# Laboratorio 2 electrónica digital 2

1st Camilo Álvarez Muñoz *Facultad de ingeniería*
*Universidad de Antioquia*
Medellín, Colombia
camilo.alvarezm@udea.edu.co
2nd Juan Camilo Arteaga Ibarra *Facultad de ingeniería*
*Universidad de Antioquia*
Medellín, Colombia
camilo.arteaga1@udea.edu.co

***Resumen*—En este laboratorio se diseñó e implementó una unidad central de proceso o CPU mediante el lenguaje de descripción de hardware SystemVerilog y el sistema de desarrollo DE10-Lite. Se diseñaron dos bloques esenciales conocidos como la unidad de control y datapath, las cuales permitieron el desarrollo de una CPU básica capaz de ejecutar un programa de instrucciones, realizando una operación aritmética básica, procesada por cada uno de los componentes que hacen parte de las unidades mencionadas.**

***Index Terms*—CPU, ALU, registros, memoria, instrucciones.**

## I. Introducción

Este documento contiene el reporte de laboratorio sobre la CPU simple realizada, mediante la unidad de control, datapath, la memoria de instrucciones y la memoria de datos, estas dos últimas, para este caso están incluidas en la unidad de control y en el datapath respectivamente. La CPU se encarga de realizar el procesamiento de datos de entrada mediante el uso de periféricos de la FPGA DE10-lite, para la cual, a partir de una función aritmética descrita como un paso a en la memoria de instrucciones, se leen, escriben y procesan los datos introducidos por el usuario para mostrar el resultado de realizar la operación $2x + 6y - 5z$, en hexadecimal, haciendo uso de unos displays siete segmentos de la tarjeta de desarrollo.

## II. Procedimiento experimental y resultados

*Unidad datapath:*

Esta unidad es la encargada de los procesos aritméticos de la CPU simple que desarrollamos en esta práctica, compuesta por una ALU, un archivo de registros y tres MUX, aparte de esto hay un componente llamada Data Memory que se encarga de guardar los datos que se procesan. Para este práctica esta memoria de datos se incluyó dentro de la unidad de Datapath.

El datapath recibe unas señales desde la unidad de control, las cuales permiten el control del tráfico de datos para el procesamiento de cada uno de los módulos que componen el datapath.

Al incluir la memoria de datos en el datapath, para el diagrama de bloques obtenido mediante el software ”Quartus Prime”, esta se ubicará dentro de la unidad, de modo que las conexiones son internas y las externas aparecen de color rojo.

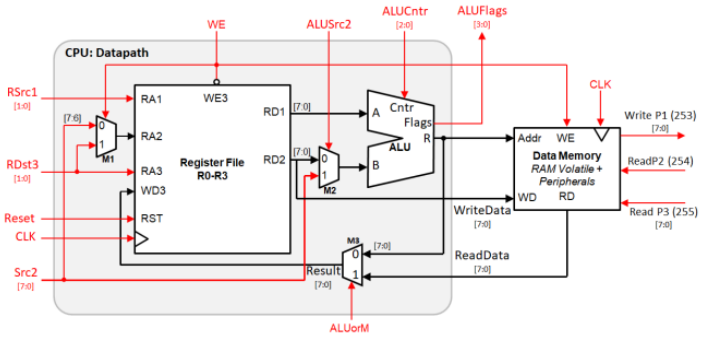


Figura 1: diagrama de conexiones propuesto para el datapath.

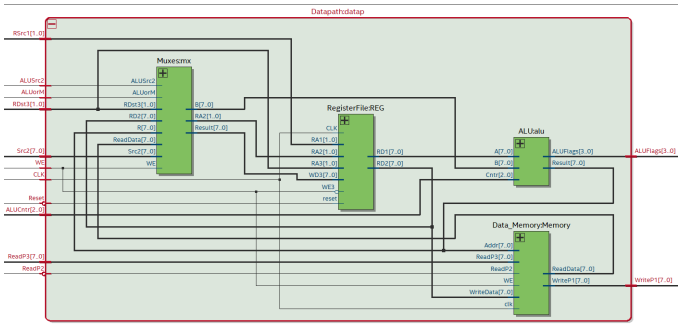


Figura 2: diagrama de conexiones obtenido para el datapath.

La memoria es un bloque de 256 registros, cada uno de 8-bits, destinados a guardar y leer datos que llegan al módulo. Esta controlada por los flancos de subida que provienen del reloj de la FPGA. Para su uso, una señal llamada Addr indica la dirección de memoria a utilizar y mediante la señal WE (Write Enable), se indica si es una escritura o lectura de datos almacenados en la dirección de memoria asignada. También, con las últimas 3 direcciones de memoria se asocian los 3 periféricos que se usaron durante la práctica, con la 253 los decodificadores siete segmentos, es decir, una señal de salida (solo se realiza escritura), con la 254 el pulsador y con la

255 los switchs, las cuales son señales de entrada (solo se realiza lectura).

Para el laboratorio, se desarrolló cada uno de los MUX como un MUX único, el cual recibe las señales de control y entrada, permitiendo el flujo de datos según sea necesario.

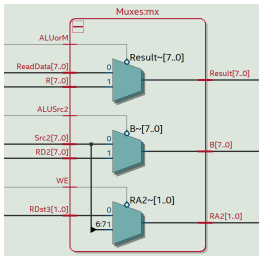


Figura 3: diagrama de conexiones de los MUX.

Luego está la ALU, la cual realiza las operaciones que llegan como instrucción desde la unidad de control para el procesamiento de los datos. Las dos funciones no implementadas, realmente son una lectura y escritura en la unidad de memoria, por lo que al momento de procesar las instrucciones, la ALU realiza una suma de los registros que entran a ella, para ello la unidad de control entrega la instrucción correcta a la ALU.

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

Tabla I: operaciones de la ALU.

Por último está el register file", el cual contiene 4 registros de 8-bits cada uno, en los que se almacena el dato leído desde la memoria de datos o el resultado de la ALU, siento controlado por la señal de reloj proveniente de la FPGA y con una instrucción de escritura de los datos que se encuentran en la señal WD3 en el registro descrito por la señal RA3, si la señal WE (Write Enable) se encuentra inactiva. Además, recibe una señal reset, la cual se utiliza en la práctica para asignar ceros a cada uno de los registros el banco de registros cuando se desee.

Para esta unidad se realizó un test-bench, haciendo uso del simulador del software, en el que se analizaron múltiples combinaciones sobre las entradas y salidas de los módulos que componen el datapath, garantizando el buen funcionamiento a medida que se iba completando su diseño.

*Unidad de control:*

La unidad de control es la responsable de controlar el flujo de datos y operaciones en la CPU; esta unidad lee y codifica las instrucciones del programa almacenado en la memoria de instrucciones (Instruction memory) para así mediante una máquina de estados coordinar el orden en que la CPU hará manejo de la memoria externa, leerá los periféricos de la FPGA o realizara una operación en la ALU de la unidad de Datapath. Esta unidad es compuesta por 3 submódulos conocidos como "FSM-UNIT", Çontroller UnitInstruction Memory".

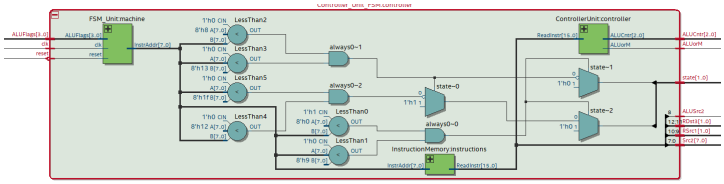


Figura 4: diagrama de conexiones unidad de control.

- Instruction Memory (Memoria de instrucciones): Como es indicado por su nombre, este submódulo contiene una memoria con 256 registros de 16-bits cada uno. En estos registros o direcciones de memorias se guardaron cada una de las instrucciones necesarias para controlar el Datapath y así realizar correctamente el programa de prueba: $f(x, y, z) = 2x + 6y - 5z$. Los valores $x$, $y$ y $z$ son números binarios de 8-bits que son ingresados mediante 8 switches de la FPGA y para ingresar en el correcto orden estos valores, es necesario usar un botón que al ser presionado indique a la memoria de datos cuándo y cómo tomar el número de los switchs y asignarlo a alguno de los registros para así realizar la multiplicación de estas variables. En la tabla se puede observar el total de instrucciones necesarias para realizar el programa de prueba:

- FSM-UNIT: Este submódulo contiene una máquina de estados que es la encargada de coordinar el flujo de ejecución de las 32 instrucciones almacenadas en la memoria, para realizar esto se realizó un contador que estará en constante aumento mediante el reloj CLK, según el valor en el que se encuentre el contador, se enviara una dirección que será leída por la memoria de instrucciones para así ejecutar la instrucción deseada. ya que los valores $x$, $y$ y $z$ deben ser ingresados sucesivamente con ayuda de los switches y botón de la FPGA es necesario de las instrucciones número 4, 12 y 23 que verificaran si el botón se encuentra pulsado mediante la señal ALUFLAGS entregada por la ALU para así ingresar a la memoria el número que se encuentre en los switches, en caso de que el botón no este presionado, el contador se devolverá 3 instrucciones y verificará nuevamente el estado del botón. Además de esto, en la máquina de estados fue necesario agregar un

| # | Function | Instruction | Binary | Hexa |
|---|---|---|---|---|
| 0 | R0←0x00 | MOV R0, #0x00 | 101_00_00_1_00000000 | A100 |
| 1 | R1←MEM[255] | LDR R1, [R0, #255] | 110_01_00_1_11111110 | C9FF |
| 2 | MEM[253]←R1 | STR R1,[R0,#253] | 111_01_00_1_11111100 | E9FD |
| 3 | R2←MEM[254] | LDR R2,[R0,#254] | 110_10_00_1_11111110 | D1FE |
| 4 | R2←R2 | MOV R2,R2 | 101_10_00_0_10000000 | B080 |
| 5 | R3←R1+R1 | ADD R3,R1,R1 | 000_11_01_0_01000000 | 1A40 |
| 6 | MEM[0]←R3 | STR R3,[R0,#0] | 111_11_00_1_00000000 | F900 |
| 7 | R2←MEM[254] | LDR R2,[R0,#254] | 110_10_00_1_11111110 | D1FE |
| 8 | R2←R2 | MOV R2,R2 | 101_10_00_0_10000000 | B080 |
| 9 | R1←MEM[255] | LDR R1[R0,#255] | 110_01_00_1_11111110 | C9FF |
| 10 | MEM[253]←R1 | STR R1,[R0,#253] | 111_01_00_1_11111100 | E9FD |
| 11 | R2←MEM[254] | LDR R2,[R0,#254] | 110_10_00_1_11111110 | D1FE |
| 12 | R2←R2 | MOV R2,R2 | 101_10_00_0_10000000 | B080 |
| 13 | R2←0x06 | MOV R2,#0x06 | 101_10_00_1_00000110 | B106 |
| 14 | R0←R0+R1 | ADD R0,R0,R1 | 000_00_00_0_01000000 | 0040 |
| 15 | R2←R2-0x01 | SUB R2,R2,#0x01 | 001_10_10_1_00000001 | 3501 |
| 16 | MEM[1]←R0 | STR R0,[R2,#1] | 111_00_10_1_00000001 | E501 |
| 17 | R0←0x00 | MOV R0,#0x00 | 101_00_00_1_00000000 | A100 |
| 18 | R2←MEM[254] | LDR R2,[R0,#254] | 110_10_00_1_11111110 | D1FE |
| 19 | R2←R2 | MOV R2,R2 | 101_10_00_0_10000000 | B080 |
| 20 | R1←MEM[255] | LDR R1[R0,#255] | 110_01_00_1_11111110 | C9FF |
| 21 | MEM[253]←R1 | STR R1,[R0,#253] | 111_01_00_1_11111100 | E9FD |
| 22 | R2←MEM[254] | LDR R2,[R0,#254] | 110_10_00_1_11111110 | D1FE |
| 23 | R2←R2 | MOV R2,R2 | 101_10_00_0_10000000 | B080 |
| 24 | R2←0x05 | MOV R2,#0x05 | 101_10_00_1_00000101 | B105 |
| 25 | R0←R0+R1 | ADD R0,R0,R1 | 000_00_00_0_01000000 | 0040 |
| 26 | R2←R2-0x01 | SUB R2,R2,#0x01 | 001_10_10_1_00000001 | 3501 |
| 27 | R1←MEM[1] | LDR R1[R2,#1] | 110_01_11_1_00000001 | CF01 |
| 28 | R3←R1-R0 | SUB R3,R1,R0 | 001_11_01_0_00000000 | 3A00 |
| 29 | R1←MEM[0] | LDR R1[R2,#0] | 110_01_10_1_00000000 | CD00 |
| 30 | R0←R1+R3 | ADD R0,R1,R3 | 000_00_01_0_11000000 | 02C0 |
| 31 | MEM[253]←R0 | STR R0,[R2,#253] | 111_00_10_1_11111101 | E5FD |

Figura 5: Lista de instrucciones del programa de prueba.

sincronizador que verificará mediante las ALUFLAGS si el botón se sigue presionando inmediatamente después de haber ingresado un valor en los switches, esta verificación es necesario hacerla para evitar realizar todas las instrucciones del programa en un solo presionado de botón debido a que el contador de la máquina de estados aumenta con el reloj CLK que trabaja a 50MHz y en una situación real un humano tardaría algunos segundos presionando y soltando el botón para ingresar cada uno de los 3 números; este sincronizador se realiza con las instrucciones 8 y 19. En la máquina de estados también se usaron las instrucciones 15 y 26 para realizar la multiplicación de los números con ayuda de un ciclo for que sumara sucesivamente el número a multiplicar hasta que una variable auxiliar sea 0, en caso de que el ciclo no haya terminado, la máquina de estados regresara al estado inmediatamente anterior.

- ControllerUnit: Este submódulo actuará como un decodificador que se encargará de leer el valor hexadecimal de cada instrucción que será entregado por la memoria de instrucciones para así determinar el valor de las señales WE, ALUorM y ALUCntr. En la memoria de instrucciones se usan las operaciones LDR Y STR se encargan respectivamente de cargar un dato de memoria sobre un registro y de cargar un dato de un registro a la memoria; estas dos instrucciones tienen asignados los códigos de operación 110 y 111, pero al analizar la ALU implementada en el Datapath estos dos códigos se indican como no implementados, por lo cual es necesario en este submódulo modificar la señal ALUCntr para indicar que las operaciones LDR y STR actúen como una suma, para esta suma se necesita un registro base y un número índice para así sumar estos dos y obtener un valor que actuará como dirección de la memoria de datos. En el caso de que se tenga una instrucción LDR, WE será 0 y ALUorM será 1, esto significa que se tomará un dato de la memoria y se escribirá en el register file. En el caso de que se tenga una instrucción STR, WE será 1 y ALUorM será 0, esto significa que un registro será enviado y escrito en la memoria.

## III. Discusión de resultados

Durante la práctica de laboratorio se presentaron múltiples dificultades al momento de realizar la unidad de control, las cuales fueron posibles de solucionar al hacer uso de la simulación del software ”Quartus Prim”, de manera que con la ayuda del profesor y con un análisis minucioso de cada una de las señales, se siguió el proceso partiendo desde un resultado general hacia cada uno de los módulos para observar si las salidas que se esperaban eran las correctas o no lo eran. Descubriendo así errores en la escritura del código sobre lo que se deseaba, como lo fue el caso de una indexación y la mala interpretación de las ”ALUFlags”, las cuales indicaban en la lógica de las instrucciones, mediante una máquina de estados, cuál era la instrucción a ejecutar, cumpliendo con los bucles de sumas para procesar cada uno de los términos que acompañan la ecuación a implementar con la CPU básica diseñada.

Por otro lado, también hubo dificultades en la definición de las instrucciones a ejecutar, ya que al intentar simplificar y disminuir el número de instrucciones, se cometieron errores que generaban resultados erróneos a los esperados, por lo que se decidió no escatimar en la cantidad de instrucciones utilizadas, para así poder asegurar que se cumplía con cada una de las etapas del procesamiento de las instrucciones, para ello, se realizó un proceso de paso a paso, implementando cada uno de los términos que componen la ecuación, mostrando los datos ingresados y el resultado de la multiplicación (sumas) realizadas por el procesador a partir de las instrucciones determinadas, primero el $2x$, luego el $6y$, el $5z$ y por último las adiciones y sustracciones de cada una de las variables con su respectivo factor de multiplicación.

A pesar de las dificultades presentadas, se logró realizar la práctica de manera satisfactoria, obteniendo un resultado correcto, cumpliendo con la función otorgada para la práctica de laboratorio.

Se realizó la simulación del test-bench, con cada uno de los parámetros de la ecuación con entrada 1, en la imagen no se logra apreciar completamente la simulación, sin embargo es posible deducir que el resultado es 3.

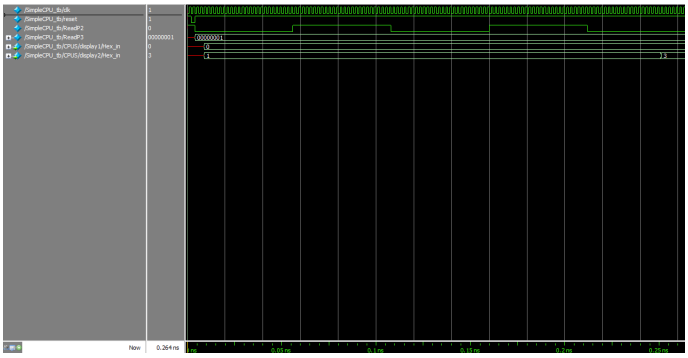

Figura 6: simulación de la CPU simple.

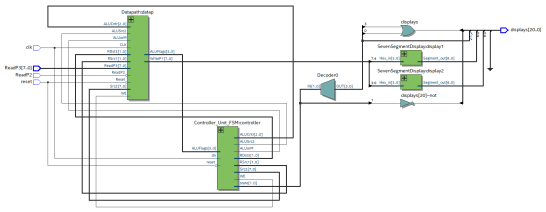

Figura 7: diagrama de bloques de la CPU simple.

## IV. Conclusiones

- Para el correcto desarrollo de la implementación de un sistema con varios módulos como lo es la CPU, es necesario realizar simulaciones del comportamiento de cada uno de los componentes a utilizar conforme se realiza la programación, ya que al momento de implementar todo el sistema en una FPGA es imposible detectar los distintos errores que pueden generarse al tener tantos componentes haciendo parte del funcionamiento.

- Para la realización de este laboratorio fueron necesarias aproximadamente 30 horas de trabajo.

- Es destacable el tiempo y el trabajo que conlleva realizar una CPU, sabiendo que la trabaja durante el laboratorio es una simple, el desarrollo de una CPU común, que se utiliza en muchos de los dispositivos electrónicos, tiene una gran cantidad de procesos que requieren de una labor muy precisa, permitiendo obtener una eficiencia idónea en la implementación de las instrucciones para los que está creada.

## Referencias

[1] S. L. Harris and D. Harris, “Digital design and computer architecture: Arm edition,” 2015.