Laboratorio de Circuitos Digitales 24 de Abril del 2022
IE-0424 Jordi Louzao von Breymann - B53993
Laboratorio #2: Anteproyecto Dualok Fonseca - B42629

1. Ejercicios

1.1. Investigue qué instrucciones forman parte de la extensión RV32M y describa cada una de estas instrucciones brevemente.

La extensión de RISCV32M le agrega 8 instrucciones que le otorgan al procesador la capacidad de multiplicar y dividir. [1]

1.1.1. Extensión estándar RV32M para multiplicación y división de enteros

Cuadro 1: Tabla de instrucciones de la extensión M de RISCV32

Instrucción	Nombre	Descripción
MUL rd,rs1,rs2	Multiply	Multiplica sin signo rs1 con rs2 y almacena la parte baja del resultado en rd
MULH rd,rs1,rs2	Multiply High Signed Signed	Multiplica con signo rs1 con rs2 y almacena la parte alta del resultado en rd
MULHSU rd,rs1,rs2	Multiply High Signed Unsigned	Multiplica rs1 con signo, con rs2 sin signo y almacena la parte alta del resultado en rd
MULHU rd,rs1,rs2	Multiply High Unsigned Unsigned	Multiplica con signo rs1 con rs2 y almacena la parte alta del resultado en rd
DIV rd,rs1,rs2	Divide Signed	Divide con signo rs1 con rs2 y almacena el resultado de la división en rd
DIVU rd,rs1,rs2	Divide Unsigned	Divide sin signo rs1 con rs2 y almacena el resultado de la división en rd
REM rd,rs1,rs2	Remainder Signed	Almacena el residuo de la división sin signo de rs1 y rs2 Es decir rd = rs1 modulo rs2
REMU rd,rs1,rs2	Remainder Unsigned	Almacena el residuo de la división sin signo de rs1 y rs2 Es decir rd = rs1 modulo rs2

1.2. Describa: ¿Cómo se definen y cómo se usan los parámetros de instancias en módulos implementados en el lenguaje Verilog?

Los parámetros son declarados con la palabra **parameter** justo después del nombre del modulo y antes de declarar los puertos de entrada y salida del modulo comenzando con un signo de #, usualmente se escribe el nombre del parámetro con mayúscula.

```
module modulo1 #(parameter X = 0, parameter Y = 0, parameter Z = 0)
```

Estos parámetros pueden ser redefinidos al instanciar el modulo, comúnmente dentro de un tesbench, utilizando la palabra **defparam** o se pueden pasar los nuevos parámetros dentro de #() al instanciar el modulo. [2]

```
modulo1 #(.X(1), xY(1)) inst([lista de puertos]);
defparam inst.Z = 1;
```

En el ejemplo anterior, al instanciar el modulo1 con el nombre inst se cambian los parámetros X y Y a 1, luego utilizando defparam se cambia el valor de Z a 1.

1.3. Investigue qué es una LUT (look up table).

Una LUT(look up table) es una técnica comúnmente utilizada para acelerar procesos numéricos en aplicaciones con requerimientos demandantes de temporización, consiste en un arreglo que contiene una serie de resultados precalculados para operaciones complejas, de modo que estos resultados se obtienen mas rápido accediendo a su valor en el arreglo que calculándolo desde cero. Se suelen usar LUTs en procesamiento digital de señales(DSP). Se usan especialmente porque las operaciones de punto flotante, en especial las multiplicaciones tienden a demandar mucho tiempo de procesamiento.[3]

1.4. Investigue qué es el alineamiento de memoria y para qué sirve.

El alineamiento de memoria se refiere a la forma en que algunos datos están acomodados en ciertas direcciones de memoria. Especifícamele se refiere a que posiciones de memoria se puede almacenar un dato dependiendo de su tamaño. Por ejemplo, un word debe de ser almacenado en una dirección que sea divisible por 4, ya que un word corresponde a 4 bytes. Esto se hace para que fácilmente se pueda determinar cual es la siguiente posición lógica dependiendo del tipo de dato siendo almacenado (byte, half-word, word, double-word, etc...). Típicamente los procesadores únicamente pueden leer datos de word en word. Si un word esta desalineado esto implica que el procesador tendría que leer los dos words alineados que contienen las partes del word desalineado, y juntarlas tal que se obtenga el word desalineado. Este procesamiento adicional afecta la velocidad de manera considerable.[4]

1.5. Investigue acerca de los siguientes términos y realice una breve descripción del significado y uso de cada uno:

1.5.1. Síntesis de lógica

La síntesis de lógica es el proceso automatizado de convertir una descripción en alto nivel, por ejemplo código en Verilog, a un diseño de compuertas optimizado. La utilidad de esto es evidente, provee un proceso comparativamente fácil y rápido para generar diseños de circuitos digitales complejos a partir de un código en alto nivel. Obvia la necesidad de utilizar mapas de Karnaugh, proceso que consume mucho tiempo y es sujeto a errores, y le facilita al diseñador un circuito final optimizado que puede implementar.[5]

1.5.2. RTL (register transfer level)

El RTL es una abstracción para definir las partes digitales de un diseño. Típicamente es una parte constitutiva de lenguajes de descripción de hardware (HDL) como Verilog. El nivel RTL tiene tres partes primarias, los registros que contienen información de los estados, lógica combinacional que define como se reacciona ante entradas, y relojes que controlan la temporización de cuando cambian los estados.[6]

1.5.3. Place and route

Place and route se refiere a un proceso de tomar un netlist generado por síntesis de lógica, cargarlo a un FPGA, extraer los componentes y redes del archivo, poner los distintos componentes

en el FPGA, y interconectarlos utilizando las conexiones del FPGA. De esta manera se pueda cargar algún programa o lógica a un FPGA y conectar los componentes para poder realizar pruebas sin necesidad de manualmente realizar todas las conexiones y buscar los componentes específicos.[7]

1.5.4. Floorplan

El floorplan se refiere a la configuración de diferentes elementos en un chip para maximizar algún parámetro. Típicamente para FPGAs como los trabajados con Xilinx el diseño con menor área tiende a ser el que tiene mejor desempeño. Esto se debe a que un diseño en donde se reduce el área requiere necesariamente reducir la cantidad de recursos de enrutamiento. Esto también resulta en menores distancias de interconexión, señales que llegan más rápido a su destino final, y mayor velocidad al realizar el proceso de place and route. Típicamente el floorplan se divide entre secciones automatizadas y secciones manualmente definidas. Las secciones de lógica con path de datos son las que más se benefician del control manual, estas siendo secciones con procesamiento en paralelo utilizando sumadores, restadores, contadores, registros, y multiplexores. Mientras tanto, secciones de lógica aleatoria, maquinas de estado, y lógica no estructurada se pueden automatizar en su posición sin mayor perdida de desempeño.[8]

1.5.5. FPGA

Un FPGA es un *Field Programmable Gate Array* por sus siglas en ingles. Es decir es una unidad programable en el campo de arreglos de compuertas lógicas. Más específicamente, un FPGA es un dispositivo semiconductor que esta basado alrededor de una matriz de bloques de lógica configurables, CLBs por sus siglas en ingles. Estos están conectados a través de interconexiones programables. Estos dispositivos son programables después de ser manufacturados para cumplir una variedad de funciones dependiendo de su programación. Los FPGAs son ampliamente utilizados en una variedad de industrias por su flexibilidad para prototipado de ASICs, circuitos integrados con aplicaciones especificas, y también para control de una variedad de dispositivos en industrias desde aeroespacial hasta médica.[9]

1.5.6. Slices de FPGAs

Un slice es un subcomponente de un CLB de un FPGA. En un FPGA de la arquitectura Xilinx serie 7 se tienen dos tipos de slices: SLICEM y SLICEL. El SLICEL exclusivamente lidia con operaciones lógicas, mientras que le SLICEM es configurable para que sus LUTs funcionen como shift registers o memoria de datos. Cada slice contiene cuatro LUTs, ocho flip-flops, una red de lógica de acarreo, y tres tipos de multiplexores. Los elementos de los slices funciona para implementar la lógica diseñada que se carga al FPGA.[10]

1.6. Investigue cuál es la FPGA que tiene la tarjeta Nexys 4 DDR (nombre y número de parte) e investigue acerca del número de celdas lógicas y slices que contiene, y el número de pines de entrada y salida.

La tarjeta Nexys 4 DDR que utilizamos en el curso utiliza un FPGA Artix XC7A100T-CSG324 de Xilinx. Hay un total de 99000 celdas lógicas con 15850 slices lógicos, cada uno con cuatro LUTs de 6 entradas y ocho flip-flops. Para I/O se tienen 16 switches, 4 botones, 16 LEDs, 2 LEDs tricolor, 2 pantallas de siete segmentos de cuatro dígitos, un sensor de temperatura, un micrófono, salida de audio con PWM, salida VGA de gráficos, host de USB, Ethernet, microSD, 4 puertos de Pmod, y un UART.[11]

Referencias

[1] *The RISC-V Instruction Set Manual*. SiFive Inc. 2017. URL: https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf.

[2] ChipVerify. *Verilog parameters*. 2015. URL: https://www.chipverify.com/verilog/verilog-parameters.

[3] Carlos Moreno. *Look-Up Tables (LUT) Operations in C++*. 2015. URL: https://cal-linux.com/tutorials/LUT.html.

[4] Andrew Chin y Elizabeth Rodger. *Alignment and Memory*. 1997. URL: https://www.cs.umd.edu/~meesh/cmsc411/website/projects/outer/memory/align.htm.

[5] Sunil Sahoo. *Logic Synthesis Basics For FPGA*. 2020. URL: https://semiengineering.com/logic-synthesis-basics-for-fpga/.

[6] Semiconductor Engineering. *RTL (Register Transfer Level)*. 2021. URL: https://semiengineering.com/knowledge_centers/eda-design/definitions/register-transfer-level/.

[7] Douglas L. Perry. «VHDL: Programming By Example». En: 4.ª ed. 2002. Cap. 16. URL: https://www.globalspec.com/reference/80307/203279/chapter-16-place-and-route.

[8] Fliptronics. *FPGA Floorplanning (1 of 1)*. 2002. URL: http://www.fliptronics.com/floorplanning1.html.

[9] Xilinx. *What is an FPGA?* 2022. URL: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html.

[10] Nate Eastland. *FPGA – Configurable Logic Block*. 2021. URL: https://digilent.com/blog/fpga-configurable-logic-block/.

[11] Digilent. *Nexys4 DDR™ FPGA Board Reference Manual*. Digilent. 2014. URL: https://www.xilinx.com/content/dam/xilinx/support/documents/university/XUP%20Boards/XUPNexys4DDR/documentation/Nexys4-DDR_rm.pdf.