Laboratorio de Electrónica Digital II
Práctica No. 1: CPU Básica - Parte 1

Profesor Teoría
Felipe Cabarcas Jaramillo. (felipe.cabarcas@udea.edu.co)
Profesores Laboratorio
Andrés Benavides Arévalo (bernardo.benavides@udea.edu.co)
John Byron Buitrago (john.buitrago@udea.edu.co)

Mayo 31, 2022

Fecha de entrega:	Junio 14-17 de 2022
Medio de entrega:	Classroom
Sustentación:	Horario de Laboratorio
Valor Práctica:	8% del curso

1 Introducción

La unidad central de proceso o CPU (por sus siglas en inglés: Central Process Unit) de un computador, se encarga de manipular los datos dados de acuerdo con las instrucciones en la memoria. En su forma más básica, la CPU se compone de dos bloques funcionales esenciales: el *Datapath* (o camino de los datos) y la unidad de Control. En esta práctica de laboratorio, el grupo de trabajo (máximo dos integrantes) llevará a cabo el diseño de la unidad de *Datapath* de una CPU básica. La

descripción del hardware será realizada mediante el uso del lenguaje de descripción de hardware SystemVerilog, mientras que las pruebas de funcionamiento se llevarán a cabo haciendo uso de herramientas de simulación.

2 Objetivo de la Práctica

Diseñar el *Datapath* de una CPU básica junto con la memoria de datos, empleando el lenguaje de descripción de hardware SystemVerilog y herramientas de simulación, capaz de ejecutar operaciones simples de 8-bits sobre los datos disponibles tanto en los registros como en la memoria de datos.

3 Procedimiento

Leer completamente esta guía para planear el diseño del *Datapath* y la memoria de datos. Como primer paso, el grupo de estudiantes realizará una propuesta gráfica usando componentes digitales que describan cada componente. A continuación, procederán a codificar cada componente en SystemVerilog y a simular el código propuesto para verificar su correcto funcionamiento, llevando a cabo las correcciones que sean pertinentes. Posteriormente, conectarán los componentes individuales en un diseño final el cual será descrito en SystemVerilog y simulado con un test de prueba para verificar su correcto funcionamiento. Finalmente, el grupo de trabajo redactará un corto reporte y lo enviará junto con los archivos del proyecto en SystemVerilog. Más detalles sobre la entrega se darán a conocer al final de esta guía. La correspondiente sustentación se realizará en el horario de laboratorio correspondiente en la semana establecida para la entrega.

4 Especificaciones y Funcionamiento del *Datapath*

La Fig. 1 muestra una CPU básica junto con dos memorias: *Instruction Memory*, donde se almacenan las instrucciones a ejecutar y *Data Memory*, donde se almacenan los datos a procesar o ya procesados. En este modelo, la CPU tomará las instrucciones de la *Instruction Memory*, las decodificará dentro de su unidad de Control y, posteriormente, las ejecutará empleando la unidad de *Datapath* y la memoria de datos (Data Memory). En esta práctica, el grupo de trabajo diseñará la unidad de *Datapath*, junto con la memoria de datos (*Data Memory*) como se muestra en la Fig. 2.

Los módulos a diseñar junto con sus características se describen a continuación:

- **Register File** (Banco de Registros): contiene 4 registros de 8-bits cada uno. El banco de registros se emplea para almacenar el resultado de la ALU o el valor leído desde la memoria de datos en cualquiera de los 4 registros. También se emplea para tomar los operandos que empleará la ALU para realizar una operación específica. A continuación se describe la funcionalidad de las entradas y salidas:
  - RA1 y RA2: entradas de 2-bits para escoger los dos registros cuyos valores serán llevados a las salidas RD1 y RD2, respectivamente.

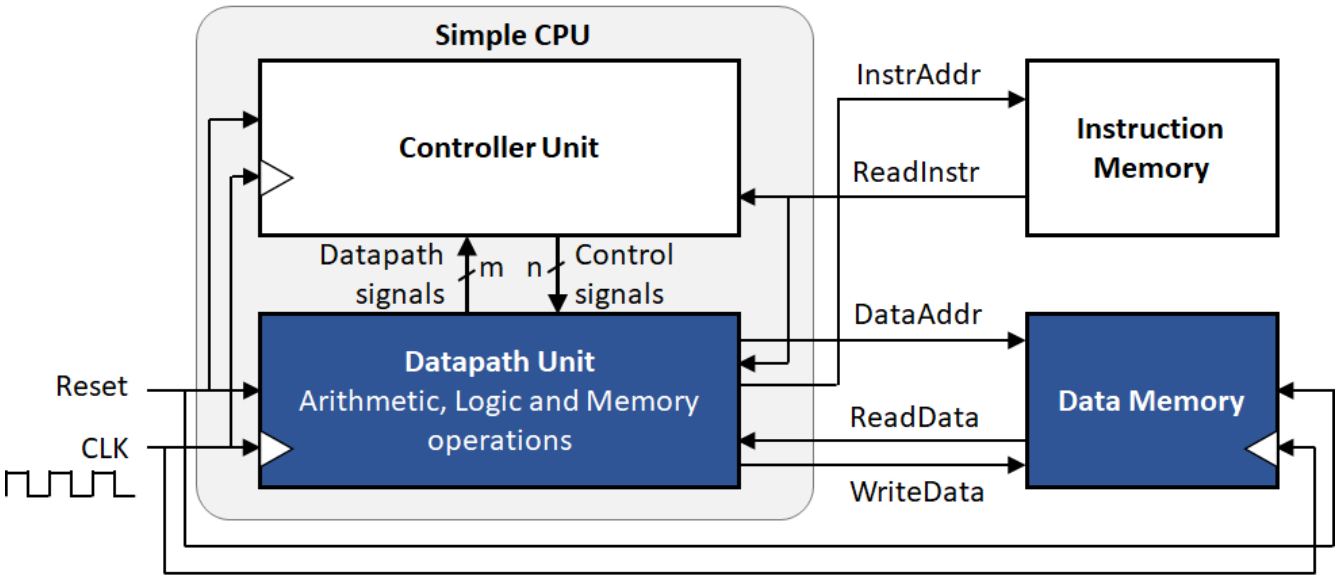

Fig. 1: CPU Simple + Memoria

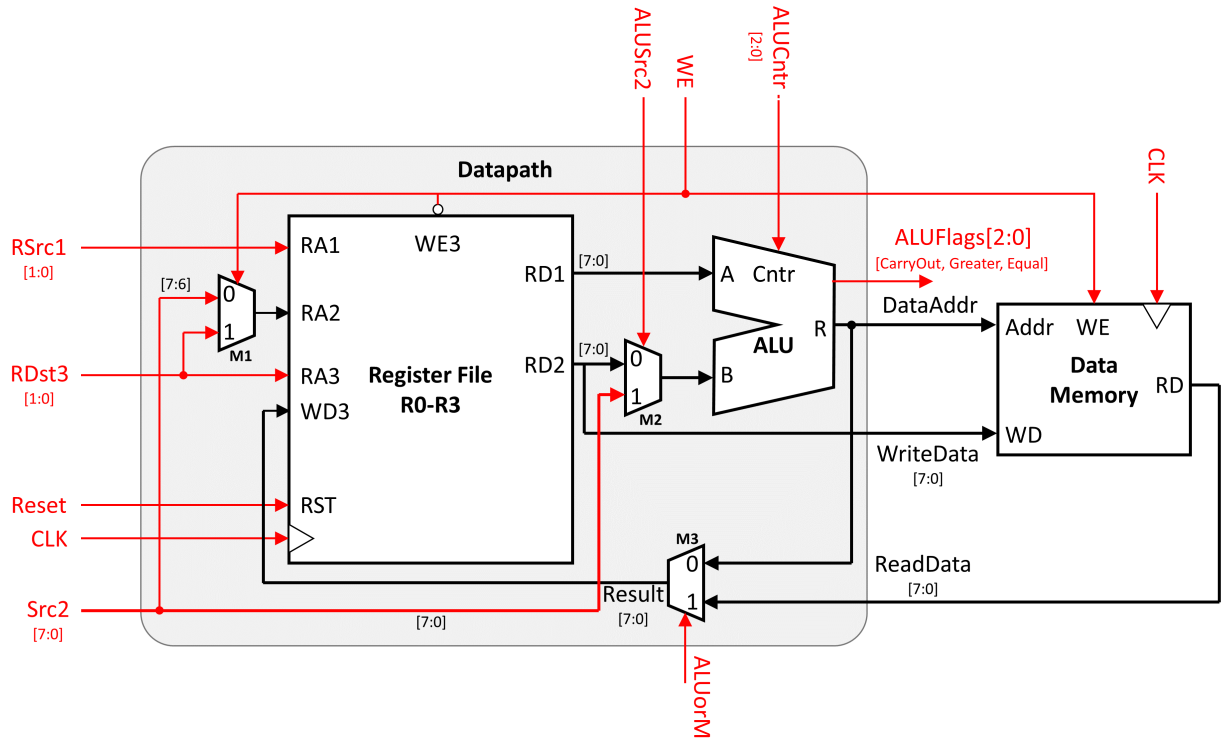

Fig. 2: *Datapath* + Memoria de Datos a diseñar

– RA3: entrada de 2-bits para selecionar el registro donde se escribirá el resultado de la ALU o el valor leído desde la memoria de datos.

- WD3: entrada de 8-bits que contiene el resultado de la ALU o el valor leído desde la memoria de datos y que será almacenado en el registro seleccionado con la entrada RA3 cuando haya un flanco de subida de la señal de reloj.
- WE3: señal de habilitación (enable) para permitir la escritura del valor disponible en WD3 en el registro seleccionado con la entrada RA3 cuando haya un flanco de subida de la señal de reloj.
- RD1 y RD2: salidas de 8-bits que mantienen los valores de los registros seleccionados con RA1 y RA2, respectivamente.
- RST: entrada que, estando en uno, hace que los registros se carguen con ceros.
- CLK: señal de reloj para los registros.

- **ALU** (Unidad Aritmético Lógica): encargada de realizar diversas operaciones lógicas y aritméticas de acuerdo con las líneas de selección *ALUCntr*. A continuación se describe la funcionalidad de las entradas y salidas:
  - A y B: entradas de 8-bits que contienen los valores a operar. La entrada A corresponde al valor del registro seleccionado con la entrada RA1 del Register File. La entrada B puede provenir de dos fuentes distintas:
    * valor del registro seleccionado con la entrada RA2 del Register File (si ALUSrc2 es igual a 0)
    * valor de 8-bits de la entrada Src2 (si ALUSrc2 es igual a 1)
  - R: salida de 8 bits que contiene el resultado de la operación realizada en la ALU.
  - Cntr: entrada de 3-bits que permite escoger la operación a realizar como se muestra en la Tab. 1.

- **Multiplexores**: tres multiplexores se encuentran ubicados dentro de la unidad de *Datapath* para controlar el flujo de los datos.
  - M1: cuando se está llevando a cabo una operación de almacenamiento en la memoria de datos (WE = 1), M1 pasa el valor del registro destino (RDst3) hacia RA2 para que, a través de la salida RD2, se obtenga el valor que irá hacia la memoria de datos para ser almacenado. En caso contrario (WE = 0), M1 pasa el valor de las líneas [7:6] de Src2 hacia RA2, para escoger el valor del registro que servirá como segundo operando de la ALU.
  - M2: escoge el segundo operando para la ALU entre el valor del registro seleccionado con RA2 (ALUSrc2 = 0) o el valor de 8-bits proporcionado en la señal de entrada Src2 (ALUSrc2 = 1).
  - M3: selecciona el valor a almacenarse en el registro seleccionado por RA3 entre el resultado de la ALU (ALUorM = 0) o el valor leído desde la memoria de datos (ALUorM = 1).

Tab. 1: Operaciones disponibles en la ALU

ALUCntr			Operación
0	0	0	R = A AND B
0	0	1	R = NOT B
0	1	0	R = A XOR B
0	1	1	R = A + B
1	0	0	R = A - B
1	0	1	R = B
1	1	0	No implementada
1	1	1	No implementada

- **Data Memory** (Memoria de Datos): bloque de 256 registros o direcciones de memoria, de 8-bits cada uno, que conforman una memoria de acceso aleatorio. Cualquiera de los registros o direcciones de memoria se puede seleccionar mediante la entrada Addr para realizar una lectura o escritura. Cuando WE = 0, el valor del registro seleccionado con Addr es llevado a la salida RD. En caso contrario (WE = 1), el valor disponible en la entrada WD será escrito en el registro seleccionado mediante Addr cuando se produzca un flanco de subida de la señal de reloj.

El resumen de las operaciones que puede realizar la unidad de Datapath se muestran en la Fig. 3.

5 Diseño

El módulo principal de su diseño en SystemVerilog debe tener la siguiente declaración:

```
module DP_DM_yy(input logic CLK,
                input logic RST,
                input logic [1:0] RSrc1,
                input logic [7:0] Src2,
                input logic [1:0] RDst3,
                input logic WE,
                input logic ALUorM,
                input logic [2:0] ALUCntr,
                input logic ALUSrc2,
                output logic [7:0] DataAddr,
                output logic [7:0] ReadData,
                output logic [7:0] WriteData,
);
```

donde, yy (DP_DM_yy) corresponde a las iniciales de los integrantes del grupo.
Guarde su módulo en un archivo de nombre DP_DM_yy.sv. El reloj o señal **CLK** que entra al controlador será de 1khz. A continuación describa en SystemVerilog su módulo de manera que

DP Inst	ALUCntr [2:0]			ALUorM	RDst3 [1:0]	WE	RSrc1 [1:0]	ALUSrc2	Src2 [7:6]	Src2 [5:0]						Operación (Flanco de subida de CLK)
AND	0	0	0	0	RegD	0	Reg1	0	Reg2	X	X	X	X	X	X	RegD ← Reg1 & Reg2
AND	0	0	0	0	RegD	0	Reg1	1	8-bit Value							RegD ← Reg1 & Value
NOT	0	0	1	0	RegD	0	XX	0	Reg2	X	X	X	X	X	X	RegD ← ~Reg2
NOT	0	0	1	0	RegD	0	XX	1	8-bit Value							RegD ← ~Value
XOR	0	1	0	0	RegD	0	Reg1	0	Reg2	X	X	X	X	X	X	RegD ← Reg1 ^ Reg2
XOR	0	1	0	0	RegD	0	Reg1	1	8-bit Value							RegD ← Reg1 ^ Value
ADD	0	1	1	0	RegD	0	Reg1	0	Reg2	X	X	X	X	X	X	RegD ← Reg1 + Reg2
ADD	0	1	1	0	RegD	0	Reg1	1	8-bit Value							RegD ← Reg1 + Value
SUB	1	0	0	0	RegD	0	Reg1	0	Reg2	X	X	X	X	X	X	RegD ← Reg1 - Reg2
SUB	1	0	0	0	RegD	0	Reg1	1	8-bit Value							RegD ← Reg1 - Value
MOV	1	0	1	0	RegD	0	XX	0	Reg2	X	X	X	X	X	X	RegD ← Reg2
MOV	1	0	1	0	RegD	0	XX	1	Immediate Value 8-Bits							RegD ← Value
LDR	0	1	1	1	RegD	0	Reg1	1	Immediate Value 8-Bits							RegD ← DATA_MEM[Reg1 + Value]
STR	0	1	1	X	RegD	1	Reg1	1	Immediate Value 8-Bits							DATA_MEM[Reg1 + Value] ← RegD

Fig. 3: Funciones a realizar con la unidad de *Datapath* y la Memoria de Datos

realice la funcionalidad pedida y que pueda ser sintetizado y simulado. Incluya sub-módulos para tener un diseño modular.

6 Simulación

Cree un archivo denominado testbench_yy.sv que demuestre, de manera convincente, que el *Datapath* y la memoria de datos que usted ha descrito en SystemVerilog desarrolla toda la funcionalidad requerida de manera correcta, de acuerdo con la Fig. 3. Emplee ModelSIM o cualquier otra herramienta de simulación para esta actividad.

Es posible que la primera vez usted obtenga errores en el módulo de su diseño o en el testbench (banco de pruebas). Interprete de manera apropiada los mensajes que recibe de la herramienta de simulación y corrija los problemas que se han presentado.

7 Entrega

El grupo de trabajo deberá escribir un reporte que contenga los siguientes elementos:

a. Una breve introducción del diseño del *Datapath* y la Memoria de Datos.

b. Tabla de instrucciones mostrando un ejemplo del binario por cada instrucción disponible para la CPU (Ver Fig. 3)

c. Diagrama digital de cada componente de su diseño con una breve explicación. Se aceptan imágenes escaneadas o fotografías siempre que sean claras.

d. Resultados de la simulación mostrando que su diseño realiza correctamente la funcionalidad descrita en esta guía.

e. Conclusiones.

Crear un archivo comprimido que incluya el reporte y los archivos importantes de su proyecto en Xilinx Vivado o Quartus Prime como se describe a continuación:

a. Reporte: archivo con extensión .pdf

b. Archivos fuente: DP_DM_yy.sv.sv, testbench_yy.sv, otros archivos .sv que haya creado.

El nombre del archivo comprimido debe tener el siguiente formato:
p1_primerapellidointegrante1_primerapellidointegrante2_horariolaboratorio.zip.
Ejemplo: si el primer apellido de ambos integrantes es **Benavides** y **Buitrago**, respectivamente, y el laboratorio es el Martes 9-12, entonces el archivo debe ser nombrado: *p1_benavides_buitrago_m9-12.zip*.

8 Evaluación

La evaluación de la práctica se divide en tres partes: funcionamiento (40%), sustentación (40%) y reporte (20%). La nota del funcionamiento se asigna por igual a todos los integrantes del grupo de trabajo (máximo dos personas por equipo), mientras que la nota de sustentación es individual. En caso de que un estudiante obtenga una nota inferior a 3.0 en la sustentación, la nota final de la práctica para el estudiante en mención será la que obtuvo en la sustentación, es decir, no se tendrá en cuenta el funcionamiento en el cálculo.

Cada grupo de trabajo, con un máximo de dos integrantes, deberá sustentar la práctica en un tiempo de 20 minutos, 10 minutos para revisar la simulación y 10 minutos para preguntas. Durante la sustentación, el profesor hará un par de preguntas a cada uno de los integrantes del grupo de trabajo.

9 Referencias

a. Xilinx Vivado WebPack
`https://www.xilinx.com/support/download.html`

b. Quartus Prime Lite Edition
`https://fpgasoftware.intel.com/?edition=lite`

c. Ambiente de Simulación Online
`https://www.edaplayground.com/home`

d. Tutorial de SystemVerilog
`https://verilogguide.readthedocs.io/en/latest/`