ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

PRACTICA 1

Alumno/s: Dickinson Bedoya Perez,
Anna Gracia Colmenarejo
Asignatura: Estructura de computadores
Enseñamiento:Ing. Informática
Profesor/es:Carles Aliagas, Carlos Molina
Fecha:2021

ÍNDICE

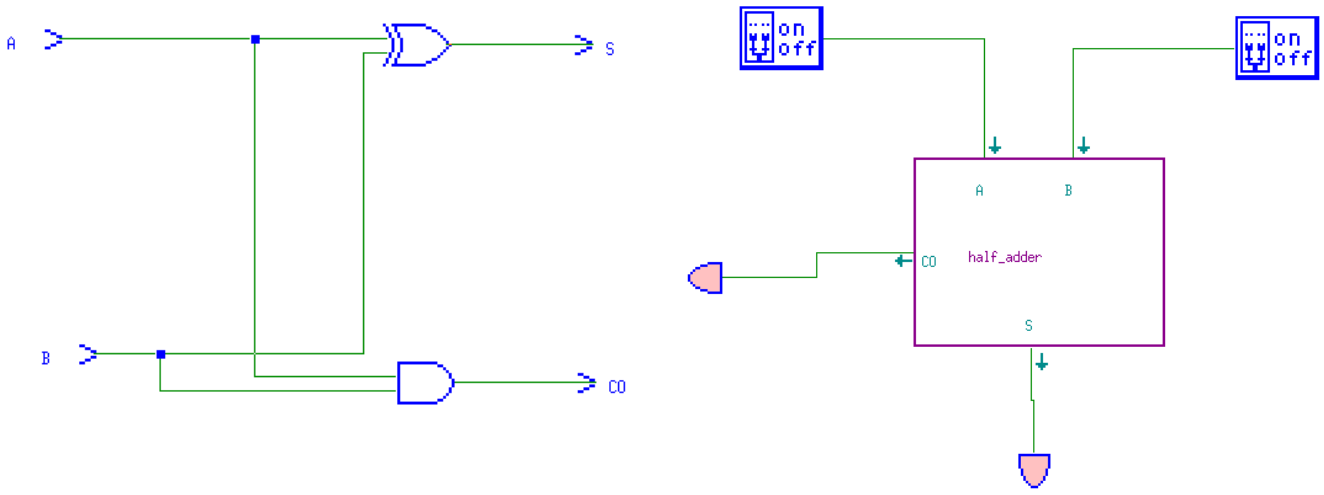
FASE 1: Sumadores HA, FA

TAREA 1: HALF ADDER

El *Half-Adder* (HA) es un sumador de 1 bit que tiene 2 entradas *A* y *B*, los dos bits a sumar, y 2 salidas el resultado de la suma y el carry que genera la suma.

Para generar el *Half-Adder* hemos puesto las dos entradas *A* y *B*, un conmutador para cada una, por las que entrarán los dos bits a sumar. Para calcular el resultado de la suma, las dos entradas pasarán por una puerta lógica XOR. Esta puerta devuelve 1 solo cuando una de las dos entradas es 1, es la que se usa para la implementación de la adición binaria. Finalmente para calcular si la suma lleva *Carry,* ambas entradas A y B pasan por una puerta AND que solamente devuelve 1 cuando ambas entradas sean 1, esta situación es la que nos generará Carry (ya que al sumar 1 y 1 nos da $10_{(b)}$) porque este resultado es de dos bits y el *HA* es de 1 bit nos estaríamos "llevando uno" que lo llamamos *CarryOut*.

Para las dos salidas *CarryOut* y el resultado de la suma (S) hemos puesto dos leds uno para cada una.



Para probar el funcionamiento del circuito hemos realizado el juego de pruebas realizando diferentes sumas de 1 bit y comprobando a la vez con la tabla de la verdad.

ENTRADAS		**SALIDA ESPERADA**		**SALIDA OBTENIDA**	
A	**B**	**S**	**C**	**S**	**C**
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	1	0	1

El área del *Half-Adder* sería la suma de las dos puertas lógicas que lo forman, la AND que ocupa 6 y la XOR que ocupa 8, esta suma (6+8) nos daría que el área es 14.

A_HA = **K * (XOR+AND) = K * (8 + 6) = 1 * (14) = 14**

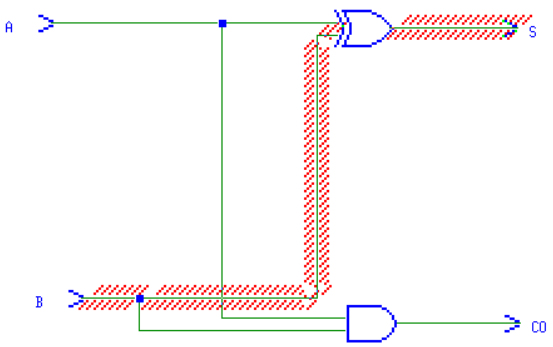
Siendo la K un bit

TIEMPOS TEÓRICOS Y REALES

Para la salida *S* hemos calculado que el tiempo sería S = XOR = 3T porque es la suma del retardo de las puertas por las que pasa y para la salida *C* sería C = AND = 2T. Con estos cálculos sacamos que el camino crítico del circuito sería de 3T.

R_HA = XOR = 3T

Con la interfície TKGate hemos comprobado que los tiempos reales se corresponden a los teóricos calculados previamente. Donde S = 3T y C = 2T.

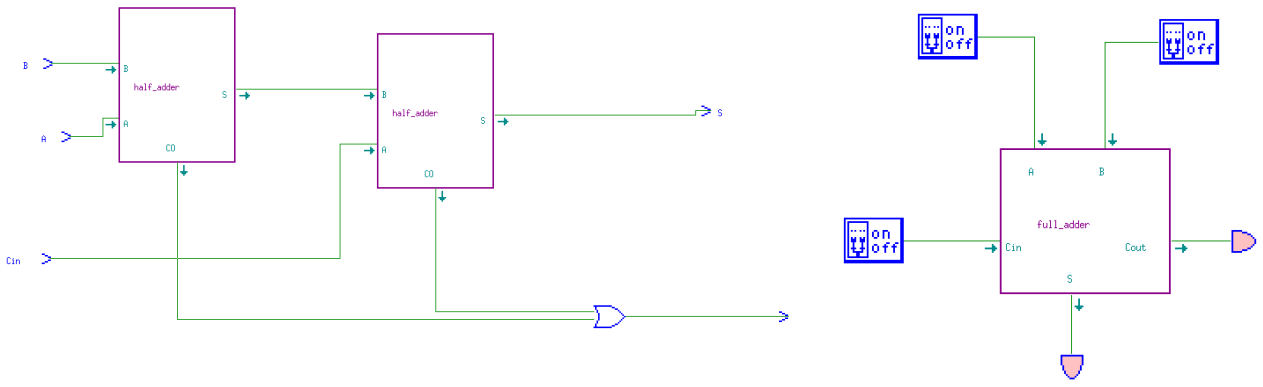


TAREA 2: FULL ADDER

El *Full-Adder* es un sumador de un bit muy parecido al *Half-Adder* pero que tiene 3 entradas y dos salidas. Las entradas son *A* y *B* (los dos números de 1 bit a sumar) y un *Carry* de entrada, es decir, podemos sumar tres números de un bit. El *Carry* de entrada es un bit trasladado de la etapa anterior, corresponde al bit que "nos llevamos" por ejemplo al sumar 1+1 en binario. Y las dos salidas son el resultado de la suma *S* y el *Carry* de salida.

Para la implementación del *Full-Adder* se pueden utilizar dos Half-Adders. En el primero se calcula A+B, el resultado de esta suma será una de las entradas del segundo *HA* y el *Carry* de salida de este primero se llevará a una puerta OR con la que posteriormente se calculará el *Carry* de salida total del *FA*. En el segundo *HA*, como hemos dicho antes la primera entrada es el resultado de A+B y la segunda será el *Carry* de entrada del *FA*, el resultado de esta suma será el total de la suma (S) del *FA* y el *Carry* de salida se llevará igual que el anterior a la puerta OR que nos indicará si hay *Carry* en el *FA*.

A partir del circuito dado en el enunciado sacamos el Full-Adder mediante dos módulos de *Half-Adders*.



Para probar el funcionamiento del circuito hemos realizado el juego de pruebas realizando diferentes sumas de 1 bit y comprobando a la vez con la tabla de la verdad.

ENTRADAS			SALIDA ESPERADA		SALIDA OBTENIDA	
A	**B**	**Cin**	**S**	**Cout**	**S**	**Cout**
0	0	0	0	0	0	0
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	1	1	1	1

El área del *Full-Adder* sería la suma del área de los dos módulos *HA* y la última puerta OR que tiene área 6 (según el tkgate).

A_FA = K * (2*A_HA + OR) = K * (2*14+6) = K*(34) = 34

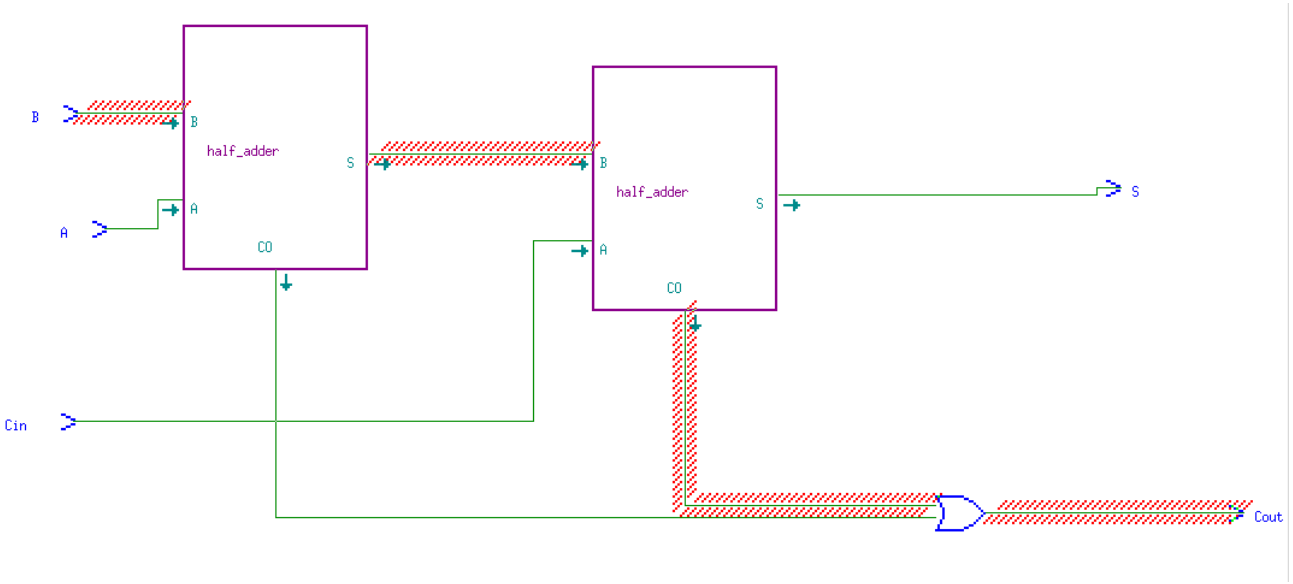
Siendo K un bit.

TIEMPOS TEÓRICOS Y REALES

Para la salida S tenemos que S = XOR + XOR = 6T y para C tenemos C = XOR + AND + OR = 7T. Por lo tanto el camino crítico del *FullAdder* será el de la salida C:

R_FA = XOR + AND + OR = 3 + 2 + 2 = 7T

Viene dado según los retardos de las puertas en las que se genera el retardo más grande, siendo este el camino crítico.



Contrastando estos resultados con los valores que da el tkgate vemos que son los correctos.

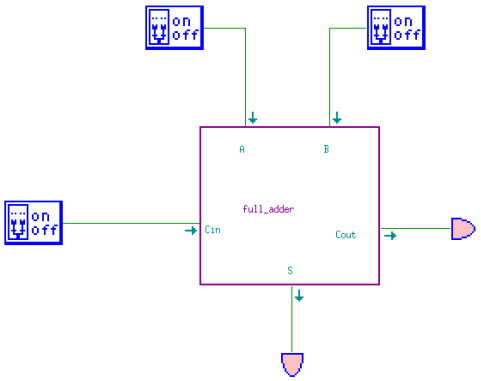
Estimated area=34.

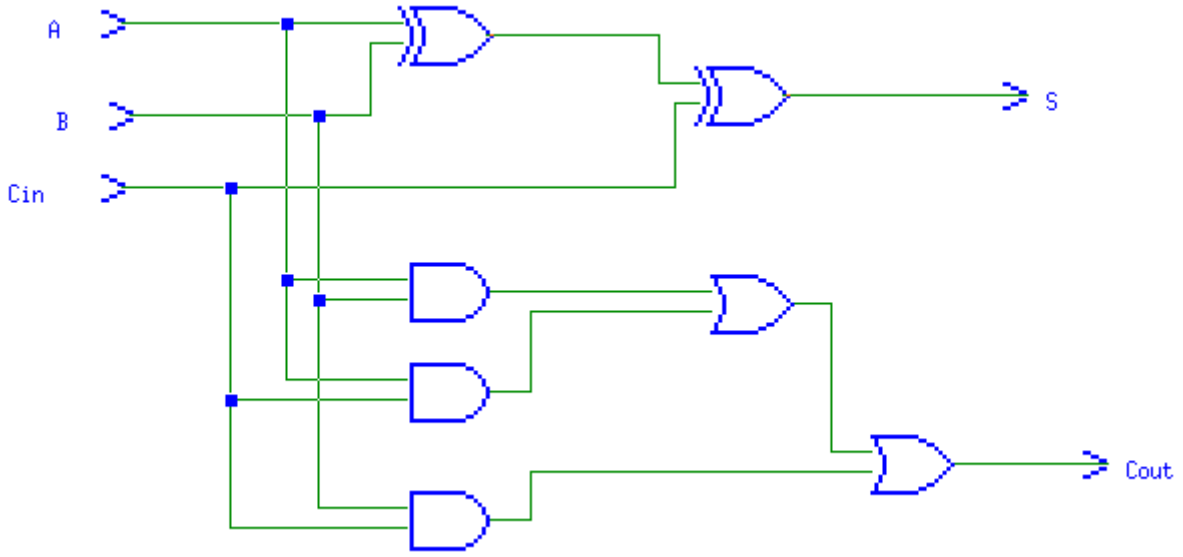
Retard del camí: 7

TAREA 3: FULL ADDER

Para la implementación de este tipo de *Half-Adder* se sigue la misma base que el anterior, un módulo con tres entradas (A, B y Cin) y dos salidas (S, Cout). La diferencia que tiene este con el anterior es que no se usan dos Half-Adders sino que se propone otro circuito distinto, compuesto por puertas lógicas. Dos puertas OR que realizarán la suma, la primera hará A+B y la segunda el resultado de A+B + Cin. Estas dos sumas nos darán el resultado de la suma total por la salida S.

Para comprobar que hay Carry de salida, cada una de las entradas se lleva a una de las puertas AND en las que se irán comprobando con las otras dos entradas, las salidas de estas 3 puertas AND se comprobarán pasando por dos puertas OR que que indicarán si hay Carry ya que devolverán 1 siempre que alguna AND lo devuelva.





Para probar el funcionamiento del circuito hemos realizado el juego de pruebas realizando diferentes sumas de 1 bit y comprobando a la vez con la tabla de la verdad.

ENTRADAS			SALIDA ESPERADA		SALIDA OBTENIDA	
A	**B**	**Cin**	**S**	**Cout**	**S**	**Cout**
0	0	0	0	0	0	0
0	0	1	1	0	1	0

0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	1	1	1	1

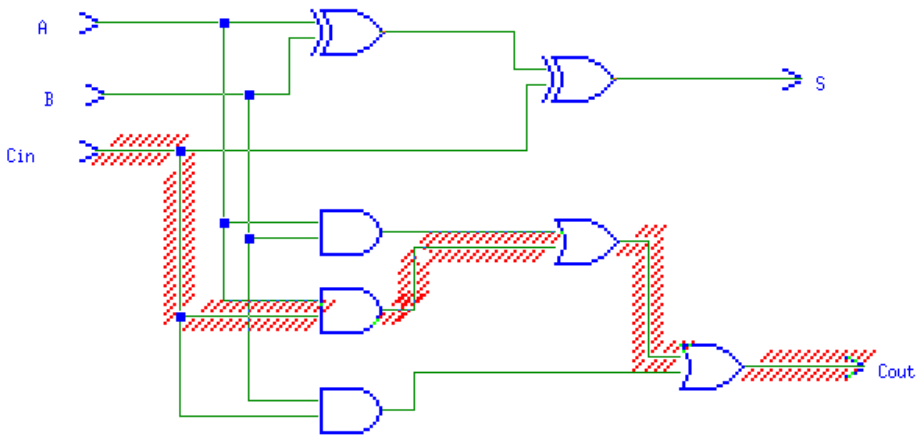
El área del Full-Adder sería la suma de las cinco puertas lógicas que lo forman, las tres AND que ocupa 6 cada una, la XOR de tres entradas que ocupa 11 y la OR de tres entradas que ocupa 8:

A_FA = (3*AND + 2*XOR + 2*OR) = 18 + 16 + 12 = 46

TIEMPOS TEÓRICOS Y REALES

Para la salida S tenemos que **S = 2*XOR = 6T** y para C tenemos **C = AND + 2*OR = 6T**. Por lo tanto el camino crítico del *FullAdder* será el de la salida C o la de S, ya que ambas tienen el mismo retardo:

R_FA = 6T

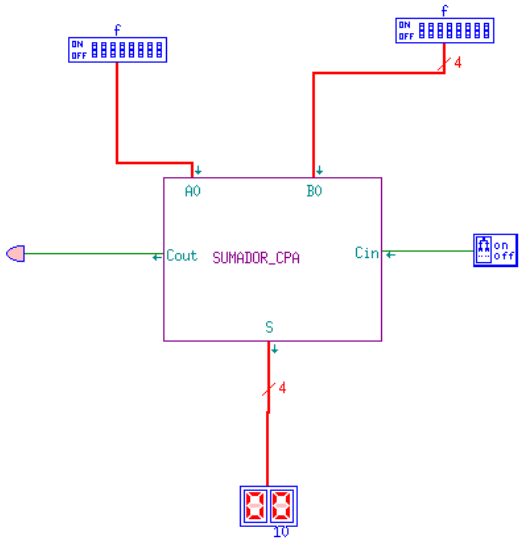


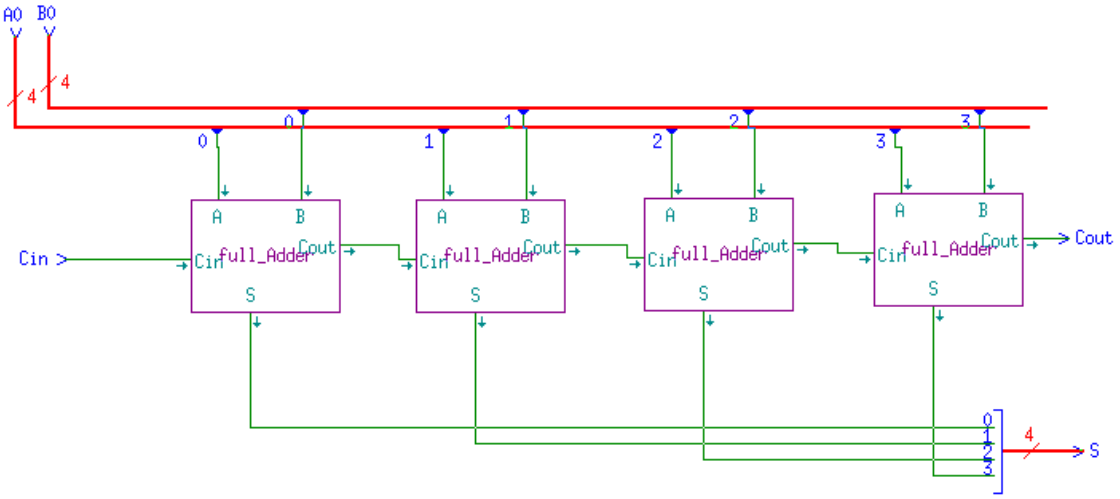
FASE 2: Sumadores CPA

El *Carry Propagate Adder,* también conocido como *Ripple Carry Adder,* es un sumador de números de *N* bits. El *CPA* tiene 3 entradas y dos salidas. En las entradas tenemos *A* que es el número de *N* bits a sumar, *B* el otro número de *N* bits a sumar y *Cin*, de 1 bit, es la entrada que corresponde al *Carry* que estaría arrastrando de una etapa anterior. En las salidas tenemos *S*, de *N* bits, que devolvería el resultado de la suma y *Cout*, de 1 bit, que devolvería si la suma que se ha realizado tiene *Carry*.

TAREA 4: Carry Propagate Adder (4 bits)

Su implementación consta de colocar tantos *Full-Adders* en cascada como bits tengan los números a sumar, en nuestro caso hemos realizado un *CPA* de 4 bits (N=4), por lo tanto, tenemos 4 *Full-Adders*. Para este *CPA* los *FA* que hemos utilizado son los que están implementados a partir de *HA* (Tarea2). Cada *FA* se encarga de realizar la suma de dos bits uno de *A* y otro de *B*, por ejemplo el bit 3 de *A* con el bit 3 de *B*, si este *FA* recibiese un *Carry* de entrada lo sumaría también y devolvería el resultado de la suma que correspondría a un bit de *S* (resultado de la suma del *CPA*), por ejemplo el bit 3 de *S*. El *Carry* de salida que generan se va propagando a la etapa siguiente, es decir el *Carry* de salida de un *FA* será el *Carry* de entrada del siguiente.





Para probar el funcionamiento del circuito hemos realizado un juego de pruebas realizando diferentes sumas de 4 bits que hemos calculado previamente a mano y comprobado luego simulando el circuito en el TKGate. *A* y *B* están en hexadecimal, *Cin* en binario, *S* en decimal y *Cout* en binario.

ENTRADAS			**SALIDA ESPERADA**		**SALIDA OBTENIDA**	
A	**B**	**Cin**	**S**	**Cout**	**S**	**Cout**
0	0	0	0	0	0	0
F	F	0	14	1	14	1
F	F	1	15	1	15	1
1	1	1	3	0	3	0
A	B	1	6	1	6	1
F	3	0	2	1	2	1
D	7	1	5	1	5	1

4	7	0	11	0	11	0
E	9	0	7	1	7	1

El área del *CPA* sería la suma de :

A_CPA = **k*(A_FA)= 4*(34)= 136**

K corresponde al número de bits del *CPA* que será el número de *FA* que deberemos colocar, en este caso es 4.

TIEMPOS TEÓRICOS Y REALES

Para calcular los tiempos teóricos de las 2 salidas *S* y *Cout:*

Inicialmente tenemos que un *FA* tarda 6T en calcular su salida *S* = XOR +XOR = 3T+3T = 6T*,* pero la salida *S* del *CPA* no correspondría a sumar 6T por cada FA que tengamos, ya que la suma de todos los bits de A y B se realizan a la vez en el tiempo 0. Por lo que cada señal de S_i tardará 6T menos el retardo de los cálculos solapados anteriormente.

Y para calcular la salida *Cout* nos pasaría los mismo ya que para cada señal C_i también necesitamos la suma de A_i+B_i, entonces tenemos que para calcular C = XOR + AND + OR = 7T, por lo que nos quedaría que C_i = 7T menos el retardo de los cálculos solapados anteriormente. En ambas salidas se solapa la suma de Ai y Bi que se realiza con una XOR=3T.

Lo que hemos explicado anteriormente sobre como calcular el retardo de cada salida podríamos resumirlo en la siguiente fórmula.

S_3 = TC(bit0) + (n-2 bits)*(TC-ParteParalela) + (TS-ParteParalela) = 7T + 2*(7T-3T) + (6T-3T) = 18T

C_4 = TC(bit0) + (n-1 bits)*(TC-ParteParalela) = 7T + 3*(7T-3T) = 19T

Donde TC es el tiempo de Carry, n es número de FA, TS es el tiempo de S y la ParteParalela es la parte que ya se ha realizado al comienzo.

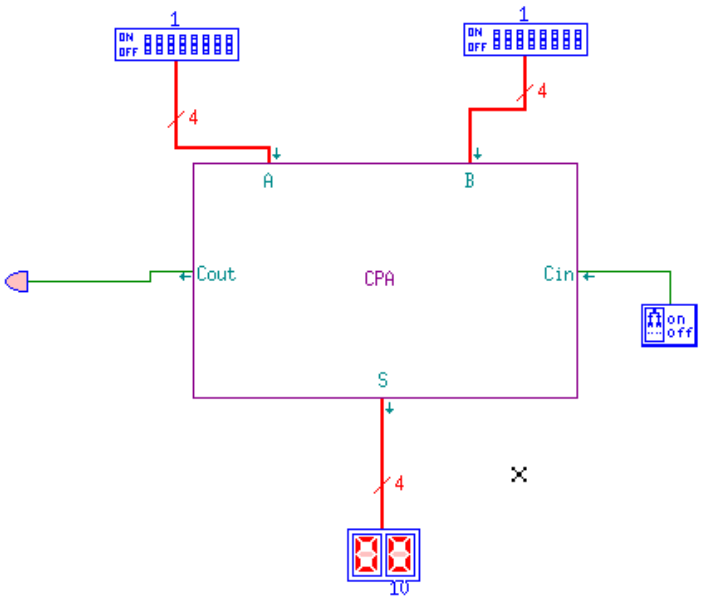
Para saber el camino crítico directamente habría que calcular los retardos de las señales S_3 y C_4 ya que son las últimas en ser calculadas porque ambas salidas necesitan de sus etapas anteriores. Entonces el camino crítico sería el de C_4.

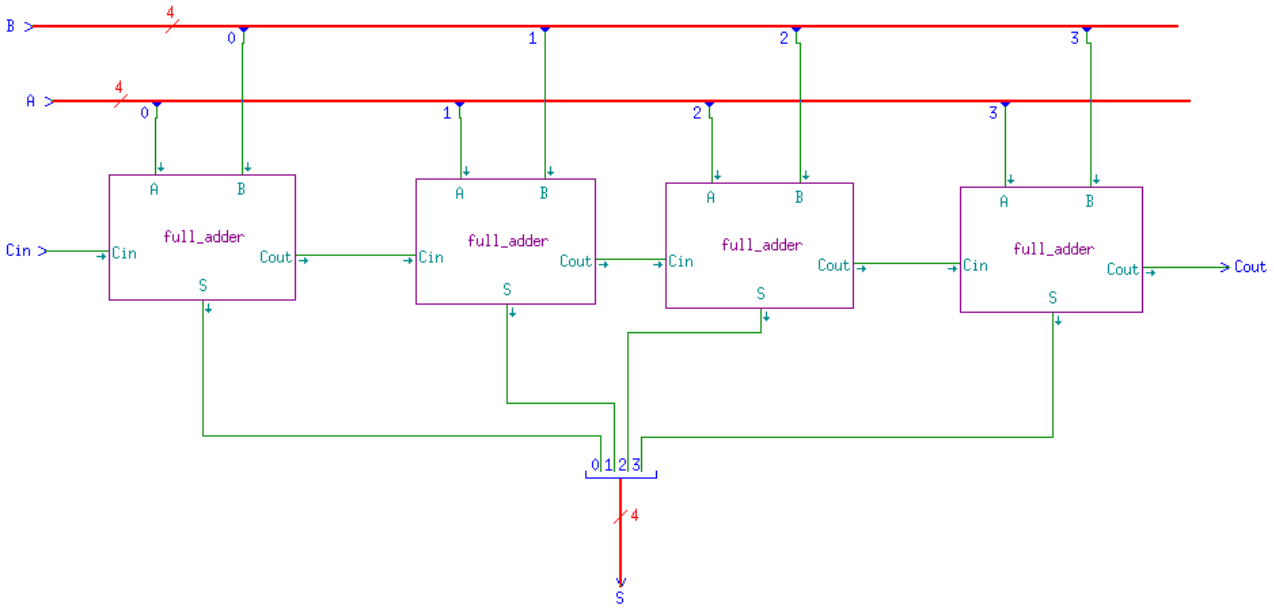
R_CPA = TC(bit0) + (n-1 bits)*(TC-ParteParalela) = 7T + 3*(7T-3T) = 19T

Finalmente comprobando los retardos de C i S con el TKGate podemos asumir que los cálculos son correctos porque coinciden los tiempos reales y teóricos.

TAREA 5: Carry Propagate Adder (4 bits)

En este Carry Propagate Adder en comparación al de la tarea anterior sólo hemos cambiado los FA que lo forman. En este caso los FA que hemos utilizados no están implementados a partir de *HA* sinó que están implementados con puertas lógicas (Tarea 3).





Para probar el funcionamiento del circuito hemos realizado un juego de pruebas realizando diferentes sumas de 4 bits que hemos calculado previamente a mano y comprobado luego simulando el circuito en el TKGate. *A* y *B* están en hexadecimal, *Cin* en binario, *S* en decimal y *Cout* en binario.

ENTRADAS			**SALIDA ESPERADA**		**SALIDA OBTENIDA**	
A	**B**	**Cin**	**S**	**Cout**	**S**	**Cout**
0	0	0	0	0	0	0
0	F	1	0	1	0	1
F	F	0	14	1	14	1
F	F	1	15	1	15	1
1	1	1	3	0	3	0
A	B	1	6	1	6	1
F	3	0	2	1	2	1

D	7	1	5	1	5	1
4	7	0	11	0	11	0
E	9	0	7	1	7	1

El área del *CPA* sería la suma de :

A_CPA = k*(A_FA) = 4*(46) = 184

K corresponde al número de bits del *CPA* que será el número de *FA* que deberemos colocar, en este caso es 4.

TIEMPOS TEÓRICOS Y REALES

Para esta implementación del CPA tenemos que los retardos del *FA* son C = 6T y S = 6T.

S_3 = TC(bit0) + (n-2 bits)*(TC-ParteParalela) + (TS-ParteParalela) = 6T + 2*(6T-0T) + (6T-3T) = 21T

C_4 = TC(bit0) + (n-1 bits)*(TC-ParteParalela) = 6T + 3*(6T-0T) = 24T

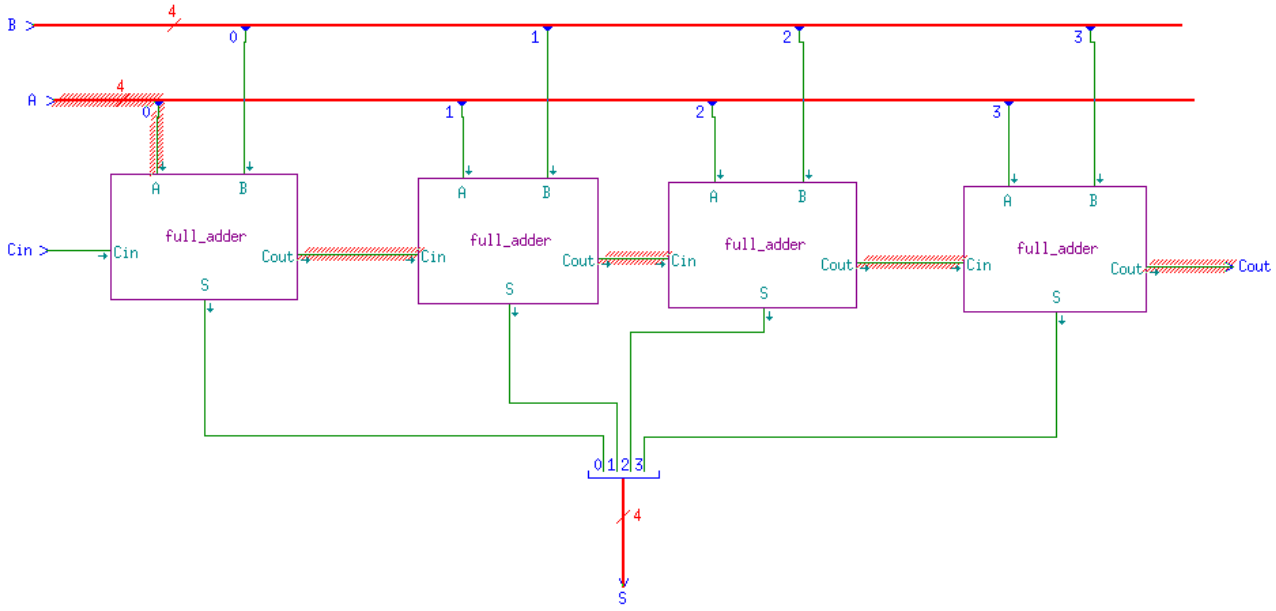
Donde TC es el tiempo de Carry, n es número de FA, TS es el tiempo de S y la ParteParalela es la parte que ya se ha realizado al comienzo.

Para la salida S la parte paralela corresponde a la XOR que calcula A + B, en cambio para la salida C no tenemos parte paralela.

Para saber el camino crítico directamente habría que calcular los retardos de las señales *S_3* y *C_4* ya que son las últimas en ser calculadas porque ambas salidas necesitan de sus etapas anteriores. Entonces el camino crítico sería el de *C_4*.

R_CPA = TC(bit0) + (n-1 bits)*(TC-ParteParalela) = 6T + 3*(6T-0T) = 24T

Finalmente comprobando los retardos de C i S con el TKGate podemos asumir que los cálculos son correctos porque coinciden los tiempos reales y teóricos.



TAREA 6: Fórmulas de los retardos del CPA

Las fórmulas para calcular el retardo del circuito CPA son las siguientes:

S = TC(bit0) + (n-2 bits)*(TC-ParteParalela) + (TS-ParteParalela)

C = TC(bit0) + (n-1 bits)*(TC-ParteParalela)

Donde n es el número de bits que se va a usar en ese CPA.

CPA TAREA 4:

C(4 bits) = 7T + 3*(7T-3T) =7T + 3*4T = **19T**

S(4 bits) = 7T + 2*(7T-3T) + (6T-3T) =7T + 2*4T + 3 = **18T**

C(8 bits) = 7T + 7*4T = **35T**

S(8 bits) = 7T + 6*4T + 3 = **34T**

C(16 bits) = 7T + 15*4T = **67T**

S(16 bits) =7T + 6*4T + 3 = **66T**

C(32 bits) = 7T + 31*4T = **131T**

S(32 bits) = 7T + 30*4T + 3 = **130T**

C(64 bits) = 7T + 63*4T = **259T**

S(64 bits) = 7T + 62*4T + 3 = **258T**

C(128 bits) = 7T + 127*4T = **515T**

S(128 bits) = 7T + 126*4T + 3 = **514T**

CPA TAREA 5:

$C_{(4\ bits)}$ = 6T + 3*(6T-0T) = 6T + 3*6 = **24T**

$S_{(4\ bits)}$ = 6T + 2*(6T-0T) + (6T-3T) = 6T + 2*6T + 3T = **21T**

$C_{(8\ bits)}$ = 6T + 7*6 = **48T**

$S_{(8\ bits)}$ = 6T + 6*6T + 3T = **45T**

$C_{(16\ bits)}$ = 6T + 15*6 = **96T**

$S_{(16\ bits)}$ = 6T + 14*6T + 3T = **93T**

$C_{(32\ bits)}$ = 6T + 31*6 = **192T**

$S_{(32\ bits)}$ = 6T + 30*6T + 3T = **189T**

$C_{(64\ bits)}$ = 6T + 63*6 = **384T**

$S_{(64\ bits)}$ = 6T + 62*6T + 3T = **381T**

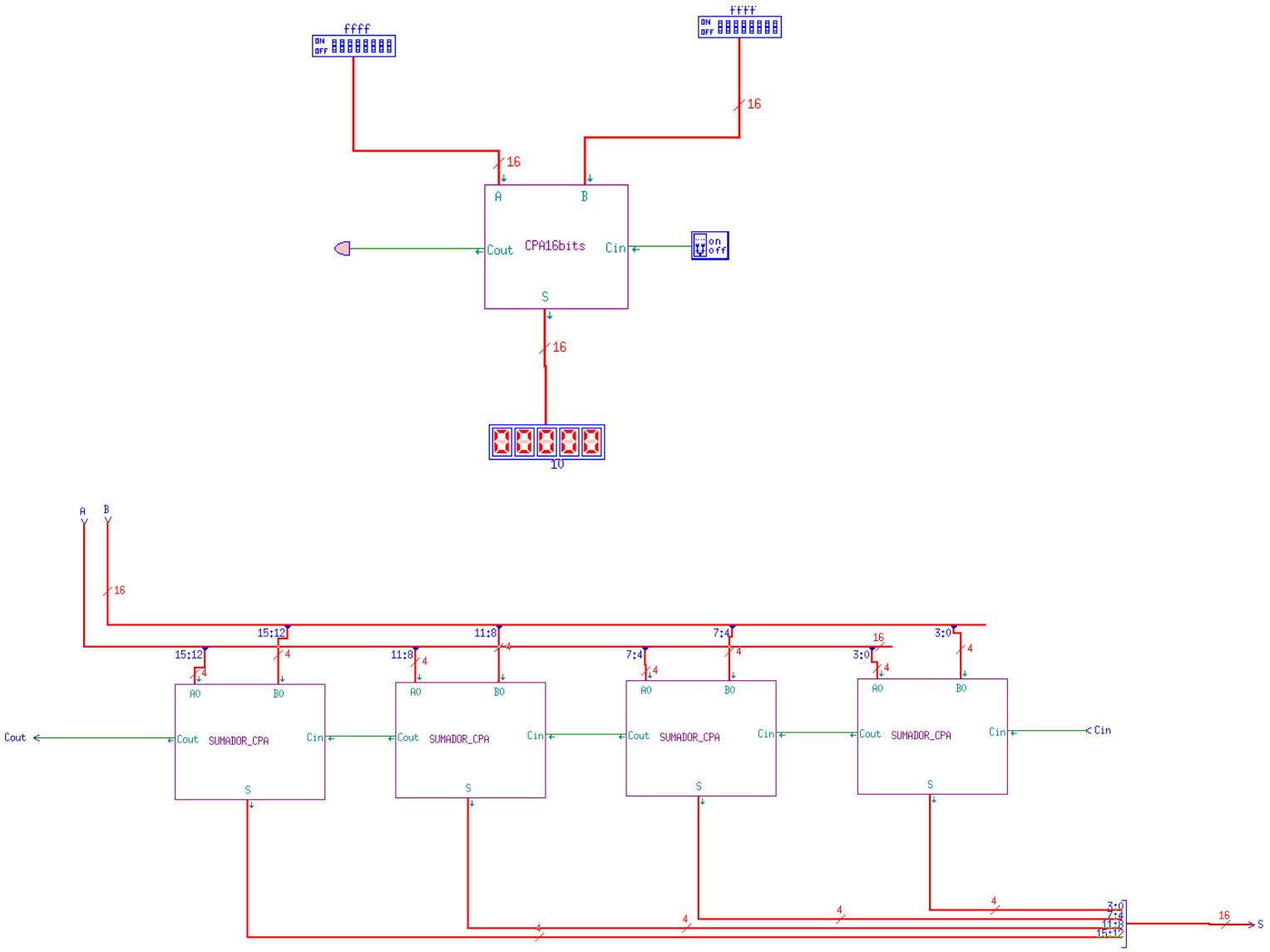
$C_{(128\ bits)}$ = 6T + 127*6 = **768T**

$S_{(128\ bits)}$ = 6T + 126*6T + 3T = **765T**

Como se puede ver en los cálculos, el que tiene mejores resultados en los retardos es el primer CPA, el que se compone del FA de la tarea 2, prefiriendo este en vez del segundo.

TAREA 7: Carry Propagate Adder (16 bits)

Para la implementación del CPA de 16 bits se sigue la estructura de un CPA de 4 bits, tres entradas, las dos de suma y una de carry, y dos salidas, la suma y el carryout. Para esto se han colocado cuatro CPA de 4 bits implementados anteriormente, por lo tanto, tenemos 16 *Full-Adders* (uno por cada CPA de 4 bits). Cada CPA obtiene un rango de 4 bits del bus de la entrada correspondiente y realiza la suma con esos 4 bits. El *Carry* de salida que generan se va propagando a la etapa siguiente, es decir el *Carry* de salida de un *CPA* será el *Carry* de entrada del siguiente CPA. El resultado de la suma se agrupa en un "*Linies Bus*" para tener los 16 bits agrupados en la suma.



ENTRADAS			SALIDA ESPERADA		SALIDA OBTENIDA	
A	**B**	**Cin**	**S**	**Cout**	**S**	**Cout**
0	0	0	0	0	0	0
FFFF	FFFFF	0	65534 (16 bits)	1	65534	1
FFFF	FFFF	1	65535 (16 bits)	1	65535	1
1	1	1	3	0	3	0
A	B	1	22	0	22	0
F	3	0	18	0	18	0
D	7	1	21	0	21	0
ABCD	F	0	43996	0	43996	0
ABCD	FF	0	44236	0	44236	0
ABCD	FFF	0	48076	0	48076	0
ABCD	FFFF	0	43980(16 bits)	1	43980	1

El área del *CPA* sería la suma de :

A_CPA = k*(A_CPA) = 4*(136) = 544

Como la implementación del CPA consiste en 4 CPA de 4 bits y tenemos 4 CPA con un area de 136 cada uno (calculado anteriormente) vemos que el área es de 544.

A_CPA = k*(A_FA)= 16*(34) = 544

También se puede hacer como los cálculos hecho anteriormente, con el número de *Full Adders*. En este caso hay 16 con un área de 34 cada uno.

Como se puede ver, el área es la misma.

TIEMPOS TEÓRICOS Y REALES

Para calcular los tiempos teóricos de las 2 salidas *S* y *Cout* se debe tratar igual que un CPA de 4 bits pero adaptándolo este a 16 bits. Por eso, igual que los CPA anteriores el retardo de los *FA* tarda 6T en calcular su salida S, para la salida C sería 7T y para la parte común, la XOR de 3T.

Con esto cogemos las fórmulas usada anteriormente:

S_3 = TC(bit0) + (n-2 bits)*(TC-ParteParalela) + (TS-ParteParalela) = 7T + 14*(7T-3T) + (6T-3T) = 66T

C_4 = TC(bit0) + (n-1 bits)*(TC-ParteParalela) = 7T + 15*(7T-3T) = 67T

Donde TC es el tiempo de Carry, n es número de FA, TS es el tiempo de S y la ParteParalela es la parte que ya se ha realizado al comienzo.

Para saber el camino crítico directamente habría que calcular los retardos de las señales *S_3* y *C_4* ya que son las últimas en ser calculadas porque ambas salidas necesitan de sus etapas anteriores. Entonces el camino crítico sería el de *C_4*.

R_CPA = TC(bit0) + (n-1 bits)*(TC-ParteParalela) = 7T + 15*(7T-3T) = 67T

FASE 3: Sumadores CSA

El Carry Save Adder es un circuito que realiza la suma de dos números de N bits utilizando CPAs y multiplexores. El CSA consta de tres entradas y dos salidas. En las entradas tenemos A, el primer número de N bits, B, el otro número de N bits y Cin que será el acarreo de entrada de la suma. En las salidas tenemos S, que será el resultado de la suma y Cout, el acarreo de salida de la suma.

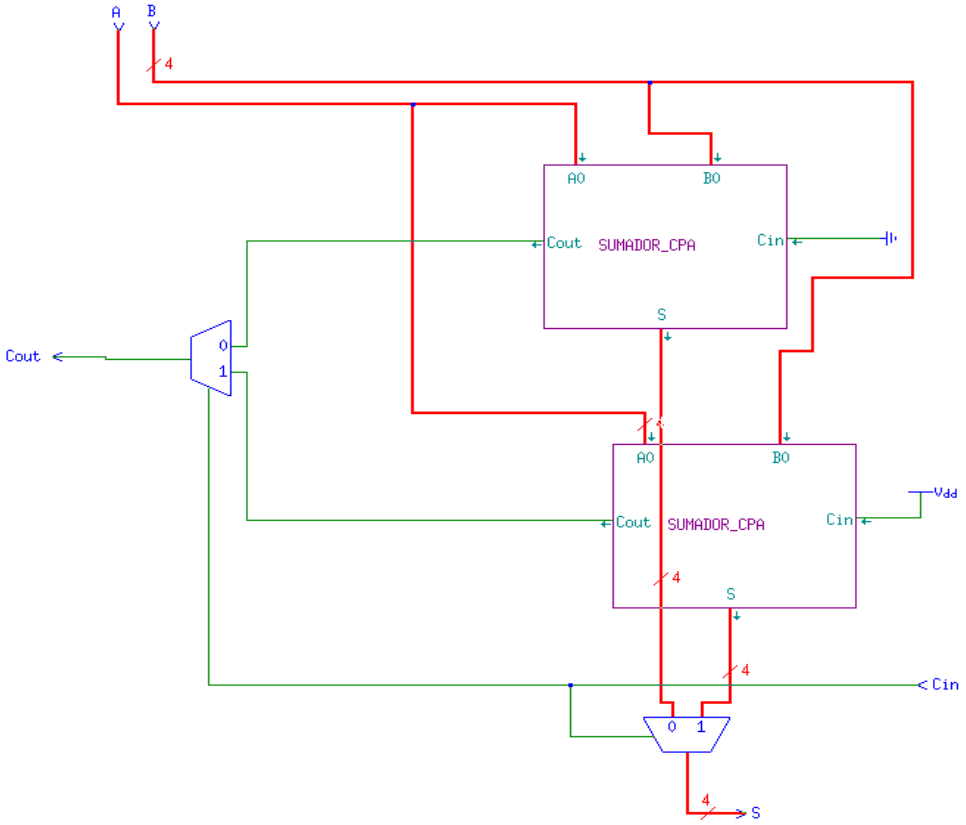
El CSA es un circuito que realiza la suma con acarreo de entrada y sin, por lo que los resultados de la suma y el acarreo de entrada serán seleccionados por los multiplexores que a través del acarreo de entrada que reciben elegirán los resultados correspondientes. Por ejemplo, si el multiplexor recibe acarreo de entrada elegirá los resultados de la suma y de acarreo de salida que hayan sido calculados con acarreo de entrada.

El multiplexor se compone de dos entradas de datos (*A* y *B*), una salida de datos y una entrada de control. Cuando la entrada de control se pone a 0, la señal de datos *A* se conecta a la salida; cuando la entrada de control se pone a 1 lógico, la señal de datos *B* es la que se conecta a la salida.

La diferencia con el CPA es que en el CPA tenemos el problema de tener que esperar al Carry que se va propagando a lo largo de las sumas por todo el circuito, en cambio con el CSA se realiza la suma de los dos números con y sin Carry y una vez se sepa si hay Carry de entrada con el multiplexor se elegirá una de las dos sumas.

TAREA 8: Carry Save Adder (4 bits)

Para el caso de 4 bits usamos la implementación básica de dos CPA que realizarán la suma de los dos números con y sin Carry y dos Multiplexores donde uno elegirá entre los dos resultados de la suma y el otro que decidirá el acarreo de salida.



ENTRADAS			SALIDA ESPERADA		SALIDA OBTENIDA	
A	**B**	**Cin**	**S**	**Cout**	**S**	**Cout**
0	0	0	0	0	0	0
0	F	1	0	1	0	1
F	F	0	14	1	14	1
F	F	1	15	1	15	1
1	1	1	3	0	3	0
A	B	1	6	1	6	1
F	3	0	2	1	2	1

F	A	0	9	1	9	1
D	7	1	5	1	5	1
4	7	0	11	0	11	0
E	9	0	7	1	7	1
D	D	0	10	1	10	1
8	8	0	0	1	0	1
8	8	1	1	1	1	1

El área del *CSA* sería la suma de :

A_CSA = k*(A_CPA) + A_Multiplexores = 2*(136) + 32(multiplexor 4 bits) + 8(multiplexor 1 bit) = 312

Como la implementación del CSA consiste en 2 CPA de 4 bits, donde cada uno tiene un área de 136 y dos multiplexores, uno de 4 bits que tiene área 32 y el otro de 1 bit que tiene área 8, vemos que el área es de 312.

TIEMPOS TEÓRICOS Y REALES

Para calcular el retardo del CSA depende de la implementación del CPA que hayamos utilizado y de los multiplexores que se utilizan. En nuestro caso el CSA es de 4 bits por lo que hemos utilizado 2 CPAs de la Tarea 4 con un retardo de 19T cada uno y dos multiplexores de 1T cada uno.

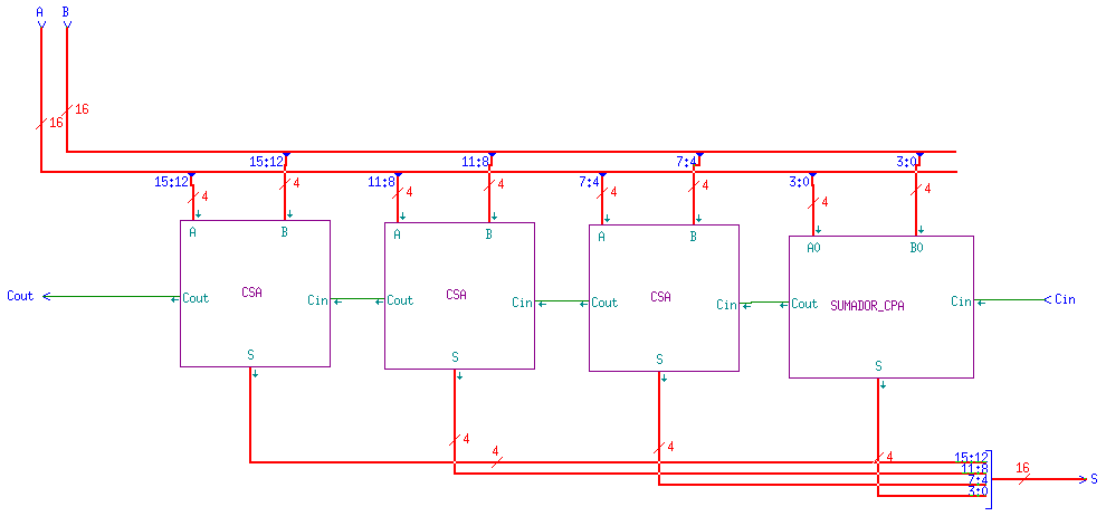
R_CSA = RetardoCPA + Retardomux = 19T + 1*1T = 20T

La suma de ambos CPAs se realizan en el mismo tiempo por lo que solo tendremos que sumar una vez su retardo y a los dos multiplexores les llega el Cin a la vez por lo que también se ejecutan en el mismo tiempo.

El camino crítico será también 20T.

TAREA 9: Carry Save Adder (16 bits)

Para el CSA de 16 bits hemos hecho un circuito formado por 3 CSAs de 4 bits y un CPA de 4 bits. El CPA realizará la suma de los 4 bits de menos peso de las entradas A y B y sumará el Carry de entrada, si es que hay. El Carry de salida que devuelve el CPA será el de entrada del CSA a continuación. Los CSA también realizarán la suma de los bits de A y B e irán propagando sus acarreos de salida para que los multiplexores vayan devolviendo la salida esperada según el Carry de entrada.



El área del *CSA* sería la suma de :

A_CSA = k*(A_CSA4bits) + A_CPA = 3*312 + 136 = 1072

Donde k corresponde a los módulos de CSAs de 4 bits, A_CSA4bits al área del CSA de 4 bits de la Tarea 8 y A_CPA al área del CPA de la Tarea 4.

TIEMPOS TEÓRICOS Y REALES

Para calcular el retardo del CSA depende de la implementación del CPA que hayamos utilizado y de los multiplexores que se utilizan. En este caso el CSA es de 16 bits por lo que hemos utilizado 3 CSA de la Tarea 8, un CPA de la Tarea 4 con un retardo de 19T y dos multiplexores de 1T cada CSA.

R_CSA = RetardoCPA$_0$ + (M módulos-1)*Retardomux = 19T + (4-1)*1T = 22T

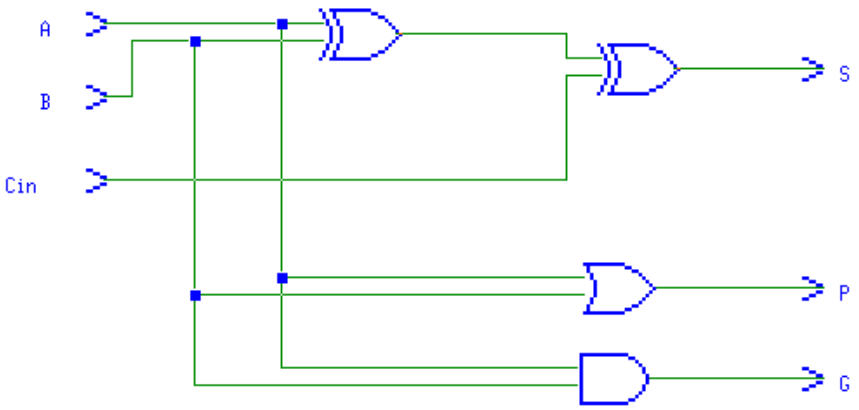
Las sumas de todos los CPAs se realizan en el mismo tiempo por lo que solo tendremos que sumar una vez su retardo y a los dos multiplexores de cada CSA les llega el Cin a la vez por lo que también se ejecutan en el mismo tiempo y solo se tendrá que sumar el retardo de un multiplexor por cada módulo de CSA.
El camino crítico será también 22T.

FASE 4: Sumadores CLA

El Carry Look-Ahead Adder es un circuito que suma dos números introducidos por las señales A y B. La diferencia con el resto de sumadores implementados anteriormente es que para los CLAs se implementa un nuevo circuito para los Full Adders llamado Partial Full Adder. Esta nueva implementación supone un cambio en el cálculo del carry de salida, ya que ahora se calculará a partir de unas nuevas señales P y G que se explicarán en detalle a continuación.

TAREA 10: Partial Full Adder (1bit)

Para explicar el Partial Full Adder debemos introducir primero dos conceptos nuevos. El primero sería el de generar acarreo (G), se genera acarreo cuando al sumar 2 bits ambos valen 1. Por lo que este primer concepto podríamos representarlo lógicamente como **G(Ai, Bi)=Ai · Bi**. El segundo concepto sería el de propagar acarreo (P), se puede propagar acarreo cuando recibimos acarreo de una etapa anterior y uno (o ambos) de los bits que estamos sumando vale 1. Este último concepto representado lógicamente sería **P(Ai,Bi) = Ai + Bi**.
Este circuito es un sumador que tiene 3 entradas A, B y Cin, y 3 salidas S, P y G.
A y B son los dos números de 1 bit que se suman mediante una puerta XOR y posteriormente con otra XOR se suma el Carry de entrada heredado de una etapa anterior. La salida S sería el resultado de estas dos sumas. A diferencia de los anteriores Full Adders es que en este no se devuelve si hay Carry de salida sino que se devuelve P y G, que corresponden a los conceptos explicados anteriormente. Estas dos señales nos serán útiles para posteriormente calcular el Carry de salida de la suma.



JUEGOS DE PRUEBA

ENTRADAS			SALIDA ESPERADA			SALIDA OBTENIDA		
A	**B**	**Cin**	**S**	**P**	**G**	**S**	**P**	**G**
0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0
0	1	0	1	1	0	1	1	0
0	1	1	0	1	0	0	1	0
1	0	0	1	1	0	1	1	0
1	0	1	0	1	0	0	1	0
1	1	0	0	1	1	0	1	1
1	1	1	1	1	1	1	1	1

AREA

El área del PFA sería la suma de todas sus puertas lógicas.

A_PFA = 2*XOR + OR + AND = 2*8 + 6 + 6 = 28

RETARDO

Para el PFA los retardos de cada señal son:

S = 2*XOR = 6T

P = OR =2T

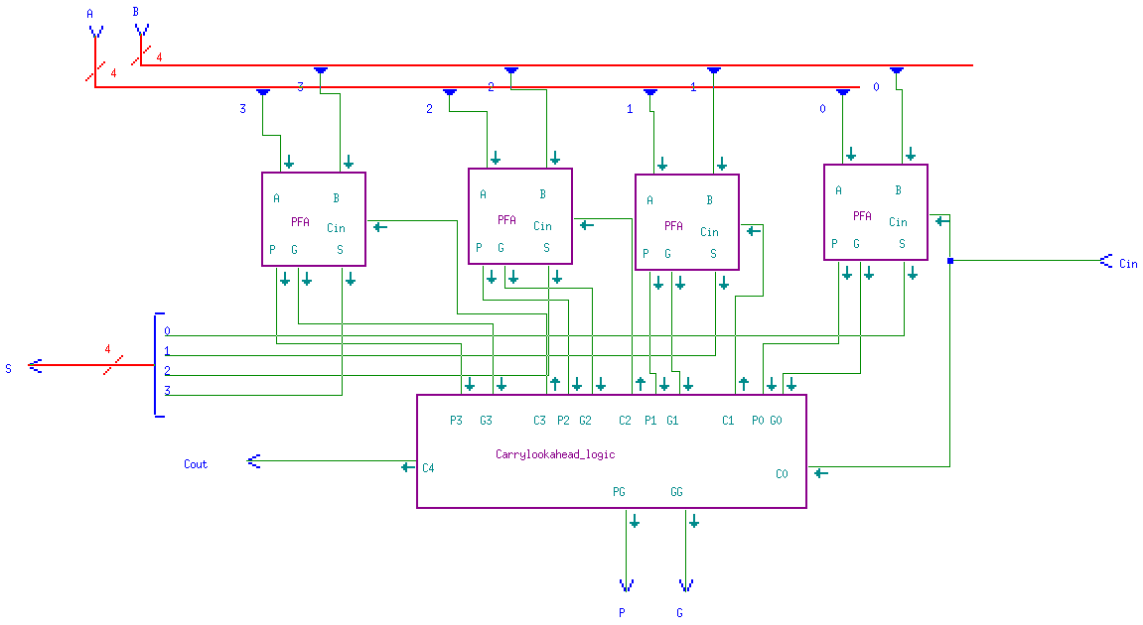
G = AND = 2T

El camino crítico del PFA sería la salida de la señal S.

R_PFA = 6T

TAREA 11: Carry Look-Ahead Adder (4bits)

Para la implementación del CLA de 4 bits usamos la combinación de 4 PFA diseñados en la tarea anterior. Cada uno de estos PFA calculará la suma del bit *i* de A con el bit *i* de B que será de un bit para posteriormente juntar los cuatro por la salida S de cuatro bits, que será el resultado de A + B. También cada PFA calcula su P y G que será llevada al Carry-lookahead logic que es un circuito que calcula el Carry de salida de la suma, las señales P y G del CLA y el carry de entrada de cada PFA menos el del primer PFA que su carry de entrada será el de la entrada Cin.



El Carry lookahead logic como hemos dicho antes se encarga de calcular los carrys de entrada de cada PFA, el carry de salida y las señales P y G del CLA.

Para el cálculo de acarreos de entrada tenemos un cálculo específico para cada uno que corresponden a las siguientes ecuaciones lógicas:

$$C_1 = G_0 + (P_0 \cdot C_0)$$

$$C_2 = G_1 + (P_1 \cdot C_1)$$

$$C_3 = G_2 + (P_2 \cdot C_2)$$

Tendremos Carry si Gi es 1 porque significa que A y B son 1. O tendremos Carry cuando P_{i-1} y C_{i-1} sean 1 porque significa que A o B son 1 y además tenemos carry de entrada.

El problem de estas ecuaciones es que dependen del cálculo del acarreo anterior y generarían un retardo mayor, por lo que para mantener un retardo más bajo estas ecuaciones lógicas se ponen en función de C_0 que es el carry de entrada del CLA el cual nos vendrá dado desde el tiempo 0 y no tendremos que esperar cálculos anteriores.

$$C_1 = G_0 + (P_0 \cdot C_0)$$

$$C_2 = G_1 + P_1 \cdot G_0 + P1 \cdot P_0 \cdot C_0$$

$$C_3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

La ventaja de utilizar estas fórmulas el acarreo de salida depende de A, B y el Carry de entrada ya que tendremos el valor de las tres señales desde el principio y podremos tener un retardo pequeño para todo el circuito.

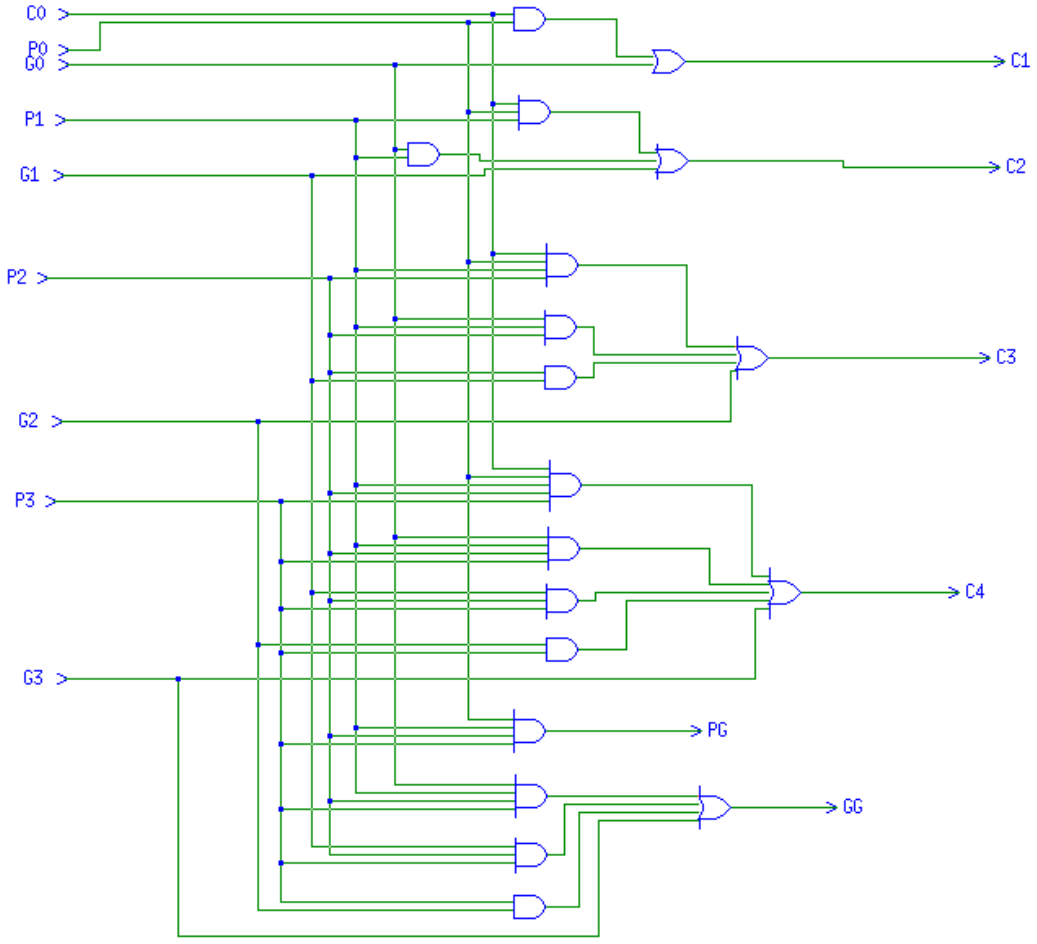
Para el carry de salida del CLA tenemos un fórmula que sigue la misma lógica que la de los acarreos de cada PFA:

$$C_4 = G_3 + (P_3 \cdot C_3) = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_2 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

Y el cálculo de P y G generales serían:

$$PG = P3 \cdot P2 \cdot P1 \cdot P0$$

$$GG = G3 + P3 \cdot P2 \cdot P1 \cdot G0$$


C0
P0
G0
P1
G1
P2
G2
P3
G3
C1
C2
C3
C4
PG
GG

JUEGOS DE PRUEBA

ENTRADAS			SALIDA ESPERADA				SALIDA OBTENIDA			
A	**B**	**Cin**	**S**	**Cout**	**P**	**G**	**S**	**Cout**	**P**	**G**
0	0	0	0	0	0	0	0	0	0	0
0	F	1	0	1	1	0	0	1	1	0
F	F	0	14	1	1	1	14	1	1	1
F	F	1	15	1	1	1	15	1	1	1
1	1	1	3	0	0	0	3	0	0	0
A	B	0	5	1	0	1	5	1	0	1
F	3	0	2	1	1	1	2	1	1	1
D	7	1	5	1	1	1	5	1	1	1
4	7	0	11	0	0	0	11	0	0	0
E	9	0	7	1	1	1	7	1	1	1

AREA

Para el cálculo del área del CLA de 4 bits se tienen que tener en cuenta el área de los cuatro PFA que hay más el área del Carry-lookahead logic.

A_PFA = 28

Para el cálculo del área del Carry-lookahead logic hay que tener en cuenta que cuantas más entradas tenga una puerta más área va a generar así que tenemos los siguientes datos extraídos del tkgate:

PUERTA	AREA 2 ENTRADAS	AREA 3 ENTRADAS	AREA 4 ENTRADAS	AREA 5 ENTRADAS
AND	6	8	10	12
OR	6	8	10	12

Para las and 's tenemos que hay 5 de 2 entradas, 4 de 3 entradas, 4 de 4 entradas y 1 de 5 entradas.

total_ands = (5*6) + (4*8) + (4*10) + (12) = 114

Para las or ’s tenemos que hay 1 de 2 entradas, 1 de 3 entradas, 2 de 4 entradas y 1 de 5 entradas.

total_ors = (6) + (8) + (2*10) + (12) = 46

A_CARRY_LOGIC = total_ands + total_ors = 114 + 46 = 160

A_CLA = 4 *A_PFA + A_CARRY_LOGIC = (4*28) + 160 = 112 + 160 = 272

RETARDO

Para el cálculo del retardo hay que tener en cuenta diversas partes paralelas, así como las suma de A y B y el cálculo de P y G. Como en el Carry-lookahead logic le llegan a la vez todas las P y G, hay que calcular una única vez el retardo de el carry y las sumas.

Según el PFA con los retardos diferentes para cada puerta (AND y OR = 2, XOR = 3), tenemos que para el primer PFA la salida S tiene un retardo de 6T, la salida P y G un retardo de 2T (también para el resto P y G).

S_0 = 2*XOR = 2*3T = 6T

Para la suma, a partir del segundo PFA debe esperar al retardo del carry (6T) para poder realizar la siguiente suma. Por eso, a partir del segundo PFA las salidas S tiene retardo de 9T por el carry de entrada y la XOR (=3T) de la suma.

C_i = calculoPG + AND + OR = 6T

$S_{1,2,3,}$ = llegadaCin + XOR = 6T + 3T = 9T

Por último, hay que calcular el retardo de PG y GG. Cómo PG depende de todas las P 's anteriores (=2T) y una and adicional, el retardo de PG = 4T. GG depende de todas las P ‘s y todas las G ‘s además de una and y una or adicionales, el retardo de GG = 6T.

Como P y G siempre van a ser 2T solo falta añadirle el retardo de las puertas que calculan el carry que son una AND y una OR (2T cada una), dando un retardo de carry de

6T. Al entrar P y G a la vez se puede calcular todos los carrys a la vez dando un carry de 6T a cada uno.

PG = calculoPG + AND = 2T + 2T = 4T

GG = calculoPG + AND + OR = 6T

El camino crítico del CLA sería el de la señal $S_{1,2,3}$ = 9T.

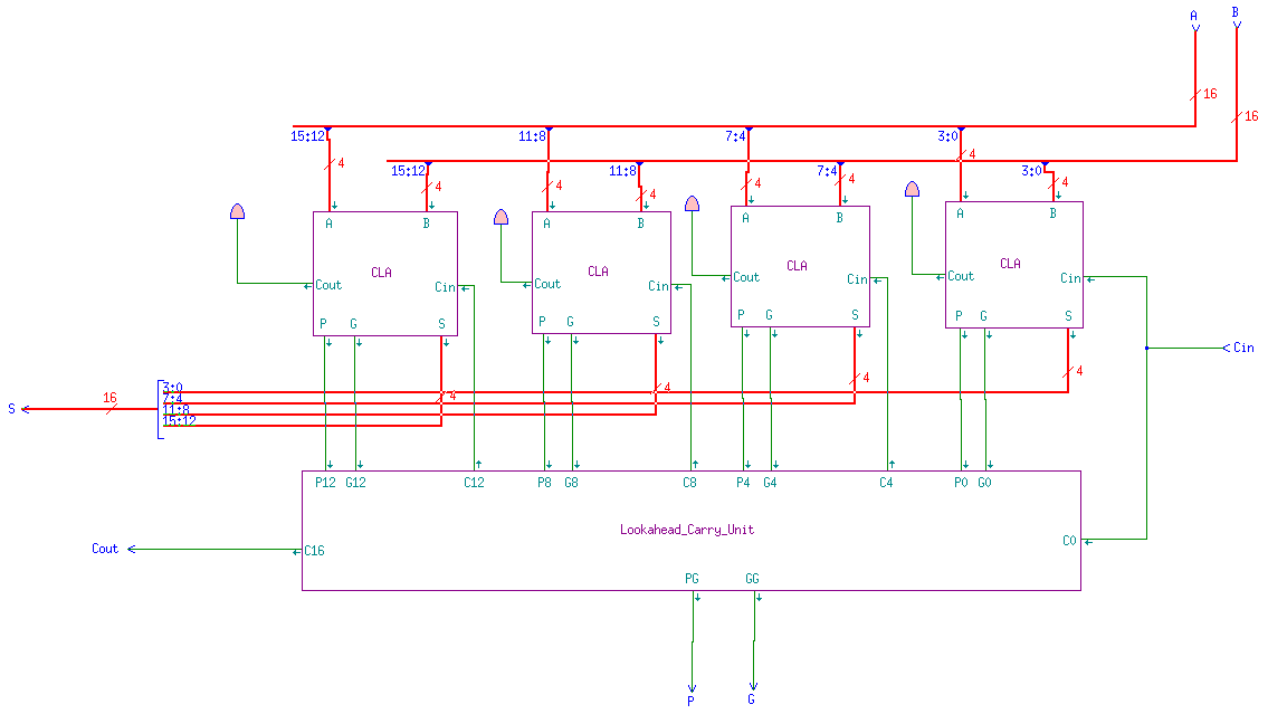
R_CLA = 9T

TAREA 12: Carry Look-Ahead Adder (16bits)

El CLA de 16 bits es una ampliación del circuito anterior extendiendolo para poder hacer el cálculo de una suma de 16 bits. Uniendo 4 módulos CLA de 4 bits junto con un 16-bit Look-ahead carry unit se puede lograr efectivamente la suma de los 16 bits.

A cada CLA de 4 bits le llega un rango de bits de las entradas de 16, A y B, y, también, un carry de entrada que inicialmente viene dado desde fuera pero luego se calcula gracias al 16-bit Lookahead carry unit.

El 16-bit Lookahead carry unit calcula la propagación, la generación de carry y el carry mismo, y funciona igual que Carry look-ahead logic de la tarea 11.



JUEGOS DE PRUEBA

ENTRADAS			SALIDA ESPERADA				SALIDA OBTENIDA			
A	**B**	**Cin**	**S**	**Cout**	**P**	**G**	**S**	**Cout**	**P**	**G**
0	0	0	0	0	0	0	0	0	0	0
FFFF	FFFFF	0	65534 (16 bits)	1	1	1	65534	1	1	1
FFFF	FFFF	1	65535 (16 bits)	1	1	1	65535	1	1	1
1	1	1	3	0	0	0	3	0	0	0
A	B	1	22	0	0	0	22	0	0	0
F	3	0	18	0	0	0	18	0	0	0
D	7	1	21	0	0	0	21	0	0	0
ABCD	F	0	43996	0	0	0	43996	0	0	0
ABCD	FF	0	44236	0	0	0	44236	0	0	0
ABCD	FFF	0	48076	0	0	0	48076	0	0	0
ABCD	FFFF	0	43980(16 bits)	1	1	1	43980	1	1	1
ABCD	ABCD	0	22426	1	0	1	2242	1	0	1

FFFF	0	0	65535	0	1	0	65535	0	1	0

AREA

Para el cálculo del área se tiene que tener en cuenta el área de los cuatro CLA de 4 bits además del área del 16-bit Lookahead carry unit.

De la misma manera que la tarea anterior, cuantas más entradas tenga una puerta más área va a tener esta, dato que se tiene que tener en cuenta para el cálculo del área del 16-bit Lookahead carry unit.

Como hay el mismo número de puertas de cada tipo el área del 16-bit Lookahead carry unit es igual que el área del Carry-lookahead logic.

A_LOOKAHEAD = A_CARRY_LOGIC = 160

A_CLA16 = (4 * A_CLA4) + A_LOOKAHEAD = (4*272) + 160 = 1088 * 160 = **1248**

RETARDOS

El retardo de este CLA de 16 bits se debe calcular para cada uno de los CLA de 4 bits que lo forman, viendo como se van generando los retardos para cada una de las señales:

<u>CLA0:</u>

Para el primer CLA nos encontramos en la misma situación que si estuviésemos trabajando con uno de 4 bits.

S_0 = 2*XOR **= 6T**

$S_{1,2,3}$ = $C_{1,2,3}$ + S_0 = 3T + 6T **= 9T**

PG = 4T, GG = 6T

<u>CLA1:</u>

El cálculo del C4 viene dado la lógica de carry, sabiendo previamente los valores de PG y GG, siendo estos valores los mismos para el resto del circuito ya que se calculan a la vez:

PG = 4T, GG = 6T

C4 = calculoPG + AND + OR = 6T + 2T + 2T **= 10T**

Ahora el carry de entrada para este CLA llegará en 10T (C4), por ello hay que recalcular.

S_4 = C4 + XOR = 10T + 3T **= 13T**

$C_{5,6,7}$ = Cin(=C4) + AND + OR = 10T + 2T + 2T **= 14T**

$S_{5,6,7}$ = $C_{5,6,7}$ + XOR = 14T + 3T **= 17T**

C_8 = GG + AND + OR = 6T + 2T + 2T **= 10T**

CLA2:

Como el carry de entrada de este CLA es el mismo que el anterior (C4 = C8 = 10T) los valores van a ser los mismos.

S_8 = C4 + XOR = 10T + 3T **= 13T**

$C_{9,10,11}$ = Cin(=C4) + AND + OR = 10T + 2T + 2T **= 14T**

$S_{9,10,11}$ = $C_{5,6,7}$ + XOR = 14T + 3T **= 17T**

C_{12} = GG + AND + OR = 6T + 2T + 2T **= 10T**

CLA3:

Como el carry de entrada de este CLA es el mismo que el anterior (C4 = C8 = C12 = 10T) los valores van a ser los mismos.

S_{12} = C4 + XOR = 10T + 3T **= 13T**

$C_{13,14,15}$ = Cin(=C4) + AND + OR = 10T + 2T + 2T **= 14T**

$S_{13,14,15}$ = $C_{5,6,7}$ + XOR = 14T + 3T **= 17T**

C_{16} = GG + AND + OR = 6T + 2T + 2T **= 10T**

En definitiva, el retardo de el CLA de 16 bits corresponde a la señal S:

R_CLA16 = 17T

TAREA 13: Comparación de tiempos de CLA, CPA y CSA.

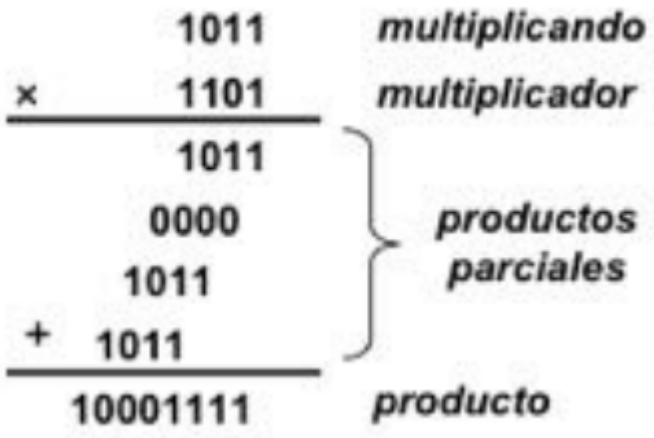
Después de haber realizado los 3 circuitos podemos sacar varias conclusiones sobre cada uno y sobre sus retardos. El CPA para una suma de pocos bits como 4 tiene un retardo aceptable, por ejemplo 19T, pero si se aumentan los bits su retardo se dispara, por ejemplo con una suma de 16 bits su retardo, 67T, es casi 4 veces mayor. Lo cual nos indica que no es el circuito más idóneo para sumas de muchos bits. Para el CSA nos encontramos un escenario diferente, para una suma de pocos bits, 4 por ejemplo, no sería la mejor opción pero a la que los bits a suamr aumentan su retardo crece mucho más lentamente y es mucho más rápido que un CPA. Y finalmente, la mejor opción para sumar números de muchos y pocos bits sería el CLA ya que presenta retardos más pequeños

que el CSA y CPA, tanto en números de 4 bits como de 16. Así que en términos de retardos el CLA sería la mejor opción para realizar un sumador.

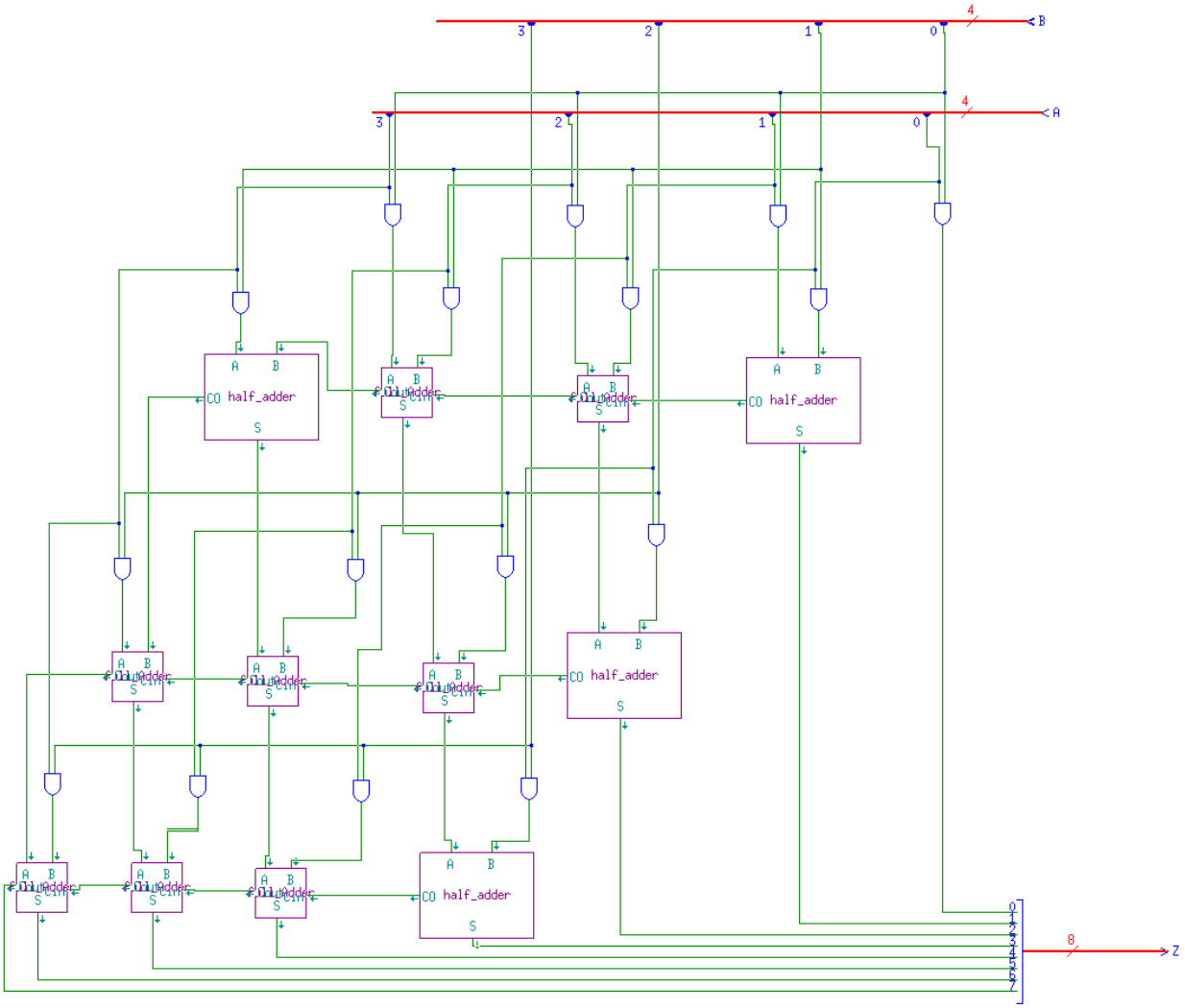
FASE 5: Multiplicador CRA

TAREA 14: Ripple Carry Array (4 bits)

El multiplicador, a grandes rasgos, se basa en un circuito que multiplica dos números de 4 bits A y B, que generan un resultado de 8 bits Z. Es decir tiene dos entradas y una salida.
Para entender la multiplicación de bits se puede dividir en dos partes, la generación de productos parciales y la suma de estos productos parciales.
Para generar estos productos parciales hay que mirar el valor del bit del multiplicador, si este vale 1 el producto parcial será el multiplicando y si fuese 0 se pondrían todos los bits a 0. Para realizar la suma de estos productos obtenidos tenemos que colocarlos uno debajo de otro desplazándose un bit a la izquierda y se realizaría la suma.



Para generar el circuito del RCA se necesitan módulos de suma como FA y HA, para hacer la suma de los productos parciales, y puertas AND que son las que generan los productos parciales.
Las puertas AND son las necesarias para crearlos ya que necesitaremos colocar un 1 en el producto parcial sólo cuando ambos bits sean 1, ya que solo 1*1=1.
Para multiplicar los dos primeros bits (A_0 y B_0) solo necesitamos una AND ya que no se tendrá que sumar con nada el resto de productos se desplazarán a la izquierda.
Para el resto de bits se irán generando los productos parciales y se llevarán a los HA y FA. El uso de HA podría ser sustituido por FA con un carry de entrada a 0, pero para simplificar el circuito hemos utilizado HA cuando no precisamos de un acarreo de entrada.



JUEGOS DE PRUEBA

ENTRADAS		SALIDA ESPERADA	SALIDA OBTENIDA
A	B	Z	Z
0	0	0	0
2	2	4	4
F	F	225	225

A	B	110	110
A	C	120	120
C	F	180	180
A	A	100	100
F	E	210	210
7	E	98	98
8	5	40	40
2	F	30	30
5	5	25	25

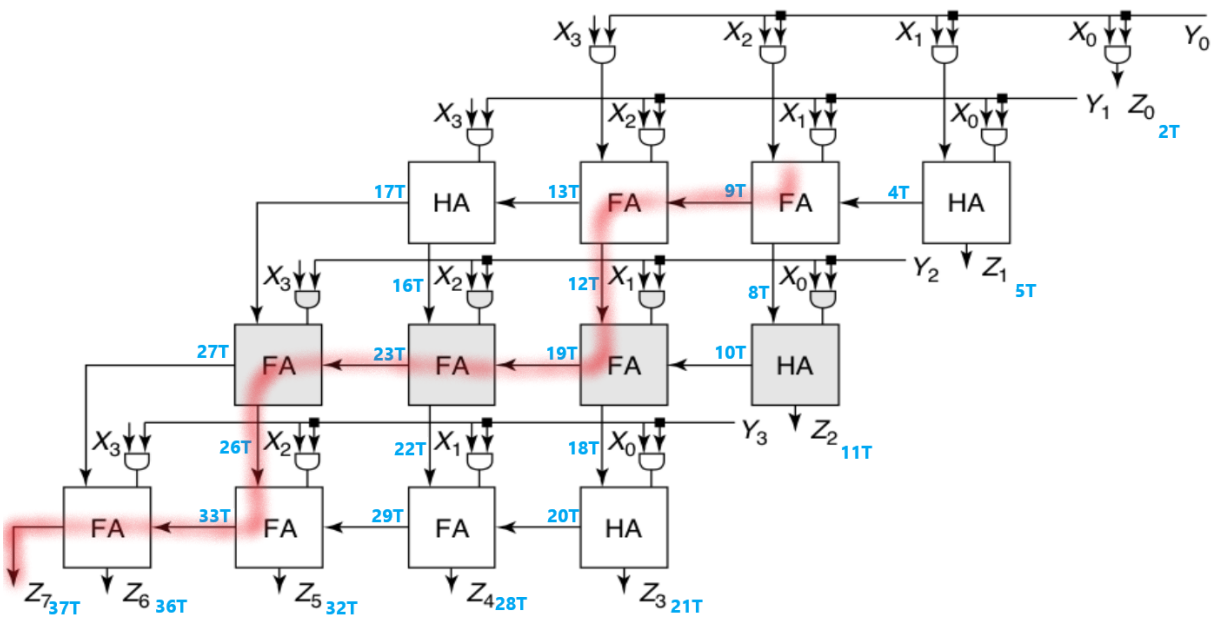
AREA

El área de RCA correspondía a la suma de todas las puertas AND, el área de los HA y la de los FA.

A_RCA = 4*HA + 8*FA + 16*6 = 56 + 272 + 96 = 424

RETARDOS

Los retardos se han ido calculando a lo largo del circuito y para enseñarlo más claro se han puesto en una foto del esquema.



Por eso el camino crítico del circuito es:
R_CRA = 37T