operar. Las señales de control mencionadas anteriormente se utilizan para controlar estos multiplexores, la escritura del registro acumulador, y el comportamiento de la memoria de datos (lectura o escritura).

2.2. RAM de Instrucciones y de Datos

Estos bloques se generaron mediante los templates que provee el entorno de desarrollo. Consisten en memorias de 16 bits, 2048 registros para las instrucciones y 1024 registros para los datos. La memoria de instrucciones fue inicializada con un archivo en texto plano, que contiene las instrucciones en binario utilizadas para probar el funcionamiento del BIP.

Un punto importante a destacar es el empleo del modo **Low Latency** para la memoria de datos. Esto fue necesario ya que, como las instrucciones deben completarse en un ciclo, se encontraron escenarios en los que esto no era posible: Por ejemplo, si una instrucción solicita sumar el valor del acumulador con el guardado en una posición de la memoria de datos, la secuencia de operaciones a realizar es:

1. Leer el opcode de la memoria de programa.
2. Interpretarla y emitir las señales de control.

3. Leer el valor en memoria de datos a sumar.

4. Sumar el acumulador con el valor leído.

5. Guardar el resultado donde corresponda.

Todo el proceso debe realizarse en un ciclo, por requerimiento de diseño del procesador. Pero si se trabaja con un solo flanco no es posible, por lo que las operaciones con la memoria de programa se realizan en un flanco, y las de datos en el otro. Esto permite cumplir con los requisitos planteados sin comprometer el rendimiento del sistema.

2.3. UART

El módulo UART es el desarrollado en el práctico número 2 de la materia. Se utilizan los bloques transmisor, receptor y un generador de baud rate. Al igual que en el anterior trabajo, se opera a 9600 baudios, con palabras de 8 bits, 1 bit de stop y sin paridad.

2.4. ALU

El módulo ALU es el desarrollado en el primer práctico. Es puramente combinacional, con dos entradas para operandos, y una entrada para el operador.

2.5. Interfaz

Para esto se desarrolló un módulo situado entre la UART y el procesador BIP, que consiste en una máquina de estados con las siguientes características:

- Debe enviar por la UART, una vez que las instrucciones lleguen a un HALT, la cantidad de ciclos de clock empleados, y el resultado final en el acumulador.
- Necesita conocer el opcode actual, para saber si se llegó a un HALT o no.
- Necesita conocer el valor del acumulador, para enviarlo por la UART.
- Posee señales de entrada y salida para indicar comienzo y fin de transmisión de un dato.
- Su bus de salida es de 8 bits, que concuerda con el ancho de palabra utilizado por la UART.
- Posee un registro para contabilizar el número de ciclos empleados.

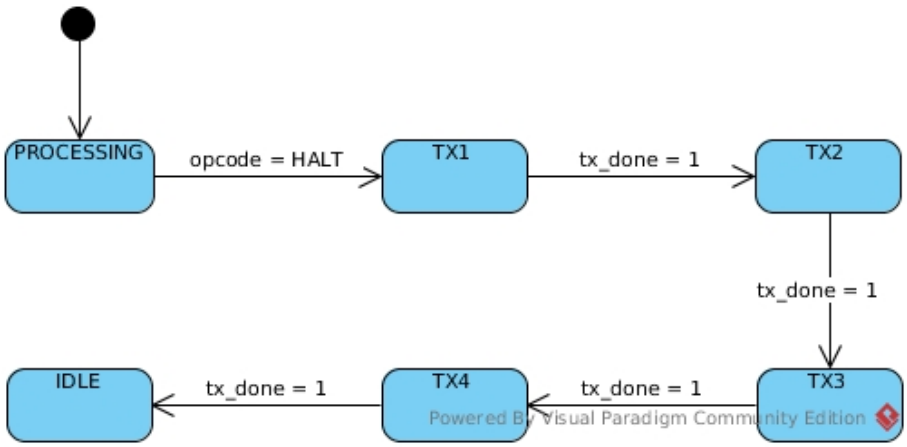

Figura 2: Diagrama de estados del bloque Interfaz

Luego de un reset, la máquina de estados inicia en **PROCESSING**, que modela el intervalo en el que se están ejecutando las instrucciones. En este estado, con cada ciclo de clock se comprueba si se llegó a una instrucción HALT. Si no fue así, se cuenta un ciclo. Una vez en HALT, se pasa a los múltiples estados de transmisión de datos: **TX1, TX2, TX3 Y TX4**, cada uno corresponde

a un byte distinto de los 4 que se deben enviar por UART. Finalizado este proceso, se pasa a un estado **IDLE**, que espera un reset para volver a empezar el proceso.

Con un reset se cambia al estado **PROCESSING**, y se limpia el valor del PC del BIP, para que comience a ejecutar desde la primera instrucción en la memoria de programa.

2.6. Clock Wizard

En las siguientes figuras se ve el valor de Worst Negative Slack **0.925ns**, el cual es el camino que más tiempo demanda. Para el proyecto usamos un clock de 100Mhz.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.925 ns	Worst Hold Slack (WHS):	0.184 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	456	Total Number of Endpoints:	456	Total Number of Endpoints:	174

All user specified timing constraints are met.

Figura 3: Timing Report

Este camino es el determinado entre la ram de instrucciones y la ram de datos.

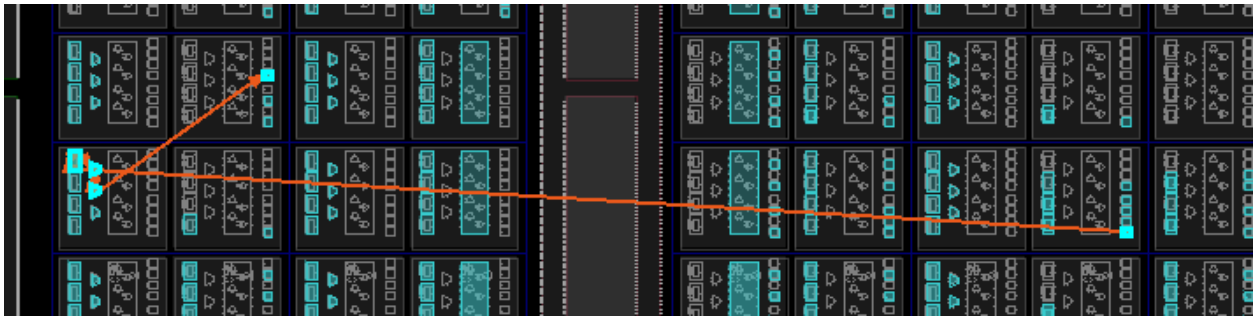

Figura 4: Path critico

3. Test Bench

Se realizaron dos archivos de Test Bench:

3.1. Bloque de Control

Comprueba el correcto funcionamiento del bloque de control, dado un conjunto de instrucciones en la memoria de programa. Permite visualizar la salida del circuito combinacional que maneja el bloque de datapath.

En la captura de pantalla pueden verse los valores de las distintas señales de control luego de interpretadas diversas instrucciones.

3.2. CPU

Permite verificar el correcto funcionamiento del procesador BIP en su totalidad, al interactuar con ambos bloques de memoria. Brinda información sobre el resultado en el acumulador, los valores escritos en memoria de datos, y el flujo de ejecución en general.

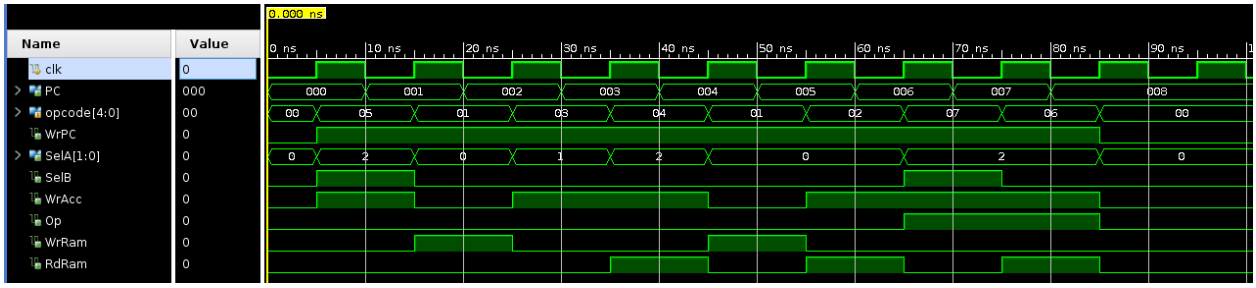

Figura 5: Señales observadas al simular la ejecución del test bench

4. Interfaz con la PC

Para probar el funcionamiento de todo el proyecto, se programó un script en Python que se comunica con la FPGA mediante conexión USB, utilizando la librería PySerial. El script lee el puerto serie 4 veces, para obtener los 2 campos esperados, ambos de 16 bits cada uno: el número de ciclos de clock empleados y el valor final en el acumulador