Trabajo Práctico

ISI “A”

Arquitectura de computadoras

Primer nivel

Integrantes:

Chort, Julio Alberto

Giancarelli, Francisco

Pacheco Pilan, Federico Ignacio

Reynoso, Valentín

2019

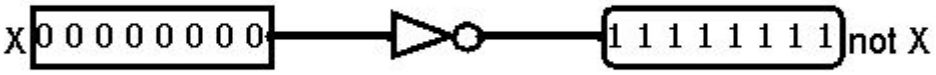
1. Parte 1

1. 1. Ejercicio A:

1.1.1. Complemento a 1:

Véase *CA1* en *ejerA.circ*.

En tanto que lo único que ha de realizarse es el intercambio de los 1 por 0 y viceversa, se recurre a la compuerta NOT.



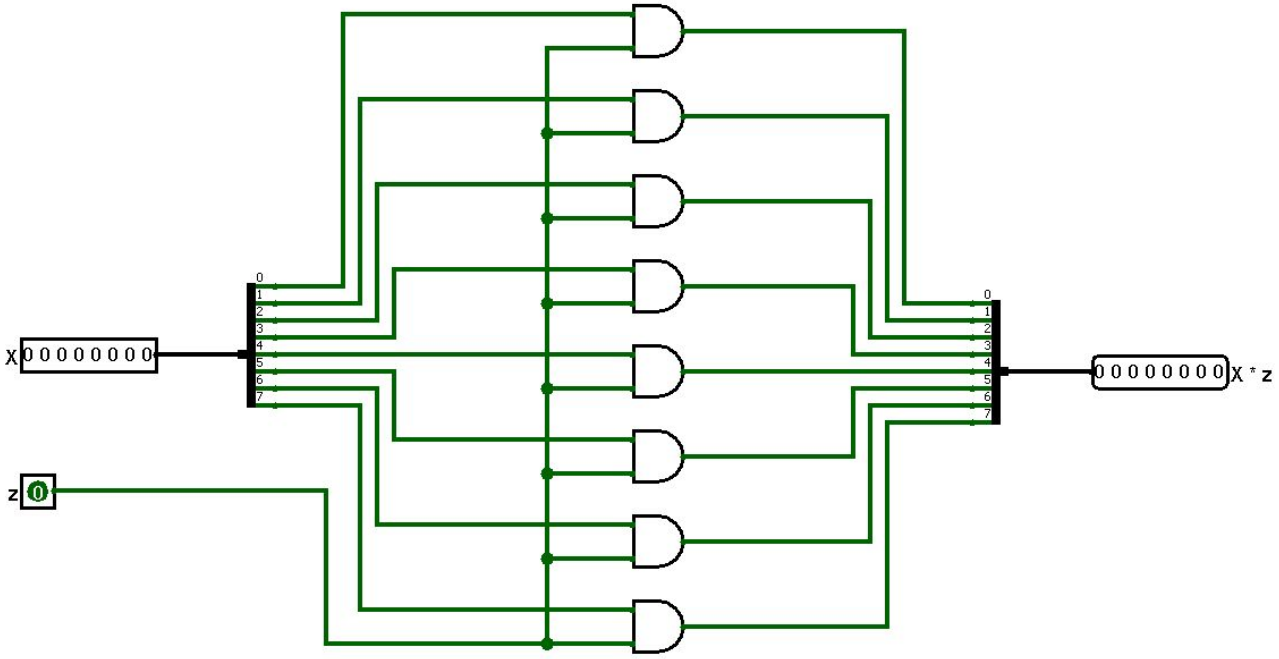
Nótese que sólo hay una de ellas pero que resulta de 8 bits de datos. Esto es equivalente a que se negasen bit a bit las entradas con compuertas "tradicionales", pero de manera mucho más compacta.

1.1.2. Multiplicador de un bit:

Véase *MultiplicadorBit* en *ejerA.circ*.

En la multiplicación binaria sólo existen, como su nombre lo indica, dos estados posibles: o se multiplica por 1, con lo que se obtiene el resultado original (en tanto que el 1 es el neutro del producto); o se multiplica por 0, que coloca un resultado nulo en la salida (obsérvese que el 0 es absorbente). De esta manera, se recurre a compuertas AND que efectúan $x_i \times z$ $(0 \leq i \leq 7)$ para cada bit de datos, simulando lo antes descrito.

Nótese que lo que se encuentra a la derecha de las entradas e izquierda de las salidas es simplemente un "separador" en el simulador *Logisim*: divide / reúne los datos desde / hacia un cable de múltiples bits.



1.1.3. SumTot2:

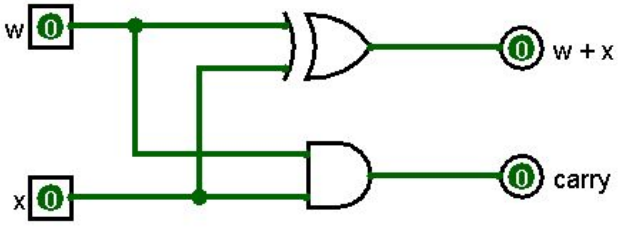
Véase *SumTot2* y *AuxSumTot8* en *ejerA.circ*.

Para la construcción[1] se hizo uso de un circuito auxiliar denominado *Semisumador* en el contexto del .circ. Este obedece a la siguiente tabla de verdad para una suma simple de dos bits:

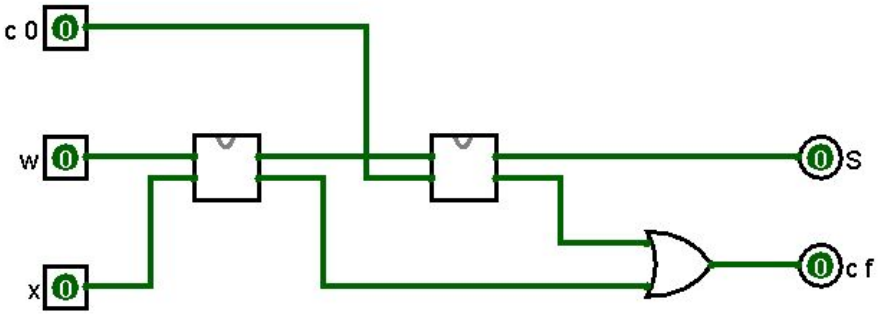
w	x	w + x	carry (acarreo)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Se observa que w + x se corresponde con una compuerta XOR de dos entradas, y el acarreo con una compuerta AND.

[1] Se desarrolló la alternativa etiquetada como *Primer forma de encarar el diseño* del apunte de cátedra *Sistemas lógicos combinacionales*.



Posteriormente se recurrió a lo antes explicitado para efectuar la suma de la columna completa con carry incluido.

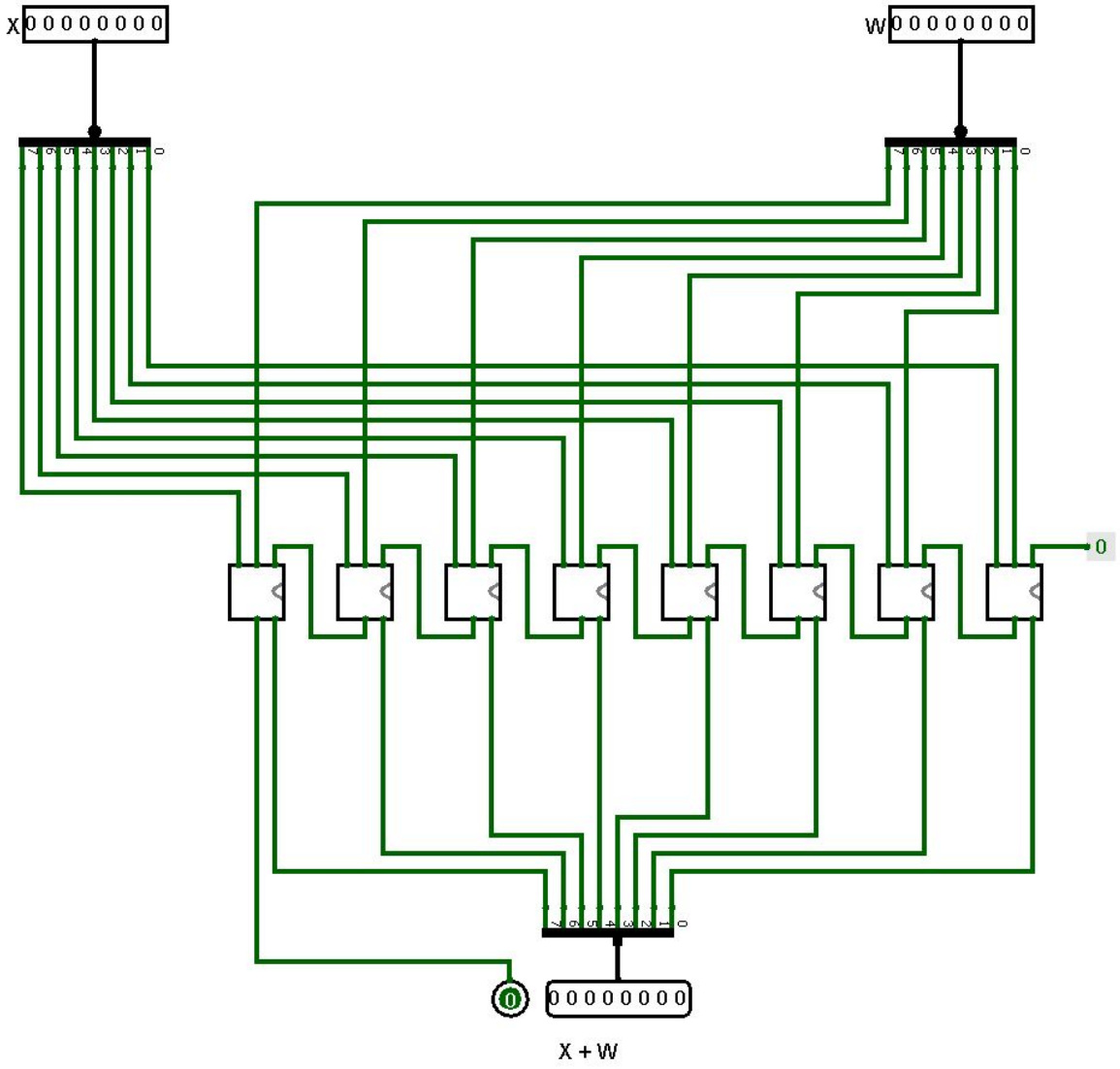


Allí se realiza w + x y luego se le agrega el carry inicial. Por un lado se crea la salida, y por otro el carry final (nótese que habrá acarreo cuando en alguna de las dos sumas se produzca, pero nunca podrá haber dos 1 simultáneamente en la entrada de la compuerta OR).

1.1.4. sumTot8:

Véase *SumTot8* en *ejerA.circ*.

Se recurre a efectuar una concatenación de 8 sumadores totales de 2 dígitos binarios + acarreo. Los acarreos de tales sumas parciales se transfieren al bit más significativo inmediatamente posterior (hasta llegar a $(x + w)_8$, el último bit del resultado de 8 + 1 bits). A la primer columna se conecta una constante nula para que no quede desconectado el "primer acarreo".



1.1.5. Observaciones:

En el .circ se recurrieron a circuitos auxiliares para construir una solución más amigable con el usuario (ver *main*).

1.2. Ejercicio B: multiplicador secuencial:

1.2.1. Conceptos básicos:

Cuando se realiza una multiplicación tradicional de 4 x 4 cifras (con el algoritmo para la multiplicación estándar) se obtiene el producto de cada cifra del multiplicador con cada una de las del multiplicando. Éstas se van disponiendo como sub términos corridos a la izquierda en una posición cada vez para finalmente sumarse y dar el resultado final.

En contraste, para el desarrollo del trabajo se realizó, en cierto modo, "al revés" para facilitar la resolución. Así, en lugar de desarrollar todas las multiplicaciones desplazando a la izquierda los sub términos y sumarlas todas al final, se recurre a realizar las sumas parciales y correr a la derecha el resultado (agregando el último acarreo si lo hubiera).

Considérese el ejemplo de 15_{10} x 15_{10} (1111_2 x 1111_2) = 225_{10} (11100001_2), el peor caso para 4 dígitos binarios:

Operación	**Resultado**								
	C	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Inicialización	-	0	0	0	0	0	0	0	0
Mult. y suma	0	1	1	1	1	-	-	-	-
Reinserción	-	1	1	1	1	0	0	0	0
Desplazamiento	-	0	1	1	1	1	0	0	0
Mult. y suma	1	0	1	1	0	-	-	-	-
Reinserción	-	0	1	1	0	1	0	0	0
Desplazamiento	-	1	0	1	1	0	1	0	0
Mult. y suma	1	1	0	1	0	-	-	-	-
Reinserción	-	1	0	1	0	0	1	0	0
Desplazamiento	-	1	1	0	1	0	0	1	0
Mult. y suma	1	1	1	0	0	-	-	-	-
Reinserción	-	1	1	0	0	0	0	1	0
Desplazamiento	-	1	1	1	0	0	0	0	1

Referencias:

- *Inicialización:* se establece el resultado con los ocho espacios en 0 (como ocurriría en un registro de desplazamiento); el acarreo es indiferente.
- *Mult. y suma:* se multiplica un bit del multiplicador con los 4 del multiplicando como se haría en la multiplicación tradicional -en este caso siempre dando como resultado 1111_2- (se omite para simplificar la tabla), y se suman a b_7, b_6, b_5, b_4.
- *Reinserción:* se coloca lo operado en el resultado principal de ocho espacios.
- *Desplazamiento:* se desplaza a la derecha en una posición el resultado. Se pierde el bit menos significativo. Se coloca el acarreo en la posición más significativa.

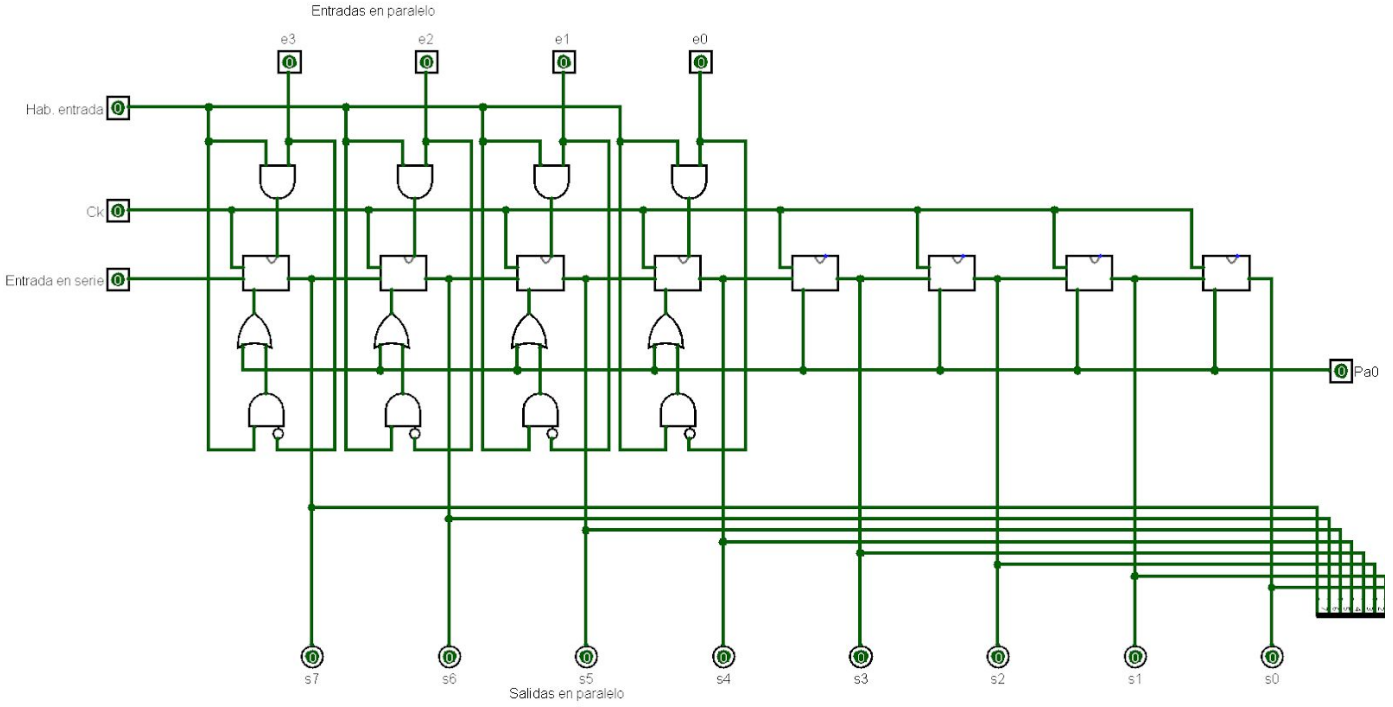
1.2.2. Circuitería asociada:

Para la creación del multiplicador secuencial se recurrió al uso de varios circuitos auxiliares. Primeramente se procederá a explicar brevemente cada uno de ellos.

1.2.2.1. Registro de desplazamiento modificado (registro principal):

Véase *R. de desp. 8 bits modificado (registro principal)* en *ejerB.circ.*

Primeramente se tiene una estructura básica de los circuitos lógicos secuenciales, un shift register, pero con algunas modificaciones a los fines de posibilitar el correcto funcionamiento del circuito principal. Las “cajas” que se observan en medio son biestables tipo D, los cuales presentan de manera reorientada (por limitaciones del simulador), la puesta a 1 (Pa1), puesta a 0 (Pa0), reloj (Ck) y entrada y salida estándar. Al registro de desplazamiento se le insertan en paralelo valores en los primeros 4 bits gracias a una concatenación de compuertas AND, más una entrada de habilitación independiente de Ck. También presenta una puesta a 0 general del circuito y una salida paralela de datos. Nótese que se producirá el desplazamiento de los bits de izquierda a derecha cuando Ck se active.

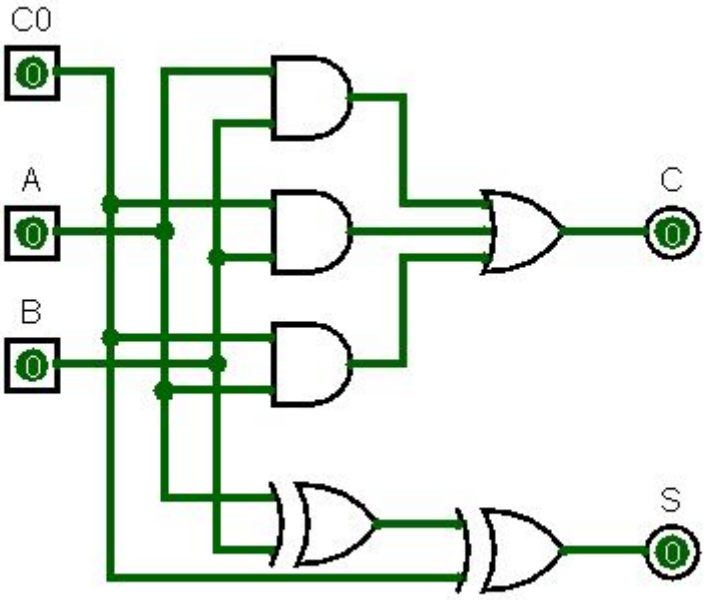


A lo largo del circuito principal también se emplearon registros de desplazamiento propios del simulador *Logisim,* en tanto que eran bloques más compactos y mostraban de mejor manera el desplazamiento de los bits. Sin embargo, su funcionamiento es muy similar en comparación con el que se acaba de mostrar. Difieren en el sentido en que para ingresar entradas en paralelo el clock debe estar activo (empleando una compuerta AND y la habilitación de la entrada se puede lograr esto) y que cada posición del registro presenta entrada paralela.

1.2.2.2. Sumador columna de bits:

Véase *Sum Tot 1 bit* en *ejerB.circ*.

Se realizó un sumador etiquetado en el apunte de cátedra de *Sistemas lógicos combinacionales* como *Segunda forma de encarar el diseño* en el apartado de *Sumadores y restadores*. El funcionamiento es idéntico al explicitado en el primer ejercicio.

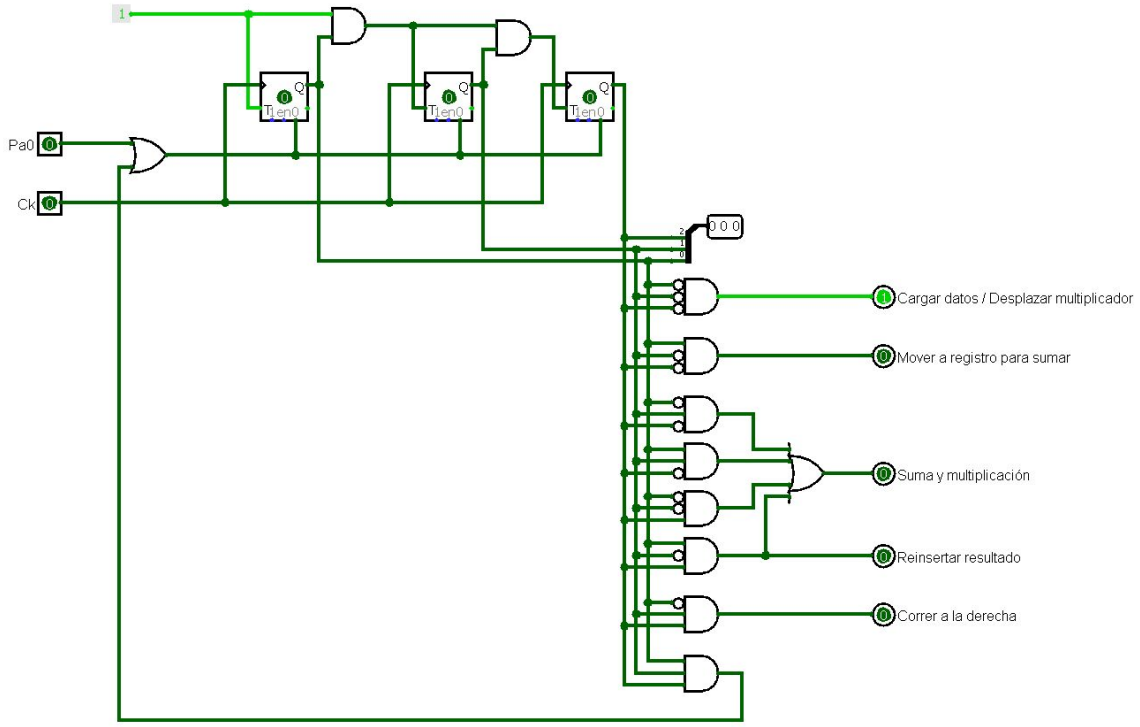


1.2.2.3. Contador principal:

Véase *Contador 0 - 6 modificado (control multiplicación)* en *ejerB.circ.*

Como estructura de control primaria se usó un contador que, a medida que progresa en sus valores, habilita distintas tareas a realizar; a saber:

Momento	Descripción
0	Se cargan el multiplicando y multiplicador (si *Enter* está activo) / Se desplaza a la derecha en un factor de un bit el multiplicador.
1	Se capturan los 4 primeros bits del registro principal (o lo que es lo mismo, los 4 más significativos) y se guardan en un registro de desplazamiento para realizar la suma secuencial.
2, 3, 4, 5	Se efectúa la multiplicación y suma bit a bit (secuencial) del multiplicando y los valores del registro anteriormente nombrado. Los resultados se guardan en otro registro de desplazamiento.
5	Se reinsertan los resultados al registro principal. Se carga además el último acarreo de la suma.
6	Se corre a la derecha una posición el shift register principal.



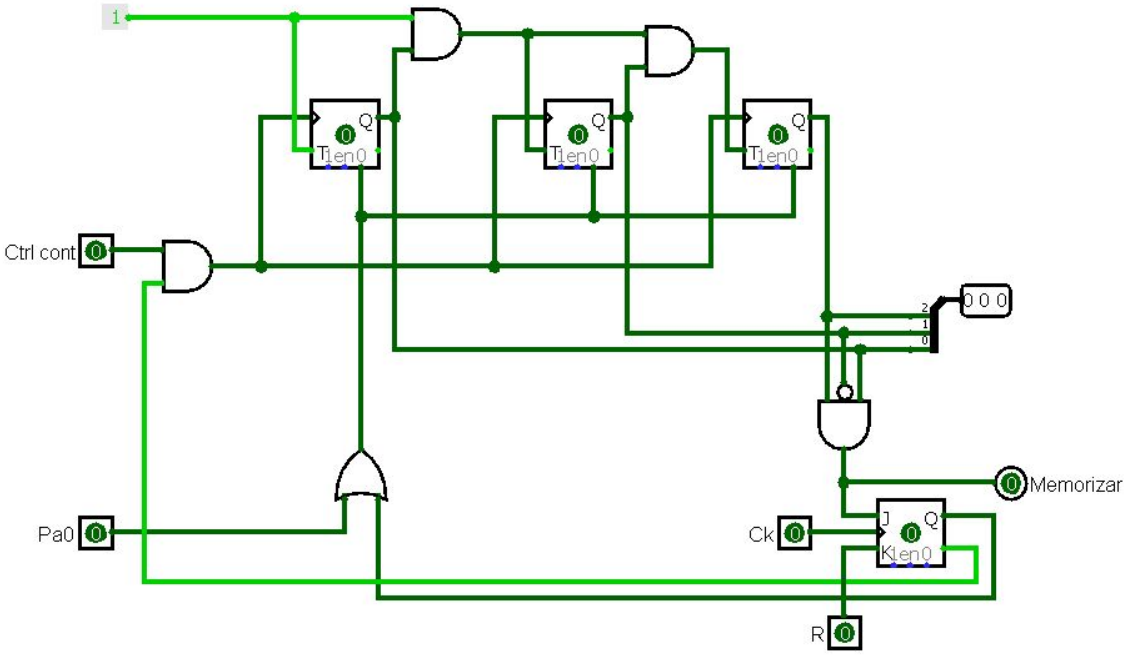
Cabe señalar que el "7" (111) se empleó para efectuar la Pa0, restringiendo el conteo del circuito ligeramente.

1.2.2.4. Contador de memorización:

Véase *Contador 1 - 4 (memorización resultado)* en *ejerB.circ.*

Por último se halla un contador secundario, cuyo único objetivo es el de emitir una señal para capturar el resultado correcto luego de producidos los cuatro ciclos de multiplicación y suma. Dicho valor se guarda en una memoria.

En cuanto a su funcionamiento, este efectúa un conteo de "0" (técnicamente, por cómo fue la implementación, empieza a efectos prácticos desde 1) hasta 4. Al llegar a "5" (101), se habilita brevemente la señal de memorización así como un biestable J K que, conectada su salida negada con un AND junto a Ck, posibilita que el conteo se efectúe una única vez (hasta que, manualmente, se active "R").



1.2.2.5. Circuito principal:

Véase *main* en *ejerB.circ.*

Se produce la conjunción de las partes ya detalladas:

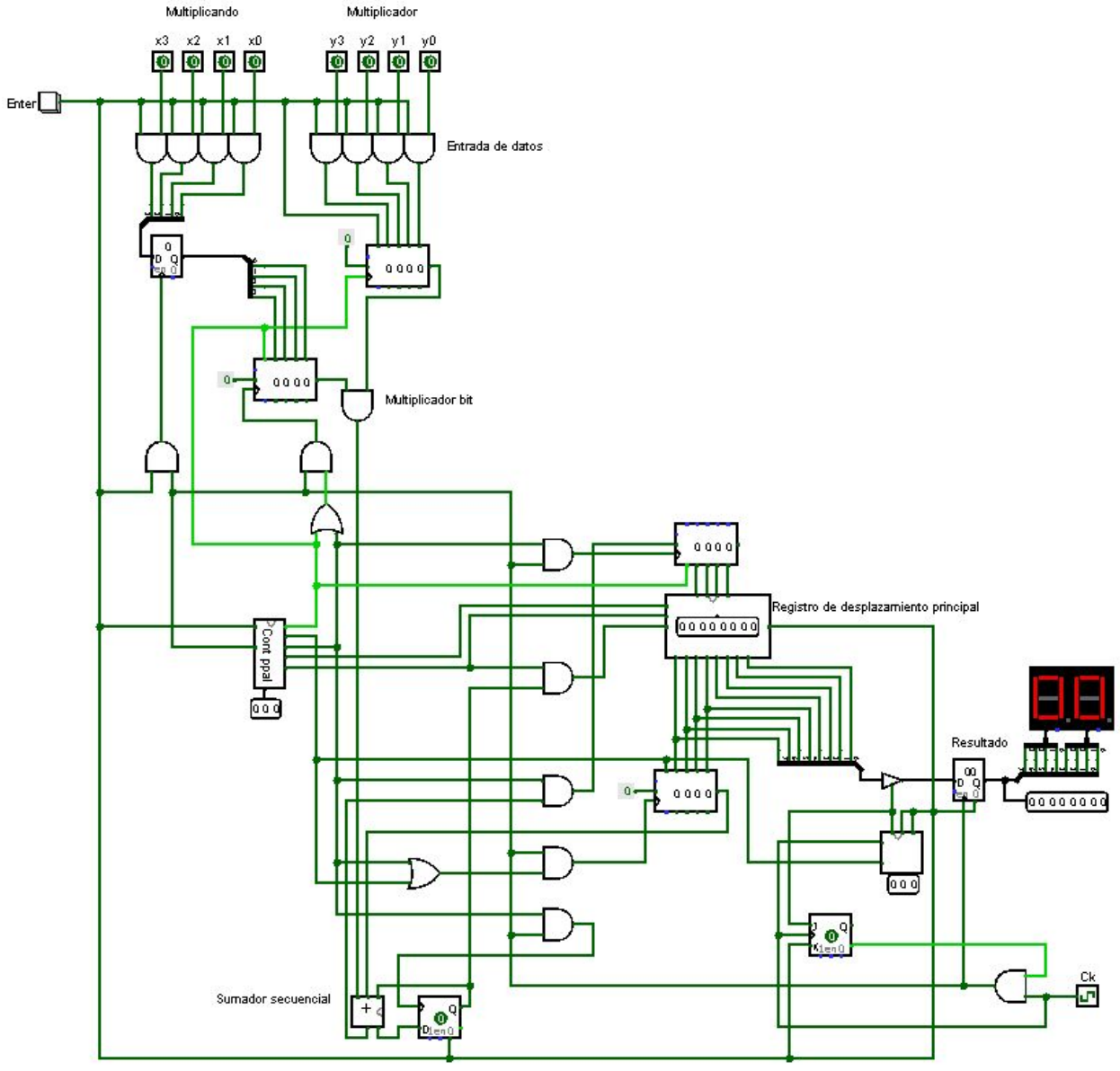
Primeramente (momento 0), luego de ingresados *X* e *Y* y presionado el botón *enter*, se hace la Pa0 general del circuito, se inicia el conteo del "contador de memorización" y se ingresan los valores a multiplicar: uno de ellos se guarda en un registro acumulador para su inserción en cuatro ocasiones y el otro en un registro de desplazamiento.

Seguidamente (momento 1), se toman los valores de los cuatro bits más significativos del registro principal, y se guardan en otro registro de desplazamiento.

Posteriormente se efectúa la multiplicación del i-ésimo bit del registro del multiplicador con los del multiplicando. Este resultado inmediatamente se suma secuencialmente (es decir, una "columna" de bits a la vez; guardando el acarreo en un biestable D) con lo obtenido del segundo registro nombrado. Esto se guarda en otro registro más (momento 2, 3, 4, 5). Cuando se tienen los 4 bits de resultado, se ingresan nuevamente al registro principal (momento 5).

Finalmente (momento 6), se corre a la derecha una posición el registro principal para concluir el ciclo y se ingresa el acarreo de la última suma. Cuando se concluye el cuarto ciclo, se memoriza el resultado en otro registro y se bloquea el Ck.

Cabe aclarar que se recurre al uso de ANDs asociadas al contador principal y una señal de Ck para "filtrar" cuándo debe producirse el desarrollo de las tareas de cada componente, evitando el paso de señales indeseadas en los "momentos" no correspondidos; y ORs cuando más de un "momento" debe activar el funcionamiento de cada parte.



1.2.2.6. Observaciones:

Para usar el circuito construìdo -como ya se ha anticipado- bastará con ingresar los operandos *X* e *Y* y presionar *enter* (se presupone que el reloj -*Ck*- está activado). El resultado

se mostrará en dos 7 segmentos en hexadecimal a la derecha, así como la representación en binario natural más abajo, con mayor o menor celeridad dependiendo de la frecuencia de *Ck*.

Conclusiones

En términos generales, se puede concluir que los circuitos lógicos combinacionales desarrollados a lo largo del trabajo, en tanto funciones de un conjunto de entradas preestablecidas, implicaron una mayor facilidad en su implementación aunque quizá conllevaron a un uso de mayores cantidades de recursos (conexiones, compuertas, etc.).

Por otra parte, el diseño e implementación del circuito secuencial del multiplicador resultó ser de una dificultad mayor, dado la reiterada utilización de múltiples estructuras a lo largo proceso de obtención del producto, como registros de desplazamientos, sumador de 2 + 1 bits, contadores, etc., pero presenta la ventaja de ser *escalable*: ampliando la cantidad de estados en los que el contador principal habilita la suma y multiplicación y el tamaño de los registros de desplazamiento es posible lograr con relativa facilidad una mayor "capacidad operativa" del multiplicador.