Propuesta de Diseño

Instituto Tecnológico de Costa Rica
CE 3201 - Taller de Diseño Digital
Laboratorio 2
Problema 1: Calculadora Parametrizable

Estudiantes:
Juan Pablo Rodríguez
Jafet Díaz Morales

2 de septiembre de 2025

Para diseñar una calculadora parametrizable es necesario diseñar primeramente las operaciones que la componen. En este laboratorio, se realizó a base de operaciones lógicas con compuertas las operaciones suma, resta y multiplicación.

Para todos estos casos se tiene en cuenta en 1 bit al menos. Para parametrizarlo es cuestión de definir la cantidad necesaria y conectar varios de 1 bit en cascada. En general, se conecta la salida Cout con el Cin del siguiente bit.

1. Operación de resta

A continuación se presenta la tabla de verdad correspondiente a una operación binaria de resta completa: B-A considerando un bit de acarreo

Se inició el diseño partiendo de un restador completo de 1 bit. Este restador puede hacerse de múltiples formas pues tiene dos salidas (S y C_{out}). Además, cada una de las dos salidas puede ser representada con distintas cooompuertas lógicas.

Para S, usando la técnica de Suma de Productos se parte de la siguiente ecuación booleana:

$$S = \overline{C_{in}}BA + \overline{C_{in}}B\overline{A} + C_{in}\overline{BA} + C_{in}BA$$

Aplicando leyes de matemática discreta (distributiva) puede llegarse a la siguiente ecuación:

índice	C_{in}	B	A	S	C_{out}
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	0
4	1	0	0	1	1
5	1	0	1	0	1
6	1	1	0	0	0
7	1	1	1	1	1

Cuadro 1: Tabla de verdad de un restador completo de 1 bit.

$$S = \overline{C_{in}}(\overline{B}A + B\overline{A}) + C_{in}(\overline{BA} + BA) \tag{1}$$

Luego puede transformarse usando leyes y la operación XOR en la siguiente ecuación:

$$S = C_{in} \oplus (B \oplus A) \tag{2}$$

Esta es la ecuación final para la salida S.

Cabe aclarar que para S, utilizar Mapas de Karnough no trae ningun otro diseño representativo.

Podría diseñarse tanto con la ecuación (1) que tiene compuertas AND, OR y NOT como con su siguiente simplificación (2), la que usa XOR y AND. Sin embargo, se escoge la ecuación (2) con XOR pues utiliza menos compuertas (es más facil de programar y utiliza menos componentes).

Para C_{out}, usando la técnica de Suma de Productos se parte de la siguiente ecuación booleana:

$$C_{out} = \overline{C_{in}B}A + C_{in}\overline{BA} + C_{in}\overline{B}A + C_{in}BA$$

Aplicando leyes de matemática discreta (distributiva y Complemento) puede llegarse a la siguiente ecuación:

$$C_{out} = C_{in}\overline{B} + A\overline{C_{in}B} + C_{in}BA \tag{3}$$

Esta ecuación esta bastante simplificada y podría servir así. Sin embargo, en esta ocasión si es conveniente usar Mapas de Karnough para simplificarla aún más:

C_{in} \ BA	00	01	11	10
0	0	1	0	0
1	1	1	1	0

Con este mapa se puede llegar a la siguiente ecuación booleana:

$$C_{out} = C_{in}\overline{B} + C_{in}A + \overline{B}A \tag{4}$$

Así, podría diseñarse tanto con la ecuación (3) que tiene compuertas AND, OR y NOT o bien con la ecuación obtenida con el mapa de Karnough (4). Se escoge la ecuación (4) con mapa de Karnugh pues utiliza menos compuertas (la ecuaciôn 3 tiene AND de 3 entradas mientras que la 4 solo tiene AND de 2 entradas por lo que utiliza menos componentes y es más facil de programar).

Así, el diseño para el restador completo de 1 bit se puede realizar con las ecuaciones 2 y 4 de la siguiente forma:

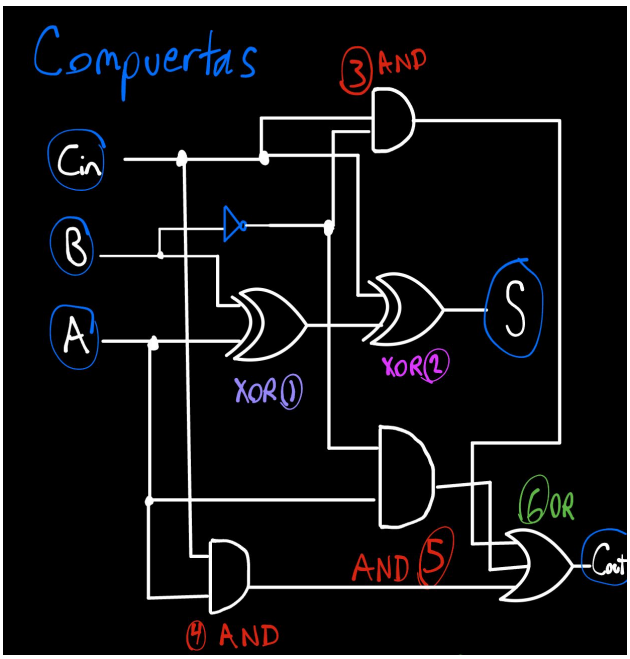


Figura 1: Esquemático de restador completo de 1 bit

Note que el restador de 1 bit produce 2 salidas (Cout y S). Para hacer el restador de bits parametrizables con n bits es necesario poner en cascada n de estos restador. Además, para que cada operación se realice correctamente, el acarreo de salida de uno se convierte en el

acarreo de entrada del otro.

El siguiente es un ejemplo de como se vería si fuera n=4:

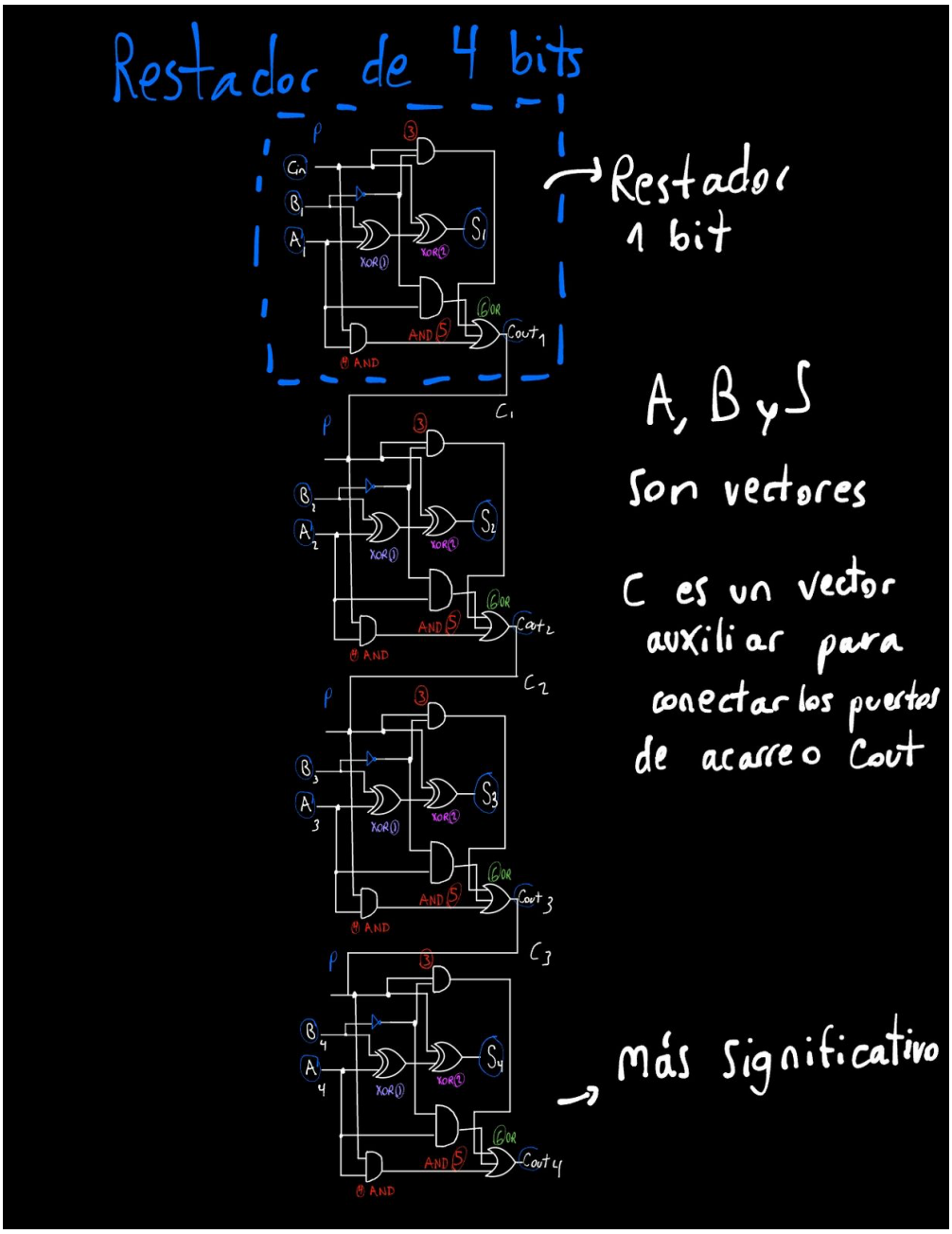


Figura 2: Esquemático de restador completo de 4 bit

2. Operación de suma

A continuación se presenta la tabla de verdad correspondiente a una operación binaria de suma completa: A + B considerando un bit de acarreo

índice	A	B	C_{in}	S	C_{out}
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Cuadro 2: Tabla de verdad de un sumador completo de 1 bit.

Se inició el diseño partiendo de un sumador completo de 1 bit. Este sumador puede hacerse de múltiples formas pues tiene dos salidas (S y C_{out}). Además, cada una de las dos salidas puede ser representada con distintas compuertas lógicas.

Para S, usando la técnica de Suma de Productos se parte de la siguiente ecuación booleana:

$$S = \overline{AC_{in}}B + \overline{C_{in}B}A + \overline{AB}C_{in} + C_{in}BA$$

Aplicando leyes de matemática discreta (distributiva) puede llegarse a la siguiente ecuación:

$$S = \overline{C_{in}}(\overline{A}B + A\overline{B}) + C_{in}(\overline{AB} + AB) \tag{5}$$

Luego puede transformarse usando leyes y la operación XOR en la siguiente ecuación:

$$S = \overline{C_{in}}(B \oplus A) + C_{in}(\overline{B \oplus A}) \tag{6}$$

Y a su vez, con XOR:

$$S = (A \oplus B) \oplus C_{in} \tag{7}$$

Esta es la ecuación final para la salida S.

Cabe aclarar que para S, utilizar Mapas de Karnough no trae ningun otro diseño representativo.

Podría diseñarse tanto con la ecuación (5) que tiene compuertas AND, OR y NOT o bien con su siguiente simplificación (7) que solo utiliza XOR y AND. Sin embargo, se escoge la ecuación (7) con XOR pues utiliza menos compuertas (es más facil de programar y utiliza menos componentes). La otra propuesta sería util si no se dispusiera de compuertas XOR, pero en este caso, es la más óptima.

Para C_{out}, usando la técnica de Suma de Productos se parte de la siguiente ecuación booleana:

$$C_{out} = \overline{A}BC_{in} + A\overline{B}C_{in} + AB\overline{C_{in}} + ABC_{in}$$

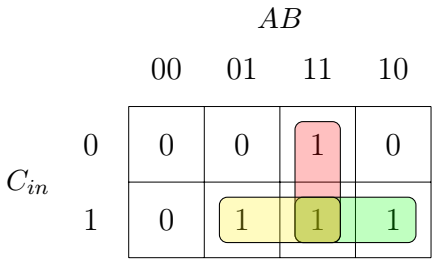
Aplicando leyes de matemática discreta (distributiva y Complemento) puede llegarse a la siguiente ecuación:

$$C_{out} = AB + C_{in}(\overline{A}B + B\overline{A}) \tag{8}$$

A su vez con XOR:

$$C_{out} = AB + C_{in}(A \oplus B) \tag{9}$$

Esta ecuación esta bastante simplificada y podría servir así. Ahora bien, en esta ocasión utilizar Mapas de Karnough nos lleva a una ecuación diferente:



Con este mapa se puede llegar a la siguiente ecuación booleana:

$$C_{out} = AB + C_{in}(A + B) \tag{10}$$

Así, podría diseñarse tanto con la ecuación (9) que tiene compuertas AND, OR y XOR o bien con la ecuación obtenida con el mapa de Karnough (10) que tiene únicamente OR y AND.

En este caso la cantidad de compuertas necesarias es prácticamente la misma (solo cambia una XOR por una OR). La opción (10) podría ser útil si no se tienen compuertas XOR.

Sin embargo, en este caso, debido a que en S ya utilizamos un OR de A + B, es útil escoger la ecuación 9 pues así reducimos la cantidad de compuertas de 6 a 5.

Por lo tanto, se escoge la opción 9 que contiene el XOR para reducir las compuertas totales en el sumador necesarias.

Así, el diseño para el restador completo de 1 bit se puede realizar con las ecuaciones (7) y (9) de la siguiente forma:

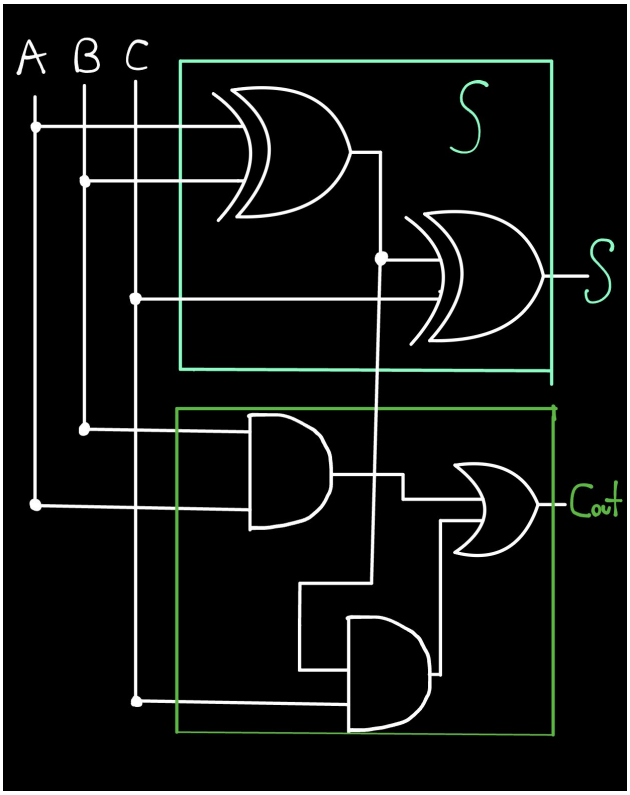


Figura 3: Esquemático de sumador completo de 1 bit (propuesta con XOR

Note que en la opción utilizando XOR (9) puede reutilizar el XOR que ya se había determinado para S, permitiendo un mejor uso del hardware y las compuertas ya dispuestas.

Por otro lado, el esquemático si se hubiera elegido (7) y (10) corresponde a la siguiente imagen:

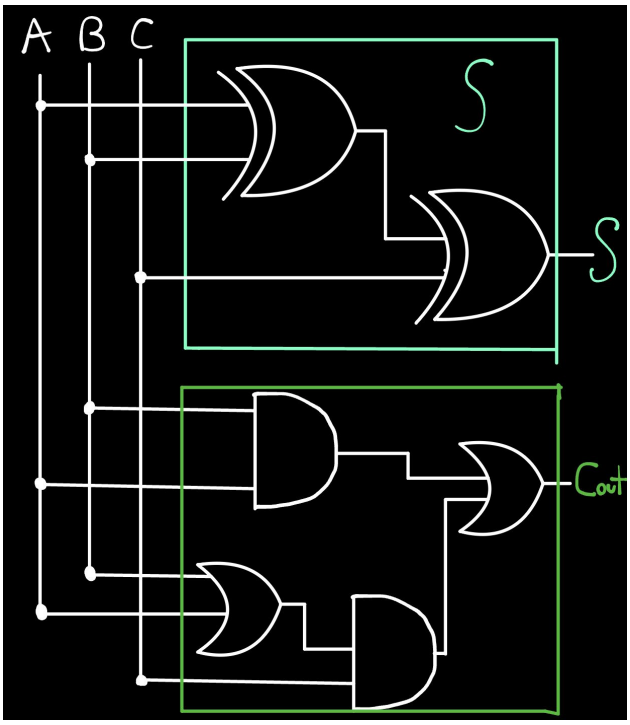


Figura 4: Esquemático de sumador completo de 1 bit (propuesta con OR)

Note que el sumador de 1 bit produce 2 salidas (Cout y S). Para hacer el sumador de bits parametrizables con n bits es necesario poner en cascada n de estos sumadores (de forma similar al restador). Además, para que cada operación se realice correctamente, el acarreo de salida de uno se convierte en el acarreo de entrada del otro.

3. Operación de la multiplicación

Como el resultado de una multiplicación no tiene una relación general para cualquier par de dígitos si no que depende de la cantidad de dígitos también no se puede hacer una tabla de verdad que contenga los dígitos del multiplicando y multiplicador que resulten en un resultado y que este se cumpla siempre para N dígitos. Lo que se propone para el diseño de un multiplicador es un simple algoritmo que es el que utilizamos al hacer multiplicación a mano y se resume en 3 reglas:

1. Se suma el multiplicando cada vez que el dígito menos significativo es un 1.
2. Se corre el multiplicador a la derecha una posición cada iteración.
3. Se corre el multiplicando a la izquierda una posición cada iteración

Como se puede ver, el resultado de este algoritmo depende de una suma, por lo que se va a aprovechar el módulo de sumas que se diseñó para este proyecto.

Para implementar este algoritmo se hacen 2 propuestas:

1. Máquina de estados donde cada estado activa una bandera que indica una operación
2. Se utiliza un contador para saber cuándo terminar el algoritmo

La primera propuesta se puede consolidar en la siguiente tabla

Estado Actual	i	Q_0	add_done	cntr	Siguiente Estado	add_flag	shift_flag
IDLE	1	X	X	X	CHECK	0	0
CHECK	X	1	X	0	ADD	1	0
CHECK	X	0	X	0	SHIFT	0	1
ADD	X	X	0	1	SHIFT	0	1
SHIFT	X	X	X	1	DONE	0	0
SHIFT	X	X	X	0	CHECK	0	0

Cuadro 3: Tabla de estados de un multiplicador. (i es la señal inicio, Q_0 es el LSB del multiplicador cntr es el estado donde contador =N

La segunda propuesta es mucho más simple y se adhiere mejor a la lógica combinacional que se requiere para este laboratorio. Su lógica se consolida en la siguiente tabla.

counter<N	Q_0	op_active	add_flag	shift_flag	done_flag	inc_flag
0	X	0	0	0	1	0
1	0	1	0	1	0	1
1	1	1	1	0	0	1

Cuadro 4: Tabla de condiciones de un multiplicador. (Q_0 es el LSB del multiplicador

Con esta tabla se puede entonces implementar un diseño de lógica combinacional donde se tienen los siguientes valores de las banderas:

- add_flag = Q_0
- shift_flag = $Cntr$
- done_flag = $\overline{Cntr}$
- inc_flag = $Cntr$

Se opta entonces por el segundo diseño ya que no requiere codificar los estados y las relaciones entre las variables de entrada y las banderas son bastante simples para implementar el algoritmo de una manera sencilla.