

**Práctica:** Diseño de un Sumador-Restador de 3 bits

**Alumnos:** Villanueva Contreras Omar Manuel G12211400

Madrazo Valladares Yoanky G18211246

**Profesor:** Dr. José Ricardo Cárdenas Valdez

Tijuana B.C. a 15 de octubre de 2018

ÍNDICE

[I. INTRODUCCIÓN 1](#_Toc526845021)

[II. DESARROLLO 2](#_Toc526845022)

[III. RESULTADOS 11](#_Toc526845023)

[IV. CONCLUSIÓN 15](#_Toc526845024)

[REFERENCIAS 16](#_Toc526845025)

[ANEXO A 17](#_Toc526845026)

# I. INTRODUCCIÓN

Las operaciones matemáticas son la base para el procesamiento computacional y el procesamiento electrónico en maquinaria electrónica e industrial. Pero estas operaciones siempre han representado una problemática a la hora del diseño de los circuitos electrónicos que realizan estas operaciones, sobre todo en tiempos recientes donde se prioriza velocidad y reducción de tamaño por lo que es necesario la investigación y diseño para proponer nuevos diseños que cubran estas necesidades sin sacrificar las características actuales para ganar nuevas ventajas.

Los sumadores son la base de las operaciones matemáticas en los circuitos electrónicos actuales, de estos los hay de distintos tipos y con diferentes características, sin importar estas lo que interesa en la actualidad es la mejora de los mismos. Un ejemplo de esto es el diseño y la implementación de sumadores de 32 bits con varios sumadores completos, 32 bits es una arquitectura muy común en la actualidad por lo que se diseñaron varios tales como: RCA, CINA y CBYA en verilog HDL, donde estas tres implementaciones mostraron resultados de prueba satisfactorios verificados en Xilinx 14.5 ISE en un dispositivo de la familia Spartan 3E [1]. No solo se busca mejorar los sumadores de gran cantidad de bits también los de menor cantidad como es el caso de un sumador completo de 1 bit donde no solo se logro reducir la cantidad de energía que consume, también se aumento la velocidad y se redujo el tamaño a solo 45 nanómetros, las simulaciones muestran buenos resultados no solo en general sino también con variaciones de voltaje [2]. A parte de buscar mejorar los sumadores actuales con tecnología CMOS, también se están buscando alternativas para superar la barrera de reducción de tamaño que se acerca, para ello por medio de autómatas celulares de puntos cuánticos se proponen diseños de sumadores paralelos eficientes en el manejo de área, las simulaciones de estos sumadores muestran mejora de hasta 64% en algunos casos de prueba simulados [3]. Si bien es cierto que se buscan nuevas tecnologías para la mejora de los sumadores, también se obtienen buenos resultados de simulaciones con tecnologías nuevas basadas en la actual como es el caso de un sumador con compuertas XOR-XNOR donde se genero una mejora de hasta 43% porciento comparado con sumadores similares usados actualmente [4]. El manejo de la energía es muy importante cuando se habla de los componentes nanoelectrónicos es por ello que el diseño de sumadores que sean eficientes en energía a esta escala son importantes, este diseño propuesto está basado en transistores de nanotubos de carbono, las simulaciones muestran mejora del 57% en términos de energía en comparación con otros similares [5].

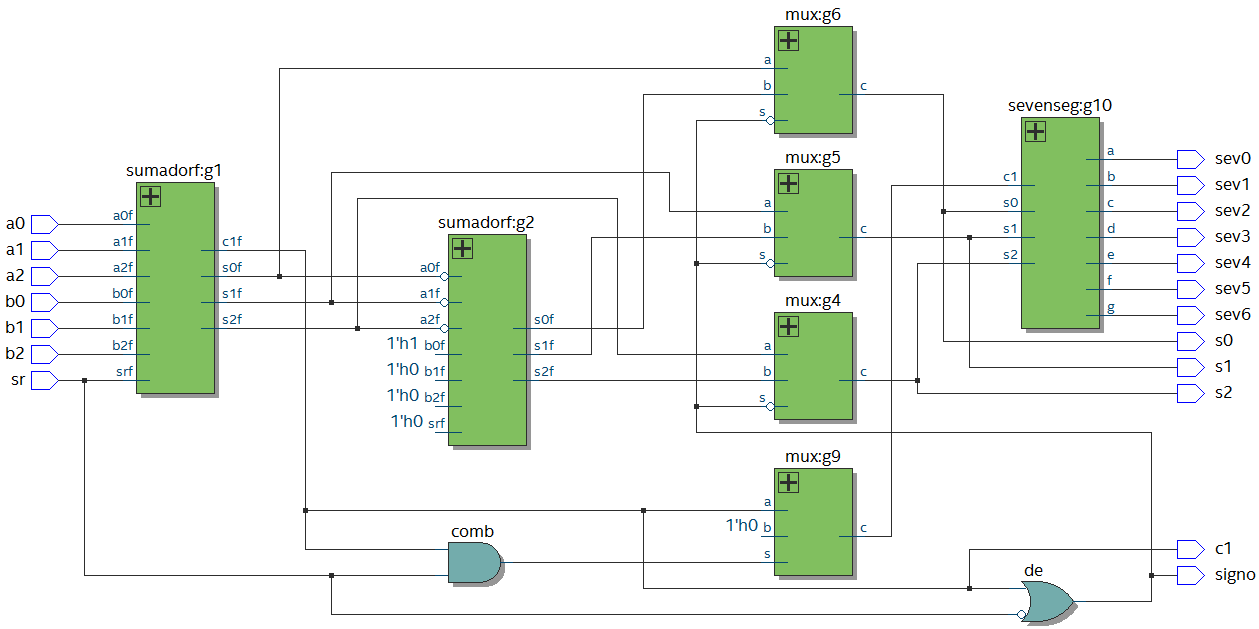
Gran cantidad de computadoras, maquinaria, circuitos, etc. contienen unidades aritméticas lógicas. La presente práctica muestra el panorama para el diseño de un sumador- restador. Estas dos operaciones constituyen la base para realizar todas las operaciones matemáticas. todo esto para una mejora en la comprensión de la teoría y ver una muestra de lo aprendido en un aspecto físico con la implementación en un FPGA.

# II. DESARROLLO

Un sumador completo es un circuito lógico que realiza la operación de adición en 3 números binarios de un bit, dos entradas y el valor de acarreo. Puede combinarse con otros sumadores completos para realizar sumas de mayor numero de bits.

*La práctica*

Realizar un sumador-restador de 2 números binarios de 3 bits cada uno, mostrar el resultado en leds en binario en el FPGA y en un display de 7 segmentos. Esta implementación se realizo por medio de dos entidades sumadoras de 3 bits para la suma y/o la resta, multiplexores para la selección de la salida correcta, una entidad convertidora de binario a 7 segmentos y sumadores completos de un bit como (que alambrando en cadena forman los sumadores de 3 bits), finalmente unas compuertas extras ayudan al proceso de decisión con el bit de acarreo del primer sumador y si es necesario mostrar el signo en los displays de 7 segmentos. La figura 1 muestra el diagrama de conexión de los componentes descritos en este párrafo.



**Fig. 1** Diagrama RTL del sumador-restador.

Cada parte de la práctica será explicada por medio de las entidades que se realizaron en VHDL (4) desde la mas simple hasta la entidad principal.

*Multiplexor*

Consiste en un multiplexor de dos canales de un bit cada uno, utilizado para la selección en las salidas (LEDS, 7 segmentos) si se muestra la salida normal o la salida con complemento a uno para la resta. La tabla numero 1 nos muestra las entradas y salidas del multiplexor.

**Tabla 1**. Entradas y salidas del multiplexor.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Tipo** | **Tipo VHDL** | **Descripción** |
| a | Entrada | std\_logic | Entrada de canal #1 |
| b | Entrada | std\_logic | Entrada de canal #2 |
| s | Entrada | std\_logic | Selección de canal |
| c | Salida | std\_logic | Canal seleccionado |
| CAND | Salida | std\_logic | Salida XOR |

El código VHDL de la arquitectura del multiplexor corresponde a la selección del canal adecuado por medio de un case. La figura 2 muestra el diagrama RTL del multiplexor.

PROCESS (s, a, b) IS

BEGIN

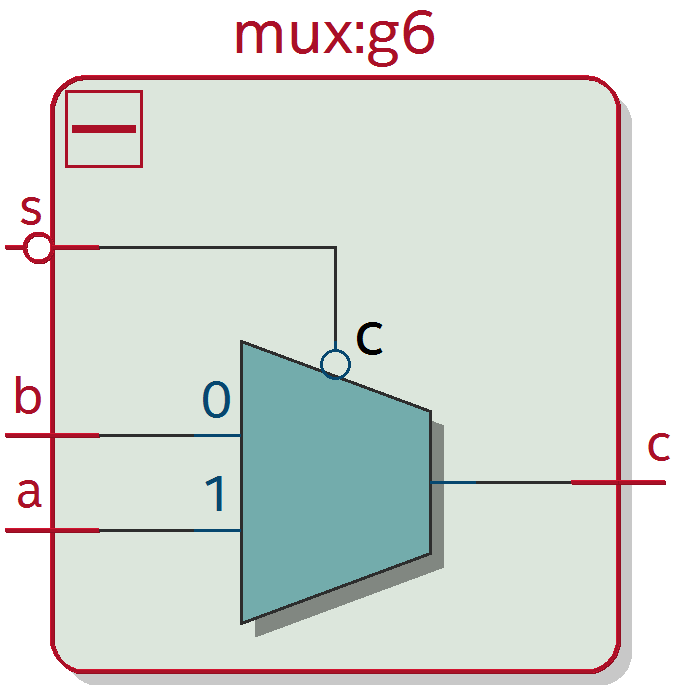
CASE s IS

WHEN '0' => c <= a;

WHEN '1' => c <= b;

END CASE;

END PROCESS;



**Fig. 2** Diagrama RTL del multiplexor.

*Sumador un bit*

Es un sumador completo de un bit con entrada de acarreo, utilizado para la elaboración de un sumador de 3 bits con la conexión en cadena de 3 de un bit. La tabla numero 2 describe las entradas y salidas del sumador.

**Tabla 2**. Entradas y salidas del sumador de un bit.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Tipo** | **Tipo VHDL** | **Descripción** |
| A | Entrada | std\_logic | Bit #1 |
| B | Entrada | std\_logic | Bit #2 |
| Ci | Entrada | std\_logic | Bit de acarreo de entrada |
| S | Salida | std\_logic | Resultado de suma |
| Co | Salida | std\_logic | Bit de acarreo de salida |

El código VHDL de la arquitectura del sumador nos muestra el alambrado estándar de uno de ellos por medio de 2 compuertas XOR, 2 AND y un OR. En la figura 3 se puede apreciar el alambrado de estas compuertas por medio de un diagrama RTL.

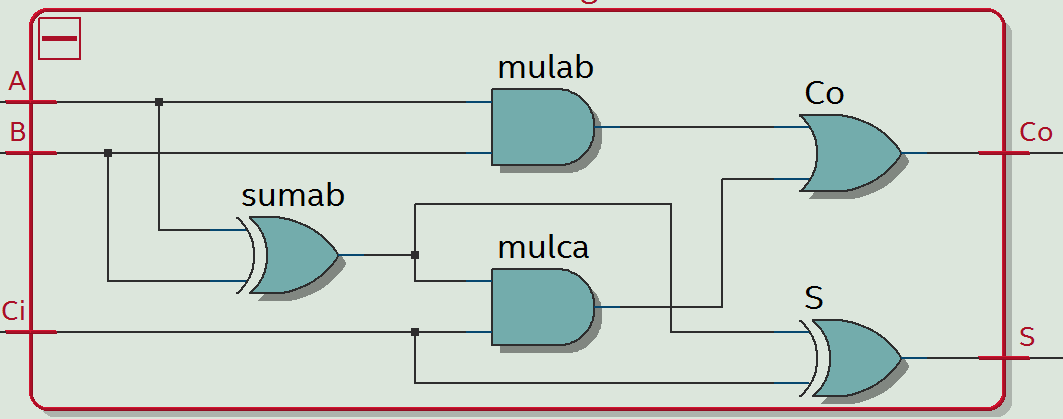
sumab <= A xor B;

mulab <= A AND B;

S <= sumab xor Ci;

mulca <= Ci AND sumab;

Co <= mulca or mulab;



**Fig. 3** Diagrama RTL del sumador de un bit.

*Convertidor a 7 segmentos*

Convierte una entrada de 3 bits binarios a 7 segmentos correspondientes. La tabla numero 3 describe las entradas y salidas del convertidor.

**Tabla 3**. Entradas y salidas del convertir a 7 segmentos.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Tipo** | **Tipo VHDL** | **Descripción** |
| s2 | Entrada | std\_logic | Bit binario 22 |
| s1 | Entrada | std\_logic | Bit binario 21 |
| s0 | Entrada | std\_logic | Bit binario 20 |
| c1 | Entrada | std\_logic | Bit de acarreo |
| a | Salida | std\_logic | Segmento a |
| b | Salida | std\_logic | Segmento b |
| c | Salida | std\_logic | Segmento c |
| d | Salida | std\_logic | Segmento d |
| e | Salida | std\_logic | Segmento e |
| f | Salida | std\_logic | Segmento f |
| g | Salida | std\_logic | Segmento g |

El código VHDL de la arquitectura del convertidor corresponde a las compuertas reducidas en los mapas de Karnaugh descritos en las tablas 4-11 y en las ecuaciones 1-7.

a <= NOT(((NOT s2) and (NOT s0)) or ((NOT c1) and s1) or (c1 and (NOT s0)) or ((NOT c1) and s2 and s0) or (c1 and (NOT s2) and (NOT s1)) );

b <= NOT(((NOT c1) and (NOT s2)) or ((NOT s2) and (not s0)) or ((NOT c1) and (not s1) and (not s0)) or ((NOT c1) and s1 and s0) or (c1 and (NOT s1) and s0) );

c <= NOT(s0 or ((NOT c1) and (not s1)) or ((NOT c1) and s2) or (c1 and (not s2)));

d <= NOT((c1 and s2) or ((NOT s2) and (not s1) and (not s0)) or ((NOT c1) and s1 and (not s0)) or ((NOT s2) and s1 and s0) or (s2 and (NOT s1) and s0) );

e <= NOT(((NOT s2) and (NOT s0)) or ((NOT s0) and s1) or (c1 and s1) or (c1 and s2) );

f <= NOT(((NOT s1) and (NOT s0)) or ((NOT s0) and s2) or (c1 and (NOT s2)) or (s2 and (NOT c1) and (NOT s1)) );

g <= NOT(((NOT s2) and s1) or ((NOT s0) and s1) or (c1 and (NOT s2)) or (c1 and s0) or (s2 and (NOT c1) and (NOT s1)) );

**Tabla 4**. Tabla de entradas y salidas lógicas esperadas en el convertidor 7 segmentos

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **c1** | **s2** | **s1** | **s2** | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | x | x | x | x | x | x | x |

**Tabla 5**. Mapa Karnaugh de segmento a

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **s1s0**  **c1s2** | **00** | **01** | **11** | **10** |
| **00** | 1 | 0 | 1 | 1 |
| **01** | 0 | 1 | 1 | 1 |
| **11** | 1 | 0 | x | 1 |
| **10** | 1 | 1 | 0 | 1 |

**Tabla 6**. Mapa Karnaugh de segmento b

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **s1s0**  **c1s2** | **00** | **01** | **11** | **10** |
| **00** | 1 | 1 | 1 | 1 |
| **01** | 1 | 0 | 1 | 0 |
| **11** | 0 | 1 | x | 0 |
| **10** | 1 | 1 | 0 | 1 |

**Tabla 7**. Mapa Karnaugh de segmento c

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **s1s0**  **c1s2** | **00** | **01** | **11** | **10** |
| **00** | 1 | 1 | 1 | 0 |
| **01** | 1 | 1 | 1 | 1 |
| **11** | 0 | 1 | x | 0 |
| **10** | 1 | 1 | 1 | 1 |

**Tabla 8**. Mapa Karnaugh de segmento d

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **s1s0**  **c1s2** | **00** | **01** | **11** | **10** |
| **00** | 1 | 0 | 1 | 1 |
| **01** | 0 | 1 | 0 | 1 |
| **11** | 1 | 1 | x | 1 |
| **10** | 1 | 0 | 1 | 0 |

**Tabla 9**. Mapa Karnaugh de segmento e

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **s1s0**  **c1s2** | **00** | **01** | **11** | **10** |
| **00** | 1 | 0 | 0 | 1 |
| **01** | 0 | 0 | 0 | 1 |
| **11** | 1 | 1 | x | 1 |
| **10** | 1 | 0 | 1 | 1 |

**Tabla 10**. Mapa Karnaugh de segmento f

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **s1s0**  **c1s2** | **00** | **01** | **11** | **10** |
| **00** | 1 | 0 | 0 | 0 |
| **01** | 1 | 1 | 0 | 1 |
| **11** | 1 | 0 | X | 1 |
| **10** | 1 | 1 | 1 | 1 |

**Tabla 11**. Mapa Karnaugh de segmento g

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **s1s0**  **c1s2** | **00** | **01** | **11** | **10** |
| **00** | 0 | 0 | 1 | 1 |
| **01** | 1 | 1 | 0 | 1 |
| **11** | 0 | 1 | x | 1 |
| **10** | 1 | 1 | 1 | 1 |

De las tablas 5-11 se obtienen las siguientes ecuaciones

(1)

(2)

(3)

(4)

(5)

(6)

(7)

*Sumador de 3 bits*

Es un sumador de 3 bits con la capacidad de realizar un switch para poner las entradas en complemento a 1 para realizar la resta a medias (sin el complemento a 1 en el resultado). La tabla numero 12 describe las entradas y salidas del sumador.

**Tabla 12**. Entradas y salidas del sumador de 3 bits.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Tipo** | **Tipo VHDL** | **Descripción** |
| a2f | Entrada | std\_logic | Bit binario 22 deentrada #1 |
| a1f | Entrada | std\_logic | Bit binario 21 deentrada #1 |
| a0f | Entrada | std\_logic | Bit binario 20 deentrada #1 |
| b2f | Entrada | std\_logic | Bit binario 22 deentrada #2 |
| b1f | Entrada | std\_logic | Bit binario 21 deentrada #2 |
| b0f | Entrada | std\_logic | Bit binario 20 deentrada #2 |
| srf | Entrada | std\_logic | Selector suma/resta |
| c1f | Salida | std\_logic | Bit de acarreo |
| s2f | Salida | std\_logic | Salida bit binario 22 |
| s1f | Salida | std\_logic | Salida bit binario 21 |
| s0f | Salida | std\_logic | Salida bit binario 20 |

El código VHDL de la arquitectura del sumador de 3 bits nos muestra el alambrado en cadena de los sumadores de un bit y las compuertas XOR para la selección de suma-resta En la figura 4 se puede apreciar el alambrado de estos componentes por medio de un diagrama RTL.

b0r<= b0f xor srf;

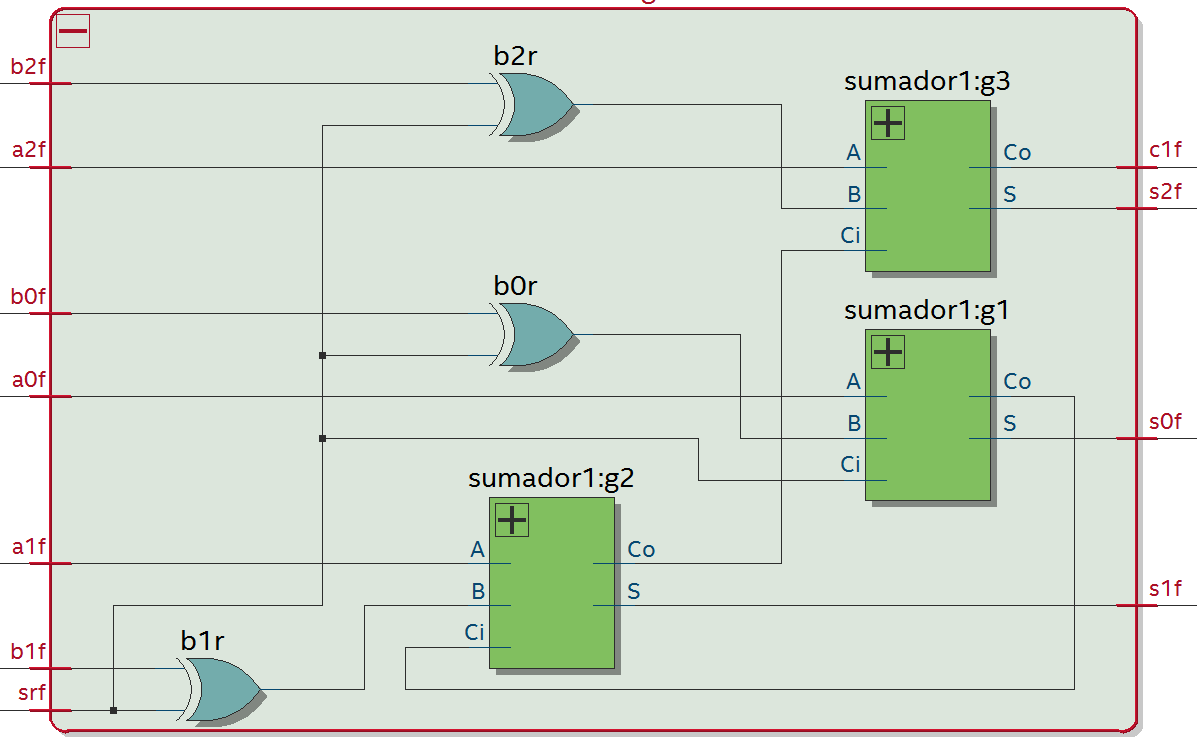
b1r<= b1f xor srf;

b2r<= b2f xor srf;

g1: sumador1 port map (A=>a0f,B=>b0r,Ci=>srf,S=>s0f,Co=>temp1);

g2: sumador1 port map (A=>a1f,B=>b1r,Ci=>temp1,S=>s1f,Co=>temp2);

g3: sumador1 port map (A=>a2f,B=>b2r,Ci=>temp2,S=>s2f,Co=>C1f);



**Fig. 4** Diagrama RTL del sumador de un bit.

*Entidad Principal*

Es el alambrado del proceso de implementación descrito en el segundo párrafo de esta sección. La tabla número 13 describe las entradas y salidas de la entidad principal.

**Tabla 13**. Entradas y salidas de la entidad principal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Tipo** | **Tipo VHDL** | **Descripción** |
| a2 | Entrada | std\_logic | Bit binario 22 deentrada #1 |
| a1 | Entrada | std\_logic | Bit binario 21 deentrada #1 |
| a0 | Entrada | std\_logic | Bit binario 20 deentrada #1 |
| b2 | Entrada | std\_logic | Bit binario 22 deentrada #2 |
| b1 | Entrada | std\_logic | Bit binario 21 deentrada #2 |
| b0 | Entrada | std\_logic | Bit binario 20 deentrada #2 |
| sr | Entrada | std\_logic | Selector suma/resta |
| c1 | Salida | std\_logic | Bit de acarreo |
| s2 | Salida | std\_logic | Salida bit binario 22 |
| s1 | Salida | std\_logic | Salida bit binario 21 |
| s0 | Salida | std\_logic | Salida bit binario 20 |
| signo | Salida | std\_logic | Signo para display 7seg |
| sev0 | Salida | std\_logic | Segmento a |
| sev1 | Salida | std\_logic | Segmento b |
| sev2 | Salida | std\_logic | Segmento c |
| sev3 | Salida | std\_logic | Segmento d |
| sev4 | Salida | std\_logic | Segmento e |
| sev5 | Salida | std\_logic | Segmento f |
| sev6 | Salida | std\_logic | Segmento g |

El código VHDL de la arquitectura de la entidad principal es el alambrado descrito en el segundo párrafo de esta sección.

g1: sumadorf port map (a2f=>a2,b2f=>b2,a1f=>a1,b1f=>b1,a0f=>a0,b0f=>b0,srf=>sr,c1f=>c1t,s2f=>s2t,s1f=>s1t,s0f=>s0t);

g2: sumadorf port map (a2f=>not s2t,b2f=>'0',a1f=>not s1t,b1f=>'0',a0f=>not s0t,b0f=>'1',srf=>'0',c1f=>d,s2f=>s2tf,s1f=>s1tf,s0f=>s0tf);

g3: de<= not ((not sr) or c1t);

g4: mux port map (a=>s2t,b=>s2tf,s=>de,c=>s27);

g5: mux port map (a=>s1t,b=>s1tf,s=>de,c=>s17);

g6: mux port map (a=>s0t,b=>s0tf,s=>de,c=>s07);

s2<=s27;

s1<=s17;

s0<=s07;

g7: c1<=c1t;

g8: signo<=not de;

g9: mux port map (a=>c1t,b=>'0',s=>c1t and sr ,c=>c7s);

g10: sevenseg port map (c1=>c7s,s2=>s27,s1=>s17,s0=>s07,a=>sev0, b=>sev1, c=>sev2, d=>sev3, e=>sev4, f=>sev5, g=>sev6);

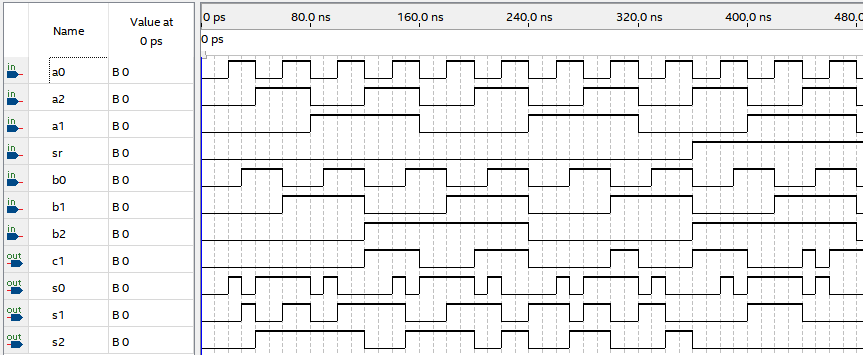
La asignación de pines se describe en la tabla 14.

**Tabla 15**. Asignación de pines para él contador.

|  |  |  |
| --- | --- | --- |
| **Nombre** | **PIN** | **Elemento** |
| a2 | AB28 | SW2 |
| a1 | Y27 | SW1 |
| a0 | AB30 | SW0 |
| b2 | V25 | SW5 |
| b1 | W25 | SW4 |
| b0 | AC30 | SW3 |
| sr | AA30 | SW9 |
| c1 | AD24 | LEDR3 |
| s2 | AC23 | LEDR2 |
| s1 | AB23 | LEDR1 |
| s0 | AA24 | LEDR0 |
| signo | V17 | HEX1(7) |
| sev0 | W17 | HEX0(1) |
| sev1 | V18 | HEX0(2) |
| sev2 | AG17 | HEX0(3) |
| sev3 | AG16 | HEX0(4) |
| sev4 | AH17 | HEX0(5) |
| sev5 | AG18 | HEX0(6) |
| sev6 | AH18 | HEX0(7) |

# III. RESULTADOS

Primeramente, se utiliza el Model Sim de Quartus Prime para asegurarnos que según los valores de entrada serán los valores de salida esperados, en esta etapa se configuran los rangos de valores de las variables de entrada consecuentemente de manera que se cumplan todas las combinaciones reales, así se muestra en la figura 3.1.



**Fig. 3.1** Diagrama de salida del Model Sim

Posteriormente se descarga la programación a la tarjeta después de constatar los resultados del Model Sim. A continuación, se muestra en la figura 3.2 las posiciones de los pines y el resultado en los leds y el displays de 7 segmentos, donde en este caso no se activará el swith de resta y se sumaran los valores de los swith. Se muestra en la tabla 3.1 los swith y leds activados.

**Tabla 3.1** Diagrama de swith y leds activados.

|  |  |  |  |
| --- | --- | --- | --- |
| **in** | **out** | **Swith de der-izq** | **Leds de der-izq** |
| a(0)=1 | -- | 1 | - |
| b(0)=1 | -- | 4 | - |
| sr=0 | suma | 10 | - |
|  | s(2)=2 |  | 2 |
|  | sev(0,1.3,4,6)=2 |  | Displays 1 |

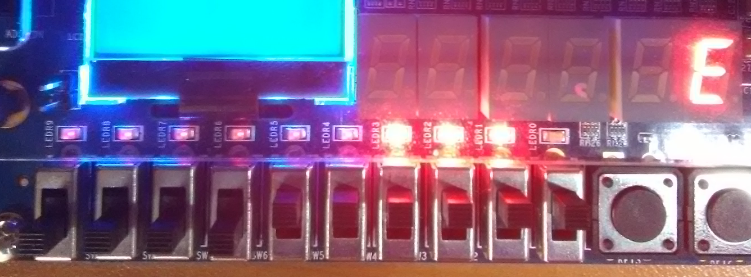


**Fig. 3.2** Diagrama de salida del Model Sim para la suma de a(0) + b(0).

En la tabla 3.2 se muestra los valores de los swith y leds donde la resta no está activada y se suman todas las entradas del vector a + vector b, se muestran los resultados en la figura 3.3.

**Tabla 3.2** Diagrama de swith y leds activados.

|  |  |  |  |
| --- | --- | --- | --- |
| **in** | **out** | **Swith de der-izq** | **Leds de der-izq** |
| a(0,1,2)=7 | -- | 1,2,3 | - |
| b(0,1,2)=7 | -- | 4,5,6 | - |
| sr=0 | suma | 10 | - |
|  | s(2,3,4)=14 |  | 2,3,4 |
|  | sev(0,3,4,5,6)=E |  | Displays 1 |



**Fig. 3.3** Diagrama de salida del Model Sim para la suma de a(0,1,2) + b(0,1,2).

En la tabla 3.3 se muestra los valores de los swith y leds donde la resta si está activada y se le resta el vector a al vector b, para este caso se activan todas las entradas del vector a y b y el resultado es cero, se muestran los resultados en la figura 3.4.

**Tabla 3.3** Diagrama de swith y leds activados.

|  |  |  |  |
| --- | --- | --- | --- |
| **in** | **Out** | **Swith de der-izq** | **Leds de der-izq** |
| a(0,1,2)=7 | -- | 1,2,3 | - |
| b(0,1,2)=7 | -- | 4,5,6 | - |
| sr=1 | resta (a-b) | 10 | - |
|  | s(-)=0 |  | ninguno |
|  | sev(0,1,2,3,4,5)=0 |  | displays 1 |



**Fig. 3.4** Diagrama de salida del Model Sim para la suma de a(0,1,2) - b(0,1,2).

En la tabla 3.4 se muestra los valores de los swith y leds donde la resta si está activada y se le resta al vector el vector b, para este caso se activan todas las entradas del vector b y el uno y dos del vector a, el resultado es -2, se muestran los resultados en la figura 3.5.

**Tabla 3.4** Diagrama de swith y leds activados.

|  |  |  |  |
| --- | --- | --- | --- |
| **in** | **out** | **Swith de der-izq** | **Leds de der-izq** |
| a(0,2)=5 | -- | 1,3 | - |
| b(0,1,2)=7 | -- | 4,5,6 | - |
| sr=1 | resta (a-b) | 10 | - |
|  | s(2) =2 |  | 2 |
|  | sev(0,1,3,4,6)=-2 |  | displays 1 |



**Fig. 3.5** Diagrama de salida del Model Sim para la suma de b(0,1,2) - a(0,2).

# IV. CONCLUSIÓN

*Madrazo*

El trabajo con la tarjeta Cyclone V de Altera es usado en la actualidad por las prestaciones que brinda, es una red de compuertas programables en campo (FPGA) con un procesador ARM® integrado que permite el diseño flexible de hardware periférico, esto ayuda a la formación de los estudiantes de postgrado y pregrado porque eleva su formación académica y permite el estudio y desarrollo en ramas de la ingeniería con problemas reales que pueden ser resueltos con implementaciones de baja o alta complejidad, para el trabajo presentado se usaron las entradas digitales por switch y para la salida se usaron los led y el displays de 7 segmentos de manera que con unas aplicaciones simples se pueda probar, estudiar y profundizar en la comunicación con las entradas y salidas de la Cyclone V, se usa como forma de comprobar el resultado el Model Sim de Quartus Prime para simular el comportamiento de todos los bloques RTL programados y poder corroborar todos las combinaciones de entrada y salida posibles, esto ayuda la comprensión y asegura el buen funcionamiento posteriormente de la lógica de programación en la tarjeta FPGA.

*Villanueva*

VHDL es otra manera de realizar el diseño de circuitos digitales, la programación en este lenguaje descriptivo requiere de una lógica especial similar a la programación de las ciencias computacionales. Las aplicaciones de VHDL tienen un gran rango de aplicaciones donde no solo sirven para el diseño de futuras implementaciones de circuitos experimentales sino también ayudan en la mejora de sistemas actuales. El diseño en VHDL si bien al principio resulta un tanto confuso por el cambio de paradigma a la hora de programar, su implementación en un medio físico (FPGA) para poder ver los resultados de forma tangible ayuda a la comprensión del tema. No solo ayuda a la comprensión de VHDL, también refuerza otros conceptos referentes a la electrónica digital. La programación de circuitos en VHDL nos ofrece otras ventajas como la posibilidad de visualizar los circuitos en diagramas RTL, la simulación de estos circuitos dependiendo de la herramienta que se este utilizando para la programación VHDL y si esta soporta estos complementos adicionales. La programación VHDL tiene un gran potencial que se ha estado utilizando en estos últimos años y aún hay por descubrir/codificar, nos queda a nosotros decidir si es que se quiere aplicar esta herramienta o buscar alternativas similares con las mismas ventajas u otras diferentes pero que al final lleguen al mismo resultado.

# REFERENCIAS

1. Maurya K., Lakshmanna Y., Sindhuri K. and Kumar N., "Design and implementation of 32-bit adders using various full adders," in 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), pp. 1-6, Vellore, India, 2017
2. Yadav T., Shrivatava B. and Dadoriya A., "Low power high speed 1-bit full adder circuit design at 45nm CMOS technology," in 2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE), pp. 427-432, Bhopal, India, 2017
3. Ramesh B. and Rani M., "Implementation of parallel adders using area efficient quantum dot cellular automata full adder," in 2016 10th International Conference on Intelligent Systems and Control (ISCO), pp. 1-5, Coimbatore, India, 2016
4. Naseri H. and Timarchi S., "Low-Power and Fast Full Adder by Exploring New XOR and XNOR Gates," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 8, pp. 1481-1493
5. Sedighiani S. and Kazemi A., “An Energy-Efficient Quaternary Serial Adder for Nanoelectronics” in 2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL), pp. 44-49, Linz, Austria, 2018

# ANEXO A, FIRMA

