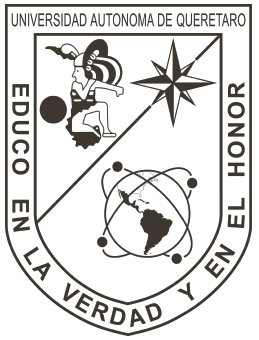
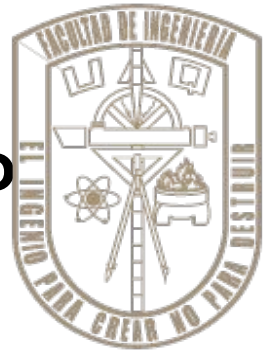


Universidad Autónoma de Querétaro

Facultad de Ingeniería

Ingeniería en Automatización

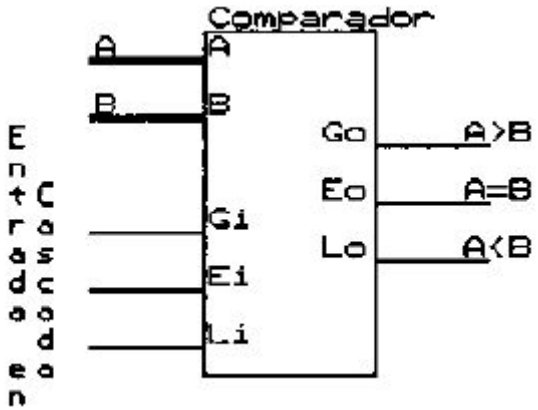


SDLR – Sumador, comparador, restador y multiplicador

M en C. Marcos Romo Avilés

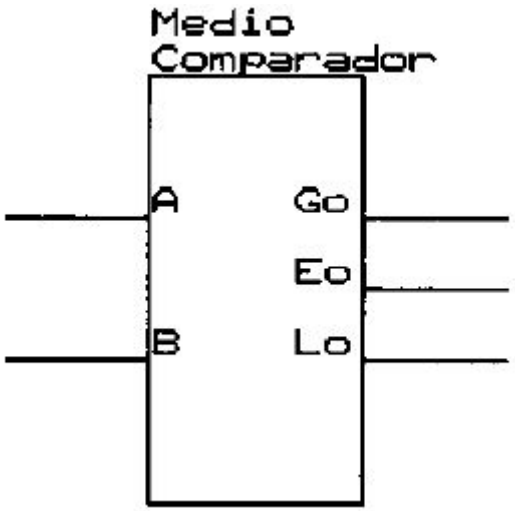
Practica 3: Comparador

- comparador realiza la prueba entre dos palabras binarias del mismo formato y código, y determina si el resultado es mayor, menor o igual. Al no ser conmutativa la comparaci6n, se debe especificar el orden de la comparación.
- El comparador de la figura realiza la comparación del operando A con respecto al operando B, permitiendo que se incorpore información generada de manera previa para procesar el resultado de forma algorítmica, muy parecida a la tecnica de papel y lápiz.



Practica 3: Comparador

- Se analiza primero el media comparador de 1 bit, es decir, el comparador de palabras de 1 bit que no tiene entradas de comparación de mayor significación. Este media comparador se muestra en la figura.
- La relación lógica en el media comparador aparece en la tabla.
- Las ecuaciones lógicas que rigen al medio comparador vienen dadas por:



A	B	Mayor Go	Igual Eo	Menor Lo
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$$G_o = AB'$$

$$E_o = (A \oplus B)'$$

$$L_o = A'B$$

Practica 3: Comparador

- El comparador completo de 1 bit es como el que aparece en la figura, tomando A y B como palabras de 1 bit. En la tabla se encuentran las relaciones lógicas del comparador complete de 1 bit.
- Las relaciones lógicas que rigen al comparador completo de 1 bit se encuentran en las ecuaciones:

Gi	Ei	Li	A	B	Go	Eo	Lo
1	0	0	x	x	1	0	0
0	0	1	x	x	0	0	1
0	1	0	0	0	0	1	0
0	1	0	0	1	0	0	1
0	1	0	1	0	1	0	0
0	1	0	1	1	0	1	0

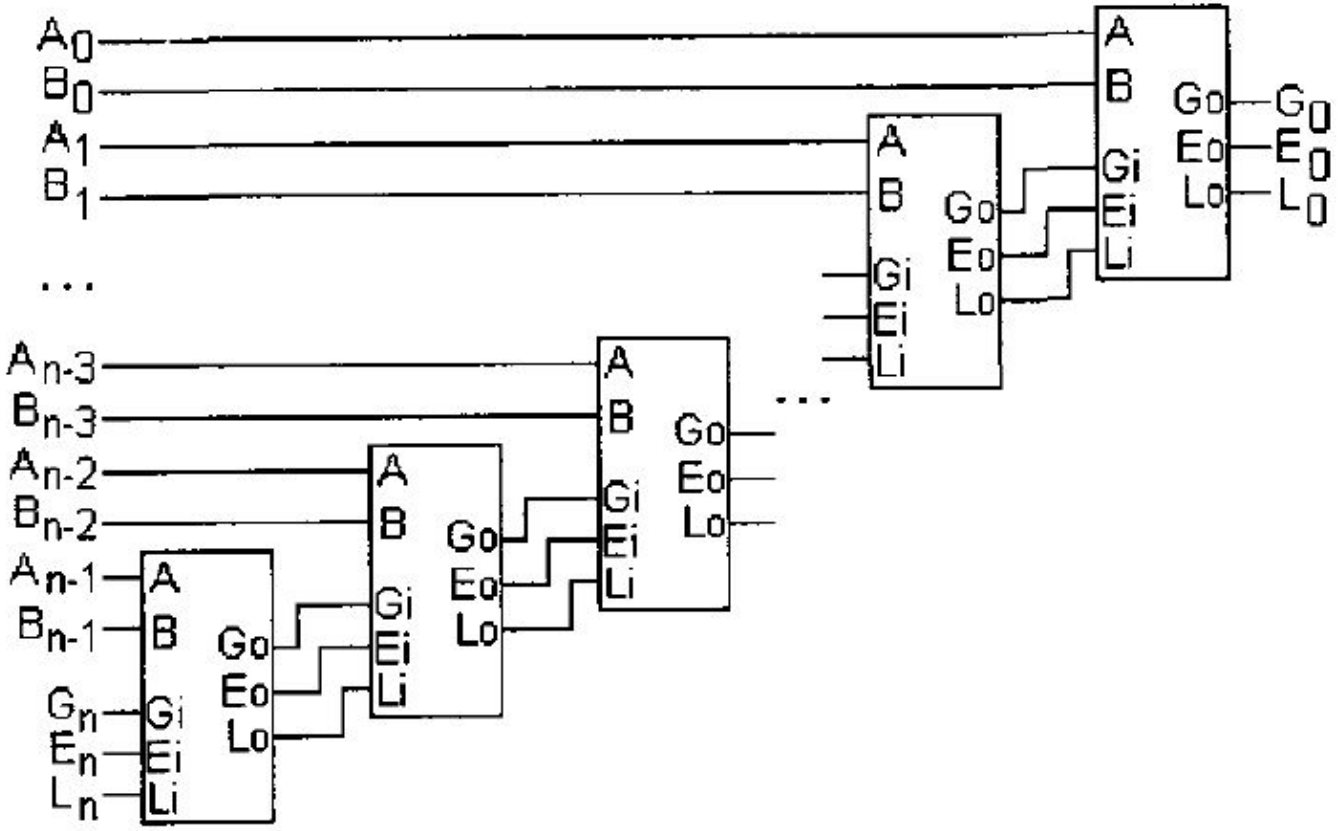
$$G_o = G_i + E_i AB'$$

$$E_o = E_i (A \oplus B)'$$

$$L_o = L_i + E_i A'B$$

Practica 3: Comparador

- El comparador completo de 1 bit puede ser conectado en cascada para obtener comparadores de palabras más grandes, tal como se muestra en la figura.

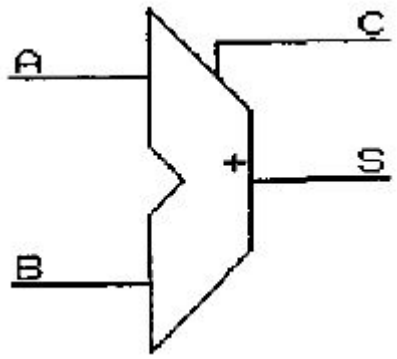


Practica 3: Comparador

- Generar un comparador de 5 bits
- Usar 5 switches para asignarle un valor a A y 5 para B.
- Generar la simulación.
- Mostrar el resultado en decimal en los 7 segmentos y en binario en los leds.

Practica 3: Sumador

- La operación aritmética más básica es la suma. El circuito que realiza la suma de dos números binarios se conoce como sumador binario.
- El elemento básico de los circuitos aritméticos es el medio sumador. Este circuito suma dos palabras de 1 bit y entrega como resultado la suma y acarreo correspondiente.



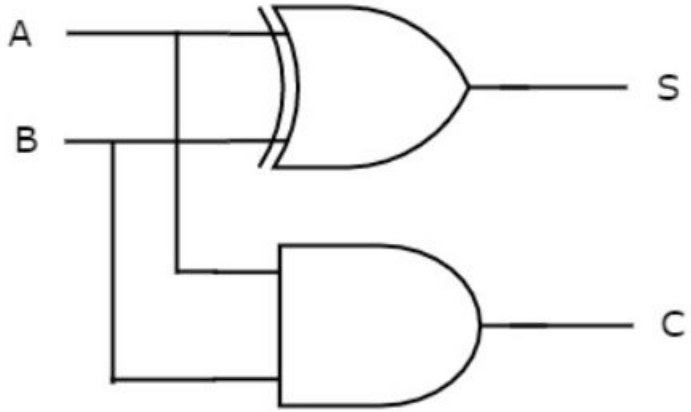
Entradas		Salidas	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Practica 3: Sumador

- Las funciones lógicas que describen al medio sumador se muestran en las ecuaciones:

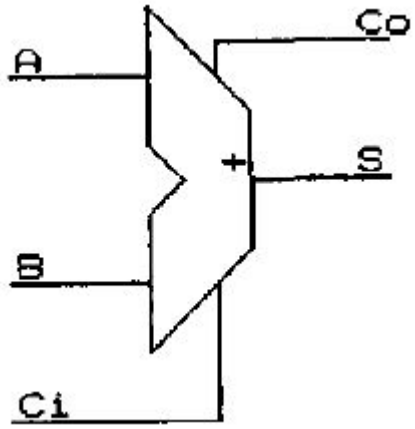
$$S = A \oplus B$$

$$C = A\,B$$



Practica 3: Sumador

- El circuito medio sumador no contempla la posibilidad de sumar en forma secuencial y para poder realizar esto, se necesita un sumador completo.
- El sumador completo realiza la suma de dos palabras de 1 bit y además agrega el acarreo producido en una etapa anterior.



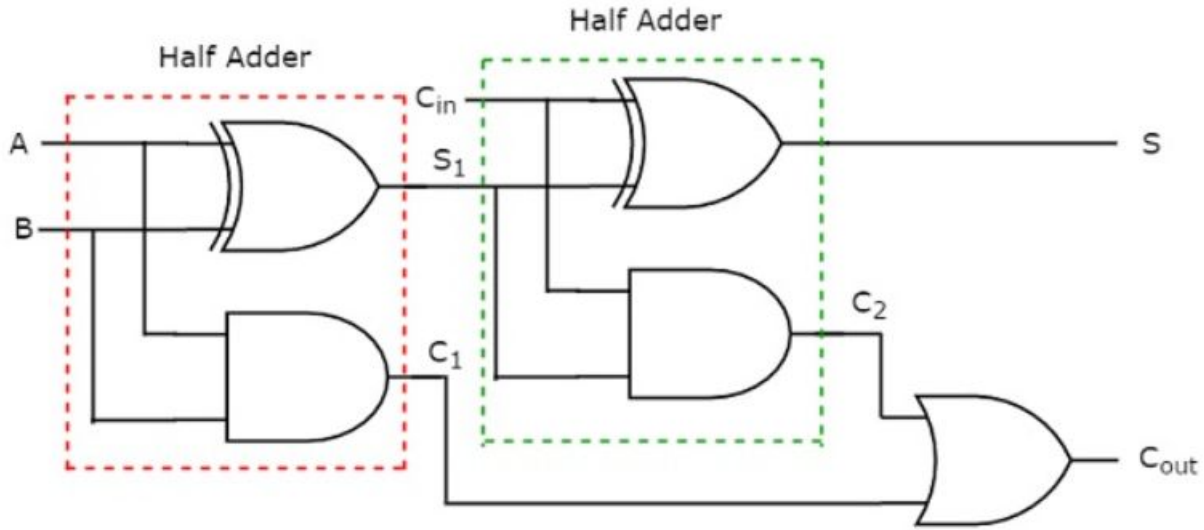
Entradas			Salidas	
A	B	C_{en}	C_{fuera}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Practica 3: Sumador

- Donde, S es bit menos significativo y lleve, Cout el bit más significativo de la suma resultante.

$$S = A \oplus B \oplus C_i$$

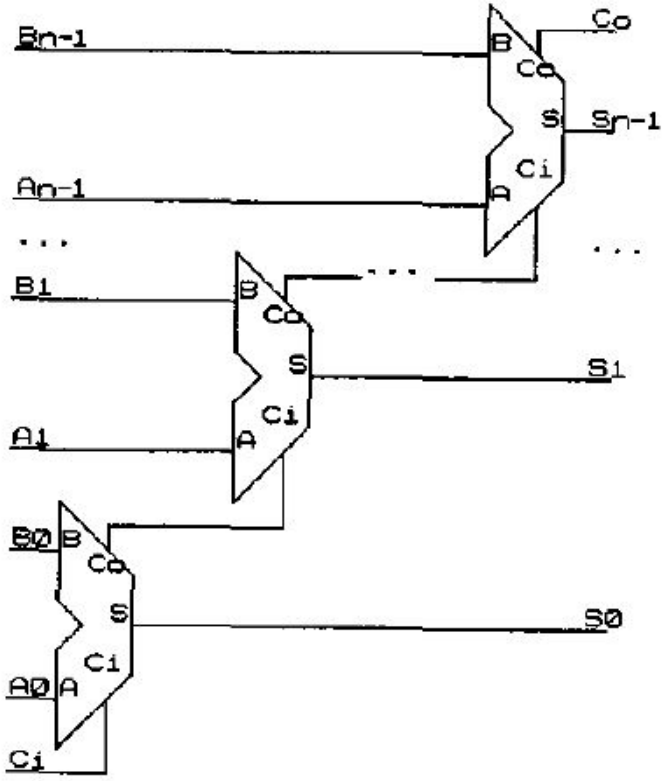
$$C_o = AB + AC_i + BC_i$$



Practica 3: Sumador

- El sumador de un solo bit no es practico en aplicaciones reales. El diseño de sumadores más grandes tiene dos formas principales de realizarlo:
  - Cascada
  - Acarreo adelantado
- En la Figura se muestra un sumador de n bits utilizando sumadores completos.
- La formación en cascada es simple y requiere pocos recursos. Sin embargo, presenta un retardo del circuito en función del número de bits (puede ser económico, pero resulta demasiado lento).

Figura 5.9. Sumador de *n* bits en cascada



Practica 3: Sumador

- Para el primer bit, o menos significativo, las ecuaciones lógicas correspondientes son las presentadas:

$$S_0 = A_0 \oplus B_0 \oplus C_0$$

$$C_1 = A_0B_0 + A_0C_0 + B_0C_0$$

- Para obtener los siguientes bits se utiliza la recursión de fórmulas de tal manera que:

$$S_1 = A_1 \oplus B_1 \oplus C_1$$

$$C_2 = A_1B_1 + A_1C_1 + B_1C_1$$

$$S_1 = A_1 \oplus B_1 \oplus (A_0B_0 + A_0C_0 + B_0C_0)$$

$$C_2 = A_1B_1 + A_1(A_0B_0 + A_0C_0 + B_0C_0) + B_1(A_0B_0 + A_0C_0 + B_0C_0)$$

Practica 3: Sumador

- Existe una solución denominada acarreo de propagación adelantada o CLA (Cany Look-Ahead).
- El retardo es mucho menor que en el sumador en cascada con la ventaja de un diseño de mediana complejidad.
- Esta tecnica consiste en generar un primer nivel lógico con funciones parciales y en base a estas, generar las funciones de la suma.
- Las funciones intermedias de un sumador con acarreo de propagación adelantada. Estas funciones se llaman generadora G_i y propagadora P_i.

$$G_i = A_i B_i$$

$$P_i = A_i \oplus B_i$$

Practica 3: Sumador

- A partir de estas funciones se pueden definir las operaciones de suma y de acarreo en un segundo nivel lógico utilizando las siguientes relaciones:

$$S_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + P_i C_i$$

- Para que el segundo nivel lógico permanezca en una realización de dos niveles de compuertas, se desarrollan, recursivamente, las expresiones para el acarreo, de tal forma que:

$$C_1 = G_0 + P_0 C_0$$

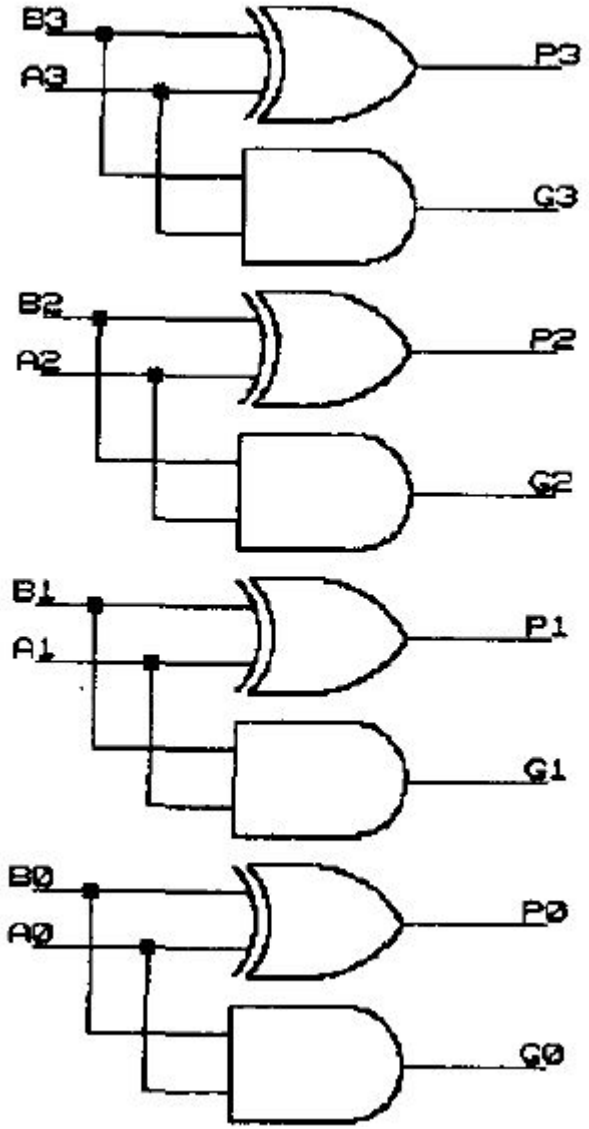
$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 \left(G_0 + P_0 C_0 \right) = G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

Practica 3: Sumador

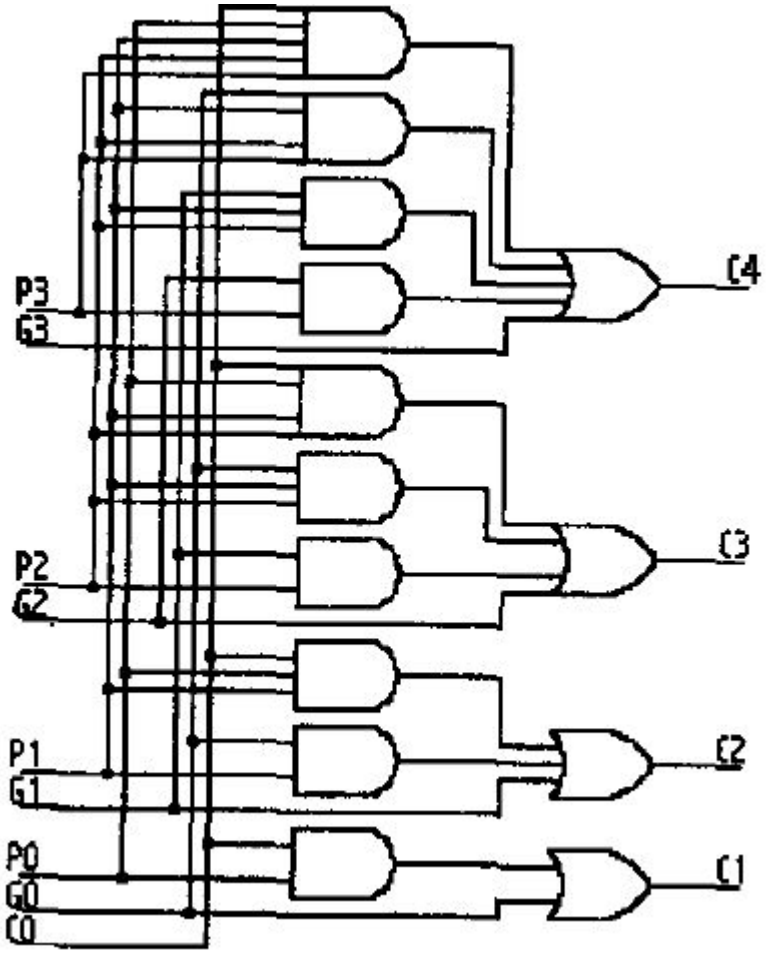
- Los bits sucesivos se pueden seguir calculando per recursión. Para ejemplificar el método CLA se precede al diseño de un sumador complete de cuatro bits con CLA.

Practica 3: Sumador



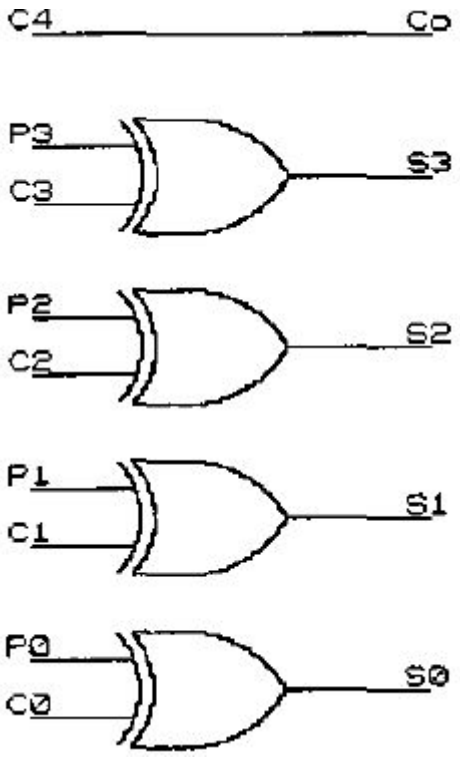
Practica 3: Sumador

- La sección del acarreo adelantado

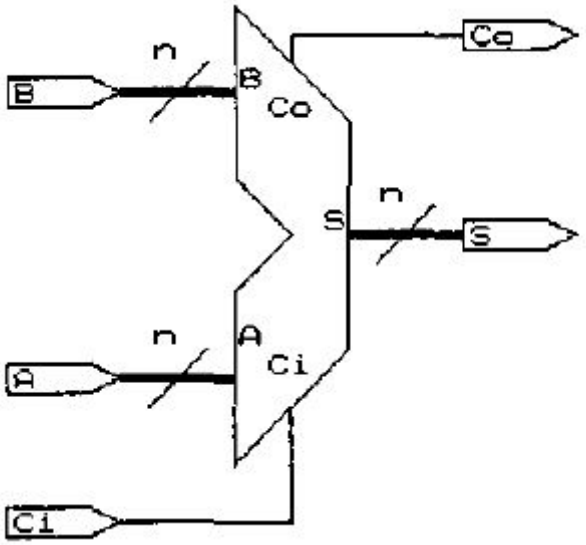


Practica 3: Sumador

- Sección de la suma



- El símbolo normalizado para un sumador complete de n bits



Practica 3: Sumador

- Generar un sumador en cascada y un sumador en CLA de 5 bits
- Usar 5 switches para asignarle un valor a A y 5 para B.
- Generar la simulación.
- Mostrar el resultado en decimal en los 7 segmentos y en binario en los leds.

Practica 3: Restador

- Al igual que la suma binaria, también se puede definir la operación de resta binaria. Como la resta no es conmutativa, se tiene que establecer el orden de los operandos.
- Primera se obtiene la funci6n lógica de un media restador de un bit para la operaci6n A-B.
- Para la resta, el resultado se obtiene en R y el acarreo se encuentra en β.

A	B	β	R
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$$R = A \oplus B$$

$$\beta = A' B$$

Practica 3: Restador

- La función lógica para un restador completo de 1 bit se muestra en la tabla.
- Las ecuaciones contienen las relaciones lógicas del restador completo de 1 bit.

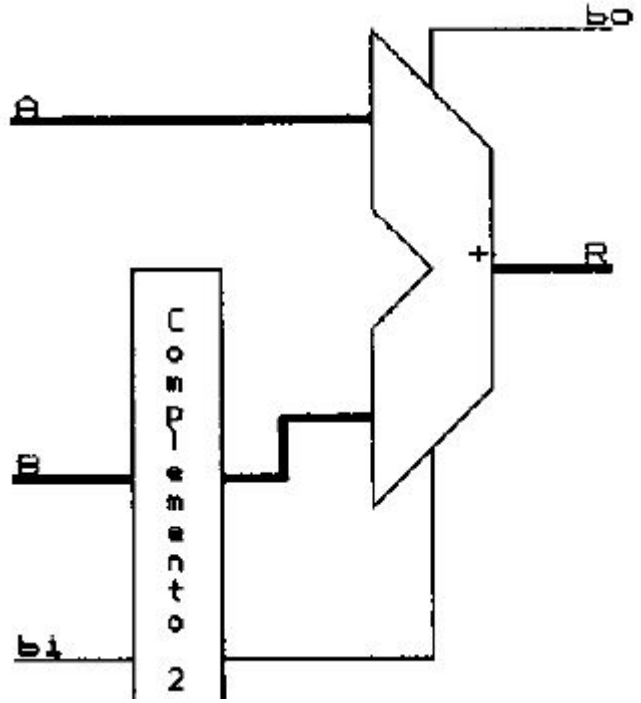
β_i	A	B	β_o	R
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

$$R = A \oplus B \oplus \beta_i$$

$$\beta_o = A'B + \beta_i A' + \beta_i B$$

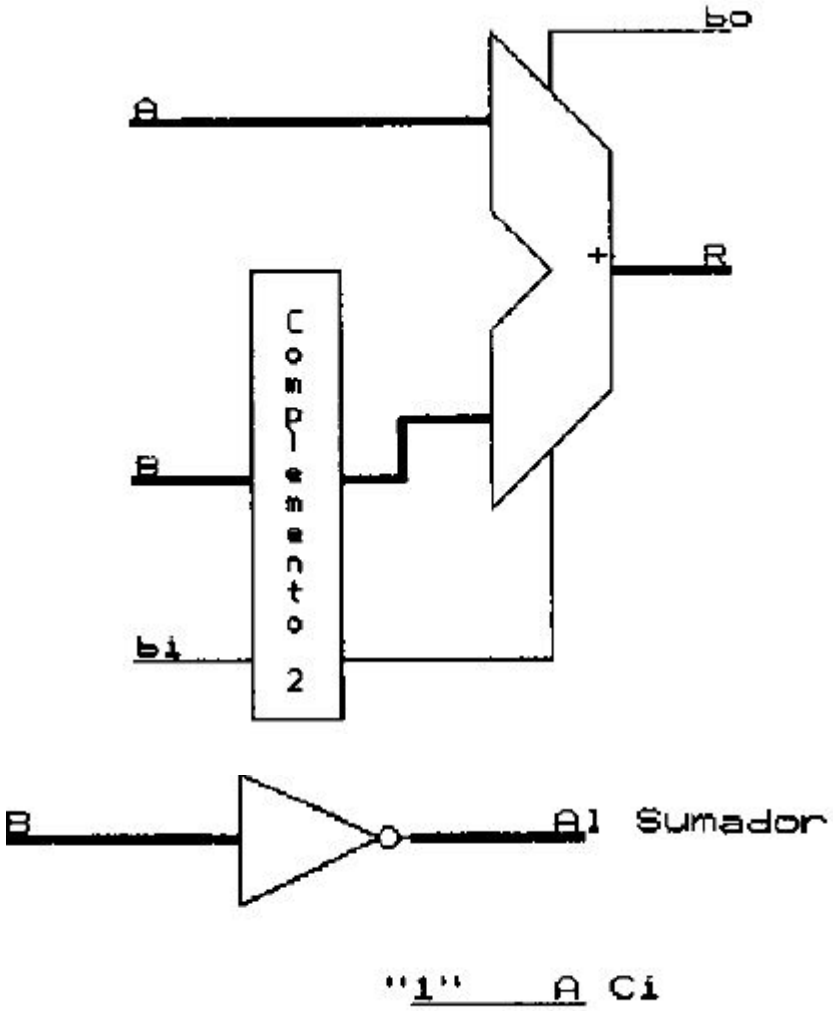
Practica 3: Restador

- Para la realización de restadores de un mayor número de bits se pueden utilizar las mismas técnicas utilizadas en las sumadores: conexión en cascada y CLA.
- Otra forma de realizar un restador es basar el diseño en un sumador ya existente y un circuito que obtenga el complemento 2 de la palabra binaria B.



Practica 3: Restador

- Para la realización de restadores de un mayor número de bits se pueden utilizar las mismas técnicas utilizadas en las sumadores: conexión en cascada y CLA.
- Otra forma de realizar un restador es basar el diseño en un sumador ya existente y un circuito que obtenga el complemento 2 de la palabra binaria B.



Practica 3: Restador

- Generar un restador de 5 bits usando complemento a 2.
- Usar 5 switches para asignarle un valor a A y 5 para B.
- Generar la simulación.
- Mostrar el resultado en decimal en los 7 segmentos y en binario en los leds.

Practica 3: Multiplicador

- El circuito multiplicador, a pesar de ser combinacional como todos los circuitos cubiertos hasta este momento, tiene una complejidad tal en su construcción y definición que se han propuesto una serie de técnicas de todo tipo para llevar a cabo la tarea deseada.
- La tecnología actual permite que los circuitos multiplicadores puedan ser inferidos con lenguajes de descripción de una manera sencilla, hacienda uso de los módulos aritméticos de la librería IEEE.

```
1   LIBRARY ieee ;
2   USE ieee.std_logic_1164.all ;
3   USE ieee.std_logic_unsigned.all ;
4
5   ENTITY ArthmeticUnit IS
6       PORT (  S0:IN STD_LOGIC;
7               S1:IN STD_LOGIC;
8               x1 : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
9               x2 : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
10              y : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
11  END  ArthmeticUnit;
12
13  architecture beh of ArthmeticUnit is
14  begin
15  process (x1,x2,S0,S1)
16  begin
17  If S0='0' and S1='0' Then
18      y<=x1+x2;
19  End IF;
20  If S0='1' and S1='0' Then
21      y<=x1-x2;
22  End IF;
23  If S0='0' and S1='1' Then
24      y<=x1+1;
25  End IF;
26  If S0='1' and S1='1' Then
27      y<=x1-1;
28  End IF;
29  END PROCESS ;
30  end beh;
31
```

Practica 3: Multiplicador

- El producto entre un operando de n bits con otro operando de m bits genera un resultado que requiere $m + n$ bits en su representación. Además, las reglas de los signos deben tomarse en cuenta cuando se trata de operandos que se encuentran expresados en complemento 2.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity Multiplicador_n is
   generic(
      n: integer:= 4
      );
   port(
      A, B   : in  std_logic_vector(n-1   downto 0); -- Operandos
      M1, M2 : out std_logic_vector(2*n-1 downto 0)  -- Producto
      );
   end Multiplicador_n;

architecture Aritmetica of Multiplicador_n is
begin
   process(A, B)
   variable MUU : unsigned(2*n-1 downto 0);
   variable MSS :   signed(2*n-1 downto 0);
   begin

     MUU := unsigned(A) * unsigned(B);
     MSS :=   signed(A) *   signed(B);
     M1  <= std_logic_vector(MUU);
     M2  <= std_logic_vector(MSS);
  end process;
end Aritmetica;
```

Practica 3: Multiplicador

- Generar un multiplicador de 4 bits usando variables unsigned y signed.
- Genera la multiplicación 8×8, 1×8,1×1, $F \times 1$ y $F \times F$ y explica la diferencia entre ambos resultados.