Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

Taller de Diseño Digital

Diagramas Diseño Modular Laboratorio #3

Estudiantes:
Arturo Chinchilla S.
Jorge Agüero Z.
Luis Murillo R.

Profesor:
Ing. Jeferson González Gómez,
M.Sc.

25 de agosto de 2018

1. Diagrama Nivel 1

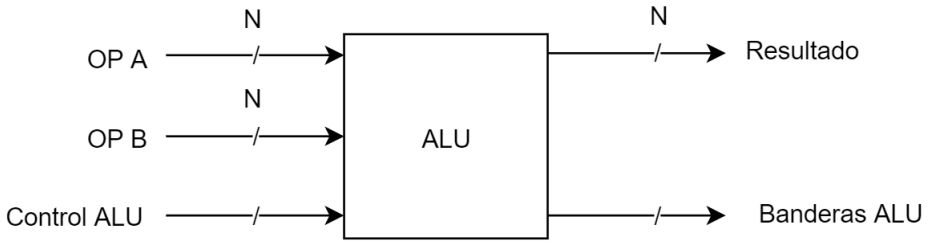

Figura 1: Diagrama Nivel 1

2. Diagrama Nivel 2

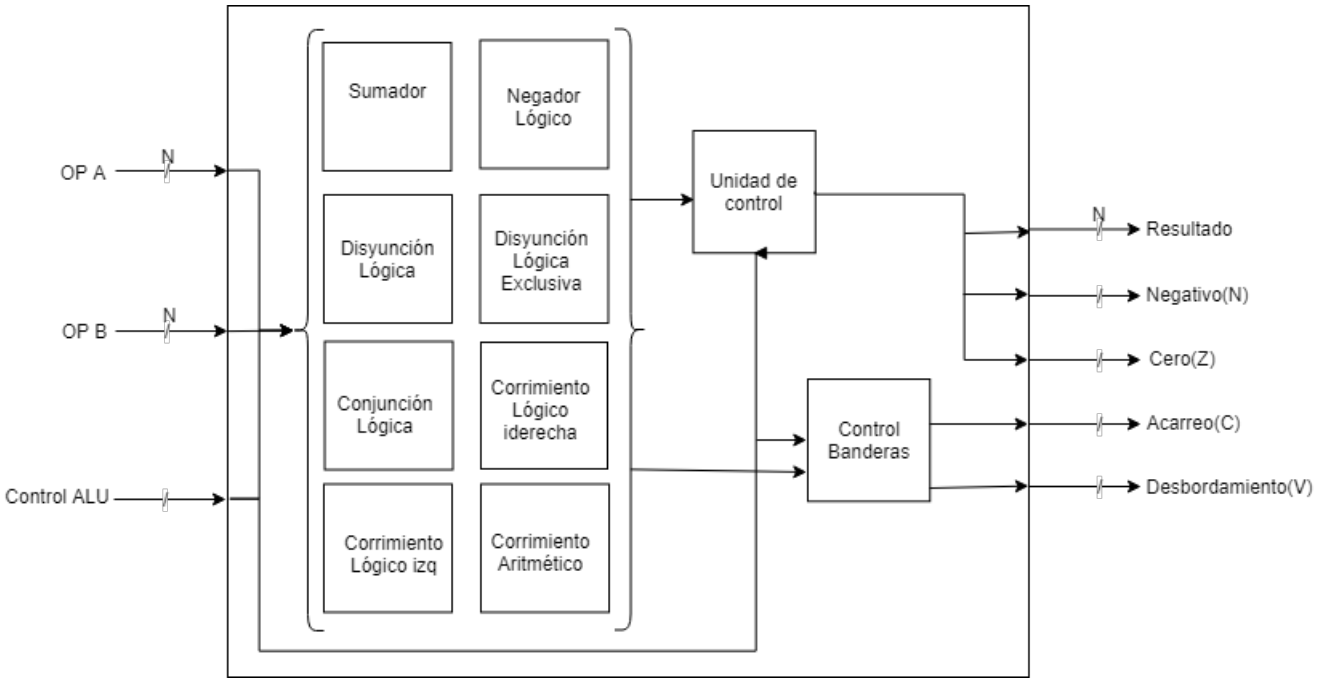

Figura 2: Diagrama Nivel 2

3. Diagrama nivel 3

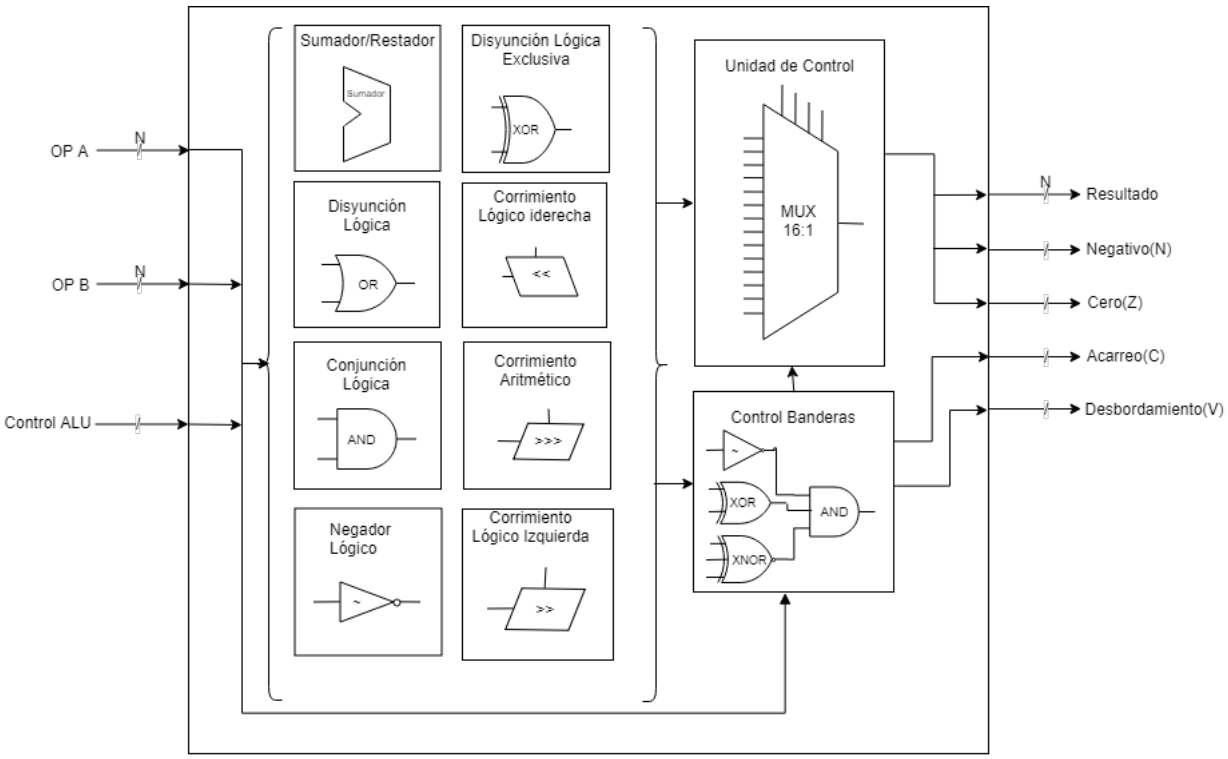

Figura 3: Diagrama Nivel 3

4. Diagrama Nivel 4

4.1. Circuito Sumador/Restador

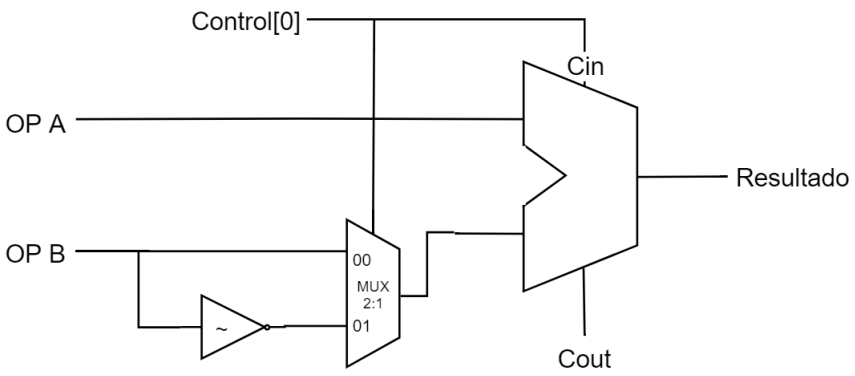

Figura 4: Sumador

Objetivo Dadas dos entradas muestra la suma o la resta de ambas entradas en la patilla de salida.

Entradas

- Control[0]: Es el bit 0 de la entrada de control, para el mux sirve como seleccionador del signo para la segunda entrada (suma o resta), mientras que para el sumador se encarga de introducir el carry de entrada.
- OP A: Operando A
- OP B: Operando B

Salidas:

- Resultado: Es el resultado de la operación, ya sea suma o resta.
- Cout: Carry de salida generado debido a la operación realizada.

Relación con otros módulos: El resultado mostrado en la salida del sumador se conecta al MUX de control que seleccionará la salida de la ALU.

Explicación de funcionamiento: Teniendo los dos operandos (OP A y OP B) el bit 0 de la entrada de control le dice al MUX del Operando B el signo que tendrá esta entrada, lo que ocasiona que en el sumador se realicen las operaciones de suma o resta, además este bit se conecta como carry de entrada al sumador, cuando el bit 0 de control es 0 el signo de OP B es positivo y se presenta un $C_{in} = 0.$, mientras que si el bit 0 de control es 1 el signo de de OP B es negativo y se tiene un $C_{in} = 1$.

Cuadro 1: Tabla de verdad Sumador/Restador

Tabla de Verdad				
Entradas			Salidas	
$Control$	OPA	OPB	C_{out}	$Resultado$
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

Diseño:

4.2. Circuito AND

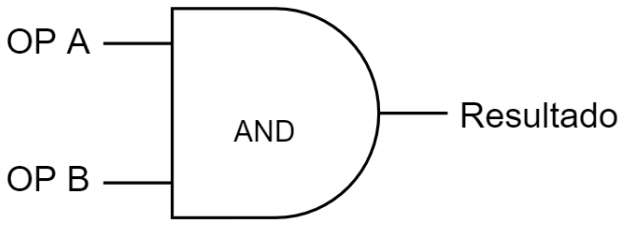

Figura 5: AND

Objetivo Dadas dos entradas muestra operación de conjunción lógica de ambas entradas en la patilla de resultado.

Entradas

- OP A: Operando A
- OP B: Operando B

Salidas:

- Resultado: Es el resultado de la operación conjunción lógica para ambas entradas.

Relación con otros módulos: El resultado mostrado en la salida de la compuerta AND se conecta al MUX de control que seleccionará la salida de la ALU.

Explicación de funcionamiento: El AND compara cada bit de A con B mediante la operación de conjunción lógica para obtener un resultado. El mismo funciona de la manera mostrada en la siguiente tabla de verdad.

Cuadro 2: Tabla de verdad AND

Tabla de Verdad		
Entradas		Salidas
OPA	OPB	*Resultado*
0	0	0
0	1	0
1	0	0
1	1	1

Diseño:

4.3. Circuito OR

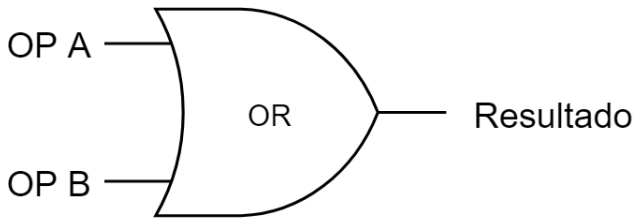

Figura 6: OR

Objetivo Este módulo es únicamente un OR, el cual se encarga de hacer una disyunción lógica de cada bit de A con B.

Entradas

- OP A: Operando A
- OP B: Operando B

Salidas:

- Resultado: Es el resultado de la operación de disyunción lógica para ambas entradas.

Relación con otros módulos: El resultado mostrado en la salida de la compuerta OR se conecta al MUX de control que seleccionará la salida de la ALU.

Explicación de funcionamiento: El OR compara cada bit de A con B mediante disyunción lógica para obtener un resultado. El mismo funciona mientras un valor sea A o B positivo el resultado final va a ser positivo.

Cuadro 3: Tabla de verdad OR

Tabla de Verdad		
Entradas		Salidas
OPA	OPB	*Resultado*
0	0	0
0	1	1
1	0	1
1	1	1

Diseño:

4.4. Circuito XOR

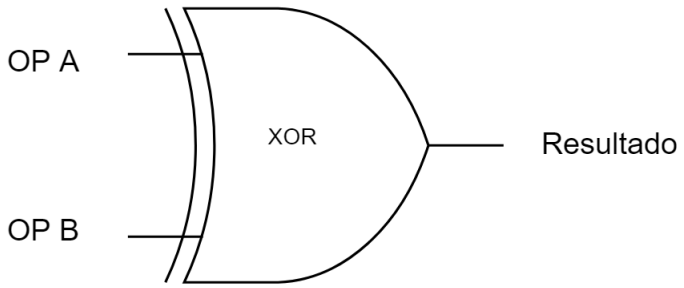

Figura 7: XOR

Objetivo Dadas dos entradas muestra operación de disyunción lógica exclusiva, conocida como XOR, de ambas entradas en la patilla de resultado.

Entradas

- OP A: Operando A
- OP B: Operando B

Salidas:

- Resultado: Es el resultado de la operación de disyunción lógica exclusiva para ambas entradas.

Relación con otros módulos: El resultado mostrado en la salida de la compuerta XOR se conecta al MUX de control que seleccionará la salida de la ALU.

Explicación de funcionamiento: El XOR compara cada bit de A con B mediante la operación de disyunción lógica exclusiva para obtener un resultado. El mismo funciona de la manera mostrada en la siguiente tabla de verdad

Cuadro 4: Tabla de verdad XOR

Tabla de Verdad		
Entradas		Salidas
OPA	OPB	$Resultado$
0	0	0
0	1	1
1	0	1
1	1	0

Diseño:

4.5. Circuito Corrimiento Izquierda

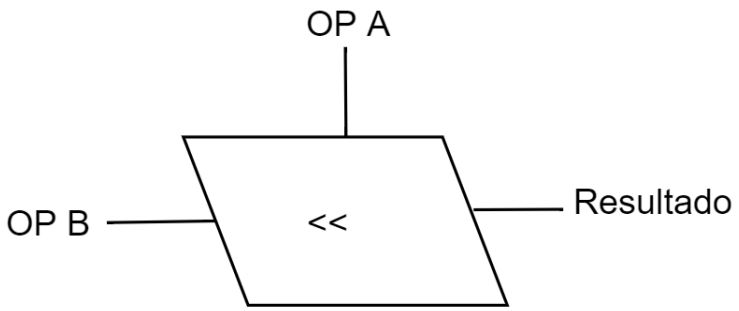

Figura 8: Corrimiento Izquierda

Objetivo Corre los bits de una secuencia hacia la izquierda, agregando 0 al lado derecho para completar el número.

Entradas

- OP A: Cantidad de posiciones a correr. Tamaño $log_2 N$. Para nuestro diseño se utilizara los bits bajos de la entrada OP A correspondientes a $A_{[log_2 N:0]}$
- OP B: Valor que se quiere correr.

Salidas:

- Resultado: Valor de tamaño N, con corrimiento hacia la izquierda la cantidad de posiciones deseada de 0 a N-1.

Relación con otros módulos: El módulo de corrimiento hacia la izquierda recibe sus entradas directamente de los operandos de entrada de la ALU, y su salida se conecta con el MUX de control.

Explicación de funcionamiento: Una entrada con N bits, puede ser corrida en N-1 posiciones. Este módulo corre el OP B la cantidad de posiciones dada por OP A. Se utilizan los bits bajos de la entrada OP A correspondientes a $A_{[log_2 N:0]}$,para codificar la cantidad de corrimientos que se quiere. Esto se puede implementar con N muxes de N entradas cada uno, pero en nuestro diseño lo haremos modelado por comportamiento en HDL.

4.6. Circuito Corrimiento Derecha

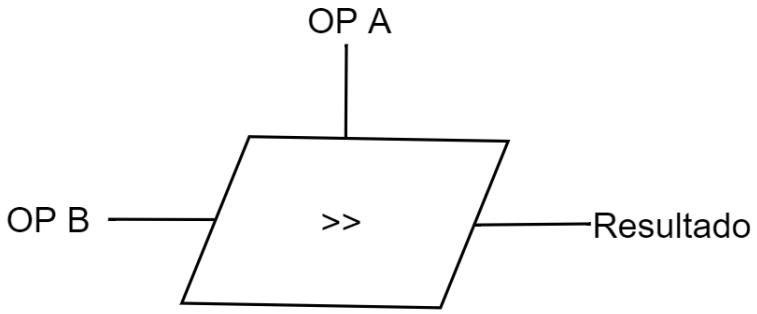

Figura 9: Corrimiento Derecha

Objetivo Corre los bits de una secuencia hacia la derecha, agregando 0 al lado izquierdo para completar el número.

Entradas

- OP A: Cantidad de posiciones a correr. Tamaño $log_2 N$. Para nuestro diseño se utilizara los bits bajos de la entrada OP A correspondientes a $A_{[log_2 N:0]}$
- OP B: Valor que se quiere correr.

Salidas:

- Resultado:Valor de tamaño N, con corrimiento hacia la derecha la cantidad de posiciones deseada de 0 a N-1.

Relación con otros módulos: El módulo de corrimiento hacia la izquierda recibe sus entradas directamente de los operandos de entrada de la ALU, y su salida se conecta con el MUX de control.

Explicación de funcionamiento: Una entrada con N bits, puede ser corrida en N-1 posiciones. Este módulo corre el OP B la cantidad de posiciones dada por OP A. Se utilizan los bits bajos de la entrada OP A correspondientes a $A_{[log_2 N:0]}$,para codificar la cantidad de corrimientos que se quiere. Esto se puede implementar con N muxes de N entradas cada uno, pero en nuestro diseño lo haremos modelado por comportamiento en HDL.

4.7. Circuito Corrimiento Aritmético

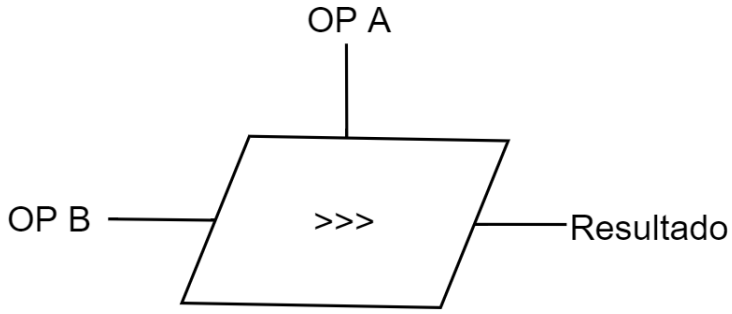

Figura 10: Corrimiento Aritmético

Objetivo Corre los bits de una secuencia hacia la derecha, manteniendo el valor del bit más significativo para llenar los espacios en la izquierda, es decir se copia, o se corre, el signo, moviéndolo hacia la derecha.

Entradas

- OP A: Cantidad de posiciones a correr. Tamaño $log_2 N$. Para nuestro diseño se utilizara los bits bajos de la entrada OP A correspondientes a $A_{[log_2 N:0]}$
- OP B: Valor que se quiere correr.

Salidas:

- Resultado: Valor de tamaño N, con corrimiento hacia la derecha la cantidad de posiciones deseada de 0 a N-1, copiando el valor del primer bit en los bits agregados a la izquierda.

Relación con otros módulos: El módulo de corrimiento hacia la izquierda recibe sus entradas directamente de los operandos de entrada de la ALU, y su salida se conecta con el MUX de control.

Explicación de funcionamiento: Una entrada con N bits, puede ser corrida en N-1 posiciones. Este módulo corre el OP B la cantidad de posiciones dada por OP A. Se utilizan los bits bajos de la entrada OP A correspondientes a $A_{[log_2 N:0]}$, para codificar la cantidad de corrimientos que se quiere. Esto se puede implementar con N muxes de N entradas cada uno, pero en nuestro diseño lo haremos modelado por comportamiento en HDL.

4.8. Circuito Desbordamiento

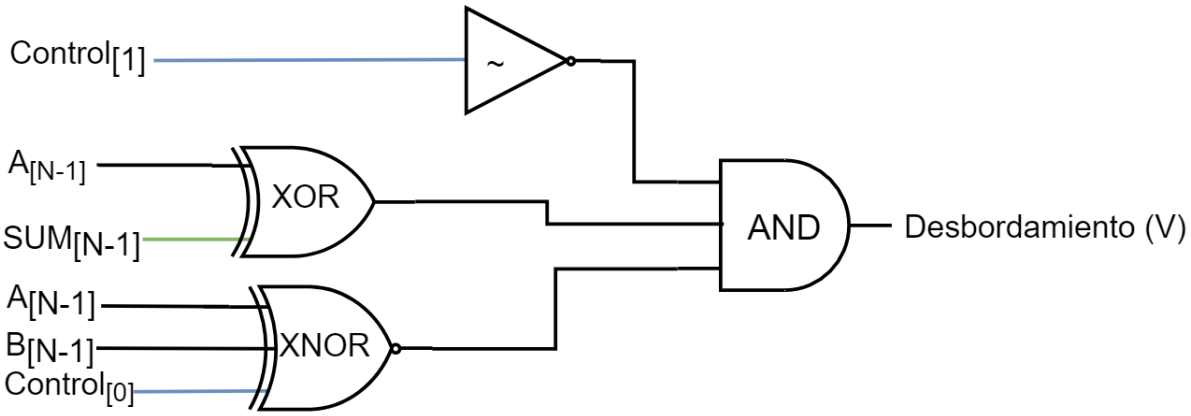

Figura 11: Desbordamiento

Objetivo Se encarga de calcular si hubo desbordamiento ($Overflow$) durante la operación aritmética dada.

Entradas

- $Control_{[1]}$: Se utiliza para saber si se está haciendo una operación de suma o resta.
- $Control_{[0]}$: Se utiliza para distinguir si se está haciendo una suma (0) o una resta (1).
- $A_{[N-1]}$: Bit más significativo del primer operando A, codifica el signo del operando.
- $B_{[N-1]}$: Bit más significativo del segundo operando B, codifica el signo del operando.
- $SUM_{[N-1]}$: Bit más significativo de la salida del sumador, codifica el signo del resultado del sumador.

Salidas:

- Resultado: "1" ó "0" dependiendo de si hubo desbordamiento o no.

Relación con otros módulos: El módulo de cálculo de desbordamiento se conecta los bits más significativos de los operandos de entrada de la ALU directamente, así como de los primeros dos bits de la señal de control. Utiliza la señal de $SUM_{[N-1]}$ para saber el signo del resultado de la suma, por lo que recibe de entrada la salida del sumador. Su salida es un bit de bandera de la ALU.

Explicación de funcionamiento: El desbordamiento se dá si se está haciendo una operación de suma con operandos del mismo signo, o resta con operandos de signo opuesto, y el resultado tiene un signo diferente al del primer operando. Para esto entonces se prueba que se esté haciendo una operación de suma o resta con la entrada de $Control_{[1]}$, puesto que esta es 0 cuando se elige la suma o la resta esta se invierte. Luego se revisa con el XOR si el signo de alguna de las entradas es diferente del signo de la salida (en este caso $A_{[N-1]} \oplus Resultado_{[N-1]}$) y también se debe cumplir que se de una suma ($Control_{[0]} = 0$)con valores positivos de entrada, o una resta ($Control_{[0]} = 1$) y que los signos de A y B sean diferentes, lo cual es revisado por el XNOR. Los resultados de la XOR y XNOR se pasan por un AND para verificar que todas estas condiciones se den en conjunto.

4.9. Circuito Acarreo

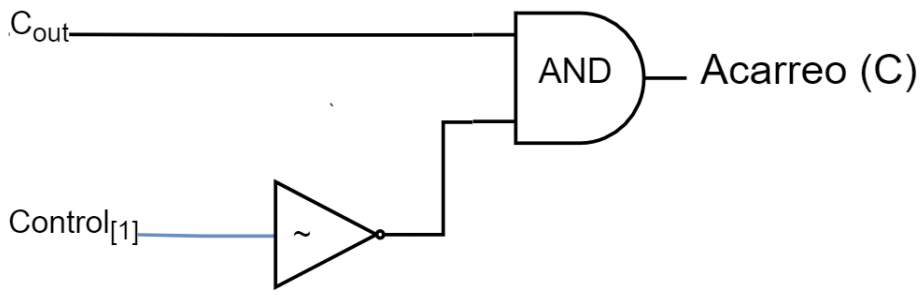

Figura 12: Acarreo

Objetivo Cuando se realiza el módulo sumador/restador el mismo da dos salidas, la primera es el resultado de la suma realizada y la segunda el C_{out} o acarreo de salida el cual es, el sobrante de la última suma. Este módulo se encarga de dar este valor.

Entradas

- C_{out}: Salida de acarreo del último sumador de 1 bit.
- $Control_{[1]}$: Bit de control.

Salidas:

- Resultado: "1" ó "0" dependiendo de si hubo acarreo o no.

Relación con otros módulos: Este módulo se relaciona con el módulo del sumador ya que se utiliza la salida del mismo para el valor C_{out}(acarreo de salida) como entrada. Además a esto presenta el valor del acarreo final del sumador.

Explicación de funcionamiento: El acarreo como se menciono se encarga de obtener el ultimo bit de acarreo del sumador, este valor se puede utilizar posterior a la ALU como un dato obtenido aparte. El mismo solo funciona para la suma de valores(esto se plantea en el diseño).

Cuadro 5: Tabla de verdad Acarreo

Tabla de Verdad		
Entradas		Salidas
OPA	OPB	*Resultado*
0	0	0
0	1	0
1	0	1
1	1	0

Diseño:

4.10. Control ALU

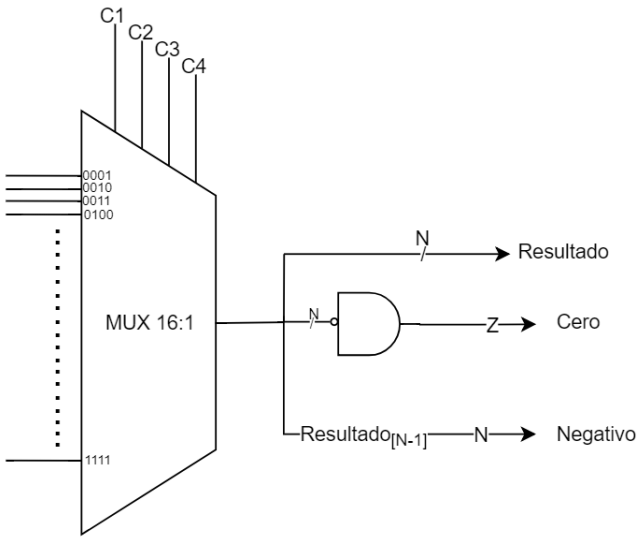

Figura 13: Control ALU

Objetivo Este módulo se encarga de controlar la salida de la ALU, ya que todos los módulos se encuentran trabajando al mismo tiempo. El multiplexor recibe la salida de los módulos y mediante los 4 bits de control selecciona cual de estos resultados se verán en la salida de la ALU..

Entradas

- C1: Es el bit 0 de selección.
- C2: Es el bit 1 de selección.
- C3: Es el bit 2 de selección.
- C4: Es el bit 3 de selección.

Salidas:

- Resultado: El valor de entrada seleccionado por el MUX.
- Cero: Obtiene un Bit de 0.
- Negativo: La salida sin el último bit (bit de signo).

Relación con otros módulos: Este módulo se relaciona con todos los módulos anteriores, ya que está encargado de seleccionar una de las salidas de los módulos anteriores, para la salida final de la ALU.

Explicación de funcionamiento: El multiplexor tiene 2^N entradas donde "N" se toma como las entradas de selección, en este caso son las 4 entradas (C1,C2,C3,C4) y solo presenta una salida. Esta misma salida se utiliza para obtener un bit 0 y la un mismo resultado negado para diferentes funciones

Cuadro 6: Control ALU

Tabla de Verdad		
Control	**Operación**	**Descripción**
0000	$suma$	$A+B$
0001	$resta$	$A-B$
0010	AND	$A\&B$
0011	NOT	B
0100	OR	$A\|B$
0101	XOR	$A \oplus B$
0110	$corrimientoizquierdo \ll$	$B \ll A_{log_2 N}$
0111	$corrimientoderecho \gg$	$B \gg A_{log_2 N}$
1000	$corrimientoaritmtico \ll$	$B \ll A_{log_2 N}$
...		
$default$		0

5. Diagrama Nivel 5

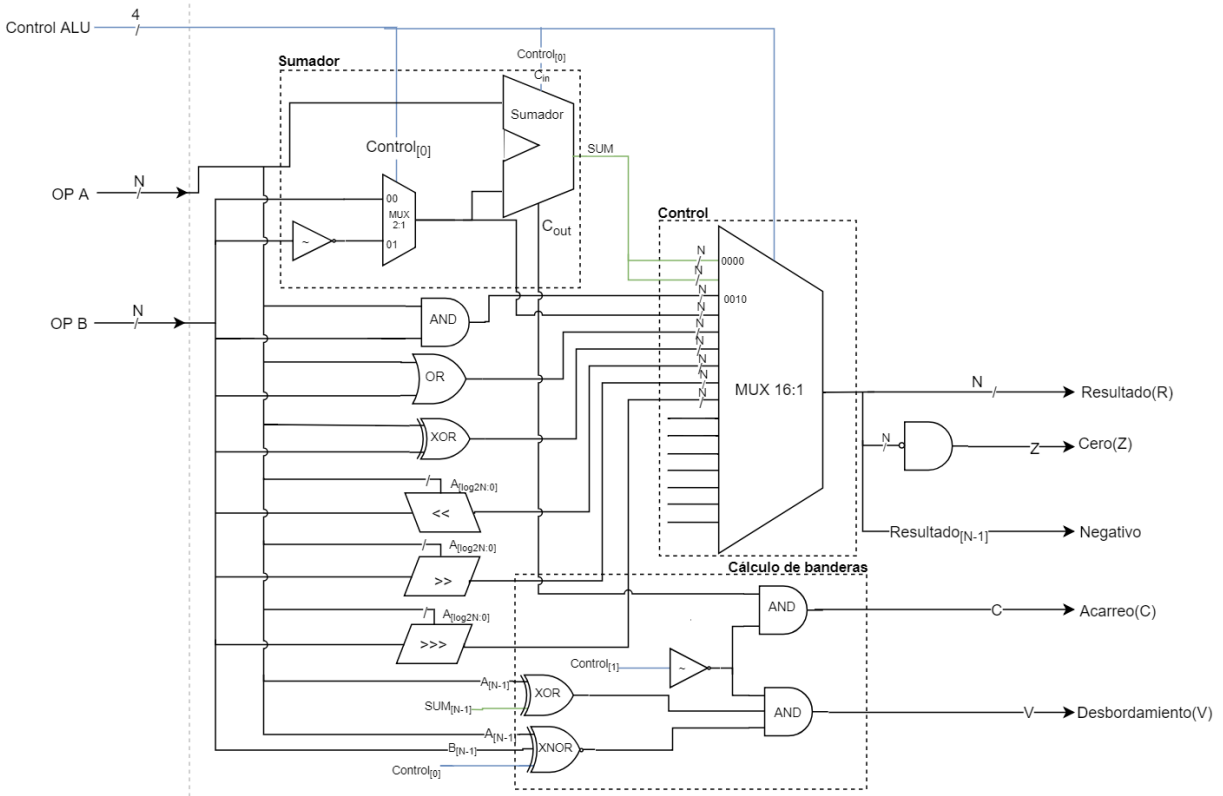

Figura 14: Diagrama Nivel 5