Laboratorio 3: Diseño Modular ALU

Kenneth Hernández, Marco Herrera, and Jasson Rodríguez

Ingeniería en Computadores, Instituto Tecnológico de Costa Rica

Agosto 2019

1. Comprensión del problema

Las operaciones lógicas y aritméticas corresponden a una parte esencial para el funcionamiento de circuitos digitales desde los más simples hasta un procesador completo. El diseño de estas operaciones de forma separada, implica limitaciones de costo y tamaño; además, estos circuitos son dependientes del tamaño de sus operandos; es decir, representan poca escalabilidad y ajuste.

2. Investigación

Con el fin de comprender, probar y optimizar el comportamiento del diseño en cuestión, es de suma importancia conocer parámetros característicos de los circuitos combinacionales; Harris y Harris [1] afirman que el tiempo de propagación t_{pd} por sus siglas en inglés, es el tiempo máximo desde que una entrada cambia, hasta que las salidas alcanzan su valor final. Por otra parte, el tiempo de contaminación t_{cd} es el tiempo mínimo desde que un cambio en la entrada, hasta que cualquier salida empieza a cambiar su valor. Ambos términos se aprecian con mayor facilidad en la figura 2

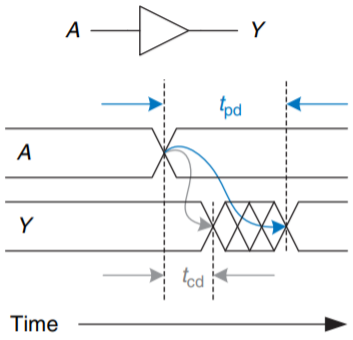


Figura 1: Tiempos de propagación y contaminación

Se sabe que toda ALU tiene bits de entrada que serán operados de diversas maneras y bits de selección que le permiten decidir cuál será la operación a realizar, normalmente dividido entre aritméticas y lógicas, entonces suponiendo que se tienen 8 operaciones aritméticas y 8 lógicas, estas con entradas A y B como bits a operar, y tenemos 4 bits de selección llamados C D E F,queremos escoger las operaciones aritméticas o lógicas usando C, por lo que tomando en cuenta que Ar es cualquier operación aritmética que involucre A y B, y que Log es cualquier operación lógica, tendríamos una tabla como la siguiente:

C	D	E	F	OP
0	0	0	0	Ar1
0	0	0	1	Ar2
0	0	1	0	Ar3
0	0	1	1	Ar4
0	1	0	0	Ar5
0	1	0	1	Ar6
0	1	1	0	Ar7
0	1	1	1	Ar8
1	0	0	0	Log1
1	0	0	1	Log2
1	0	1	0	Log3
1	0	1	1	Log4
1	1	0	0	Log5
1	1	0	1	Log6
1	1	1	0	Log7
1	1	1	1	Log8

Tabla 1: Tabla de verdad de AND

Lo cual genera el siguiente diagrama:

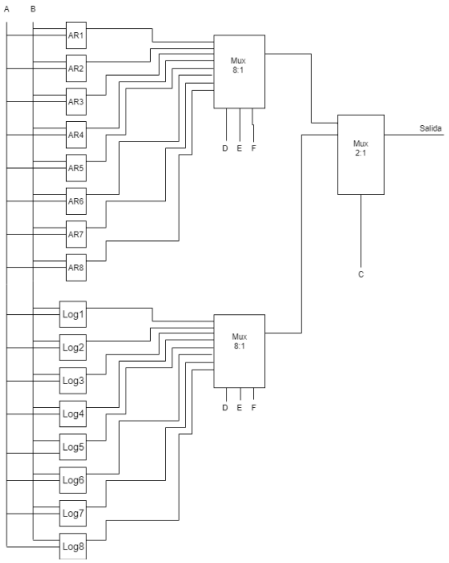


Figura 2: Ejemplo de una ALU

Según las operaciones que se realiza la ALU esta puede generar banderas, algunos ejemplos de ellas son:

- N: indica si el bit número almacenado es negativo, es el bit mas significativo
- Z: indica si es 0
- C: indica si hubo acarreo, pero depende del tipo de operacion
- V: esta es la que indica si hubo overflow en una operacion aritmetica.

- I: desactiva interrupciones IFQ

- F: desactiva interrupciones FIQ

Siguiendo con el tema de los tiempos se tiene el concepto de ruta crítica,la cual es la ruta que ocasiona el mayor retraso en el circuito para dar una respuesta, para esto hay que tomar en cuenta los tiempos de propagación y de contaminación anteriormente mencionados, ya que dependiendo de estos se podría tener una ruta critica mayor o menor, lo cual hará que nuestro circuito o procesador funcione a frecuencias mas bajas de las esperadas. Un ejemplo lo es el pipelining , que es utilizado en el diseño de procesadores para ejecutar más instrucciones por ciclo de reloj, consiste en dividir la ejecución de una instrucción en partes como: recuperación de la instrucción, decodificación, ejecución, acceso a memoria y escritura a registro. El pipelining no mejora el retraso del circuito pero aumenta la cantidad de instrucciones al mismo tiempo. Además, la ruta crítica es importante a la hora de diseñar un sistema de pipelining ya que es necesario que la señal no llegue a la siguiente etapa ni muy tarde ni muy temprano,por lo que si se quiere es hacer que los circuitos funcionen con ciclos de reloj mas rápidos se buscara la reducción de la ruta critica y a su vez los tiempos de propagación y contaminancion.

3. Estimación de la solución

A partir del problema planteado en la sección anterior, surge la necesidad de diseñar una unidad que combine una variedad de operaciones lógicas y aritméticas en un único modulo, conocida como ALU por sus siglas en inglés. Aprovechando la versatilidad que tienen los lenguajes de descripción de hardware para realizar módulos parametrizables.

4. Objetivos de la solución

- Diseñar una Unidad Lógica Aritmética parametrizable; es decir, con capacidad de recibir operandos de N bits.
- Implementar el diseño de la ALU, mediante un lenguaje de descripción de hardware.
- Optimizar la cantidad de componentes requeridos además de la ruta crítica de operación.
- Incorporar elementos de escalabilidad para el diseño.

5. Diseño del hardware del sistema

Las operaciones implementadas en la unidad lógica y aritmética en diseño cumplen con las especificaciones de selección que se muestran en la tabla 2. El código de operación, se encuentra controlado por la señal ALUControl, esta señal es de 4 bits pues existen 9 posibles operaciones y se debe poder seleccionar cada uno de ellos. En la tabla 2 se aprecia como el bit más significativo (MSB) representa la selección entre la unidad lógica y la aritmética.

ALUControl	Tipo	Operación
0000	L	OR
0001	L	XOR
0010	L	AND
0011	L	Barrel Shift
0100	L	Right Shift
0101	L	Left Shift
0110	L	X
0111	L	X
1000	A	Add
1001	A	Substract
1010	A	Right Shift
1011	A	X
1100	A	X
1101	A	X
1110	A	X
1111	A	X

Tabla 2: Código de selección de operaciones

5.1. Primer Nivel

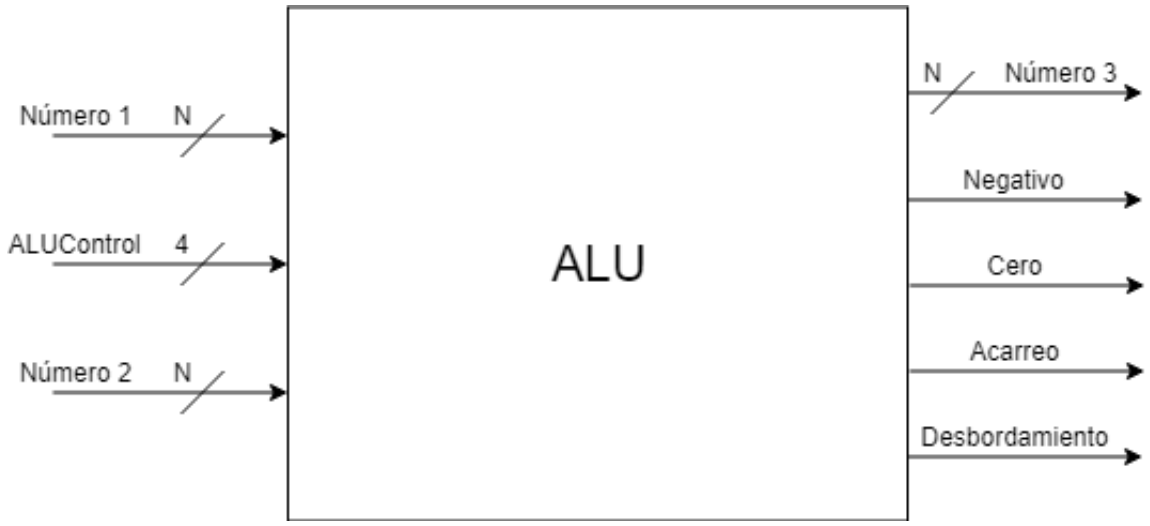


Figura 3: Diagrama de primer Nivel ALU

Objetivo: Unidad capaz de llevar a cabo operaciones lógicas y aritméticas a sus operandos de entrada; además de generar banderas según el resultado.

Entradas:

- Número1: Primer operando de N bits para operación
- Número2: Segundo operando de N bits para operación
- ALUControl: Señal de control de 4 bits, permite seleccionar la operación que se aplicará a los operandos

Salidas

- Número3: Resultado de operación entre Número1 y Número2 según seleccionado por ALUControl.

- Negativo: Bandera activa en alto; indica si el resultado de la operación aritmética es negativo.
- Cero: Bandera activa en alto; indica si el resultado de la operación lógica o aritmética es cero.
- Acarreo: Bandera activa en alto; indica si se produjo un acarreo en la operación de suma o resta.
- Desbordamiento: Bandera activa en alto; indica si el resultado de una suma o resta no se puede representar en los bits dados; es decir, se activa cuando hay un desbordamiento al bit de signo.

Explicación General: Este modulo representa la funcionalidad total de la ALU, esta unidad recibe dos valores numéricos de N bits; además de una señal de selección de 4 bits. Este componente genera como resultado un número de N bits a partir de la operación lógica o aritmética realizada según la define la señal de selección. Además según la operación realizada y el resultado obtenido se generan banderas que lo caracterízan y brindan información extra.

5.2. Segundo Nivel

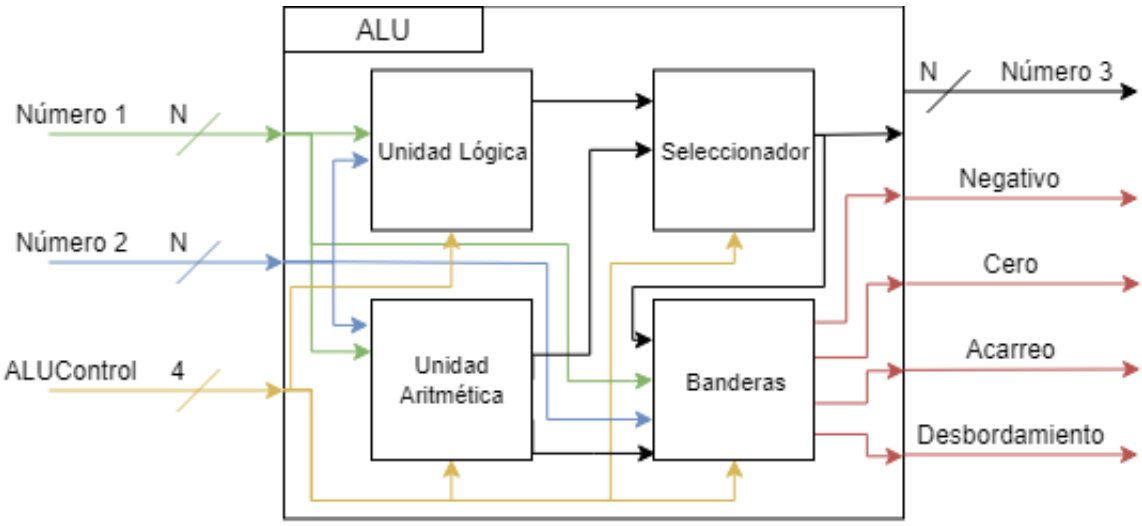


Figura 4: Diagrama de segundo Nivel ALU

5.2.1. Unidad Lógica

Objetivo: Unidad capaz de llevar a cabo operaciones lógicas a sus operandos de entrada.
Entradas:

- Número1: Primer operando de N bits para operación
- Número2: Segundo operando de N bits para operación
- ALUControl: Señal de selección para decidir cual operación lógica se debe realizar.

Salidas

- Resultado de la operación lógica seleccionada por la señal ALUControl.

Explicación General: Este modulo representa la funcionalidad lógica de la ALU, esta unidad recibe dos valores numéricos de N bits; además de una señal de selección de 4 bits. Este componente genera como resultado un número de N bits a partir de la operación lógica realizada según la define la señal de selección.

5.2.2. Unidad Aritmética

Objetivo: Unidad capaz de llevar a cabo operaciones Aritméticas a sus operandos de entrada.
Entradas:

- Número1: Primer operando de N bits para operación
- Número2: Segundo operando de N bits para operación
- ALUControl: Señal de selección para decidir cual operación Aritmética se debe realizar.

Salidas

- Resultado de la operación Aritmética seleccionada por la señal ALUControl.
- Bit de acarreo de salida para controlar la bandera de acarreo.

Explicación General: Este modulo representa la funcionalidad aritmética de la ALU, esta unidad recibe dos valores numéricos de N bits; además de una señal de selección de 4 bits. Este componente genera como resultado un número de N bits a partir de la operación lógica realizada según la define la señal de selección.

5.2.3. Seleccionador

Objetivo: Unidad capaz de seleccionar entre la unidad aritmética y la unidad lógica.
Entradas:

- Resultado de N bits, obtenido de la unidad aritmética
- Resultado de N bits, obtenido de la unidad lógica
- ALUControl: Señal de selección para decidir si se debe realizar una operación aritmética o lógica.

Salidas

- Resultado final,de la operación lógica y aritmética seleccionada por la señal ALUControl.

Explicación General: Este módulo representa la selección entre la unidad aritmética y la unidad lógica de la ALU, este modulo recibe dos valores numéricos de N bits; además de una señal de selección de 1 bit. Este componente genera un único resultado, tras seleccionar cual de las dos entradas es la que se desea en la salida.

5.2.4. Banderas

Objetivo: Unidad capaz de calcular las banderas de salida características del resultado.
Entradas:

- Bit más significativo del Número1
- Bit más significativo del Número2
- Señal de selección ALUControl.
- Resultado final de N bits, obtenido del seleccionador.
- Señal de acarreo de la unidad aritmética.

Salidas

- Bandera de signo negativo, activa en alto.
- Bandera de cero, activa en alto.

- Bandera de acarreo, activa en alto.
- Bandera de desbordamiento, activa en alto.

Explicación General: Este módulo calcula las banderas de caracterización del resultado; además selecciona los valores de las banderas, pues algunas de estas no tienen sentido en operaciones lógicas o aritméticas específicas.

5.3. Tercer Nivel

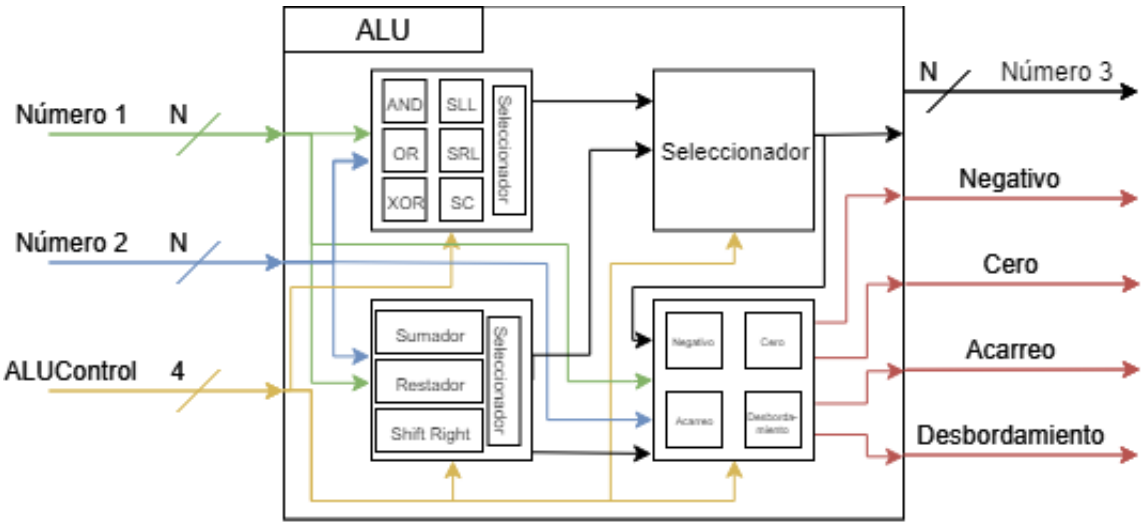


Figura 5: Diagrama de tercer Nivel ALU

5.3.1. AND

Objetivo: Unidad capaz de calcular la operación lógica AND a los números de entrada.
Entradas:

- Número1: valor lógico de N bits
- Número2: valor lógico de N bits

Salidas

- N bits de resultado de la operación AND.

Explicación General: Este módulo calcula la operación lógica AND; a partir de compuertas lógicas de esta naturaleza.

5.3.2. OR

Objetivo: Unidad capaz de calcular la operación lógica OR a los números de entrada.
Entradas:

- Número1: valor lógico de N bits
- Número2: valor lógico de N bits

Salidas

- N bits de resultado de la operación OR.

Explicación General: Este módulo calcula la operación lógica OR; a partir de compuertas lógicas de esta naturaleza.

5.3.3. XOR

Objetivo: Unidad capaz de calcular la operación lógica XOR a los números de entrada.
Entradas:

- Número1: valor lógico de N bits
- Número2: valor lógico de N bits

Salidas

- N bits de resultado de la operación XOR.

Explicación General: Este módulo calcula la operación lógica XOR; a partir de compuertas lógicas de esta naturaleza.

5.3.4. Corrimiento lógico a la izquierda

Objetivo: Unidad capaz de calcular la operación de corrimiento a la izquierda al número de entrada, en una cantidad definida.
Entradas:

- Número1: valor lógico de N bits para realizar el corrimiento.
- Número2: valor lógico de N bits, cantidad de espacios por correr.

Salidas

- N bits de resultado de la operación de corrimiento lógico a la izquierda.

Explicación General: Este módulo calcula la operación lógica de corrimiento a la izquierda.

5.3.5. Corrimiento lógico a la derecha

Objetivo: Unidad capaz de calcular la operación de corrimiento a la derecha al número de entrada, en una cantidad definida.
Entradas:

- Número1: valor lógico de N bits para realizar el corrimiento.
- Número2: valor lógico de N bits, cantidad de espacios por correr.

Salidas

- N bits de resultado de la operación de corrimiento lógico a la derecha.

Explicación General: Este módulo calcula la operación lógica de corrimiento a la derecha.

5.3.6. Corrimiento circular

Objetivo: Unidad capaz de calcular la operación de corrimiento circular al número de entrada, en una cantidad definida.

Entradas:

- Número1: valor lógico de N bits para realizar el corrimiento.
- Número2: valor lógico de N bits, cantidad de espacios por correr.

Salidas

- N bits de resultado de la operación de corrimiento lógico a la izquierda.

Explicación General: Este módulo calcula la operación lógica de corrimiento a la izquierda.

5.3.7. Sumador

Objetivo: Unidad capaz de calcular la operación de suma y acarreo de salida a los números de entrada.

Entradas:

- Número1: valor aritmético de N bits para realizar la suma.
- Número2: valor aritmético de N bits para realizar la suma.

Salidas

- N bits de resultado de la operación de suma.
- 1 bit de acarreo de la suma.

Explicación General: Este módulo calcula la operación aritmética de suma a los números de entrada; además calcula el acarreo de la suma.

5.3.8. Restador

Objetivo: Unidad capaz de calcular la operación de resta.

Entradas:

- Número1: valor aritmético de N bits al que se le restará el valor de Número2.
- Número2: valor aritmético de N bits para realizar la resta.

Salidas

- N bits de resultado de la operación de resta.

Explicación General: Este módulo calcula la operación aritmética de resta a los números de entrada; este cálculo se realiza en complemento a 2 lo cual permite reutilizar el sumador de la sección anterior.

5.3.9. Corrimiento aritmético a la derecha

Objetivo: Unidad capaz de calcular la operación aritmética de corrimiento a la derecha.
Entradas:

- Número1: valor numérico de N bits al que se le aplicará la operación.
- Número2: valor numérico de N bits define la cantidad de bits de corrimiento.

Salidas

- N bits de resultado del corrimiento aritmético a la derecha.

Explicación General: Este módulo calcula la operación aritmética de corrimiento a la derecha de la primer entrada, en una cantidad definida por la segunda entrada; este cálculo se realiza reemplazando los nuevos bits de la izquierda con el bit de signo del número.

5.3.10. Seleccionador

Objetivo: Unidad capaz de escoger entre diversas señales, según definido por la señal de selección.
Entradas:

- Numero variable de entradas de las cuales puede escoger.
- Señal de control para escoger cual entrada se propaga a la salida.

Salidas

- N bits de la entrada seleccionada.

Explicación General: Este módulo escoge una de las señales de entrada a partir de la señal de control, se le conoce como multiplexor y su función en este caso es definir la operación que se muestra en la salida.

5.3.11. Bandera de Negativo

Objetivo: Unidad capaz de calcular el signo de un número.
Entradas:

- bit más significativo (MSB) del número a calcular signo.
- señal de control que indica tipo de operación.

Salidas

- 1 bit de resultado que indica si el resultado es negativo.

Explicación General: Este módulo calcula la bandera de negativo a partir del bit más significativo y un bit que define si la operación realizada fue aritmética, pues en operaciones lógicas no tiene sentido hablar de números negativos, sino que todos son sin signo.

5.3.12. Bandera de Cero

Objetivo: Unidad capaz de calcular si un número es cero.
Entradas:

- Número1: N bits que representan el número por comprobar.

Salidas

- 1 bit de resultado que indica si el resultado es cero.

Explicación General: Este módulo calcula la bandera de cero a partir de los bits del número haciendo uso de una compuerta NOR aplicada a todos los bits de entrada.

5.3.13. Bandera de Acarreo

Objetivo: Unidad capaz de calcular si existe un acarreo luego de una suma o resta.
Entradas:

- bit de acarreo generado por la unidad aritmética

Salidas

- 1 bit de resultado que indica si hubo acarreo.

Explicación General: Este módulo calcula la bandera de acarreo a partir del bit de acarreo; su función consiste en comprobar que se esté realizando una operación de suma o una de resta; pues en las demás operaciones no tiene sentido hablar de acarreo.

5.3.14. Bandera de Desbordamiento

Objetivo: Unidad capaz de calcular si existe un desbordamiento luego de una suma o resta.
Entradas:

- bit más significativo del resultado final de la unidad lógica
- bit más significativo del primer operando
- bit más significativo del segundo operando

Salidas

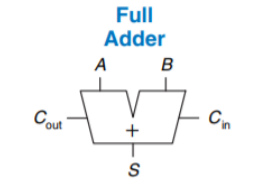
- 1 bit de resultado que indica si hubo desbordamiento.

Explicación General: Este módulo calcula la bandera de desbordamiento, eso ocurre cuando se cumplen tres condiciones: (1) La ALU está realizando una operación de suma o resta, (2) el primer operando y el resultado tienen signos contrarios, (3) ambos operandos tienen el mismo signo y se está realizando una suma, o los operandos tienen signo contrario y se está realizando una resta.

5.4. Cuarto Nivel

5.4.1. Sumador

En la siguiente figura se representa el símbolo del sumador; además de la tabla de verdad para un sumador simple diseñado por Harris y Harris [1]. Sin embargo, en la figura 7 se presenta la implementación utilizada, en este circuito se puede obtener el resultada así como el bit de acarreo.



C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

Figura 6: Símbolo y tabla de verdad de un sumador completo

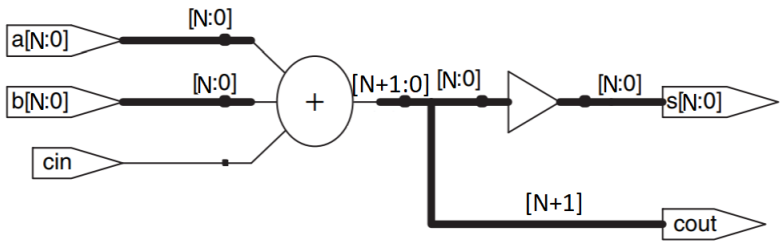


Figura 7: Sumador

Objetivo: Esta operación aritmética realiza la suma entre dos números binarios.
Entradas:

- a: Primer operando de N bits.
- b: Segundo operando de N bits.
- cin: Acarreo de entrada.

Salidas

- S: el resultado de sumar ambos operandos.
- cout: Acarreo de salida.

Explicación General: Este módulo calcula el resultado de la suma aritmética entre dos operandos de entrada, además del acarreo de salida, el resultado de la suma es después seleccionado por el multiplexor en caso de ser

necesario este resultado. El acarreo de salida sirve como entrada para el módulo de cálculo de banderas, para así calcular la bandera de acarreo cuando esta sea requerida.

5.4.2. Restador

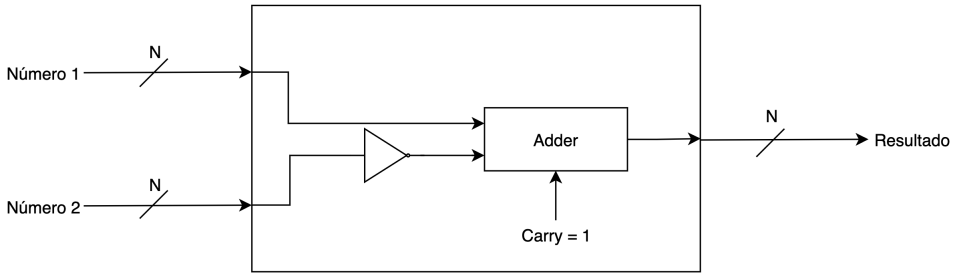


Figura 8: Restador

Objetivo: Esta operación aritmética realiza la resta entre dos números binarios.
Entradas:

- Número 1: Primer operando de N bits.
- Número 2: Segundo operando de N bits.

Salidas

- Resultado: el resultado de restar ambos operandos.
- cout: Acarreo de salida.

Explicación General: Este módulo calcula el resultado de la resta aritmética entre dos operandos de entrada, además del acarreo de salida, el resultado de la resta es después seleccionado por el multiplexor en caso de ser necesario este resultado. El acarreo de salida sirve como entrada para el módulo de cálculo de banderas, para así calcular la bandera de acarreo cuando esta sea requerida. Cabe destacar que la resta es realizada mediante el complemento a dos del segundo operando y el sumador completo desarrollado anteriormente.

5.4.3. Corrimiento aritmético

A continuación se presenta la implementación de un corrimiento aritmético según Harris y Harris [1], a partir de multiplexores.

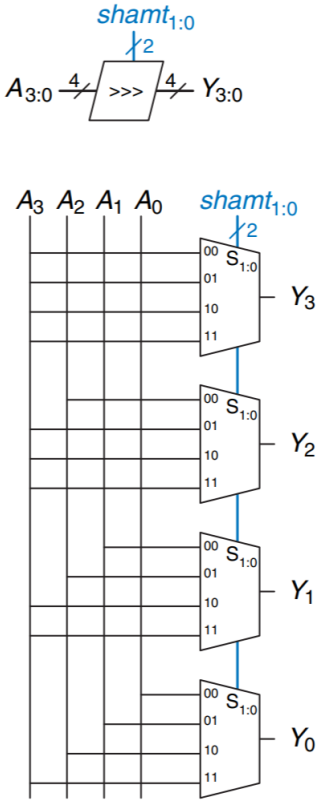


Figura 9: Corrimiento aritmético

Objetivo: Esta operación aritmética realiza el corrimiento a la derecha de un número, según la cantidad especificada.

Entradas:

- A: Operando de N bits.
- Shamt: Cantidad de posiciones a mover.

Salidas

- Y: el resultado de realizar el corrimiento.

Explicación General: El corrimiento aritmético consiste en correr el número hacia la derecha y rellenar el bit más significativo con el valor del bit de signo anterior con el fin de preservar el signo del número. La salida de este componente sirve como entrada para el multiplexor de selección de operaciones.

5.4.4. OR

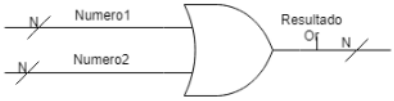


Figura 10: Compuerta OR

Objetivo: realizar la operación OR entre las dos números que son entradas
Entradas:

- Numero 1: Un numero de N bits
- Numero 2:Un numero de N bits

Salidas:

- Resultado Or: el resultado de operar con OR, Numero 1 y Numero 2, este resultado entra en el multiplexor para luego ser seleccionado para la salida de la ALU

Tabla de verdad compuerta OR:

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Tabla 3: Tabla de verdad de OR

5.4.5. XOR

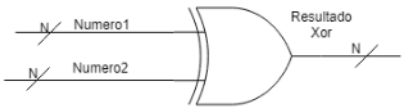


Figura 11: Compuerta Xor

Objetivo: realizar la operación XOR entre las dos entradas que son numeros binarios de N bits
Entradas:

- Numero 1: uno de los operandos de N bits
- Numero 2: uno de los operadores de N bits

Salidas:

- Resultado Xor: el resultado de operar con XOR Numero 1 y Numero 2, este resultado entra en el multiplexor para luego ser seleccionado para la salida.
  Tabla de verdad compuerta XOR:

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 4: Tabla de verdad de XOR

5.4.6. AND

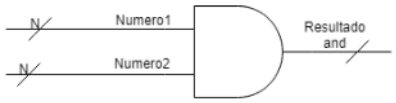


Figura 12: Compuerta AND

Objetivo: realizar la operación AND entre los dos operandos entrantes de N bits.
Entradas:

- Numero 1:uno de los operandos de N bits
- Numero 2:uno de los operadores de N bits

Salida:

- Resultado And: el resultado de operar con AND bit por bit Numero 1 y Numero 2, este resultado entra en el multiplexor para luego ser seleccionado para la salida.

Tabla de verdad compuerta AND:

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 5: Tabla de verdad de AND

5.4.7. Corrimiento lógico circular

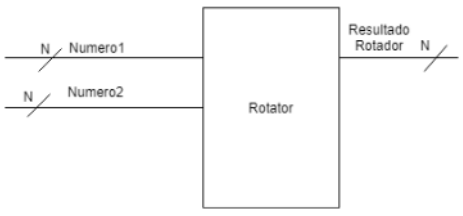


Figura 13: Desplazamiento Circular

Objetivo: mover una cantidad de bits del número 1 en forma circular.
Entradas:

- Numero 1: es el número binario que será operado.
- Numero 2: es la cantidad de bits que serán desplazados del Numero 1.

Salida:

- Resultado rotador: es Numero 1 luego de correrlo en forma circular Numero 2 bits a la derecha, este entra en el multiplexor para luego ser seleccionado para la salida.

Funcionamiento: el circuito lo que hara es ir corriendo N bits a la derecha como si fuera un Right Shifter, pero estos bits que han sido desplazados reaparecen en el inicio del número binario.
Ejemplos de corrimiento lógico circular:

A	B	S
0011 1100 0011	0000 0000 0011	0110 0111 1000
1110 0011 0110	0000 0000 0100	0110 1110 0011
0110 1101 1100	0000 0000 1000	1101 1100 0110

Tabla 6: Ejemplo corrimiento lógico circular

Siguiendo el ejemplo anterior el modelo del circuito diseñado por Harris y Harris [1] utilizado para esta función sera el siguiente:

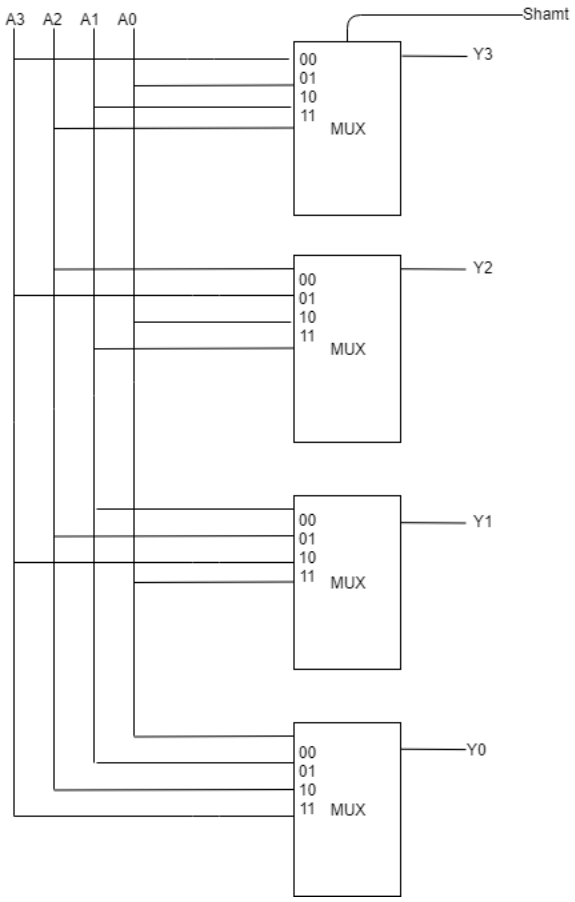


Figura 14: Desplazamiento circular

5.4.8. Corrimiento Lógico a la derecha

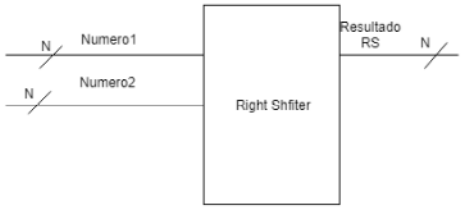


Figura 15: Desplazamiento a la derecha

Objetivo: Desplazar una cantidad de bits definida por Numero 2 del Numero 1 hacia la derecha. **Entradas:**

- Numero 1: es el número binario que será operado
- Numero 2: es la cantidad de bits a lo que será movido el número de la entrada A

Salidas:

- Resultado RS: es el Numero 1 luego de correrlo una cantidad definida por Numero 2 bits a la derecha,este entra en el multiplexor para luego ser seleccionado para la salida

Funcionamiento: Desplaza los bits de Numero 1 una cantidad de veces definida por Numero 2, los espacios vacíos son reemplazados con 0.

A	B	S
0011 1100 0011	0000 0000 0011	0000 0111 1000
1110 0011 0110	0000 0000 0100	0000 1110 0011
0110 1101 1100	0000 0000 1000	0000 0000 0110

Tabla 7: Ejemplo corrimiento a la derecha

Siguiendo el ejemplo anterior el modelo que nos funcionaria seria:

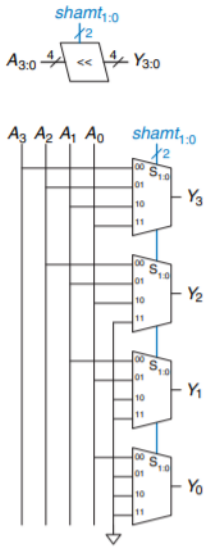


Figura 16: Circuito right shifter

5.4.9. Corrimiento lógico a la izquierda

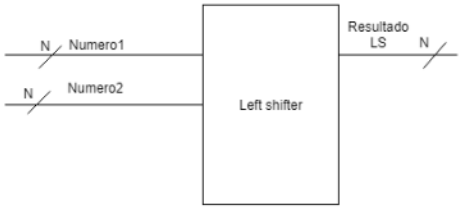


Figura 17: Desplazamiento a la Izquierda

Objetivo: mover una cantidad de bits descritas por Numero 2 del Numero 1 hacia la izquierda.
Entradas:

- Numero 1: es el número binario que será operado
- Numero 2: es la cantidad de bits a lo que serán desplazados del Numero 1.

Salida:

- Resultado LS: es Numero 1 luego de correrlo una cantidad de bits descrita por Numero 2 a la derecha, este entra en el multiplexor para luego ser seleccionado para la salida.

Funcionamiento: Este circuito la operación que realiza es correr los bits del Numero 1 hacia la izquierda, la cantidad que serán desplazados esta dada por Numero 2, los espacios vacíos son rellenados con un 0 lógico

A	B	S
0011 1100 0011	0000 0000 0010	1111 0000 1100
1110 0011 0110	0000 0000 0100	0110 0011 000
0110 1101 1100	0000 0000 1000	1100 0000 0000

Tabla 8: Ejemplo corrimiento a la izquierda

Siguiendo el ejemplo anterior el modelo del circuito que se utilizara sera el siguiente:

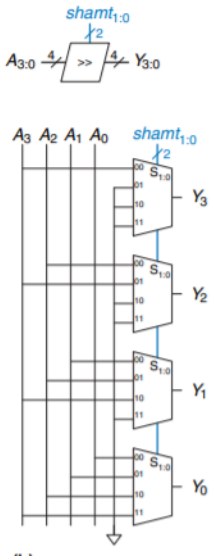


Figura 18: Modelo del circuito de left shifter

5.4.10. Seleccionador

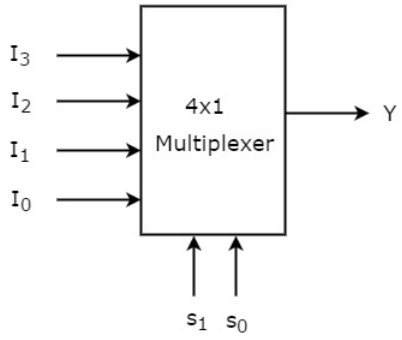


Figura 19: Modelo del circuito seleccionador

Objetivo: Unidad capaz de escoger entre diversas señales, según definido por la señal de selección.
Entradas:

- Numero variable de entradas de las cuales puede escoger.
- Señal de control para escoger cual entrada se propaga a la salida.

Salidas

- N bits de la entrada seleccionada.

Explicación General: Este módulo escoge una de las señales de entrada a partir de la señal de control, se le conoce como multiplexor y su función en este caso es definir la operación que se muestra en la salida.

5.4.11. Bandera de Negativo

Objetivo: El objetivo de este módulo es calcular la bandera de negativo.
Entrada:

- El bit más significativo del resultado de una suma o resta.

Salida:

- Un bit indicando si el numero es negativo, siendo 1 si es negativo y 0 si no lo es, dando así la bandera negativo.

Funcionamiento: Esta realiza una operación AND entre el bit mas significativo del resultado y el bit de control de la ALU, esto para generar la bandera solo cuando se realicen operaciones aritméticas.

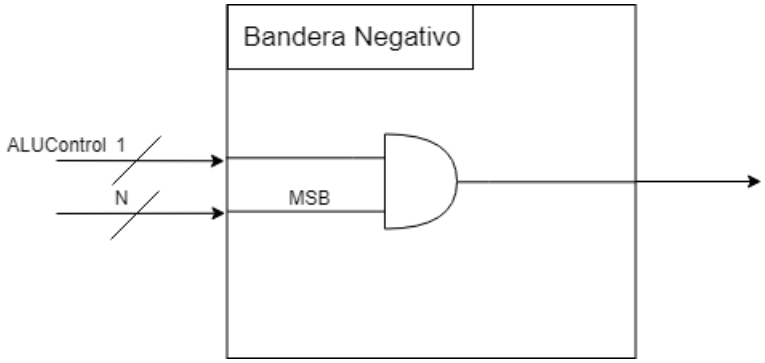


Figura 20: Modelo del circuito de construcción de bandera de negativo

---

5.4.12. Bandera de Cero

Objetivo: Determinar si el resultado obtenido de una operación aritmética es cero.
Entrada:

- Resultado del seleccionador
- Salida de la ALU

Salida:

- Un bit indicando si el numero es cero, siendo 1 indicación de que es cero y 0 si no lo es,dando así la bandera cero.

Funcionamiento:mediante una compuerta NOR, la cual tiene como entradas, cada uno de los bits del resultado, tendrá como salida 1 solo si ambas entradas son un 0 logico.

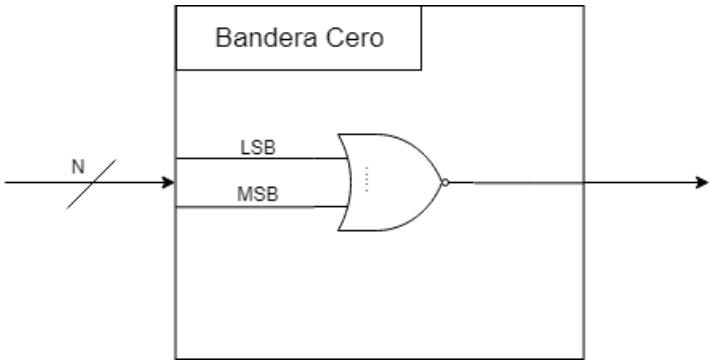


Figura 21: Modelo del circuito de construcción de bandera de cero

---

5.4.13. Bandera de Acarreo

El objetivo de esta bandera es determinar si el resultado de una suma o resta produce acarreo.
Entradas:

- bit de acarreo generado por la unidad aritmética

Salidas

- 1 bit de resultado que indica si hubo acarreo.

Explicación General: Este módulo calcula la bandera de acarreo a partir del bit de acarreo; su función consiste en comprobar que se esté realizando una operación de suma o una de resta; pues en las demás operaciones no tiene sentido hablar de acarreo.

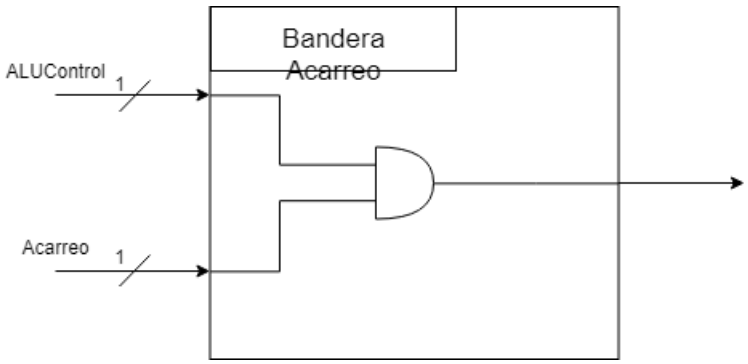


Figura 22: Modelo del circuito de construcción de bandera de acarreo

---

5.4.14. Bandera de Desbordamiento

Objetivo: Unidad capaz de calcular si existe un desbordamiento luego de una suma o resta.
Entradas:

- bit más significativo del resultado final de la unidad lógica
- bit más significativo del primer operando
- bit más significativo del segundo operando

Salidas

- 1 bit de resultado que indica si hubo desbordamiento.

Explicación General: Este módulo calcula la bandera de desbordamiento, eso ocurre cuando se cumplen tres condiciones: (1) La ALU está realizando una operación de suma o resta, (2) el primer operando y el resultado tienen signos contrarios, (3) ambos operandos tienen el mismo signo y se está realizando una suma, o los operandos tienen signo contrario y se está realizando una resta.

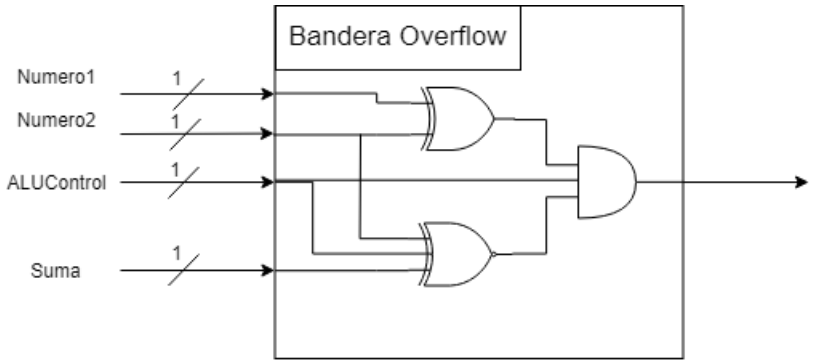


Figura 23: Modelo del circuito de construcción de bandera de desbordamiento

---

5.5. Quinto Nivel

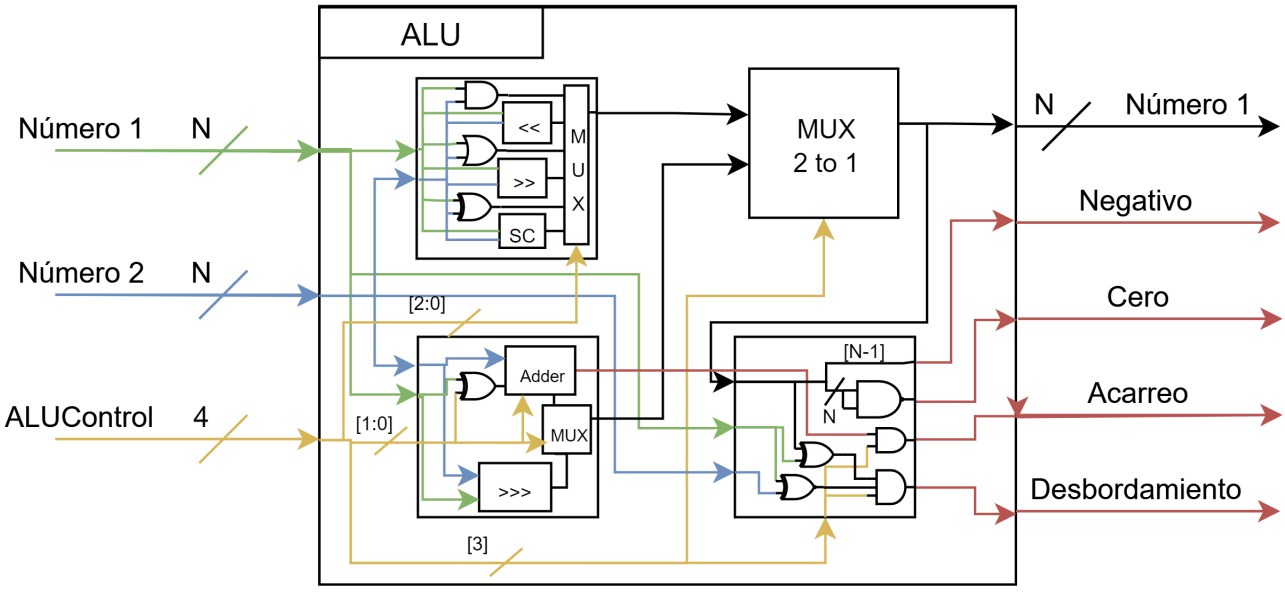


Figura 24: Diagrama de quinto nivel

Como se explico anteriormente la ALU trabaja con operandos de N bits, los cuales llamaremos Numero 1 y Numero 2, los cuales entraran a todos los circuitos anteriormente mencionados en el diagrama de cuarto nivel

(AND, OR, ADD, etc) que realizan operaciones, si dicha operación es aritmética y cumple la condición de que ha sido seleccionada, se calcularan las banderas de Negativo, Cero, Acarreo y desbordamiento, si no lo es, estas banderas no se calcularan, dando como resultado un 0 lógico. Continuando con las operaciones que involucran a Numero1 y Numero2, el resultado de la operación ingresara a un circuito de selección, que terminaría siendo un multiplexor, la salida de este multiplexor es escogida con la ALUControl, la cual es un numero binario de 4 bits, el resultado escogido es llamado Numero3 y sera el resultado final de todo el proceso de la ALU.

Referencias

[1] Sarah Harris and David Harris. Digital Design and Computer Architecture: ARM Edition. Morgan Kaufmann, 2015.