|  |  |  |  |
| --- | --- | --- | --- |
| Informe de trabajo  Trabajo final integrador  Facultad de Ingeniería UNLP  E0301 Introducción al Diseño Lógico  Curso 2023 | |  | G1  31 de julio de 2023 |
|  | |  |
|  | |  |  |
| s | |  |  |
|  |  |  | |
| * Caciani Toniolo, Melina   melicaciani@gmail.com | |  | 02866/1 |
| * Chanquía, Joaquín   joaquin.chanquia@alu.ing.unlp.edu.ar | |  | 02887/7 |
| * Larsen, Mateo Emmanuel   larsenmateo.ml@gmail.com | |  | 02993/7 |
| * Ollier, Gabriel   gabyollier@hotmail.com | |  | 02958/4 |
|  | |  |  |

# ENUNCIADO

**Descripción del sistema a desarrollar**

Debe diseñar una ALU que permita operandos de cuatro (4) bits, con la

funcionalidad que se describe a continuación.

Deberá poseer tres registros denominados A (Acumulador), B y Estado, de cuatro (4) bits cada uno.

Los registros A y B deberán poseer entradas externas que permitan su carga. La carga del registro A se producirá cuando la entrada externa LOADA se ponga en nivel ALTO. La carga del registro B se producirá cuando la entrada externa LOADB se ponga en nivel ALTO.

El registro Estado, deberá tener cuatro bits (E3 E2 E1 E0) y almacenar los siguientes flags:

**→ E3**: Flag de Overflow que denominaremos V.

**→ E2**: Flag Negativo, que indica que el contenido del Acumulador representa

un número negativo en Complemento a 2, al que denominaremos N.

**→ E1**: Flag que indica que el contenido del Acumulador tiene todos sus bits en

BAJO, como esta situación corresponde a la representación de 0, se

denominará Flag Zero o Z.

**→ E0**: Flag de Acarreo que indica cuando se produjo un acarreo como resultado

de la última operación y que denominaremos C.

Las operaciones que realice la ALU deberán ser las siguientes:

Tabla

Descripción generada automáticamente

**Referencias:**

# ↕ : El valor puede ser 0 o 1 dependiendo del resultado de la operación

# ●: El valor no se modifica

# R: el valor se establece en 0

# S: el valor se establece en 1

* Cuando se realice una operación que involucre un único registro, no deberá afectarse el contenido del otro.
* La operación A?B (COMPARAR) realizará la operación A – B y deberá dejar ambos registros inalterados, pero deberá establecer los flags.
* La operación LSL(A) (Desplazamiento lógico a izquierda) desplaza los bits a la izquierda desde el LSB hacia el MSB. El MSB original se pierde y en el LSB se ingresa un 0.
* La operación LSR(A) (Desplazamiento lógico a derecha) desplaza los bits a la derecha desde el MSB hacia el LSB. El LSB original se pierde y en el MSB se ingresa un 0.
* Las operaciones AND , OR y XOR , se realizan bit a bit entre los registros A y B.
* La operación NOT se realiza invirtiendo todos los bits del registro A.
* La operación ASR(A) (Desplazamiento aritmético a izquierda) desplaza los bits a la derecha desde el MSB hacia el LSB. El LSB original se pierde y se extiende el signo, es decir, se propaga hacia la derecha el MSB (0 si A es positivo o 1 si A es negativo).
* La operación A = B reemplaza el contenido del registro A con el contenido del registro B.

# IMPLEMENTACIÓN

Para Organizar de una mejor manera la implementación del trabajo separamos las operaciones en dos grupos, según si las mismas utilizaban el registro B o no. De esta manera nos resulto mas fácil separar las operaciones en dos Bloques principales:

El bloque ABALU, el cual incluye a todas las operaciones que utilizan al registro B. Y también incluye a la operación Incrementar para agruparla con el resto de operaciones de suma. Y el otro bloque es el registro A, en el cual quedaron todas las operaciones que no requerían del valor en el registro B.

Para distinguir entre las operaciones llamaremos una operación Interna a las que se realicen en el Registro A, por lo que estas tendrán un 1 en el bit I. y cada una tendrá su respectivo código de operación interna (OPI). A su vez las operaciones que se realicen en el bloque ABALU se denominaran Externas, tendrán un 0 en el bit I y contaran con su código de operación Externo (OPE). La tabla que muestra el código correspondiente a cada operación es la siguiente:

Imagen que contiene Gráfico

Descripción generada automáticamente

Los mapas de Karnaugh utilizados para definir la función que corresponde a cada bit, la cual será implementada en el Decodificador de Operaciones son los siguientes:

Calendario

Descripción generada automáticamente

Las funciones lógicas resultantes son:

I = /Op3./Op2.Op1+Op3./Op2./Op1+Op2.Op1./Op0+/Op3.Op2./Op0+Op3./Op1.Op0

I = /Op2.(Op3 xor Op1) + Op2./Op0.(Op1 + /Op3) + Op3./Op1.Op0

OPI:

OPI[2] = Op1.Op2 + /Op2.Op3

OPI[1] = Op0 + Op3.Op1

OPI[0] = Op1

OPE:

OPE[2] = Op3

OPE[1] = Op2

OPE[0] = Op1.Op2 + /Op2.Op0 + Op0.Op3

Y el decodificador de operaciones implementado en quartus:

Pantalla de computadora

Descripción generada automáticamente con confianza media

Esquemas de cada operación:

Diagrama, Esquemático

Descripción generada automáticamenteA = A + B 0000

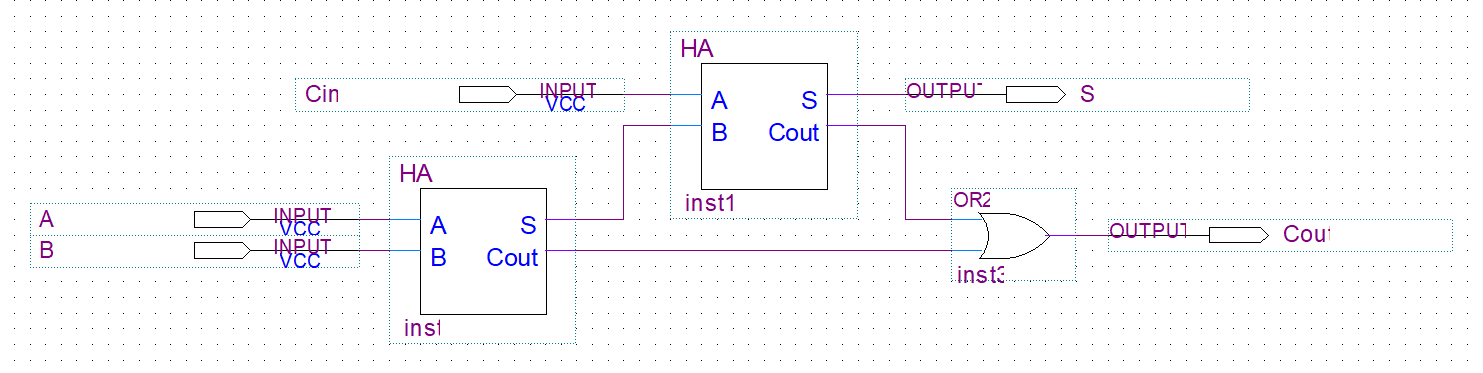
Este bloque lo realizaremos utilizando semisumadores y sumadores completos en cadena para realizar la suma de dos variables de 4 bits. Las entradas son A y B y las salidas son R (el resultado), Cout (El bit que indica un posible acarreo en la operación de suma) y V el bit que indica un posible overflow en la suma.

Bloque Semisumador (HA):

Diagrama, Esquemático

Descripción generada automáticamente

Bloque Sumador Completo (FA):



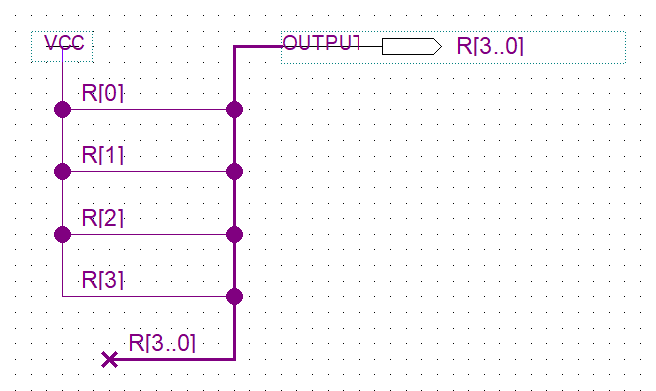
A = A – B 0001

Diagrama

Descripción generada automáticamente con confianza media

Para realizar la resta utilizaremos el bloque de suma y el de negación. Las entradas son A y B, de las cuales A ingresara a la suma directamente y B pasara primero por el bloque de negación para luego entrar a la suma. Las salidas son R, Cout y V igual que con la suma.

A = 0xF (SET) 0010



Para realizar el set creamos un bus en el cual todos sus bits tengan el valor 1 y haremos que esa sea la salida del bloque (este bloque no necesita entradas).

Diagrama

Descripción generada automáticamenteA = – A 0011

Para conseguir negar el valor A utilizaremos los bloques de NOT y el de INC A. Para conseguir el complemento a 2 del número A, primero negaremos todos sus bits y luego sumaremos 1 utilizando el Incrementador. El bloque cuenta con una entrada

A = 0x0 (CLEAR) 0100

Imagen que contiene Diagrama

Descripción generada automáticamente

Para realizar el clear, como hicimos con el set, creamos un bus con todos sus bits en cero y hacemos que la salida sea ese mismo.

A = A+1 (INC A) 0101 Diagrama

Descripción generada automáticamente con confianza media

Para realizar el incrementador utilizaremos el bloque de suma, en el cual ingresaremos la entrada A en la suma y de entrada B de la suma utilizaremos un bus en el que haremos que sus bits sean 0001 para realizar una suma de una sola unidad. Las salidas son las mismas que la suma, R (Respuesta al incremento), Cout (Flag de Carry) y V (Flag de overflow).

NOP () 0110

Calendario

Descripción generada automáticamente con confianza media

Para la operación Nop haremos que la salida sea igual a la única entrada (A).

A?B (COMPARAR) 0111

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

Para la operación de comparación utilizaremos el bloque de resta para realizar la operación y conseguir los flags resultantes. Pero la respuesta proveniente de la resta no la usaremos como salida para este bloque sino que conectaremos la entrada A para que esta no sufra cambios. Pero la respuesta de la resta la usaremos para los flags Z (Flag de cero) y N (Flag de negativo). Las ultimas salidas son las de los flags de acarreo (C) y de overflow (V), las cuales provienen del bloque de resta.

A = LSL (A) 1000

Diagrama

Descripción generada automáticamente

Para realizar un desplazamiento logico de bits a izquierda asignaremos los bits de la entrada A en la salida R pero corridos un lugar hacia la izquierda en el bus. El bit A[0] se asigna a R[1], el bit A[1] a R[2] y el bit A[2] a R[3]. En el lugar de R[0] se inserta un 0.

(Las compuertas AND en el diagrama las usamos para poder pasar un bit de un bus a otro ya que quartus no nos permitía poner dos nombres a una misma línea)

A = LSR (A) 1001

Diagrama, Esquemático

Descripción generada automáticamente

Para el desplazamiento lógico a derecha se prosigue con la misma lógica que en el desplazamiento a izquierda pero las posiciones en R se asignan de la siguiente manera: A[3] se inserta en R[2], A[2]se inserta en R[1], A[1] en R[0] y a R[3] se le asigna un 0.

Diagrama, Esquemático

Descripción generada automáticamenteA = A & B (AND)

Para realizar la operación AND utilizamos compuertas AND de dos entradas para realizar la operación bit a bit entre A y B. Las entradas de esta operación son A y B y la salida es R, la respuesta de la operación.

Diagrama, Esquemático

Descripción generada automáticamenteA = A | B (OR) 1011

Para realizar la operación OR utilizamos compuertas OR de dos entradas para realizar la operación bit a bit entre A y B. Las entradas de esta operación son A y B y la salida es R, la respuesta de la operación.

Diagrama

Descripción generada automáticamenteA = A ^ B (XOR) 1100

Para realizar la operación XOR utilizamos compuertas XOR de dos entradas para realizar la operación bit a bit entre A y B. Las entradas de esta operación son A y B y la salida es R, la respuesta de la operación.

A = ~ A (NOT) 1101

Gráfico

Descripción generada automáticamente

Para realizar la operación NOT utilizamos la función de quartus que permite negar todos los bits de un bus utilizando una única compuerta NOT.

A =ASR(A) 1110

Diagrama, Esquemático

Descripción generada automáticamente

Para realizar el desplazamiento aritmético a derecha, se realiza todo de la misma manera que en el desplazamiento lógico pero en el bit R[3] se le asigna A[3] para que se mantenga el ultimo bit sin cambios.

A = B

Imagen que contiene Gráfico de líneas

Descripción generada automáticamente

Para igualar A con el valor de B, pusimos como única salida R, el cual esta conectada directamente con B para que A sea reemplazada por B.

Operaciones en el bloque ABALU:

Imagen que contiene Gráfico

Descripción generada automáticamente

Diagrama del bloque ABALU:

Diagrama, Esquemático

Descripción generada automáticamente

El bloque se puede separar en dos partes, una (en la parte superior del diagrama) son todas las operaciones que implican una operación de suma o resta, que para simplificar el proceso se lleva a cabo todo en un único bloque de suma y resta. El cual se encarga de definir si se trata de una suma o de una resta mediante la entrada Op. a la cual le enviamos el bit 0 de OPE, el cual es un 0 para las sumas (Sumar e incrementar) y es un 1 para las restas (Restar y comparar). El diagrama de SumaResta es el siguiente:

Diagrama, Esquemático

Descripción generada automáticamente

La entrada A es igual para todas las operaciones. La entrada B en cambio en la operación Incrementar se cambia por un bus al cual le pusimos el valor 0001. Para seleccionar esta entrada en este caso en especifico utilizamos un multiplexor que tiene como entrada una función lógica que solo devuelve 1 en el caso que OPE termine en 10 (que es como termina el código del incrementar).

La salida de la operación en la comparación no debe ser la que resulte de la resta, y para que no resulte modificado el registro A seleccionamos que la salida de la operación sea la entrada A del bloque. Aun así para que se modifiquen los flags Z y N correctos estos se calculan antes del multiplexor que elije la entrada A como salida cuando se genera un 0 solo en el caso que los dos últimos bits de la OPE sean 11 (cosa que solo sucede con la operación comparar entre las operaciones de suma). El diagrama que calcula los flags Z y N activa el primero si todos los bits son 0 (una compuerta NOR) y el segundo si el bit mas significativo es un 1 (conecta este bit a la salida):

Diagrama

Descripción generada automáticamente

La parte inferior del diagrama es mas sencilla en el sentido que solo se compone de los respectivos bloques a las operaciones logicas AND, OR, y XOR y a la operación A=B. Para la cual conectamos la entrada B como salida. Luego de esto los resultados de las operaciones son seleccionados con un multiplexor de buses de cuatro entradas. Los flags V y C en estas operaciones deben dejarse sin modificar y para lograr eso se utiliza la entrada Fa (Flags Anteriores) del bloque y se extraen los bits correspondientes a V y C. Con el bloque Flags2, cuyo diagrama es:

Diagrama, Esquemático

Descripción generada automáticamente

Operaciones en el bloque RegistroA:

Imagen que contiene Gráfico

Descripción generada automáticamente

Diagrama del bloque RegistroA:

Diagrama, Esquemático

Descripción generada automáticamente

En el recuadro verde se encuentran las operaciones de CLEAR y SET, donde se toma el bit menos significativo de OPI el cual usa el bit menos significativo del OPI para introducir un valor 1 en todos los bits del bus para el Set y un valor de 0 para todos los bits en el Clear (ya que estos son los valores que toma este bit en el OPI para estas operaciones)

En el recuadro amarillo se encuentran las operaciones de desplazamiento de bits (LSL, LSR, ASR) y la operación Nop. Las operaciones de desplazamiento a derecha e llevan a cabo en el bus llamado SR[3..0], diferenciando entre el LSR y el ASR mediante el multiplexor de la izquierda, el cual se encarga de establecer el bit mas significativo como un 0 para el desplazamiento lógico y deja igual al bit menos significativo anterior para el aritmético. El desplazamiento a izquierda se realiza en el bus llamado SL[3..0], el cual se inserta en un multiplexor junto a la operación Nop (el NOT antes de la entrada de selección del multiplexor es porque el bit menos significativo de la operación Nop y del LSL quedaron invertidos en las entradas del multiplexor)

En el recuadro celeste se encuentran las operaciones NOT y el Inversor (-A), ambas se realizan invirtiendo todos sus bits, donde ya se obtiene la primer operación, sin embargo el inversor necesita incrementar al valor obtenido en uno para obtener el valor esperado, se determina la operación entre las mencionadas con un busmux dependiendo del bit menos significativo del código de operación.

El recuadro rojo se utiliza para determinar que flags deben ser utilizados en la salida del registro. Para todas las operaciones excepto para el Nop, Set y Clear los flags de V y C deben mantenerse sin modificar y los flags N y Z dependen de la respuesta. Para separar primero los flags de la operación Nop (que deben mantenerse todos estos sin modificar) utilizamos un multiplexor que como entrada tiene el código de operación interna del nop. Luego de pasar este multiplexor hay otro multiplexor que sirve para definir el bus utilizado para las operaciones de Set y Clear que a diferencia de las demás operaciones debe imponer e valor 0 para los flags V y C. Este multiplexor tiene como entrada una función que utiliza los dos bits mas significativos del código de operación interna que tanto para el Set como para el Clear son ambos 0 para seleccionar entre las entradas.

Finalmente en el diagrama se encuentran dos multiplexores. El primero encargado de definir si el valor que debe acomodarse en el registro proviene de una operación interna o de una operación realizada en el bloque ABALU. utilizando como selector la entrada I, proveniente del DecOp y como entradas la respuesta de la operación de la ABALU, que viene como entrada al bloque o la respuesta de la operación que se desarrollo en este mismo bloque.

El segundo multiplexor se encarga de cargar cuando se activa el LoadA el valor proveniente de memoria.

El registro se realizo utilizando FF de tipo D. Su diagrama es el siguiente:

Diagrama

Descripción generada automáticamente

## Bloque general de la ALU:

Diagrama, Esquemático

Descripción generada automáticamente

Este bloque esta compuesto por los bloques ya explicados anteriormente, siendo RegistroA el que contiene la lógica de las operaciones internas, ABALU el bloque que contiene las operaciones externas, DecOp el decodificador de operaciones explicado al principio de este informe y los registros que aparecen son el correspondiente al registro B en la izquierda y al registro que guarda los flags a la derecha. El multiplexor ubicado antes de la entrada del registro de Flags se utiliza para definir de que operación se tomaran los flags resultantes. Para actualizar el valor del registro B en un tiempo de clock valido se utiliza una compuerta AND para que este solo se actualice cuando la entrada LoadB se encuentre encendida.

Pruebas realizadas para corroborar el funcionamiento de cada una de las operaciones:

Prueba de Suma-Resta:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Prueba Set, CA2, NOT, CLEAR:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Prueba A?B, Nop, Inc, A=B:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

Prueba LSL, ASR, LSR y ASR nuevamente (para mostrar el funcionamiento con números positivos):

Escala de tiempo

Descripción generada automáticamente con confianza baja

Prueba AND, OR, XOR:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente