Organización y Arquitectura de Computadoras
Practica 4: Unidad Aritmético Lógica

Profesor: José Galaviz Casas
Ayudante de Laboratorio: Ricardo Enrique Pérez Villanueva

20 de marzo de 2023

1. Objetivos

1.1. Generales

El alumno practicara el proceso de diseño de circuitos combinacionales.

1.2. Particulares

Al finalizar la practica el alumno estará familiarizado con:

- El diseño modular de los circuitos combinacionales.
- Los componentes básicos de una unidad aritmético lógica, ası como el diseño de la misma.

2. Requerimientos

2.1. Conocimientos Previos:

- Funciones de conmutación.
- Minimización de funciones de conmutación por medio de manipulación algebraica y mapas de Karnaugh.
- Los componentes básicos del diseño de circuitos combinacionales: transistores y compuertas AND, OR y NOT.
- Las funciones de una unidad aritmético lógica. Se puede consultar los temas en [**Mano**] y [**Patterson**].

2.2. Tiempo de realización sugerido: 5 horas

2.3. Numero de colaboradoras: Parejas (Equipos de 2 personas)

2.4. Software a utilizar.

- Java Runtime Environment versión 5 o superior.
- El paquete Logisim.

3. Planteamiento

Un componente elemental de un procesador es la unidad aritmético lógica o ALU (siglas en ingles de Arithmetic Logic Unit), como su nombre lo indica es la encargada de realizar las operaciones aritméticas y lógicas de una computadora. En esta practica se diseñara y simulara en Logisim un sumador y restador de 8 bits y en el lenguaje de programación C, una función que simule el funcionamiento de una ALU de 32 bits que realice las siguientes operaciones:

- Las operaciones aritméticas de adición y sustracción.

- Las operaciones lógicas bit a bit de disyunción y conjunción.
- Comparar si las entradas son iguales, o menor que.

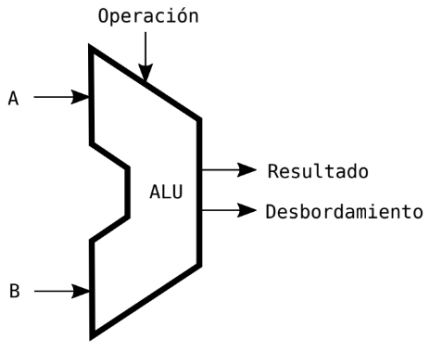

Figura 1: Unidad aritmético lógica.

4. Desarrollo:

4.1. Sumador

Para crear una sumador y restador de 8 bits aprovecharemos la capacidad del diseño modular de los circuitos combinacionales y comenzaremos diseñando un sumador de 1 bit, posteriormente integraremos 8 sumadores de 1 bit para obtener una de 8 bits y finalmente agregaremos otros circuitos para obtener el restador.

4.1.1. Sumador de 1 bit.

Nuestro sumador de 1 bit sera un sumador completo o full adder, este contara con las entradas A, B y C_i para los operandos y el acarreo de entrada respectivamente y las salidas O y C_{i+1} para el resultado de la operacion y el acarreo de salida.

4.1.2. Sumador de 8 bits.

Obtenemos el sumador de 8 bits conectando ocho sumadores de 1 bit de la siguiente forma: la salida C_{i+1} del sumador n se conecta a la entrada C_i del sumador n + 1. Ası cada A_n, B_n y O_n corresponderán al n-esimo bit del operando A, el operando B y la salida Resultado, comenzando con el bit menos significativo. Queda sin conexión la entrada C_i del primer sumador, la del bit menos significativo, y la salida C_{i+1} del ultimo, la del bit mas significativo; la primera nos sera de utilidad mas adelante en el desarrollo de la sustracción y la segunda nos sirve para detectar el desbordamiento aritmético, la cual solo se deberá activar cuando se realiza la adición.

4.1.3. Sustracción

Primero adoptaremos la siguiente convención: para representar los inversos aditivos en nuestro sumador, usaremos la representación de complemento a dos. Entonces en el diseño del circuito que realizara la sustracción, podemos aprovechar los circuitos que ya tenemos, esto es, para obtener $A - B$ sumaremos al minuendo A el inverso aditivo del sustraendo B. Gracias a la convención adoptada, para obtener el inverso aditivo de un numero, debemos invertir cada bit de la entrada B y sumarle 1, por lo que para obtener las sustracción, la ALU debe de llevar a cabo la operación $A + \hat{B} + 1$, en donde $\hat{B}$ es la entrada B con todos sus bits negados.

4.2. ALU de 32 bits

Se escribirá una función en el lenguaje de programación que recibirá tres argumentos:

- **opA y opB** - Cadenas de tamaño 32 formadas por 0 y 1, las cuales representaran los operandos en binario.
- **op** - Una cadena de tamaño 3 formada por 0 y 1, representando la operación que llevara a cabo la ALU, los códigos de operación se muestran en la tabla

Código operación	Operación
000	AND
001	OR
010	Adición
011	Sustracción
100	Igualdad
101	Menor que

Tabla 1: Códigos de operación.

La salida sera una cadena de tamaño 32 formada por 0 y 1, la cual representara el resultado de la operación solicitada, en particular, para las salidas que representan TRUE y FALSE (las comparaciones), la salida debe ser 31 ceros y un 1, (00000000000000000000000000000001) para TRUE y 32 ceros para FALSE. (00000000000000000000000000000000)

5. Procedimiento

Deberás entregar un archivo de Logisim con las soluciones de los ejercicios, un archivo de código fuente escrito en el lenguaje de programación C con la función que simula el funcionamiento de una ALU y un documento PDF con las respuesta a las preguntas planteadas. Para la simulación de Logisim solo puedes hacer uso de compuertas lógicas **AND, OR, NOT y XOR**, multiplexores, separadores y pines de entrada y salida. Recuerda etiquetar las entradas y salidas de cada uno de los subcircuitos.

6. Ejercicios

1. Da una breve introducción a que son las ALUs en tu reporte.
2. Desarrolla un sumador completo de 1 bit.
3. Desarrolla un sumador de 8 bits descrito en la sección 4.1.2 utilizando el circuito del ejercicio anterior.
4. Desarrolla un sumador y restador de 8 bits utilizando el circuito del ejercicio anterior y agregando los componentes necesarios para la sustracción descrita en la sección 4.1.3. Se tiene que agregar un multiplexor para seleccionar la operación de salida, 0 para la adición y 1 para sustracción.
5. En un subcircuito de tu elección, coloca como un componente el sumador y restador final del ejercicio anterior con pines de entradas y salida necesarios con el fin de llevar a cabo la simulación.
6. Escribe la función con los requerimientos solicitados en la sección 4.2, la función debe realizar la aritmética y lógica en binario.

7. Preguntas

1. ¿Quien propuso por primera vez el diseño de una ALU? ¿Para que computador se propuso?
2. ¿Es que es VHDL? ¿Como se ve un full-adder en VHDL?
3. ¿Que operaciones aritméticas y lógicas son básicas para un procesador? Justifica tu respuesta.

4. El diseño utilizado para realizar la adición resulta ser ineficiente, ¿por que? ¿Que tipo de sumador resulta ser mas eficiente?[1]

5. Bajo este diseño, en la ALU se calculan todas las operaciones de forma simultanea pero solo se entrega un resultado, ¿se realiza trabajo inútil? ¿Toma tiempo adicional? ¿Cual es el costo?

6. ¿Cuantas operaciones mas podemos agregar al diseño de esta ALU? ¿Que tendrıamos que modificar para realizar mas operaciones?

[1] Olvide darlo en clase, como pista, investiguen que es un half-adder.