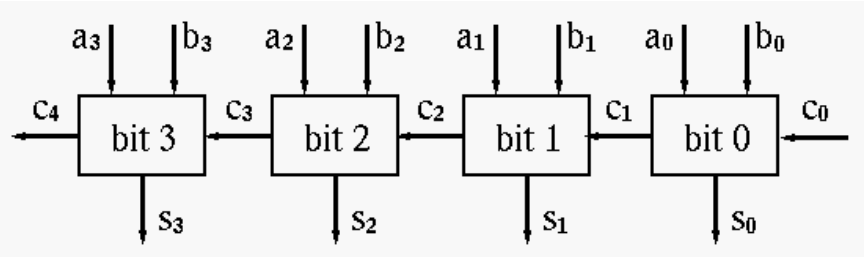
5.1.1 Sumadores con anticipación de Acarreo.

El sumador paralelo de n bits que se ha mostrado hasta ahora, tiene un nivel de retardo de 2*n puertas, pues necesita 2*n etapas de puertas lógicas para que las salidas queden completamente estabilizadas, al tener que calcular y propagar los acarreos entre todos los sumadores completos, como se muestra en la figura siguiente:



Pero existen otras implementaciones para ganar rapidez en el cálculo del resultado, que consisten en generar el acarreo para cada etapa sin esperar de que llegue de etapas anteriores, a este método se le llama **anticipación de acarreo**.

En el caso anteriormente visto el acarreo debía recorrer todas las etapas sumadoras para dar el resultado correcto, esta nueva implementación se basa en la generación adelantada del acarreo, circuito que se suele denominar anticipador de acarreo. Para ello se definen dos funciones lógicas g y p:

- Generador de acarreo: $g_i = a_i \cdot b_i$
- Propagador de acarreo: $p_i = a_i + b_i$

La función **g** informa de que en la etapa i se ha producido un acarreo. La función **p** informa que un acarreo de etapa anterior será propagado. Aplicando estas fórmulas al calculo del resultado de la suma y el acarreo para cada etapa, tenemos que:

$$c_i = c_{i-1} \cdot p_i + g_i$$

$$s_i = p_i + c_{i-1}$$

y desarrollando para cada acarreo obtenemos:

$$c_0 = g_0 + c_{-1} \cdot p_0$$

$$c_1 = g_1 + g_0 \cdot p_1 + c_{-1} \cdot p_0 \cdot p_1$$

$$c_2 = g_2 + g_1 \cdot p_2 + g_0 \cdot p_1 \cdot p_2 + c_{-1} \cdot p_0 \cdot p_1 \cdot p_2$$

$$c_3 = g_3 + g_2 \cdot p_3 + g_1 \cdot p_2 \cdot p_3 + g_0 \cdot p_1 \cdot p_2 \cdot p_3 + c_{-1} \cdot p_0 \cdot p_1 \cdot p_2 \cdot p_3$$

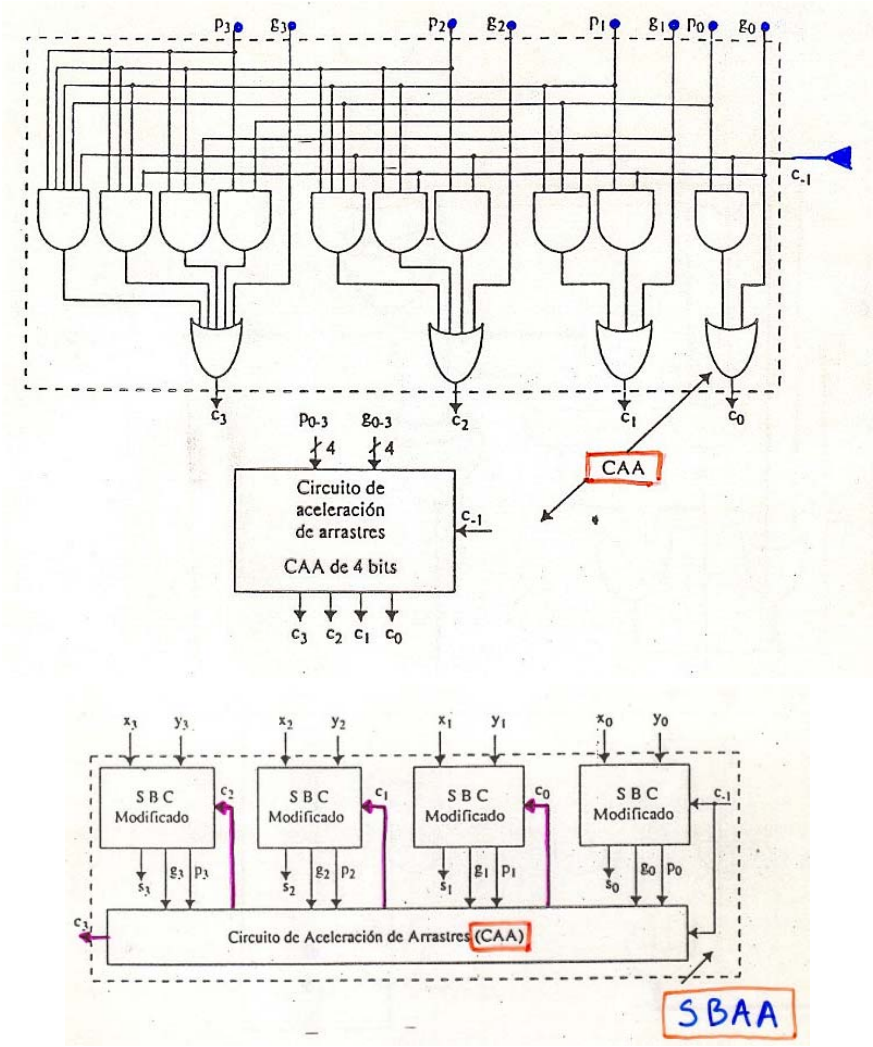
$$\cdots$$

$$c_i = g_i + g_{i-1} \cdot p_i + \ldots + g_0 \cdot p_1 \cdot p_2 \cdot \ldots \cdot p_i + c_{-1} \cdot p_0 \cdot p_1 \cdot \ldots \cdot p_i$$

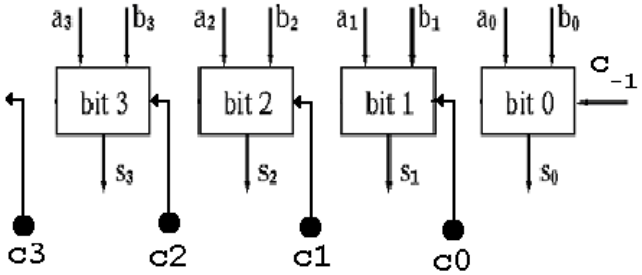
La función p nos da una noción de si por el bit considerado se puede transmitir un acarreo, mientras que g nos informa de si en ese bit se produce un acarreo. De esta forma es posible obtener todos los acarreos en el tiempo de tres retardos de puerta lógica después de que los operandos y el primer acarreo estén disponibles, el primero de los retardos vendrá dado por el calculo de g y p y dos para cada acarreo. Para obtener el resultado además tendremos que esperar dos retardos más para calcular **s**. Así independientemente del número de bits a sumar el tiempo de espera total para obtener la suma será siempre de cinco retardos de

puerta lógica. En el caso del sumador paralelo de n bits era de 2n retardos de puerta, ya que cada sumador tenía dos retardos y el acarreo debía recorrer n sumadores.

Lo podemos ver en la figura siguiente:



Con lo cual el circuito sumador con acarreo adelantado para números de 4 bits queda como aparece a continuación:



Un problema práctico que puede aparecer es el hecho de que las puertas lógicas tienen un número limitado de entradas. La expresión c_{i+1} requiere i+2 entradas al mayor término AND e i+2 al OR. Por lo general el número de entradas está restringido a 5 como máximo.

Si usamos cuatro de estos sumadores conectados en cascada, como en el caso del sumador paralelo de 4 bits, tenemos un sumador de 16 bits que requiere un tiempo de retraso total de 10 retrasos de puerta, ya que se requiere un retraso para g y p, dos por cada c_4, c_8, c_{12}, c_{15} y unos más para las sumas del último bloque de 4 bits. Veamos lo en la figura siguiente:

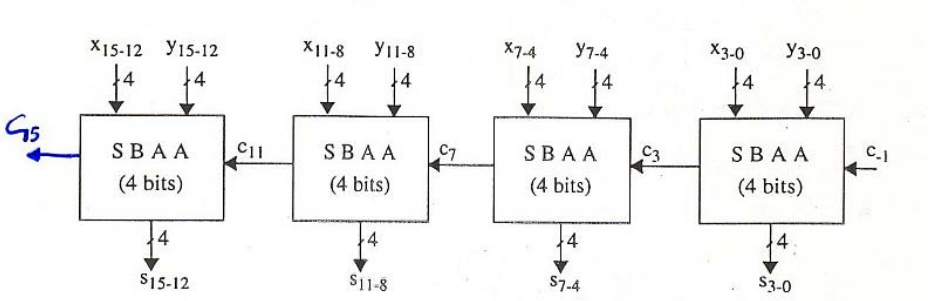


Figura 4-18 Sumador de 16 bits construido con 4 SBAA's de 4 bits

Además este tipo de sumador se puede acelerar aún más con la técnica del realizar un **segundo nivel de anticipación**. Para ello definimos dos $\mathbf{G_k^*}$ y $\mathbf{P_k^*}$ donde k indica el bloque de 4 bits que genera estas señales.

$$P_0^* = p_3p_2p_1p_0$$

$$G_0^* = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0$$

Dicho con otras palabras, si se dice g_i y p_i determina si una etapa i de bits genera o propaga un acarreo o no, entonces G_k^* y P_k^* determinan si el bloque k genera o propaga otro acarreo, o no. Si se dispone de estas nuevas funciones, no es necesario esperar que los acarreos se propaguen en cascada a través de bloques para la generación de los nuevos acarreos. Veámoslo:

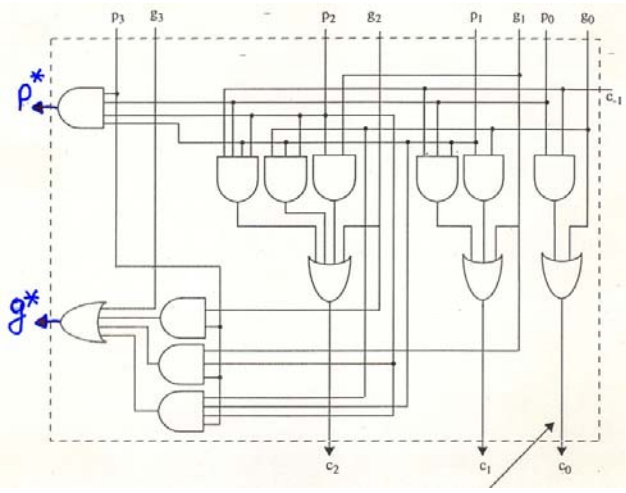
$$c_3 = G_0^* + P_0^* c_{-1}$$

$$c_7 = G_1^* + P_1^* G_0^* + P_1^* P_0^* c_{-1}$$

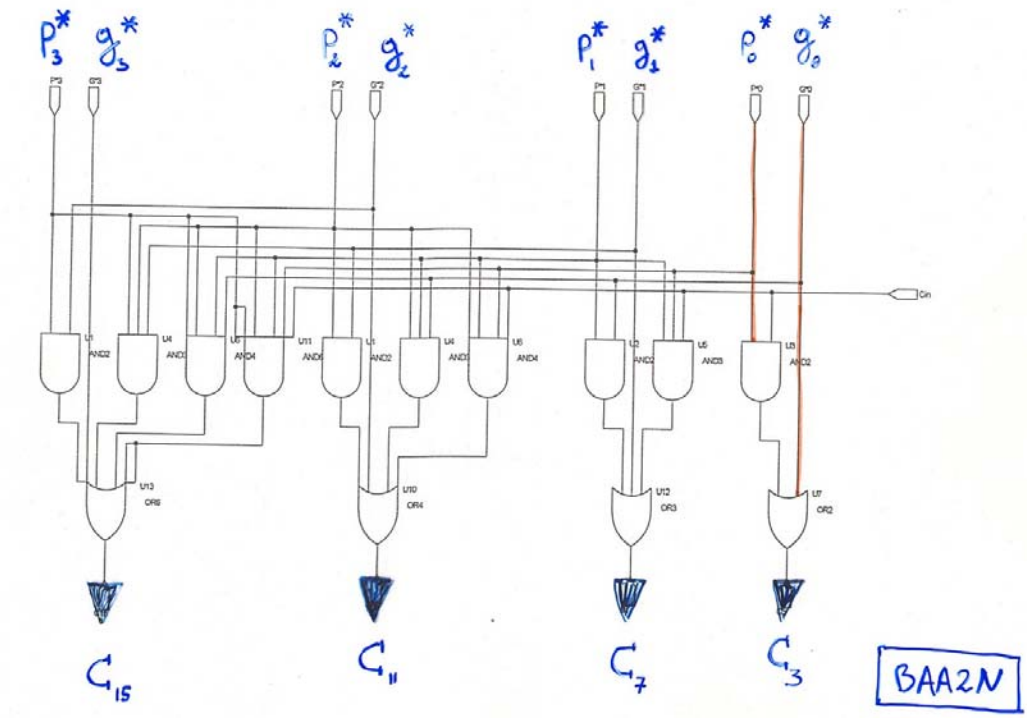
$$c_{11} = G_2^* + P_2^* G_1^* + P_2^* P_1^* G_0^* + P_2^* P_1^* P_0^* c_{-1}$$

$$c_{15} = G_3^* + P_3^* G_2^* + P_3^* P_2^* G_1^* + P_3^* P_2^* P_1^* G_0^* + P_3^* P_2^* P_1^* P_0^* c_{-1}$$

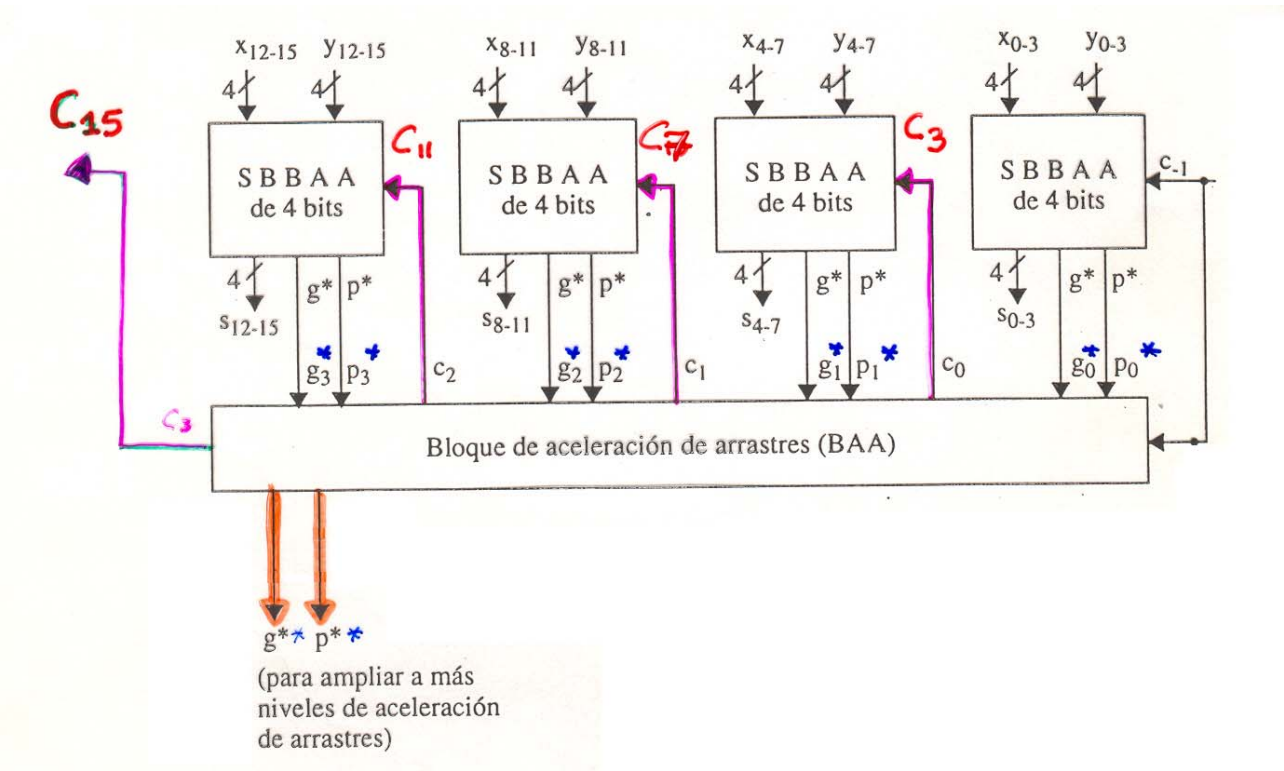
En la figura siguiente apreciamos como cambia el primer nivel de anticipación para calcular las funciones p* y g*:



y en la siguiente imagen vemos como quedan definidas las funciones de los acarreos anticipados de bloque de 4 bits:

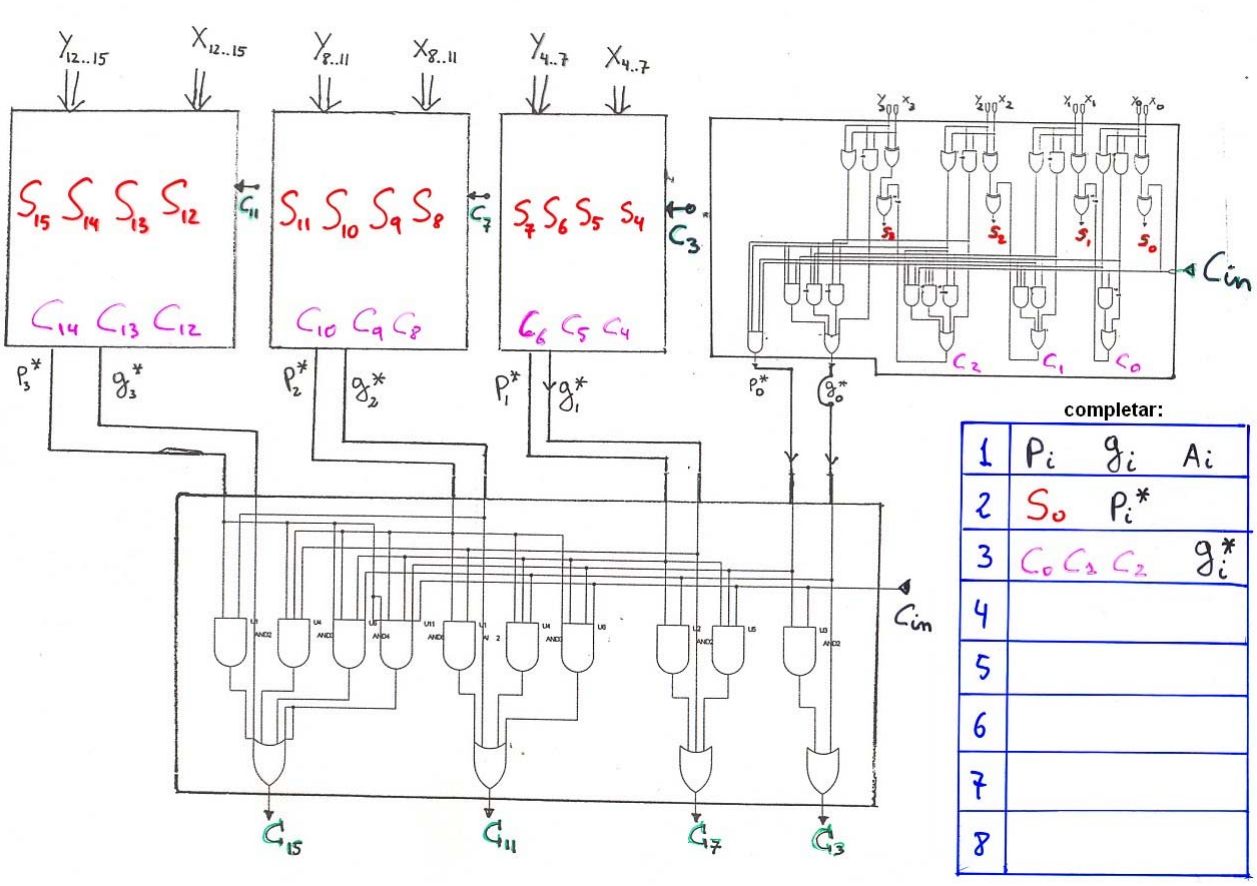


Con todo esto, el diseño por bloques del sumador de 16 bits con dos niveles de anticipación queda como se muestra en la figura siguiente:



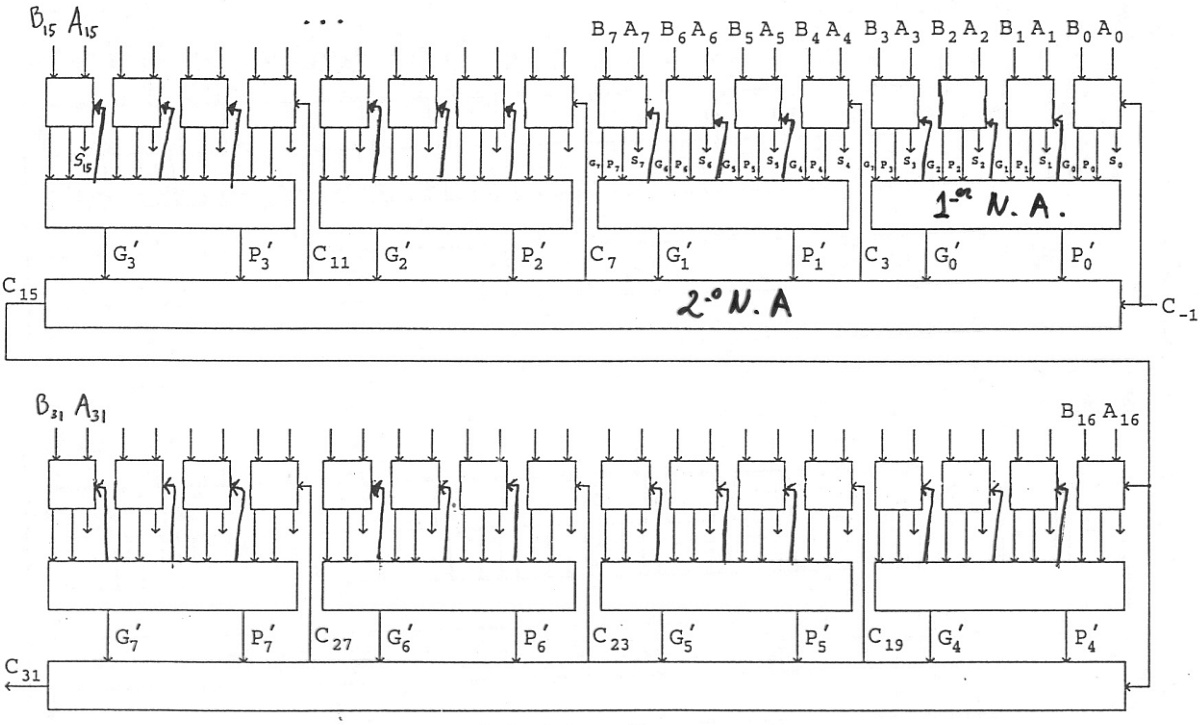
Detalle de un Sumador Rápido de 16 bits. (con dos niveles de anticipación de acarreo).

Para construir un sumador rápido de 16 bits se puede utilizar cuatro módulos de sumadores rápidos de 4 bits y dos niveles de anticipación, tal como aparece en la figura siguiente:

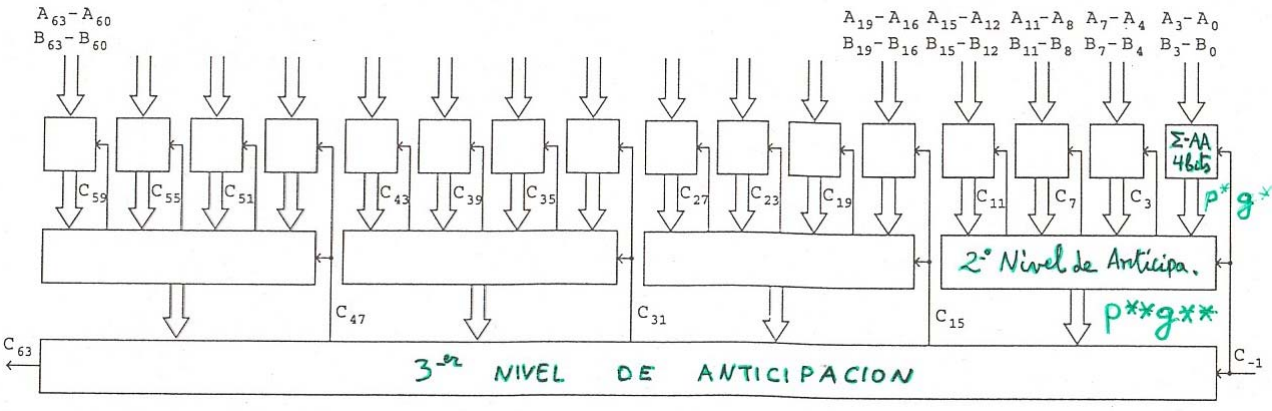


Completar la tabla lateral con las funciones lógicas.

Ejercicio P1: Realizar la tabla de tiempos de un sumador de números de 32 bits y de otro de 64 bits que utilizan dos niveles de anticipación. Realización en puertas lógicas de todo el circuito. A continuación se muestra el diseño detallado del de 32 bits.



Ejercicio P2: Realizar la tabla de tiempos de un sumador de números de 64 bits que utiliza tres niveles de anticipación. Además mostrar las funciones del tercer nivel de anticipación y la realización en puertas lógicas de todo el circuito. *(para este último apartado vendrá bien fotocopiar los circuidos vistos con anterioridad y replicarlos).*



Restador.

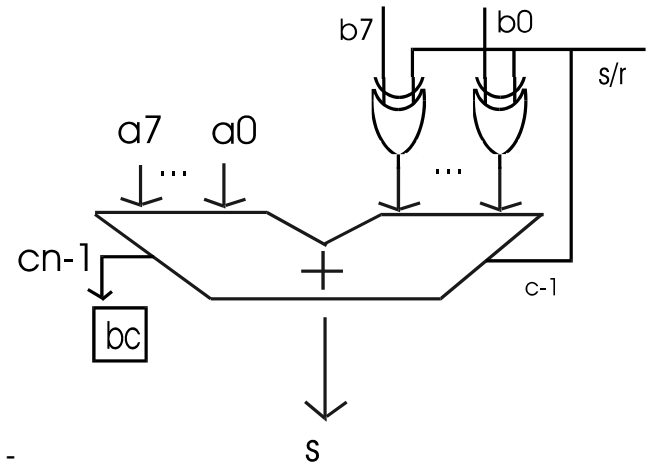
En el caso de la **resta** podemos utilizar un restador elemental y actuar igual que para el caso de la suma, es decir, a continuación ir ampliando mediante n circuitos restadores elementales. Pero ya que hemos introducido el sumador parece más útil utilizarlo para realizar la resta también esto es así debido a:

$$A - B = A - B + 2^n - 2^n$$

o bien

$$A - B = A + (2^n - B) - 2^n$$

Lo que supone que para realizar la resta se debe realizar el complemento a dos de B, sumar el valor de A y restar 2^n. Para realizar el complemento a dos de B debemos realizar la negación lógica de B y sumar uno al resultado. La negación se puede realizar mediante una etapa XOR y la suma introduciendo un 1 en el primer acarreo c_{-1}. Por otro lado, restar 2^n al resultado sería restar uno al bit s_n inexistente por lo cual lo único que debemos hacer es ignorar el bit de acarreo c_{n-1} producido en la suma. Así el esquema del sumador restador se muestra a continuación, la entrada s/r=0 realiza operación de suma y s/r=1 es operación de resta:



Sumador restador.

Ejemplo:

```
  0 0 1 0
+ 1 0 1 1
  1 1 0 1
```

- En complemento a 2 la operación anterior sería: (1011)=-5, (0010)=2 y resultado (1101)=-3.
- Para el caso de números sin signo sería: (1011)=11, (0010)=2 y resultado (1101)=13.