Organización y Arquitectura de Computadoras
2018-1
Práctica 4: Unidad Aritmético Lógica

Profesor: José Galaviz Casas
Ayudante de laboratorio: Luis Soto Martinez

1. Objetivos

Generales:

- El alumno practicará el proceso de diseño de circuitos combinacionales.

Particulares:

Al finalizar la práctica el alumno estará familiarizado con:

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

2. Requisitos

- **Conocimientos previos:**
  - Funciones de conmutación.
  - Minimización de funciones de conmutación por medio de manipulación algebráica 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 pueden consultar los temas en [**Mano**] y [**Patterson**].
- **Tiempo de realización sugerido:**
  5 horas.
- **Número de colaboradores:**
  Individual.

- **Software a utilizar:**
  - *Java Runtime Environment* versión 5 o superior.
  - El paquete *Logisim* [**Logisim**].

3. Planteamiento

Un componente elemental de un procesador es la unidad aritmético lógica o ALU (siglas en inglés de *Arithmetic Logic Unit*), como su nombre lo indica es la encargada de realizar las operaciones aritméticas y lógicas de una computadora. En esta práctica se diseñara y simulará 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 qué.

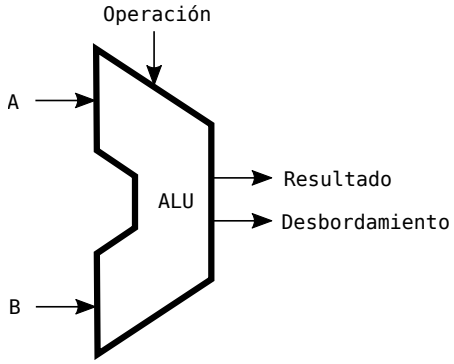

Figura 1: Unidad aritmético lógica.

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 será un sumador completo o *full adder*, este contará con las entradas A, B y C_i para los operandos y el acarreo de entreada respectivamente y las salidas O y C_{i+1} para el resultado de la operación y el acarreo de salida.

4.1.2. Sumador de 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 $n+1$. Así cada A_n, B_n y S_n corresponderán al n-ésimo 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 último, la del bit más significativo; la primera nos será de utlidad más adelante en el desarrollo de la sustracción y la segunda nos sirve para detectar el desbordamiento aritmético, la cual sólo 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 realizará la sustracción, podemos aprovechar los circuitos que ya tenemos, ésto 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 número, 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 $A + \bar{B} + 1$, en donde $\bar{B}$ 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:

1. *opA* y *opB* - Cadenas de tamaño 32 formadas por 0 y 1, las cuales representarán los operandos en binario.

2. *op* - Una cadena de tamaño 3 formada por 0 y 1, representando la operación que llevará a cabo la ALU, los códigos de operación se muestran en la tabla 1.

La salida será una cadena de tamaño 32 formada por 0 y 1, la cual representará el resultado de la operación solicitada.

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

Tabla 1: Códigos de operación.

5. Procedimento

Deberás entregar un archivo de *Logisim* con las soluciones de los ejercicios, un archivo de código fuente escriro en el lenguaje de progamación C con la función que simula el funcionamiento de una ALU y un documenton PDF con las respuesta a las preguntas planteadas.

Para la simulación de Logisim sólo puedes hacer uso de compuertas lógicas **AND**, **OR** y **NOT**, multiplexores, separadores y pines de entrada y salida. Recuerda etiquetar las entradas y salidas de cada uno de los subcircuitos.

6. Ejercicios

1. Desarrolla un sumador completo de 1 bit.
2. Desarrolla un sumador de 8 bits descrito en la sección 4.1.2 utilizando el circuito del ejercicio anterior.
3. Desarrolla un sumador y restador de 8 bits utilizando el circuito del ejercicio anterior y agregando los componentes necesarios para la sustacción descrita en la sección 4.1.3. Se tiene que agregar un multiplexor para selecciónar la operación de salida, 0 para la adición y 1 para sustracción.
4. En un subcircuito 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.
5. Escribe la función con los requerimienros solicitados en la sección 4.2, la función debe realizar la aritmética y lógica en binario.

7. Preguntas

1. ¿Qué operaciones aritméticas y lógicas son básicas para un procesador? Justifica tu respuesta.
2. El diseño utilizado para realizar la adición resulta ser ineficiente, ¿por qué? ¿Qué tipo de sumador resulta ser más eficiente?

3. ¿Cuántas operaciones más podemos agregar al diseño de esta ALU? ¿Qué tendríamos que modificar para realizar más operaciones?

8. Bibliografía