FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

UNIDADES ARITMÉTICAS

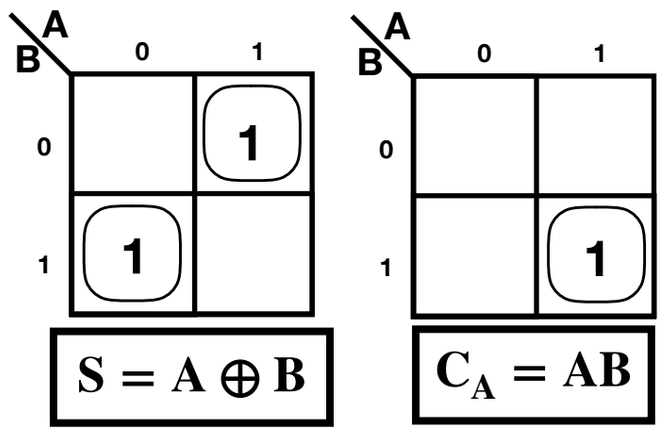
Prof. L. F. Caram

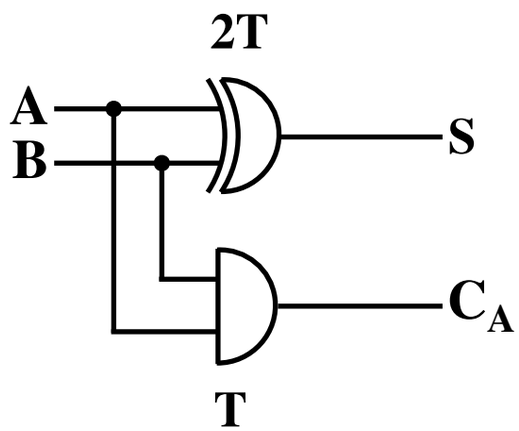
UNIDADES ARITMÉTICAS

Suma de dos palabras de 1 bit: S=A+B

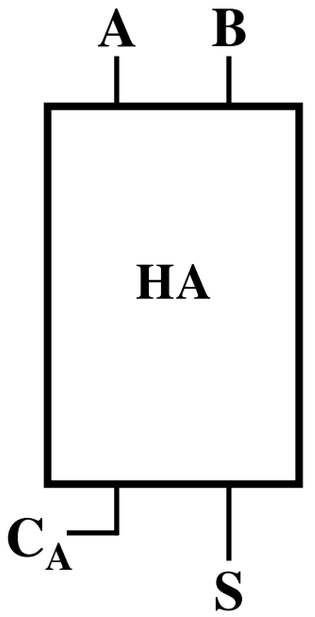
SEMI SUMADOR
HALF ADDER

A	B	S	C_A
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





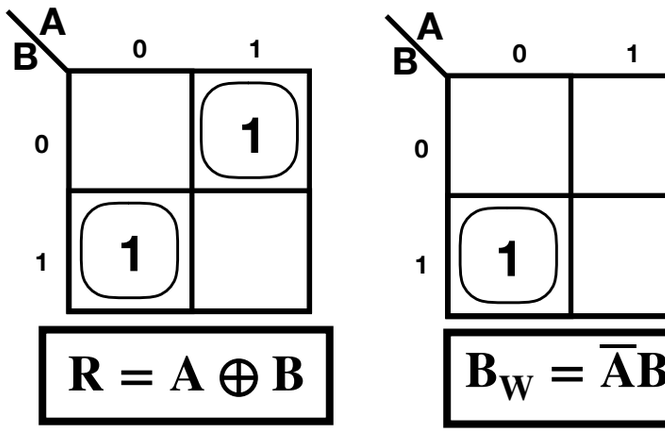
≡

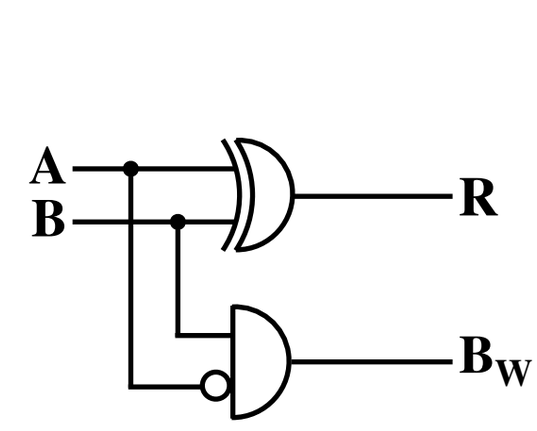


Resta de dos palabras de 1 bit: R=A-B

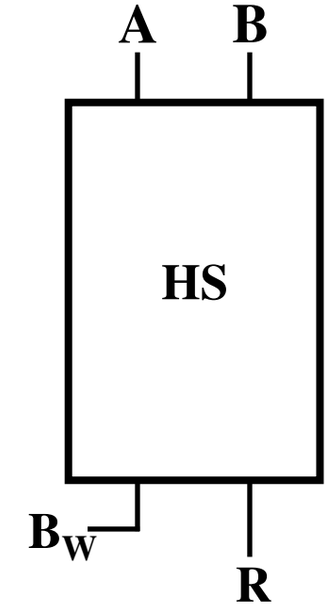
SEMI RESTADOR
HALF SUBTRACTOR

A	B	R	B_W
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0





≡

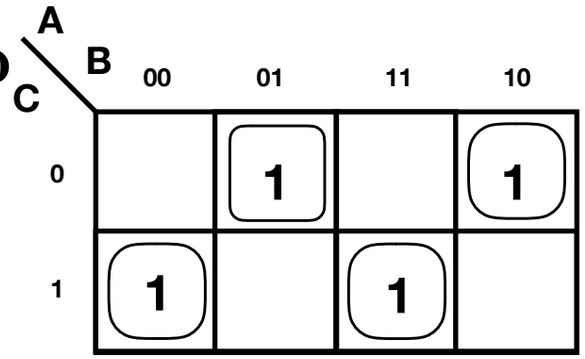


El concepto de Borrow $\mathbf{B_W}$ es perdir prestado al vecino mas significaitivo, para poder hacer la resta. Esto es necesario cuando el minuendo es menor al sustraendo.

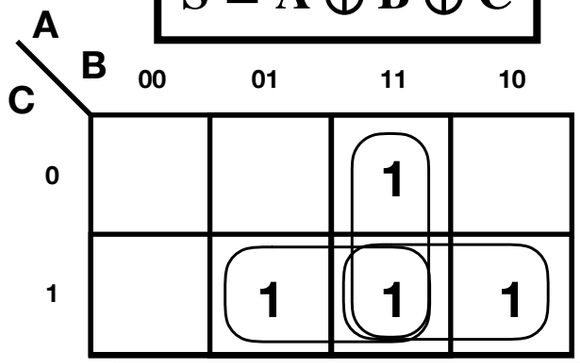
UNIDADES ARITMÉTICAS

SUMADOR COMPLETO
FULL ADDER

A	B	C	S	C_A
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



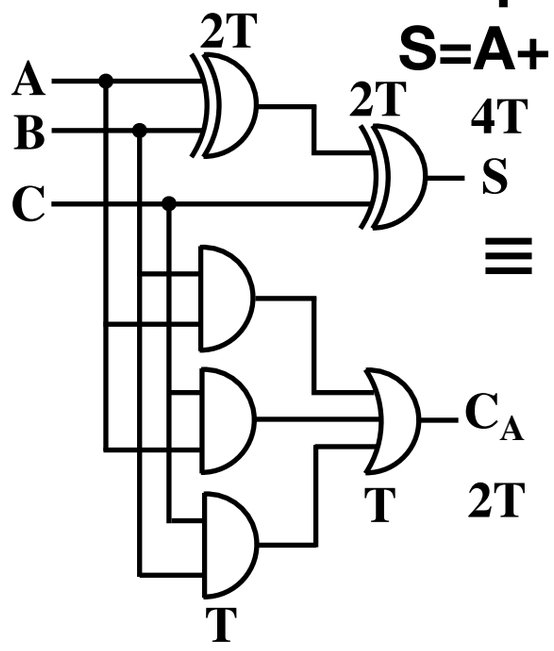
$$S = A \oplus B \oplus C$$



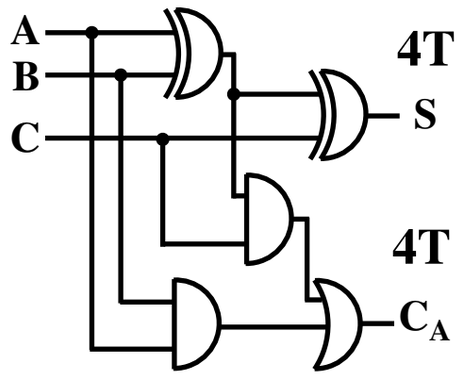
$$C_A = AB + AC + BC$$

Suma de tres palabras de 1 bit

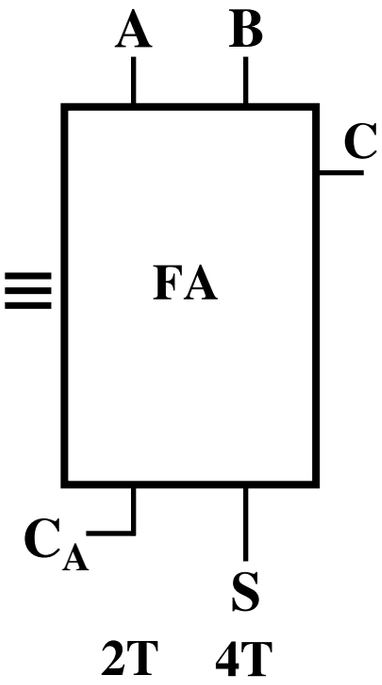
S=A+B+C



≡

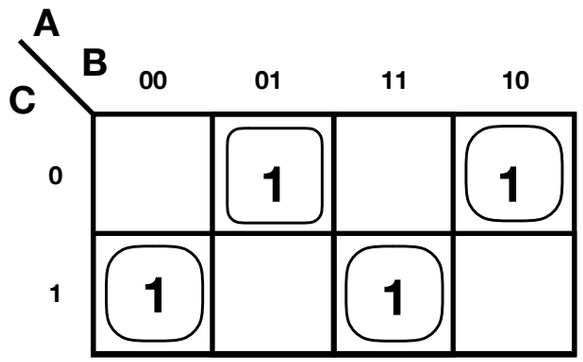


≡

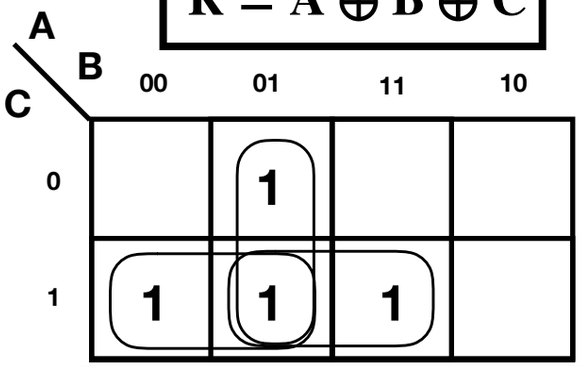


RESTADOR COMPLETO
FULL SUBTRACTOR

A	B	C	R	B_W
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



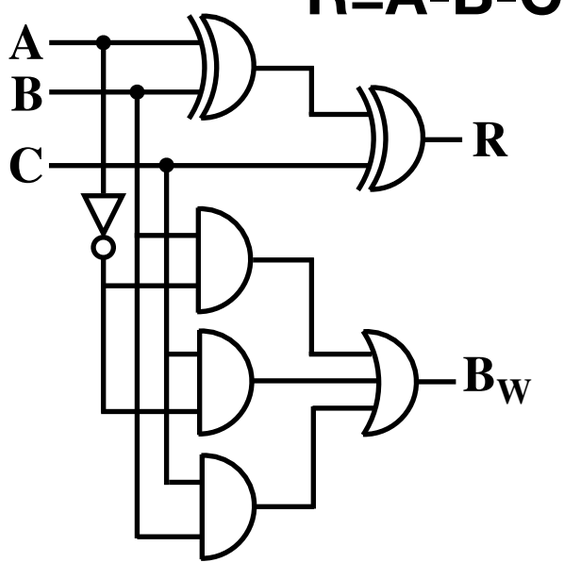
$$R = A \oplus B \oplus C$$



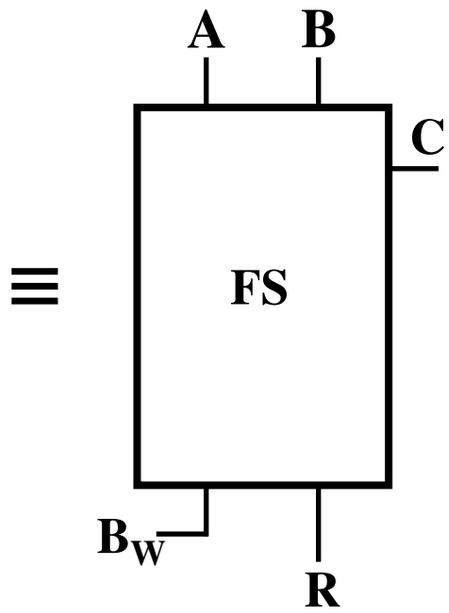
$$B_W = \overline{A}B + \overline{A}C + BC$$

Resta de tres palabras de 1 bit

R=A-B-C

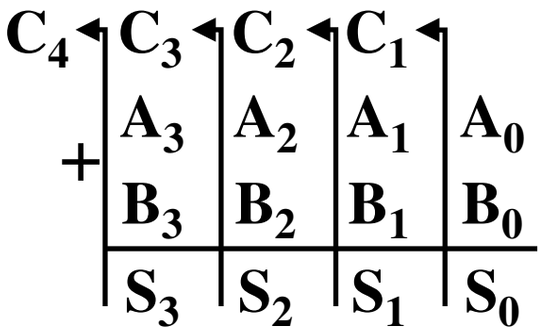


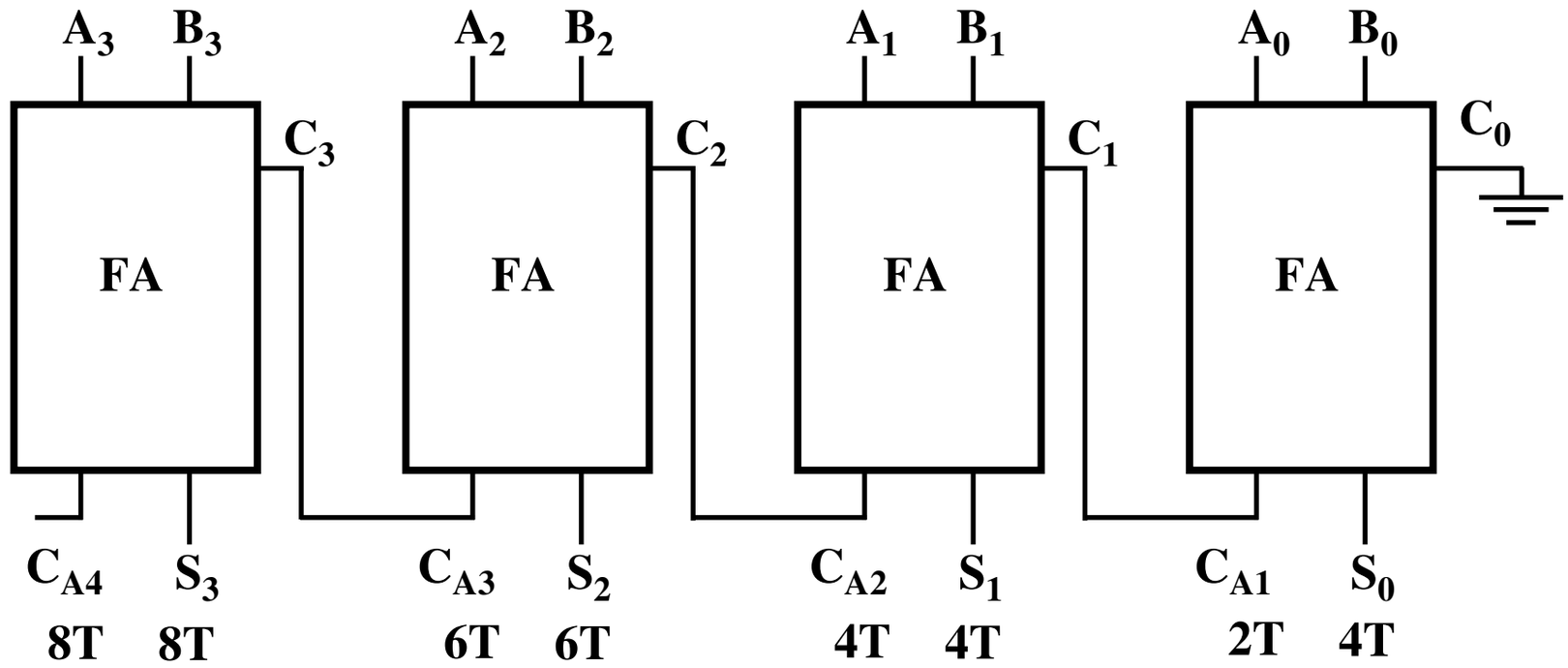
≡



UNIDADES ARITMÉTICAS

Sumador Paralelo de dos palabras de 4 bits y transporte (acarreo-CARRY) serie



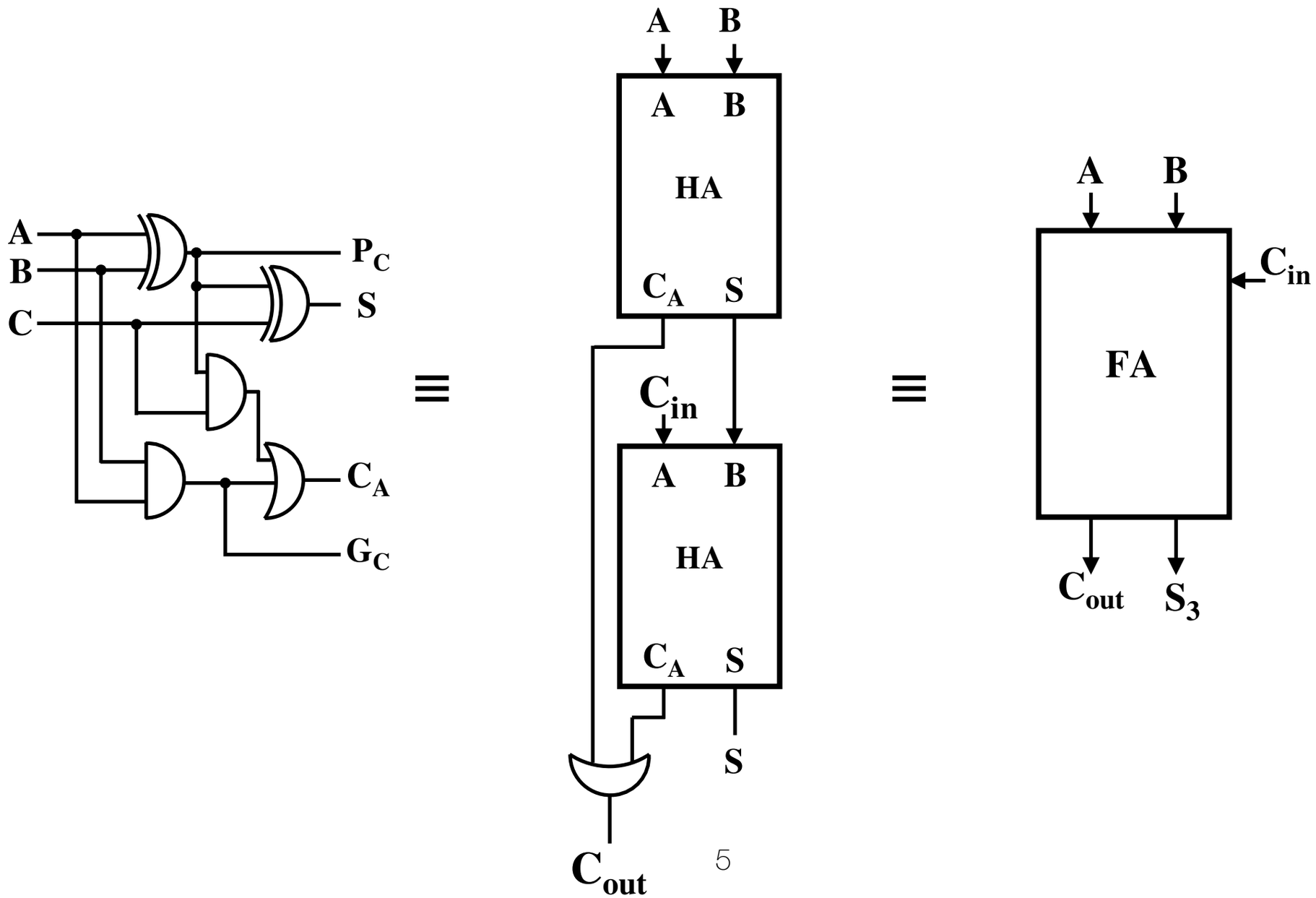


<u>PROBLEMAS DE RETARDO (DELAY):</u> Si el Retardo de Propagación de cada compuerta **AND-OR-NAND-NOR** es $\mathbf{T}$, y el de cada compuerta **XOR** es $\mathbf{2T}$, entonces el tiempo de funcionamiento de cada $\mathbf{FA}$ puede verse en la figura. El problema de este circuito es que al tener el carry (acarreo-transporte) en serie se van acumulando los retardos en cada full adder en obtener cada suma y cada carry, eso afecta a cada uno de los full adder mas significativos. Por lo tanto al aumentar el ancho de palabra es decir de los registros se acumulan mas retardos y el resultado definitivo de la suma total se incrementará en $\mathbf{2T}$ lo cual no es óptimo, a éste efecto se lo conoce como **RIPPLE CARRY (Acarreo o arrastre en ondas)**. La solución a este problema es acelerar los carry en forma simultanea, para eso se usa una **unidad aceleradora o anticipadora de carry (carry look ahead)**.

UNIDADES ARITMÉTICAS

Sumador Completo (Full Adder) en base a dos Semisumadores (Half Adder)

Existe otra implementación posible para un full adder en base a dos half adder, como puede verse, en lugar de tomar $\mathbf{C_A = AB + AC + BC}$ se lo toma, $\mathbf{C_A = AB + A\overline{B}C + \overline{A}BC}$, no es óptima, pero en el caso que $\mathbf{A = 1}$ o $\mathbf{B = 1}$ entonces $\mathbf{C_A}$ valdrá lo mismo lógicamente. Por lo tanto son funciones equivalentes ya que responden a la misma tabla de verdad. Y estoy usando una compuerta AND menos.

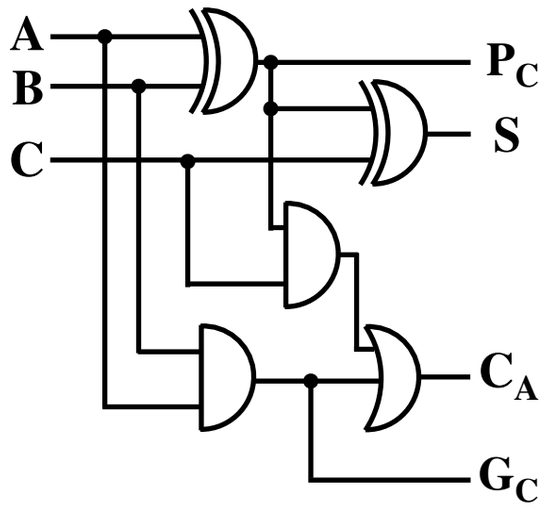


UNIDADES ARITMÉTICAS

Unidad Aceleradora de Carry (Carry Look Ahead)

La tabla muestra cuando los Carries son Propagados o Generados

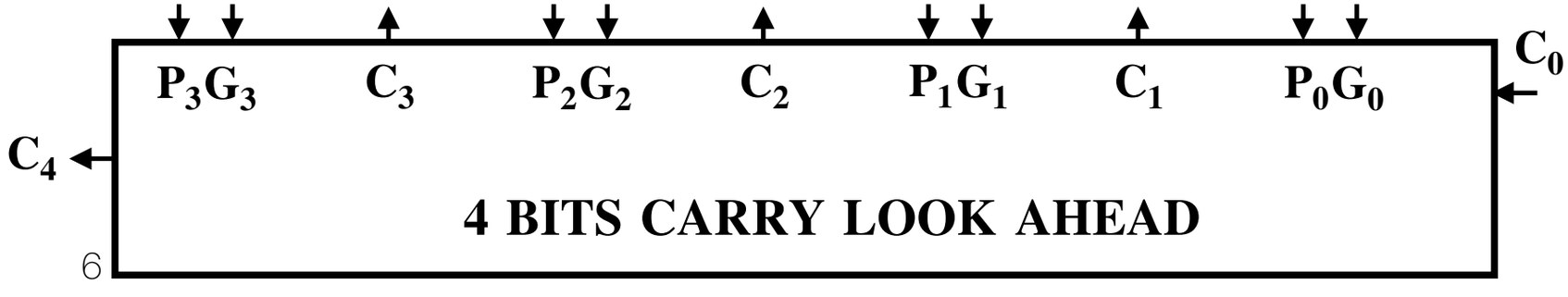
A	B	C_{in}	C_{out}	TIPO DE CARRY
0	0	0	0	NO
0	0	1	0	NO
0	1	0	0	NO
0	1	1	1	PROPAGA
1	0	0	0	NO
1	0	1	1	PROPAGA
1	1	0	1	GENERA
1	1	1	1	GENERA/PROPAGA



$$G_i = A_i \cdot B_i \qquad C_1 = G_0 + P_0 \cdot C_0$$
$$P_i = A_i + B_i \qquad C_2 = G_1 + P_1 \cdot C_1$$
$$P_i = A_i \oplus B_i \qquad C_3 = G_2 + P_2 \cdot C_2$$
$$C_4 = G_3 + P_3 \cdot C_3$$

La lógica de anticipación de carries usa los conceptos de **generación y propagación de carries**. Es decir la suma de 2 palabras de 1 bit **A** y **B** generan carry independientemente si hay o no carry anterior (C_{in}), cuando ambas $A = B = 1$. Mientras que hay propagación de carry cuando hay carry anterior $(C_{in} = 1)$ y alguna o ambas son iguales a $A = 1$ o $B = 1$ o $A = B = 1$. Todos los casos de generación y propagación de carry pueden verse en la tabla y se concluye que $G_i = A_i \cdot B_i$ y $P_i = A_i + B_i$. Propagar y generar se definen con respecto a un solo dígito de suma y no dependen de ningún otro dígito en la suma, por lo tanto son propios de cada full adder.

Así, el carry de la suma de dos bits, se produce cuando la suma genera o el siguiente bit menos significativo acarrea y la suma lo propaga, es decir $C_{i+1} = G_i + (P_i \cdot C_i)$ con C_i el bit de carry del digito **i**, P_i y G_i la progación y la generación de bits del digito **i** respectivamente.

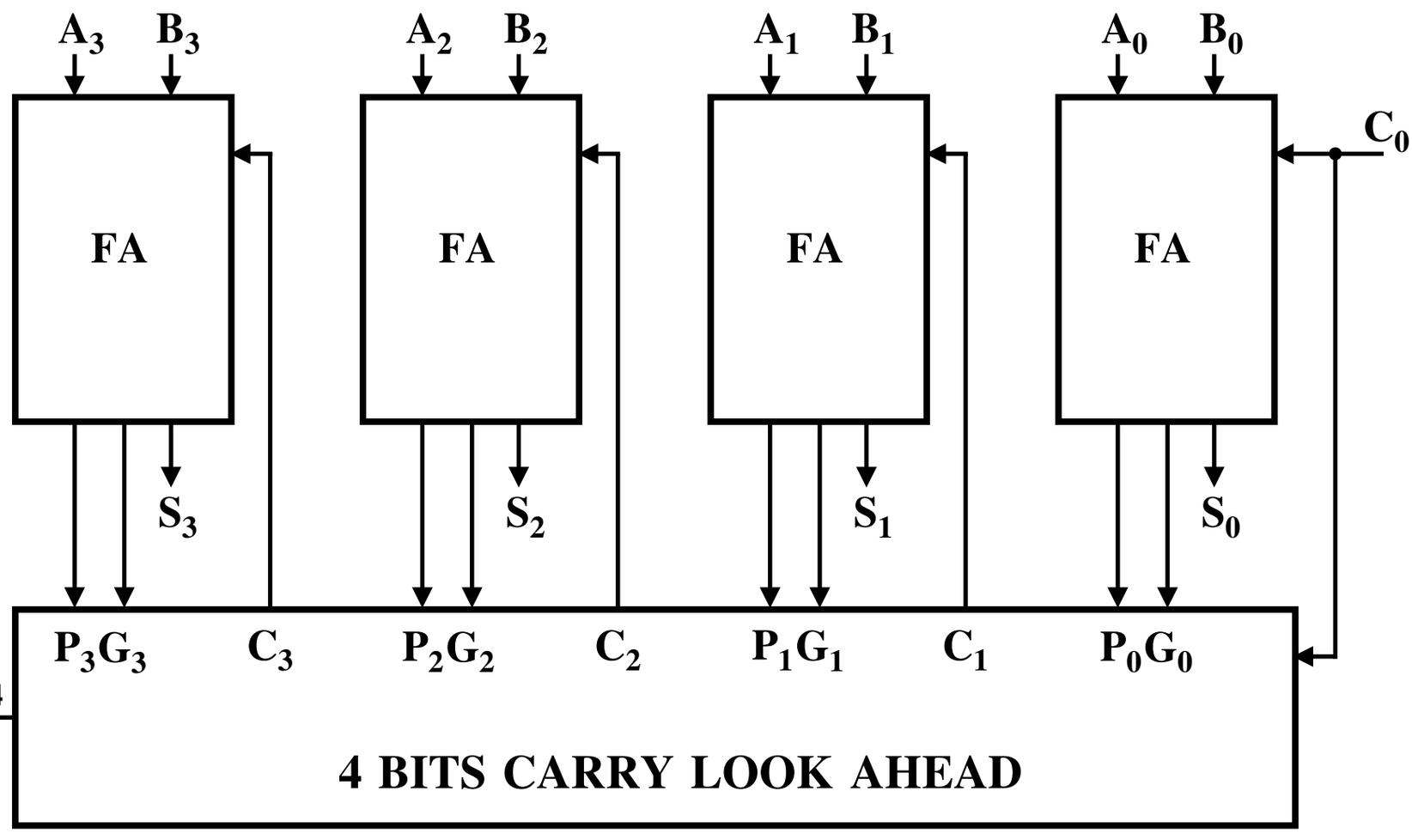


UNIDADES ARITMÉTICAS

Sumador Paralelo de dos palabras de 4 bits tipo Carry Look Ahead (Unidad Aceleradora de Carry)

C_4	C_3	C_2	C_1	
	A_3	A_2	A_1	A_0
+	B_3	B_2	B_1	B_0
	S_3	S_2	S_1	S_0

Para cada bit que a sumar, la lógica de anticipación de carry determinará si ese par de bits generará o propagará un carry. Esto permite que el circuito "procese previamente" los dos números que se suman para determinar el carry antes de tiempo. Luego, cuando se realiza la suma real, no hay demora esperando el efecto del carry en ondas (o el tiempo que tarda el carry desde el primer full adder hasta el último).



A veces es mas simple usar $\mathbf{P_i = A_i \oplus B_i}$ para propagar ya que el único caso diferente es cuando $\mathbf{A = B = 1}$, pero aqui el término $\mathbf{G_i = A \cdot B = 1}$ es 1 y el término $\mathbf{P_i \cdot C_i}$ se vuelve irrelevante.

$$\mathbf{G_i = A_i \cdot B_i}$$
$$\mathbf{P_i = A_i + B_i}$$
$$\mathbf{P_i = A_i \oplus B_i}$$

$$\mathbf{C_1 = G_0 + P_0 \cdot C_0}$$
$$\mathbf{C_2 = G_1 + P_1 \cdot C_1}$$
$$\mathbf{C_3 = G_2 + P_2 \cdot C_2}$$
$$\mathbf{C_4 = G_3 + P_3 \cdot C_3}$$

Reemplazando $\mathbf{C_1}$ en $\mathbf{C_2}$, luego $\mathbf{C_2}$ en $\mathbf{C_3}$, luego $\mathbf{C_3}$ en $\mathbf{C_4}$ se produce las siguientes ecuaciones expandidas.

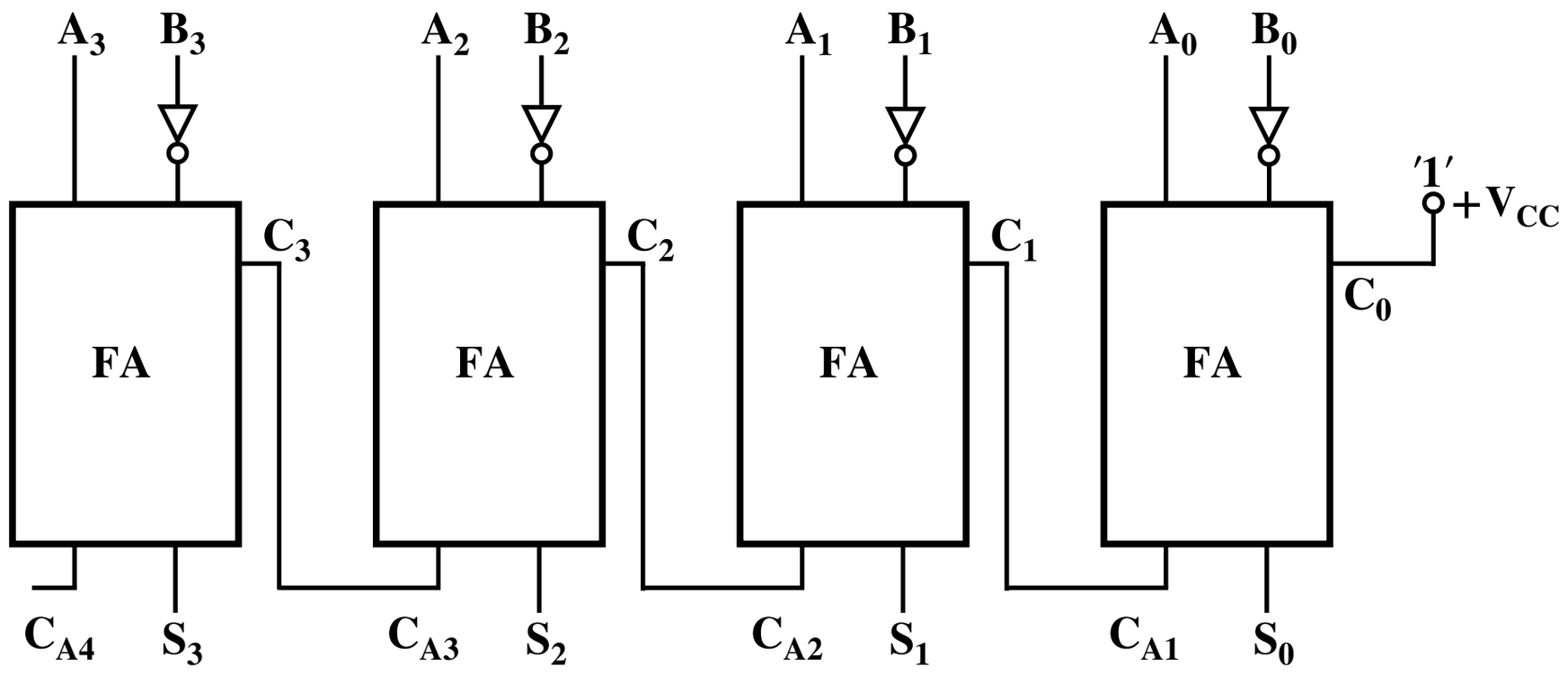
$$\mathbf{C_1 = G_0 + P_0 \cdot C_0}$$
$$\mathbf{C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1}$$
$$\mathbf{C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2}$$
$$\mathbf{C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3}$$

UNIDADES ARITMÉTICAS

Restar dos palabras de 4 bits con un Sumador Paralelo de transporte (acarreo) serie y Complemento al Módulo.

$$
\begin{array}{cccccccccc}
 & C_4 & \leftarrow & C_3 & \leftarrow & C_2 & \leftarrow & C_1 & \leftarrow & \\
 & & & A_3 & & A_2 & & A_1 & & A_0 \\
+ & & & C_M B_3 & & C_M B_2 & & C_M B_1 & & C_M B_0 \\
\hline
 & & & S_3 & & S_2 & & S_1 & & S_0
\end{array}
$$

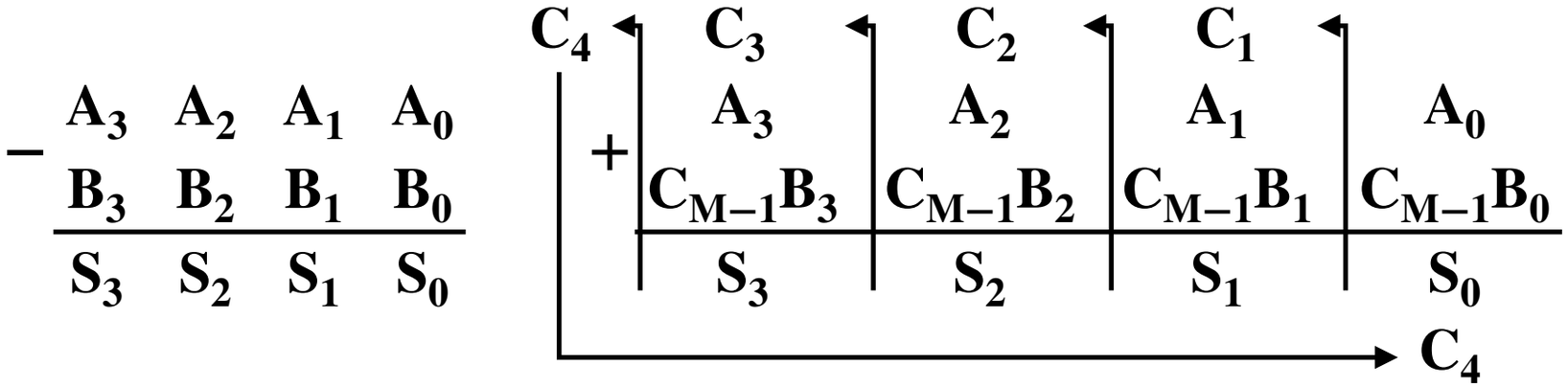
$$
\begin{array}{cccc}
 & A_3 & A_2 & A_1 & A_0 \\
- & B_3 & B_2 & B_1 & B_0 \\
\hline
 & S_3 & S_2 & S_1 & S_0
\end{array}
$$

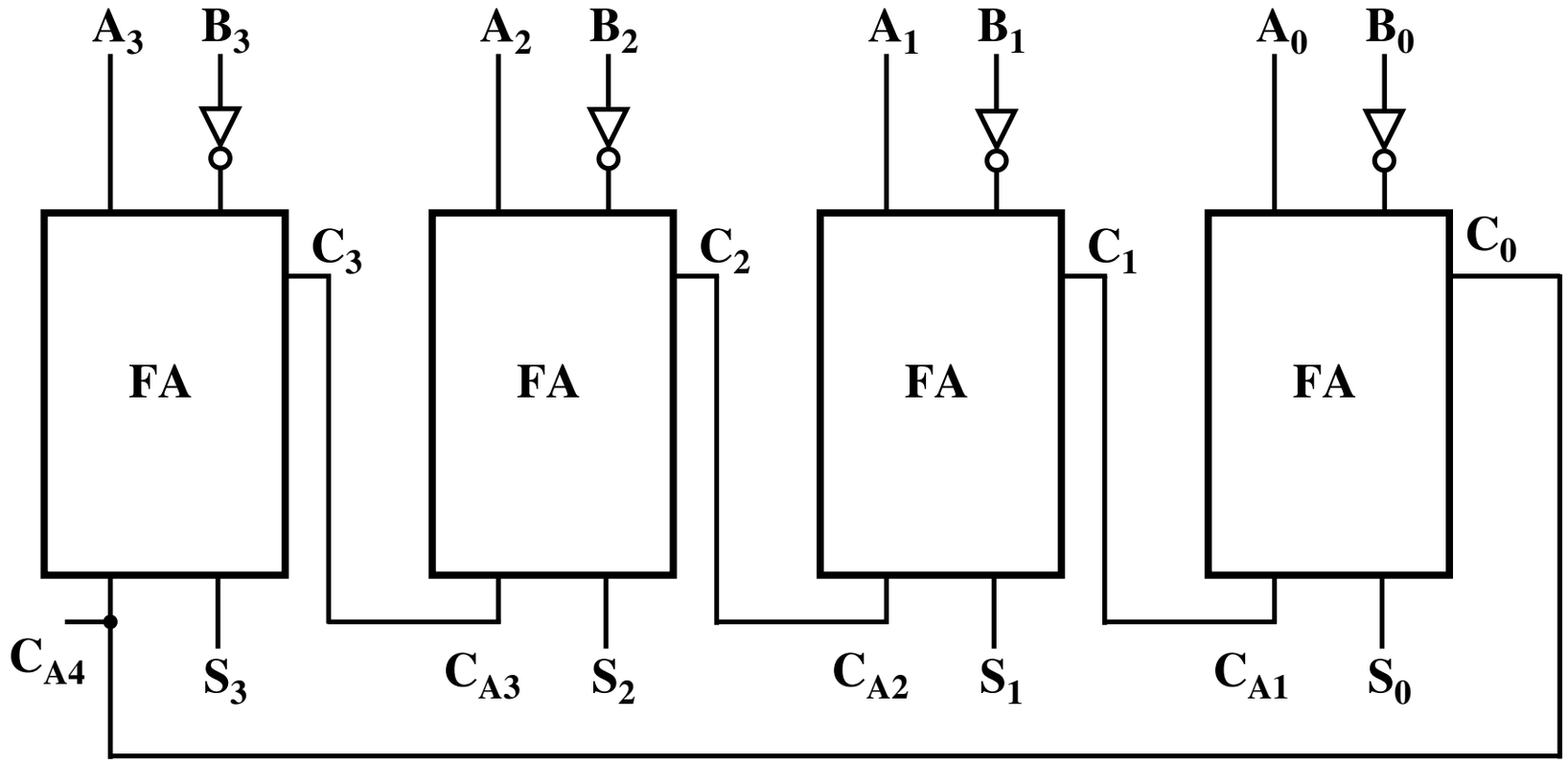


Se complementa al módulo menos 1 a B, $\mathbf{C_{M-1}B}$ y se suma **1** para lograr el complemento al módulo de B, $\mathbf{C_M B}$.

UNIDADES ARITMÉTICAS

Restar dos palabras de 4 bits con un Sumador Paralelo de transporte (acarreo) serie y Complemento al Módulo menos uno.





Se complementa al módulo menos 1 a B, $\mathbf{C_{M-1}B}$ y se realimenta el carry, ya que cuando sumabamos en $\mathbf{C_{M-1}}$ si había carry se tenía que sumar para finalizar la operación.

UNIDADES ARITMÉTICAS

Sumar o Restar dos palabras de 4 bits con un Sumador Paralelo de transporte (acarreo) serie y Complemento al Módulo con señal externa.

$$
\begin{array}{cccc}
 & A_3 & A_2 & A_1 & A_0 \\
+ & B_3 & B_2 & B_1 & B_0 \\
\hline
 & S_3 & S_2 & S_1 & S_0
\end{array}
\qquad
\begin{array}{ccccc}
C_4 & C_3 & C_2 & C_1 & \\
 & A_3 & A_2 & A_1 & A_0 \\
+ & C_MB_3 & C_MB_2 & C_MB_1 & C_MB_0 \\
\hline
 & S_3 & S_2 & S_1 & S_0
\end{array}
$$

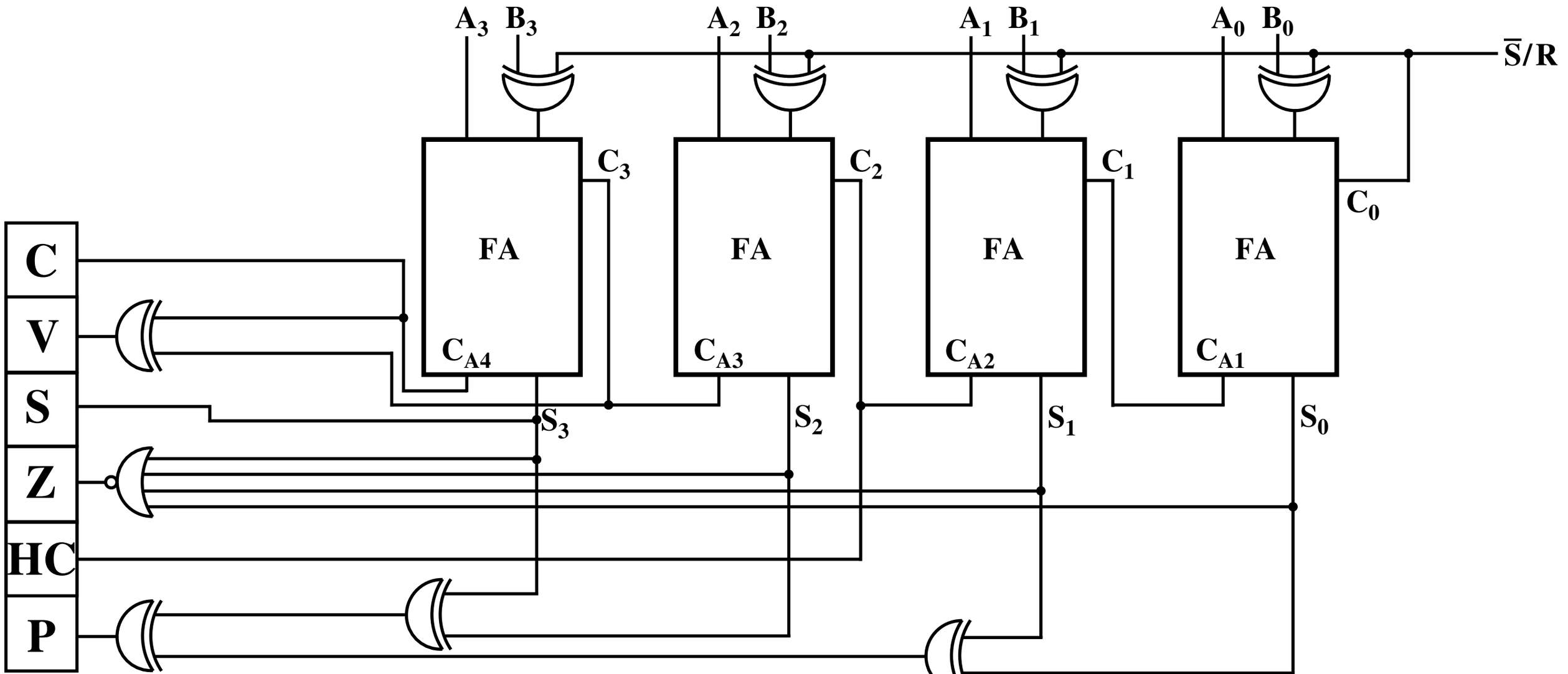
La señal externa $\overline{\mathbf{S}}/\mathbf{R}$ determina que operación se realizará si es $\mathbf{0}$ sumará y si es $\mathbf{1}$ restará, sumando el complemento al módulo de B, $\mathbf{C_MB}$. Se combinarán, usando la señal externa, el sumador y el restador con full adder visto en las páginas anteriores. Para ello tengo que invertir B, complementar al módulo menos 1 B, $\mathbf{C_{M-1}B}$ **y sumar** $\mathbf{1}$ y asi lograr el complemento al módulo de B, $\mathbf{C_MB}$. Entonces uso una compuerta $\mathbf{XOR}$ donde si una de sus entradas es $\mathbf{0}$, entonces la salida será igual a la otra entrada $\mathbf{0 \oplus B = B}$, y si una de sus entradas es $\mathbf{1}$, la salida será igual a la otra entrada invertida o complementada $\mathbf{1 \oplus B = \overline{B}}$. Por tal razón cuando la señal externa sea $\mathbf{0}$, $\overline{\mathbf{S}}/\mathbf{R} = \mathbf{0}$, el circuito sumará y cuando $\overline{\mathbf{S}}/\mathbf{R} = \mathbf{1}$, restará.

UNIDADES ARITMÉTICAS

Sumar o Restar dos palabras de 4 bits con un Sumador Paralelo de transporte (acarreo) serie y Complemento al Módulo con señal externa.

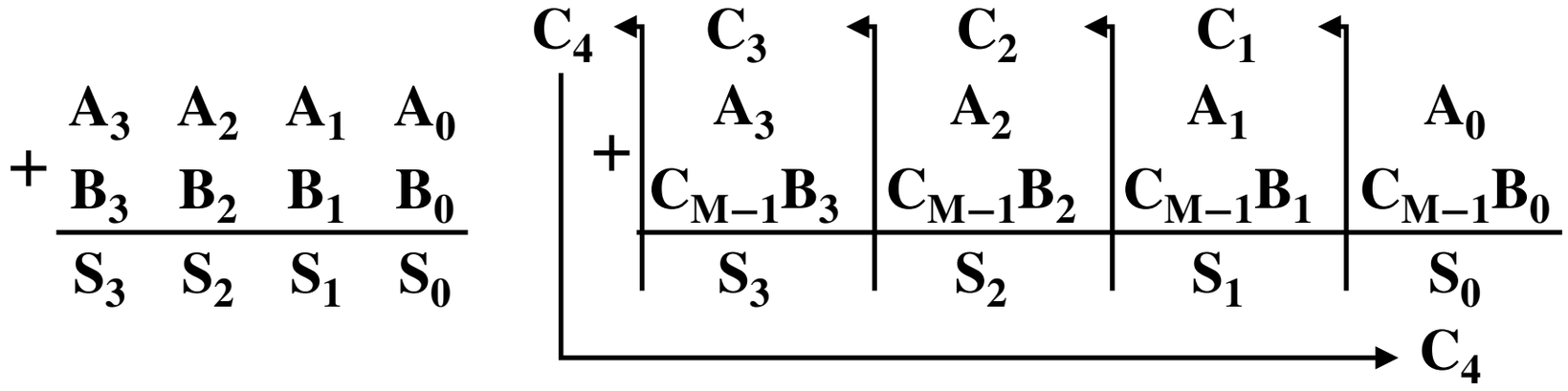
$$\begin{array}{r} A_3\ A_2\ A_1\ A_0 \\ +\ B_3\ B_2\ B_1\ B_0 \\ \hline S_3\ S_2\ S_1\ S_0 \end{array}$$

$$\begin{array}{cccccc} C_4 & C_3 & C_2 & C_1 & & \\ & A_3 & A_2 & A_1 & A_0 & \\ + & C_MB_3 & C_MB_2 & C_MB_1 & C_MB_0 & \\ \hline & S_3 & S_2 & S_1 & S_0 & \end{array}$$



UNIDADES ARITMÉTICAS

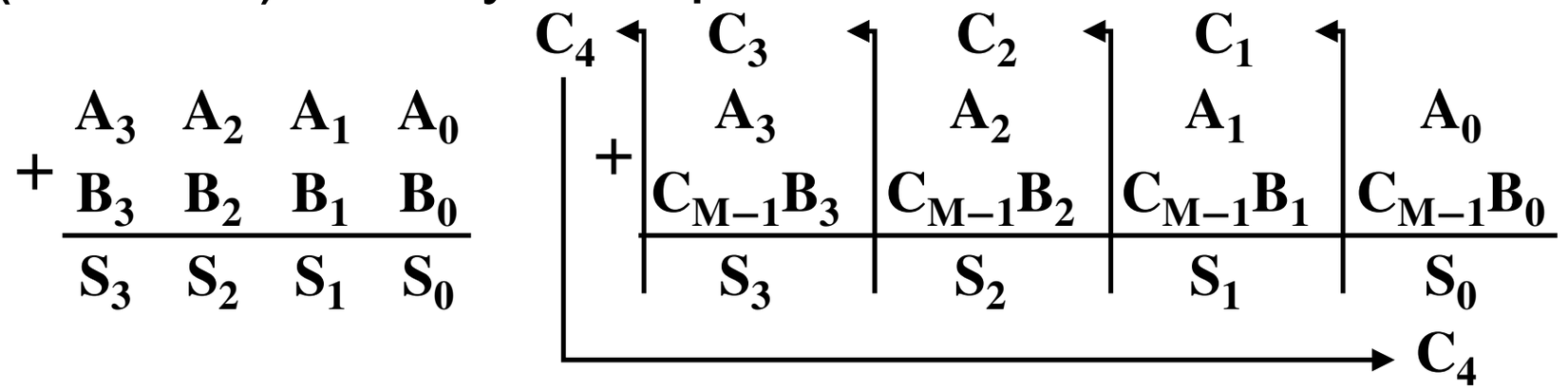
Restar dos palabras de 4 bits con un Sumador Paralelo de transporte (acarreo) serie y Complemento al Módulo menos uno con señal externa.

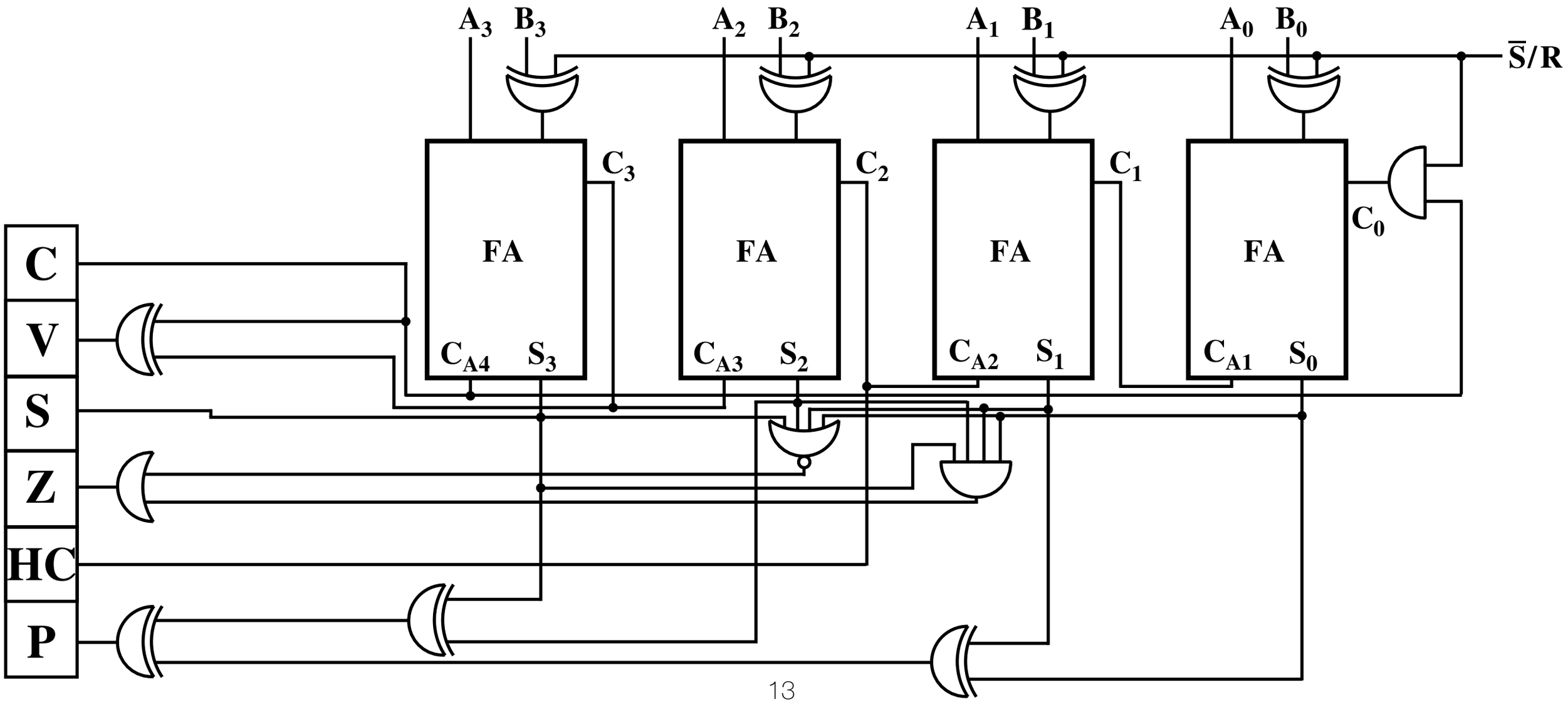


La señal externa $\overline{\mathbf{S}}/\mathbf{R}$ determina que operación se realizará si es $\mathbf{0}$ sumará y si es $\mathbf{1}$ restará, sumando el complemento al módulo menos 1 de B, $\mathbf{C_{M-1}B}$. Se combinarán, usando la señal externa, el sumador y el restador con full adder visto en las páginas anteriores. para ello tengo que invertir B, complementar al módulo menos 1 B, $\mathbf{C_{M-1}B}$ y asi lograr el complemento al módulo menos 1 de B, $\mathbf{C_{M-1}B}$. Para esto uso una compuerta $\mathbf{XOR}$ donde si una de sus entradas es $\mathbf{0}$, entonces la salida será igual a la otra entrada $\mathbf{0 \oplus B = B}$, y si una de sus entradas es $\mathbf{1}$, la salida será igual a la otra entrada invertida o complementada $\mathbf{1 \oplus B = \overline{B}}$. Por tal razón cuando la señal externa sea $\mathbf{0}$, $\overline{\mathbf{S}}/\mathbf{R} = \mathbf{0}$, el circuito sumará y cuando $\overline{\mathbf{S}}/\mathbf{R} = \mathbf{1}$, restará. Como aquí debo realimentar el carry y no debo sumar 1 como el caso de complemento al módulo, tengo que usar una compuerta $\mathbf{AND}$, que permitira inhabiltar la realimentación en el caso de la suma y habilitará la realimentación en el caso de la resta, por lo que conecto la señal externa $\overline{\mathbf{S}}/\mathbf{R}$ a una de sus entradas, mientras que a la otra conecto la realimentación del carry.

UNIDADES ARITMÉTICAS

Restar dos palabras de 4 bits con un Sumador Paralelo de transporte (acarreo) serie y Complemento al Módulo menos uno con señal externa.





UNIDADES ARITMÉTICAS

Indicadores o Banderas (Flags)

Como puedo extraer los indicadores o banderas (flags):

- El bit de carry **C**, será el carry del último full adder, el más significativo.
- El bit de **V**, será el **XOR** entre los dos últimos carries $\mathbf{V = C_4 \oplus C_3}$, es decir **si son iguales** el overflow será **0**, $\mathbf{V = 0}$, mientras que si son distintoslos dos últimos carries el overflow será **1**, $\mathbf{V = 1}$.
- El bit de signo o negativo **S** o **N**, es el bit más significativo del resultado de la operación, es decir $\mathbf{S_3}$.
- El bit de cero **Z**, será la **NOR** de todos los bits del resultado, ya que $\mathbf{Z = 1}$ unicamente si todos los bits del resultado son **0**, en cualquier otro caso será $\mathbf{Z = 0}$. Recordar que en complemento al módulo hay un solo cero $\mathbf{+0}$. En el caso de **complemento al módulo menos 1**, existen dos ceros, $\mathbf{+0}$ y $\mathbf{-0}$, por lo tanto tendré que considerar la opción cuando todos los bits del resultado sean **1**, entonces la única comperta que se pone en **1** cuando todas sus entradas son **1**, es la **AND**. Finalmente como es una u otra opción debo vincular ambas por medio de una compuerta **OR**.
- El bit de paridad **P**, será la **XOR** de todos los bits del resultado y será igual a 1, $\mathbf{P = 1}$, si la cantidad de **1** del resultado es impar y será igual a 0, $\mathbf{P = 0}$, si la cantidad de **1** del resultado es par o cero.
- El bit half carry **HC**, es el acarreo (carry) de la mitad del registro del resultado, cuando el número de bits es par. En este caso sería el carry del segundo full adder. Es solo para información ya que no se utiliza más.

UNIDADES ARITMÉTICAS

Comparadores

A	B	A > B
0	0	0
0	1	0
1	0	1
1	1	0

A	B	A ≥ B
0	0	1
0	1	0
1	0	1
1	1	1

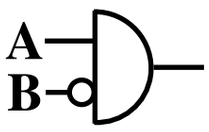
A	B	A = B
0	0	1
0	1	0
1	0	0
1	1	1

A	B	A ≤ B
0	0	1
0	1	1
1	0	0
1	1	1

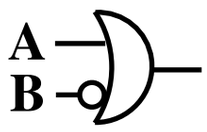
A	B	A < B
0	0	0
0	1	1
1	0	0
1	1	0

A	B	A ≠ B
0	0	0
0	1	1
1	0	1
1	1	0

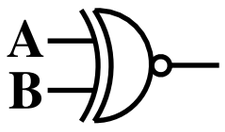
$F = A\overline{B}$



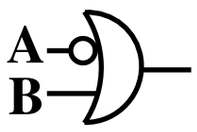
$F = A + \overline{B}$



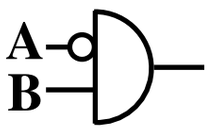
$F = \overline{A}\overline{B} + AB$



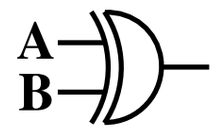
$F = \overline{A} + B$



$F = \overline{A}B$



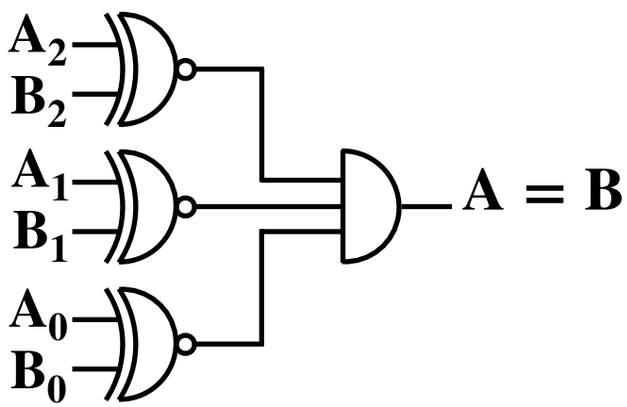
$F = A\overline{B} + \overline{A}B$



<u>Ejercicio 17:</u> Encontrar la función A = B para palabras de 3 bits

A = B en ripple (en ondas)

$$A = B: \quad (A_2 = B_2) \wedge (A_1 = B_1) \wedge (A_0 = B_0)$$



A = B en simultaneo

A_2	A_1	A_0	B_2	B_1	B_0	A = B
0	0	0	0	0	0	1
0	0	0	0	0	1	0
0	0	0	0	1	0	0
…	…	…	…	…	…	…
…	…	…	…	…	…	…
1	1	1	1	0	1	0
1	1	1	1	1	0	0
1	1	1	1	1	1	1

En simultaneo es mucho mas compleja la resolución ya que con 3 bits tengo 64 combinaciones a comparar entre si, lo cual implica un mapa de Karnaugh de 6 bits. Por eso siempre es mas sencillo resolver usando lógica proposicional $\wedge$ y $\vee$, es decir AND'S y OR'S, en ripple.

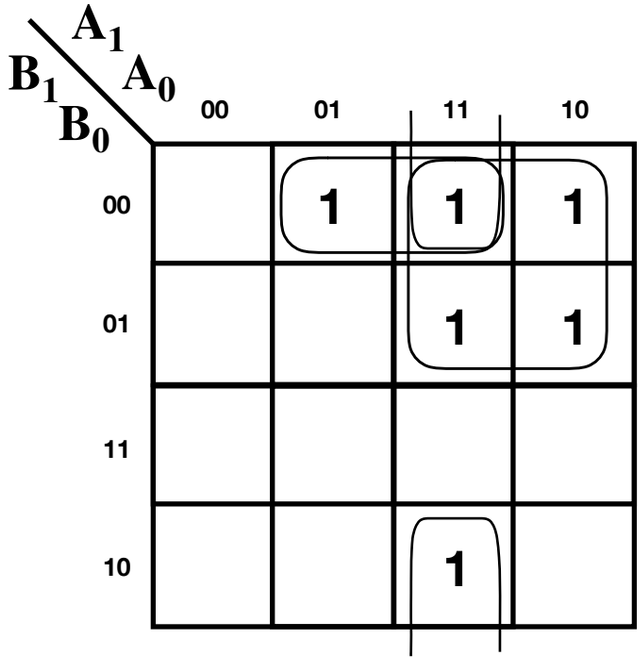
UNIDADES ARITMÉTICAS

Comparadores

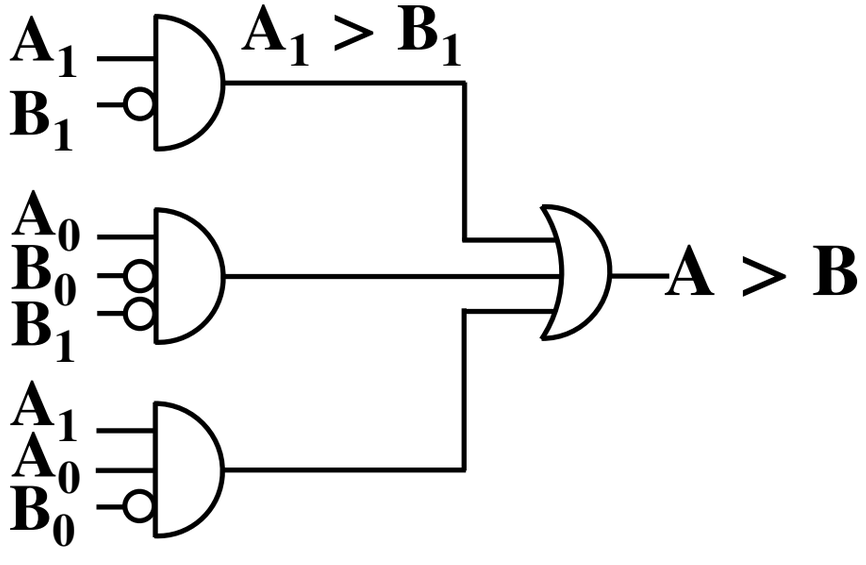
Ejemplo: Encontrar la función $\mathbf{A > B}$ para dos palabras de 2 bits, es decir un circuito que ponga su salida en 1 cuando $\mathbf{A > B}$, siendo $\mathbf{A}$ y $\mathbf{B}$ dos palabras de 2 bits cada una. O sea $\mathbf{A = A_1A_0}$ y $\mathbf{B = B_1B_0}$

Tabla de verdad para A > B en simultaneo

A_1	A_0	B_1	B_0	A > B
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



$$\mathbf{F_{min} = A_1\overline{B}_1 + A_0\overline{B}_0\overline{B}_1 + A_1A_0\overline{B}_0}$$

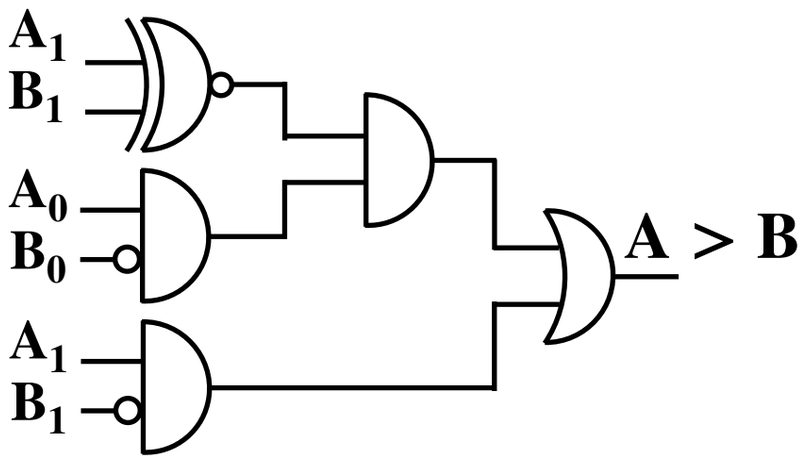


UNIDADES ARITMÉTICAS

Comparadores

Si lo planteamos ahora usando lógica proposicional, para que $\mathbf{A > B}$, en dos números de dos bits cada uno, se tiene que cumplir que:

A = B en ripple (en ondas) $\mathbf{A > B}$ **:** $\quad (A_1 > B_1) \vee \left[(A_1 = B_1) \wedge (A_0 > B_0) \right]$



Escrito con la función booleana llegamos a lo mismo.

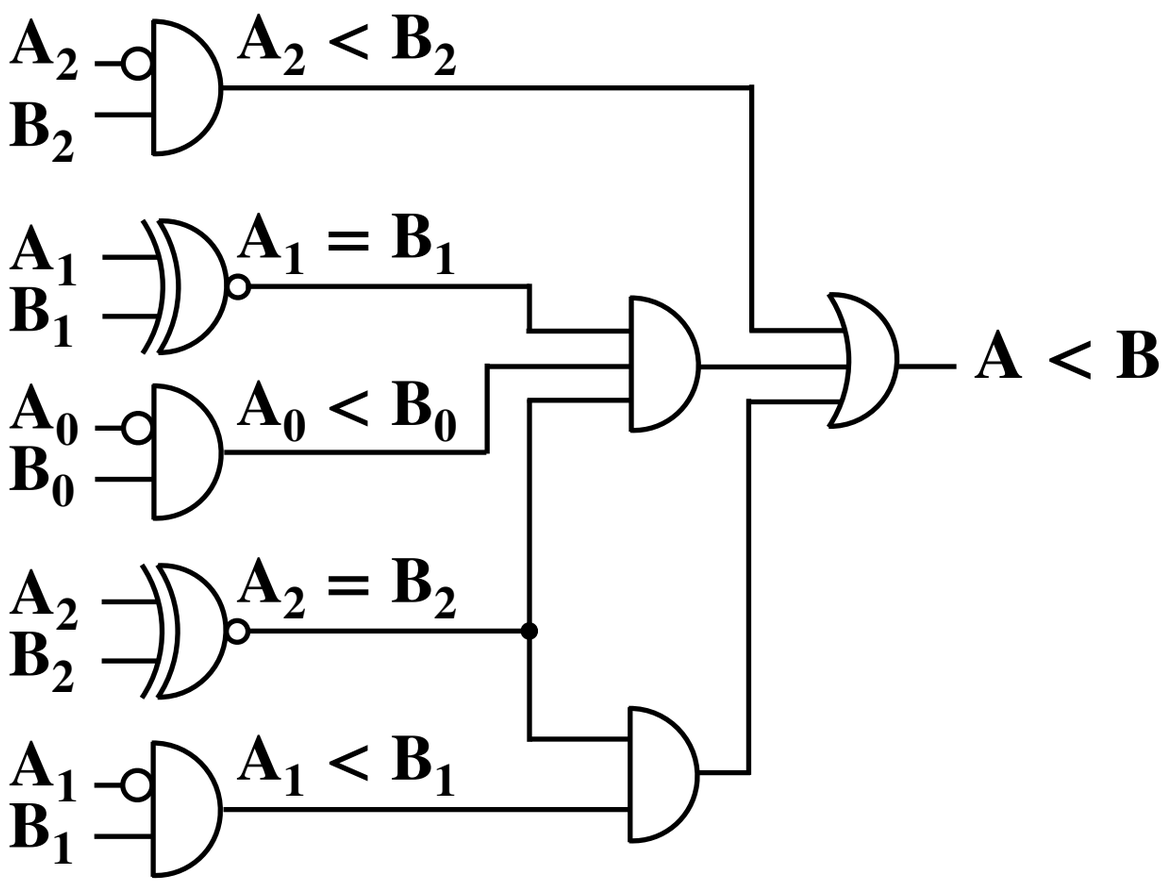
Esto conviene hacer cuando el número de variables aumenta (ver ejercicio 19).

UNIDADES ARITMÉTICAS

Comparadores

Ejercicio 18: Encontrar la función $\mathbf{A < B}$ para palabras de 3 bits

$$\mathbf{A < B:} \qquad (A_2 < B_2) \vee \left[(A_2 = B_2) \wedge (A_1 < B_1)\right] \vee \left[(A_2 = B_2) \wedge (A_1 = B_1) \wedge (A_0 < B_0)\right]$$



UNIDADES ARITMÉTICAS

Comparadores

Ejercicio 19: Encontrar la función $\mathbf{A > B}$ para palabras de 3 bits

$$\mathbf{A > B:} \qquad (A_2 > B_2) \vee \left[(A_2 = B_2) \wedge (A_1 > B_1)\right] \vee \left[(A_2 = B_2) \wedge (A_1 = B_1) \wedge (A_0 > B_0)\right]$$