

INSTITUTO TECNOLÓGICO DE BUENOS AIRES

22.01 ELECTRÓNICA III

TRABAJO PRÁCTICO 1

Introducción a la Electrónica Digital - Circuitos Combinacionales

Grupo N°3:

Nahuel AGUILAR 54607

Agustin BARRACHINA 53790

Matias DWEK 54181

Julián TACHELLA 53233

Profesores:

Hector Pablo POLENTA

Mathías ANGÉLICO

ENGELHARDT

Jueves 28 de Agosto

Contents

1	Sección I: Lectura y comprensión de hojas de datos (warm-up)	2
1.1	Análisis de Datasheets	2
1.2	Medición de Rise Time, Fall Time y propagación	2
1.3	Compuertas de entrada sin conectar	2
1.4	Valores prohibidos de voltaje en una compuerta	3
1.5	Circuito a probar	3
1.6	Alimentación de los integrados	3
2	Sección II: Ejercicios Karnaugh	3
2.1	Motor con pulsadores	3
2.2	Simplificación por Karnaugh	4
2.3	Circuito lógico para un sistema electrónico de alarma	4
2.4	Minimizar un Karnaugh	4
3	Sección III: Armado y medición de un circuito simple	5
3.1	Sumador de 1 bit	5
3.2	Sumador de 3 bits	6
4	Sección V: Simulación Avanzada	7
5	Sección VI: Más Karnaugh y Riesgos	7
5.1	Riesgos de la Sección II	7
5.2	Otros Ejemplos de Riesgos	7
6	Sección VII: Circuitos Combinacionales Avanzados	8
6.1	Carry Look Ahead	8
6.2	Multiplicador	8
6.3	Comparador	9
7	Anexo	10

1 Sección I: Lectura y comprensión de hojas de datos (warm-up)

1.1 Análisis de Datasheets

Se utilizó las hojas de datos del 74HC(/T)02 Y 74LS02. A partir de estos, se creó una tabla con los márgenes de ruido de los mismos.

Margen de ruido	74HC02	74HCT02	74LS02
High	1.25	1.25	0.7
Low	1.25	1.25	0.3

Figure 1: Los valores corresponden a 25°C con 4.5/5 V de Vcc

Se puede observar en la hoja de datos, que el VIL (Voltage input low) máximo del 74HC02/HCT028 (tecnología CMOS) es igual a 0.8V, mientras que el VOL (Voltage output low) máximo del 74LS02 son 0.5 V. Por lo tanto, el margen de ruido bajo de TTL a CMOS sería 0.3V. Es decir, cualquier ruido que los 0.3V provocaría un error. Por otro lado, el VOH mínimo del 74LS02 es de 2.7 V mientras que el VIH mínimo del 74HC02/HCT02 es de 3.15 V, por lo que una señal “high” del 74LS02 tiene muchísimos riesgos de no ser interpretada por el 74HC02/HCT02. En conclusión, de conectar la salida del 74LS02 a la entrada del 74HC02/HCT02, generaría varios problemas. Sin embargo, se puede conectar la salida de las compuertas CMOS a las compuertas TTL. El Fan-out de una compuerta de output es la cantidad de compuertas de input que puede alimentar, y esta relacionado con la corriente mínima que necesita recibir la compuerta de input para funcionar correctamente. En el caso de la tecnología CMOS, el Fan-out del 74HC02/HCT02 es de 20, y en el caso de TTL, el Fan-out del 74LS02 es de 4.

1.2 Medición de Rise Time, Fall Time y propagación

Se midió el Rise Time y Fall time de una compuerta tipo 74HC02. Las mediciones realizadas se pueden observar en las tablas a continuación:

	Rise Time	Fall Time		Rise Time	Fall Time
1	8.22	7.42		52.05	10.35
2	8.57	7.67		51.45	9.30
3	8.67	7.65		51.15	9.60
4	9.00	7.55		52.25	9.35
5	7.67	7.40		51.25	10.50

(a) Sin carga

(b) Con carga

Figure 2: Mediciones sobre un integrado 74HC02

El tiempo de propagación para la compuerta sin carga a la salida corresponde a $t_{PR} = 12.0ns$ (tiempo de propagación en la subida) y $t_{PF} = 20.0ns$ (tiempo de propagación en la bajada). Sin embargo, al medir el tiempo de propagación de la compuerta cargada se obtuvo $t_{PR} = 12.8ns$ y $t_{PF} = 24.4ns$.

Al cargar la compuerta estudiada con otras cuatro compuertas, los tiempos de Rise time y Fall time se vieron incrementados. Esto se debe a que los transistores que forman el integrado se asemejan a un capacitor en el hecho de que requieren de un lapso de tiempo para alcanzar la tensión con la que se lo alimenta. Al haber más compuertas para ‘cargar’, el tiempo necesario para salir del régimen transitorio aumenta.

1.3 Compuertas de entrada sin conectar

Se alimentó un integrado 74LS02 con 5,0 V y se dejaron sin conectar todas sus entradas. Midiendo con el osciloscopio la tensión en los contactos, las entradas resultaron tener una tensión entre 1,20 y 1,50 V, y las salidas entre 100 y 120 mV. La tensión de entrada está dentro del rango de tensiones no permitidas, y la tensión de salida corresponde a un 0 lógico. Dado a que se está trabajando con una compuerta NOR, las entradas están siendo interpretadas como 1 lógicos. Repitiendo la experiencia con una 74HC02, las entradas resultaron tener una tensión entre -0,2 y 1,0V, y en las salidas se apreciaba una onda periódica e irregular que oscila de 0 a 6 V a 50Hz. Dado a que esta frecuencia es coincidente con la frecuencia de la electricidad doméstica, se deduce que las salidas del integrado HC

son más sensible al ruido del ambiente. Debido a esto, al trabajar con estos integrados será necesario conectar a tierra todas las entradas no utilizadas, y aquellas que correspondan a un 0 lógico. En ninguno de los dos casos el integrado levantó temperatura.

1.4 Valores prohibidos de voltaje en una compuerta

Trabajando con una compuerta NAND 74HC00, se estudió la tensión de salida utilizando valores de entrada prohibidos. La experiencia se realizó de dos formas diferentes. Primero colocando una entrada a 5 V y variando el valor de la segunda entrada; y luego colocando las dos entradas en corto y variando la tensión de ambas. En los dos casos el resultado fue el mismo. En tensiones de entrada menores a 2V, la salida se mantuvo constante en 5V. En tensiones mayores a 2,5 la tensión se mantuvo también constante en 0 V. Variando la tensión de entrada entre 2,0 y 2,5 V, la salida variaba entre valores de 5 a 0 V manteniendo siempre una tensión constante. Se concluye que al trabajar entre 2,0 y 2,5 V, es difícil predecir si la salida de la compuerta será interpretada como un 0 o como un 1 lógico. Por lo tanto es de mayor importancia trabajar fuera de las tensiones no permitidas para garantizar el correcto funcionamiento de las compuertas.

1.5 Circuito a probar

El circuito funcionó correctamente sin levantar temperatura. Nada sucedió y no se sospecha de ninguna razón por la cual el integrado deba calentarse. Si se utiliza una cuadrada de valor medio nulo, con valores entre -2.5V y 2.5V, la compuerta empezó a calentarse y el circuito no funcionó.

1.6 Alimentación de los integrados

Al estudiar la alimentación de las compuertas, se ve que la señal tiene ruido. Al colocar el capacitor en paralelo a la alimentación, actúa como un filtro pasabajos eliminando el ruido y dando una alimentación constante.

También se vio que sin el capacitor, al cambiar el valor de input de las compuertas de un '0' a un '1', se generaba un sobrepico, generando una oscilación en la tensión de entrada. La presencia del capacitor también eliminó estos picos, generando un input más estable.

2 Seccion II: Ejercicios Karnaugh

2.1 Motor con pulsadores

Un motor con 3 pulsadores (A,B y C) y un testigo luminoso los cuales cumplen las siguientes condiciones.

- Si se pulsan los tres pulsadores el motor se activa.
- Si se pulsan dos pulsadores cualesquiera, el motor se activa y se enciende una lámpara adicional como señal de emergencia.
- Si sólo se pulsa un pulsador, el motor no se excita, pero se enciende la lámpara indicadora.
- Si no se pulsa ninguno, no se activa ni el motor ni la lámpara.

Se realizó una tabla de verdad del circuito, obteniendo la siguiente expresión POS reducida: $M = AB + BC + AC$. Luego se implementó en Proteus, obteniendo el circuito lógico de la figura 3a. Además se realizó el circuito del testigo luminoso obteniendo SOP reducida $L = (A' + B' + C')(A + B + C)$, implementandose en la figura 3b.

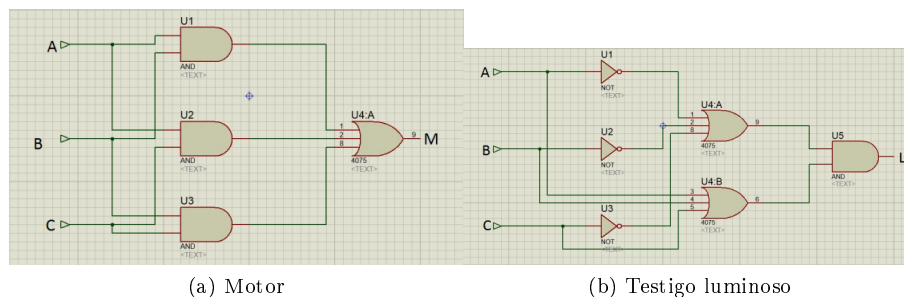


Figure 3: Motor con pulsadores

2.2 Simplificación por Karnaugh

La función $S = c'd + ab'cd' + ab'c'd' + abc'd' + bcd$ se simplificó en el mapa de Karnaugh de la siguiente tabla , obteniéndose la expresión simplificada: $S = bd + c'd + ab'd' + ac'$

	C'D'	C'D	CD	CD'
A'B'	0	1	0	0
A'B	0	1	1	0
AB	1	1	1	0
AB'	1	1	0	1

Table 1: Mapa de Karnaugh

2.3 Circuito lógico para un sistema electrónico de alarma

El circuito necesario puede ser expresado por la función $F = \sum_{ABCD}(1, 7, 11, 13, 14, 15) + d(3, 5, 6, 9, 10, 12)$, que se puede observar en el siguiente mapa de Karnaugh:

	C'D'	C'D	CD	CD'
A'B'	0	1	?	0
A'B	0	?	1	?
AB	?	1	1	1
AB'	0	?	1	?

Table 2: Mapa de Karnaugh

Eligiendo convenientemente los valores don't care(3,5,9,12=1 y 6,10=0), se llega a la siguiente expresión simplificada $F = AB + D$. El circuito lógico resultante se muestra a continuación:

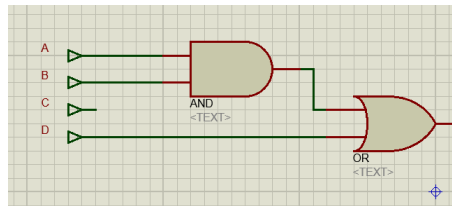


Figure 4: Circuito lógico de la alarma

2.4 Minimizar un Karnaugh

La siguiente expresión $F = \sum_{ABCD}(3, 4, 6, 7)$ se puede expresar en el mapa de Karnaugh de la tabla 3. La expresión de F se puede simplificar a $F = A'BD' + ACD$. Se realizó el circuito correspondiente usando A=0, B=1, C=1 y D variando entre 0 y 1 con una cuadrada. En el circuito apareció un glitch, ya que se estaba obviando un riesgo estático de tipo 1. Para solucionarlo se agregó el término $A'BC$, y prevenir dicho riesgo estático.

	C'D'	C'D	CD	CD'
A'B'	0	0	1	0
A'B	1	0	1	1
AB	0	0	0	0
AB'	0	0	0	0

Table 3: Mapa de Karnaugh

3 Sección III: Armado y medición de un circuito simple

3.1 Sumador de 1 bit

A continuación estudiaremos un circuito sumador de 1 bit con carry in y carry out, representado en la figura 5.

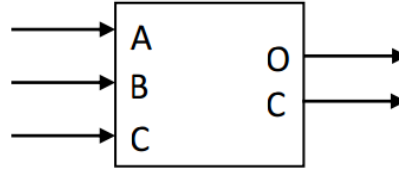


Figure 5: Sumador simple de 1 bit

Para este circuite, se tiene la tabla de verdad representada en el cuadro

A	B	C	O	C
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

Table 4: Tabla de verdad del sumador

Por lo tanto tendremos:

$$\begin{aligned}
 O &= A'B'C + A'BC' + AB'C' + ABC \\
 &= \sum(1, 2, 4, 7) \\
 &= \prod(0, 3, 5, 6)
 \end{aligned}$$

y

$$\begin{aligned}
 C &= A'BC + AB'C + ABC' + ABC \\
 &= \sum(3, 5, 6, 7) \\
 &= \prod(0, 1, 2, 4)
 \end{aligned}$$

Implementando los mapas de Karnaugh llegamos a:

	C'	C		C'	C
A'B'	0	1	A'B'	0	0
A'B	1	0	A'B	0	1
AB	0	1	AB	1	1
AB'	1	0	AB'	0	1
(a) Output			(b) Carry Out		

Figure 6: Mapas de Karnaugh

Para el Output no podemos simplificar ninguna expresión, mientras que para el Carry Out podemos formar tres grupos de dos.

Por ende, obtenemos:

$$\begin{aligned}
 O &= A'B'C + A'BC' + AB'C' + ABC \\
 &= A'(B'C + BC') + A(B'C' + BC) \\
 &= A'(B \oplus C) + A(B \oplus C)' \\
 &= A \oplus (B \oplus C)
 \end{aligned}$$

y para el Carry Out:

$$C = AB + BC + AC$$

Por lo tanto, implementamos estos resultados en el circuito de la figura 7.

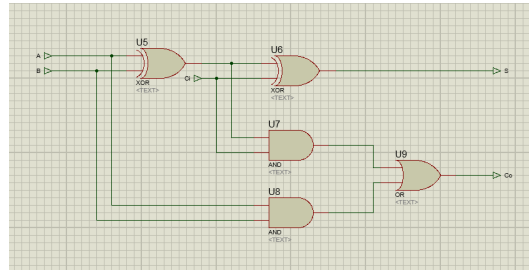


Figure 7: Circuito del sumador de 1 bit

Cabe aclarar que la operación XOR es asociativa y que, con el objetivo de maximizar la escalabilidad del circuito, aprovechamos una compuerta lógica al utilizar la salida de $U5$ como entrada de $U7$, ya que el caso en el que las entradas A y B estén las dos altas, la salida también será alta ($U8$ garantiza esta condición).

Si consideramos un retardo de 11 ns por compuerta, entonces la salida S tendrá un retardo de 22 ns y la salida C_o , 33 ns . Experimentalmente, encontramos $24,9\text{ ns}$ y $35,3\text{ ns}$ respectivamente.

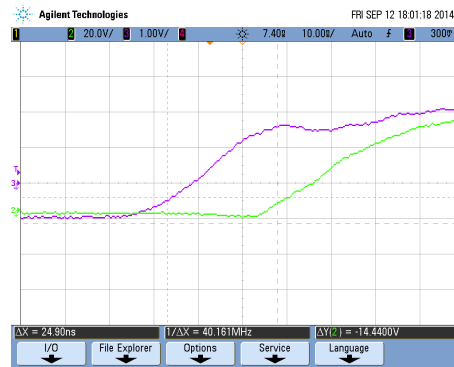


Figure 8: Retardo

3.2 Sumador de 3 bits

Se puede utilizar el sumador simple estudiado en la sección ?? para crear un sumador de 3 bits. Esto se implementa a través de la conexión ilustrada en la figura 9.

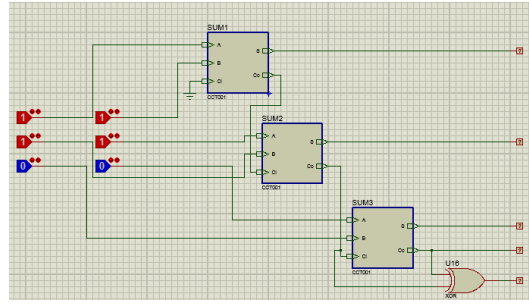


Figure 9: Circuito del sumador de 1 bit

Si deseamos detectar el overflow en una operación, entonces debemos leer la salida de la compuerta XOR entre el Carry In del tercer sumador y el Carry Out de todo el circuito, como se presenta en la figura 9.

4 Sección V: Simulación Avanzada

Se simuló un codificador y un decodificador Hamming 7,4 en Proteus. El código Hamming consiste en agregarle 3 bits a un número de 4 bits haciéndolo un total de 7 bits. De esta forma, se consigue que, si hubo un error en uno solo de los 7 bits, este mismo se corrija automáticamente al decodificarlo. El método consiste en darle un bit de paridad en grupos de 3 bits. De esta forma se consiguen 7 condiciones de paridad, que, según la combinación de ellos se detecta exactamente cuál de los bits tuvo el error. Codificador: Para tener el bit de paridad de los 3 bits en cuestión, simplemente se suman (ignorando el carry) y se coloca el resultado de la suma. Decodificador: Para decodificar el mensaje, se verifica las 3 condiciones que se tienen que dar por bit simultáneamente. Si se cumplen todas, quiere decir que hubo un error, con lo cual se coloca un 1 en el bit que contiene el error y un cero en el resto. Finalmente se utiliza un XOR entre la salida del decodificador y el mensaje que recibió el mismo para invertir el número de ser necesario. Fue realizada una simulación del circuito de Hamming en Proteus, cuya imagen se puede ver en la figura 10 en el anexo.

5 Sección VI: Más Karnaugh y Riesgos

5.1 Riesgos de la Sección II

Para localizar riesgos estáticos de tipo 1 se buscan en el correspondiente mapa de Karnaugh del circuito unos adyacentes de grupos distintos. Por ejemplo, en los circuitos de la sección II, el único que posee riesgos es el ejercicio 4. Si creamos grupos con los unos que cumplen con la condición de riesgo, encontramos la siguiente función de salida:

$$F = A'CD + A'BD' + A'BC$$

5.2 Otros Ejemplos de Riesgos

Con el objetivo de profundizar el estudio de los riesgos proponemos el estudio del siguiente sistema:

$$S = \sum_{ABCD} (1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$$

El mapa de Karnaugh de ese circuito es el presente en la tabla 5.

	C'D'	C'D	CD	CD'
A'B'	0	1	1	0
A'B	1	1	0	0
AB	1	1	1	1
AB'	0	0	1	1

Table 5: Mapa de Karnaugh

Si no tenemos en cuenta los riegos, encontramos la siguiente expresión:

$$S = BC' + AC + A'B'D$$

Por otro lado, si buscamos eliminar los riesgos, tenemos que crear nuevos grupos donde se encuentran unos adyacentes a otros unos de otro grupo.

En este caso, al aplicar este procedimiento, llegamos a:

$$S = A'B'D + B'CD + A'C'D + BC' + AB + AC$$

De esta manera, agregando tres compuertas AND (una de dos entradas y otra de tres), eliminamos los riesgos anteriormente presentes. Cabe remarcar que en este tipo de situaciones se pone en juego una relación costo beneficio, ya que implementar más compuertas hace el circuito más robusto, pero al costo de más integrados.

6 Sección VII: Circuitos Combinacionales Avanzados

6.1 Carry Look Ahead

Se diseñó y se armó un sumador Carry Look Ahead. En primera instancia se estudia la suma de todos los pares de bits, con intención de predecir el próximo carry. Para esto, se utilizan dos salidas denominadas “propagate” y “generate”. Estos valores son enviados al Carry Look Ahead. En el caso de que ambos bits a sumar sean 1, ya se sabe que independientemente del carry in, la suma generará un carry al próximo bit. Por lo tanto, se genera una salida “generate”, y el Carry Look Ahead sin más espera genera el carry al próximo bit. En el caso de que los bits a sumar sean un cero y un uno, se sabe que habrá carry, solo si el bit anterior también lo genera. Se puede pensar como la propagación del carry del bit anterior al siguiente. Por ende, se genera una señal “generate”, y el Look Ahead genera una señal de carry al próximo bit solo si recibe carry del bit anterior.

Referencias: A_n : enésimo bit del input A, B_n : enésimo bit del input B, C_n : carry in del enésimo bit, P_n : propagate del enésimo bit, G_n : generate del enésimo bit, O_n : Output del enésimo bit.

$$O_n = (A_n \oplus B_n) \oplus C_n$$

$$G_n = A_n \cdot B_n$$

$$P_n = A_n \oplus B_n$$

$$C_{(n+1)} = G_n + P_n \cdot C_n$$

Al tener un Carry in y un Carry out, el circuito es altamente escalable. Se pueden usar varias plaquetas del circuito para sumar números de más de tres bits sin aumentar significativamente el tiempo de respuesta.

El diseño final se presenta en la figura 11 en el anexo.

6.2 Multiplicador

Estudiaremos como diseñar un multiplicador paralelo que recibe dos números de 3 bits y devuelve el resultado en un número de 6 bits. Este circuito será capaz de trabajar tanto con números signados como con números no signados.

Para diseñar esta red planteamos la siguiente multiplicación (en bits no signados):

$$\begin{array}{r}
 a_2 a_1 a_0 \\
 \times b_2 b_1 b_0 \\
 \hline
 00 a_2 b_0 a_1 b_0 a_0 b_0 \\
 0 a_2 b_1 a_1 b_1 a_1 b_1 0 \\
 + a_2 b_2 a_1 b_2 a_1 b_2 00
 \end{array} \tag{1}$$

El resultado de esta operación resulta en la multiplicación de dos números de 3 bits.

Por lo tanto, nos es claro que para la implementación de nuestro circuito necesitaremos al menos dos sumadores y nueve compuertas AND.

Con respecto a la multiplicación signada, tenemos la siguiente operación:

$$\begin{array}{r}
a_2 a_1 a_0 \\
\times b_2 b_1 b_0 \\
\hline
a_2 b_0 \ a_2 b_0 \ a_2 b_0 \ a_1 b_0 \ a_0 b_0 \\
a_2 b_1 \ a_2 b_1 \ a_1 b_1 \ a_1 b_1 \ 0 \\
- \ a_2 b_2 \ a_1 b_2 \ a_1 b_2 \ 0 \ 0
\end{array} \tag{2}$$

Es decir que tenemos que completar los bits de la izquierda de los primeros dos sumandos con el mismo dígito que el bit más significativo. Osea que se completa con unos si el número es negativo. El último termino se resta a la suma de los otros dos. Esto se logra con la suma del complemento a uno (en el circuito implementado con compuertas XOR). Cabe destacar que el complemento a uno también puede ser realizado con un tercer sumador, pero con el objetivo de minimizar la cantidad de integrados se utilizan compuertas XOR, que resultan en una cantidad menor de compuertas. Esta desición en el diseño tiene como consecuencia menores retardos y por lo tanto se minimiza la posibilidad de un glitch.

Presentamos el diseño final en la figura 12 en el anexo.

Con respecto a lo escalable, al circuito no se le pueden agregar más entradas en cascada ya que el resultado final depende de las sumas 1 y 2, lo que hace que se tenga que rediseñar la red si se quisiesen sumar más bits. Otra característica a tener en cuenta cuando se considera la escalabilidad del circuito es la dependencia del número de salida con respecto al de entradas. Si se suman 2 números de n bits, entonces la salida será de $2 \cdot n$ bits, lo que hace que escalar el circuito se dificulte.

Para representar la salida se utilizaron LEDs y un JUMPER para activar la operación con signados.

Para calcular o estimar el retardo del circuito buscamos en los Datasheet de los fabricantes los tiempos estimados para cada compuerta o integrado. Buscamos el camino más largo que se presenta desde la entrada hasta la salida y nos encontramos con 3 compuertas ANDs HC08 y dos sumadores 74HC283. Para las primeras, se presentan retardos de 11 ns . Por otro lado, el retardo de los sumadores depende de si el carry propaga de un bit a otro. Por ejemplo, si se presenta C_{in} a \sum_2 , tendremos un retardo de 18 ns . En el peor caso, C_{in} a \sum_4 , ocurrirá un retardo de 23 ns . Por ende, tomando en cuenta todos los retardos en el peor caso, encontramos el total igual a 79 ns .

6.3 Comparador

El comparador compara dos números no signados de 3 bits cada uno y te devuelve cuál es mayor a cual. Para los bits de entrada se usaron dip-switchs conectados a 5V con un pull down a la salida. Y se usaron 3 LEDs de distintos colores para cada resultado ($A = B$, $A < B$, $A > B$). Para evitar el cruce de pistas, se debe ingresar los números dos veces. El circuito puede dividirse a grandes rasgos, en dos circuitos más pequeños. Uno de ellos compara bit a bit para encender el LED correspondiente si se cumple esas condiciones. El otro circuito verifica si $A > B$. Por último, si no se cumplen ninguna de las condiciones anteriores, quiere decir que $A < B$, con lo que se invierten los resultados anteriores y su salida andeada se utiliza para el último LED. Se buscó hacer todo el circuito con integrados CMOS para evitar problemas de compatibilidad con las TTL, como por ejemplo el margen de ruido. No se buscó eliminar glitches, ya que los mismos no afectan el resultado del mismo. Se puede usar este circuito de forma escalable para comparar números de n bits. Para ello, se le da prioridad a la salida de los 3 bits más significativos. Si son iguales, se verifican los siguientes 3 bits, y así hasta encontrar una condición de mayor o menor. Si en todos esos casos, el resultado fue que los bits son iguales, entonces ambos números de n bits son iguales.

El diseño final del comparador se presenta en la figura 13 en el anexo.

7 Anexo

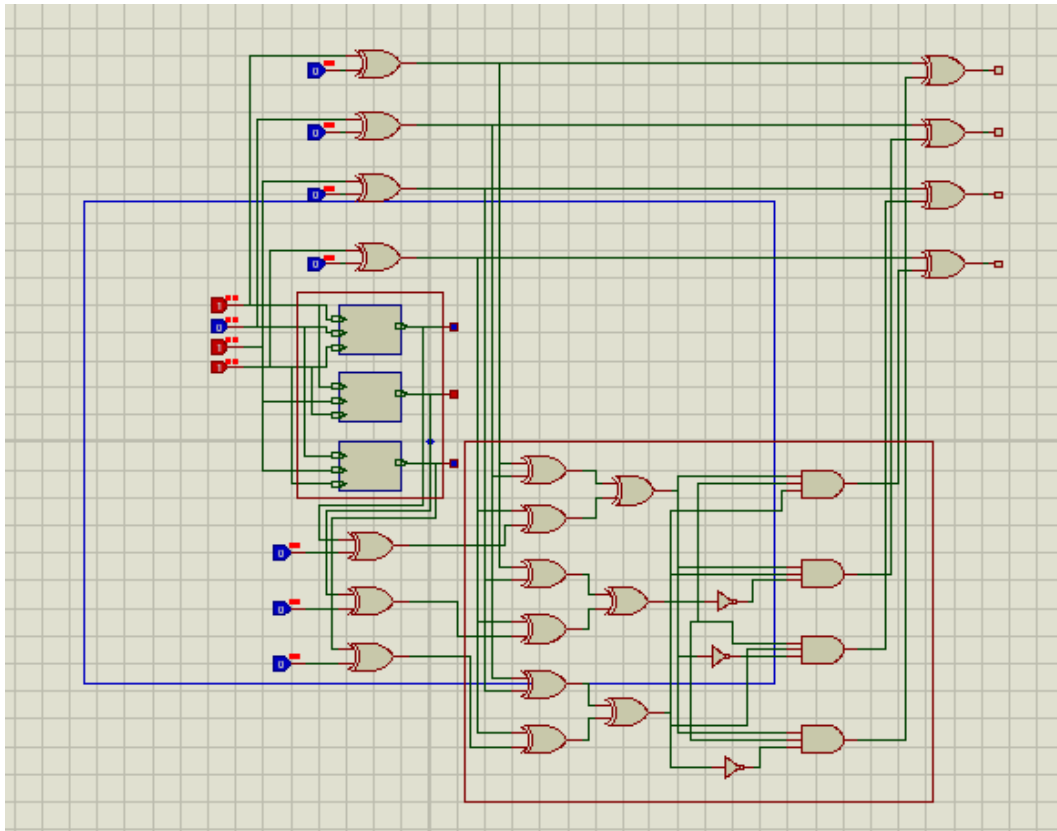


Figure 10: Circuito completo de Hamming simulado en Proteus

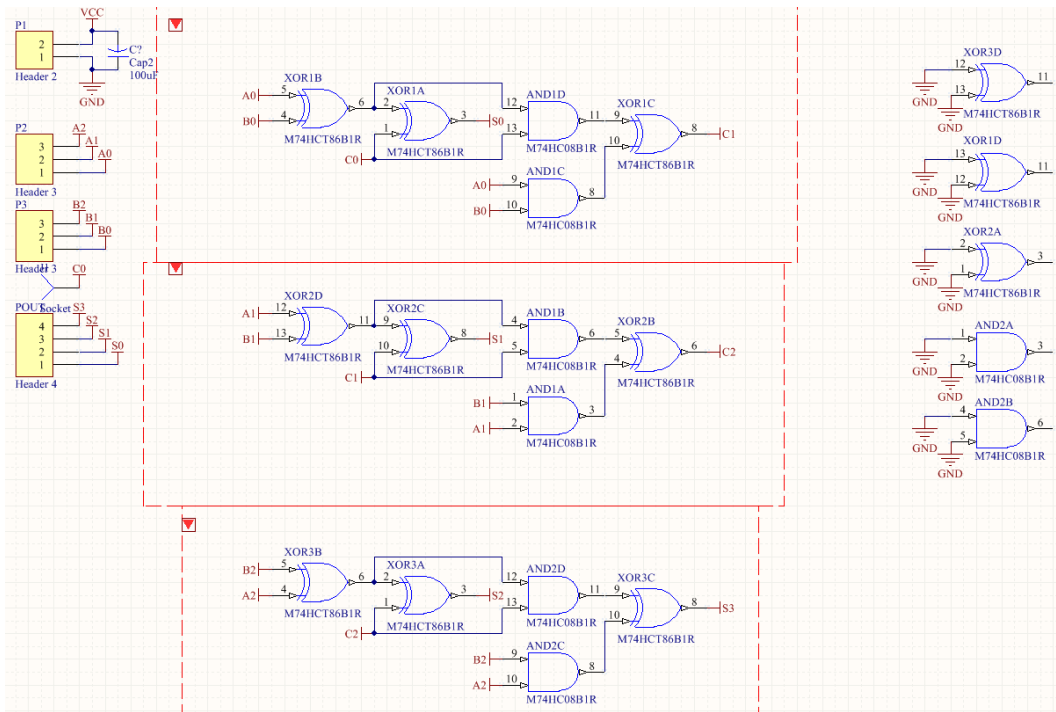


Figure 11: Sumador Carry Look Ahead

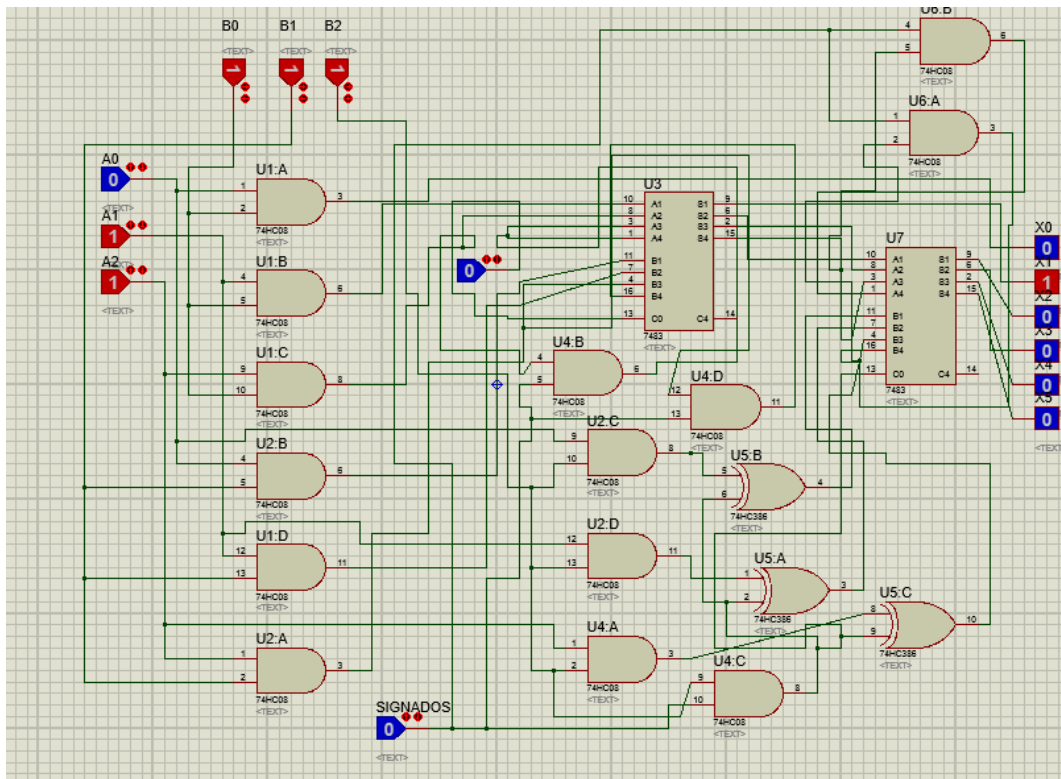


Figure 12: Multiplicador

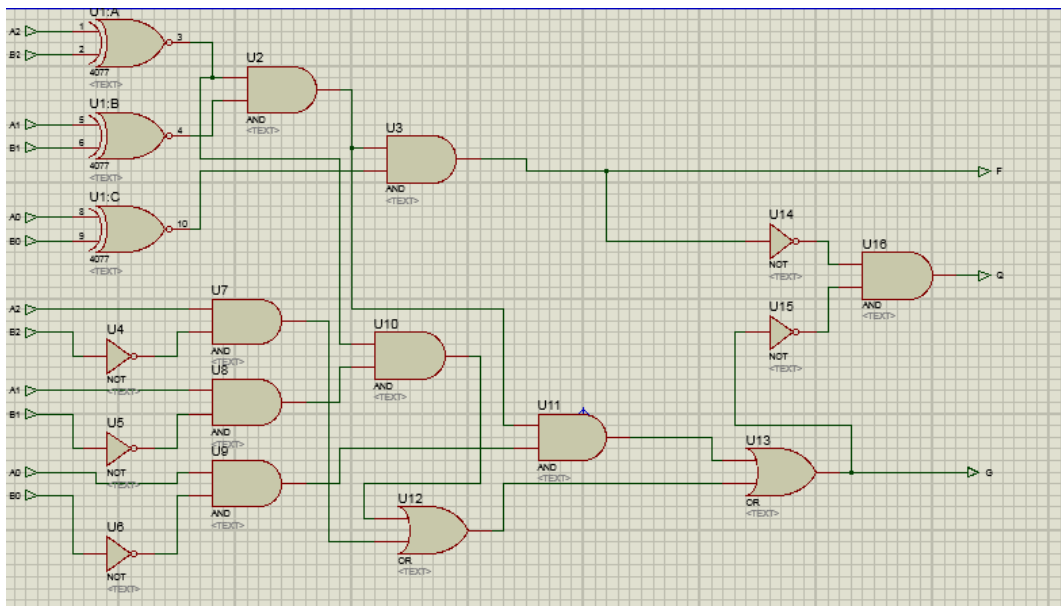


Figure 13: Compare