TRABAJO PRÁCTICO ELECTRÓNICA DIGITAL
MULTIPLICADOR BINARIO

Integrantes: Bartolomé Oscar Meritello - 151722477

Arturo Andre Cueva - 151620542

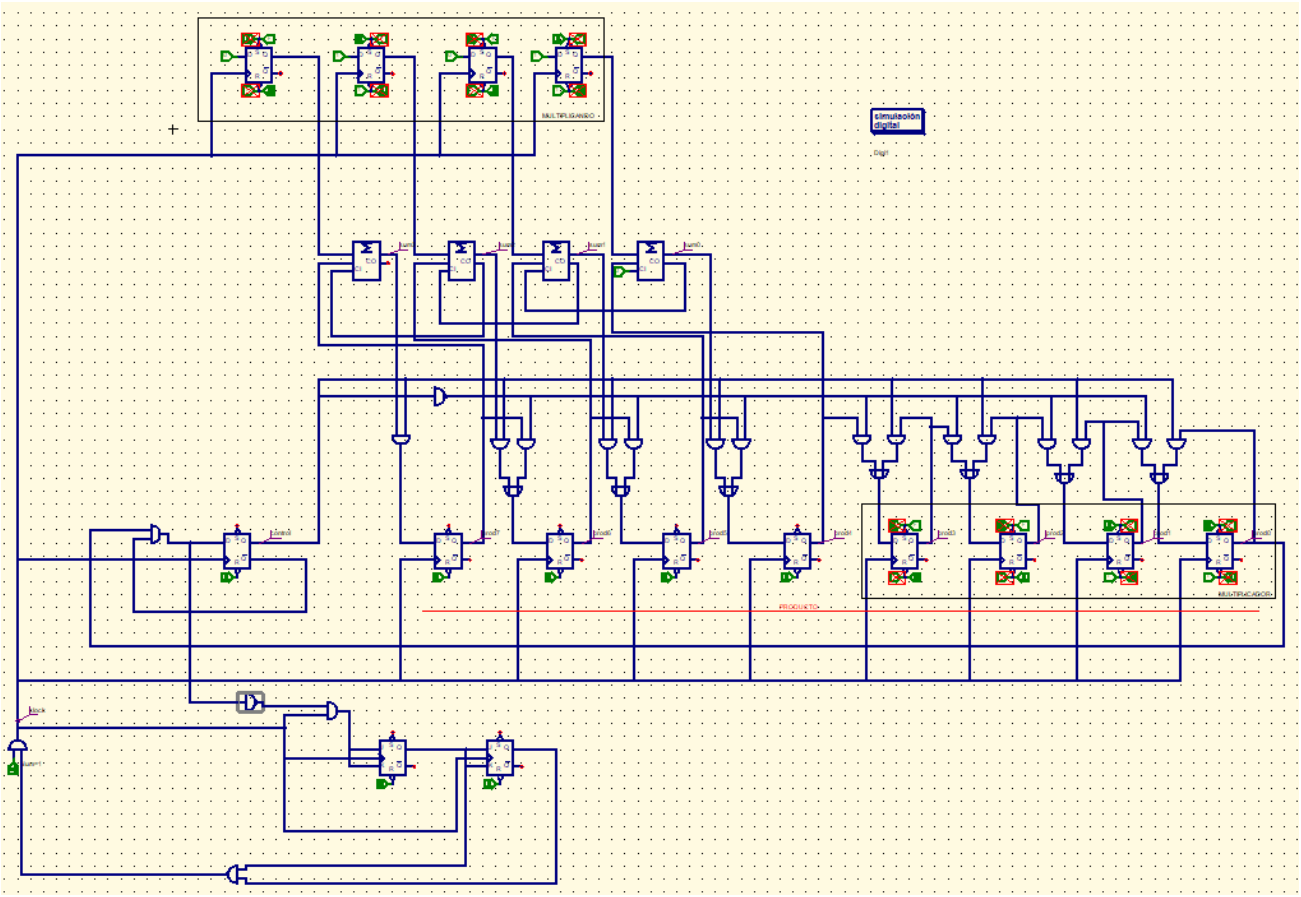
Agustín Claret Vormitag - 151621422

Candela Nostro González – 151620979

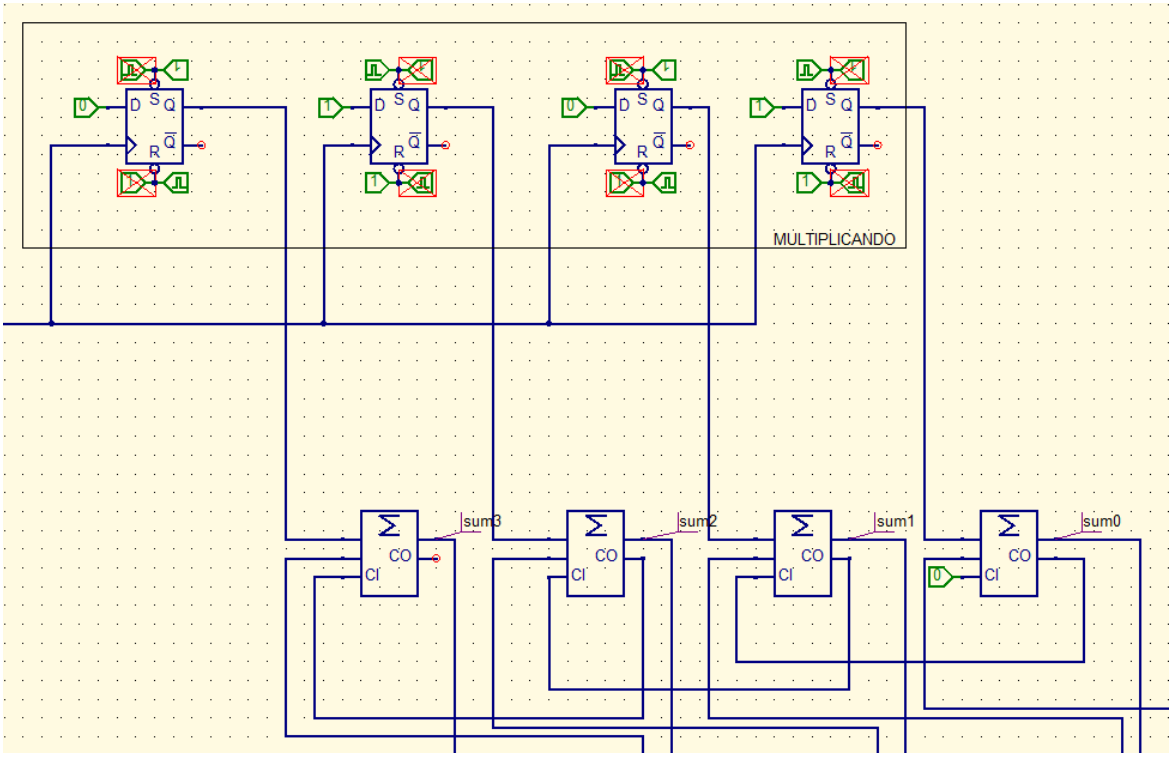
Profesor: José Luis Hamkalo

Fecha: 19/06/2020

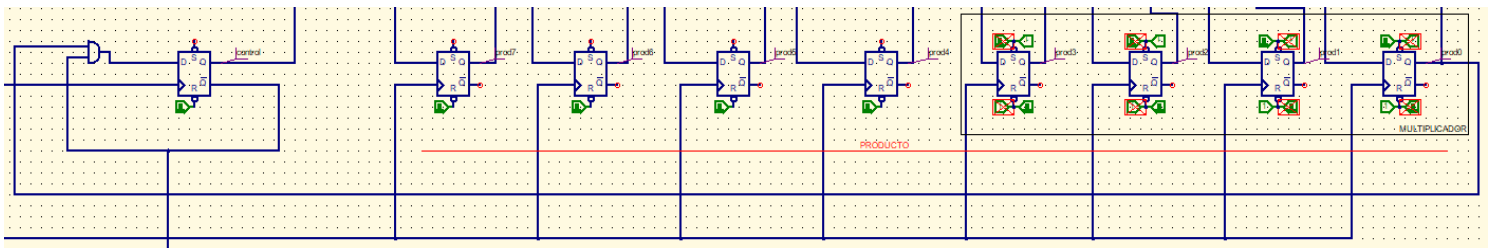
La multiplicación binaria requiere únicamente la suma y el desplazamiento. La secuencia en que estos se lleven a cabo dependerá del valor del bit menos significativo, pero siempre finalizará la operación cuando se hayan realizado 4 desplazamientos (multiplicación de números de 4 bits). A continuación, explicamos la lógica del circuito y la forma en que llevamos a la práctica el algoritmo secuencial visto en clase.



En primer lugar, podemos analizar el sumador, compuesto por los 4 full adder. Observando el full adder de la derecha, notamos que la suma se produce entre el bit menos significativo del multiplicando y el 5to bit (desde la derecha) de lo que será el producto final. Al ser la primera suma, no hay carry interno, por lo que colocamos un 0 que no afectará al cálculo. Sin embargo, la suma del bit siguiente (2do menos significativo del multiplicando) tendrá como carry interno al carry externo actual, debiendo introducirlo en la suma siguiente como se muestra en el circuito. Este proceso se repite en los 3 full adder siguientes, sumándose el carry externo anterior, el bit del multiplicando y el bit correspondiente de lo que será el producto. El último carry externo no tiene efecto en el cálculo.

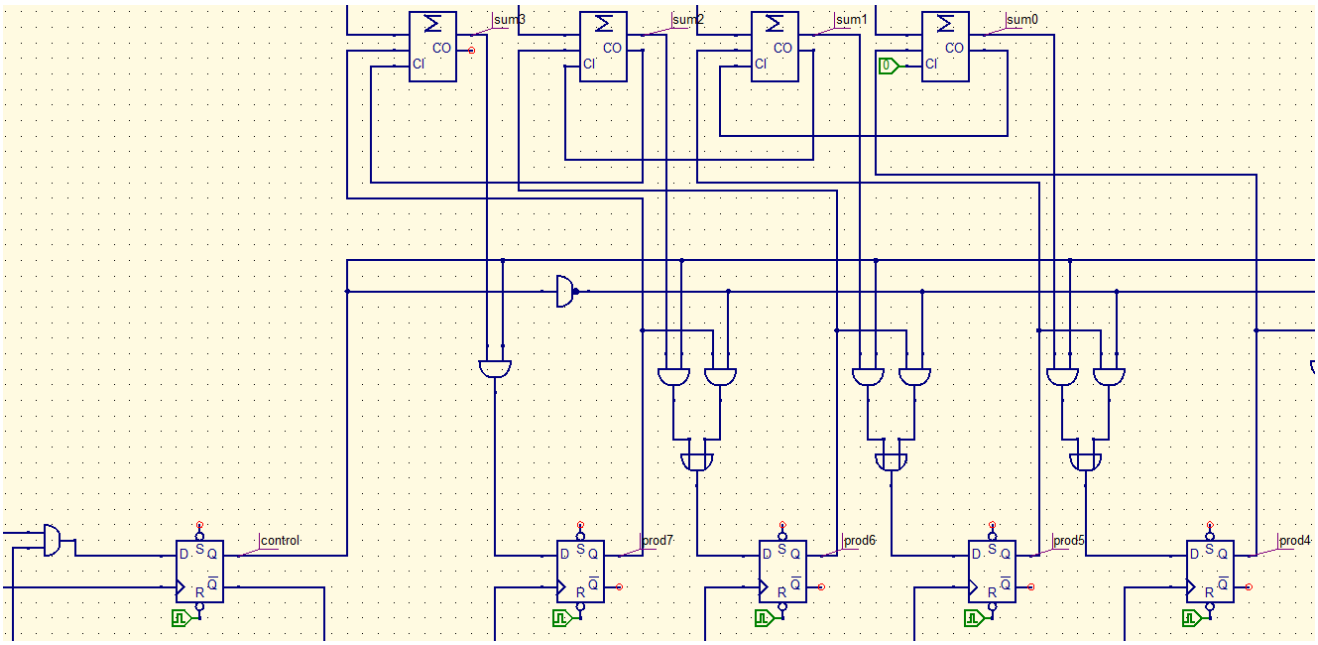


Continuamos analizando el proceso de desplazamiento, llevado a cabo por los 8 flip flops. Como vimos en la teoría, este es regido por el bit menos significativo del multiplicador: si vale 1 se suma y luego se desplaza, y si vale 0 solo se desplaza. Para lograr esto, pasamos el Test a la unidad de control, que determinará la forma de proceder. Situándonos en el flip flop de la derecha, observamos cómo la salida Q se lleva al AND del control, y el resultado de la operación al flip flop del control. La salida de éste se utiliza luego en una serie de operaciones lógicas que dan lugar a la suma o al desplazamiento, según corresponda. Antes de adentrarnos en ellas, es importante tener en cuenta que, si Test vale 1, la salida del control valdrá 1 pero su negado 0. Consecuentemente, en el paso siguiente el AND del control dará 0 como resultado, sea cual sea el valor del bit menos significativo, y se realizará un corrimiento seguido de la suma.

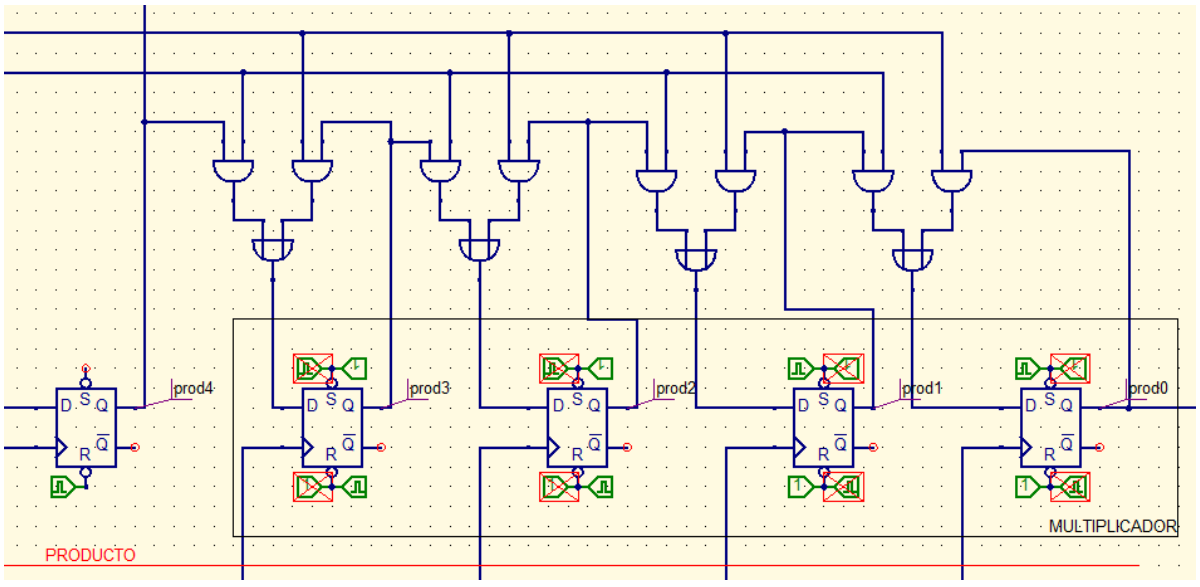


Para entender mejor la lógica de estas operaciones, nos posamos en el conjunto de dos AND y un OR que se encuentra encima de cada flip flop. En un AND entran el valor de salida del control y el resultado de la suma realizada en el full adder correspondiente, por lo que, de tener que realizarse la suma (valor 1 de la salida de control), la salida será

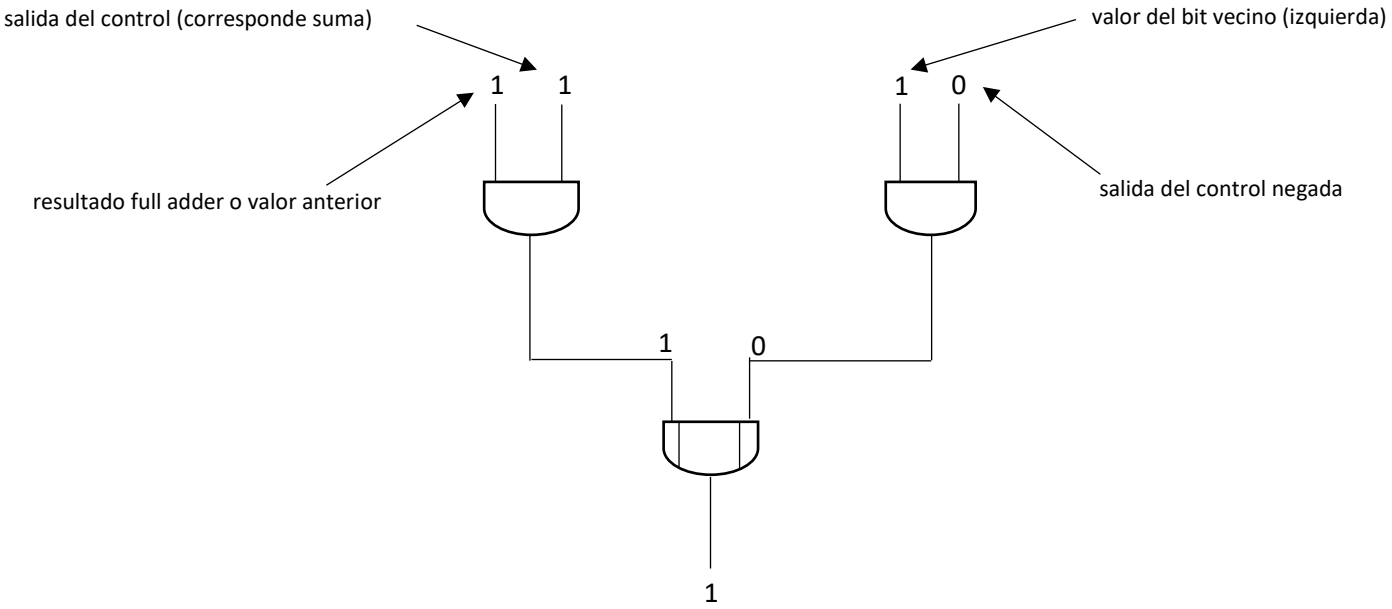
la sumatoria. Caso contrario, la salida será el valor 0 (valor 0 de la salida del control). Respecto al otro AND, éste opera sobre el negado de la salida del control y el bit vecino de la izquierda. Si lo que debe realizarse es la suma, se obtendrá 0 como resultado. Caso contrario (valor 1 del negado del control), prevalecerá el valor del bit vecino. Consecuentemente, el OR que sigue pondrá como entrada del flip flop inferior el resultado de la suma o el valor del bit vecino, según corresponda. Este paso se repetirá sobre cada flip flop, quedando los 8 bits del producto final afectados por la suma o el corrimiento. Al llegar al bit menos significativo (Test), la salida del flip flop se mandará al control nuevamente.



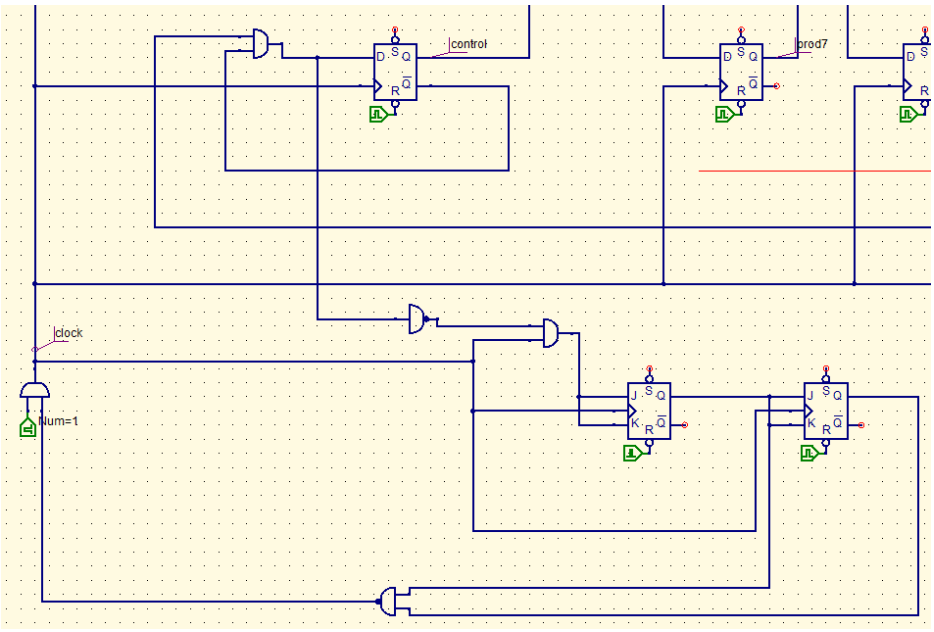
Cabe destacar que en los flip flops correspondientes al multiplicador este procedimiento se replica, sin tomar el resultado de una suma como entrada del AND. En este caso, se toma el valor anterior del bit (memoria del flip flop) para ponerlo nuevamente en ese lugar en caso de tener que realizarse la suma (valor 1 de la salida del control). De lo contrario, se coloca el valor del bit vecino como se hacía en los 4 bits más significativos del producto final.



A continuación, mostramos un ejemplo para clarificar el funcionamiento de las operaciones mencionadas



Por último, queda analizar el funcionamiento del contador, que corta los pulsos del reloj una vez realizados cuatro desplazamientos. Éste es síncrono de 2 bits, compuesto por dos flip flops JK, permitiéndonos representar 4 estados. Siendo estos 00, 01, 10 y 11, cuando ambas Q valgan 1 se debe cortar el reloj. Verificamos esta condición con el operador NAND y llevamos su resultado a un AND con el reloj. Cuando las salidas de ambos flip flops valgan 1, el resultado del NAND será 0 y se cortarán los pulsos del reloj. Por otro lado, es importante notar que la entrada al flip flop de la izquierda es el resultado del AND entre el reloj y la entrada al flip flop de control. Como ya se explicó, esta entrada valdrá 0 cuando Test valga 0 o cuando se haya hecho una suma previamente, indicando en ambos casos que se realiza un desplazamiento.



Los resultados obtenidos los expresamos en las imaganes siguientes. Obtenemos el resultado deseado para multiplicación 3x5 en binarios de 4 bits, pero no conseguimos concluir el por que de la falla del contador. El resultado luego de los 4 shifteos es el correcto pero el programa continua la ejecución hasta el limite de tiempo, es decir no se corta en el momento deseado.

dtime	clock.X	control.X	prod7.X	prod6.X	prod5.X	prod4.X	prod3.X	prod2.X	prod1.X	prod0.X
0	0	0	0	0	0	0	0	0	1	1
1e-09	1	0	0	0	0	0	0	0	1	1
2e-09	0	0	0	0	0	0	0	0	1	1
3e-09	1	0	0	0	0	0	0	0	1	1
4e-09	0	0	0	0	0	0	0	0	1	1
5e-09	1	0	0	0	0	0	0	0	1	1
6e-09	0	0	0	0	0	0	0	0	1	1
7e-09	1	0	0	0	0	0	0	0	1	1
8e-09	0	0	0	0	0	0	0	0	1	1
9e-09	1	0	0	0	0	0	0	0	1	1
1e-08	0	0	0	0	0	0	0	0	1	1
1.1e-08	1	0	0	0	0	0	0	0	1	1
1.2e-08	0	0	0	0	0	0	0	0	1	1
1.3e-08	1	0	0	0	0	0	0	0	1	1
1.4e-08	0	0	0	0	0	0	0	0	1	1
1.5e-08	1	0	0	0	0	0	0	0	1	1
1.6e-08	0	0	0	0	0	0	0	0	1	1
1.7e-08	1	0	0	0	0	0	0	0	1	1
1.8e-08	0	0	0	0	0	0	0	0	1	1
1.9e-08	1	0	0	0	0	0	0	0	1	1
2e-08	0	0	0	0	0	0	0	0	1	1
2.1e-08	1	1	0	0	0	0	0	0	0	1
2.2e-08	0	1	0	0	0	0	0	0	0	1
2.3e-08	1	0	0	1	0	1	0	0	0	1
2.4e-08	0	0	0	1	0	1	0	0	0	1
2.5e-08	1	1	0	0	1	0	1	0	0	0
2.6e-08	0	1	0	0	1	0	1	0	0	0
2.7e-08	1	0	0	1	1	1	1	0	0	0
2.8e-08	0	0	0	1	1	1	1	0	0	0
2.9e-08	1	0	0	0	1	1	1	1	0	0
3e-08	0	0	0	0	1	1	1	1	0	0
3.1e-08	1	0	0	0	0	1	1	1	1	0
3.2e-08	0	0	0	0	0	1	1	1	1	0
3.3e-08	1	0	0	0	0	0	1	1	1	1
3.4e-08	0	0	0	0	0	0	1	1	1	1