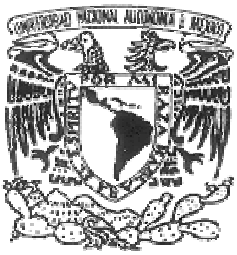


UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

PROGRAMA DE MAESTRIA Y DOCTORADO EN INGENIERIA

FACULTAD DE INGENIERÍA

"DISEÑO DE UN CIRCUITO LÓGICO PARA FUNCTORES LÓGICOS"

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

INGENIERÍA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

P R E S E N T A :

VEGA RAMÍREZ ALEJANDRO ANTONIO

TUTOR:
M. en C. JOSÉ LUIS PEREZ SILVA

2006

Índice

Pagina

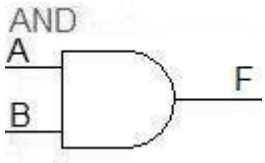
Introducción

Introducción

El diseño de sistemas digitales es una área de la electrónica que se caracteriza por resolver problemas desde el punto de vista discreto, es decir, verifica si está o no presente una variable definida, o bien, si está o no activada una señal. Por ello, maneja los estados verdaderos o falsos, comúnmente conocidos como 1's ó 0's. Esta lógica de 1's y 0's no considera el tipo de dato que está procesando y operando, sino que simplemente lo asume como un conjunto de números binarios de tamaño variable, sobre los cuales hace las operaciones necesarias para resolver dichos problemas.

La base de esta lógica "binaria" es el algebra de Boole (ver anexos). George Boole fue un matemático inglés que en 1854 publicó *Las leyes del pensamiento* sobre las cuales son basadas las teorías matemáticas de Lógica y Probabilidad. Boole aproximó la lógica en una nueva dirección reduciéndola a un álgebra simple, incorporando lógica en las matemáticas. Agudizó la analogía entre los símbolos algebraicos y aquellos que representan formas lógicas. Su álgebra consiste en un método para resolver problemas de lógica que recurre solamente a los valores binarios 1 y 0 y a tres operadores: AND (y), OR (o) y NOT (no). Son estos tres operadores los elementos fundamentales para construir sistemas digitales diseñados para la resolución de problemas en general.

El funcionamiento del operador AND consiste en las combinaciones mostradas a continuación:

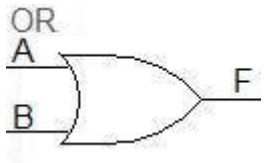


$0 \cdot 0 = 0$
$0 \cdot 1 = 0$
$1 \cdot 0 = 0$
$1 \cdot 1 = 1$

Figura i.1. Operador AND.

De acuerdo a lo anterior, se define que el operador AND presenta un 1 a la salida cuando se cumple la condición de que ambas entradas estén presentes, es decir, que en ambas exista un 1, de lo contrario no se tiene ese resultado.

El funcionamiento del operador OR consiste en las combinaciones siguientes:

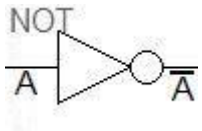


$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 1$

Figura i.2. Operador Or.

Según lo anterior, se define que el operador OR presenta un 1 a la salida cuando se cumple la condición de que al menos una de las entradas esté presente, es decir, que al menos una de ellas tenga un 1.

El operador NOT consiste en invertir el valor que se tenga a la entrada, es decir:



0 = 1
1 = 0

Figura i.3. Operador NOT.

Se presenta un ejercicio para ejemplificar la manera de resolver un problema de diseño basado en esta lógica, aplicando los operadores mencionados.

Se ha convenido formar un comité en la universidad para tomar decisiones claras. Este comité está compuesto por el director, un jefe de departamento, un profesor y un estudiante. Una mayoría de 2/3 decide, pero para hacer las cosas adecuadamente los votos se reparten como sigue: 4 votos para el director y para el jefe de departamento, 3 votos para el profesor, y uno para el estudiante. Cada miembro del comité dispone de un botón que, al ser oprimido, cierra el interruptor para aprobar, o bien, no aprobar, al no oprimirse.

Diséñese un sistema mínimo en el que se encienda el foco si y solo si el numero de votos cumple con el mínimo necesario.

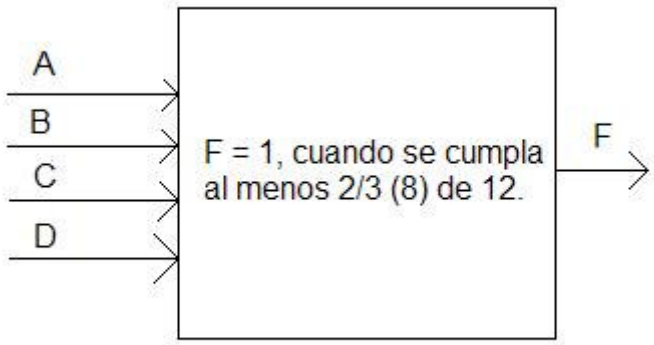


Figura i.4. Bloque que representa el comportamiento del sistema.

Lo primordial en la resolución de un problema de este tipo es la buena interpretación del comportamiento deseado. El sistema tiene entonces 4 variables de entrada, cada una asignada a los integrantes del comité, de la forma siguiente:

- A para el director (4 votos).
- B para el jefe de departamento (4 votos).
- C para el profesor (3 votos).
- D para el alumno (1 voto).

Haciendo un total de 12 votos. Y una sola salida, que será verdadera de acuerdo con la condición establecida, figura (i.4).

Posterior a esto, se plantea una tabla de verdad donde se consideran todas las posibles combinaciones que las variables de entrada pueden tomar.

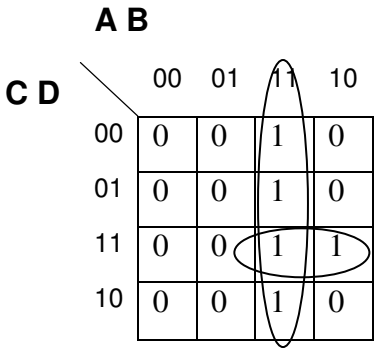
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Tabla de verdad que muestra el comportamiento del sistema.

De acuerdo con la tabla mostrada, la salida será verdadera cuando se cumpla la condición de que existan 2/3 del total, esto es, 8 de 12 votos.

Se observa que salida F toma valores verdaderos a partir de la combinación binaria "1011", cuando el director, el profesor y el alumno oprimen su botón respectivo, haciendo un total de 8. Las siguientes combinaciones satisfacen la misma condición, por lo que también se consideran como verdaderas.

Posterior a esto, se procede a la reducción por algún método de minimización que existen para esta lógica binaria.



De acuerdo con los encierros mostrados, utilizando el método de reducción de mapas de Karnough, la función resultante para el control del sistema es la siguiente:

$$F = AB + ACD$$
$$F = A(B + CD)$$

El circuito resultante se muestra en la figura (i.5). Al factorizarse A, la función de salida utiliza un operador And entre esta variable A y la salida del operador Or, el cual considera las variables B o CD, en este último hay un operador And entre estas variables C y D, cuyas salidas están acopladas al operador Or, como lo muestra la figura (i.5).

De acuerdo con el circuito, existirá una salida verdadera siempre y cuando el director presione su botón correspondiente, y el jefe de departamento (operador And) esté de acuerdo y haga lo propio, o (operador Or), cuando el profesor y el alumno (operador And), también lo estén.

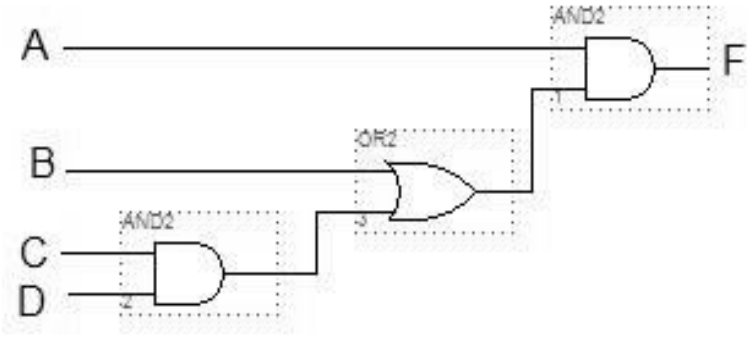


Figura i.5. Circuito resultante del ejemplo.

Pero, ¿que sucede si se interpretó mal la condición? Es decir, existirá una salida verdadera cuando exista una mayoría arriba de 2/3, esto es, arriba de 8. Este pequeño inconveniente implica reconsiderar la tabla de verdad, de tal forma que se definan que combinaciones de las variables satisfacen la nueva condición. Así, la corrección sería como sigue:

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Tabla de verdad que muestra el comportamiento del sistema.

De acuerdo con esta nueva tabla, la salida será verdadera cuando se cumpla la condición de que existan arriba de 2/3 del total, esto es, al menos 9 de 12 votos.

Se observa que salida F toma valores verdaderos a partir de la combinación binaria "1101", cuando el director, el jefe de departamento y el alumno oprimen su botón respectivo, haciendo un total de 9. Las siguientes combinaciones satisfacen la misma condición, por lo que también se consideran como verdaderas.

Posterior a esto, se procede a la reducción por el mismo método de minimización que en la anterior situación.

C D \ A B	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	0	1	0
10	0	0	1	0

De acuerdo con los encierros mostrados, utilizando el mismo método de reducción de mapas de Karnough, la función resultante para el control del sistema es la siguiente:

$$F = ABD + ABC$$
$$F = AB(D + C)$$

Cuyo circuito resultante se muestra en la figura (i.6).

La función de salida, optimizada al factorizar AB, muestra que se aplica un operador And entre las variables A, B, y el resultado del operador Or, entre las variables C o D. Esto se observa en la figura (i.6).

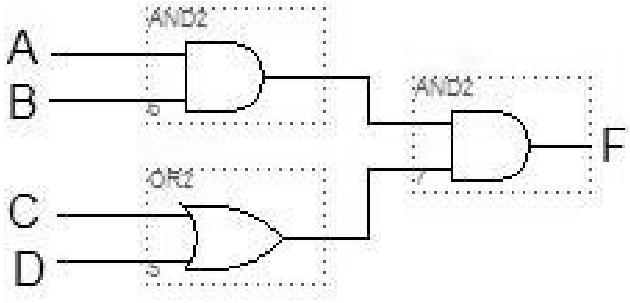


Figura i.6. Circuito resultante para la segunda consideración.

Según este nuevo circuito, se observa que la salida será verdadera cuando el director, el jefe de departamento y alguno de los otros integrantes del comité (operador And), el profesor o el alumno (operador or), presionen su botón correspondiente, de lo contrario, salida no opera, es decir, no enciende el indicador.

De acuerdo con este ejemplo mostrado, y las dos consideraciones tomadas para el diseño del sistema, se aprecia que existe, entre ambas, una reestructuración, o reconstrucción del circuito del sistema en cuestión; aun cuando los componentes utilizados son los mismos, la interconexión entre ellos cambia. Este pequeño ejemplo acusa uno de los principales inconvenientes al trabajar con los conectores convencionales de la lógica booleana, reconectar e, incluso, rediseñar el circuito del sistema en desarrollo.

El objetivo de este trabajo es evitar estos pequeños inconvenientes, introduciendo un nuevo elemento capaz de resolver estos detalles con un simple reajuste de ciertos parámetros internos, de modo que pueda responder en concordancia con

requerimientos establecidos, tanto para el funcionamiento del sistema en un momento determinado, como los arrojados por el proceso de diseño y minimización de la lógica booleana.

Por ello, a continuación, se presenta este nuevo elemento conector, cuyos fundamentos de operación, descritos en el capitulo 1 de este trabajo, le hacen capaz de adecuarse a los principios del algebra booleana, trabajando como alguno de los conectores ya mencionados.

Posteriormente, se sugieren dos plataformas de desarrollo sobre las cuales se pueda diseñar e implementar dicho elemento, en los capítulos 2 y 3, para ser aplicado en cualquier problema que involucre el algebra booleana y sus conectores básicos. En estos capítulos se explica el proceso de diseño y los resultados arrojados al haberse realizado pruebas con dichos diseños.

En seguida, en el capitulo 4, se hace un comparativo entre los resultados de ambas plataformas y definir cual de ellas es la mas idónea para la implantación de dicho elemento.

Finalmente, en el capitulo 5, se hacen algunas consideraciones que se deben tomar al momento de diseñar en la plataforma de desarrollo con la que se ha decidido trabajar, y las conclusiones que del desarrollo de este trabajo se han conseguido.

Capitulo 1

1.1.- La neurona Mc Cullock-Pitts

El neurofisiólogo W. S. Mc Cullock y el matemático W. Pitts en 1943 introducen el concepto de neurona formal, que es la base de su proyecto de red neuronal, el cual esquematiza un combinador lineal con umbral, con datos binarios múltiples a la entrada y un solo dato binario a la salida. En su modelo, la neurona formal asume dos posibles estados, uno activo y el otro inactivo, definiendo una clase de objetos caracterizados de la siguiente manera:

- Un numero *r* finito de entradas, cada una susceptible de tomar los valores 0 ó 1;
- Una salida única *y*, en la cual los valores posibles son 0 ó 1;
- Un valor de umbral *s*, entero positivo;
- Una regla de funcionamiento expresada como:

$$y = \begin{cases} 1 \; si \sum_{i=1}^{r} x_i \geq s \\ 0 \; en \; otros \; casos \end{cases} \tag{1}$$

Dichas neuronas reciben los estímulos en la entrada y los elaboran. Esta puede ser muy sofisticada, pero en un caso simple se puede pensar en valores entrantes sencillos, sean directos o multiplicados por un valor de peso dado. Estos valores, o el resultado de las multiplicaciones, se suman, y si esta adición supera un cierto valor umbral, la neurona se activa operando su salida. El peso indica la eficiencia sináptica de la línea de ingreso y sirve para cuantificar la importancia, un valor importante tendrá un peso elevado, mientras que un valor poco útil tendrá un peso inferior. Se puede pensar que si dos neuronas se comunican entre si utilizando algunas conexiones, entonces estas tendrán un peso mayor. Figura (1.1).

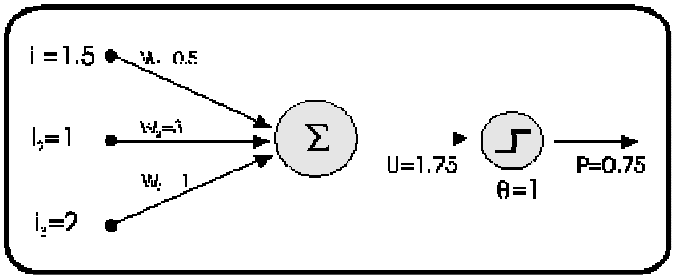


Figura 1.1. Representación general del funcionamiento de la neurona artificial de Mc Cullock y Pitts, conocida también como unidad lógica de umbral. W son los pesos, Θ es el valor del umbral para activar la función.

Mediante las neuronas formales se puede construir una maquina del tipo "semiautoma" con un numero finito *r* de entradas *x* y un numero finito *s* de estados *q* de la maquina.

1.2.- Concepto de Functor Lógico

El diseño de sistemas digitales es una rama de la ingeniería en electrónica, cuyo objetivo es el control de sistemas aplicando como principio de diseño y operación la lógica booleana. Los elementos básicos sobre los cuales se estructuran son las compuertas lógicas básicas, que son la Or, la And y la Not, así como combinaciones de estas para crear otras variantes en cuanto a operación se refiere, además de conformar dispositivos digitales mas complejos por el numero de componentes que lo integran. Para lograr un sistema digital es necesario acoplar varios de estos componentes mencionados, de tal manera que el problema planteado se solucione con un sistema de este tipo, basado en los principios antes mencionados. Existen diseños que son planteados para solucionar un solo problema, por lo que si se desea agregar alguna otra variable, o hacer un cambio dentro de este, es necesario reconsiderar el problema para rediseñarlo con las correcciones necesarias, lo que ocasiona que el circuito armado se tenga que reconectar, o incluso, en el peor de los casos, rearmar. Para reducir este inconveniente, se plantea el concepto de functor.

Un functor lógico (Término acuñado por nosotros) es un elemento lógico a partir del cual se pueden generar todos los conectivos lógicos booleanos como casos particulares de éste. Este functor es un resultado inmediato de la lógica dependiente del umbral. Se mostrará que los conectivos de la lógica booleana se generan, para un conjunto definido de entradas, a partir de este functor como una dependencia del umbral del modelo neuronal presentado por Mc Cullock y Pitts. Si la función que gobierna el umbral es una función estocástica se puede tener un comportamiento azaroso del functor, de tal manera que los operadores lógicos cambien de acuerdo al valor de las entradas y del umbral en el tiempo en forma estocástica. Con todo esto, se puede desarrollar un sistema que se base en un sólo elemento lógico, que es el propio functor.

1.3.- Definición de los Operadores Generalizados

Se pueden generalizar los operadores neuronales empleando el caso de entradas excitadoras solamente de la forma siguiente, figura (1.2):

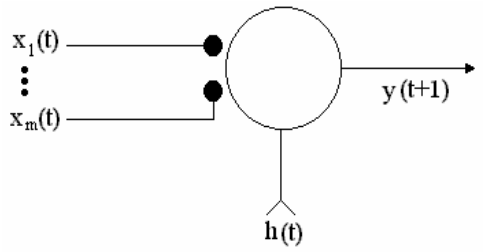


Figura 1.2. Diagrama que muestra la generalización del evento de un retardo y múltiples entradas excitadoras.

En este caso la regla de disparo es:

$$Dx(t) = \begin{cases} 1 \; si \; \sum_{i=1}^{m} \overset{\bullet}{x}_i(t) \geq h(t) \\ 0 \; si \; \sum_{i=1}^{m} \overset{\bullet}{x}_i(t) < h(t) \end{cases} \tag{2}$$

Como todos los pesos son iguales a uno, se puede decir que el operador generalizado es: al menos *h* (Valor umbral) de *m* (número de entradas), que tendrá una respuesta retardada si al menos *h* de las entradas excitadoras de las *m* están presentes. El símbolo con el cual se le representa es:

$$\prec_h^m$$

Se puede ver que en el caso particular "al menos 1 de 2" es el operador lógico temporal "Or".

$$\prec_1^2 \equiv D^t \vee$$

Y al menos "2 de 2" es el operador dependiente del tiempo "And", que se expresa como:

$$\prec_2^2 \equiv D^t \wedge .$$

Analizando el operador generalizado con entradas inhibitorias, se verá que en este caso se tienen *n* entradas y un umbral *h (t)*, como se muestra en la figura (1.3).

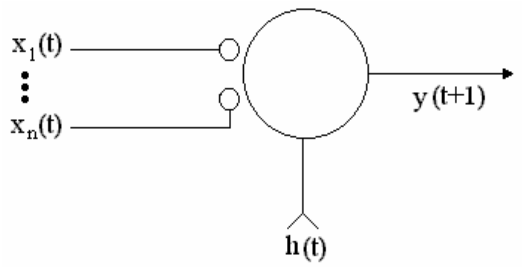


Figura 1.3. Diagrama que muestra un evento con n entradas inhibitorias y un umbral h (t).

Así que la regla de disparo será:

$$Dx(t) = \begin{cases} 1 \; si \sum_{i=1}^{m} \overset{\circ}{x}_i(t) \le h(t) \\ 0 \; si \sum_{i=1}^{m} \overset{\circ}{x}_i(t) > h(t) \end{cases} \tag{3}$$

que será descrita como cuando mucho *h* (valor de umbral) de *n* (número de entradas) que es el operador simétrico de al menos *h* de *m*.

Para ser representado, se eligió el símbolo:

$$\prec_{h}^{n}$$

En particular, el caso "cuando mucho 0 de 1", es el operador negación "Not" dependiente del tiempo, representado como:

$$\prec_{0}^{1} \equiv D\neg_{t}$$

y el operador "cuando mucho 0 de 2" es el operador lógico dependiente del tiempo "Nor".

$$\prec_{0}^{2} \equiv D\neg_{t} \overset{t}{\vee}.$$

Si se opera tanto con entradas excitadoras como con inhibitorias, se generaliza el caso, figura (1.4):

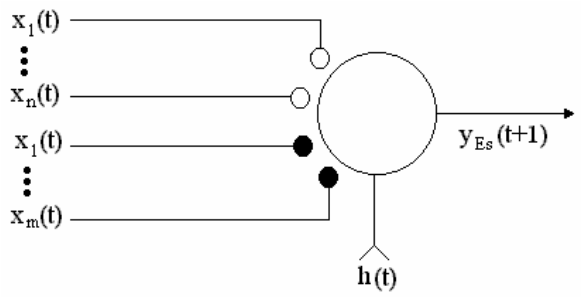


Figura 1.4. Diagrama que muestra la generalización del evento de un retardo y múltiples entradas tanto excitadoras como inhibitorias.

En este caso la regla de disparo será:

$$Dx(t) = \begin{cases} 1 \; si \; \sum_{i=1}^{m} \dot{x}_i(t) - \sum_{i=1}^{m} \circ x_j(t) \geq h(t) \\ 0 \; si \; \sum_{i=1}^{m} \dot{x}_i(t) - \sum_{i=1}^{m} \circ x_j(t) < h(t) \end{cases} \quad (4)$$

Este es el operador fundamental el cual se representará por el símbolo:

$$\overset{m}{\underset{n}{\updownarrow}}_{h}$$

Se le define de la siguiente manera: El operador dispara si al menos *m* (entradas excitadoras) menos *n* (entradas inhibitorias) es igual a *h* (umbral). Como se puede ver, el operador:

$$\overset{m}{\underset{n}{\updownarrow}}_{h}$$

se reduce a:

$$\overset{m}{\underset{h}{\succ}} \; para \; n = 0$$

y a:

$$\overset{n}{\underset{h}{\prec}} \; para \; m = 0.$$

Algunos casos típicos son:

$$\overset{1}{\underset{1}{\updownarrow}}_{0} \equiv D\overset{t}{\rightarrow}, \quad \overset{0}{\underset{1}{\updownarrow}}_{0} \Rightarrow \overset{1}{0} \equiv D\neg_{t} \; y \; \overset{2}{\underset{0}{\updownarrow}}_{2} \Rightarrow \overset{2}{2} \equiv D\overset{t}{\wedge},$$

Como es claro, el operador generalizado permite que un umbral dependiente del tiempo pueda estar modificando los operadores lógicos.

1.4.- El Operador Generalizado como Base de una Lógica Estocástica

Si el umbral no fuese simplemente una función dependiente del tiempo sino una función estocástica, el valor del umbral sería estocástico y por ende el operador que se aplique a la señal de entrada lo sería. Planteado de esta manera se puede presentar un nuevo operador generalizado estocástico cuya representación sería la siguiente, figura (1.5):

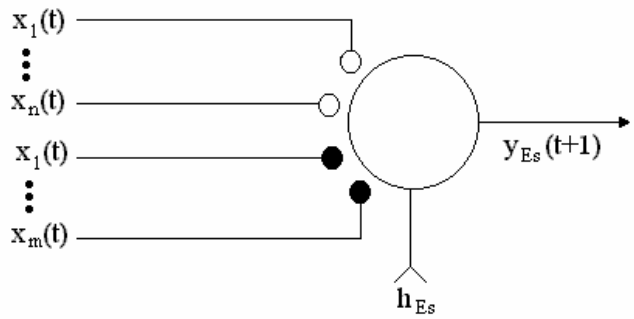


Figura 1.5. Diagrama representativo de un operador generalizado estocástico.

donde h_{Es} y y_{Es} *(t+1)* son el umbral estocástico y la respuesta del operador estocásticamente seleccionado respectivamente. x_1, ..., x_m son las entradas excitadoras y x_1, ..., x_n son las entradas inhibitorias.

Para este operador generalizado la regla de disparo será:

$$D_{Es}x = \begin{cases} 1\ si \sum_{i=1}^{m} \dot{x}_i(t) - \sum_{i=1}^{m} \overset{\circ}{x}_j(t) \geq h_{Es} \\ 0\ si \sum_{i=1}^{m} \dot{x}_i(t) - \sum_{i=1}^{m} \overset{\circ}{x}_j(t) < h_{Es} \end{cases} \tag{5}$$

Esta regla de disparo se puede entender como si al menos *m* entradas excitadoras menos *n* entradas inhibitorias, son mayores o iguales al valor estocástico de h_{Es}.

Para este operador fundamental de la lógica estocástica se empleará el siguiente símbolo:

$$h_{Es} \Updownarrow_{n}^{m}$$

Observando el desarrollo de este análisis, referido al concepto y funcionamiento del functor lógico, y para verificar que la operación de dicho elemento es comprobable utilizando un dispositivo físico que opere de acuerdo al planteamiento citado, se desarrollaron dos modelos del operador generalizado basados en dos plataformas distintas, las cuales se describen en los siguientes capítulos. Ambas ofrecen características de trabajo e implementación completamente diferentes, pero los resultados que estas arrojan son semejantes y

equiparables, como se verá mas adelante. Todo de acuerdo con la teoría de operación del operador generalizado.

Capitulo 2

2.1.- Implantación Electrónica del Operador generalizado utilizando la plataforma de desarrollo MAX + Plus II de Altera

Se construyó un circuito electrónico digital que representa al operador generalizado. El circuito electrónico básico es una neurona del tipo Mc Cullock y Pitts que se diseñó con base en compuertas básicas digitales, tanto para los circuitos sumadores de entradas excitadoras e inhibitorias, el comparador para la función de disparo, y para un circuito de retardo. Como operador generalizado temporalmente dependiente de umbral se empleó una entrada digital de 8 bits que simula el voltaje de umbral, que sería el voltaje de comparación del comparador de la función de disparo.

En la figura (2.1) se muestra el circuito completo del operador, que es el que se encarga de realizar las funciones ya mencionadas. Debido a que el editor de gráficos en el ambiente MAX + Plus II de Altera no permite la visión detallada de cada una de las partes, fue necesario reemplazarlas por bloques que representan las operaciones de suma, resta y comparación, así como el dispositivo que se encarga de operar el retardo.

A continuación, se describen los bloques de acuerdo con el funcionamiento que realizan dentro del circuito operador. Para los bloques sumadores se utilizaron los elementos básicos del sumador, como lo muestra la figura (2.2). Este sumador está estructurado con el arreglo de compuertas Xor y And, que son las básicas para generarlo. El proceso de diseño es el siguiente:

Diseño de un circuito sumador de dos números de 4 bits:

	C_2	C_1	C_0		
A	A_3	A_2	A_1	A_0	+
B	B_3	B_2	B_1	B_0	
	S_3	S_2	S_1	S_0	
	C_3	C_2	C_1	C_0	

→ MEDIO SUMADOR

→ SUMADOR COMPLETO

Tabla de verdad del comportamiento para el MEDIO SUMADOR.

A_0	B_0	S_0	C_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S_0 = \overline{A}_0 B_0 + A_0 \overline{B}_0$ $C_0 = A_0 B_0$

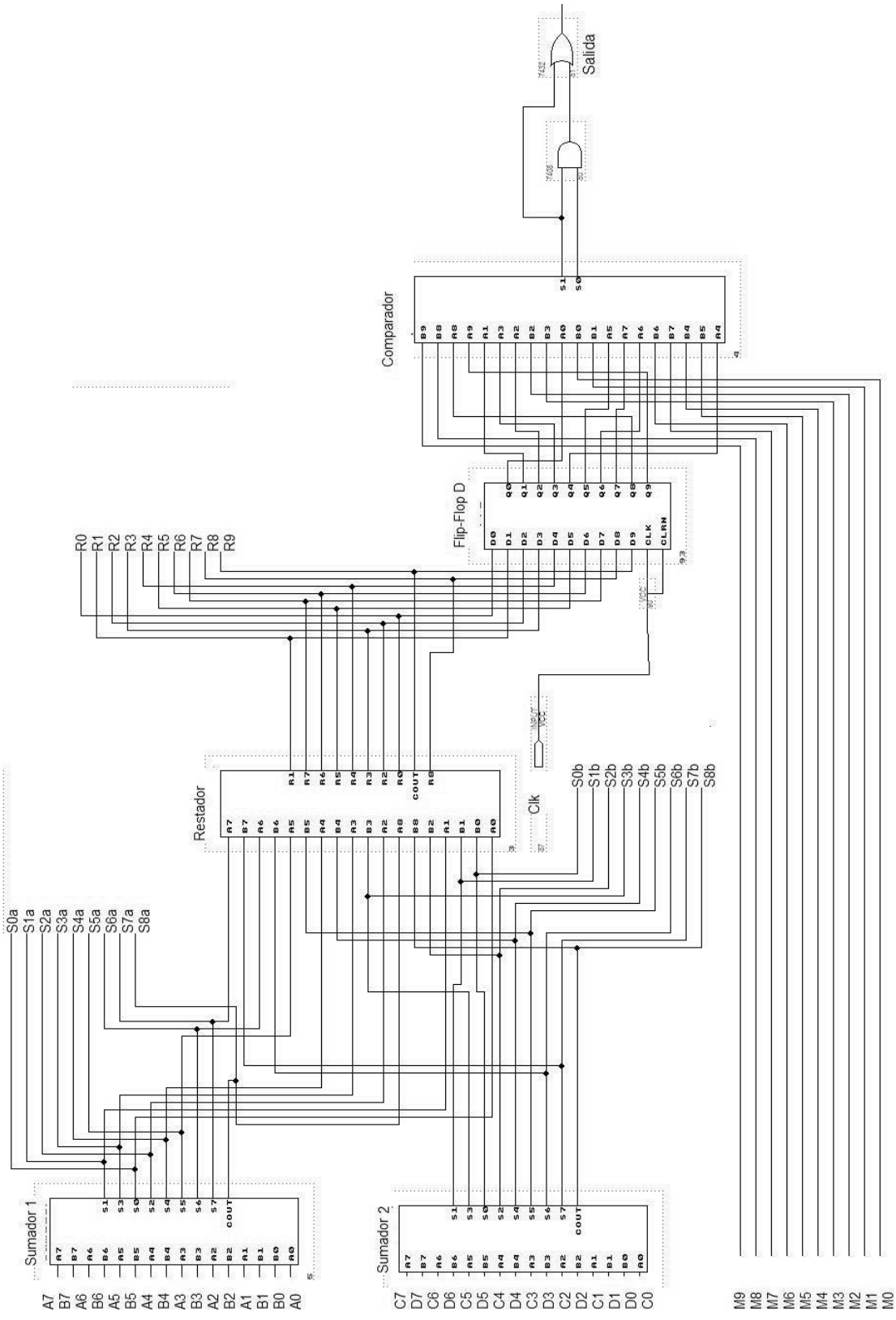


Figura 2.1. Circuito esquemático del operador generalizado dependiente del tiempo.

Las salidas muestran las variables a operar: para S_0, reduciendo la expresión con una compuerta equivalente, la Xor, tenemos:

$$S_0 = \overline{A}_0 B_0 + A_0 \overline{B}_0$$

$$S_0 = A_0 \oplus B_0$$

Para C_0, la combinación de las variables queda igual, utilizando una compuerta And:

$$C_0 = A_0 B_0$$

Y el circuito, para el medio sumador, es el siguiente, figura (2.3):

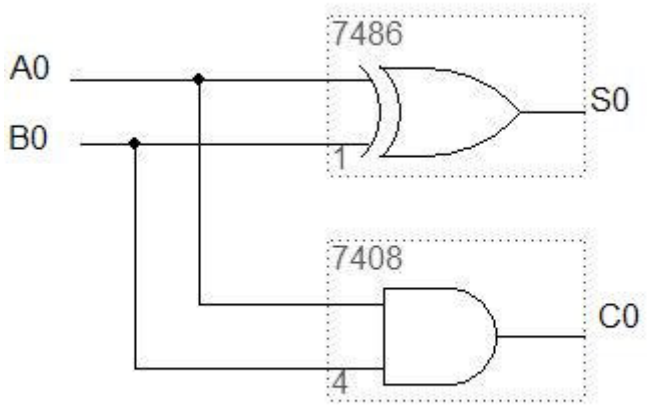


Figura 2.3. Medio Sumador.

Tabla de verdad para el comportamiento del SUMADOR COMPLETO.

A_1	B_1	C_0	S_1	C_1
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

$$S_1 = \overline{A}_1 \overline{B}_1 C_0 + \overline{A}_1 B_1 \overline{C}_0 + A_1 \overline{B}_1 \overline{C}_0 + A_1 B_1 C_0$$

$$S_1 = \overline{A}_1 (\overline{B}_1 C_0 + B_1 \overline{C}_0) + A_1 (\overline{B}_1 \overline{C}_0 + B_1 C_0)$$

$$S_1 = \overline{A}_1 (B_1 \oplus C_0) + A_1 \overline{(B_1 \oplus C_0)}$$

$$S_1 = A_1 \oplus (C_0 \oplus B_1)$$

El proceso mostrado reduce la expresión booleana con compuertas especiales Xor, por lo cual, la salida S_1 tiene como circuito de operación, el siguiente, figura (2.4):

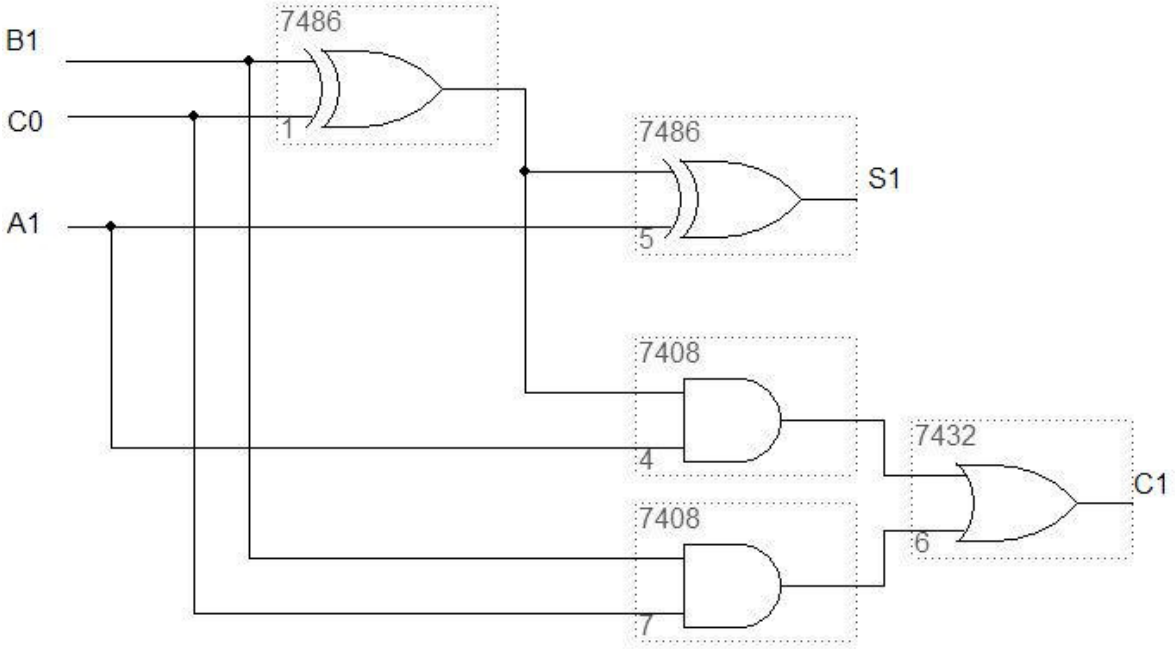


Figura 2.4. Sumador Completo.

Para el acarreo C_1, como lo muestra la figura anterior (2.4), el proceso de reducción es el siguiente:

$$C_1 = \overline{A}_1 B_1 C_0 + A_1 \overline{B}_1 C_0 + A_1 B_1 \overline{C}_0 + A_1 B_1 C_0$$

$$C_1 = B_1 C_0 (\overline{A}_1 + A_1) + A_1 (\overline{B}_1 C_0 + B_1 \overline{C}_0)$$

Para la expresión $(\overline{A}_1 + A_1)$, de acuerdo con el algebra booleana (ver anexos), recibe un valor de 1, por lo que la expresión queda como:

$$C_1 = B_1 C_0 + A_1 (B_1 \oplus C_0)$$

Aplicando la compuerta equivalente para el segundo termino, como se observa en la figura (2.4), es afectada por una compuerta And y la variable A_1, además de otra And para B_1 y C_0. Finalmente, una conexión en paralelo de un medio sumador, como el de la figura (2.3), y 7 sumadores completos permite obtener un sumador en su aplicación para 8 bits, figura (2.2).

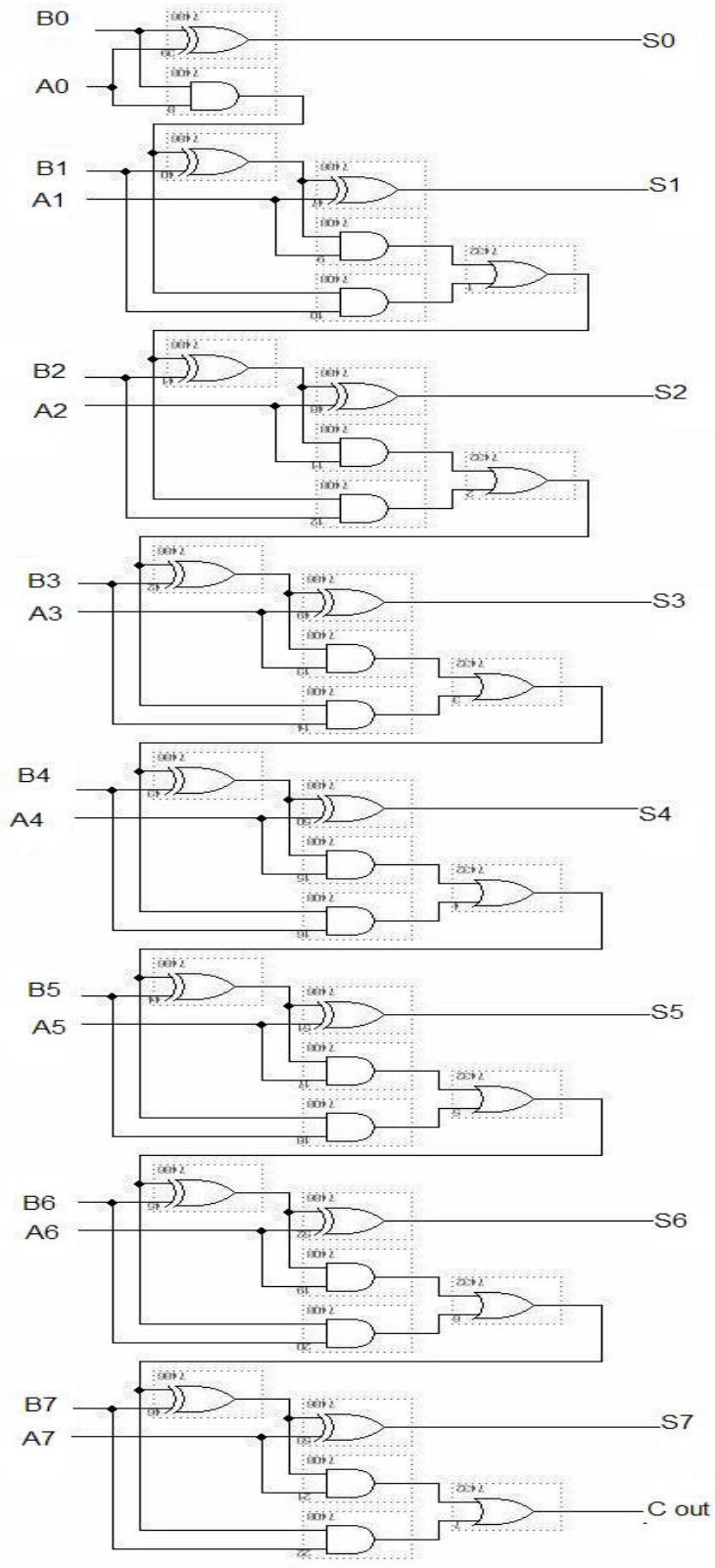


Figura 2.2. Circuito Esquemático del sumador, en este caso, desarrollado para dos números de 8 bits.

De la misma manera, el circuito básico del restador se muestra en la figura (2.5). Se trata de un reajuste del esquema del sumador, para obtener la operación mencionada, también consiste de los mismos elementos básicos del sumador, la compuerta Xor, y compuertas And para los acarreos. El diseño se presenta a continuación:

Diseño de un circuito restador de dos números de 4 bits:

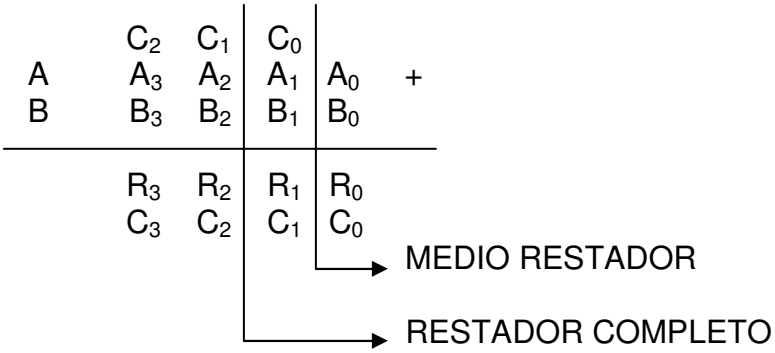


Tabla de verdad del comportamiento para el MEDIO RESTADOR.

A_0	B_0	R_0	C_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$R_0 = \overline{A}_0 B_0 + A_0 \overline{B}_0$ $C_0 = \overline{A}_0 B_0$

Las salidas muestran las variables a operar, para R_0, reduciendo la expresión con una compuerta equivalente, la Xor, tenemos:

$$R_0 = \overline{A}_0 B_0 + A_0 \overline{B}_0$$

$$R_0 = A_0 \oplus B_0$$

Para C_0, la combinación de las variables queda igual, utilizando una compuerta And:

$$C_0 = \overline{A}_0 B_0$$

Y el circuito, para el medio restador, es el siguiente, figura (2.6):

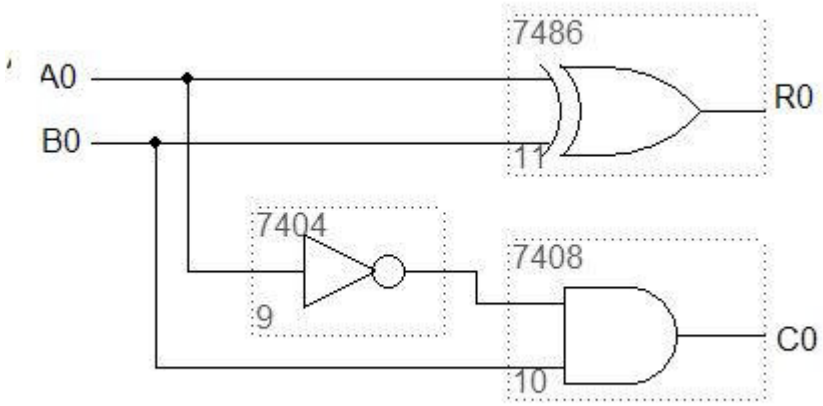


Figura 2.6. Medio Restador.

Tabla de verdad para el comportamiento del RESTADOR COMPLETO.

A_1	B_1	C_0	R_1	C_1
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$R_1 = \overline{A}_1\,\overline{B}_1\,C_0 + \overline{A}_1\,B_1\,\overline{C}_0 + A_1\,\overline{B}_1\,\overline{C}_0 + A_1\,B_1\,C_0$$

$$R_1 = \overline{A}_1\,(\overline{B}_1\,C_0 + B_1\,\overline{C}_0) + A_1\,(\overline{B}_1\,\overline{C}_0 + B_1\,C_0)$$

$$R_1 = \overline{A}_1\,(B_1 \oplus C_0) + A_1\,\overline{(B_1 \oplus C_0)}$$

$$R_1 = A_1 \oplus (B_1 \oplus C_0)$$

El proceso mostrado reduce la expresión booleana con compuertas especiales Xor, por lo cual, la salida R_1 tiene como circuito de operación, el siguiente, figura (2.7):

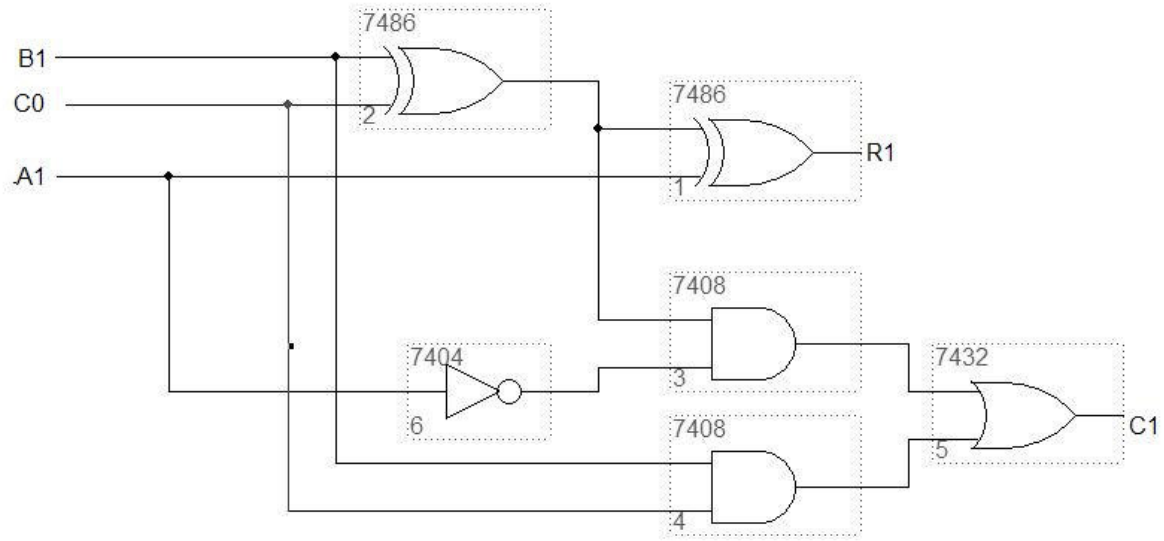


Figura 2.7. Restador Completo.

Para el acarreo C_1, como lo muestra la figura anterior, el proceso de reducción es el siguiente:

$$C_1 = \overline{A}_1\,\overline{B}_1\,C_0 + \overline{A}_1\,B_1\,\overline{C}_0 + \overline{A}_1\,B_1\,C_0 + A_1\,B_1\,C_0$$

$$C_1 = B_1\,C_0\,(\overline{A}_1 + A_1) + \overline{A}_1\,(\overline{B}_1\,C_0 + B_1\,\overline{C}_0)$$

Para la expresión $(\overline{A}_1 + A_1)$, de acuerdo con el algebra booleana (ver anexos), recibe un valor de 1, por lo que la expresión queda como:

$$C_1 = B_1\,C_0 + \overline{A}_1\,(B_1 \oplus C_0)$$

Aplicando la compuerta equivalente para el segundo termino, se obtiene el circuito como se observa en la figura (2.7), es afectada por una compuerta And y la variable $\bar{A}_1$, además de otra And para B_1 y C_0. Una conexión en paralelo de un medio restador, figura (2.6), y 8 restadores completos permite un restador de 9 bits, esto debido a que los sumadores presentan salidas de 8 bits y un acarreo, por lo que en total, las salidas son de 9 bits, figura (2.5).

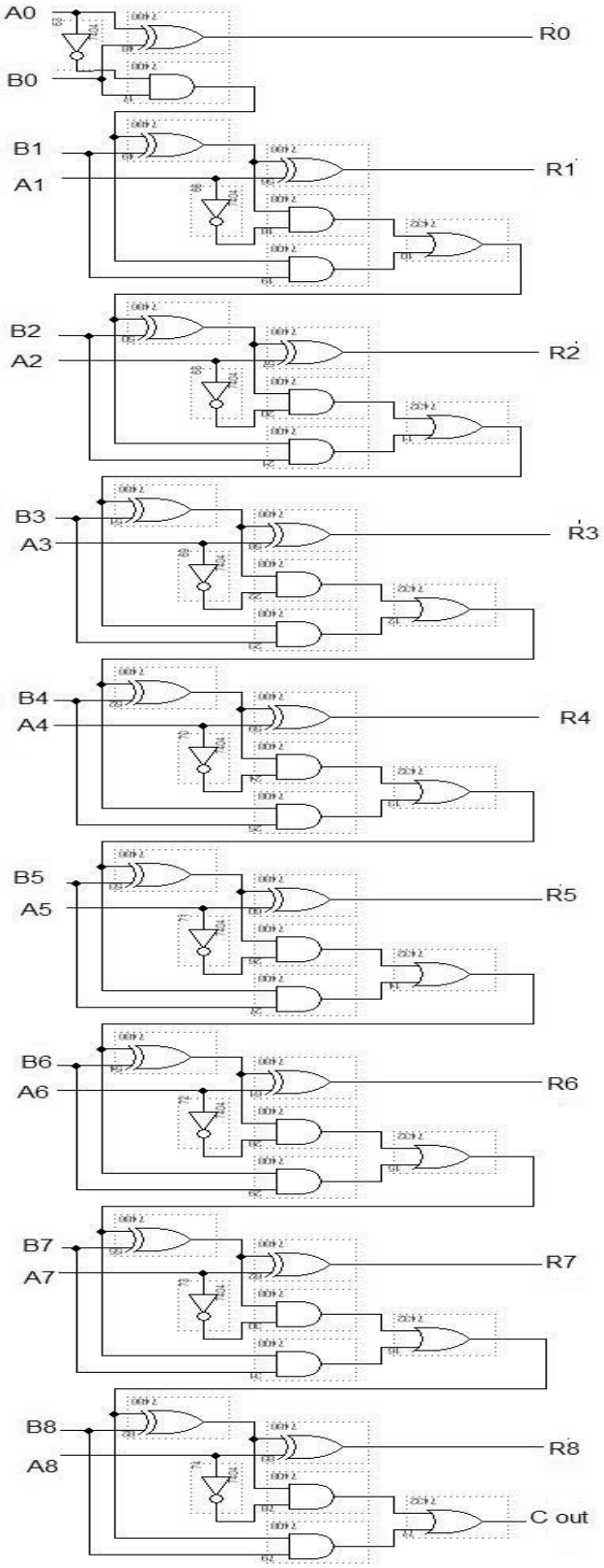


Figura 2.5. Circuito esquemático del restador desarrollado, para este caso, de 9 bits, ya que los sumadores presentan resultados de sumas con 8 bits y un acarreo.

Y la figura (2.8) muestra el circuito que realiza la operación de comparación. Este comparador está generado a partir de varios bloques, para reducir la complejidad en el diseño, por ello es que inicia en la figura (2.8b), donde se presenta el diseño para un comparador de 4 bits. El desarrollo se muestra a continuación:

Para el comparador, se planteó un diseño que pueda comparar dos números A y B, ambos de dos bits, cumpliendo las siguientes restricciones, las cuales son útiles para conformar uno de 10 bits.

La operación es de acuerdo con lo siguiente:

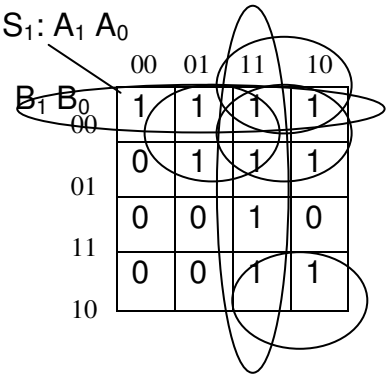
$A > B \Rightarrow S_1 = 1, \quad S_O = 0$
$A < B \Rightarrow S_1 = 0 \quad S_O = 1$ Siendo A = 2 bits y B = 2 bits.
$A = B \Rightarrow S_1 = 1, \quad S_O = 1$

A continuación se muestra la tabla de verdad, tabla (2.1), que plantea el comportamiento del circuito comparador de 2 números de 2 bits cada uno, además del proceso de reducción aplicando el método de mapas de Karnough, cuyo resultado plantea la figura (2.9).

A_1	A_0	B_1	B_0	S_1	S_0
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	1

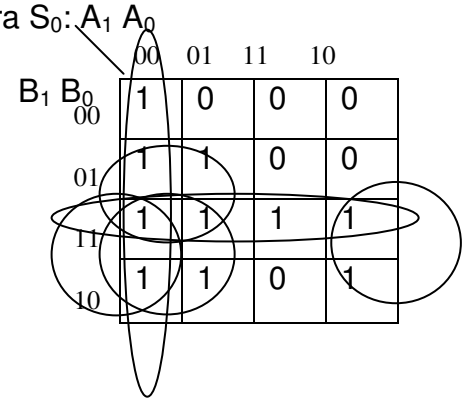
Tabla 2.1.

Para S_1: A_1 A_0



$S_1 = \bar{B}_1 \bar{B}_0 + A_1 A_0 + A_1 \bar{B}_1 + A_0 \bar{B}_1 + \bar{B}_0 A_1$

Para S_0: A_1 A_0



$$S_0 = B_1 B_0 + \overline{A}_1 \overline{A}_0 + \overline{A}_1 B_1 + \overline{A}_0 B_1 + B_0 \overline{A}_1$$

Las expresiones anteriores, obtenidas por el método de reducción de mapas de Karnough, generan en circuito comparador, para este primer caso, de dos bits, como lo muestra la figura siguiente (2.9):

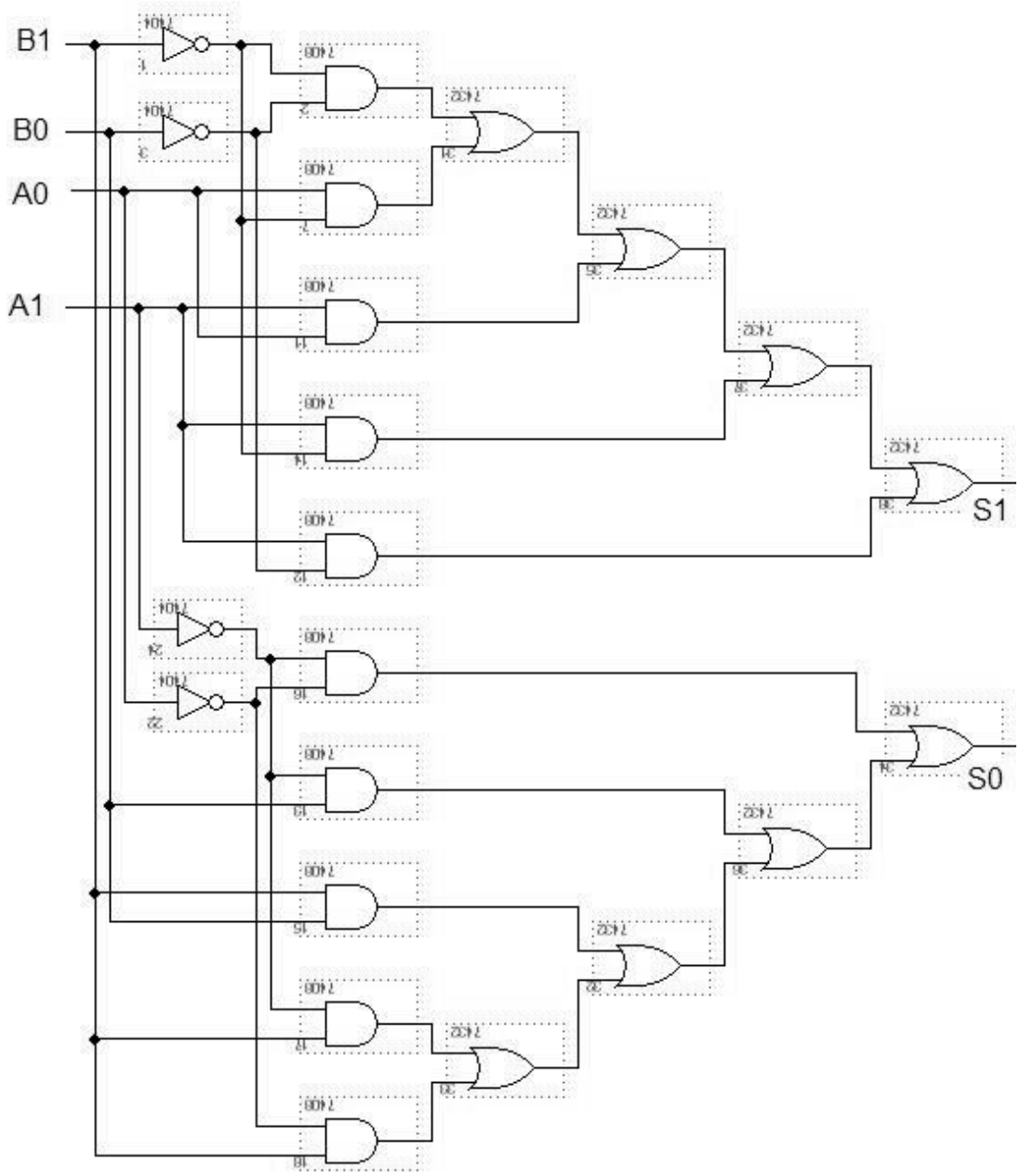


Figura 2.9. Comparador de dos números de dos bits.

Para generar el comparador para números de mas de dos bits, el procedimiento es similar, solo se deben de cuidar los subíndices y los agrupamientos de las

salidas para evitar errores en la construcción del circuito y en el funcionamiento del mismo.

Considerando el procedimiento descrito en el principio, para la construcción del comparador de dos números de mayor tamaño, esto es, de 4 bits, se plantea la misma operación, y la tabla (2.2) describe el comportamiento de la parte alta del número de 4 bits.

El contenido de la tabla es similar al de la primera, tabla (2.1), que describe el funcionamiento del comparador de dos números de dos bits, por lo que las funciones de salida son las mismas, solo varían los subíndices, como se observa en los resultados.

A_3	A_2	B_3	B_2	S_1	S_0
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	1

Tabla 2.2.

$$S_1 = \overline{B}_3\,\overline{B}_2 + A_3\,A_2 + A_3\,\overline{B}_3 + A_2\,\overline{B}_3 + \overline{B}_2\,A_3$$

$$S_0 = B_3\,B_2 + \overline{A}_3\,\overline{A}_2 + \overline{A}_3\,B_3 + \overline{A}_2\,B_3 + B_2\,\overline{A}_3$$

Ahora, considerando ambas salidas S_1 como S_{1H} y S_{1L} para el número A, y S_{0H} con S_{0L} para el número B, tenemos la siguiente tabla (2.3):

S_{1H}	S_{1L}	S_{0H}	S_{0L}	S_1	S_0
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	1

Tabla 2.3

De la cual obtenemos lo siguiente:

$$S_1 = \overline{S}_{0H}\,\overline{S}_{0L} + S_{1H}\,S_{1L} + S_{1H}\,\overline{S}_{0H} + S_{1L}\,\overline{S}_{0H} + \overline{S}_{0L}\,S_{1H}$$

$$S_0 = S_{0H}\,S_{0L} + \overline{S}_{1H}\,\overline{S}_{1L} + \overline{S}_{1H}\,S_{0H} + \overline{S}_{1L}\,S_{0H} + S_{0L}\,\overline{S}_{1H}$$

De lo cual se obtiene el circuito de la siguiente figura (2.8b):

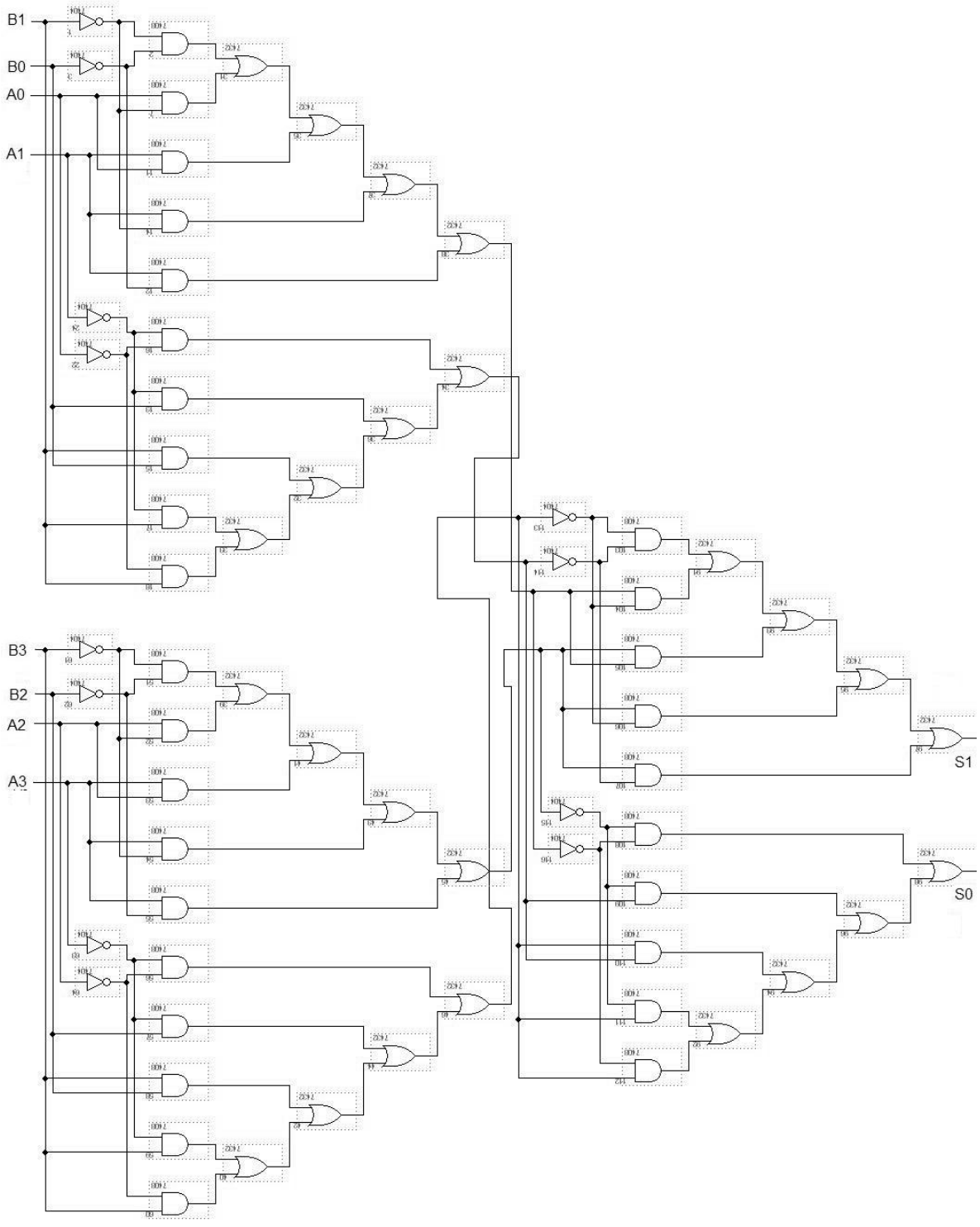


Figura 2.8b. Comparador para dos números de 4 bits.

Con este procedimiento obtenemos el comparador de dos números de 4 bits. Siguiendo el mismo procedimiento, planteado en la tabla (2.3), se puede generar un comparador de 8 bits, a partir de dos de 4 bits. Y finalmente, lograr el

comparador de 10 bits, a partir de uno de 8 y otro de 2 bits, que es el objetivo para construir el operador generalizado.

De acuerdo al planteamiento anterior, la figura (2.8a) muestra el comparador, formado por dos bloques comparadores de 4 bits cada uno para generar uno de 8 bits. Y la figura (2.8) presenta el bloque comparador de 8 bits y la adición de dos bloques comparadores más para tener un comparador de 10 bits. Se maneja uno de 10 bits, ya que el restador es de 9 bits, mas un acarreo, y para evitar errores de incongruencia con el paquete simulador, se asumen 10 bits.

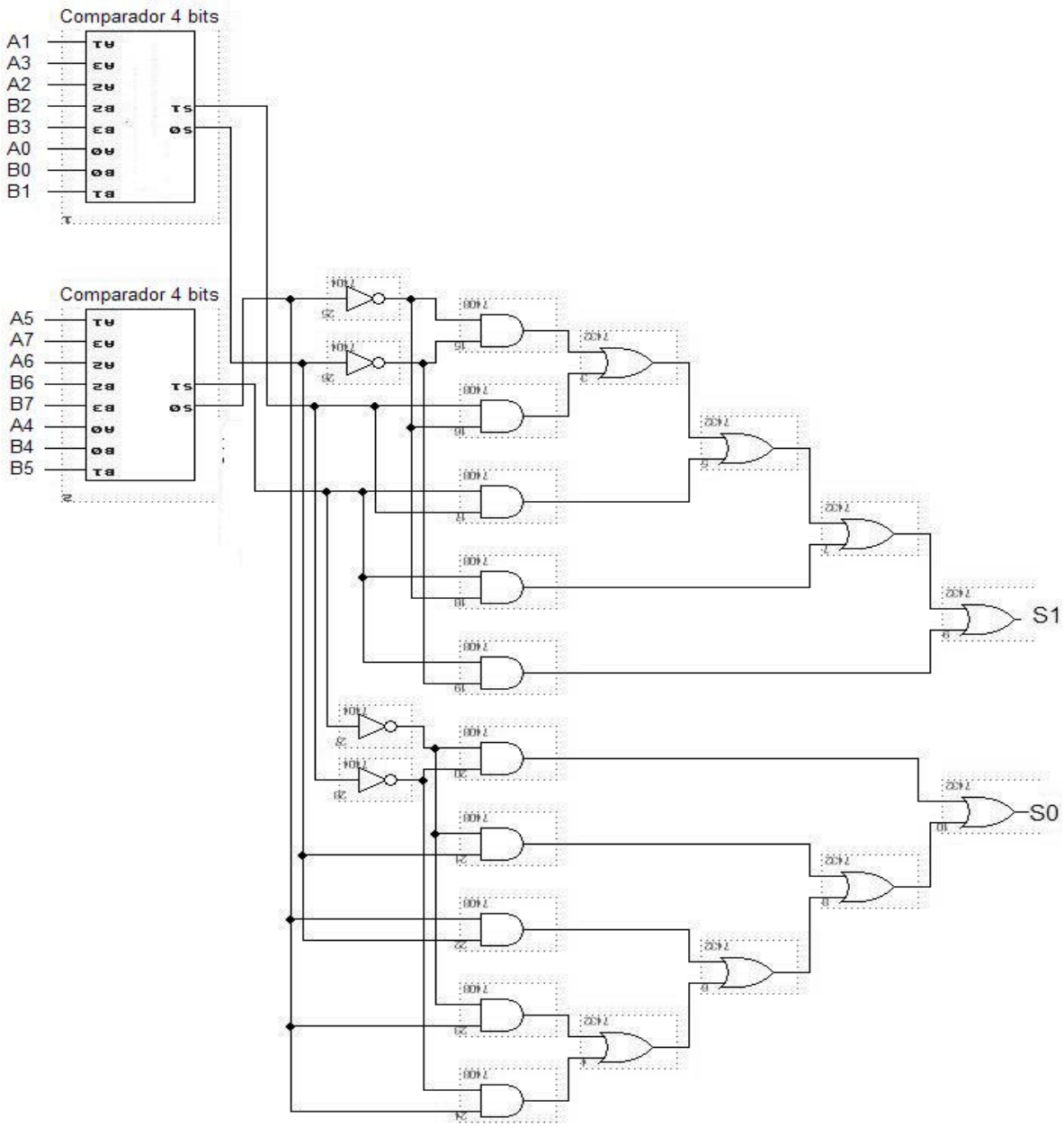


Figura 2.8a. Circuito esquemático del comparador, de 8 bits, ya que el editor grafico no es suficientemente amplio para detallar los componentes completos.

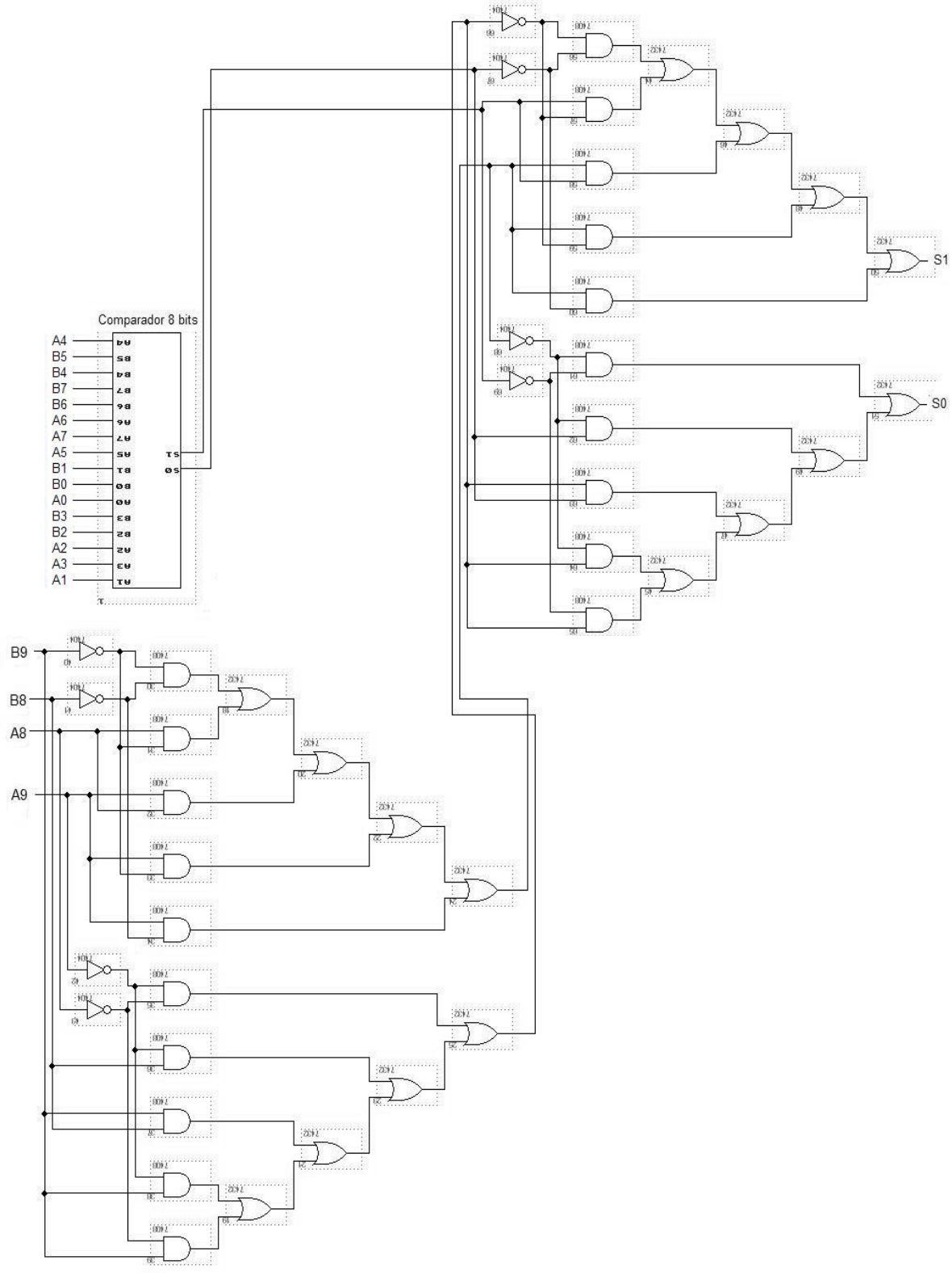


Figura 2.8. Circuito esquemático del comparador de 10 bits, ya que el restador genera 9 bits y una carga en sus resultados.

El arreglo de flip-flop's, que aparece en la figura (2.10), es el que se encarga de retardar las señales, por un periodo de 1ms, para que estas tengan un valor estable, cuando sean aplicadas en el elemento comparador, y evitar lecturas incorrectas, o "glitches", a la salida.

La figura (2.11) muestra la constitución interna del flip-flop tipo D.

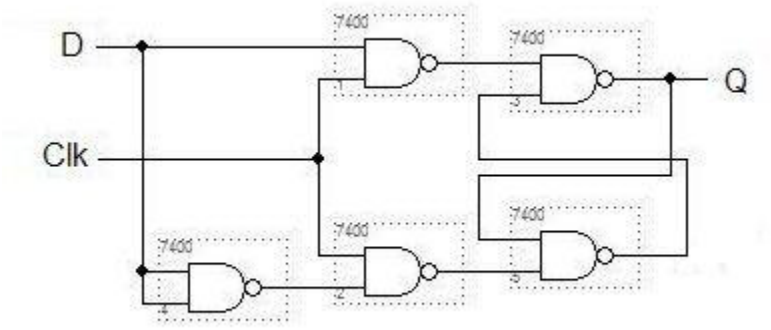
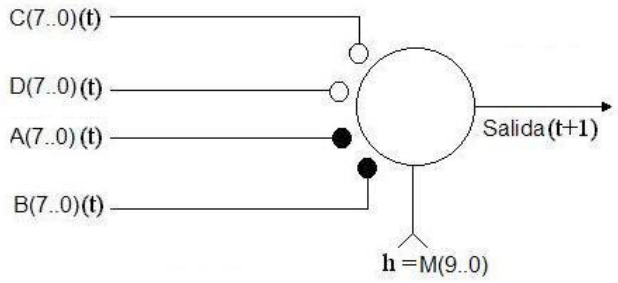


Figura 2.11. Estructura interna del flip-flop tipo D. Un arreglo de 10 de estos elementos permite el retardo de 1 ms.

Finalmente, el arreglo de compuertas Or y And, en la figura (2.1), permite obtener la salida especificada para el operador generalizado, de acuerdo al planteamiento realizado en el primer capitulo de este trabajo.



$$Salida\ (t+1) = \begin{cases} 1\ si \sum_{i=A}^{B} \bullet\, i(t) - \sum_{j=C}^{D} \circ\, j(t) \geq M(9..0) \\ 0\ si \sum_{i=A}^{B} \bullet\, i\ (t) - \sum_{j=C}^{D} \circ\, j(t) < M(9..0) \end{cases}$$

Figura 2.12. Diagrama representativo del operador generalizado basado en la plataforma de Altera. A(7..0) y B(7..0) son las entradas excitadoras, C(7..0) y D(7..0) son las entradas inhibitorias, M(9..0) es el valor de umbral.

Para comprobar que todo este desarrollo realizado opere de acuerdo con los fundamentos del operador generalizado, figura (2.12), se realizan pruebas con dicho diseño, lo cual arroja como resultado, las siguientes graficaciones, mostradas a continuación, en las figuras (2.13) y consecuentes.

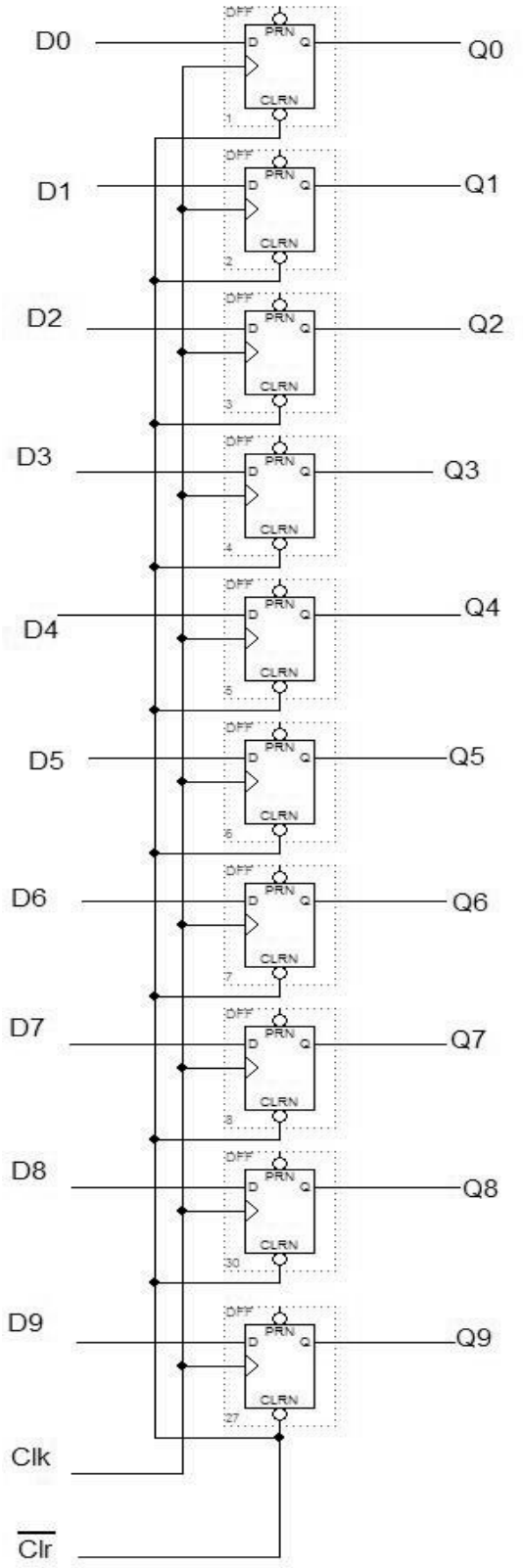


Figura 2.10. Circuito esquemático del dispositivo de retardo, para obtener los datos de operación sin rebotes.

2.2.- El Operador Generalizado estocástico con Altera

Para este caso se utiliza el circuito del operador generalizado, desarrollado con MAX + Plus II, considerando como generador del umbral a la entrada de 8 bits, como ya antes se había mencionado, y cuyo diagrama esquemático se muestra en la figura (2.1). Para ello se introduce una señal periódica del tipo rampa descendente, de 255 a 0, en su forma digital, figura (2.13).

Con este circuito se consideraron varios casos de generación estocástica y las gráficas de respuesta se muestran en las siguientes figuras:

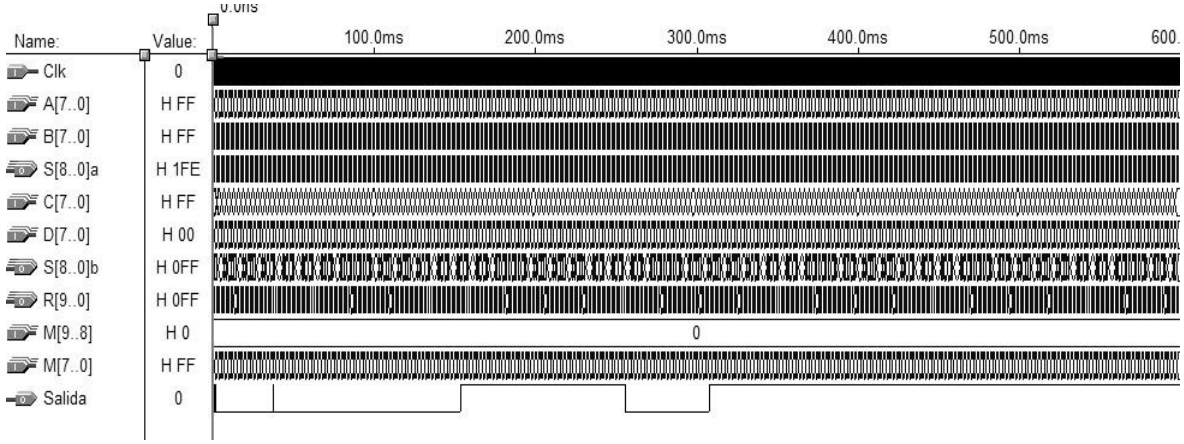


Figura 2.13. Respuesta del operador generalizado con un patrón estocástico a combinaciones binarias en las entradas, y una señal de umbral tipo rampa.

Para observar a detalle la actividad del operador, y haciendo referencia al diagrama representativo del operador, figura (2.12), se presentan figuras parciales donde se aprecia el comportamiento de este, de acuerdo con lo planteado.

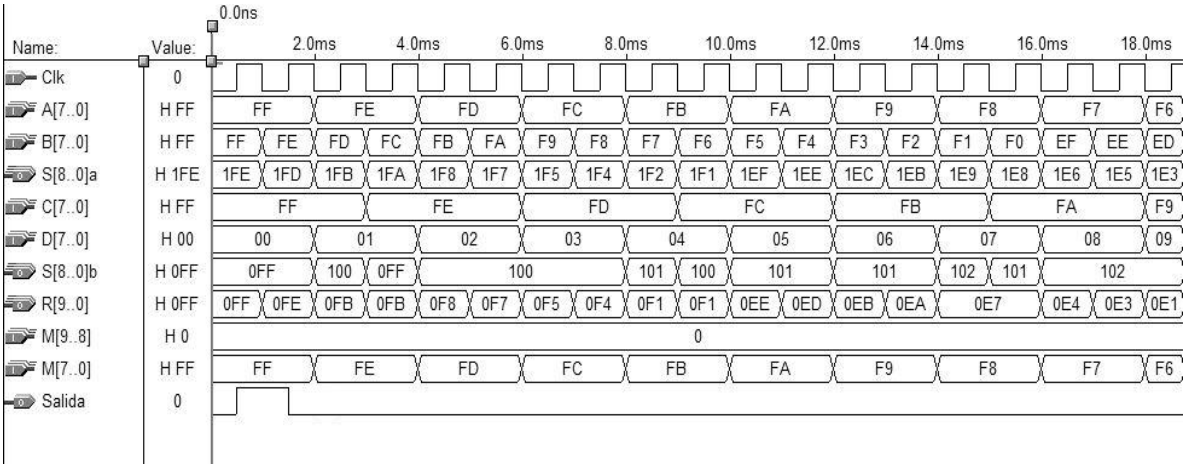


Figura 2.13a. Respuesta del operador en los primeros 18 ms de simulación.

En la figura (2.13a) se observa el funcionamiento planteado para el operador, la señal M (7..0) muestra el patrón estocástico. Como se mencionó, se trata de una señal rampa descendente, de 255, o FF, a 0, la cual cambia de valor cada 2 milisegundos. Las cantidades observadas están en base hexadecimal, y dado que se manejan 8 bits, es más útil esta representación, ya que en hexadecimal solo se requieren dos dígitos para representar los 8, en base binaria. M (7..0) es la señal de umbral que se compara con el resultado de la sustracción, que se observa en la figura como R(9..0), aquí también se observan cantidades en hexadecimal para mayor practicidad. A(7..0) y B(7..0) se suman para generar el resultado en S(8..0)a, mientras que C(7..0) y D(7..0) se suman para generar la cantidad resultante S(8..0)b, todas estas cantidades también están en hexadecimal, pues son datos de 8 bits. Ambas señales se restan para generar la señal R(9..0) que, como se mencionó, se compara con la señal de umbral. R(9..0), que es una salida, cambia su valor de acuerdo con B(7..0), ya que esta ultima es la cantidad que cambia con mas frecuencia, cada milisegundo, que es con la que opera el circuito, lo cual produce un resultado nuevo en la suma S(8..0)a, y por ende, una cantidad distinta en R(9..0). Con esto, salida cambia a 1, si M(7..0) ≤ R(9..0), de lo contrario, salida es 0, como se aprecia en la figura (2.13a). Salida es 1 cuando M(7..0), FF, ≤ R(9..0), 0FF, esto es, "al menos FF de FF" (operador And), dado que la suma de las entradas excitadoras, S(8..0)a = 1FE hexadecimal, es mayor que la suma de las inhibitorias, S(8..0)b = 0FF, y, de acuerdo con el capitulo primero, hay un 1 a la salida cuando las entradas excitadoras no están inhibidas, y el resultado de la resta es igual al umbral.

Después, salida es 0 cuando M(7..0), FF, > R(9..0), 0FE, ocurre el caso "cuando mucho FF de FE" (operador Nor), esta situación ocurre dado que la suma de las entradas excitadoras, S(8..0)a = 1FD, es mayor que la suma de las inhibitorias, S(8..0)b = 0FF, y de acuerdo con el capitulo primero, hay un 0 cuando las excitadoras no están inhibidas, pero el umbral es mayor al resultado de la resta. Todos los valores presentados en esta figura, y en las posteriores, están en base hexadecimal.

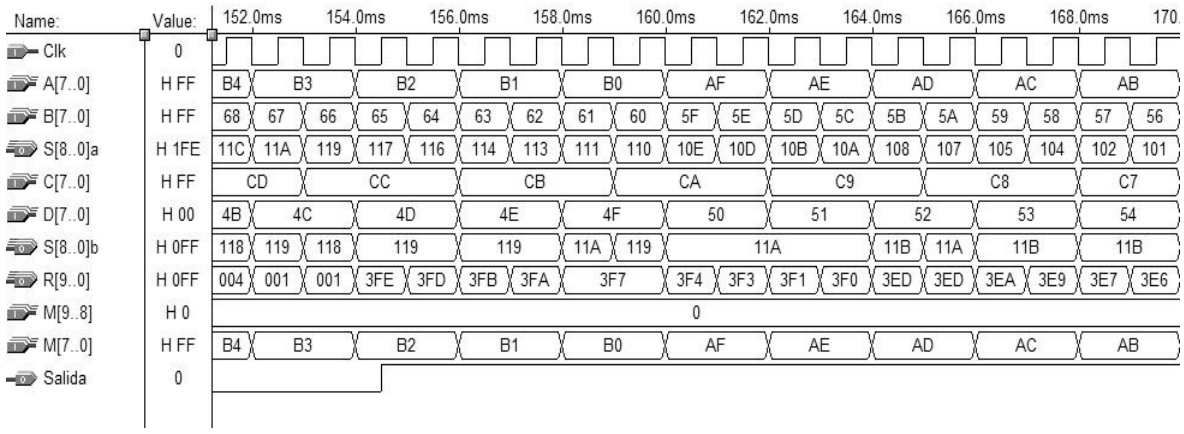


Figura 2.13b. Respuesta del functor lógico para las mismas combinaciones binarias en las entradas y en el umbral, pero en el periodo 152 ms a 170 ms.

La figura (2.13b) muestra el detalle del comportamiento de la salida de acuerdo con el resultado de la resta R(9..0) y el valor del umbral, M(7..0). Se observa como R(9..0) cambia de valor, de 001 hexadecimal a 3FE hexadecimal, con lo cual salida toma el valor de uno, previo retardo de 500 microsegundos, puesto que M(7..0) tiene un valor de B2 hexadecimal, cumpliéndose la condición, salida es 1 cuando R(9..0) ≥ M(7..0), es decir, "cuando mucho B2 de 3FE" (operador Nor), este caso se cumple debido a que la suma de las entradas excitadoras, S(8..0)a, 117 hexadecimal, es menor que la suma de las inhibitorias S(8..0)b, 119 hexadecimal. Las entradas inhibitorias "inhiben" a las excitadoras, por ello, como se ha descrito en el capitulo 1, hay un 1 a la salida cuando están inhibidas las entradas excitadoras, y el resultado de la resta es mayor al umbral. La frecuencia de trabajo del circuito es de 1 mS, por lo cual, cada vez que se tiene un flanco de subida del pulso en este periodo, los datos entran al comparador. De ahí que se note un retardo de 500 µS, esto debido a que hay una espera al flanco para entrar el dato 3FE hexadecimal al comparador. Con ello, salida toma el valor correcto, de acuerdo con el funcionamiento establecido. Este comportamiento es el mismo que se ha comentado en la figura (2.13a), solo que en esta figura (2.13b) el periodo es de 152 ms a 170 ms.

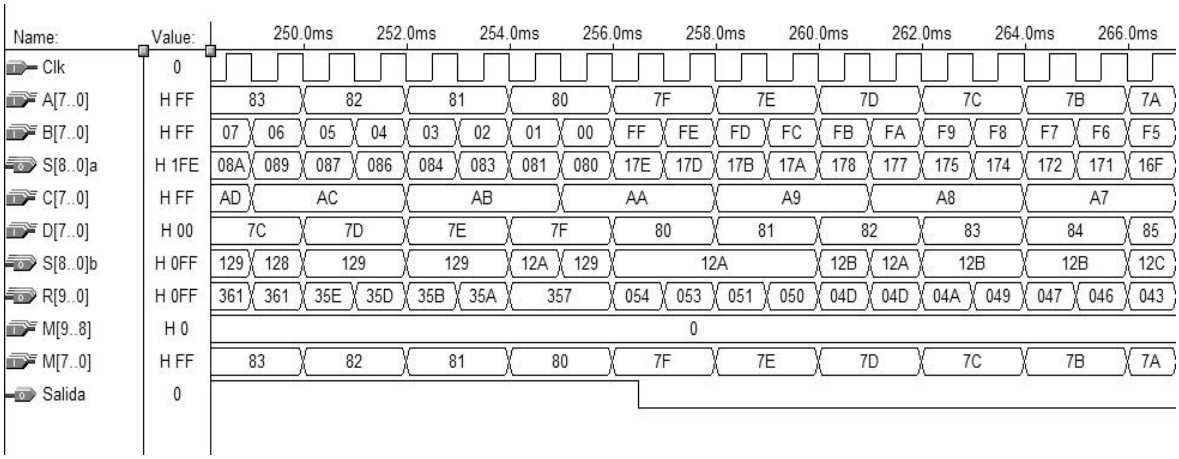


Figura 2.13c. Respuesta del functor lógico para las mismas combinaciones binarias en las entradas y en el umbral, pero en el periodo 248 ms a 268 ms.

La figura (2.13c) muestra los detalles de comportamiento de la salida con respecto a la señal de umbral, M(7..0), y el resultado de la resta, R(9..0). En este periodo se observa como R(9..0) cambia su valor, de 357 hexadecimal a 054 hexadecimal, el cual, al compararse con M(7..0), 7F, hace que salida, nuevamente previo retraso de 500 µS, vaya de 1 a 0. Obsérvese como salida cambia cuando el flanco del pulso sube, es decir, cuando el comparador toma el valor de R(9..0) para operarlo con el de M(7..0). Ahora se cumple la condición de que salida vale 0 cuando M(7..0) > R(9..0), ocurre el caso "cuando mucho 7F de 054" (operador Nor), esta situación ocurre dado que la suma de las entradas excitadoras, S(8..0)a = 17E, es mayor que la suma de las inhibitorias, S(8..0)b = 12A, y de acuerdo con el capitulo primero, hay un 0 cuando las entradas excitadoras no están inhibidas, pero el umbral es mayor al resultado de la resta.

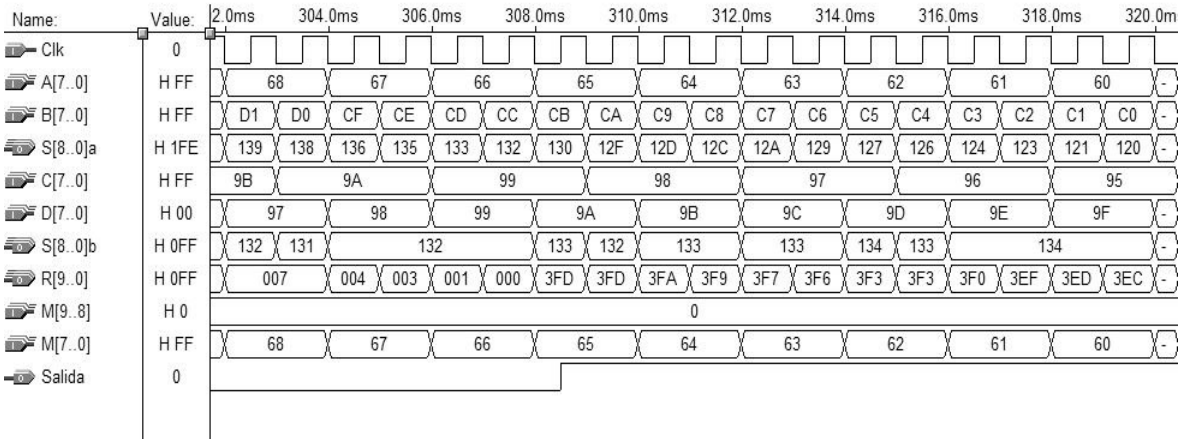


Figura 2.13d. Respuesta del functor lógico para las mismas combinaciones binarias en las entradas y en el umbral, pero en el periodo 302 ms a 320 ms.

La figura (2.13d) muestra los detalles del comportamiento de la salida con respecto de la señal de umbral, M(7..0), y el resultado de la resta, R(9..0). Nuevamente se observa el cambio de R(9..0) de 000 hexadecimal a 3FD hexadecimal, lo cual hace que salida vaya de 0 a 1, previo retardo de 500 µS, al compararse con M(7..0), cuyo valor es 65 hexadecimal. Cumpliéndose la condición de que salida vale 1 cuando R(7..0) ≥ M(7..0), es decir, "cuando mucho 65 de 3FD" (operador Nor), este caso se cumple debido a que la suma de las entradas excitadoras, S(8..0)a, 130 hexadecimal, es menor que la suma de las inhibitorias S(8..0)b, 133 hexadecimal. Las entradas inhibitorias "inhiben" a las excitadoras, por ello, como se ha descrito en el capitulo 1, hay un 1 a la salida cuando están inhibidas las entradas excitadoras, y el resultado de la resta es mayor al umbral.

Cabe señalar que en la simulación, cada una de las entradas A(7..0), B(7..0) C(7..0) y D(7..0), que entregan datos a los sumadores, y M(7..0), valor del umbral, varían sus cantidades en un cierto periodo, distinto para cada una, lo que produce diversos resultados, tanto en las sumas parciales, S(8..0)a y S(8..0)b, como en la resta R(9..0), y por consecuencia, en la salida. Estas combinaciones de datos en las entradas las permite el simulador del paquete MAX + plus II.

Las siguientes figuras, (2.14) y subsecuentes, presentan la función del operador generalizado estocástico, cuyo diagrama representativo se muestra en la figura (2. 12), pero ahora con una señal cuadrada como umbral.

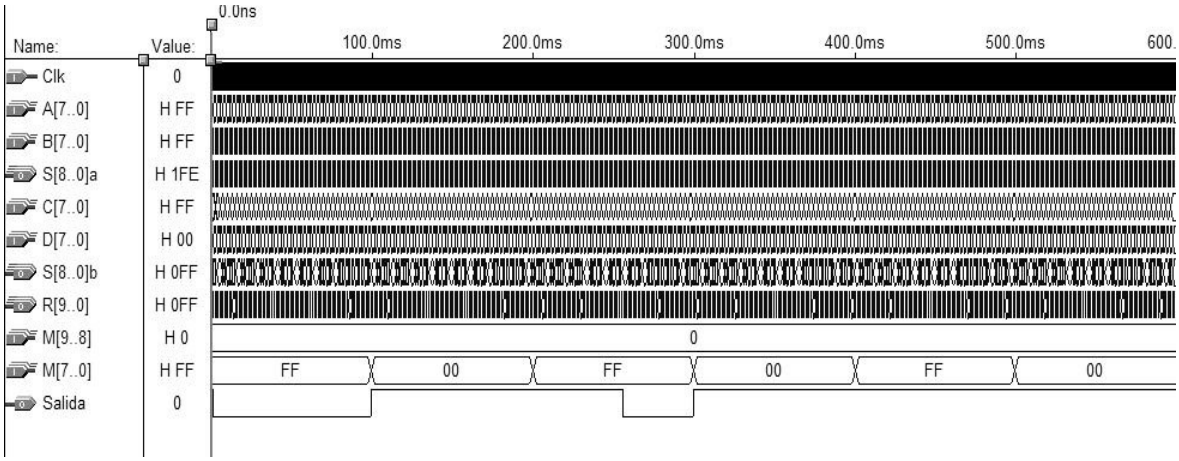


Figura 2.14. Respuesta del operador generalizado con un patrón estocástico a combinaciones binarias en las entradas, y una señal de umbral tipo cuadrada.

La figura (2.14a) muestra, en esencia, la misma operación del functor, sumar las entradas A(7..0), que varía cada 2 mS, y B(7..0), que cambia cada milisegundo, y las entradas C(7..0), que toma valores distintos cada 3 mS, y D(7..0), que varía cada 2 mS. Estos resultados se restan, observándose el dato en R(9..0), cuyo cambio está en función de B(7..0), que es la que opera en un periodo menor. El valor de R(9..0) es comparado con la señal de umbral, M(7..0), que para esta ocasión, se trata de una señal cuadrada, cuyo periodo de variación es de 100 mS. Se observa que salida toma el valor de 1, con el correspondiente retraso de 500 microsegundos, cuando el comparador toma el dato de R(9..0) con una cantidad 0FF en hexadecimal, operándolo con el umbral, M(7..0), también en FF hexadecimal. Cumpliéndose la condición de que salida es 1 cuando R(9..0) ≥ M(7..0), esto es, "al menos FF de FF" (operador And), dado que la suma de las entradas excitadoras, S(8..0)a = 1FE hexadecimal, es mayor que la suma de las inhibitorias, S(8..0)b = 0FF, y, de acuerdo con el capitulo primero, hay un 1 a la salida cuando las entradas excitadoras no están inhibidas, y el resultado de la resta es igual al umbral.

Salida regresa nuevamente a 0, puesto el resultado de R(9..0) cambia, ahora con un valor de 0FE hexadecimal, manteniéndose el FF hexadecimal de M(7..0). Esto es, "cuando mucho FF de 0FE" (operador Nor), este caso se cumple debido a que la suma de las entradas excitadoras, S(8..0)a, 1FD hexadecimal, es mayor que la suma de las inhibitorias S(8..0)b, 0FF hexadecimal. Para este caso, las entradas inhibitorias no "inhiben" a las excitadoras, por ello, como se ha descrito en el capitulo 1, hay un 0 a la salida cuando no están inhibidas las entradas excitadoras, y el resultado de la resta es menor al umbral.

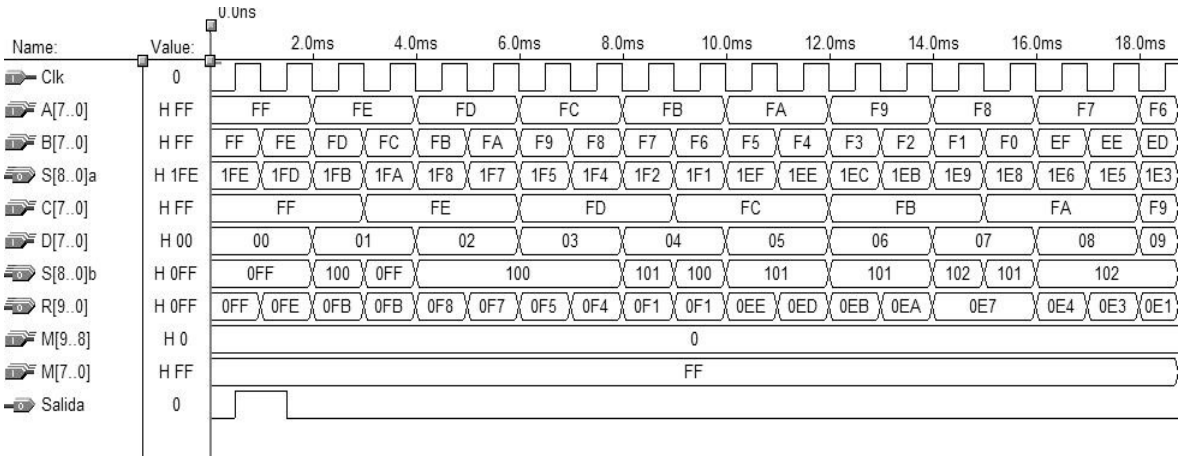


Figura 2.14a. Detalle de la operación del functor lógico para el periodo de 0 ms a 18 ms.

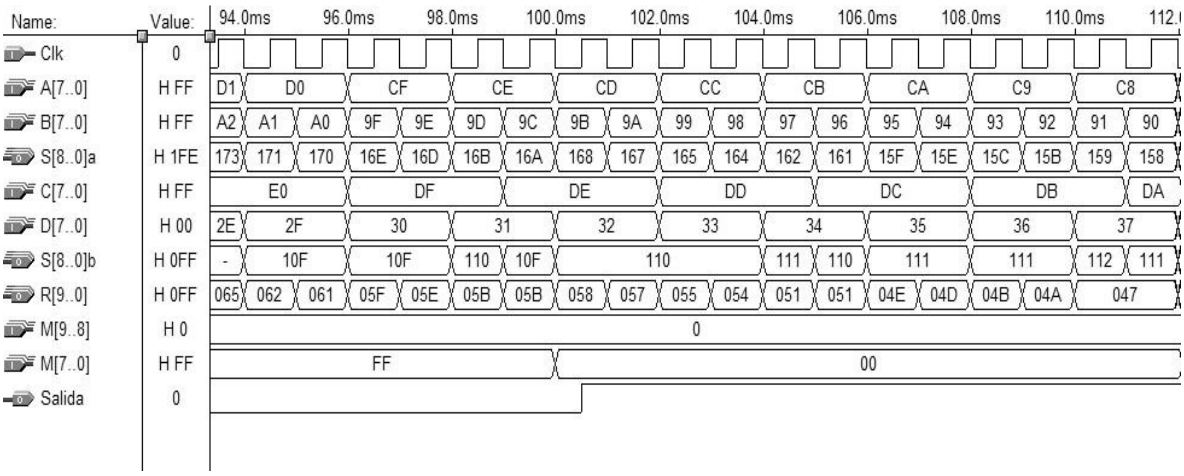


Figura 2.14b. Detalle de la operación del functor lógico para el periodo de 94 ms a 112 ms.

La figura (2.14b) muestra la misma operación del functor, sumando las entradas A(7..0) y B(7..0) y las entradas C(7..0) y D(7..0). Estos resultados se restan, observándose el dato en R(9..0), cuyo valor es comparado con la señal de umbral, M(7..0). Se observa que salida toma el valor de 1, con el correspondiente retraso de 500 microsegundos, cuando viene el flanco de subida del pulso de reloj y el comparador toma el dato de R(9..0) con una cantidad 058 en hexadecimal, operándolo con el umbral, M(7..0), ahora en 00 hexadecimal. Cumpliéndose la condición de que salida es 1 cuando R(9..0) ≥ M(7..0), es decir, "al menos 00 de 058" (operador Or), dado que la suma de las entradas excitadoras, S(8..0)a = 168 hexadecimal, es mayor que la suma de las inhibitorias, S(8..0)b = 110, y, de acuerdo con el capitulo primero, hay un 1 a la salida cuando las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

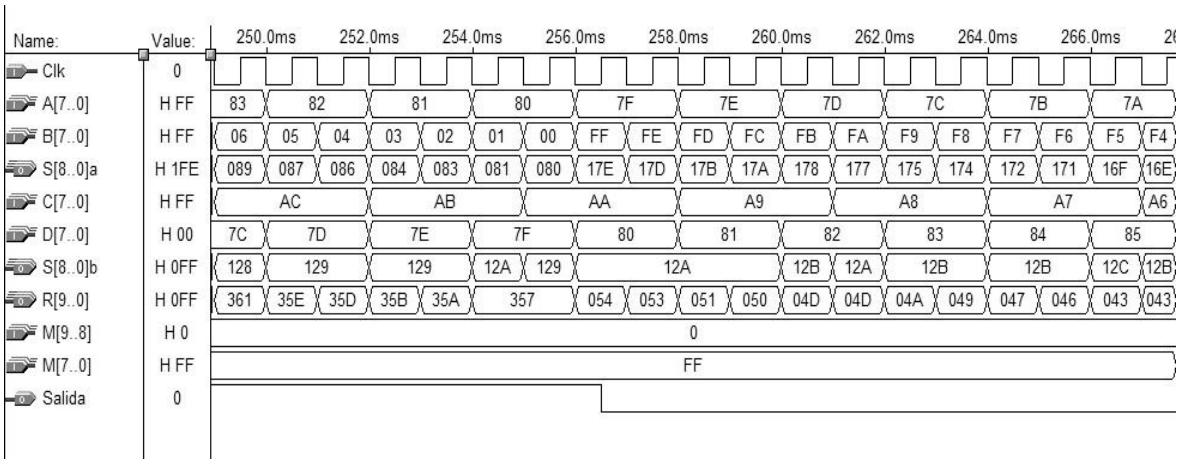


Figura 2.14c. Detalle de la operación del functor lógico para el periodo de 248 ms a 268 ms.

La figura (2.14c) muestra la misma operación del functor, sumando las entradas A(7..0) y B(7..0) y las entradas C(7..0) y D(7..0). Estos resultados se restan, observándose el dato en R(9..0), cuyo valor es comparado con la señal de umbral, M(7..0). Se observa que salida toma el valor de 0, con el correspondiente retraso de 500 microsegundos, cuando viene el flanco de subida del pulso de reloj y el comparador toma el dato de R(9..0) con una cantidad 054 en hexadecimal, operándolo con el umbral, M(7..0), ahora en FF hexadecimal. Cumpliéndose la condición de que salida es 0 cuando R(9..0) < M(7..0), es decir, "cuando mucho FF de 054" (operador Nor), este caso se cumple debido a que la suma de las entradas excitadoras, S(8..0)a, 17E hexadecimal, es mayor que la suma de las inhibitorias S(8..0)b, 12A hexadecimal. Ahora, las entradas inhibitorias no afectan a las excitadoras, por ello, como se ha descrito en el capitulo 1, hay un 0 a la salida cuando no están inhibidas las entradas excitadoras, y el resultado de la resta es menor al umbral.

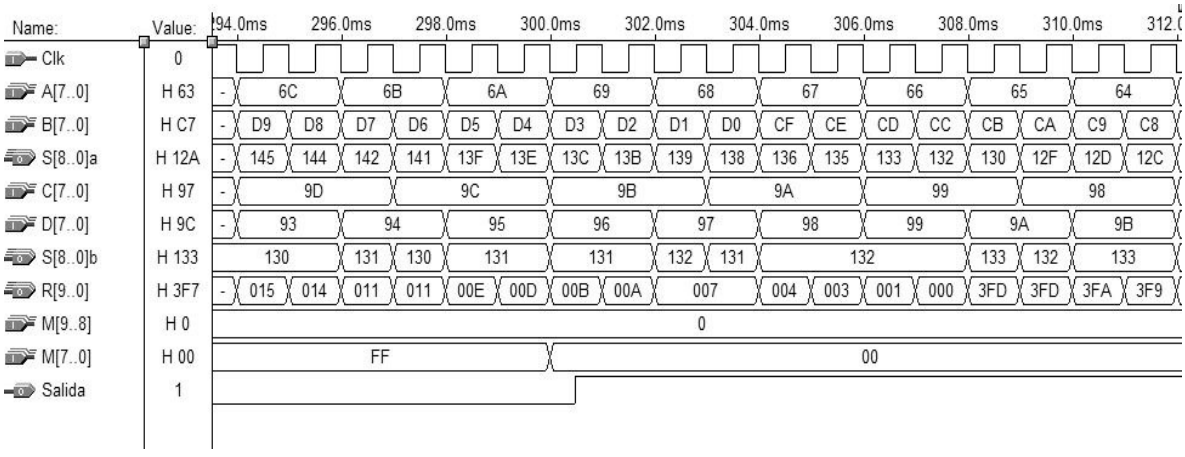


Figura 2.14d. Detalle de la operación del functor lógico para el periodo de 294 ms a 312 ms.

La figura (2.14d) es similar, se observa la suma de las entradas A(7..0) y B(7..0) y las entradas C(7..0) y D(7..0), cuyos resultados se restan, observándose el dato en R(9..0). Este valor es comparado con la señal de umbral, M(7..0). Se observa que salida toma el valor de 1, con el correspondiente retraso de 500 microsegundos, cuando viene el flanco de subida del pulso de reloj y el comparador toma el dato de R(9..0) con una cantidad 00B en hexadecimal, operándolo con el umbral, M(7..0), ahora en 00 hexadecimal. Cumpliéndose la condición de que salida es 1 cuando R(9..0) ≥ M(7..0), es decir, "al menos 00 de 00B" (operador Or), dado que la suma de las entradas excitadoras, S(8..0)a = 13C hexadecimal, es mayor que la suma de las inhibitorias, S(8..0)b = 131, y, de acuerdo con el capitulo primero, hay un 1 a la salida cuando las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

En esta simulación, también, cada una de las entradas A(7..0), B(7..0) C(7..0) y D(7..0), que entregan datos a los sumadores, y M(7..0), valor del umbral, variaron sus cantidades en un cierto periodo, distinto para cada una, lo que produjo diversos resultados, tanto en las sumas parciales, S(8..0)a y S(8..0)b, como en la resta R(9..0), y por consecuencia, en la salida. Esto lo permite el simulador del paquete MAX + plus II.

El diseño e implementación del operador generalizado, utilizando la plataforma de desarrollo de Altera y la paquetería de diseño MAX + plus II, ha permitido comprobar y corroborar las bases y planteamientos descritos en el primer capitulo de este trabajo, toda vez que el proceso no ha sido complicado, considerando que el diseño está basado en compuertas básicas. Esta es una ventaja que ofrece el software de Altera, pues permite trabajar con esquemáticos para ir conformando el circuito final, como se vio a lo largo de este capitulo. Además de ello, permite observar el comportamiento de cada uno de los circuitos parciales a través de su visualizador de salidas, que son con las cuales se pudieron observar las respuestas de cada uno de los bloques, y la salida, elemento principal para comprobar que el operador generalizado diseñado trabaja como se planteó en el primer capitulo.

Esta plataforma de desarrollo es una buena opción para generar este circuito, quizá, una desventaja es que solo trabaja con señales digitales, es decir, datos binarios entrados al sistema en grupos de bits o bits individuales. Además de esto, los tiempos que utilizan cada una de las componentes del circuito genera un tiempo de retardo o propagación, haciendo que las salidas se retrasen al presentar el dato correspondiente. Lo anterior se puede comprobar en este desarrollo, ya que se debió incorporar un elemento retardador, un grupo de 10 flip-flop's tipo D, para adecuar los datos antes de entrar a comparador, evitando con ello, en las graficas, los retardos, o "glitches", que impiden tener datos correctos en las salidas.

Por lo anterior, a continuación se plantea la opción de desarrollar un operador generalizado, ahora utilizando un microcontrolador, que de inicio permita trabajar también con datos analógicos, y a velocidades de operación mucho mas altas que

las involucradas en este primer diseño, y desechando en lo posible, los retardos mencionados, que pueden afectar las salidas.

Capitulo 3

3.1.- Implantación Electrónica del Operador generalizado utilizando el microcontrolador 16F877A y la plataforma de desarrollo MPLab de Microchip

Se diseñó un sistema electrónico utilizando un microcontrolador, el PIC 16F877A de Microchip, para emular al operador generalizado.

Se eligió este dispositivo en particular debido a que cuenta con la opción, en su arquitectura interna, de un convertidor analógico-digital, lo que permite involucrar señales analógicas en esta aplicación. Esta fue una de las características mas importantes al momento de seleccionarlo, ya que por lo demás, la mayoría de los microcontroladores del mercado cuenta con una arquitectura que se puede adecuar a las necesidades y requerimientos de la aplicación. Utilizar uno mas sencillo, implicaría mantener su uso en el campo de las señales digitales, como lo hace la tarjeta de desarrollo de Altera y su ambiente de trabajo MAX + plus II. Para abarcar el campo analógico, y ampliar el uso de este operador, se necesita entonces agregar un elemento adicional, el convertidor analógico-digital. Con lo anterior, el microcontrolador idóneo para esta aplicación es el ya comentado.

Existen otras compañías que manufacturan microcontroladores con convertidores, pero algunas sobresalen de otras debido a las facilidades que otorgan en cuanto a la obtención del software de desarrollo, así como de circuitos para la grabación de estos dispositivos, Microchip es una de ellas, quien a través de su pagina de Internet, provee estas herramientas de trabajo.

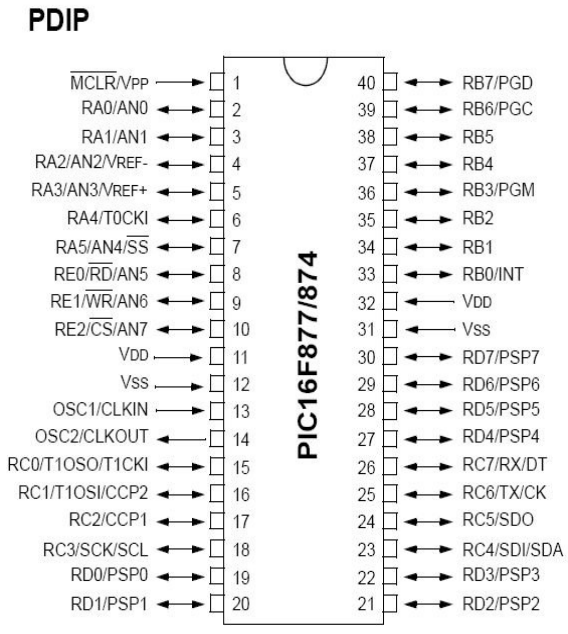


Figura 3.1. Diagrama de pines para el microcontrolador PIC16F877A.

Este microcontrolador seleccionado, figura (3.1), cuenta con 5 puertos, A, B, C, D y E, de los cuales, para esta aplicación, se utilizan el A, el B, el C y el D. Según la arquitectura de este microcontrolador, el puerto A puede ser operado como entrada o salida digital de 8 bits, o como 8 entradas analógicas, las cuales pueden ser digitalizadas aplicando el convertidor analógico-digital con el que cuenta. En este caso particular, este puerto es usado para entradas analógicas, utilizando 4 de ellas, además del pin 8, bit 5, usado como salida del operador para monitorear su comportamiento. El puerto B, de 8 bits, es configurado como entrada, a través de la cual se aplica la señal de umbral, y el puerto C, también de 8 bits, que puede ser manejado como entrada/salida, o, de manera separada, controlador para

diferentes funciones periféricas, en esta aplicación, se encarga de ilustrar las operaciones de conversión analógico-digital, utilizando para ello el bit 4, pin 23, del puerto C, además, se utiliza el bit 5, pin 24, para indicar que el dispositivo está encendido. De acuerdo con lo anterior, se utilizan los pines 2 a 5, que son el puerto A como entradas analógicas, y el pin 8, bit 5, para monitorear la salida del sistema. Los pines 33 a 40, que son el puerto B como entrada de control de umbral, y el bit 3 del puerto D, pin 22, muestra la actividad del dispositivo. La tabla siguiente, (3.1), resume el funcionamiento de todos los pines del microcontrolador PIC 16F877A.

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Tabla 3.1. Resumen de la operación de los pines del microcontrolador PIC 16F877.

						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Tabla 3.1. Resumen de la operación de los pines del microcontrolador PIC 16F877. (Continuación).

Al igual que en el desarrollo anterior, la aplicación se basa en el comportamiento de la neurona tipo Mc Cullock y Pitts, descrito en el capitulo 1. El diagrama de flujo de la figura (3.2) describe el comportamiento del microcontrolador para realizar las operaciones de suma entre las señales excitadoras y la suma entre las señales inhibitorias, para este caso, se utiliza el puerto A, configurado como entradas analógicas, a través de las cuales se aplican estas señales, excitadoras e inhibitorias. La resta entre estos resultados, que se realiza dentro del dispositivo, y la comparación entre el resultado de la resta y el generador de umbral, este último se ingresa a través del puerto B del dispositivo, así como el manejo de la salida resultante para ser sacada a través del puerto A, y visualizada con algún dispositivo de graficación. Se maneja el puerto C para mostrar la actividad de conversión analógico- digital, cuando esta se realiza dentro del dispositivo, por

medio de uno de sus bits, el número 4. El bit 3 del puerto D indica la operación del dispositivo, y el bit 5 indica que este está encendido.

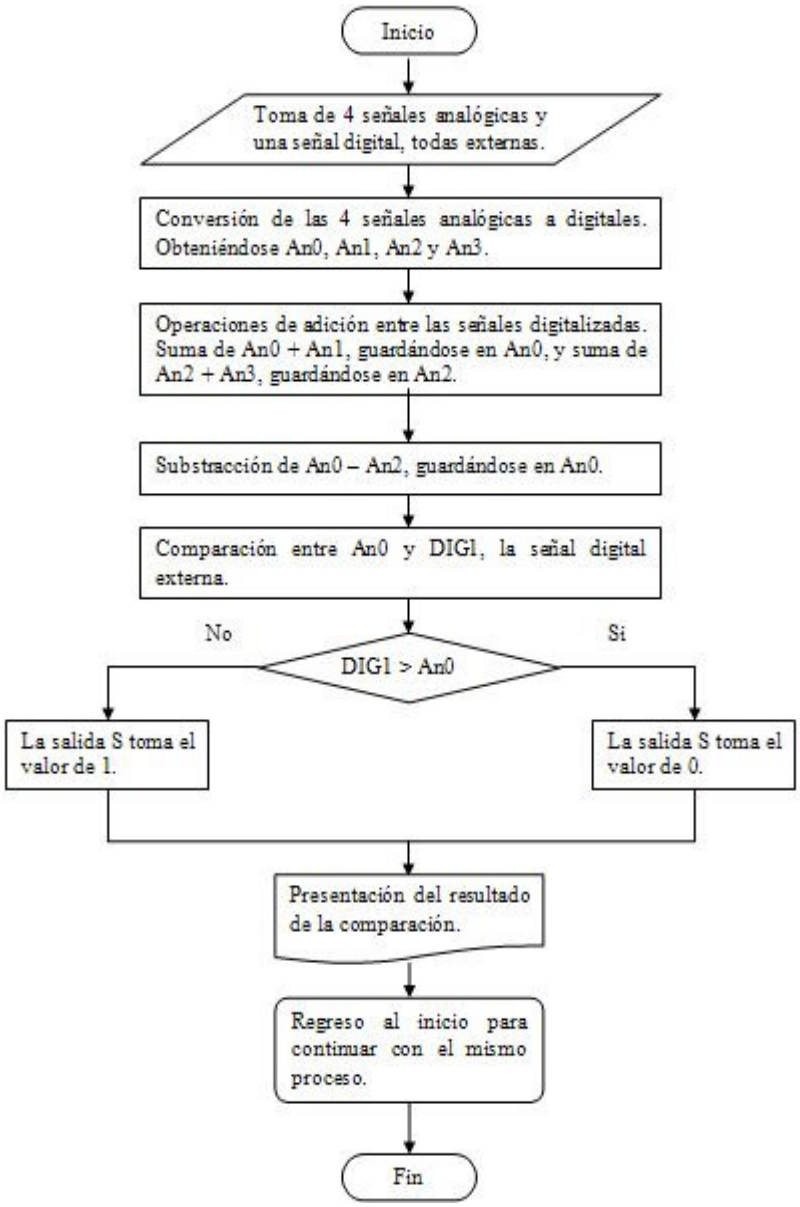


Figura 3.2. Diagrama de flujo descriptivo de la operación del microcontrolador como operador generalizado.

A continuación se muestra el listado del algoritmo para operar el microcontrolador, en lenguaje ensamblador. Este algoritmo describe, de forma detallada, la activación y configuración de los diferentes componentes internos que permiten el desarrollo de la aplicación, así como también, las rutinas y operaciones lógico-aritméticas necesarias para este objetivo. Como se señaló en párrafos anteriores, este dispositivo cuenta con un convertidor analógico digital integrado, por lo cual

se especifica, en el algoritmo, la activación del mismo desde donde se obtendrán los valores de las señales excitadoras e inhibitorias que, junto con la señal de umbral generada digitalmente y entrada a través del puerto B, hacen funcionar al operador generalizado, además de aquellas que muestran la actividad del dispositivo y el movimiento de la información, utilizando pines de los puertos C, D y A de dicho dispositivo.

```
;-------------------------------------------------------------------------------
;Objetivo
;Programa que usa el conversor A/D que contiene el microprocesador pic16F877A.
;Se utilizarán cuatro entradas analógicas AN0, AN1, AN2, AN3 y se convertirán
;en señales digitales de 8 bits, donde se realizaran las siguientes
;operaciones: (AN0+AN1)-(AN2+AN3), y el resultado de estas operaciones se
;comparará con una entrada digital de 8 bits y si el resultado de las señales
;análogas es mayor o igual a la señal digital se presentará un "1" en un pin de
;salida. Si en el resultado de la comparación la señal digital es más grande se
;presentará un "0" a la salida.
;-------------------------------------------------------------------------------
LIST P=16f877
INCLUDE "P16f877A.INC"
;_CONFIG _CP_OFF & _PWRTE_ON & WDT_OFF & HS_OSC
;-------------------------------------------------------------------------------
; Definición de registros y variables utilizadas
;-------------------------------------------------------------------------------
status  equ  03   ;Habilitación del registro STATUS
ptoa    equ  05   ;Habilitación del puerto A
ptob    equ  06   ;Habilitación del puerto B
ptoc    equ  07   ;Habilitación del puerto C
ptod    equ  08   ;Habilitación del puerto D
pclath  equ  0a   ;Habilitación del registro PCLATH
adresh  equ  1e   ;Habilitación del registro de resultado de conversión, parte alta
adcon0  equ  1f   ;Habilitación del registro de operación de la conversión
trisa   equ  85   ;Habilitación del registro de operación del puerto A
trisb   equ  86   ;Habilitación del registro de operación del puerto B
trisc   equ  87   ;Habilitación del registro de operación del puerto C
trisd   equ  88   ;Habilitación del registro de operación del puerto D
adresl  equ  9e   ;Habilitación del registro de resultado de conversión, parte baja
adcon1  equ  9f   ;Habilitación del registro de control de entradas analógicas
rp0     equ  5    ;Posición del bit dentro del registro STATUS
rp1     equ  6    ;Posición del bit dentro del registro STATUS
irp     equ  7    ;Posición del bit dentro del registro STATUS
z       equ  2    ;Posición del bit dentro del registro STATUS
c       equ  0    ;Posición del bit dentro del registro STATUS
go      equ  2    ;Posición del bit dentro del registro ADCON0
AN0     equ  22   ;Definición de variable sobre un registro
AN1     equ  23   ;Definición de variable sobre un registro
AN2     equ  24   ;Definición de variable sobre un registro
AN3     equ  25   ;Definición de variable sobre un registro
DIG1    equ  26   ;Definición de variable sobre un registro
selan   equ  27   ;Definición de variable sobre un registro
carr1   equ  28   ;Definición de variable sobre un registro
carr2   equ  29   ;Definición de variable sobre un registro
PDel0   equ  2A   ;Definición de variable sobre un registro
```

En esta primera parte del listado del programa, se establece el objetivo del algoritmo, particularmente, el de tomar 4 señales analógicas, convertirlas a digitales, sumar las primeras dos y las dos siguientes, cuyos resultados serán restados, procediendo la comparación de este resultado con el dato digital entrado por otro de los puertos del microcontrolador. Posterior al objetivo, se hace alusión, o se declara, el numero del dispositivo con el cual se ha de trabajar, además de su modelo especifico, con lo cual el paquete de programación toma las condiciones y características necesarias de este dispositivo, para la posterior compilación y depuración del programa, además de otras consideraciones, tales como el tamaño y tipo de memoria de solo lectura con la que cuenta. Se utiliza un (;), punto y coma, en el listado, para que el compilador no tome en cuenta lo que sigue después de estos caracteres. De esta forma se pueden escribir comentarios en el programa para explicar mejor cada una de las líneas, sin provocar errores al momento de ser compilado.

Posterior a esto, se invocan, o se "llaman", a los registros necesarios con los cuales se va a trabajar. Muchos de estos registros son básicos para el correcto funcionamiento del dispositivo, puesto que cada uno de ellos hace operar los diferentes bloques internos, de los cuales el microcontrolador está constituido. Además de ello, se seleccionan registros vacíos que servirán como lugar donde guardar temporalmente datos intermedios en las operaciones a realizar. Esta es una especie de definición de "variables" con las que el algoritmo se auxilia en sus operaciones internas. También se consideran ciertos bits de los registros necesarios, sobre los cuales el programa tendrá una cierta vigilancia, ya que dependiendo del valor que tomen, se realizará una u otra rutina interna. Al final de cada línea también se comenta cada uno de estos registros.

```
;-------------------------------------------------------------------------------
; Inicio del Programa
;-------------------------------------------------------------------------------
        org     0x00            ;Se declara el origen del programa
        goto    inicio          ;Inicio
inicio  bsf     status,rp0      ;Se pone a "1" el bit RP0 del registro STATUS
        movlw   0x1f            ;Se carga en el registro de trabajo "w" el dato 1F
        movwf   trisa           ;y después al puerto A, para definir las entradas como
                                ;Analógicas
        movlw   b'11111111'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   trisb           ;para definir al puerto B como entrada Digital de 8 bits
        movlw   b'10000111'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   trisc           ;para definir a RC7=Rx, RC6=Tx, RC5,RC4,RC3 como
                                ;salidas para verificación Led
        movlw   b'00000000'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   trisd           ;para definir al puerto D como salidas

        movlw   b'00000010'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   adcon1          ;y luego en adcon1, que sitúa datos en ADRESH y 2 bits en
                                ;ADRESL,
                                ;Selecciona Vref interno (+5V y GND)
                                ;ADFM = 0: Justificado a la izquierda
                                ;PCFG<3..0> = 0010 -->   AN7 = Digital, AN6= Digital,
                                ;AN5= Digital
```

```
                                ;AN4 = Analógico, AN3 = Analógico, AN2= Analógico
                                ;AN1 = Analógico, AN0= Analógico
                                ;VREF+ = Vdd (interno), VREF- = GND (interno)

        bcf     status,rp0      ;Se pone a cero el bit rp0 del registro STATUS

        bsf     ptoc,5          ;Se pone a 1 el bit 5 del puerto C para encender LED
        bcf     ptoc,4          ;Se pone a cero el bit 4 para apagar LED
        bcf     ptod,3          ;Se pone a cero el bit para apagar LED

sigue   movlw   b'10000001'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   adcon0          ;Selecciona canal Analógico 0 y reloj de ADC=osc/32
        movlw   b'00000001'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   selan           ;que indica se guardará la conversión en An0
        call    conver          ;Llama rutina de conversión AD
        movlw   b'10001001'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   adcon0          ;Selecciona canal Analógico 1 y reloj de ADC=osc/32
        movlw   b'00000010'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   selan           ;que indica se guardará la conversión en An1
        call    conver          ;Llama rutina de conversión AD
        movlw   b'10010001'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   adcon0          ;Selecciona canal Analógico 2 y reloj de ADC osc/32
        movlw   b'00000100'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   selan           ;que indica se guardará la conversión en An2
        call    conver          ;Llama rutina de conversión AD
        movlw   b'10011001'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   adcon0          ;Selecciona canal Analógico 3 y reloj de ADC osc/32
        movlw   b'00001000'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   selan           ;que indica se guardará la conversión en An3
        call    conver          ;Llama rutina de conversión AD
        movf    ptob,0          ;Se obtiene el numero digital del puerto B y
        movwf   DIG1            ;se guarda en DIG1
```

En esta parte se muestra el inicio del programa. En la primera línea se indica a partir de que localidad se ha de grabar en la memoria del dispositivo, saltando hacia la primera instrucción. En este bloque del listado se configuran los puertos, para definirlos como entradas o como salidas, definiéndose también en que parte de la memoria están dispuestos sus registros de configuración. También se invoca al registro que se encarga de definir cuantas señales analógicas existen, en este caso, son las entradas AN0, AN1, AN2 y AN3, con esto, el registro *adcon1* indica al convertidor que tiene que operar en procesos de conversión, además de los rangos dentro de los cuales ha de trabajar. El registro STATUS es el que mas se utiliza, ya que con el se opera la mayoría de las funciones. Se observa como cambia sus valores para ir saltando entre los espacios de la memoria que contiene todos los registros, invocados al principio del listado, y con esto, presentar los datos en los puertos correspondientes, o realizar alguna otra operación. Se utilizan los bits 4 y 5 del puerto C, y el 3 del puerto D, para indicar que hay una conversión analógico-digital, que el dispositivo está encendido, y que está en operación, respectivamente. La rutina indica entonces que el dispositivo esta encendido, pero aun no realiza alguna operación. Este bloque contiene la rutina que se encarga de realizar las conversiones analógico-digitales, a través del registro *adcon0*. Este selecciona cada uno de los 4 canales. Se trabaja con la variable auxiliar *selan*, la

cual controla los diferentes lugares, o registros, donde se han de guardar los datos posteriores a la conversión, es decir, ya digitalizada la señal. Son cuatro señales, por lo cual, son cuatro espacios destinados para esto. Por ultimo, se toman los datos que están presentes en el puerto B, guardándose en el registro auxiliar DIG1. Este es el de umbral, con el cual se compara el resultado de la resta, del operador generalizado.

```
;-------------------------------------------------------------------------------
; Inicia el procesamiento digital de las señales
;-------------------------------------------------------------------------------
        movf    AN1,0           ;Se carga el valor en AN1
        addwf   AN0,1           ;Se suma AN0 y AN1 y se guarda en AN0
        btfss   status,c        ;Revisa acarreo en STATUS, brinca si es 1
        goto    sum0            ;No hay acarreo
        goto    sum1            ;Si hay acarreo
sum0    bcf     carr1,0         ;Se pone un cero, en bit 0 de carr1, para saber que no hay
                                ; acarreo
        goto    continu         ;ir a etiqueta
sum1    bsf     carr1,0         ;Se pone un 1, en bit 0 de carr1, para saber que hay
                                ; acarreo
continu movf    AN3,0           ;Se carga valor el AN3
        addwf   AN2,1           ;Suma AN2 y AN3 y se guarda en AN2
        btfss   status,c        ;Revisa acarreo en STATUS, brinca si es 1
        goto    sum2            ;No hay acarreo
        goto    sum3            ;Si hay acarreo
sum2    bcf     carr2,0         ;Se pone un cero, en bit 0 de carr2, para saber que no hay
                                ;acarreo
        goto    conti2          ;ir a etiqueta
sum3    bsf     carr2,0         ;Se pone un uno, en bit 0 de carr2, para saber que si hay
                                ;acarreo
conti2  movf    carr2,0         ;Se carga valor en carr2
        subwf   carr1,1         ;Se restan los acarreos y se guardan en carr1
        btfss   status,c        ;Revisa el acarreo, en STATUS, para saber si el segundo
                                ;digito es de 9 bits y el primero no, por lo tanto en la resta
                                ;Se obtendrá un número negativo y se debe enviar un "0".
                                ;Brinca si es 1.
        goto    negat           ;ir a la etiqueta si saldrá negativo
        btfss   carr1,0         ;Revisa la resta, bit 0 de carr1, de los acarreos para saber
                                ;como se debe realizar la resta. Brinca si es 1.
        goto    normal          ;Resta de 8bits-8bits
        goto    complej         ;Resta de 9bits-8bits
normal  movf    AN2,0           ;Se carga valor en AN2
        subwf   AN0,1           ;Resta final(AN0-AN2 guardándose en AN0)
        btfsc   status,c        ;Revisa si la resta da negativo, en STATUS, brinca si es
                                ;cero.
        goto    compara         ;Si no es negativa, se hace la comparación con DIG1. Ir a
                                ;etiqueta.
        goto    negat           ;Si es negativa, ir a etiqueta.
complejmovf     AN2,0           ;Se carga valor en AN2
        subwf   AN0,1           ;Resta final(AN0-AN2 guardándose en AN0)
        btfsc   status,c        ;Revisa si la resta da resultado de 9 bits, en STATUS,
                                ;brinca si es cero.
        goto    posit           ;Si es de 9 bits, la comparación envía "1" a la salida.
                                ;Si no, se hace la comparación con DIG1.
compara movf    DIG1,0          ;Se carga valor en DIG1.
```

```
subwf   AN0,0       ;Resta (DIG1-AN0 guardándose en AN0)
btfsc   status,c    ;Revisa acarreo en STATUS, brinca si es cero.
goto    posit       ;ir a etiqueta.
goto    negat       ;ir a etiqueta.
```

En esta parte del programa se realizan ya las operaciones lógico-aritméticas con los datos digitalizados de las señales analógicas. Se suman los dos primeros datos, quedando almacenados en AN0. Dependiendo si la operación presenta algún acarreo, se brinca hacia una u otra pequeña rutina, para indicar a la variable auxiliar *carr1* que guarde esa condición. Esto mismo sucede con los dos últimos datos, guardándose el resultado en AN2, y considerando la condición de si hay o no acarreo, para almacenarse en *carr2*. Posteriormente, se procede a la substracción de ambos resultados de las sumas, empezando por los acarreos, guardándose el resultado en *carr1*. Auxiliándose del bit C, Carry, del registro STATUS, revisa si la operación de resta produjo un acarreo, esto es, si existe un 0; si no lo hay, es decir si hay un 1, entonces se va a una rutina, *negat*, indicando que el dato es negativo. De lo contrario, continua con la substracción de los registros auxiliares AN2 y AN0. Cabe señalar que un acarreo en una substracción se representa con un cero en C del registro STATUS. Continuando con el proceso, el programa verifica si ambos datos, AN2 y AN0, son de 8 bits o si AN0 es de 9 bits, auxiliándose del bit cero del registro *carr1*; si hay un 1, se salta la siguiente instrucción, de lo contrario la ejecuta. Basado en esto, el programa se va hacia una u otra rutina, para realizar la substracción. Si AN0 es de 8 bits, se realiza la substracción y se revisa el resultado, si es un numero negativo, si C = 0, brinca hacia una rutina, de lo contrario, si C = 1, se compara con el valor DIG1, siempre observando el bit C del registro STATUS. Algo similar sucede cuando AN0 es de 9 bits, se realiza la substracción y se revisa el resultado, si C = 1, no hay un acarreo, el tamaño de bits se conserva en el resultado de la substracción, no es necesaria la comparación, ya que siempre será mayor que DIG1, de no ser así, C = 0, viene la comparación entre ambos. Finalmente, la comparación entre el resultado de la resta y el valor de umbral DIG1, consiste en una substracción, y dependiendo si hay o no acarreo en STATUS, va a una u otra rutina.

```
;-------------------------------------------------------------------------------
; Transmisión del resultado
;-------------------------------------------------------------------------------
posit   bsf     PORTD,3     ;Pone a 1 el pin 3 del puerto D, enciende LED de
                            ; operación.
        bsf     PORTA,5     ;Pone a 1 el pin 5 del puerto A, enciende LED de
                            ;operación.
        bcf     PORTD,3     ;Pone a cero el pin 3 del puerto D, apaga LED de
                            ;operación.
        goto    sigue       ;ir a etiqueta.
negat   bsf     PORTD,3     ;Pone a 1 el pin 3 del puerto D, enciende LED de
                            ;operación.
        bcf     PORTA,5     ;Pone a 0 el pin 5 del puerto A, apaga LED de
                            ;operación.
        bcf     PORTD,3     ;Pone a cero el pin 3 del puerto D, apaga LED de
                            ;operación.
        goto    sigue       ;ir a etiqueta.
```

En esta parte del listado se observa la forma en que se envía el dato a la salida. Como se muestra en el bloque anterior, la rutina *posit* utiliza el bit 3 del puerto D para indicar que está operando el dispositivo. El bit 5 del puerto A presenta la salida del sistema, esto es, toma el valor de 1 si AN0 ≥ DIG1. El bit 3 del puerto D se pone a cero, para indicar que termina la operación de sacar el dato por el bit 5 del puerto A. La rutina *negat* también utiliza los bits de los puertos ya mencionados. El bit 3 del puerto D indica que inicia la operación de sacar el dato, y el bit 5 del puerto A presenta la salida del sistema, ahora toma el valor de cero, pues AN0 < DIG1. Nuevamente el bit 3 del puerto D se pone a cero para indicar que la operación de mostrar el dato termina. Ambas rutinas terminan con un brinco hacia *sigue*, para continuar con el proceso de muestreo de las señales analógicas.

```
;-------------------------------------------------------------------------------
; Rutina de conversión A/D a 8 bits
; Devuelve la conversión en dos registros para cada entrada analógica, donde se
; despreciarán los dos últimos bits para poder tener 4 registros de 8 bits.
;-------------------------------------------------------------------------------
conver  bsf     ptoc,4          ;Pone a 1 el pin 4 del puerto C, enciende LED de
                                ;conversión.
        call    ADQUIS          ;Se da tiempo para tener una adquisición con un
                                ;valor exacto de conversión. (7.4us).
        bsf     adcon0,go       ;Pone a 1 el bit go de adcon0, inicia la conversión.
        call    TCONV           ;Tiempo de conversión AD (19.2us).

consu   btfsc   adcon0,go       ;Verifica que la conversión se haya realizado, en adcon0,
                                ;brinca si es cero.
        goto    consu           ;ir a etiqueta.
        btfsc   selan,0         ;Se prueba selan, si hay 1 se llama a rutina, brinca si es
                                ;cero.
        call    CAD1            ;ir a rutina.
        btfsc   selan,1         ;Se prueba selan, si hay 1 se llama a rutina, brinca si es
                                ;cero.
        call    CAD2            ;ir a rutina.
        btfsc   selan,2         ;Se prueba selan, si hay 1 se llama a rutina, brinca si es
                                ;cero.
        call    CAD3            ;ir a rutina.
        btfsc   selan,3         ;Se prueba selan, si hay 1 se llama a rutina, brinca si es
                                ;cero.
        call    CAD4            ;ir a rutina.
        bcf     ptoc,4          ;Pone a cero el pin 4 del puerto C, apaga LED de
                                ;conversión.
        Return
```

Esta parte del programa se encarga de convertir las señales analógicas en datos binarios. Para ello utiliza el bit 4 del puerto C, indicando que el proceso de conversión está en curso. Se llama a una rutina para dar tiempo a que se tomen los datos a digitalizarse, ADQUIS. Se activa el proceso de conversión, utilizando el registro *adcon0* con su bit *go*; si *go* = 1, hace un loop, de lo contrario, se salta esa línea del programa. Se llama a una rutina TCONV para dar tiempo a que se realice la conversión. Si la conversión terminó, se procede a verificar, en la variable auxiliar *selan*, en cual de las conversiones se encuentra el proceso. Así continua hasta completar las cuatro, poniendo a cero el bit 4 del puerto C, para indicar que

la conversión ha terminado. Cada vez que termina una conversión se brinca hacia una subrutina, que es la que se encarga de almacenar los datos en un registro específico. Como se ve a continuación.

```
;--------------------------------------------------------------------------------
; Dependiendo del canal seleccionado, se guardará la conversión en una variable
; distinta.
;--------------------------------------------------------------------------------
CAD1   movf    adresh,w      ;Se carga el valor de adresh en "w"
       movwf   AN0           ;y después a AN0.
       return
CAD2   movf    adresh,w      ;Se carga el valor de adresh en "w"
       movwf   AN1           ;y después a AN1.
       return
CAD3   movf    adresh,w      ;Se carga el valor de adresh en "w"
       movwf   AN2           ;y después a AN2.
       return
CAD4   movf    adresh,w      ;Se carga el valor de adresh en "w"
       movwf   AN3           ;y después a AN3.
       return
```

Esta parte muestra la forma en que son guardados los datos binarios resultantes de la conversión. Se trata de cuatro etiquetas que indican en cual de las cuatro variables auxiliares se han de guardar cada uno de los datos. Se toma el dato del registro que almacena temporalmente el resultado de la conversión, *adresh*, y se guarda en su registro correspondiente, previo intercambio de información entre el registro de trabajo "*w*". El proceso es el mismo hasta contar con los cuatro datos, con los cuales se procede a realizar las operaciones lógico-aritméticas ya descritas.

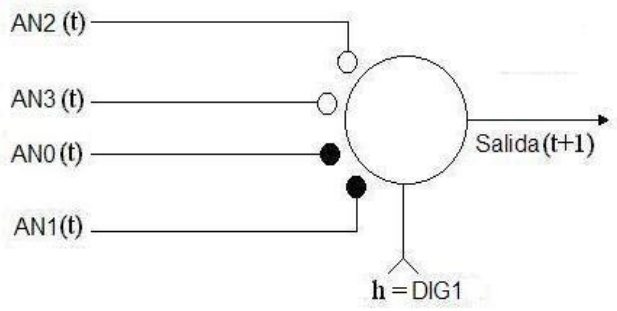
```
;-----------------------------------------------------------
ADQUIS  movlw    .8           ; 1 set numero de repetición
        movwf    PDel0        ; 1 |
PLoop0  clrwdt                ; 1 clear watchdog
        decfsz   PDel0, 1     ; 1 + (1) es el tiempo 0  ?
        goto     PLoop0       ; 2 no, loop
PDelL1  goto     PDelL2       ; 2 ciclos delay
PDelL2
        return                ; 2+2 Fin.
;-----------------------------------------------------------
;-----------------------------------------------------------
TCONV   movlw    .22          ; 1 set numero de repetición
        movwf    PDel0        ; 1 |
PLoop1  clrwdt                ; 1 clear watchdog
        decfsz   PDel0, 1     ; 1 + (1) es el tiempo 0  ?
        goto     PLoop1       ; 2 no, loop
PDelL3  goto     PDelL4       ; 2 ciclos delay
PDelL4  clrwdt                ; 1 ciclo delay
        return                ; 2+2 Fin.
;-----------------------------------------------------------
end
```

Esta parte muestra las rutinas de retardo, tanto para dar un tiempo de adquisición de la señal a convertir, ADQUIS, como para el proceso de conversión de dicha señal, TCONV. ADQUIS consiste en cargar un valor, .8, en el registro auxiliar de retardo PDel0, y se va decrementando. Si el valor de este registro es diferente de cero, continua el decremento, de lo contrario salta a la siguiente etiqueta que cierra el proceso en esta rutina. Lo mismo sucede con TCONV, se carga un valor, .22, en el registro auxiliar PDel0, y se decrementa. Si el valor en este registro es diferente de cero, continua el proceso de lazo, loop, de lo contrario brinca a la etiqueta que cierra el proceso de loop de la rutina.

Una vez revisado y comentado el algoritmo de operación del microcontrolador, en su función de operador generalizado, se compila utilizando la paquetería de Microchip, MPLab, con la cual se revisa que no existan errores de captura ni elementos incoherentes que no permitan generar el conjunto de programas que son necesarios para la programación del dispositivo. Si es así, el mismo paquete permite depurarlo, para corregir estos errores.

Finalmente se puede producir una simulación parcial, pero debido a que está diseñado para operar con señales analógicas, entonces, para comprobar que todo este desarrollo realizado opera de acuerdo con los fundamentos del operador generalizado descritos en el capitulo 1, se realizan pruebas con dicho diseño y con elementos auxiliares, tales como cuatro generadores de funciones y un osciloscopio, las cuales arrojan como resultado, los siguientes oscilogramas, mostradas a continuación, en las figuras (3.4) y consecuentes.

El diagrama representativo del operador generalizado desarrollado con la plataforma de desarrollo de Microchip se observa en la figura (3.3).



$$Salida\ (t+1) = \begin{cases} 1\ si\ \sum_{i=0}^{1} \overset{\bullet}{A}_i(t) - \sum_{j=2}^{3} \overset{\circ}{A}_j(t) \geq DIG1 \\ 0\ si\ \sum_{i=0}^{1} \overset{\bullet}{A}_i(t) - \sum_{j=2}^{3} \overset{\circ}{A}_j(t) < DIG1 \end{cases}$$

Figura 3.3. Diagrama representativo para el operador generalizado utilizando el microcontrolador PIC 16F877A. AN0 y AN1 son las entradas excitadoras, AN2 y AN3 son las entradas inhibitorias, DIG1 es el valor de umbral.

3.2.- El Operador Generalizado estocástico con Microchip

Utilizando el diseño electrónico ya mencionado del operador generalizado se manejaron varios casos. Considerando el diagrama representativo que se muestra en la figura (3.3), se introduce entonces una señal periódica del tipo senoidal de 4.88 volts de pico a pico, a una frecuencia de 1 kHz, como la que se muestra en la figura (3.4), a las 4 entradas analógicas, AN0, AN1, AN2 y AN3, del dispositivo, ver diagrama representativo de la figura (3.3).

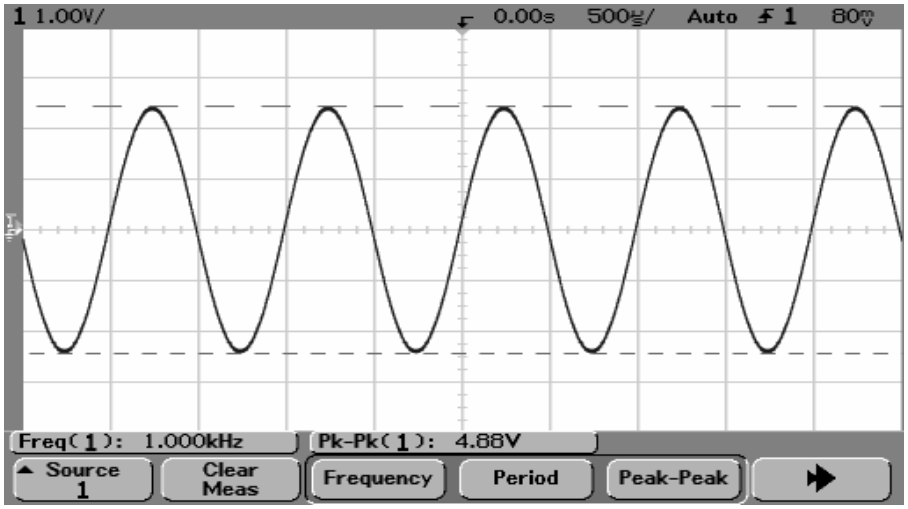


Figura 3.4. Oscilograma de una señal analógica del tipo senoidal, con frecuencia de 1 KHz y un voltaje pico a pico de 4.88 volts, aplicada a las entradas analógicas del microcontrolador.

Aplicando a las cuatro entradas del dispositivo una señal como la que muestra la figura (3.4), al ser operadas por el microcontrolador de acuerdo con el funcionamiento descrito en la sección anterior 3.1, arroja como resultado el oscilograma mostrado en la figura (3.5). En este se observa, de manera periódica, el proceso de comparación entre el valor de umbral y el resultado de la operación de resta entre los valores obtenidos de las señales analógicas digitalizadas, AN0 sumada con AN1, y AN2 sumada con AN3, figura (3.3). Se observa una periodicidad en esta salida debido a que las entradas también lo son. El valor de umbral, DIG1, en el puerto B, es de cero, valor binario "00000000"; en la salida se observa un valor de uno, esto indica que se trata de un resultado de la resta, en AN0, mayor o igual al valor de umbral, DIG1. Con esto se cumple la condición de que salida es 1, cuando AN0 ≥ DIG1. Como se trata de señales prácticamente iguales, tanto en su amplitud como en su frecuencia, la suma entre AN0 y AN1 tiene el mismo valor que el de la suma entre AN2 y AN3, por lo cual, la resta entre AN0, (AN0 + AN1 → AN0), y AN2, (AN2 + AN3 → AN2), da como resultado cero, (AN0 – AN2 → AN0), y ya que el valor de umbral, DIG1, es cero también, se trata de valores iguales, umbral y resultado de la resta, AN0 ≥ DIG1, lográndose la condición de salida = 1, esto es, el caso "al menos 00 de 00" se cumple (operador

And), ya que, de acuerdo al capitulo primero, las entradas excitadoras están inhibidas, AN0 ≈ AN2, y la resta es igual al umbral. En el oscilograma, figura (3.5), también se observan valores de cero, lo que indica que la resta tomó valores por debajo de cero, es decir negativos, que al ser comparados con el umbral, DIG1 = 00 hex, provocan que salida tome el valor 0, que es cuando AN0 < DIG1. Aquí, las entradas excitadoras siguen inhibidas, pero las inhibitorias ocasionan que la resta presente un valor por debajo de cero, cantidad fijada para el umbral, el cual es tomado como negativo, de ahí que el valor de la resta, AN0, sea menor que el umbral, DIG1, es decir, el caso "al menos 00 de 00" se cumple (operador And). Ahora también, de acuerdo al capitulo primero, las entradas excitadoras están inhibidas, AN0 ≈ AN2, pero el resultado de la resta es menor al umbral, con un valor negativo.

Cabe destacar que las señales de los diferentes generadores de funciones, aun cuando tenían la misma amplitud y frecuencia, presentaban pequeños desfases y, aunado a esto, el tiempo de adquisición fue distinto para cada señal de entrada. Es debido a estas circunstancias también, las que provocaron cantidades por debajo de cero, es decir, negativas, como consecuencia, el oscilograma no presenta un 1 constante. De acuerdo con esto, también puede cumplirse el caso "al menos 00 de 01" (operador Or), cuando salida es 1, ya que, según el capitulo 1, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

En general, salvo el detalle comentado, el comportamiento que el oscilograma muestra, figura (3.5), describe y concuerda con los fundamentos ya mencionados del operador generalizado en el capitulo 1.

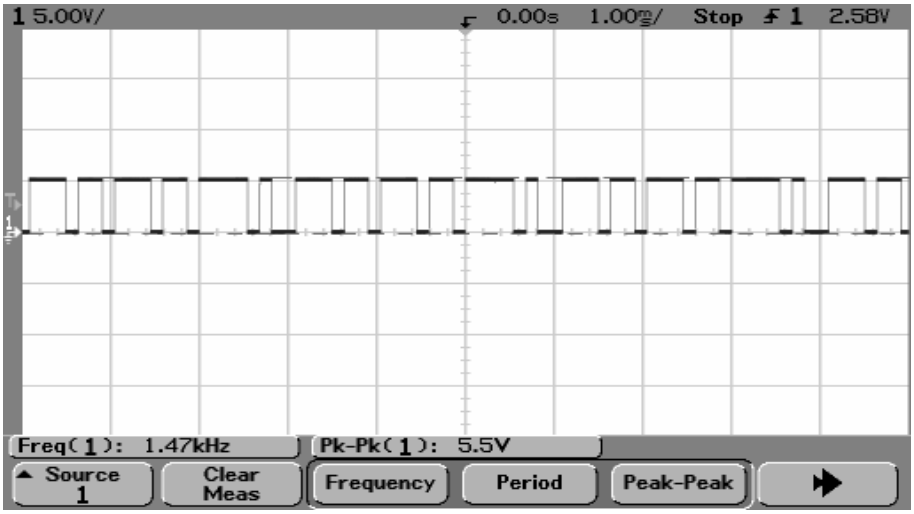


Figura 3.5. Oscilograma representativo de la salida del operador generalizado, al cual se le aplican señales senoidales. Se observa que se cumple el objetivo, aun cuando el valor varía debido al desfase entre señales y los diferentes tiempos de muestreo de estas, considerándose en estos casos, valores por debajo de cero, o negativos.

En la figura (3.6) se muestran dos señales periódicas analógicas del tipo senoidal, una de 4.88 volts de pico a pico a la frecuencia de 1 KHz, aplicada a la entrada AN0; y otra de 6.78 mv también a 1 kHz, la cual es aplicada a las entradas AN1, AN2 y AN3, ver figura (3.3).

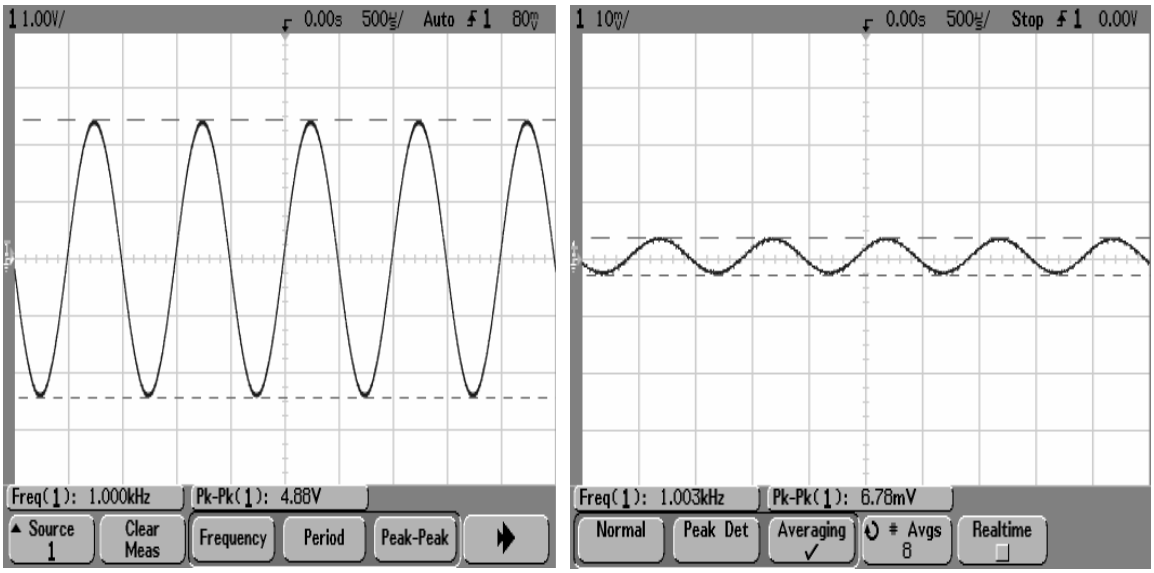


Figura 3.6. Oscilogramas de las señales de entrada. AN0 de 4.88 volts pico a pico. AN1, AN2 y AN3 de 6.78 mv pico a pico. Todas de 1 kHz.

De acuerdo con las señales mostradas en la figura (3.6), el oscilograma resultante que muestra el comportamiento de operador generalizado está en la figura (3.7), donde se observa que se va estabilizando en uno, puesto que AN0 es mucho mayor que AN1, AN2 y AN3 en la suma, (AN0 + AN1 → AN0), (AN2 + AN3 → AN2), y al compararse con el umbral, DIG1 = "00000000", el resultado de la resta, (AN0 – AN2 → AN0), AN0, es mayor que este, con esto, se cumple que salida es uno cuando AN0 ≥ DIG1, lo que satisface la condición "al menos 00 de FF" (operador Or), dado que, según el primer capitulo, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor que el umbral. Cabe señalar que se observan valores de cero, esto debido a que todas estas señales son periódicas, y llegan al cruce por cero, entonces, las entradas excitadoras pueden ser inhibidas, provocando, las inhibitorias, que el resultado de la resta quede por debajo de cero, cantidad fijada para el umbral, por lo que se trata de cantidades negativas, entonces AN0 tiene un valor por debajo de cero, siendo menor a umbral, DIG1, propiciando la condición de que salida es 0 cuando AN0 < DIG1, es decir, el caso "al menos 00 de 00" se satisface (operador And), ya que, de acuerdo con el capitulo 1, las excitadoras están inhibidas y el resultado de la resta está por debajo del umbral, con un valor negativo.

Al igual que en la prueba anterior, las señales de los diferentes generadores de funciones, aun teniendo la misma amplitud y frecuencia, presentan pequeños desfases y, considerando que el tiempo de adquisición es distinto para cada señal de entrada, se presentan cantidades debajo de cero, es decir, negativas, por lo

cual, el oscilograma no presenta un 1 constante. En esta ocasión no es tan perceptible, ya que al ser AN0 mucho mayor, 4.88 vpp, que las otras tres, 6.78 mvpp, todo está en función de su comportamiento y de los valores binarios que de ella se toman.

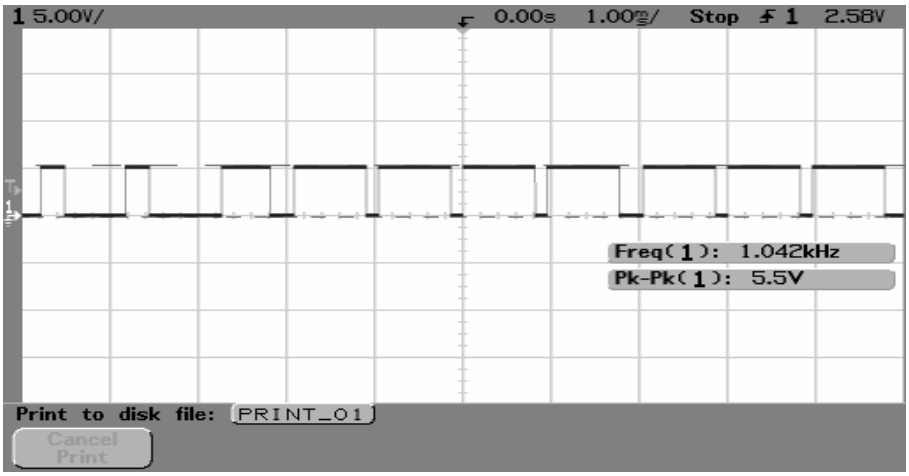


Figura 3.7. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 3.6.

En la figura (3.8) se muestra una señal periódica analógica del tipo senoidal, ahora de 6.78 mv también a 1 kHz, la cual es aplicada a las entradas AN0, AN1, AN2 y AN3, ver figura (3.3).

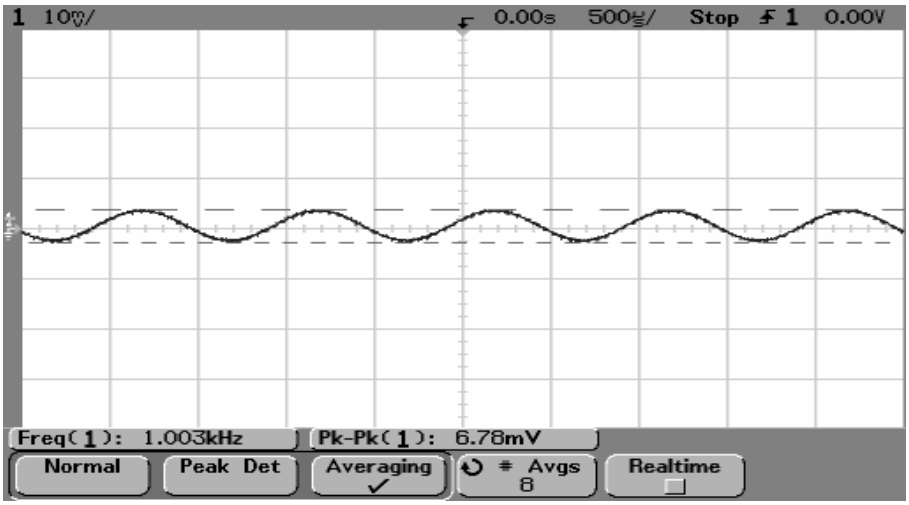


Figura 3.8. Oscilograma que muestra la señal analógica aplicada a las cuatro entradas analógicas, AN0, AN1, AN2, AN3, del microcontrolador.

La salida del operador lo muestra el oscilograma de la figura (3.9), en el cual se observa el valor puesto a 1, dado que las cuatro señales son prácticamente iguales, la resta entre AN0, (AN0 + AN1 → AN0), y AN2, (AN2 + AN3 → AN2), (AN0 – AN2 → AN0), da como resultado cero, y ya que el valor de umbral, DIG1, es cero también, DIG1 = "00000000", se trata de valores iguales, umbral y

resultado de la resta, cumpliéndose la condición de que salida es 1 cuando AN0 ≥ DIG1, satisfaciéndose el caso "al menos 00 de 00" (operador And) ya que, de acuerdo al capitulo primero, las entradas excitadoras están inhibidas, AN0 ≈ AN2, y la resta es igual al umbral. Después se observa una oscilación; como se ha comentado, cuando las señales llegan a sus valores cercanos al cruce por cero, valor fijado para el umbral, las entradas excitadoras pueden ser inhibidas, ocasionando, las inhibitorias, que la resta tome un valor negativo, considerado por debajo del valor del umbral, y entonces, dado que resta, AN0, es negativo, la salida toma el valor de cero, comportándose de acuerdo con la condición AN0 < DIG1, pues DIG1 = "00000000", cumpliéndose el caso "al menos 00 de 00" (operador And) ya que, según el capitulo 1, las entradas excitadoras están inhibidas y el resultado de la resta es menor que el umbral, con un valor negativo.

Debido a la periodicidad de las entradas y a la desincronia de ellas, que ya se ha citado, también puede cumplirse el caso "al menos 00 de 01" (operador Or), cuando salida es 1, ya que, según el capitulo 1, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

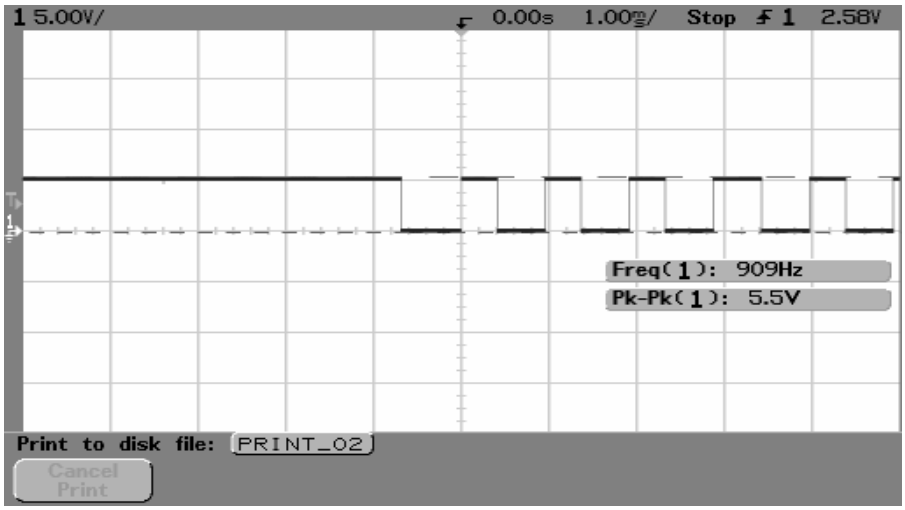


Figura 3.9. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 3.8.

A continuación se presentan las mismas pruebas mostradas anteriormente, pero aplicándose ahora otras formas de señal. En primer lugar, una señal cuadrada de 5.94 volts de pico a pico a 1 KHz, figura (3.10), aplicada a las cuatro entradas analógicas AN0, AN1, AN2 y AN3, del microcontrolador, ver figura (3.3).

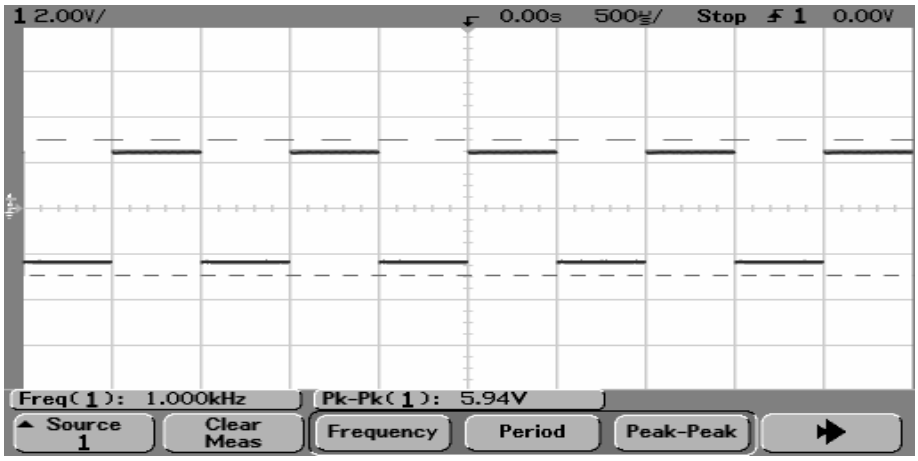


Figura 3.10. Oscilograma de una señal cuadrada de 5.94 volts pico a pico, a 1 kHz, aplicada a los cuatro canales analógicos del microcontrolador.

En la figura (3.11) se observa la salida del operador generalizado, y se aprecia como al inicio del oscilograma la salida toma un valor de cero, es decir, AN0 < DIG1, cuando DIG1 = "00000000", el caso "al menos 00 de 00" se cumple (operador And) ya que, de acuerdo al capitulo 1, las entradas excitadoras están inhibidas y el resultado de la resta esta por debajo del umbral, con un valor negativo. Como se ha comentado en párrafos anteriores, el desfase entre cada una de las cuatro señales es tal, que las excitadoras pueden ser inhibidas, provocando, las inhibitorias, que el resultado de la resta sea negativo, en AN0, quedando por debajo de cero, valor establecido para el umbral, DIG1, por lo cual se observa esta condición en la gráfica. Posterior a esto, salida empieza a tomar valores de 1 de manera periódica, lo que implica que el resultado de la resta, resta entre AN0, (AN0 + AN1 → AN0), y AN2, (AN2 + AN3 → AN2), (AN0 – AN2 → AN0), toma valores iguales al umbral, y ya que este, DIG1, es cero también, DIG1 = 00 hexadecimal, se trata entonces de valores iguales. Pero resta también toma valores por debajo de este, cuando se inhiben las entradas excitadoras, cumpliéndose las dos condiciones en forma periódica, salida vale 1 cuando AN0 ≥ DIG1, es decir, el caso "al menos 00 de 00" (operador And) se satisface, puesto que las excitadoras están inhibidas, y resta es igual a umbral, y salida vale cero cuando AN0 < DIG1, el caso "al menos 00 de 00" se cumple (operador And) dado que permanecen inhibidas las excitadoras y el resultado de la resta esta por debajo de umbral, DIG1 = "00000000", con valor negativo.

Debido a la periodicidad de las entradas y a la desincronia de ellas, que ya se ha citado, también puede cumplirse el caso "al menos 00 de 01" (operador Or), cuando salida es 1, ya que, según el capitulo 1, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

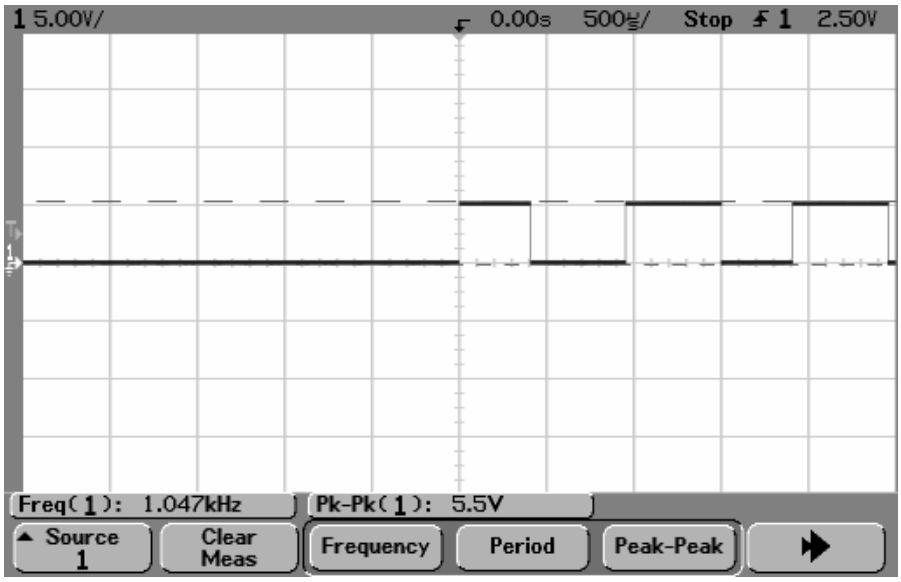


Figura 3.11. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 3.10.

En la figura (3.12) se muestran dos señales periódicas del tipo cuadrada, una de 5.94 volts de pico a pico a la frecuencia de 1 KHz, aplicada a la entrada AN0; y otra de 25.9 mv también a 1 KHz, la cual es aplicada a las entradas AN1, AN2 y AN3, ver figura (3.3).

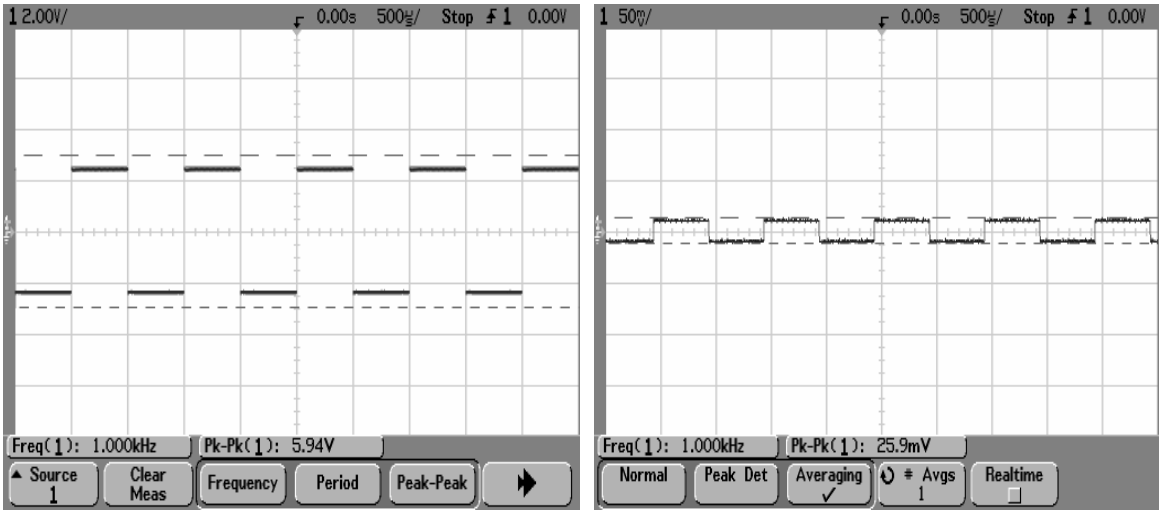


Figura 3.12. Oscilogramas de las señales de entrada. AN0 de 5.94 volts pico a pico. AN1, AN2 y AN3 de 25.9 mv pico a pico. Todas de 1 kHz.

De acuerdo con las señales mostradas en la figura (3.12), el oscilograma resultante muestra el comportamiento de operador generalizado, figura (3.13), donde se observa una periodicidad entre los dos valores que puede tomar salida. Como en la anterior situación, dado que estas señales periódicas pueden tomar valores extremos, es decir, 1 o 0, el resultado de la resta, resta entre AN0, (AN0 +

AN1 → AN0), y AN2, (AN2 + AN3 → AN2), (AN0 – AN2 → AN0), queda sobre o bajo el umbral, DIG1 = "00000000", observándose los resultados del oscilograma. La salida se comporta en forma periódica cumpliéndose ambas condiciones. Salida es 1 cuando AN0 ≥ DIG1, puesto que AN0 está en el máximo positivo, y es mucho mayor que AN1, AN2 y AN3, y al compararse la resta, AN0, con el umbral, DIG1 = "00000000", es mayor que este, satisfaciéndose el caso de que "al menos 00 de FF" ocurra (operador Or), puesto que, de acuerdo con el capitulo 1, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral. Pero salida toma luego el valor de 0 cuando AN0 < DIG1, ya que AN0 es menor, al llegar al cruce por cero, con respecto a AN2 y AN3, las entradas inhibitorias, y al restarse, dan valores por debajo de cero, es decir, cantidades negativas, que al ser comparadas con el valor de umbral propician la condición de que salida es 0, cumpliéndose el caso "al menos 00 de 00" (operador And), ya que las entradas excitadoras están inhibidas, y el resultado de la resta es menor al umbral, con un valor negativo.

Al igual que en pruebas anteriores, las señales de los diferentes generadores de funciones, aun teniendo la misma amplitud y frecuencia, presentan pequeños desfases y, considerando que el tiempo de adquisición es distinto para cada señal de entrada, se presentan cantidades debajo de cero, es decir, negativas, por lo cual, el oscilograma no presenta un 1 constante. En esta ocasión no es tan perceptible, ya que al ser AN0 mucho mayor, 5.94 vpp, que las otras tres, 25.9 mvpp, todo está en función de su comportamiento y de los valores binarios que de ella se toman.

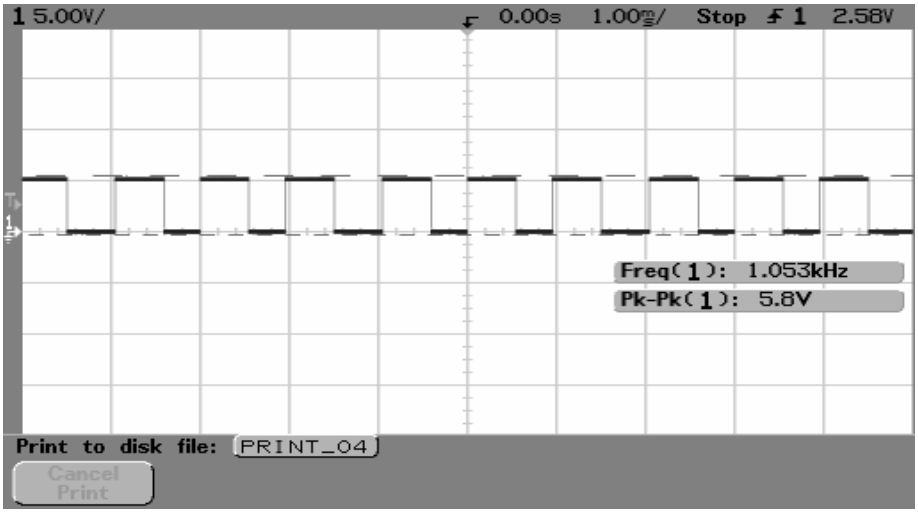


Figura 3.13. Oscilograma resultante al ser aplicadas las señales de la figura 3.12.

En la figura (3.14) se muestra una señal periódica del tipo cuadrada, ahora de 25.9 mv pico a pico también a 1 kHz, la cual es aplicada a las entradas AN0, AN1, AN2 y AN3, ver figura (3.3).

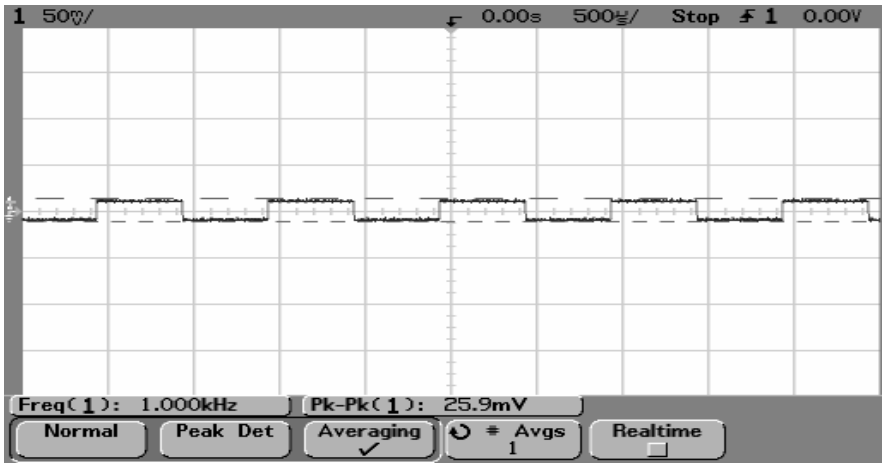


Figura 3.14. Oscilograma que muestra la señal aplicada a las cuatro entradas analógicas, AN0, AN1, AN2, AN3, del microcontrolador.

La salida del operador lo muestra el oscilograma de la figura (3.15), en el cual se observa una periodicidad entre los dos valores que puede tomar salida. Como en anteriores situaciones, dado que estas señales periódicas pueden tomar valores extremos, es decir, 1 o 0, el resultado de la resta, resta entre AN0, (AN0 + AN1 → AN0), y AN2, (AN2 + AN3 → AN2), (AN0 – AN2 → AN0), queda sobre o bajo el umbral, DIG1 = "00000000", observándose estos en el oscilograma. La salida se comporta en forma periódica cumpliéndose ambas condiciones. Salida es 1 cuando AN0 ≥ DIG1, puesto que el resultado de la resta, AN0, al compararse con el umbral, DIG1 = "00000000", es igual que este, dado que las señales en las entradas son prácticamente iguales, con ello, se cumple el caso "al menos 00 de 00" (operador And), ya que AN0 ≈ AN2, por lo que las entradas excitadoras están inhibidas, y el resultado de la resta es igual al umbral, según el capitulo primero. Pero salida toma luego el valor de 0 cuando AN0 < DIG1, ya que resta, AN0, es menor, o negativo, y al ser comparada con el valor de umbral propician la condición de que salida es 0, debido a que las señales excitadoras continúan inhibidas, pero ahora las inhibitorias hacen el resultado de la resta negativo, por debajo de umbral, satisfaciéndose el caso "al menos 00 de 00" (operador And).

Debido a la periodicidad de las entradas y a la desincronia de ellas, que ya se ha citado, también puede cumplirse el caso "al menos 00 de 01" (operador Or), cuando salida es 1, ya que, según el capitulo 1, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

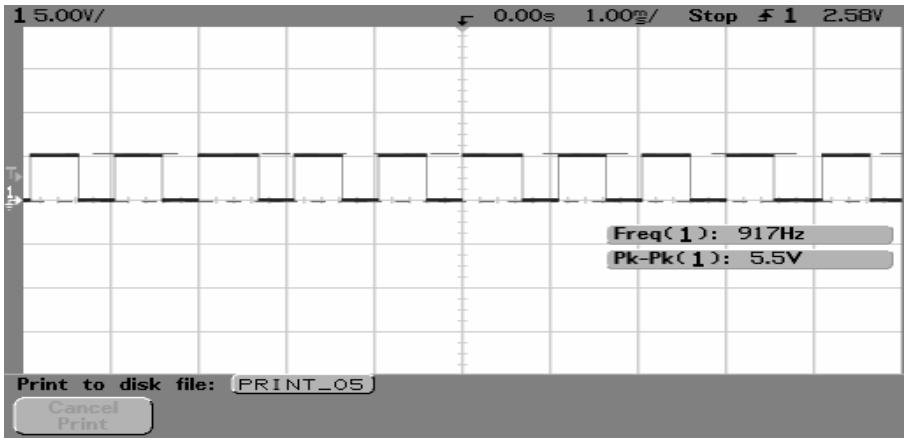


Figura 3.15. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 3.14.

En la figura (3.16) se muestra una señal periódica del tipo triangular, de 4.801volts de pico a pico a 1 kHz, la cual es aplicada a las entradas AN0, AN1, AN2 y AN3, ver la figura (3.3).

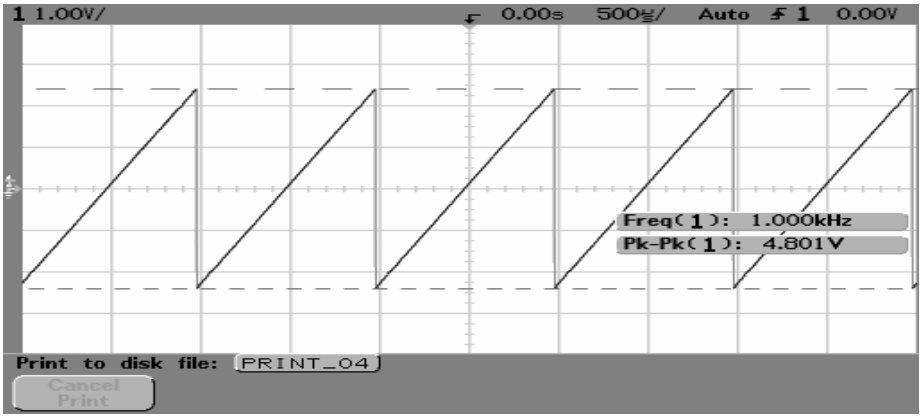


Figura 3.16. Oscilograma de una señal triangular de 4.8 volts pico a pico, a 1 kHz, aplicada a las cuatro entradas analógicas del microcontrolador.

La salida del operador lo muestra el oscilograma de la figura (3.17), en el cual se observa el valor puesto a 1, dado que las cuatro señales son prácticamente iguales, cumpliéndose la condición de que salida es 1 cuando AN0 ≥ DIG1, es decir, se cumple el caso "al menos 00 de 00" (operador And), esto debido a que las señales excitadoras están inhibidas, y el resultado de la resta es igual al umbral, aunque después se observa una oscilación. Como se ha observado en anteriores oscilogramas, salida está en función del comportamiento de las señales periódicas a la entrada, cuando las excitadoras puedan o no ser inhibidas por las otras entradas, y como se ha venido comentando, el desfase de estas también provoca que la condición de salida cambie periódicamente. Ahora se observa que en la parte inicial del oscilograma salida es 1 constante, debido a que resta, resta entre AN0, (AN0 + AN1 → AN0), y AN2, (AN2 + AN3 → AN2), (AN0 – AN2 →

AN0), AN0, es igual que umbral, DIG1 = "00000000", puesto que todas las señales son prácticamente iguales, cumpliéndose el caso "al menos 00 de 00" (operador And), las señales excitadoras están inhibidas, y el resultado de la resta es igual al umbral, pero al correr en el tiempo el desfase es notorio, provocando que resta decremente su valor por debajo de cero, como negativo, haciendo que salida sea 0, pues AN0 < DIG1, satisfaciéndose el caso "al menos 00 de 00" (operador And), debido a que las señales excitadoras continúan inhibidas, y las inhibitorias provocan que el resultado de la resta sea menor al umbral, con un valor negativo. Como las señales no están sincronizadas, los resultados de las sumas y de la resta varían en pequeñas cantidades, suficientes para sobrepasar el valor de umbral establecido.

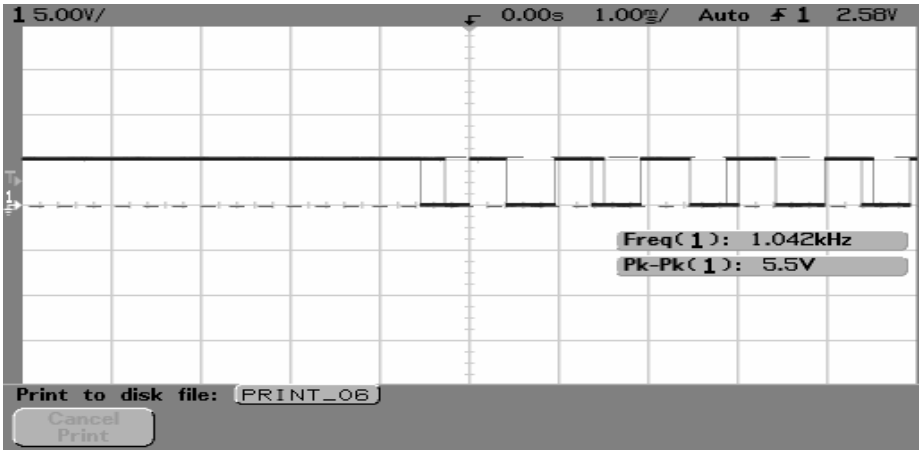


Figura 3.17. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 3.16.

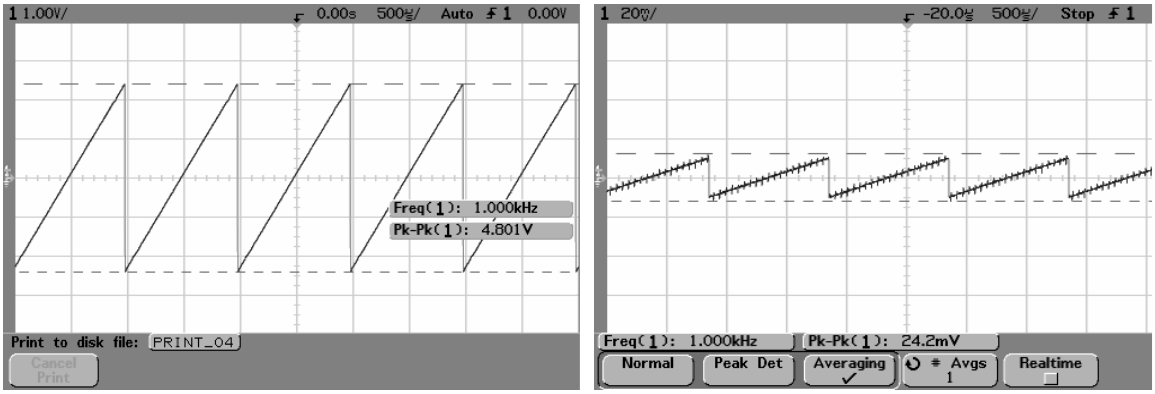


Figura 3.18. Oscilogramas de las señales de entrada. AN0 de 4.8 volts pico a pico. AN1, AN2 y AN3 de 24.2 mv pico a pico. Todas de 1 kHz.

En la figura (3.18) se muestran dos señales periódicas del tipo triangular, una de 4.801 volts de pico a pico a la frecuencia de 1 KHz, aplicada a la entrada AN0; y otra de 24.2 mv también a 1 kHz, la cual es aplicada a las entradas AN1, AN2 y AN3, ver figura (3.3).

La salida del operador lo muestra el oscilograma de la figura (3.19), en el cual se observa que salida toma ambos valores de manera periódica. Dado que estas señales periódicas toman valores extremos en el pico de la rampa, es decir, de 1 a 0, entonces, cuando la pendiente cae a cero, un mínimo, se trata de un valor menor al de umbral, DIG1 = "00000000", y al incrementarse la pendiente, hasta llegar a un máximo, sobrepasa a umbral, DIG1. Así, el resultado de la resta, resta entre AN0, (AN0 + AN1 → AN0), y AN2, (AN2 + AN3 → AN2), (AN0 – AN2 → AN0), queda sobre o bajo el umbral, DIG1, observándose los resultados del oscilograma. Salida es 1 cuando AN0 ≥ DIG1, puesto que AN0 es mucho mayor que las AN1, AN2 y AN3, y al compararse la resta, AN0, con el umbral, DIG1 = "00000000", es mayor que este, aquí se satisface el caso "al menos 00 de FF" (operador Or), dado que las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral, según lo establecido en el capitulo 1. Pero salida toma luego el valor de 0 cuando AN0 < DIG1, ya que AN0 es menor, está en el cruce por cero, con respecto a AN2 y AN3, y al restarse, AN0 da valores por debajo de cero, es decir, cantidades negativas, que al ser comparadas con el valor de umbral propician la condición de que salida es 0, cumpliéndose el caso "al menos 00 de 00" (operador And), ya que las entradas excitadoras están inhibidas, y las inhibitorias ocasionan que el resultado de la resta sea menor al umbral, con un valor negativo.

Al igual que en pruebas anteriores, las señales de los diferentes generadores de funciones, aun teniendo la misma amplitud y frecuencia, presentan pequeños desfases y, considerando que el tiempo de adquisición es distinto para cada señal de entrada, se presentan cantidades debajo de cero, es decir, negativas, por lo cual, el oscilograma no presenta un 1 constante. En esta ocasión no es tan perceptible, ya que al ser AN0 mucho mayor, 4.8 vpp, que las otras tres, 24.2 mvpp, todo está en función de su comportamiento y de los valores binarios que de ella se toman.

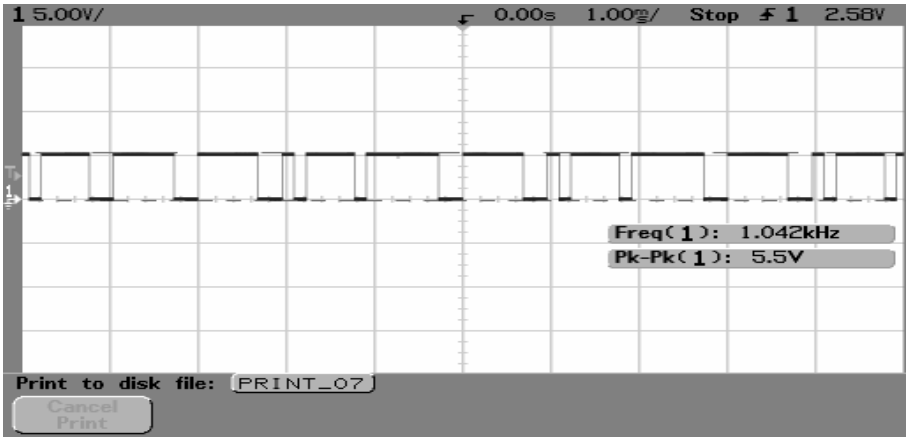


Figura 3.19. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 3.18.

En la figura (3.20) se muestra una señal periódica del tipo triangular, ahora de 24.2 mv también a 1 kHz, la cual es aplicada a las entradas AN0, AN1, AN2 y AN3, ver figura (3.3).

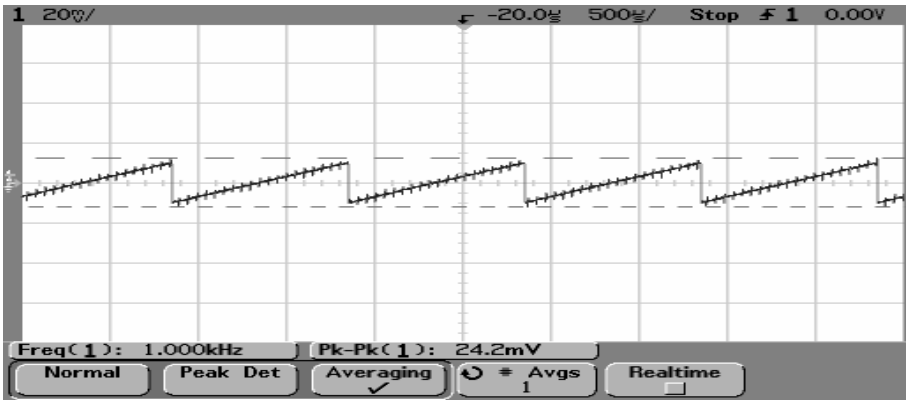


Figura 3.20. Oscilograma que muestra la señal analógica aplicada a las cuatro entradas analógicas, AN0, AN1, AN2, AN3, del microcontrolador.

La salida del operador lo muestra el oscilograma de la figura (3.21), en el cual se observa una periodicidad entre los dos valores que puede tomar salida. Como en anteriores situaciones, dado que estas señales periódicas pueden tomar valores extremos en el pico de la rampa, es decir, de 1 a 0, el resultado de la resta queda sobre, un máximo, o debajo, un mínimo, del umbral, DIG1, observándose los resultados del oscilograma. La salida se comporta en forma periódica cumpliéndose ambas condiciones. Salida es 1 cuando AN0 ≥ DIG1, puesto que el resultado de la resta, resta entre AN0, (AN0 + AN1 → AN0), y AN2, (AN2 + AN3 → AN2), (AN0 – AN2 → AN0), AN0, al compararse con el umbral, DIG1 = "00000000", es igual que este, satisfaciéndose el caso "al menos 00 de 00" (operador And), ya que las entradas excitadoras están inhibidas, AN0 ≈ AN2, y el resultado de la resta es igual a umbral. Pero salida toma luego el valor de 0 cuando AN0 < DIG1, ya que AN0 es menor, está por debajo de cero, y al ser comparada la resta, AN0, con el valor de umbral propician la condición de que salida es 0, cumpliéndose el caso "al menos 00 de 00" (operador And), ya que las entradas excitadoras permanecen inhibidas, y las inhibitorias hacen que el resultado de la resta sea un valor negativo, por debajo de umbral.

Al igual que en pruebas anteriores, las señales de los diferentes generadores de funciones, aun teniendo la misma amplitud y frecuencia, presentan pequeños desfases y, considerando que el tiempo de adquisición es distinto para cada señal de entrada, también puede cumplirse el caso "al menos 00 de 01" (operador Or), cuando salida es 1, ya que, según el capitulo 1, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

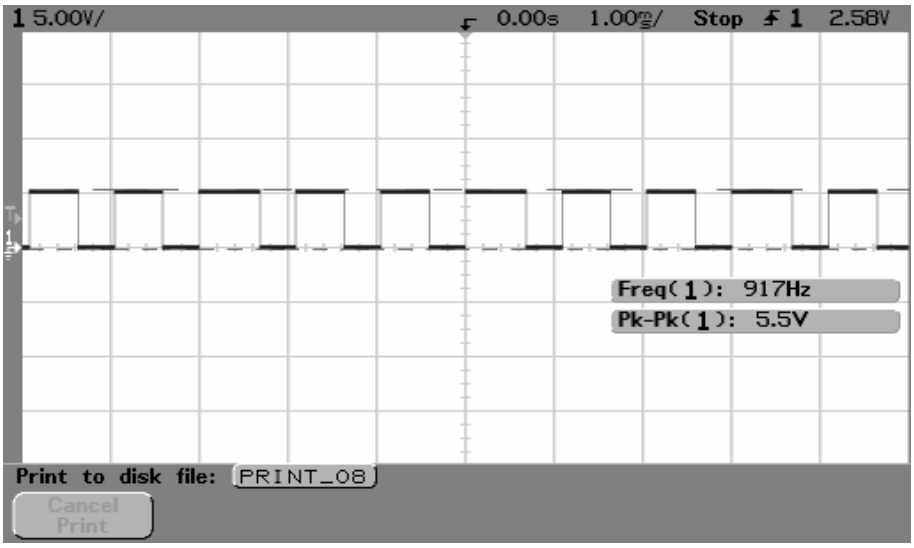


Figura 3.21. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 3.20.

En la figura (3.22) se muestran cuatro señales periódicas de diferentes tipos: dos senoidales de 4.88 volts pico a pico aplicadas, una la entrada AN0 y otra aplicada a la entrada AN3, una cuadrada de 5.94 volts pico a pico aplicada a la entrada AN1, y una triangular de 4.801 volts pico a pico aplicada a la entrada AN2, todas a 1 kHz, ver figura (3.3).

La salida del operador lo muestra el oscilograma de la figura (3.23), en el cual se observa el valor puesto a 1. Como las cuatro señales son diferentes en sus formas, las sumas de estas son distintas, ya que AN0 tiene un valor muy alto cuando la señal cuadrada está en uno, un máximo, y se mantiene este, aun cuando cambia a un mínimo, todo esto casi en forma súbita, (AN0 + AN1 → AN0). Cosa similar sucede con AN2, pero no crece tan rápido pues ambas señales se incrementan en función de su forma, (AN2 + AN3 → AN2), de ahí que, AN0 tome valores muy altos rápidamente, por lo que la resta, (AN0 – AN2 → AN0), da un valor por encima del umbral, cumpliéndose la condición de que salida es 1 cuando AN0 ≥ DIG1, aquí se satisface el caso de "al menos 00 de FF" (operador Or), debido a que las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral. Pero al decaer la señal senoidal, AN0 es menor que AN2, por lo que el resultado de la resta, (AN0 – AN2 → AN0), AN0, cae por debajo de umbral, cumpliéndose la condición de que salida es 0 cuando AN0 < DIG1, satisfaciéndose el caso de "al menos 00 de 00" (operador And), ya que las entradas excitadoras están inhibidas, y las inhibitorias generan el resultado de la resta menor al umbral, con valor negativo.

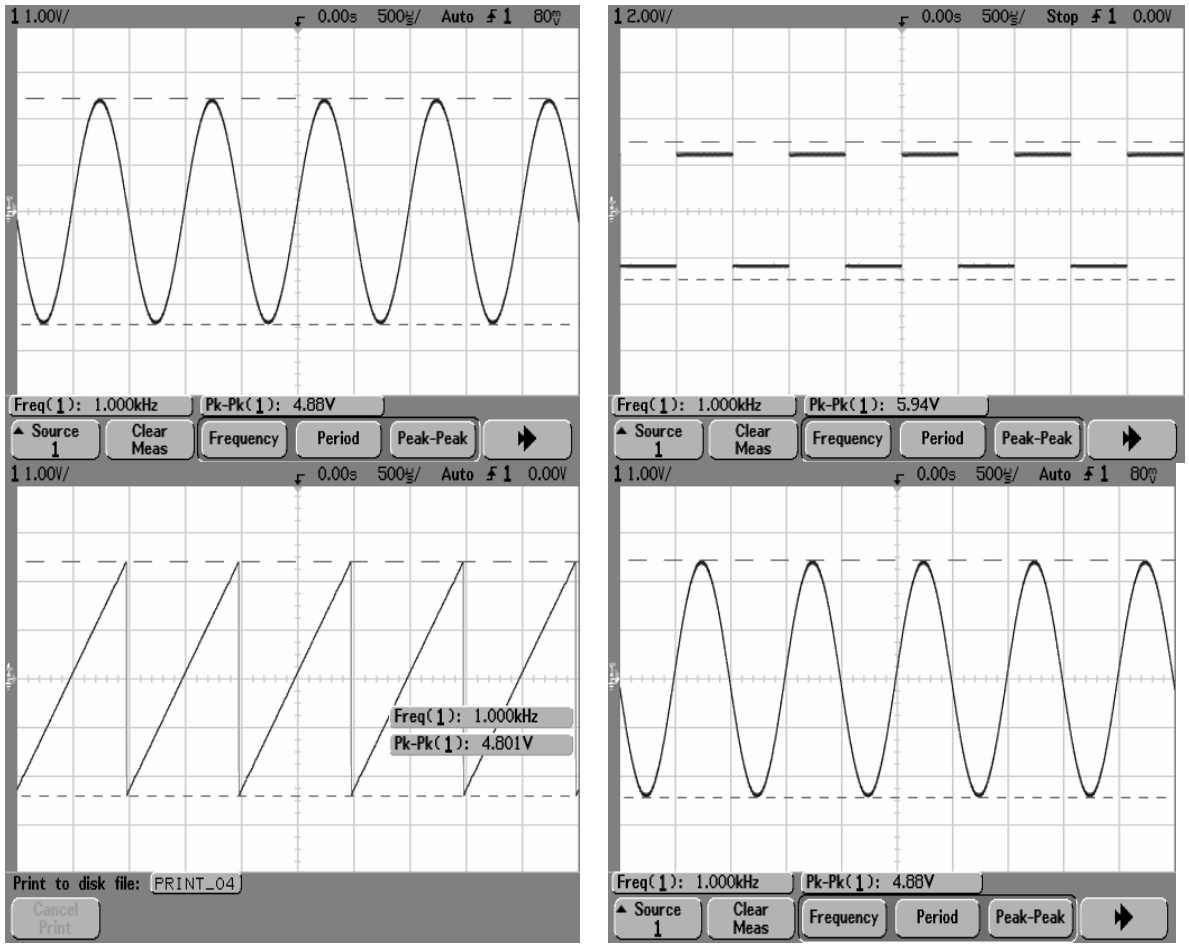


Figura 3.22. Oscilogramas de las señales que entran a los cuatro canales analógicos del microcontrolador. AN0: senoidal de 4.88 vpp, AN1: cuadrada de 5.94 vpp, AN2: triangular 4.801 vpp, AN3: senoidal de 4.88 vpp. Todas a 1 kHz.

Posteriormente se observa una oscilación. Debido a la periodicidad de las entradas y a sus diferentes formas, además de las consideraciones planteadas al momento de sumarse, nuevamente en la resta de estas señales hay valores por debajo de cero, las inhibitorias hacen que el resultado de la resta sea negativo, que propician la condición de que salida es 0 cuando AN0 < DIG1, el caso "al menos 00 de 00" se cumple, las excitadoras están inhibidas, cambiando a 1 cuando AN0 ≥ DIG1, es decir, cuando hay valores iguales o mayores al umbral, el caso "al menos 00 de FF" se satisface, las excitadoras no están inhibidas.

En esta prueba, los pequeños desfases no son tan notorios, ya que las señales de los diferentes generadores de funciones, aun teniendo la misma frecuencia, varían un poco en su amplitud. Además, considerando que el tiempo de adquisición es distinto para cada señal de entrada, todavía se presentan cantidades debajo de cero, es decir, negativas, por lo cual, el oscilograma no presenta un 1 constante.

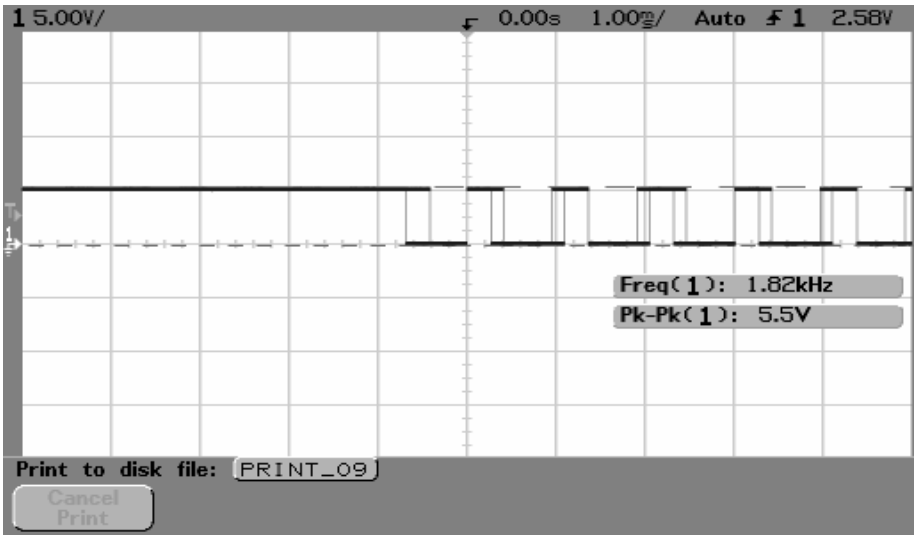


Figura 3.23. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 3.21.

Como se aprecia, el diseño e implementación del operador generalizado, utilizando la plataforma de desarrollo de Microchip y la paquetería de diseño MPLab, ha permitido comprobar y corroborar las bases y planteamientos descritos en el primer capitulo de este trabajo. En este caso, la visualización de los resultados no ha sido tan explicita como lo fue con Altera y su MAX + plus II, para interpretar el comportamiento de los oscilogramas obtenidos, pero no ha sido impedimento para confrontar estos resultados con las bases expuestas en aquel primer capitulo, la salida, elemento principal, ha permitido comprobar que el operador generalizado diseñado en este capitulo trabaja como se ha planteado. Además, se agrega un elemento importante en este sistema, un convertidor analógico-digital que permite utilizar este operador en el campo de las señales continuas, como se observa en este apartado 3.2.

Para obtener los resultados del operador generalizado, mostrados en este capitulo, trabajando con el microcontrolador 16F877A, se utilizó el circuito de la figura (3.27), cuyo arreglo consiste en la conexión al dispositivo mencionado de conectores para facilitar el acoplamiento de las señales analógicas, en el puerto A, y los 8 bits en el puerto B, y las salidas de visualización ya comentadas. Además, se agrega un dispositivo MAX 232, para comunicación serial, y un conector DB-9 para conectarla con una PC, o lap top, utilizando su puerto serie.

La conexión es sencilla, ya que se indica, en la figura (3.24), las resistencias necesarias para la protección de los pines de entrada, de 270 Ohms, por lo que los puertos A, pines del 2 al 7, y B, pines del 33 al 40, cuentan con estas resistencias. También, en el puerto C, pines 23 y 24, y D, pin 22, se utilizan tres resistencias del mismo valor, para proteger los pines utilizados para visualizar la actividad del microcontrolador, en los cuales están conectados LED's para ello.

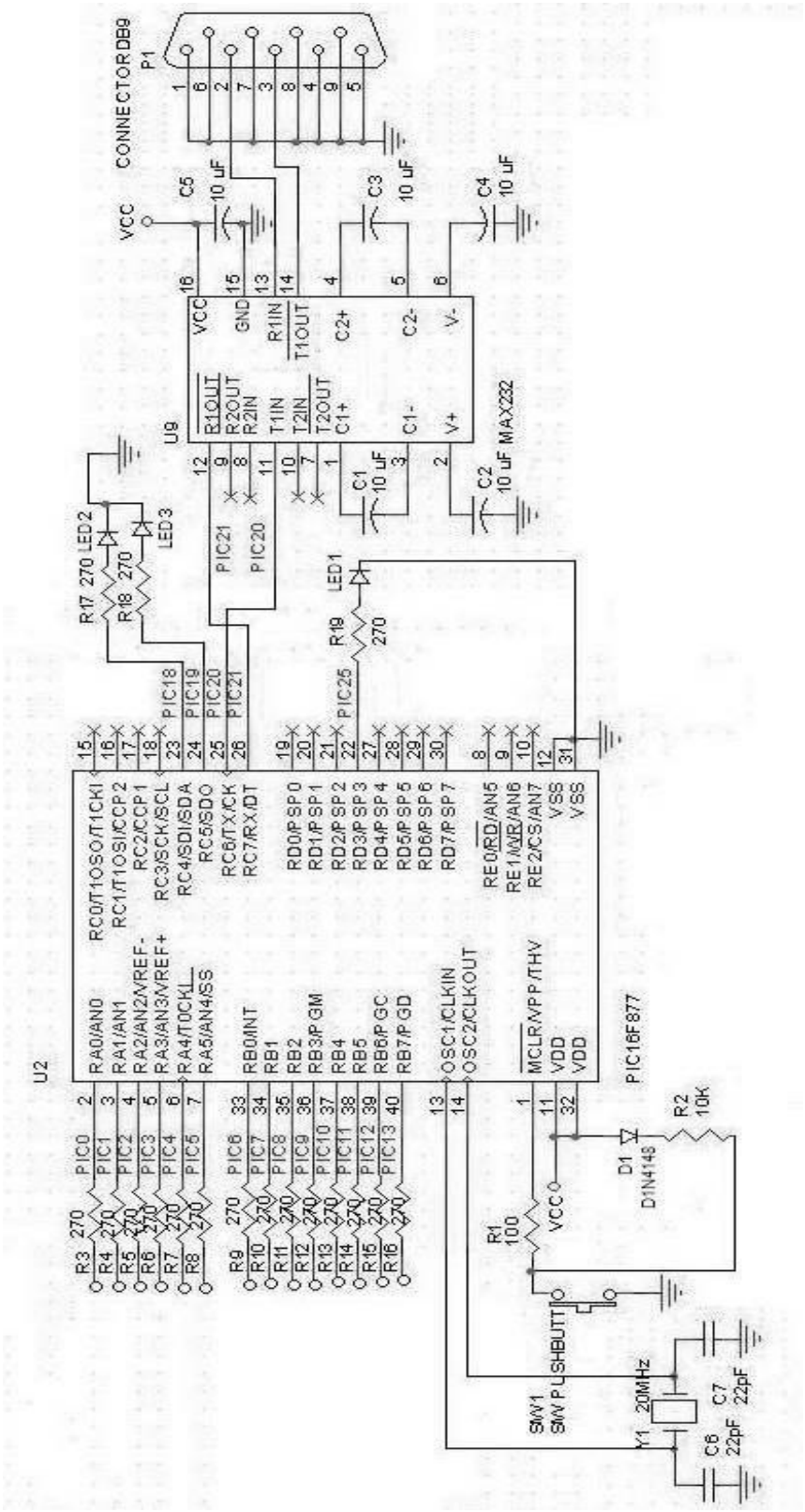


Figura 3.24. Diagrama esquemático del circuito impreso utilizado para trabajar con el microcontrolador PIC 16F877A como operador generalizado.

En esta figura (3.24), se muestra un arreglo de dispositivos, dos resistencias de 100 y 10 KOhms, así como de un diodo 1N4148, de switcheo, para protección del dispositivo al realizar un restablecimiento, o reset, en el pin 1, MCLR operando en estado bajo. Se observa la conexión del cristal, pines 13 y 14, de 20 MHz, frecuencia con la que opera el dispositivo, así como sus capacitores necesarios, dos de 22 pF. Finalmente, se observa la conexión del dispositivo MAX 232 con el PIC, a través de los pines 25 y 26, con sus 4 respectivos capacitares de valores recomendados, 10 µF, para la transmisión serie que pudiera tener con algún dispositivo a través del puerto serie del dicho equipo. Por ello, se tiene el conector DB-9, agregado a este esquemático. Un capacitor más es utilizado para eliminar corrientes parásitas en este arreglo.

De acuerdo con las conexiones descritas en este sencillo arreglo, que muestra el esquemático de la figura (3.24), y utilizando el programa de generación de circuitos impresos, Eagle, se obtiene la tarjeta mostrada en la figura (3.25). Esta muestra la distribución de los componentes ya mencionados, situando, de manera optima, a cada uno de estos elementos involucrados en el esquemático anterior, figura (3.24).

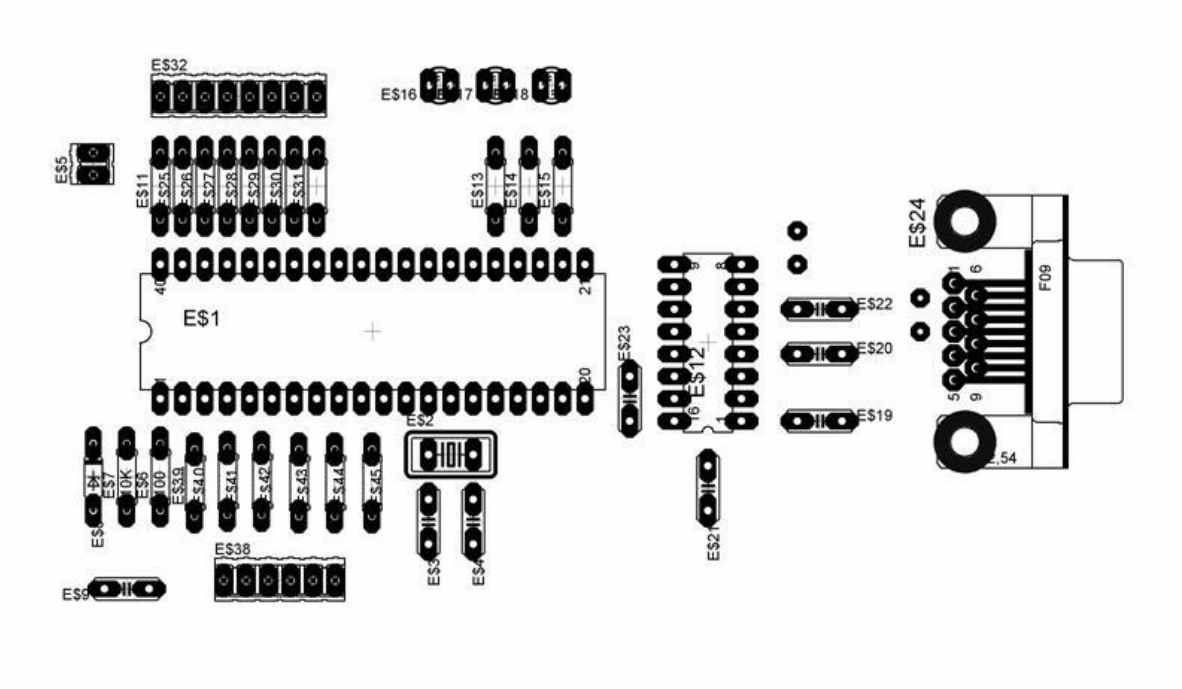


Figura 3.25. Tarjeta de desarrollo para el PIC 16F877A.

La figura (3.26) muestra las pistas, la parte inferior, de la tarjeta, cuyas trayectorias describen las interconexiones de los diferentes elementos utilizados en el esquemático de la figura (3.24). A través de estas pistas fluyen las señales necesarias para que opere el microcontrolador como se ha descrito a lo largo de este capitulo.

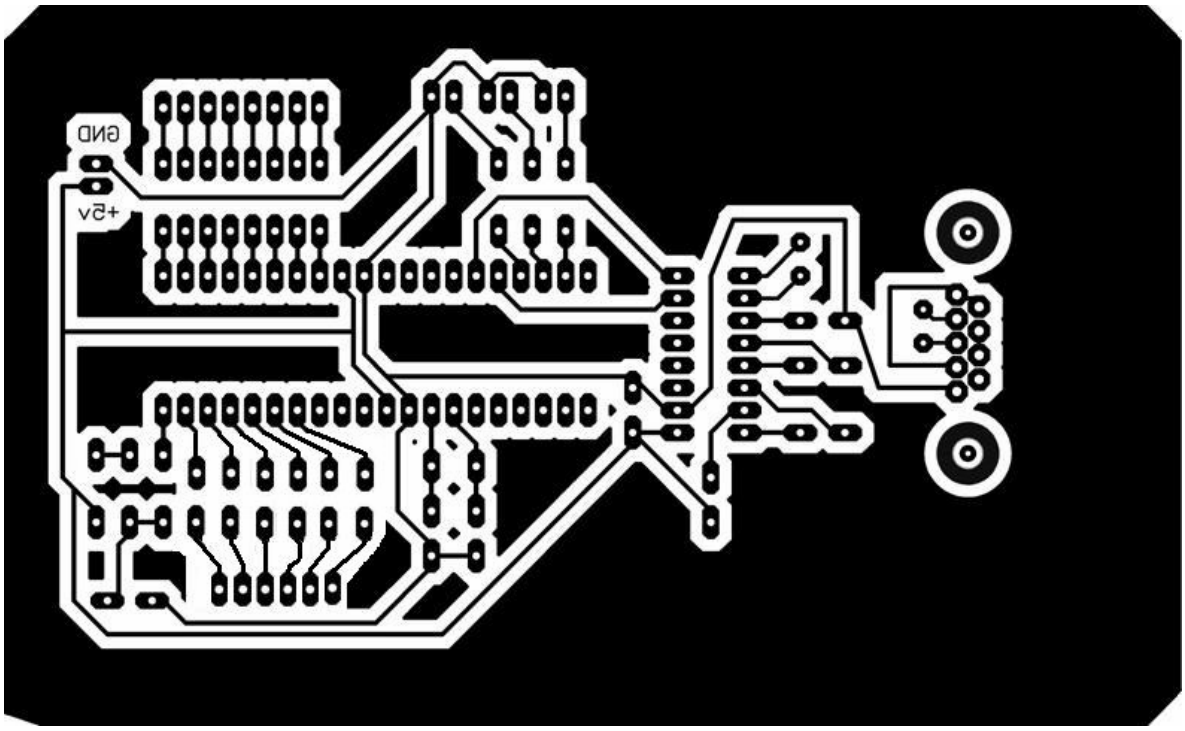


Figura 3.26. Figura del circuito impreso para hacer la tarjeta de desarrollo.

La tarjeta con todos los dispositivos y elementos necesarios, de acuerdo con el esquemático de la figura (3.24), se muestra en la figura (3.27). El detalle muestra en conjunto todos estos componentes fijos sobre la tarjeta del circuito impreso, mostrados en las figuras (3.25) y (3.26).

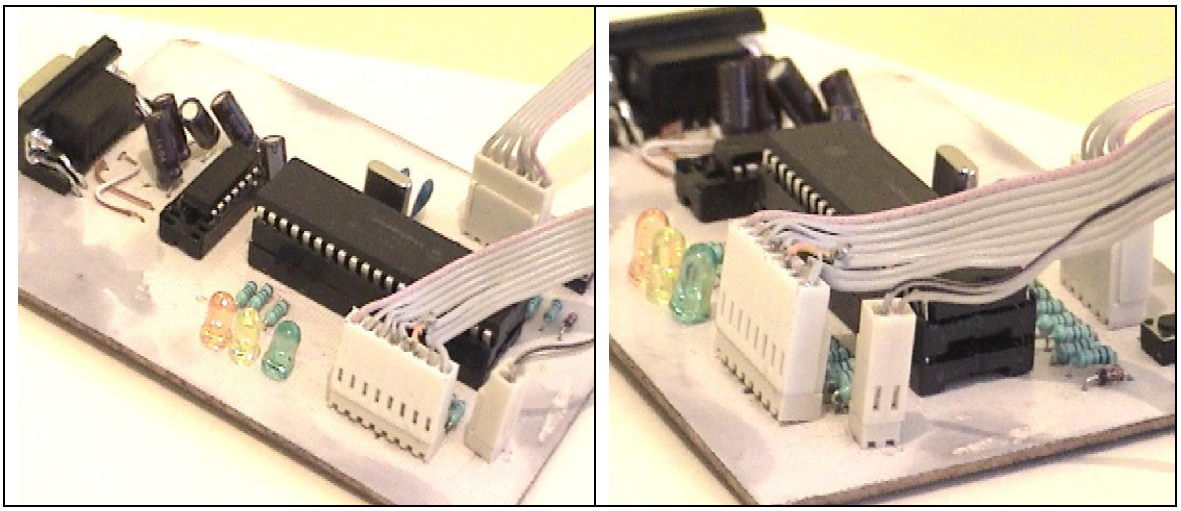


Figura 3.27. Detalle de la tarjeta sobre la cual está montado el microcontrolador, para trabajar como operador generalizado.

De acuerdo con los resultados obtenidos, tanto en este capitulo como en el anterior, se percibe que ambas plataformas ofrecen resultados alentadores para la implantación del operador generalizado. Por ello, en el siguiente capitulo se realiza un comparativo para definir que consideraciones se pueden tomar al momento de elegir alguna de estas.

Capitulo 4

Comparativo entre ambas plataformas de desarrollo

Como se ha visto en los capítulos precedentes, el desarrollo de este proyecto se ha basado en diferentes plataformas de trabajo, en las cuales se observan, desde diferente perspectiva, los resultados planteados para el funcionamiento del operador generalizado. A continuación, se hacen los comparativos entre ambos resultados. Cabe señalar que, debido a la diferencia de plataformas, los resultados no son completamente similares, puesto que Altera se mantiene en el campo de las señales digitales, mientras que Microchip puede abarcar señales analógicas gracias a que varios de sus dispositivos incluyen convertidores analógico-digitales, caso particular del microcontrolador 16F877A, lo cual incrementa la utilidad de un operador generalizado implantado en un dispositivo de estas características. Aun así, el resultado de ambas es equiparable, como se ve en este comparativo.

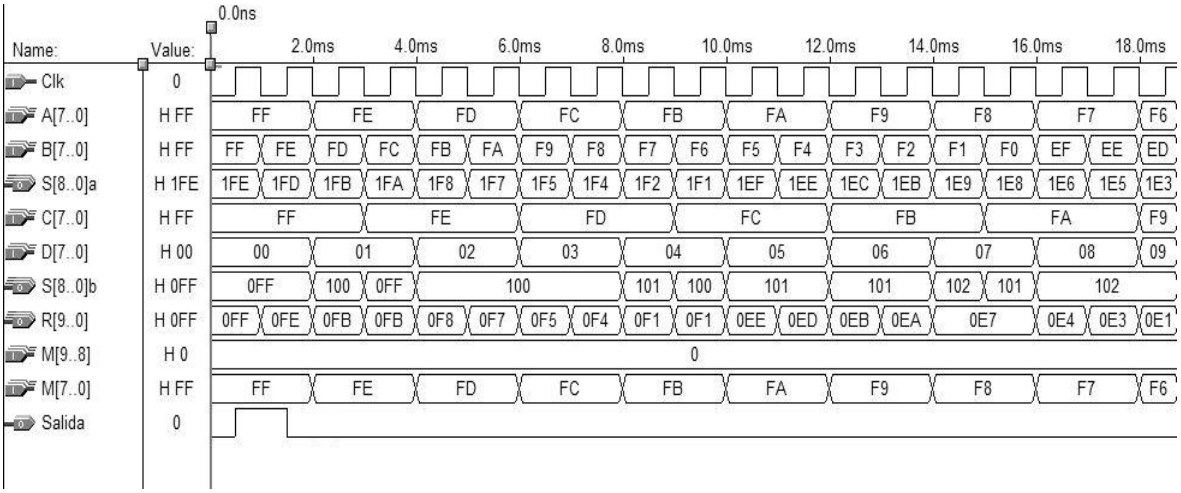


Figura 4.1. Respuesta del operador generalizado implementado con el software MAX + plus II.

La figura (4.1) muestra el funcionamiento del operador generalizado, se observan todas las entradas que afectan al circuito, los resultados intermedios y finalmente la salida del sistema. La entrada de Clk se encarga de suministrar el pulso de reloj, en cuya frecuencia el comparador entra el resultado de la resta, R(9..0), y lo opera con el valor de umbral, M(7..0). El periodo del pulso de reloj es de 1 milisegundo, esto es, la frecuencia del circuito es de 1 KHz de operación. También se observan las entradas de los sumadores, las cuales varían sus cantidades en periodos de tiempo distintos, siendo B(7..0) la mas corta, puesto que cambia su valor cada mS en forma descendente. A(7..0) varía sus cantidades cada 2 mS también de forma descendente. Estas dos variables se suman, y su resultado se observa en S(8..0)a, así que esta variable intermedia de salida cambia su valor cada milisegundo. C(7..0) cambia sus valores cada 3 mS, de forma descendente también, mientras que D(7..0) lo hace cada 2 mS, en forma ascendente. Ambas variables se suman, observándose el resultado en S(8..0)b, cuyo valor cambia cada 2 mS. S(8..0)a y S(8..0)b se restan y su resultado se observa en R(9..0), que

varía cada milisegundo, pues es el periodo mas pequeño que utiliza una de las entradas, B(7..0), y por lo tanto, en este periodo cambia el valor de salida, cuando se compara R(9..0) con M(7..0), el valor del umbral. Esta figura (4.1) permite observar todas las entradas, las salidas intermedias, y detalla el comportamiento de cada uno de los componentes internos del operador, además de la salida del mismo, que es la mas importante, todo esto gracias a que el simulador del paquete de MAX + plus II lo permite. Así, se aprecia como salida toma el valor de 1 cuando resta, R(9..0) con valor hexadecimal 0FF, es comparada con umbral, M(7..0) con valor hexadecimal FF, cumpliéndose la condición salida es 1 cuando R(9..0) ≥ M(7..0), esto es, se satisface el caso "al menos FF de FF" (operador And), ya que, según lo establecido en el capitulo 1, hay un 1 cuando la suma de las entradas excitadoras, S(8..0)a = 1FE hexadecimal, es mayor que la suma de las inhibitorias, S(8..0)b = 0FF hex, y el resultado de la resta, R(9..0), es igual al umbral, M(7..0). Al siguiente cambio en el valor de R(9..0), 0FE, con un mismo valor de M(7..0), se cumple la condición salida es 0 cuando R(9..0) < M(7..0), se cumple el caso "cuando mucho FF de 0FE" (operador Nor) debido a que, según lo expuesto en el capitulo 1, hay un cero a la salida cuando la suma de las excitadoras, S(8..0)a = 1FD hex, es mayor que la suma de las inhibitorias, S(8..0)b = 0FF hex, y el resultado de la resta, R(9..0), es menor que el umbral, M(7..0).

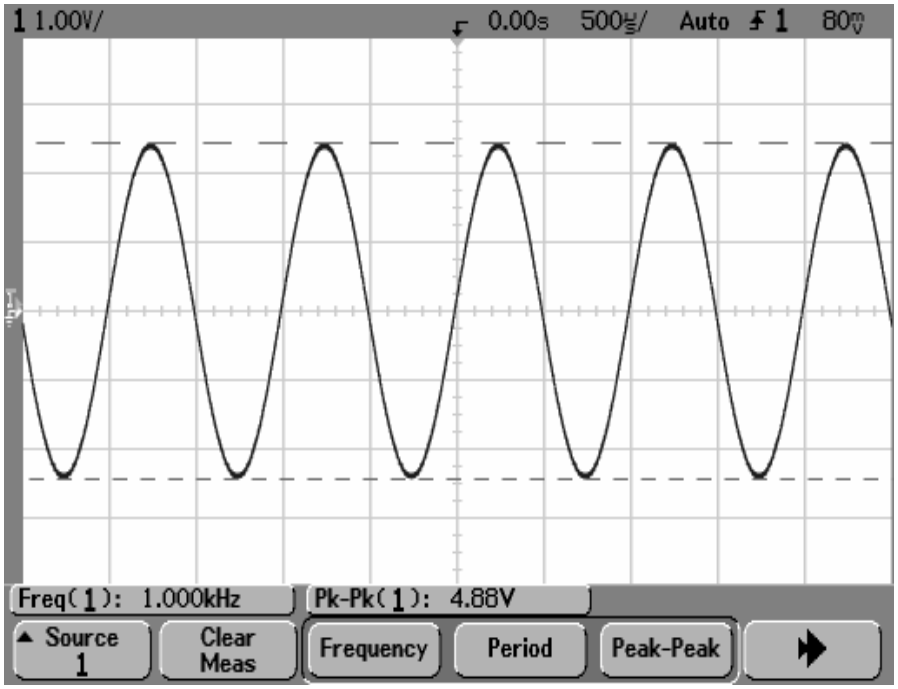


Figura 4.2. Señal analógica aplicada a las cuatro entradas analógicas del microcontrolador.

Por otro lado, en la figura (4.2) se observa un oscilograma que muestra una señal analógica del tipo senoidal de 4.88 volts pico a pico y una frecuencia de un KHz, la cual es aplicada a las cuatro entradas analógicas del microcontrolador, AN0, AN1, AN2 y AN3.

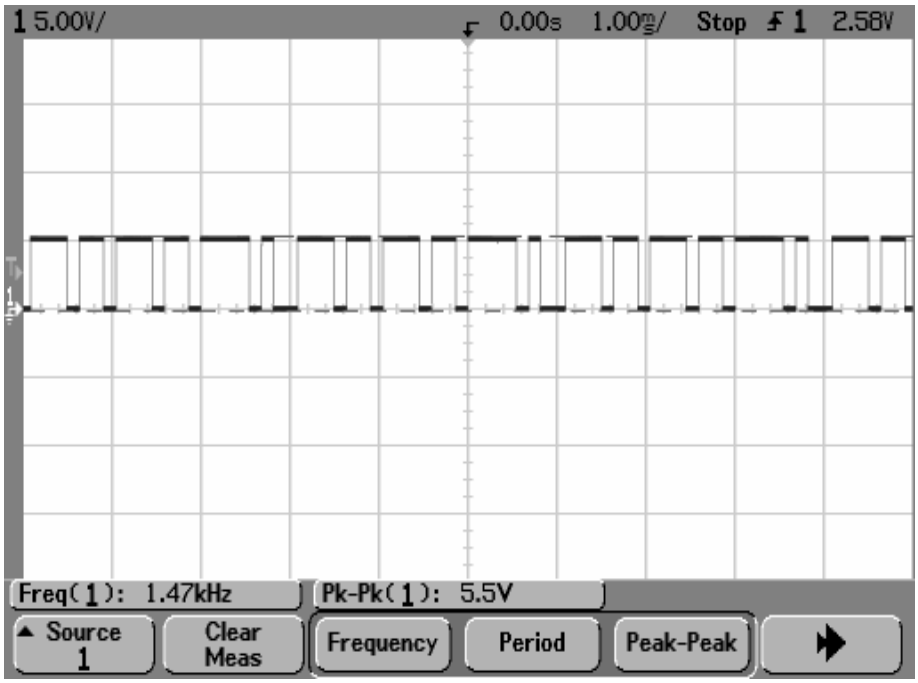


Figura 4.3. Señal resultante a la salida del microcontrolador.

En la figura (4.3) se observa un oscilograma que muestra la salida del dispositivo. La salida, que es la mas útil, nos muestra un dato mas simple, ya que un osciloscopio permite observar un máximo de dos señales, dejando fuera la posibilidad de monitorear los resultados de la conversión de las cuatro señales, la suma de AN0 y AN1, (AN0 + AN1 → AN0), así como la de AN2 y AN3, (AN2 + AN3 → AN2), la resta de AN0 y AN2, (AN0 – AN2 → AN0). Solo la comparación entre el resultado de la resta AN0 y el valor de umbral, DIG1, es lo que se puede apreciar en este oscilograma de la salida. Es posible observar cada una estas cantidades, pero el programa fuente que opere al microcontrolador crecería enormemente, y aun cuando se pudiera desarrollar para ello, en el osciloscopio no se podrían ver todas las cantidades en un mismo tiempo, y se convertiría aquello en una maraña de señales. Aun así, no es complicado interpretar el resultado, como se ve, sumar una señal periódica siempre mostrará resultados, tanto por debajo, como por arriba del umbral establecido. Cuando existan máximos positivos, dado que las cuatro señales son prácticamente iguales, en la resta existirá un valor igual al umbral, DIG1 = "00000000", cumpliéndose la condición salida es 1 cuando AN0(resta) ≥ DIG1(umbral), satisfaciéndose el caso "al menos 00 de 00" (operador And), ya que las entradas excitadoras están inhibidas, y el resultado de la resta es igual al umbral. Y cuando existen valores cercanos o en el cruce por cero, las excitadoras son inhibidas, provocando, las inhibidoras, que en la resta exista un valor menor al umbral, DIG, cumpliéndose la otra condición: salida es 0 cuando AN0(resta) < DIG1(umbral), satisfaciéndose el caso "al menos 00 de 00", el mismo operador And, puesto que las entradas excitadoras continúan inhibidas, pero el resultado de la resta es negativo, por debajo del umbral. Cabe señalar que, en el capitulo anterior, se mencionaron pequeños desfases entre

señales, además de que el tiempo de adquisición de datos para cada señal es distinto, por lo cual, existen resultados de la resta, tanto abajo del umbral, como por arriba, de ahí las oscilaciones observadas en el oscilograma, figura (4.3). Debido a esto, también puede cumplirse el caso "al menos 00 de 01" (operador Or), cuando salida es 1, ya que, según el capitulo 1, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

En esta primera comparación, considerando que, por un lado se trabajan con combinaciones de dígitos binarios en la plataforma de Altera, y por otro se manejan señales analógicas periódicas convertidas a cantidades binarias para ser manejadas por el microcontrolador, se observa un desempeño similar de ambas plataformas. La primera muestra sus datos perfectamente definidos, tanto de las operaciones intermedias como del resultado final, lo cual nos permite comprobar los fundamentos del operador generalizado descritos en este trabajo. La segunda, dada su capacidad de abarcar el campo analógico, muestra su posibilidad de emular al operador generalizado con un conjunto de datos producidos por señales periódicas, lo cual sugiere en mismo valor en un cierto periodo de tiempo, que son más cercanas al mundo real. Por ello, este último dispositivo se percibe más susceptible a ser aplicado en cualquier situación que involucre datos analógicos.

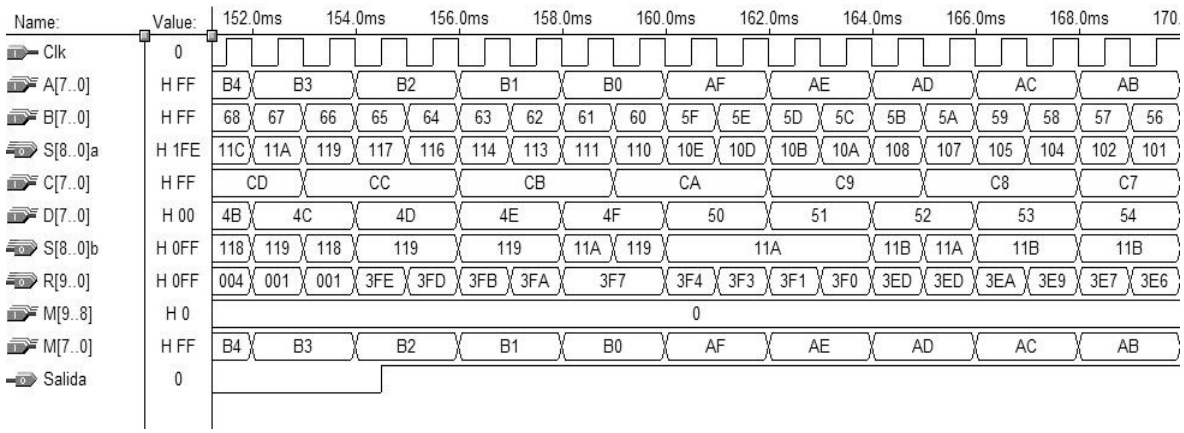


Figura 4.4. Respuesta del operador generalizado implementado con el software MAX + plus II.

La figura (4.4) muestra nuevamente detalle del comportamiento del operador generalizado. En esta, como en la anterior figura, el simulador permite observar los diferentes valores que pueden tomar las entradas, y al ser aplicadas al sistema, las salidas intermedias muestran los resultados. Como se ha descrito, A(7..0) se suma con B(7..0), y el resultado de esta suma se aprecia en la salida S(8..0)a. nuevamente se percibe que esta salida presenta diferentes valores en un periodo igual al de B(7..0), ya que esta entrada cambia su valor cada milisegundo. También C(7..0) se suma con D(7..0), y su resultado se presenta por la salida S(8..0)b, cuyos valores varían en función del periodo de D(7..0) ya que estos cambian cada 2 ms. Ambos resultados de las sumas se restan, y su resultado es mostrado a través de R(9..0), que se compara con M(7..0), valor de umbral, con lo

cual opera salida, que es la mas importante, ya que con ella se monitorea el funcionamiento del operador generalizado, de acuerdo con los fundamentos del mismo. En este caso, salida cambia su valor de 0 a 1; primero, cuando el valor de la resta es 001 en hexadecimal, y el valor de umbral es B3 hexadecimal, salida permanece en cero cumpliéndose la condición salida = 0, cuando R(9..0) < M(7..0), cumpliéndose el caso "cuando mucho B3 de 001" (operador Nor), dado que, según lo establecido en el capitulo 1, hay un cero a la salida cuando la suma de las entradas excitadoras, S(8..0)a = 119 hexadecimal, es mayor que la suma de las inhibidoras, S(8..0)b = 118, y el resultado de la resta, R(9..0) es menor al umbral, M(7..0). Después, cuando resta es 3FE hexadecimal, y umbral es B2 hexadecimal, salida cambia a 1, cumpliéndose la condición salida = 1, cuando R(9..0) ≥ M(7..0), satisfaciéndose el caso "cuando mucho B2 de 3FE" (operador Nor), puesto que, según el capitulo primero, hay un 1 a la salida cuando la suma de las entradas excitadoras, S(8..0)a = 117 hexadecimal, es menor que la suma de las inhibidoras, S(8..0)b = 119, y el resultado de la resta, R(9..0) es mayor al umbral, M(7..0).

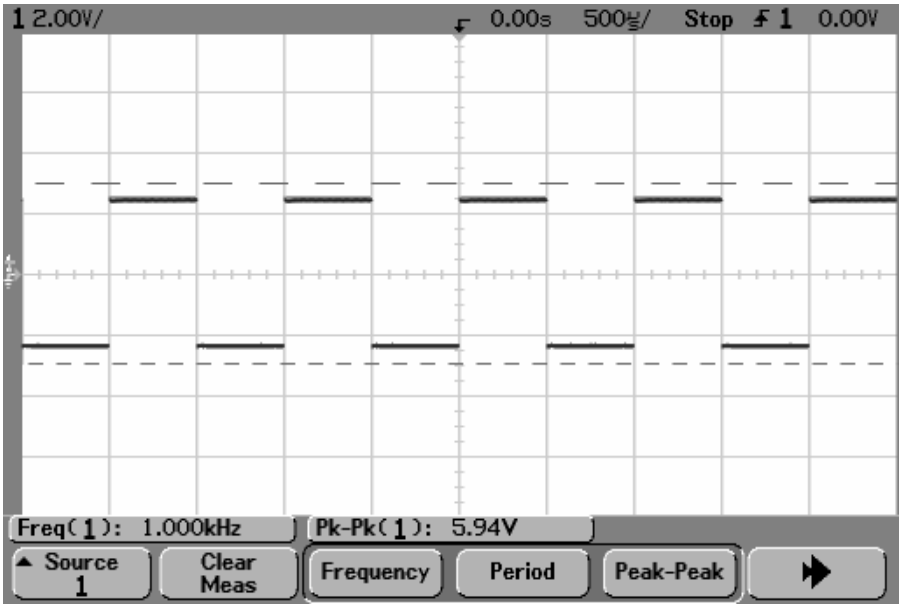


Figura 4.5. Señal cuadrada aplicada a las cuatro entradas analógicas del microcontrolador.

En la figura (4.5) se presenta el oscilograma de una señal cuadrada, de 5.94 volts pico a pico a 1 KHz, aplicada a las cuatro entradas analógicas, AN0, AN1, AN2 y AN3, del microcontrolador.

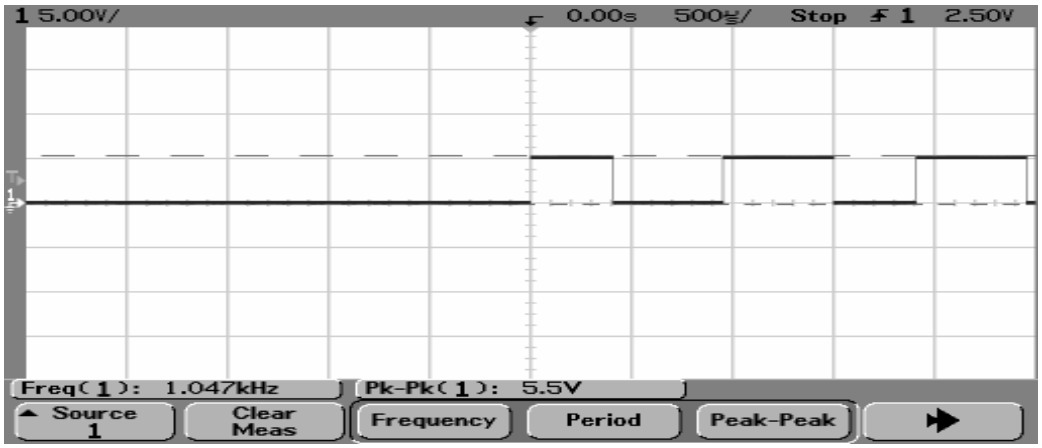


Figura 4.6. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 4.5.

La respuesta del dispositivo se aprecia en la figura (4.6). En este oscilograma se observa el comportamiento del operador generalizado al ser aplicadas cuatro señales cuadradas. La forma de la señal indica que la resta, (AN0 – AN2 → AN0), del resultado de las sumas de las cuatro señales, (AN0 + AN1 → AN0) y (AN2 + AN3 → AN2), está por debajo del valor de umbral, DIG1. Cuando las entradas excitadoras son inhibidas, y las inhibitorias provocan un resultado de la resta, AN0, negativo, este resultado está por debajo de umbral, DIG1 = "00000000", por lo cual salida está en cero; dado que salida = 0, cuando AN0(resta) < DIG1(umbral), se cumple el caso "al menos 00 de 00" (operador And), ya que las entradas excitadoras están inhibidas, y las inhibitorias hacen que el resultado de la resta sea negativo, es decir, que esté por debajo del umbral, de acuerdo con el capitulo 1. Posteriormente, cuando aparecen los máximos positivos, estos se suman dando un valor por encima de umbral, DIG1 = "00000000", y al restarse, presenta un valor de cero, ya que las cuatro señales son prácticamente iguales, por lo tanto salida = 1, cuando AN0(resta) ≥ DIG1(umbral), satisfaciéndose el caso "al menos 00 de 00", teniéndose el mismo operador And. Aquí también, las entradas excitadoras están inhibidas, pero el resultado de la resta es igual al umbral, según lo especificado en el capitulo 1. En este caso, nuevamente, se carece de resultados intermedios, solo el resultado final del funcionamiento del operador. Esto impide verificar los resultados internos, tanto de las sumas, como de la resta.

Como en la anterior situación, citadas también en el capitulo anterior, se mencionaron pequeños desfases entre señales, además de que el tiempo de adquisición de datos para cada señal es distinto, por lo cual, existen resultados de la resta, tanto abajo del umbral, como por arriba, de ahí las oscilaciones observadas en el oscilograma, figura (4.6). Debido a esto, también puede cumplirse el caso "al menos 00 de 01" (operador Or), cuando salida es 1, ya que, según el capitulo 1, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor al umbral.

Igual que en la comparación precedente, ambos resultados muestran el comportamiento del operador generalizado, desde las diferentes perspectivas que

las plataformas de desarrollo ofrecen. Nuevamente, Altera, y su MAX + plus II, permiten observar a detalle cada una de las entradas, y los valores que estas toman, así como las salidas intermedias, los resultados de las sumas, y el de la resta, cuyo valor es comparado con el umbral, cuyo resultado se observa en la salida, que permite monitorear el comportamiento del dispositivo. Microchip, por su parte, va mas allá de las combinaciones binarias que pueden estar en sus entradas, tomando señales analógicas y, después de convertirlas a datos binarios, operar con estos, mostrando una salida mucho más cercana a los sistemas electrónicos en conjunto, sin diferenciar lo analógico de lo digital.

Ambas plataformas de desarrollo son aplicables, ya que si se trabaja con la tarjeta de desarrollo de Altera, puede abarcar la parte analógica como lo hace el PIC 16F877A, utilizando para ello un convertidor analógico-digital. Con ello, esta opción adquiere una importancia relevante al momento de decidir con que dispositivo trabajar, para implantar al operador generalizado. En cuanto a la salida, al igual que sucede con el microcontrolador, es solo cuestión de utilizar un osciloscopio para verificar que el dispositivo está operando de acuerdo con los criterios de funcionamiento del operador.

En cuanto a la visualización de los resultados, ambos dispositivos pueden ser conectados a una PC a través del puerto serial, utilizando un MAX 232 para la transmisión serie. Con esto, se pueden observar los resultados utilizando el monitor de alguna PC o lap top. Para lograr esta visualización se trabaja con un lenguaje de programación del tipo visual, tal como Visual Basic, que puede hacer de interfaz para monitorear y verificar que el operador generalizado trabaja de acuerdo con sus principios de operación. La única desventaja de esto es que el puerto serie puede trabajar a frecuencias menores a 1 KHz, ya que su respuesta después de este limite es poco confiable. Para trabajar con esta opción, quien sufre algunos arreglos es el programa de operación del microcontrolador, ya que este debe trabajar una transmisión serie entre sí y la PC. Considerando esto, el arreglo en el programa es el siguiente:

```
;-------------------------------------------------------------------------------
; Transmisión serial del resultado a la PC
;-------------------------------------------------------------------------------
posit   bsf     PORTD,3     ;Se pone a 1 el bit 3 del puerto D, enciende LED que indica
                            ;Inicio de transmisión.
        movlw   d'49'       ;Se carga en w un 49 decimal, en ASCII = 1.
espera  btfss   pir1,txif   ;Verifica que no haya dato en el bufer de Tx, brinca si es 1.
        goto    espera      ;Hace un loop.
        movwf   txreg       ;Se carga el resultado final del pin de salida ="1" en txreg,
                            ;para transmitirse a la PC.
esper2  btfss   pir1,txif   ;Espera a que se termine la Tx, brinca si es 1.
        goto    esper2      ;Hace un loop.
        bcf     PORTD,3     ;Apaga LED de Transmisión, poniendo a cero el bit 3 de D.
        goto    consulta    ;Salto a rutina.

negat   bsf     PORTD,3     ;Enciende LED de Transmisión, poniendo a 1 bit 3 de D.
        movlw   d'48'       ;se carga en w un48 decimal, en ASCII = 0.
esper4  btfss   pir1,txif   ;Verifica que no haya dato en el bufer de Tx, brinca si es 1.
```

```
        goto    esper4      ;Hace un loop.
        movwf   txreg       ;Se carga el resultado final del pin de salida ="0" en txreg,
                            ; para transmitirse a la PC.
esper5  btfss   pir1,txif   ;Espera a que se termine la Tx, brinca si es 1.
        goto    esper5      ;Hace un loop.
        bcf     PORTD,3     ;Apaga LED de Transmisión, poniendo a cero el bit 3 de D.
        goto    consulta    ;Salto a rutina.
```

Este bloque opera de acuerdo con el resultado de la comparación entre AN0, resta, y DIG1, umbral. Así, dependiendo si el resultado es negativo o positivo, envía un 1 ó 0 hacia el registro de transmisión del microcontrolador, verificando que no haya otro dato en este, transmitiéndolo posteriormente. Esta operación se repite en ambas situaciones, cambiando únicamente el dato a ser enviado. Para indicar que existe una transmisión de datos, se maneja el bit 3 del puerto D, que enciende, o apaga, un led puesto para este fin.

```
;-------------------------------------------------------------------------------
; Realiza un chequeo para consultar si se sigue realizando el programa
; o se detiene
;-------------------------------------------------------------------------------
consulta btfss   pir1,5     ;Verifica si hay interrupción en el registro pir1, brinca si es 1
        goto    consulta    ;Hace un loop.
        movf    rcreg,0     ;Se carga en w lo que hay en recreg.
        movwf   respc       ;Se carga en respc lo que hay en w.
        btfss   respc,0     ;Verifica si hay dato en respc, brinca si es 1.
        goto    inicio      ;Salta a rutina.
        goto    sigue       ;Continua con rutina.
```

Esta rutina maneja la opción de continuar transmitiendo serialmente datos hacia la PC, o bien termina esa operación, reiniciando con la captura de señales analógicas.

Cabe señalar que, dado que se opera con otra opción de trabajo del microcontrolador, transmisión serie, al inicio del programa se deben invocar los registros necesarios para manejar dicha opción.

```
intcon  equ     0b      ;Habilitación del registro de interrupción INTCON
pir1    equ     0c      ;Habilitación del registro de interrupción externa PIR1
rcsta   equ     18      ;Habilitación del registro de control y estatus en la Rx serial
txreg   equ     19      ;Habilitación del registro Tx de la USART
rcreg   equ     1a      ;Habilitación del registro Rx de la USART
pie1    equ     8c      ;Habilitación del registro de interrupción de perifericos
txsta   equ     98      ;Habilitación del registro de control y estatus en la Tx serial
spbrg   equ     99      ;Habilitación del registro de control de velocidad de Tx
txif    equ     4       ;Posición del bit dentro del registro PIR1
respc   equ     2B      ;Definición de variable sobre un registro
```

Finalmente, la declaración de en qué puerto y como ha de desarrollarse la transmisión serie, se muestra a continuación:

```
        movlw   b'10000111'  ; Se carga en el registro de trabajo "w" el dato binario
        movwf   trisc        ; para definir a RC7=Rx, RC6=Tx, en la transmisión, y
```

```
                                ;RC5,RC4,RC3 como salidas para verificación Led, en el
                                ;puerto C.
;-------------------------------------------------------------------------------------------------------------------
        movlw   d'10'           ;Se carga en el registro de trabajo "w" el dato decimal
        movwf   spbrg           ;para indicar una tasa de Tx = 115200 bps (8,n,1) con reloj
                                ; de 20MHz, en spbrg.
        movlw   b'10100100'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   txsta           ;para indicar un USART asíncrono para Tx de alta tasa de
                                ; baudios, en txsta.
        movlw   b'01110000'     ;Se carga en el registro de trabajo "w" el dato binario
        movwf   pie1            ;para indicar en pie1 la activación de banderas de
                                ;interrupción para la Tx y la Rx
```

Con esto, se configura el puerto y se define a través de que pines se envía el dato serial, además, se establece la velocidad y que tipo de transmisión es, habilitándose las interrupciones necesarias para ello.

Otra opción de visualización es utilizar un convertidor digital-analógico, DAC, lo que permite utilizar una tarjeta de captura de audio, que abarca una amplia gama de frecuencias, y con la cual verificar el comportamiento de operador generalizado. A ambos dispositivos se les puede acoplar este convertidor, y utilizar la opción de la tarjeta, como la que ya se ha mencionado. Para Altera, simplemente se definen 8 salidas, iguales todas, ya que el dato a representar es 1 ó 0, valores extremos, por lo cual es entonces, 255 ó 0, valores extremos en 8 bits. El DAC, toma cada uno de estos valores, convirtiéndolo a su correspondiente valor analógico, igualmente, valores extremos. Cosa similar para el microcontrolador, pues solo se debe configurar un puerto completo para poner todos sus pines en 1 ó 0, dependiendo de la respuesta a presentar. El DAC, nuevamente, recibe cada valor extremo, 255 ó 0, convirtiéndolo a su valor analógico correspondiente. Se muestra, a continuación, la rutina para operar el resultado de la comparación en el microcontrolador:

```
;-------------------------------------------------------------------------------
; Transmisión del resultado
;-------------------------------------------------------------------------------
posit   bsf     PORTA,5         ;Pone a 1 el pin 5 del puerto A, enciende LED de
                                ;operación.
        movlw   0xff            ;Se carga en "w" el dato binario ff
        movwf   PORTD           ;para sacar "unos" a través del puerto D.
        bcf     PORTA,5         ;Pone a cero el pin 5 del puerto A, apaga LED de
                                ; operación.
        goto    sigue           ;ir a etiqueta.

negat   bsf     PORTA,5         ;Pone a 1 el pin 5 del puerto A, enciende LED de
                                ; operación.
        movlw   0x00            ;Se carga en "w" el dato binario 00,
        movwf   PORTD           ;para sacar "ceros" a través del puerto D.
        bcf     PORTA,5         ;Pone a cero el pin 5 del puerto A, apaga LED de
                                ; operación.
        goto    sigue           ;ir a etiqueta.
```

El bloque muestra la rutina que controla los pines de salida, de manera que se

tengan los valores extremos, si salida toma el valor de 1, entonces se tiene en el puerto D el dato de 255, o FF en hexadecimal, todos "unos". De lo contrario, con salida en cero, el puerto D presenta el dato 0, o 00 hexadecimal, todos "ceros". Sobre este puerto D se acopla el DAC para convertir estos datos a su contraparte analógica.

Para señales con frecuencias superiores a 5 MHz, la opción de Altera no sería la adecuada, ya que el periodo mínimo que puede tomar un pulso de reloj es de 200 ns, es decir, 5 MHz, aunado a esto, la frecuencia de operación máxima para este diseño es 1 KHz, después de ahí, los datos pueden no ser correctos, provocando retardos internos, o "glitches", que pueden afectar la salida del sistema. Para el caso de Microchip, el microcontrolador puede trabajar hasta en 20 MHz, pero dado que el tiempo de adquisición de la señal, y el posterior muestreo, o conversión de esta a datos binarios, está tasada en 1.6 µs y 19.72 µs, respectivamente, recomendado en el manual de operación del PIC 16F877A, se puede trabajar con señales de hasta 40 KHz, garantizando la no perdida de datos de las señales.

Como se ha visto en este capitulo, ambas opciones son confiables para implantar al operador generalizado, y dependerá de la capacidad del diseñador decidir por cual de ellas se inclina, puesto que, para implantar al operador generalizado en el microcontrolador, es necesario tener conocimientos de programación en lenguaje ensamblador, particularmente, de los PIC's de Microchip, que, además, ofrecen otros lenguajes, tales como PBasic y C. Si no es así, la opción de Altera es la mas indicada, ya que es cuestión solo de crear el esquemático del operador, y grabarlo en el dispositivo. Además de ello, si se tienen conocimientos de algún lenguaje de descripción de hardware, tal como verilog o vhdl, Altera también es una buena opción, ya que al compilar el programa se genera un archivo que permite programar el funcionamiento del dispositivo.

Conclusiones

C.1.- Consideraciones sobre el diseño

Como se ha visto en los capítulos correspondientes, el desarrollo de este proyecto se ha basado en diferentes plataformas de trabajo, las cuales arrojan diferentes ventajas y desventajas. A continuación, se mencionan todas aquellas que se tuvieron que considerar al momento de plantear el proyecto, objetivo de este trabajo, la implantación del operador generalizado.

Ambas plataformas, MAX + plus II de Altera, figura (5.1), y MPlab de Microchip, figura (5.2), son paquetes de software que pueden ser conseguidos desde las paginas correspondientes a dichas empresas. Por esta parte, ambas están disponibles para el usuario. Por el contrario, para el sistema de Altera, es necesario conseguir la tarjeta de desarrollo, la cual es un poco complicado de "ordenar", toda vez que es necesario escribir un correo para ponerse en contacto con el representante de Altera en nuestro país, ordenar el producto, en este caso el paquete completo de desarrollo, y pagar el costo de este, además de los impuestos por importación y de paquetería para el envío a domicilio. Todo esto utilizando tarjetas de crédito. Para el caso de Microchip, es mucho más fácil contar con la herramienta, ya que el grabador del dispositivo a utilizar se encuentra fácilmente en la página misma, o en alguna otra que se dedique a difundir circuitos eléctricos para el armado, en este caso, de grabadores de PIC´s. El único detalle es conseguir los componentes básicos, que son los mínimos, utilizando el puerto serie de cualquier PC, o Lap Top.

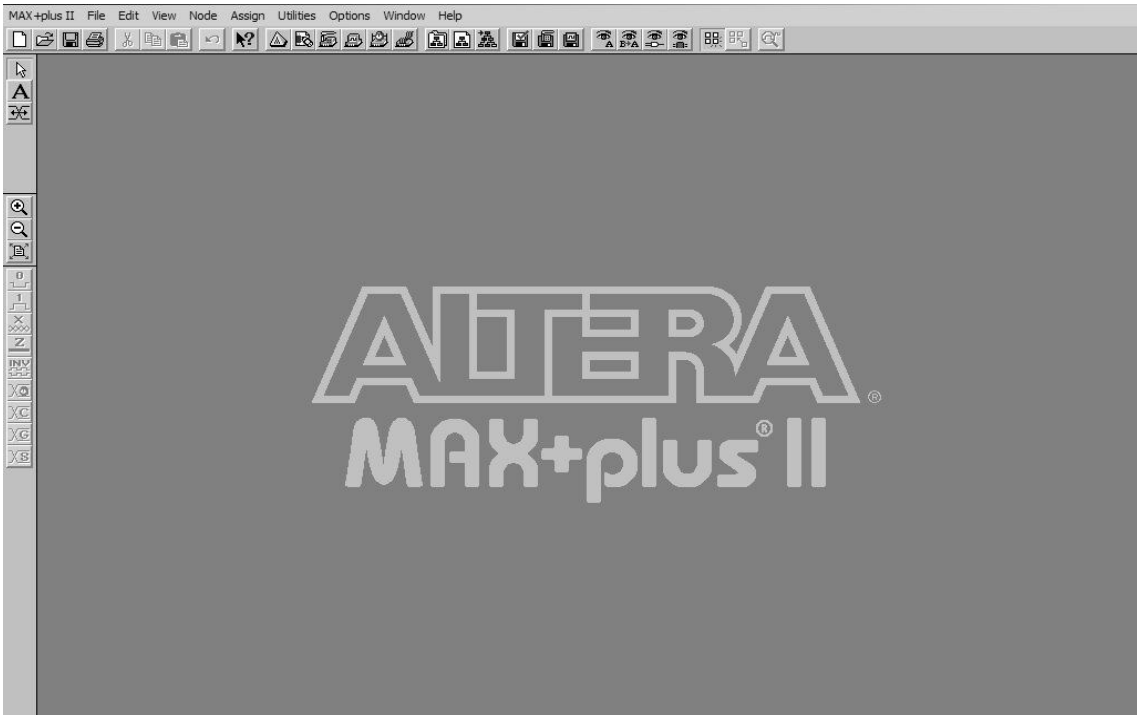


Figura 5.1. MAX + Plus II de Altera.

En cuanto a los ambientes que proporcionan estos software, el ofrecido por Altera es muy útil, ya que este permite el desarrollo de sistemas utilizando el ambiente esquemático, esto es, cuenta con librerías de dispositivos TTL, 74xx y 54xx, para "dibujar" el circuito resultante de nuestro análisis aplicando las bases de diseño digital. Además de ello, utilizando el esquema final, permite crear bloques con este mismo elemento para lograr otros de mayor tamaño, conteniendo como base el ya mencionado. Otro de los aspectos útiles de esta paquetería, es que se pueden diseñar aplicaciones desde el punto de vista de lenguajes de descripción de hardware, es decir, programas que describen el funcionamiento de un circuito, el cual, genera un bloque que puede ser grabado en la tarjeta de desarrollo para una aplicación particular, o bien, como elemento de una aplicación mayor. Además de ello, incluye un simulador grafico que muestra tanto las señales de entrada, como de salida, que permite monitorear el comportamiento del sistema antes de ser grabado en el dispositivo. Esto permite simular los valores que las entradas pueden tomar, y muestra el comportamiento de la o las salidas a partir de estas primeras. Estas son las características mas útiles y principales con que cuenta esta paquetería de la plataforma de Altera.

Figura 5.2. MPLab de Microchip.

Por otro lado, en el caso de la plataforma de Microchip no es tan rica como la de Altera, toda vez que la única forma de hacer operar a un microcontrolador es a

través del su lenguaje ensamblador, aunque últimamente ya existen otros que se pueden aplicar, tales como C, PBasic, entre otros. Pero aun así, se sigue trabajando en un listado que refleja el algoritmo con el cual el dispositivo se ha de guiar para realizar las rutinas y operaciones para las cuales ha sido considerado. Además, dado que se trata de un dispositivo digital, el simulador muestra los registros involucrados en la operación del dispositivo, y las cantidades que contienen estos en cada uno de los momentos en los cuales operan en las diferentes rutinas que los involucran. Esto es suficiente cuando se trabajan con entradas digitales, pero si se trata, como lo es en nuestro caso, de un dispositivo que además cuenta con un convertidor analógico-digital, es difícil visualizar el comportamiento que este tendría cuando se aplican señales analógicas.

Como el objetivo del proyecto es el de tomar señales continuas en el tiempo para que opere el operador generalizado, el microcontrolador PIC 16F877A es muy útil ya que cuenta con un convertidor analógico-digital que realiza la tarea de muestreo de este tipo de señales, y posteriormente, el algoritmo de operación interno, realiza las operaciones necesarias con estas señales ya digitalizadas. Esta es un a desventaja para la plataforma de Altera, ya que el diseño desarrollado bajo esta se limita tan solo a la operación de únicamente señales digitales, lo que le reduce su aplicación solo a este campo de las señales binarias. Otra de ellas, para Altera, es que se deben considerar los tiempos de propagación entre componentes, lo cual reduce el ancho de banda dentro del cual puede operar un diseño particular. En el caso de nuestro operador generalizado, fue necesario aplicar un dispositivo retardador que permitiera que todas las salidas de cada uno de los bloques se acoplaran al resultado correcto, para presentar el valor correspondiente. La figura (5.3) muestra un bosquejo de los retardos que se generan en el paso de una señal de entrada hacia la salida.

Delay Matrix

Source \ Destination	S0a	S1a	S2a	S3a	S4a	S5a	S6a	S7a	S8a
A0	12.0ns	12.0ns	12.0ns/19.0ns	19.0ns	19.0ns/26.0ns	26.0ns	19.0ns/33.0ns	33.0ns/54.0ns	26.0ns/54.0ns
A1		12.0ns	12.0ns/19.0ns	19.0ns/26.0ns	19.0ns/33.0ns	26.0ns/33.0ns	19.0ns/40.0ns	33.0ns/61.0ns	26.0ns/61.0ns
A2			12.0ns	19.0ns/26.0ns	19.0ns/33.0ns	26.0ns/33.0ns	19.0ns/40.0ns	33.0ns/61.0ns	26.0ns/61.0ns
A3				12.0ns	19.0ns	26.0ns/33.0ns	19.0ns/40.0ns	33.0ns/61.0ns	26.0ns/61.0ns
A4					19.0ns	26.0ns/33.0ns	19.0ns/40.0ns	33.0ns/61.0ns	26.0ns/61.0ns
A5						12.0ns	12.0ns/19.0ns	26.0ns/40.0ns	19.0ns/40.0ns
A6							12.0ns	26.0ns	19.0ns/26.0ns
A7								12.0ns	12.0ns
B0	12.0ns	12.0ns	12.0ns/19.0ns	19.0ns	19.0ns/26.0ns	26.0ns	19.0ns/33.0ns	33.0ns/54.0ns	26.0ns/54.0ns
B1		12.0ns	12.0ns/19.0ns	19.0ns/26.0ns	19.0ns/33.0ns	26.0ns/33.0ns	19.0ns/40.0ns	33.0ns/61.0ns	26.0ns/61.0ns
B2			12.0ns	19.0ns/26.0ns	19.0ns/33.0ns	26.0ns/33.0ns	19.0ns/40.0ns	33.0ns/61.0ns	26.0ns/61.0ns
B3				19.0ns	19.0ns/26.0ns	26.0ns/33.0ns	19.0ns/40.0ns	33.0ns/61.0ns	26.0ns/61.0ns
B4					12.0ns	26.0ns/33.0ns	19.0ns/40.0ns	33.0ns/61.0ns	26.0ns/61.0ns
B5						19.0ns	12.0ns/26.0ns	26.0ns/47.0ns	19.0ns/47.0ns
B6							12.0ns	26.0ns	19.0ns/26.0ns
B7								19.0ns	12.0ns/19.0ns

Figura 5.3. Tabla parcial que muestra los retardos que se generan en el sistema realizado para el operador generalizado.

En esta figura (5.3) se observa la parte del primer sumador de 8 bits, figura (5.4). Los bits menos significativos, A0 y B0, son los que menos tiempo tardan en operarse y salir en S0a, ambos utilizan 12 ns, y conforme aumenta el número de elementos básicos a través de los cuales deben pasar los datos binarios, figura (5.4), el tiempo de retardo aumenta. Así por ejemplo, el dato de A0 tarda en llegar hasta la última salida, S8a, en un periodo máximo de 54 ns. También se observa que A1 tarda hasta 61 ns para llegar a S8a. Esto implica que el primer sumador de 8 bits utiliza 61 ns para realizar una operación de este tipo, por lo tanto, los datos pueden estar variando a una frecuencia de hasta 16.4 MHz. Lo mismo sucede con el otro sumador de 8 bits, por lo cual ambos sumadores, en paralelo pueden operar hasta esta frecuencia sin problemas de retardos. Pero como la siguiente etapa es un restador, el cual destina un tiempo similar en la operación de los datos, la frecuencia de operación se reduce a 8.2 MHz. Por lo anterior, trabajar con datos cuya frecuencia sea menor a 8 MHz, o cuyo periodo sea de al menos 61 nanosegundos, garantiza la no perdida de datos por retardo.

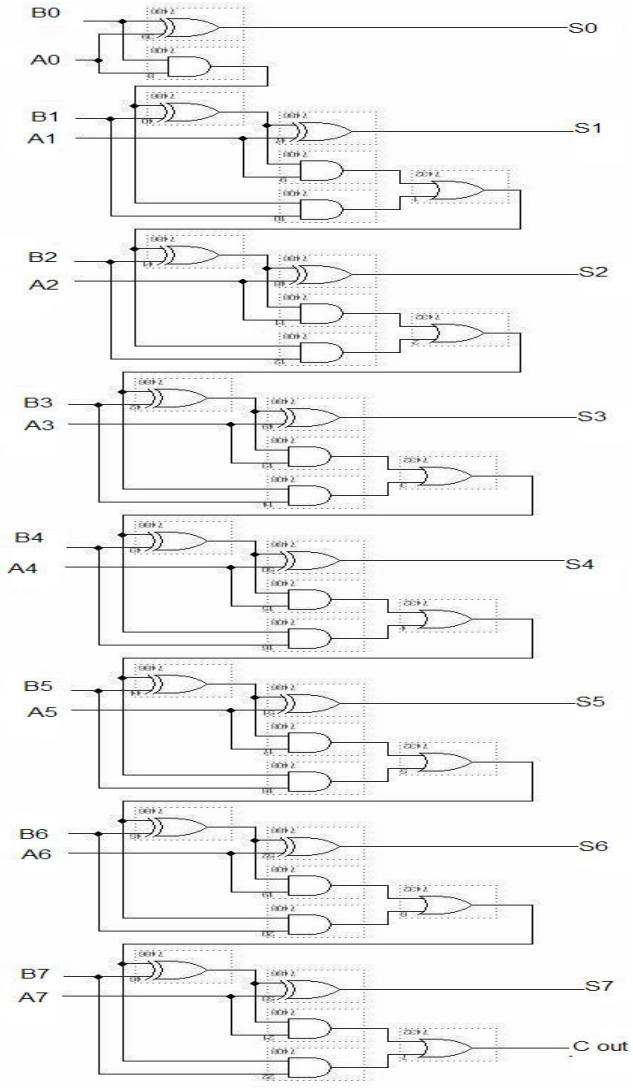


Figura 5.4. Circuito sumador de 8 bits.

Basados en estos resultados, fue necesario entonces acoplar en el sistema un elemento retardador, un flip-flop tipo D, figura (5.5), que pasa los datos cada milisegundo, lo que garantiza que no existan retardos en todas las señales involucradas. Esto evita que se presenten los conocidos "glitches", datos erróneos que existen a las salidas cuando los datos aun no se han estabilizado. De acuerdo a esto, se estableció una frecuencia máxima de operación del diseño en 1 KHz, para garantizar datos correctos en las operaciones.

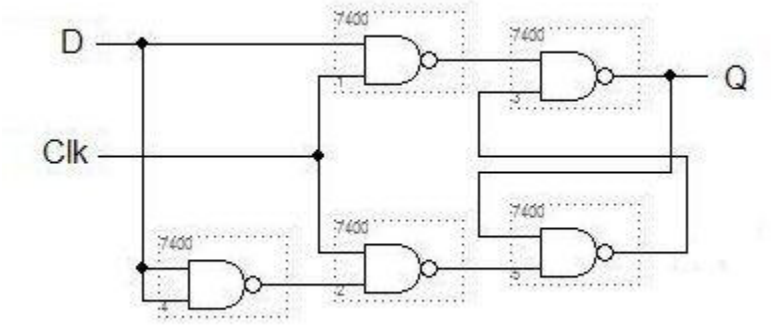


Figura 5.5. Flip-flop tipo D utilizado para retardar los datos en 1 ms.

Este inconveniente no se presenta en un microcontrolador, ya que este elemento, particularmente el PIC 16F877A, puede operar a frecuencias de hasta 20 MHz, y dado que las realiza en bloque de ocho bits, es prácticamente muy rápida la operación y manipulación de las señales obtenidas por el convertidor analógico-digital, que es el único elemento que puede reducir la velocidad de operación, ya que la adquisición tiene un tiempo de 19.74 µS y el de conversión con un tiempo de 1.6 µS. Este es el único inconveniente, si se trata de rapidez en la operación, pero para frecuencias de hasta 40 kHz es operable todavía, garantizando la no perdida de datos.

Para visualizar los resultados obtenidos en la implantación del operador generalizado, es muy fácil observar los obtenidos utilizando la plataforma de desarrollo de Altera, figura (5.6), mientras que para el desarrollado con la de Microchip fue necesario utilizar un osciloscopio, toda vez que se intentó crear una interfaz gráfica con la PC, utilizando Visual Basic, que finalmente no permitió observar los resultados, ya que, al utilizar el puerto RS-232, impedía manejar frecuencias como las de las señales periódicas de entrada, utilizadas para verificar la operación del microcontrolador, que eran de 1 kHz.

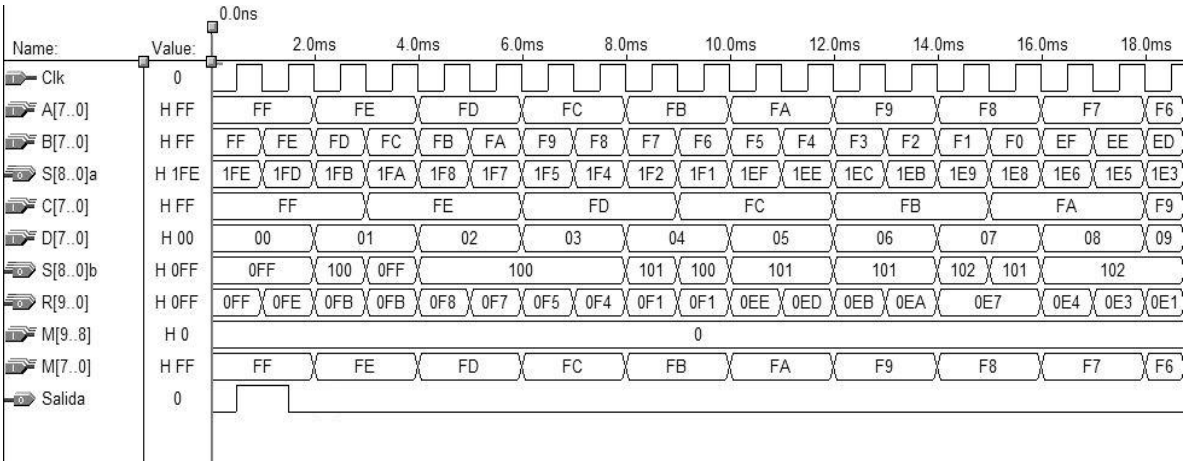


Figura 5.6. Resultados del operador generalizado utilizando la plataforma de desarrollo de Altera. En estas graficas se pueden observar las entradas, salidas, y los valores intermedios de las operaciones realizadas.

Al observar las figuras (5.6) y (5.7), se nota que el paquete de Altera, (5.6), ofrece la posibilidad de tener presentes las entradas, valores intermedios y salidas de la aplicación realizada, en nuestro caso, el functor lógico. Mientras que en el paquete de Microchip, es necesario echar mano de algún equipo visualizador para ver los resultados, particularmente, un osciloscopio permitió apreciar los datos. Un dispositivo de estos solo permite ver dos señales, lo que reduce la posibilidad de revisar todas las señales involucradas en el sistema desarrollado.

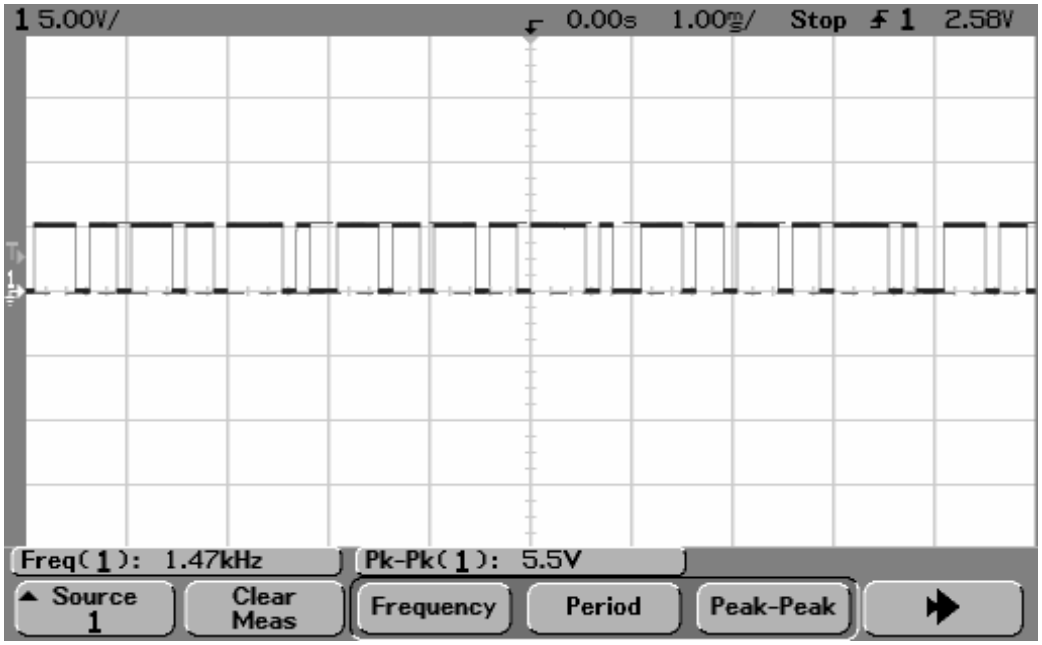


Figura 5.7. Resultados del operador generalizado desarrollado con el microcontrolador PIC 16F877. La plataforma no permite visualizar simulaciones, por lo que es necesario utilizar dispositivos de visualización externos.

Cabe señalar que el paquete de aplicación de Altera permite hacer las simulaciones sin necesidad de utilizar la tarjeta de desarrollo, lo cual reduce el tiempo invertido al trabajo en solo probar y simular, sin necesidad de grabar el dispositivo físicamente. Lo que no sucede con Microchip, ya que al trabajar con el convertidor analógico-digital, forzosamente se debe grabar el microcontrolador con la rutina de operación para comprobar que efectivamente se está desarrollando la conversión descrita en el algoritmo diseñado para el efecto, considerando que las señales a utilizar son periódicas, lo que requirió el uso de una interfaz de visualización, como Visual Basic, y finalmente el uso del osciloscopio para comprobar el correcto funcionamiento del dispositivo. Con datos digitales no es tan complicado, ya que el simulador permite probar su correcta operación; pero no se tienen graficaciones, sino más bien, datos contenidos en registros involucrados dentro del algoritmo, en lenguaje ensamblador, que describe el funcionamiento del sistema.

Para quienes no están muy familiarizados con lenguajes de programación en ensamblador, la plataforma de Altera es una opción muy fácil de manejar, ya que, como se mencionó al principio, se pueden hacer diseños de forma gráfica, a través de esquemáticos que describen la operación del sistema. Para quienes están familiarizados con los lenguajes de descripción de hardware, como verilog, Altera es una buena opción para el desarrollo de sistemas y proyectos. Su única desventaja es que no se incluye un convertidor analógico-digital, lo que se reduce a solo sistemas digitales; para cubrir la parte analógico-digital del proyecto, se puede agregar un dispositivo de este tipo. Además, las frecuencias de operación no deben estar por arriba de 5 MHz para evitar "glitches" en la salida.

En cuanto a los ya inmersos en el ambiente de los lenguajes ensambladores, es obvio que el uso de los microcontroladores es la opción mas rentable ya que son dispositivos cuyas frecuencias de operación y su capacidad de manipulación de la información les hacen sobresalir entre las otras opciones en cuanto a herramientas de desarrollo se refiere.

Ambas opciones de desarrollo su aplicables, como se ha visto en el comparativo de resultados en estas plataformas, ya que cumplen con los requerimientos que los fundamentos establecidos para el correcto funcionamiento del operador generalizado establecen. Por ello, la elección estará en función de la cantidad de operadores generalizados que se requieran aplicar, para lo cual Altera es buena opción, ya que un solo dispositivo puede albergar varios de estos diseños, existiendo, incluso, dispositivos de diferentes capacidades, la única complicación sería la aplicación con señales periódicas analógicas, lo que implicaría manejar un elemento extra, el convertidor analógico-digital. Esto no sería complicado utilizando los dispositivos de Microchip, pero el uso de estos crecería en función de cuantos operadores son necesarios en la aplicación.

No existe selección establecida, solo después de observar las características y requerimientos del problema se puede tomar una u otra opción.

C.2.- Comentarios concluyentes

De acuerdo con los fundamentos establecidos para la operación del functor, y considerando los resultados obtenidos al implantar este operador generalizado en las plataformas de desarrollo descritas en capítulos precedentes, de este trabajo se puede concluir que este operador es una posibilidad de aplicación que reduce a los operadores fundamentales de la lógica dependiendo de la función que se le aplique al umbral. Se puede proponer un algebra generalizada de operadores en la cual se emplee un operador generalizado universal representándolo de la forma siguiente, figura (5.8):

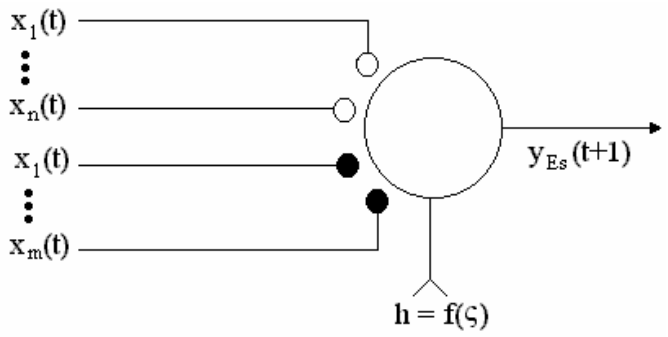


Figura 5.8. Diagrama representativo de un operador generalizado universal.

donde f(ς) es una función genérica de umbral. De esta forma un operador generalizado genérico tendría una función de disparo dada por:

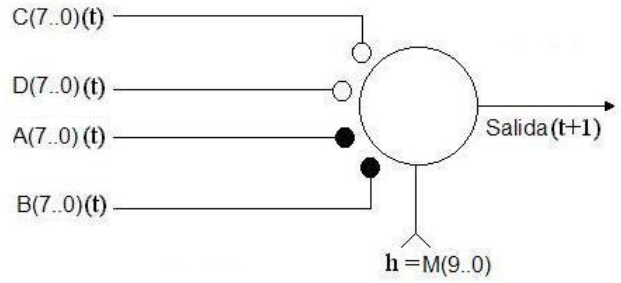
$$D_{Es}x=\begin{cases}1\ si\ \sum_{i=1}^{m}\overset{\bullet}{x_i}(t)-\sum_{i=1}^{m}\overset{\circ}{x_j}(t)\geq f(\varsigma)\\ 0\ si\ \sum_{i=1}^{m}\overset{\bullet}{x_i}(t)-\sum_{i=1}^{m}\overset{\circ}{x_j}(t)<f(\varsigma)\end{cases}$$

Y entonces, f(ς) puede ser una función dependiente del tiempo, una función estocástica, una función constante, etc.

Una sinapsis se le puede interpretar como una representación neuronal de una sentencia del tipo:

$$f(\varsigma)[n(t-1)\equiv x(t)]$$

Así, considerando lo anterior, tenemos que, de la plataforma de Altera, el operador generalizado universal será de la forma siguiente, figura (5.9):



$$Salida\ (t+1) = \begin{cases} 1\ si\ \sum_{i=A}^{B} \bullet\, i(t) - \sum_{j=C}^{D} \circ\, j(t) \geq M(9..0) \\ 0\ si\ \sum_{i=A}^{B} \bullet\, i(t) - \sum_{j=C}^{D} \circ\, j(t) < M(9..0) \end{cases}$$

Figura 5.9. Diagrama representativo del operador generalizado universal desarrollado con Altera. A(7..0) y B(7..0) son las entradas excitadoras, C(7..0) y D(7..0) son las entradas inhibitorias, M(9..0) es el valor de umbral.

Cuyo comportamiento muestra la figura (5.10):

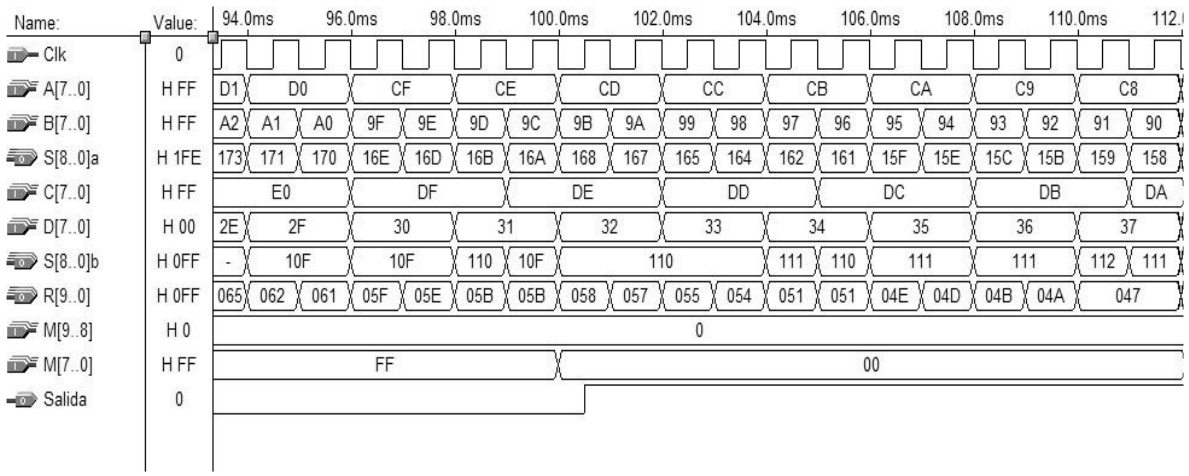
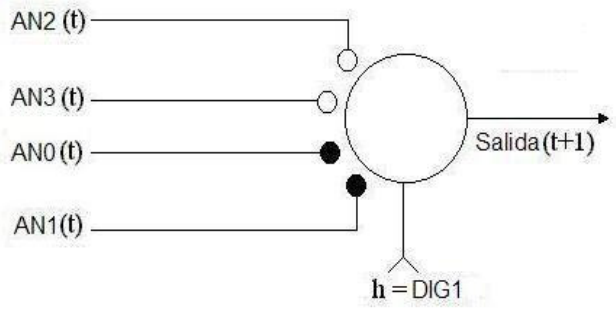


Figura 5.10. Respuesta del operador generalizado con un patrón estocástico a combinaciones binarias en las entradas, y una señal de umbral tipo cuadrada. Detalle para el periodo de 94 ms a 112 ms.

Esta figura (5.10) muestra la operación del functor, sumando las entradas A(7..0) y B(7..0) y las entradas C(7..0) y D(7..0). Estos resultados se restan, observándose el dato en R(9..0), cuyo valor es comparado con la señal de umbral, M(7..0). Se observa que salida toma el valor de 1, con el correspondiente retraso de 500 microsegundos, cuando viene el flanco de subida del pulso de reloj y el comparador toma el dato de R(9..0) con una cantidad 058 en hexadecimal, operándolo con el umbral, M(7..0), ahora en 00 hexadecimal. Cumpliéndose la condición de que salida es 1 cuando R(9..0) ≥ M(7..0), es decir, "al menos 00 de 058" (operador Or), dado que, como se establece en estas conclusiones, hay un uno a la salida cuando la suma de las entradas excitadoras, S(8..0)a = 168 hex, es mayor que la suma de las inhibitorias, S(8..0)b = 110, y el resultado de la resta, R(9..0), es mayor que el umbral, M(7..0). Mientras se mantengan las condiciones mencionadas en cuanto al resultado de las sumas entre las entradas excitadoras e inhibitorias, es decir, excitadoras > inhibitorias, y la salida siga en 1, considerando

el umbral en 00 hexadecimal, y el resultado de la resta tenga cantidades por arriba de M(7..0), el operador trabaja como un conectivo Or. Esto permanece hasta que R(9...0) llegue a este mismo valor de umbral, 00 hex, en este momento, el operador se convierte en un conectivo And, ya que se estará en el caso "al menos 00 de 00". Si resta queda por debajo del umbral, salida toma entonces el valor de cero, convirtiéndose el operador en un conectivo Nor, ya que las excitadoras serían mayores que las inhibitorias, pero el resultado de la resta está por debajo de umbral, como se establece en estas conclusiones. Como se puede apreciar, el operador puede asumir tres funciones distintas solo con variar el resultado de la resta, conectivo Or, And y Nor. Otras situaciones sucederían si se modifica el valor del umbral, o la magnitud de las entradas, tanto inhibitorias como excitadoras.

Por otro lado, considerando lo obtenido con la plataforma de Microchip, el operador generalizado universal será el siguiente, figura (5.11):



$$Salida\ (t+1) = \begin{cases} 1\ si \sum_{i=0}^{1} \overset{\bullet}{A}_i(t) - \sum_{j=2}^{3} \overset{\circ}{A}_j(t) \geq DIG1 \\ 0\ si \sum_{i=0}^{1} \overset{\bullet}{A}_i(t) - \sum_{j=2}^{3} \overset{\circ}{A}_j(t) < DIG1 \end{cases}$$

Figura 5.11. Diagrama representativo para el operador generalizado utilizando el microcontrolador PIC 16F877A. AN0 y AN1 son las entradas excitadoras, AN2 y AN3 son las entradas inhibitorias, DIG1 es el valor de umbral.

Utilizando el diagrama de la figura (5.11), y considerando las dos señales periódicas, que se muestran en la figura (5.12), del tipo triangular, una de 4.801 volts de pico a pico a la frecuencia de 1 KHz, aplicada a la entrada AN0; y otra de 24.2 mv de pico a pico también a 1 kHz, la cual es aplicada a las entradas AN1, AN2 y AN3, se tiene una respuesta en la figura (5.13).

En el oscilograma de la figura (5.13), se observa que salida toma ambos valores de manera periódica. Dado que estas señales periódicas toman valores extremos en el pico de la rampa, es decir, de 1 a 0, entonces, cuando la pendiente cae a cero, un mínimo, se trata de un valor menor al de umbral, DIG1 = "00000000", y al incrementarse la pendiente, hasta llegar a un máximo, y sobrepasar a umbral, DIG1. Así, el resultado de la resta, resta entre AN0, (AN0 + AN1 → AN0), y AN2, (AN2 + AN3 → AN2), (AN0 – AN2 → AN0), queda sobre o bajo el umbral, DIG1, observándose los resultados del oscilograma. Salida es 1 cuando AN0 ≥ DIG1, puesto que AN0 es mucho mayor que las AN1, AN2 y AN3, y al compararse la resta, AN0, con el umbral, DIG1 = "00000000", es mayor que este, aquí se satisface el caso "al menos 00 de FF" (operador Or), ya que, de acuerdo con estas

conclusiones, las entradas excitadoras no están inhibidas, y el resultado de la resta es mayor que el umbral.

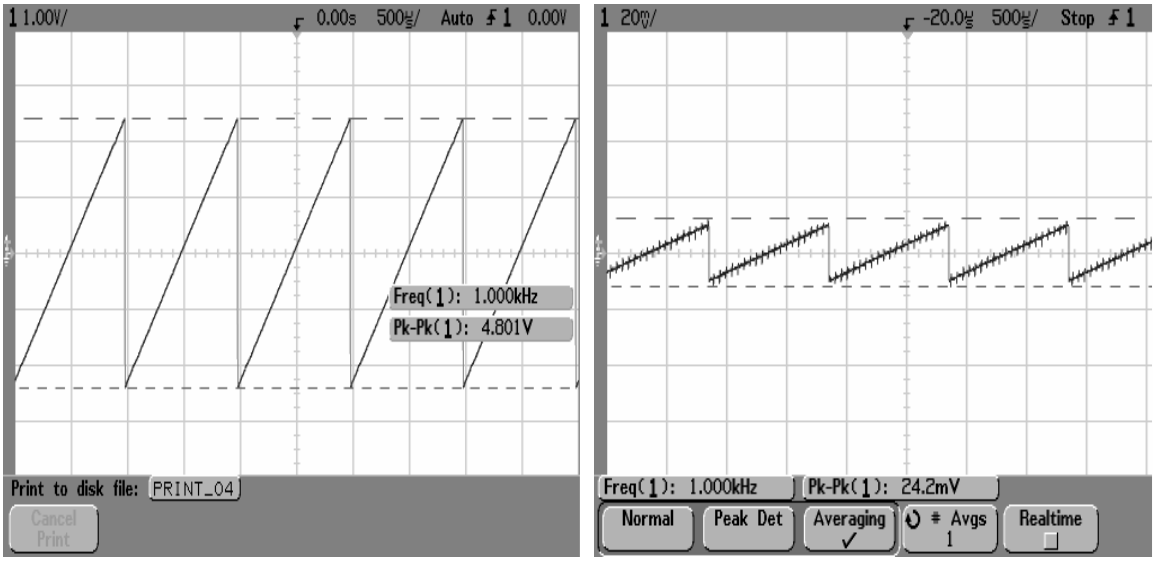


Figura 5.12. Oscilogramas de las señales de entrada. AN0 de 4.8 volts pico a pico. AN1, AN2 y AN3 de 24.2 mv pico a pico. Todas de 1 kHz.

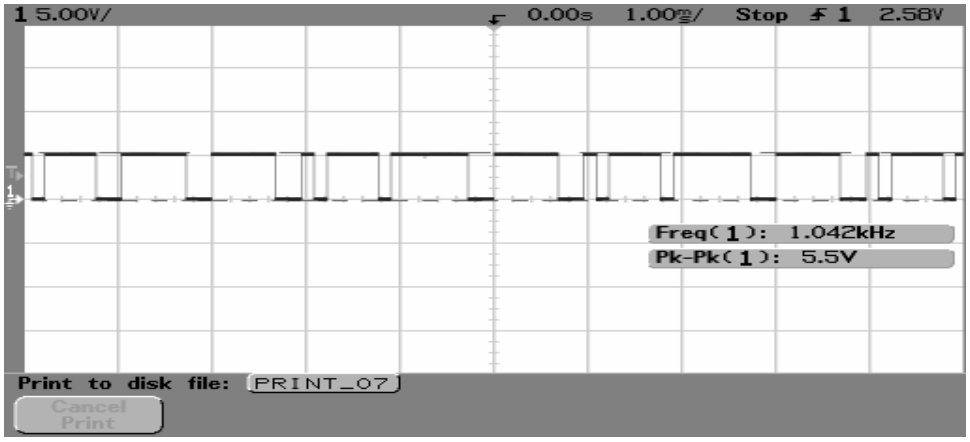


Figura 5.13. Oscilograma resultante al ser aplicadas las señales mostradas en la figura 5.12.

Pero salida toma luego el valor de 0 cuando AN0 < DIG1, ya que AN0 es menor, está cerca o en el cruce por cero, con respecto a AN2 y AN3, y al restarse, AN0 da valores por debajo de cero, es decir, cantidades negativas, que al ser comparadas con el valor de umbral propician la condición de que salida es 0, cumpliéndose el caso "al menos 00 de 00" (operador And), aquí sucede que las excitadoras están inhibidas, y las inhibitorias hacen que el resultado de la resta sea negativo, por debajo de umbral, según las conclusiones ya establecidas. Para esta opción de functor, se observa que los casos se pueden presentar en secuencia, ya que primero se satisface el caso "al menos 00 de FF", el operador trabaja como un

conectivo Or, posteriormente, se presenta el caso "al menos 00 de 00", que supone utilizar al operador como un conectivo And. Se observa que en un mismo evento se pueden involucrar ambos conectivos, ya que mientras AN0 sea menor a DIG1, o negativo, existen ambos casos, Or, "al menos 00 de *nn*", y And, "al menos 00 de 00". Cuando AN0 es igual a DIG1, se tiene el conectivo And, "al menos 00 de 00", y si AN0 aumenta con respecto a DIG1, entonces se tiene el conectivo Or, "al menos 00 de FF". También, al variar el valor de umbral, se pueden presentar los casos comentados.

Para esta plataforma, si todas las señales son iguales, el operador actúa como un conectivo And, pero si alguna de las señales excitadoras es mayor que las demás, el operador actúa como un conectivo Or.

Estos ejemplos descritos, de acuerdo con cada una de las opciones desarrolladas en este trabajo, demuestran que un operador generalizado puede ir mas allá de los conectivos convencionales, ya que dependiendo de la posición o valor que tenga el umbral, y del estado en que se encuentren las entradas, tanto excitadoras como inhibitorias, es como el sistema se comportará. De esta manera, un sistema desarrollado con un operador generalizado universal, es capaz de cambiar en su accionar dependiendo de las características de operación que sean asignadas a dicho sistema, cosa que los conectores convencionales no pueden hacer, toda vez que para lograrlo es necesario reconsiderar el diseño de sistema, y reconstruir el circuito de operación, como se ha visto en la introducción de este trabajo; en este ejemplo fue necesario rearreglar el circuito, reconexionando los conectores utilizados.

Aplicar este operador generalizado universal en el ejercicio descrito en la introducción, permite resolver esos pequeños inconvenientes de diseño al asumir las condiciones sobre las cuales ha de operar el sistema, de tal manera que se adecue a los planteamientos e, incluso, a los replanteamientos, si ha ocurrido alguna omisión al describir el funcionamiento requerido, figura (5.14). Con ello, solo es cuestión de reconsiderar el valor de umbral, para lograr que el mismo conector, el operador generalizado, tenga un comportamiento acorde a la nueva necesidad de trabajo, figura (5.15).

Al comparar las figuras (5.14) y (5.15), se observa que, utilizando el operador generalizado, se puede solucionar el inconveniente. En la figura (5.14) se aprecia el resultado, sin considerar el reajuste, descrito en el ejercicio de la introducción de este trabajo. La figura (5.15) presenta el ajuste para operar con la nueva consideración, planteada en el mismo ejercicio. Nada se ha cambiado, tan solo se ha fijado un nuevo valor de umbral para adecuar el funcionamiento a la nueva consideración de operación del sistema. Se han podido resolver ambas consideraciones del ejemplo, con el mismo operador.

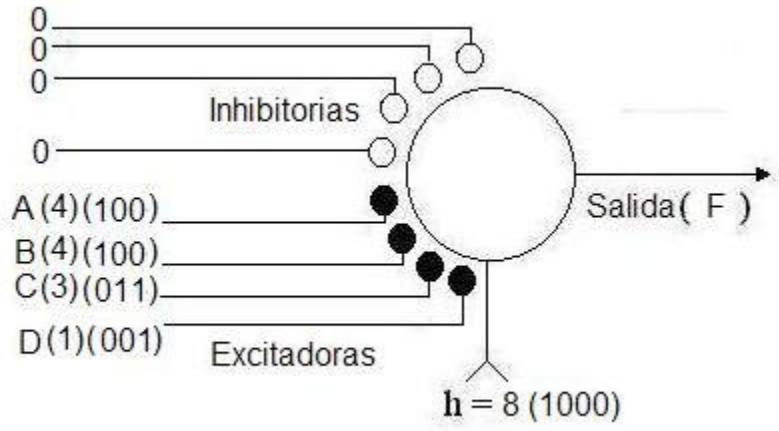


Figura 5.14. Diagrama representativo del sistema diseñado, aplicando el operador generalizado, que satisface F = A(B + CD).

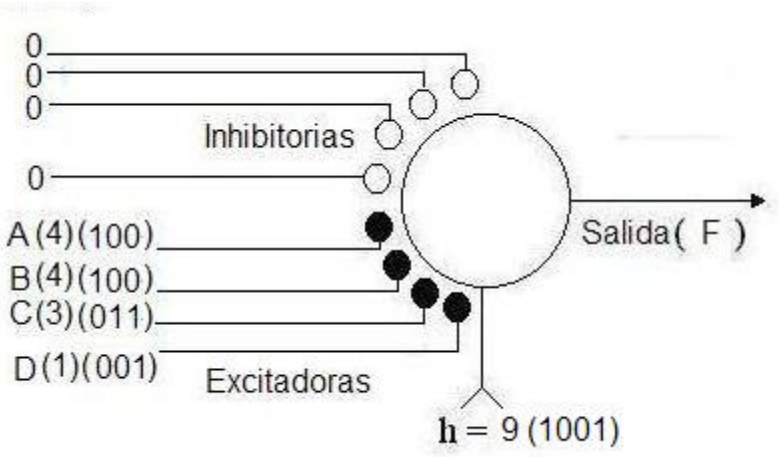


Figura 5.15. Diagrama representativo del sistema diseñado, aplicando el operador generalizado, que satisface F = AB(C +D), corregido para sobrepasar el mínimo necesario.

Las figuras anteriores, (5.14) y (5.15), muestran los diagramas representativos del sistema utilizando el operador generalizado. El proceso es sencillo, el operador suma las 4 entradas, considerando el peso de cada una de ellas. En el planteamiento del problema, en la introducción, se establece que, tanto el director como el jefe de departamento, tienen un voto con un peso de 4, mientras que el voto del profesor pesa 3, y el voto del alumno pesa 1. De acuerdo con estas condiciones, la suma de todos estos votos hará valida la condición de verdadero, cuando se tenga un resultado de la suma igual al umbral, 8, y también por arriba de este, ya que el valor máximo de la suma es 12. Por lo tanto, el operador actúa como un conector Or, ya que se cumple el caso "al menos 8 de 12", pues las entradas excitadoras no están inhibidas, y estas entradas inhibitorias están en cero. La figura (5.15) tiene un idéntico comportamiento, solo que el valor de umbral está situado en 9.

Pero el alcance del operador generalizado no se detiene ahí, ya que si el umbral es dinámico, esto es, está cambiando con el tiempo, este operador es también dinámico, ya que su operación está en función de lo que exista en la entrada de umbral, y del comportamiento de las entradas, tanto inhibitorias como excitadoras. El operador generalizado actuará entonces como un conector And en un instante, pero en el otro, ya será un conector Or, cambiando a un Nor, o algún otro de los casos que la lógica booleana plantea, como se ha visto en el desarrollo con la plataforma de Altera; todo dependiendo del valor de umbral, que rige el comportamiento de este "functor lógico estocástico". Considerando lo anterior, y volviendo al ejercicio de la introducción, se puede hablar de un sistema dinámico que puede adecuarse a las necesidades de cada sesión que realice el comité, ya que se tendrían entonces dos opciones, si los votos deben ser iguales al mínimo necesario, o bien, deben sobrepasar ese mínimo necesario. Además de esto, se pueden abarcar y establecer otros mínimos necesarios sobre este mismo sistema, y seguir utilizándolo en otras situaciones, o sesiones, que tenga dicho comité.

Este ejercicio sencillo es una prueba que los sistemas digitales diseñados para satisfacer necesidades especificas, pueden ser mejorados, u optimizados, si se utiliza este nuevo elemento propuesto, ya que reduce la posibilidad de reacondicionar o reestructurar el circuito, pues aun cuando el problema requiera de un rediseño, este solo se hace en el papel, mientras que el rearreglo del circuito se reduce a la manipulación de alguno de los parámetros involucrados en este operador generalizado universal, los ya mencionados valores de umbral, y las entradas, excitadoras e inhibitorias, como se ha visto en las figuras (5.14) y (5.15). Estas ventajas permiten su aplicación en sistemas digitales, por encima de los conectores convencionales ya conocidos.

Bibliografía

1. *Morris Mano, M.* ***"Lógica Digital y Diseño de Computadores".*** *Prentice Hall, 1982. 636 pp.*

2. *W.S. McCullock & W. Pitts,* ***"A logical Calculus of the Ideas Immanent in Nervous Activity".*** *Bulletin of Mathematical Biophysics, Num. 5, 1943, pag. 115-133.*

3. *Pérez S., J.L. Lara, F. Miranda, A. Garcés, A. Herrera, A. Bañuelos, M. Castillo, J. Quintana, S. Padrón, A.* ***"El functor lógico como elemento básico de una lógica temporal estocástica".***

4. ***"MAX + plus II Getting Started Manual"****. 2001 Altera Corporation.*

5. ***"PIC16F87X Data Sheet. 28/40-Pin 8-Bit CMOS FLASH Microcontrollers Manual"****. 2001 Microchip Technology Inc.*

6. ***"MPLAB® IDE v6.xx QUICK START Manual"****. 2002 Microchip Technology Inc.*

Anexos

Algebra Booleana

El álgebra booleana, puede definirse con un conjunto de elementos, un conjunto de operadores y un número de axiomas no probados o postulados. A continuación se presentan los principales teoremas y postulados del álgebra booleana:

Postulado 2	(a) x +0 = x	(b) x*1 = x
Postulado 5	(a) x + x' = 1	(b) x*x' = 0
Teorema 1	(a) x + x = x	(b) x*x = x
Teorema 2	(a) x + 1 = 1	(b) x*0 = 0
Teorema 3, involución	(x')' = x	
Postulado 3, conmutativo	(a) x + y = y + x	(b) x*y = y*x
Teorema 4, asociativo	(a) x + (y + z) = (x + y) + z	(b) x (y z) = (x y) z
Postulado 4, distributivo	(a) x (y + z) = x y + x z	(b) x + y z = (x + y)(x + z)
Teorema 5, de De Morgan	(a) (x + y)' = x' y'	(b) (x y)' = x' + y'
Teorema 6, absorción	(a) x + x y = x	(b) x (x + y) = x

Aplicando estos teoremas y postulados, es posible llevar una función, planteada en una tabla de verdad, a su mínima expresión, para poder ser "construida" con un numero reducido de componentes, o conectivos, que se manejan en el diseño digital, tales como: or, and, not, y variaciones de estas, nor y nand; un ejemplo de esto, es el descrito y mostrado en la introducción de este trabajo. El método de reducción por mapas de Karnough está basado en el algebra booleana, por lo que también permite reducir una función a una expresión mas sencilla, esto es, con una cantidad de conectivos mínima.

Los diseños, tanto del sumador, como del restador, descritos y mostrados en el capitulo 2, están basados, tambien, en estos teoremas y postulados.