2 Ejem

UNIVERSIDAD AUTONOMA DE GUADALAJARA

ESCUELA DE INGENIERIA EN COMPUTACION

"SIMPLIFICACION DE ECUACIONES BOOLEANAS POR COMPUTADORA"

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

BENITO ALVAREZ OJINAGA

GUADALAJARA, JAL. SEPTIEMBRE 1987

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

870116

Ing. Carlos Cortés B.
ASESOR

ING. CARLOS CORTES BUENROSTRO
DIRECTOR
ESCUELA DE INGENIERIA EN COMPUTACION

UNIVERSIDAD AUTONOMA DE GUADALAJARA
Fundada en 1935

Guadalajara, Jal., 12 de Enero de 1987.

ESCUELA DE INGENIERIA EN COMPUTACION
Al Pasante de Ingeniería en Computación
Sr. Benito Alvarez Ojinaga
P r e s e n t e.

En contestación a su solicitud de fecha 1º de Noviembre de 1986, me es grato informarle que la Comisión de Tesis que me honro en presidir, aprobó - como tema que usted deberá desarrollar para su examen de Ingeniero en Computación, el que a continuación transcribo:

" SIMPLIFICACION DE ECUACIONES BOOLEANAS POR COMPUTADORA ".

Introducción
Antecedentes.

I.- Simplificación de Ecuaciones Booleanas.

II.- Simplificación por Métodos Gráficos

III.- Simplificación de Ecuaciones Booleanas por Computadora.

IV.- Aplicaciones.

V.- Conclusiones y Recomendaciones.

Referencias Bibliográficas.

Ruego a Usted tomar nota que la copia fotografiada del presente oficio, deberá ser incluída en los preliminares de todo ejemplar de su tesis.

A T E N T A M E N T E:
" CIENCIA Y LIBERTAD "

Ing. Carlos Cortés Buenrostro.
Director
Escuela de Ingeniería en Computación.

" SIMPLIFICACION DE ECUACIONES BOOLEANAS

POR

COMPUTADORA."

CONTENIDO:

DEDICADA A:

Mis padres: Jaime y Mague, por su fuerza y cariño a la vida, motivo de ejemplo a seguir.

Mis hermanos: Magaly, Gustavo, Eréndira, Irasema, Xochitl y Malibe, por el gran apoyo y cariño brindado en todo momento, permitiéndome un lugar especial en el plano familiar, en igualdad de condiciones y derechos.

Mis compañeros: A todos aquellos que me brindaron una amistad sin límites, y en forma especial a: Minerva, Ana Delia, Carmen, Patty, Jaqueline, Miguel, Chuy y Félix quienes me apoyaron en mi vida de estudiante.

Mis Familiares: Por brindarme el calor de hogar en esta vida tan material y fría.

Y a Dios por estar conmigo en todo momento.

INTRODUCCION

El gran desarrollo electrónico que ocurre día a día, origina búsqueda de nuevos métodos matemáticos que solventen los problemas a los cuales se enfrenta el diseñador electrónico.

El Algebra Booleana proporciona las herramientas necesarias para resolver cualquier problema en cuanto al diseño, sin embargo como toda matemática, requiere de bastante dominio y agilidad para evitar caer en algun error. Error que se verá reflejado en el producto final del trabajo.

Al estar manejando el Algebra Booleana, con términos demasiado largos y complicados, las probabilidades de caer en un error aumentan,debido a lo tedioso que se pueda volver la ecuación y no tanto a la dificultad que ésta presente, por lo que es conveniente manejar la ecuación con la ayuda de la computadora.

La computadora es un camino rápido y seguro, aunque no sea el óptimo en cuanto a la minimización de la función por lo que la siguiente tesis consistirá en la aplicación de la computadora a la solución de problemas de Algebra Booleana.

Tomando en cuenta que la presente tesis es solo una herramienta más, la cual presenta limitaciones, rapidez y eficiencia de acuerdo a la máquina que se está manejando, en este caso es una Hewlett Packard 150 y el lenguaje de programación a manejar es Turbo Pascal.

" A N T E C E D E N T E S. "

A).- INTRODUCCION AL ALGEBRA BOOLEANA.

B).- DEFINICION DE ALGEBRA BOOLEANA.

A).- INTRODUCCION AL ALGEBRA BOOLEANA.

El Algebra Booleana tiene este nombre, en honor de George Boole, quién fué el que la desarrolló a mediados del año 1800, pero no fué sino hasta 100 años después que Claude E. Shannon la aplicó por primera vez, en circuitos de conmutación.

Esta Algebra permite prescindir de la intuicuión y por medio de ecuaciones algebraicas ayuda a tomar decisiones apropiadas sobre diversos problemas.

Gracias al Algebra de Boole se pueden resolver problemas que tienen cierto grado de redundancia, tales problemas pueden ser tan comunes como el caso de un enfermo que debe tomar una pastilla cada 6 horas o cuando le duela la cabeza o después de cada alimento. Otro problema común sería el caso de una madre que tuviera que recoger a su hijo a la 1 p.m. ó cuando le llame por teléfono, pero no debe olvidar comprar las tortillas cuando vaya a recogerlo.

Estos problemas no presentan en sí ningún grado de dificultad, pero pongamos un ejemplo más complicado:

" La expedición de cierta credencial para una tienda de descuento solo se otorgará a personas que cumplan los siguientes requisitos:

A).-Sean casados y ganen menos de $100.

B).- Sean casados, tengan hijos y ganen menos de $150.

C).- No sean casados y tengan 10 años trabajando como obreros en la empresa.

D).- No se otorgará credencial a personas que no hayan demostrado fidelidad hacia la empresa".

Como se puede ver la toma de una decisión correcta, requiere algo más que la simple intuición, por lo que se hace necesario utilizar una arma más fuerte que nos ayude a tomar una decisión correcta. Bien, pues esta arma es el Algebra Booleana.

Su aplicación se ha extendido mucho más, adentrándose principalmente en la solución de circuitos de conmutación, los cuales son la base de la implementación de circuitos electrónicos que rodean el mundo moderno en que vivimos.

B).- DEFINICION DE ALGEBRA BOOLEANA.

El Algebra de Boole, al igual que el álgebra de conjuntos ó lineal, posee ciertas reglas de aplicación que la hacen característica, así como en el álgebra de conjuntos se sabe que la intersección de 2 conjuntos la componen los elementos que tiene en común ambos conjuntos, en el Algebra de Boole se toma como válida la ecuación:

$$a + a = a$$

Habiéndose dado una idea de lo que es el álgebra de Boole se procederá a dar una definición, más primeramente para una mejor comprensión de la definición, se expondrán reglas básicas para el manejo de el Algebra Booleana:

" Se toma como base de combinación la operación binaria, la cuál se denotará por "^".

Así pués, una operación binaria ~ en un conjunto A es una regla de asignación para cada par ordenado (a,b) elementos de A, dando un elemento único c = a ~ b de A ".

Se entiende que una operación binaria, puede ser una adición sustracción ó cualquier manipulación de elementos pertenecientes a cierto universo A.

Las operaciones binarias poseen las siguientes características:

A).- Una operación ~ binaria en un conjunto de elementos A, es asociativa si y solo sí para todo a,b y c en A:

$$a \sim (b \sim c) = (a \sim b) \sim c$$

B).- Una operación binaria ~ en un conjunto de elementos A, es conmutativa si y solo sí para a y b en A:

$$a \sim b = b \sim a$$

C).- Si ~ y @ son dos operaciones binarias en el mismo cunjunto A, ~ es distributiva con respecto a @ si y solo sí para todo elemento a,b y c en A:

$$a \sim (b @ c) = (a \sim b) @ (a \sim c)$$

D).- Un elemento "e" de una clase A, es una identidad para la operación binaria ~ si y solo sí:

$$a \sim e = e \sim a = a$$

Para todo elemento a de A.

Estas definiciones resultan conocidas, ya que actuán de igual manera en el álgebra lineal o de conjuntos.

Ahora, se procederá a dar una definición de lo que es el álgebra Booleana, más se debe considerar que existen muchas maneras de definirla llegándose al mismo concepto. La siguiente definición se basa en la que dió Huntington en 1904, la cuál dice:

" Una clase de elementos B junto con 2 operaciones binarias: ~ y @, es una álgebra Booleana sí y solo sí se verifican los siguientes postulados:

P1).- Las operaciones ~ y @ son conmutativas.

P2).- Existen en B distintos elementos identidad 0 y 1 relativos a las operaciones ~ y @ respectivamente.

P3).- Cada operación es distributiva con respecto a la otra.

P4).- Para cada a de B, existe un elemento á de B, tal que:

$$a + a' = 1 \quad y \quad a\, a' = 0$$

Cabe hacer la aclaración que estos postulados no pueden derivarse uno de otro.

Aunque esta definición no resulta muy clara para aplicaciones de la vida real, es muy precisa desde el punto de vista matemático, ya que ahora se pensará en resolver ecuaciones algebraicas las cuales darán la solución deseada y no la elección adecuada por intuición para la expedición de una credencial o para tomarse una pastilla.

El Algebra Booleana tiene aplicaciones muy amplias, más su propósito general está enfocado principalmente a la solución de circuitos lógicos.

Su manejo tiene los siguientes fines, tomando en cuenta que estos no son los únicos:

A).- Expresar en forma algebraica una relación de tablas de verdad entre las variables.

B).- Expresar en forma algebraica la relación de Entrada-Salida de diagramas lógicos.

C).- Encontrar circuitos más simples para una misma función.

Es conveniente saber que el resultado de una ecuación se puede expresar de diferentes maneras; se verá más adelante la manera de encontrar una solución más corta para una función dada, simplificándola por teoremas y tablas de verdad.

" C A P I T U L O I "

I).- SIMPLIFICACION DE ECUACIONES BOOLEANAS.

I.1).- OPERACIONES BASICAS.

I.2).- TEOREMAS DE SIMPLIFICACION.

I.3).- FORMAS ESPECIALES DE FUNCIONES.

CAPITULO I.

I).- SIMPLIFICACION DE ECUACIONES BOOLEANAS.

I.1).- OPERACIONES BASICAS.

Una vez que se dió la definición de álgebra Booleana y sus objetivos, se darán teoremas para el manejo y simplificación de las ecuaciones booleanas siguiendo un orden, de manera que no se pierda la validez de los conceptos.

A la manera de representar las ecuaciones booleanas gráficamente se le denomina "tablas de verdad", las cuales son de mucha ayuda, ya que se puede observar el comportamiento de una variable determinada en una función.

En una tabla de verdad se representa a una variable prima o testada por medio de un 0 y por un 1 en caso contrario.

Las operaciones básicas que se manejan en esta álgebra son las operaciones lógicas AND,OR Y NOT a partir de las cuales se derivan todos los postulados y teoremas básicos para la manipulación de esta álgebra.

La operación AND se simboliza por ".", el cual por lo general se omite, y su relación entre dos variables a y b es la siguiente:

a	AND	b	=	X
0		0		0
0		1		0
1		0		0
1		1		1

Las ecuaciones correspondientes a la tabla de verdad AND serían:

$$a' \cdot b' = x'$$

$$a' \cdot b = x'$$

$$a \cdot b' = x'$$

$$a \cdot b = x$$

La operación OR se simboliza por + , el cual no debe de omitirse, su relación presentada en una tabla de verdad es siguiente:

a	OR	b	=	X
0		0		0
0		1		1
1		0		1
1		1		1

Las operaciones correspondientes serían:

$$a' + b' = x'$$

$$a' + b = x$$

$$a + b' = x$$

$$a + b = x$$

Por último se tiene la operación NOT la cual viene siendo el complemento de la variable dada, se representa de diversas maneras, como: -,' o NOT , cualquier forma es válida y no debe de omitirse alguna indicación que nos den en significado de prima.

Su relación con respecto a una variable es:

a	NOT a (a')
0	1
1	0

Como se puede ver estas tres operaciones son sencillas más su buena comprensión es fundamental para el mejor aprovechamiento y manipulación de los teoremas próximos a definir.

Cabe señalar que dos funciones son equivalentes en cuanto se produzca el mismo resultado, ya sean ambas 1 ó 0, sin importar si el acomodo a manipulación de las variables o términos es diferente.

I.2).- TEOREMAS DE SIMPLIFICACION.

Los siguientes teoremas se derivan a partir de los postulados de la definición de álgebra Booleana y las tres operaciones básicas , se omitirá su demostración debido a que unas resultan obvias o son derivados de los otros teoremas.

Estos teoremas pueden diferir en número o nombre entre un libro u otro más su significado es el mismo. Los siguientes

fueron tomados del libro de texto de J. Eldon Whitesitt, llamado " Algebra Booleana y sus aplicaciones."

Bien, los teoremas son los siguientes:

TEOREMA # 1).- Toda proposición o identidad algebraica deducible de los postulados de una álgebra Booleana sigue válida si todas las operaciones (+) y (.) y los elementos de identidad (0 y 1) son intercambiados entre sí.

Este teorema se conoce como principio de Dualidad.

TEOREMA # 2).- Para todo elemento a en una Algebra Booleana:

$$a + a = a \quad y \quad a\,a = a$$

TEOREMA # 3).- Para cada elemento a en una Algebra Booleana:

$$a + 1 = 1 \quad y \quad a\,0 = 0$$

TEOREMA # 4).- Para cada par de elementos a y b de una Algebra Booleana B:

$$a + a\,b = a \quad y \quad a\,(a + b) = a$$

TEOREMA # 5).- En toda Algebra Booleana B, cada una de las operaciones binarias (+) y (.) es asociativa, esto es, para toda a, b y c en B:

$$a + (b + c) = (a + b) + c$$

$$a\,(\,b\,c\,) = (\,a\,b\,)\,c$$

TEOREMA # 6).- El elemento a' asociado con el elemento a en el Algebra Booleana es único. Esto es solo un elemento á satisface las condiciones del Post. #4.

TEOREMA # 7).- Para toda Algebra Booleana B:

$$-(a')' = a$$

TEOREMA # 8).- En toda Algebra Booleana B:

$$0' = 1 \quad y \quad 1' = 0$$

TEOREMA # 9).- Para toda a y b en una Algebra Booleana B:

$$(a\ b)' = a' + b'$$

y

$$(a + b)' = a'\ b'$$

Antes de nombrar el siguiente teorema es necesario realizar una definición, de la cuál se partirá.

DEFINICION.- La relación de "orden" $a<=b$ se define por la proposición:

" Para toda a y b en una Algebra Booleana B, $a<=b$ si y solo sí $ab'=0$".

Donde el simbolo <= significa subconjunto.

TEOREMA # 10).- Las siguientes cuatro proposiciones de <= son Válidas en toda Algebra Booleana para elementos arbitrarios X,Y,Z.:

A).- Si X <= Y y Y <= Z , entonces X <= Z.

B).- Si X <= Y y X <= Z , entonces X <= YZ.

C).- X <= Y si y solo si Y' <= X'.

Estos teoremas son los principales más no los únicos, ya que van encaminados a la simplificación de 2 ó más variables.

Sin embargo, también se tienen otros teoremas que ayudan a manejar una ecuación en base a términos, entendiéndose como términos la relación mas simple entre un grupo de variables de una ecuación.

I.3).- FORMAS ESPECIALES DE FUNCIONES.

Existen varias formas de manejar una ecuación, como se había indicado anteriormente; estas formas son las siguientes:

A).- SUMA DE PRODUCTOS.

También se conoce como FORMA NORMAL DISYUNTIVA ó MINITERMINOS, J. Eldon Whitesitt la define de la siguiente manera:

" Se dice que un función Booleana esta en forma normal disyuntiva en N variables X1,X2,X2,..XN para

n > 0, si la función es una suma de términos del tipo:

f1(X1)f2(X2)...fn(Xn)

donde fi(Xi) es Xi o Xi' para cada i=1,2,3...n y ningún par de términos es idéntico. "

Está definición de suma de productos la cuál Whitesitt llama forma normal disyuntiva, trata de expresar que este tipo de funciones la componen suma de términos, donde los elementos de cada término están unidos por la operación básica AND. como ejemplo se puede tomar la siguiente función:

f(a,b,c)= a' c d' + a b' + a b c

Donde ningún término se repite ni sea posible llegar a una simplificación menor. Aunque la función final no contenga todas las variables las involucra, ya que desaparecieron en la simplificación pero por medio del teorema #3 y el postulado 4 es posible recuperarlas. La siguiente definición habla de ello.

DEFINICION:

Aquella forma normal disyuntiva en N variables que contenga 2 exp N términos se llama FORMA NORMAL DISYUNTIVA COMPLETA en N variables.

Con lo cuál se asienta que aquella función que tenga presente todas las variables en todos los términos, los cuales deben ser 2 exp N, al final será una forma normal disyuntiva completa o suma de productos completo.

Es posible llegar de una forma disyuntiva normal una forma disyuntiva completa, agregando variables que hagan falta, sin romper las reglas del Algebra Boolena.

B).- PRODUCTO DE SUMAS:

También se conoce como FORMA NORMAL CONJUNTIVA ó MAXITERMINOS, y es de igual utililidad que la forma DISYUNTIVA, se define de la siguiente manera:

" Se dice que una función Booleana está en forma conjuntiva en N variables X1, X2, X3 ... XN para N > 0, si la función es producto de términos del tipo:

$$f1(X1) + f2(X2) + \ldots + fn(Xn)$$

Donde fi(Xi) es Xi ó Xi' para cada i=1,2,3..N y ningún par de factores es idéntico. "

De igual manera que para la forma normal disyuntiva completa, se definirá para la forma conjuntiva:

DEFINICION:

La forma normal conjuntiva en N variables que contiene 2 exp N factores se llama FORMA NORMAL CONJUNTIVA COMPLETA.

C).- EXPRESIONES MINIMAS BOOLEANAS

Las expresiones booleanas mínimas son otra forma de manejo de expresiones booleanas enfocadas al manejo de términos, la cuál se define de diversas maneras de acuerdo hacia donde esté orientada, ya sea terminos o variables. Así:

A).- Una función mínima de producto de sumas o sumas de productos es aquella que contiene el menor número de literales.

B).- Una función mínima de producto de sumas o sumas de productos, es aquella que contiene el número mínimo de términos.

Cabe agregar que no siempre aquella que contenga el número mínimo de términos será la que contenga el número mínimo de variables, sin importar si esta en forma conjuntiva o disyuntiva, así que la elección adecuada dependerá de la aplicación que se desee dar.

Sin embargo si se desea una expresión mínima de variables, puede factorizarse,es decir sacar factores comunes que nos eliminen variables para llegar a un factor común reducido que contenga un múmero minimo absoluto de variables.

D).- FORMA CANONICA.

Otra forma especial de representar un término en el álgebra Booleana que es de gran ayuda, es la forma Canónica, la cuál contiene todos los términos elementales compuestos de todas las variables.

Así se puede decir que una función puede estar escrita de diferentes maneras, pero su forma canónica siempre será la misma.

Así,con ayuda de una tabla de verdad expresaremos la forma canónica de una función F(x,y,z) donde se tendrán 3 variables y 8 maxiterminos y 8 miniterminos:

x y z	MINITERMINOS (disyuntiva)	MAXITERMINOS (conjuntiva)
0 0 0	M0 = x`y`z`	m0 = (x`+y`+z`)
0 0 1	M1 = x`y`z	m1 = (x`+y`+z)
0 1 0	M2 = x`yz`	m2 = (x`+y+z`)
0 1 1	M3 = x`yz	m3 = (x`+y+z)
1 0 0	M4 = xy`z`	m4 = (x+y`+z`)
1 0 1	M5 = xy`z	m5 = (x+y`+z)
1 1 0	M6 = xyz`	m6 = (x+y+z`)
1 1 1	M7 = xyz	m7 = (x+y+z)

Ahora se agregará una columna en la cuál se indicará indicará mediante un 1 el término que existe en la función expresada en minitérminos, por lo que la tabla quedará:

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

La función f(x,y,z) correspondiente sería:

$f(x,y,x) = x'y'z + x'y\,z' + x\,y'z$

Si deseamos expresarla en maxitérminos tomaremos los términos que hagan 0 la ecuación, por lo que la función quedará:

$f(x,y,z,) = (x'+y'+z')(x'+y+z)(x+y'+z')(x+y+z')(x+y+z)$

Como se sabe, se puede hacer una función de maxitérminos a minitérminos y viceversa, mediante teoremas o forma tabular que nos presenta la forma canónica.

" C A P I T U L O II "

II).- SIMPLIFICACION POR METODOS GRAFICOS.

II.1).- METODO DE MAPAS.

II.2).- METODO TABULAR.

CAPITULO II

" SIMPLIFICACION BOOLEANA POR MEDIO DE METODOS GRAFICOS. "

INTRODUCCION.

Como se vió en el capitulo anterior, resolver ecuaciones en forma canónica resultó una manera sencilla de elegir el método a aplicar para lograr una mejor simplificación . Pues esa facilidad la otorgan las gráficas que se manejan, ya que por medio de éstas se puede tener una buena visualización de la función y del comportamiento de las variables.

Existen varios métodos gráficos que ayudan a una mejor simplificación, aunque se debe aclarar que el resultado final obtenido de la tabla no es siempre el más óptimo, ya que muchas veces es posible llegar a una ecuación más simple, aplicando teoremas o postulados citados anteriormente.

Bien, se empezarán a explicar métodos orientados a ecuaciones de pocas variables y al final se llegará al método Tabular, el cual puede resolver de igual manera ecuaciones de N variables.

II.1).- SIMPLIFICACION POR MAPAS.

Este método es sencillo de manejar más se requiere de cierta práctica para lograr

dominarlo.

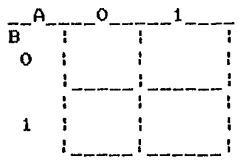
Este sistema se basa principalemnte en el método de Karnaught, quien define un mapa como una figura geométrica que asocia una región o compartimiento a cada fila de una tabla de verdad.

En cada cuadro se representará mediante un 1 el términos que se desee estar presente, un 0 el caso contrario y mediante una X los casos opcionales, los cuales no afectan a la salida. Los 0 pueden omitirse, dejándose el cuadro en blanco.

Se concideran como adyacencias cualquier grupo de 4 1's se encuentren formando un cuadro o una columna o renglón desaparesciendo las 2 variables que cambien de estado.

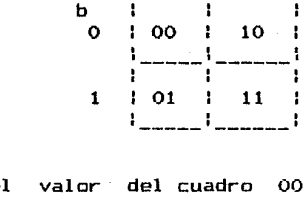
Un mapa puede ser de varias formas, todo depende del acomodo de las variables. Así se presentarán distintas formas de acomodar una función en un mapa:

Si se tienen 2 variables, se puede ordenarlas de la siguiente manera:



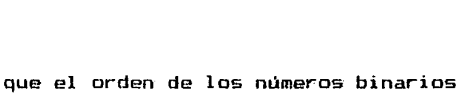
Donde la variable A estará de acuerdo con las columnas

y B con los renglones, el valor de cada cuadro es el siguiente:



Donde en el valor del cuadro 00 corresponde al término a'b',10 al término ab',01 al término a'b y 11 a ab.

También se puede representar en forma lineal:



Observese que el orden de los números binarios no llevan la secuencia lógica normal , ya que los valores 01 y 11 estan intercambiados, esto es para lograr una adyacencia de cuadros, lo cuál ayudará a simplificar.

Por lo tanto, se tomará como regla de simplificación que cuando 2 cuadros adyacentes contengan el mismo valor, ya sea 1 o 0 se agrupan, simplificandose ambos términos, desaparesciendo la variable que cambia de estado.

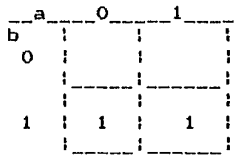
No es muy común manejar mapas para simplificar ecuaciones de 2 variables, más se hará con el fin de tomarse

como ejemplo, debido a su facilidad de manejo.

Así, si se tiene la función:

$$f(a,b) = a' b + a b$$

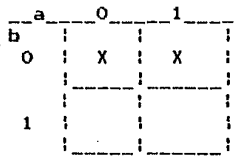
Su representación en el mapa será:



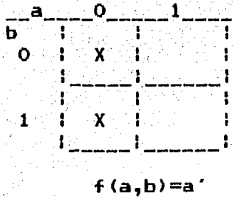
Al basarse en la regla, se da cuenta que la variable que cambia de estado es la "a", por lo que desaparece de la función, quedando simplificada a:

$$f(a,b)=b$$

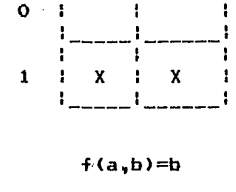
En mapas de 2 variables, se toman como cuadros adyacentes los que contienen X en los siguientes mapas:



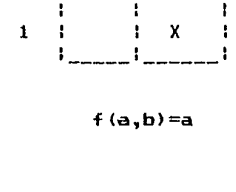
$$f(a,b)=b'$$



f(a,b)=a'



f(a,b)=b



f(a,b)=a

Inmediatamente se observa como se reduce el término de entrada. Sin embargo es conveniente agrupar un cuadro cuantas veces sea posible, y en cuanto se relacione con un número mayor de cuadros es mejor. Las agrupaciones de cuadros siempre serán en potencias de 2, ya sea 2,4,8 o 16 cuadros.

El acomodo de las variables es fundamental, ya que al aumentar el número de estas aumenta la dificultad, y en caso de no haberse definido previamente la entrada se ocacionará

una discrepancia en las agrupaciones.

Como mejor ejemplo de lo escrito anteriormente, se manejará una función de 3 variables y se acomodará de 2 maneras diferentes.

La función es la siguiente:

$$f(a,b,c) = a\ b\ c\ +\ a\ b\ c'\ +\ a'\ b'\ c'$$

Su acomodo será:

c \ ab	00	01	11	10
0	1		1	
1				1

Si se invierte el acomodo de las variables la tabla dá un giro, de la siguiente manera:

ab \ c	0	1
00	1	
01		
11	1	1
10		

La segunda tabla no resulta muy versátil de manejar más es muy claro el error en que se cae al acomodar en forma desatinada las variables. Esto resulta un problema al manejar una tabla cuadrada, por ejemplo donde se tengan 4 variables, ya que al acomodar en una u otra forma las variables no se obtendrá una buena visualización.

Bien, ahora se tratará ecuaciones de 4 variables, donde es más común el manejo de mapas y donde se prestará mayor atención.

Si se tiene una función f de cuatro variables, representación en una tabla de verdad es algo complicada, por lo que se darán varios formas de ayuda.

Si se tiene la función:

$$f(a,b,c,d) = a\ b\ c\ d \ + \ a\ b\ c'\ d \ + \ a'\ b'\ c'\ d'$$

Es conveniente cambiar a 1 y 0 cada variable, por lo que quedará:

$$f(a,b,c,d)=1111 + 1101 + 0000$$

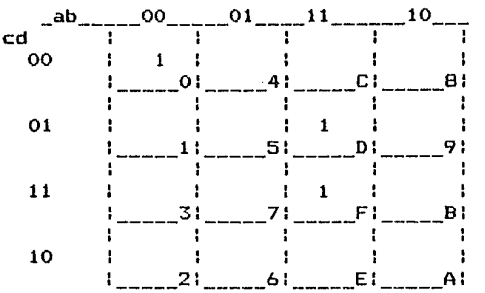
Ahora resulta mas fácil su acomodo en la tabla.

ab / cd	00	01	11	10
00	1			
01			1	
11			1	
10				

Si cada término se toma como valor haxadecimal, entonces

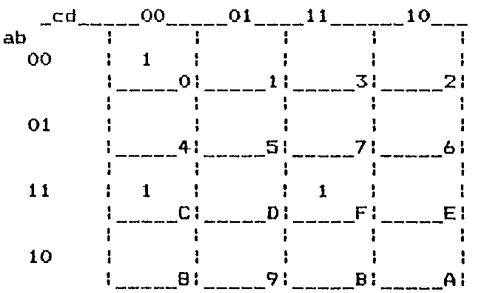
$$f(a,b,c,d) = F + C + 0$$

Y se enumera la tabla



De esta manera se ve como el acomodo de la ecuación es más sencillo.

Si se invierte el acomodo de las variables, la forma de la gráfica no cambia mas si el acomodo de las variables, por lo que resulta importante fijarse en este detalle, así , la ecuación acomodada con las variables invertidas queda:



Debido a que la tabla de 4 variables es mayor, presenta un número más grande de adyacencias, ya sea de 4, 8 o 16 cuadros, por lo que a continuación se presentarán como se agrupan estos cuadros.

Si se llegará a tener una agrupación de los 16 cuadros, que resulta casi imposible, su variable de salida sería :

$$f(a,b,c,d)=0$$

Para agrupaciones de 8 cuadros, se tienen varias adyacencias, no se enumerarán todos los casos , solo los más comunes:

Si se tuviera la siguiente tabla:

ab \ cd	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11				
10				

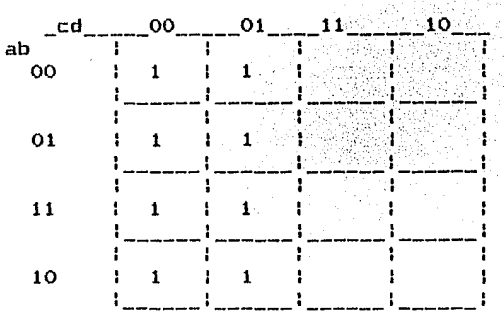
Su función de salida será:

$$f(a,b,c,d)= a'$$

Si los 1's se encontrarán en la parte de abajo, la función de salida será:

$$f(a,b,c,d)= c$$

Para el siguiente caso:

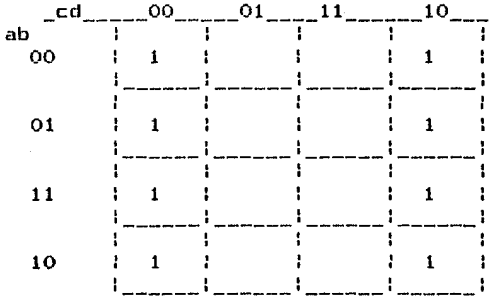


$f(a,b,c,d) = c'$

Si los 1's estuvieran del otro lado:

$f(a,b,c,d) = c$

Para el siguiente caso:



$f(a,b,c,d) = d'$

Si los 1's estuvieran en el medio:

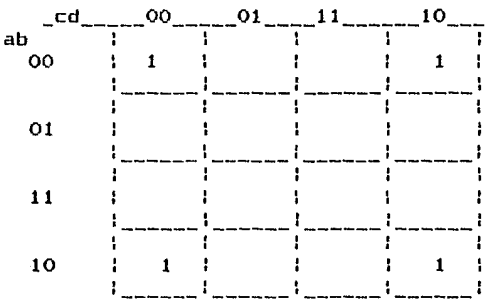
$f(a,b,c,d) = d$

Lo mismo sucede en el caso de que los 1's estuvieran en forma horizontal.

Para agrupaciones de 4 cuadros, se tienen un mayor número de adyacencias que para el caso anterior, de igual manera solo se mencionaran unos cuantos casos.

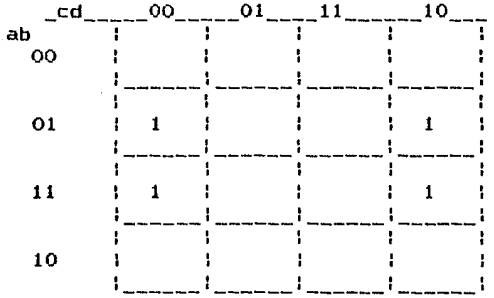
A parte se tienen otros casos muy particulares como los siguientes:

Las 4 esquinas:



f(a,b,c,d)= b'd'

Para el siguiente caso:



f(a,b,c,d)=bd'

De igual manera si los 1's se encontrarán girados 90 grados:

$$f(a,b,c,d)=b'd$$

Bien, pues estas son las agrupaciones más importantes para las simplificaciones por medio de tablas.

Si se deseara gráficar una ecuación de 5 variables este método se complica bastante, debido al hecho de que se pierde la adyacencia, y las agrupaciones resultan mas complicadas, más no es imposible su manejo, sin embargo para estos casos de 5 o más variables se cuenta con el método tabular, el cual puede reducir cualquier número de variables.

Los conceptos aplicados a tablas son basados en soluciones por medio de Minitérminos, más sin embargo también es posible llegar a una solución óptima aplicado el principio de Maxitérminos solo que el lugar de agrupar los 1 se tratará de agrupar los 0.

Existen condiciones las cuales no afectan la salida deseada, por lo que muchas veces es conveniente ignorarlas, más no siempre sucede esto, en muchas situaciones tales condiones de no importa pueden ayudar a llegar a una simplificación mas corta, por lo que esos estados se de denotan mediante una X y pueden ser conciderados como 1 o 0.

-II.2).- METODO TABULAR:

Este método es demasiado flexible, actuando de igual manera para ecuaciones de 20 variables o de 4, sin embargo sus teoremas de simplificación son menores ya que principalmente se basa en teoremas ya conocidos. Su resultado final puede que no sea, como en todo método de gráficas el más óptimo, y menos en el método tabular, pero presenta una gran ayuda para simplificar ecuaciones demasiado grandes que contengan 10, 15 o más variable, puede que esto no resulte muy común en caso de la vida diaria , sin embargo en aplicaciones de industria o procesos financieros pueden presentarse casos como estos.

Los teoremas principales en los que se basa este método son:

A).- $x\,y + x\,y' = x$

Este teorema es el postulado #4 definido en el capitulo 1 donde se habló de lo que es el álgebra Booleana, el cual nos indica quie si dos términos difieren en un 0 o 1 se pueden simplificar en un solo término,se debe conciderar a x como una cadena de variables o una sola variable y a y como una misma variable.

B).- $x + x\,y = x$

Si un término tiene una variables más que otro es

posible reducirlo, quedando el término que permanece constante.

Este teorema es el mismo que el teorema #4 definido anteriormente.

C).- $x + x = x$

Los téminos que se repiten se simplifican a un solo término, este teorema se basa en el teorema #2 donde:

$$a + a = a$$

Estos son solo 3 teoremas en los que se basa el método tabular, se debe conciderar que existen otros teoremas los cuales son posible aplicar a este método, más sin embargo no se hizo, debido a que esta tesis esta orientada a la solución por computadora del método tabular y los teoremas aqui citados serán los principales que se apliquen en el programa de soluciones.

La manera de acomodar una ecuación en una tabla del método tabular es el siguiente:

Si se tiene una ecuación Booleana de la forma:

f(a1,a2,a3,...an) = a1a2a3 ... an + a1a2a3 ...an + ...

Su gráfica será:

a1	a2	a3	. . .	an
x	x	x		x
x	x	x		x
.	.	.		.
.	.	.		.

En la cual se aplicarán los teoremas antes mencionados, llegando a una ecuación mas simple.

Donde cada renglón representará un término específico y cada columna indicará una variable determinada de la ecuación. Se debe conciderar que una variable será representada en una sola columna para todos los términos.

Estos son los métodos grácos más comunes y sencillos de manejar. Para utilizar el método de tablas, existen varios sistemas, como los son el de Kaurnaught o Vernier , más sin embargo el proceso fundamental es el mismo.

En el siguiente capitulo se tratará de resolver mediante computadora problemas booleanos, aplicando el método tabular.

" C A P I T U L O III. "

" SIMPLIFICACION DE ECUACIONES BOOLEANAS POR COMPUTADORA. "

III.1).- INTRODUCCION.

III.2).- ALGORITMO DE PROGRAMACION.

III.3).- DESARROLLO DEL PROGRAMA.

" SIMPLIFICACION DE ECUACIONES BOOLEANAS POR COMPUTADORA. "

III.1).-INTRODUCCION:

La aplicación de la computadora en la solución de ecuaciones booleanas es de gran ayuda para la implementación de cualquier tipo de aplicación, como son los circuitos de conmutación y otras más, de proceso delicado.

Debido a la repidéz y eficiencia del cálculo, al aplicar la computadora en los procesos repetitivos, el margen de error puede ser mínimo, tomando en cuenta que las operaciones a realizar son por lo general muy grandes y complicadas. La computadora actúa de igual manera en ecuaciones de 4, 10 o 30 variables.

La solución de ecuaciones booleanas por computadora se basará en el método tabular descrito anteriormente, como se pudo obserbar el proceso es algo sencillo ya que se aplican pocos teoremas los cuales involucran aquellos que están orientados principalmente hacia la aplicación de Minitérminos, principio por el cual, el programa a desarrollarse empleará solo simplificaciones hechas en base a Minitérminos, por lo que la ecuación tendrá que ser dada en forma de Minitérminos.

Si se revisa la lista de teoremas y postulados dada en el capitulo II, se podrá comprobar que son los siguientes:

A).- $x + 1 = 1$

B).- $x \cdot 1 = x$

C).- $x \cdot 0 = 0$

D).- x + 0 = x

E).- x x = x

F).- x + x = x

G).- x + x'= 1

H).- x + xy = x

I).- x + x'y = x + y

J).- xy + xy' = x

K).- xy + xy'z = xy + yz

Estos teoremas ya fueron explicados anteriormente, ahora se tratará de aplicarlo a los procesos que sean posibles de realizar por computadora.

Primeramente se desarrollará un algoritmo base a partir del cual se iniciará el proceso de programación.

III.2).- ALGORITMO DE PROGRAMACION:

Los pasos a desarrollar a partir del algoritmo se harán por bloques, tratando de que cada paso este bien definido y no exista mezcla de los bloques.

El algoritmo es el siguiente:

PASO #1).- Desplegado de condiciones de entrada de datos y formatos de manejo.

PASO #2).- Lectura de la ecuación en forma literal, y aplicación de los teoremas y postulados siguientes.

A).- x + 1 = 1

B).- x 1 = x

C).- x 0 = 0

D).- x + 0 = x

E).- x x = x

PASO #3).- Cambio de la ecuación de forma literal a numérica.

PASO #4).- Llenado de tablas a manejar e iniciación de contadores.

PASO #5).- Aplicación de los teoremas de simplificación.

5.1).- x + x = x

x + x'= 1

5.2).- x + xy = x

x + x'y= x + y

xz + xz'y = xz + xy

5.3).- xy + xy' = x

PASO #6).- Cambio de la ecuación de forma numérica a literal.

PASO #7).- Salida de la ecuación ya simplificada.

PASO #8).- Manejo de los teoremas aplicados en los pasos 5.1, 5.2 y 5.3 en orden y sentido diferente.

PASO #9).- Elección de el resultado óptimo.

A continuación se desarrollará cada paso del algoritmo, indicandose el procedimiento correspondiente, tomándose en cuenta que la elección de cualquier tipo de variables o estructura es completamente personal.

IV.3).- DESARROLLO DEL PROGRAMA.

El desarrollo del programa se hará en base al algoritmo antes indicado. El lenguaje a utilizar será Pascal debido a la gran ayuda que ofrece en la programación por bloques, recordando su gran flexibilidad en el manejo de subrutinas.

Las variables y arreglos a utilizar durante el desarrollo del programa son las siguientes:

NumTer.- Cantidad total de términos que puede manejar el programa, su número puede variar de acuerdo a las necesidades del usuario, por lo que si es necesario corregirse bastará con aumentar o disminuir este valor en la declaración del programa principal.

NumVar.- De igual manera que Numter, indica la cantidad máxima de variables que pueden que puede manejar el programa, también será posible modificarse de acuerdo a las necesidades del usuario. Tomando en cuenta que el número máximo a manejarse se refiere al número máximo de variables que puede tener un término en

particular dentro de la ecuación.

IGUAL,VADIF,TRIDIF.- Indican si existen diferencias al realizarse las comparaciones en la aplicación de el primer, segundo o tercer teorema de simplificación.

LUDIF.- Indica el lugar donde existe una diferencia, en cualquier simplificación.

PAS.- Indica la secuencia de simplificaciones a utilizar así como el sentido o dirección de comparación de los términos.

Noter.- Indical el número de términos introducidos, esto es con el fin de llevar un conteo de términos que tiene la ecuación a manejarse y no realizar comparaciones con el limite máximo.

Novar.- De igual manera que Noter, indica el número de variables que tiene un término introducido.

ELIM.- Elimina el término que desaparezca en cualquier simplificacion.

CAMB.- Indica si ya no existe ningún cambio, por lo que será igual a cero cuando ya no exista alguna otra simplificación.

TablaIn.- Es un arreglo que contiene la ecuación de entrada en forma literal.

Contemp.

ContVar.- Es un arreglo que guarda la cantidad de variables de cada Término, indicado por Novar.

TablaOut.- Es para manejar la ecuación de salida en forma literal.

ValTemp.

TablaVal.- Lleva un relación de los términos existentes y los que desaparezcan por alguna simplificación. Siendo ValTemp quien conserva los valores de la ecuación original y TablaVal quien lleva el conteo de los que van desapareciendo durante los procesos de simplificación.

TablaCamb.- Relaciona los términos que se vayan mezclando durante las simplificaciones.

TablaCanon.- Es para manejar la ecuación de entrada en forma numérica, con el fin de facilitar las simplificaciones ya que estas serán de acuerdo a núeros y posiciones no a caracteres.

TablaSal.- Una vez que la ecuación ya esta simplificada se encuentra en TablaSal, ya que al realizarse cada simplificación se fué pasando de TablaCanon a TablaSal.

TablaResul.- Almacena todos los resultados obtenidos en cada proceso de simplificación.

SelResul.- Selecciona el resultado más óptimo contenido en TablaResul. Para la elección del resultado se concidera como el más óptimo a aquel que contenga un número menor de variables ya que cada variable implica un salida más en la implementación de un circuito electrónico o una considearción más en un caso de la vida real.

CambFin.- Almacena todos los numeros de los términos que se mezclaron durante todos los procesos de simplificación.

Una vez que ya se difinieron todas las variables a manejar, se proseguirá a desarrollar el programa. El cuál se hará en una Hewlett Packard 150, usándose Turbo Pascal.

Los bloques a desarrollarse serán de acuerdo a los pasos indicados en el algoritmo indicado. Así se tiene como primer paso:

PASO #1).- DESPLEGADO DEL MANEJO DEL PROGRAMA.

Las condiciones para manejar el programa son con el fin de orientar a la persona que lo utilize,sobre los pasos y formatos a emplear, en base a las condiciones que se toman como reglas inicales.

Las reglas bases son las siguientes:

a).- La ecuacion de entrada deberá estar en forma de Minitérminos.

b).- Las variables de entrada serán libres de elegir, siempre y cuando sean minúsculas.

c).- El simbolo para las variables testadas (') se encuentra en la tecla juego-alterno (;), sino se buscará el lugar donde se encuentre este simbolo, ya que de lo contrario no será tomado en cuenta. Se debe diferencia de el acento en que este es en el sentido contrario,el acento es (´) y para las testadas es (').

Se hizo esto devido a que durante los proceso de programación causa problemas en los writeln o entrada de datos.

d).- Una vez que se introdujo la información completa,es necesario teclear dos veces <return> para salir de un doble ciclo de programación.

e).- Para el caso de que un término contenga una constante, la cual sea 1 o 0, primero se teclearán las varaibles y después las constantes. Y en el caso de que un término contenga un 0 y un 1 se tecleará primero el 0 y después el 1.

El proceso de programación de este paso no requiere indicarse debido a son solamentes letreros de salida.

PASO #2).- LECTURA DE LA ECUACION EN FORMA LITERAL.

La ecuación de entrada se dará simplemente tecleandose las variables y los términos, separando cada término por el simbolo +.

Este procedimiento realiza varias simplificaciones ya sea acerca de teoremas o postulados, los cuales se manejan desde la entrada de la ecuación. Las simplificaciones son las siguientes:

a).- $1 + x = 1$

b).- $1 \cdot x = x$

c).- $0 + x = x$

d).- $0 \cdot x = 0$

e).- $0 \cdot 1 = 0$

f).- $x \cdot x = x$

El procedimiento a manejarse puede ser desarrollado de siguiente manera:

```
PROCEDURE ENTRADA;

  (*  Almacena los datos inmediatamente de ser tecleados, tomando
      variable por variable, consecutivamente en el arreglo Término,
      después se trasladan a TablaIn en su localidad correspondiente
      de acuerdo al orden alfabético. *)

CONST
    BLANCO=' ';    (* Para mantener inicialmente las localidades en
                      blanco  o  limpias con el fin de  evitar  que
                      contengan  basura  y como  referencia    para
                      comparaciones posteriores. *)

VAR
    Termino: ARRAY[1..NUMTER,1..NUMVAR] OF CHAR;

           (* Contiene la información original al momento de
              ser tecleada.                                    *)

    LOC: INTEGER;

BEGIN
     FOR I:=1 TO NUMTER DO
      FOR J:=1 TO NUMVAR DO
          BEGIN
            Termino[I,J]:=BLANCO;
            TablaIn[I,J]:=BLANCO;
            Contemp[I]:=NumVar;
            Valtemp[I]:=4
          END;
```

```
        (*  Se limpio los arreglos Termino y TablaIn,
            inicializandose los contadores de términos
            y variables.                                    *)

CLRSCR;
WRITELN('<<    PROCEDE  A  METER  LA  ECUACION  EN  FORMA
                    LITERAL                    >>':60);
WRITELN;
I:=1;
J:=1;
Novar:=0;
Noter:=1;
Cero:=0;
READ(KBD,Termino[I,J]); (* Se lee una variable de teclado. *)
WRITE(Termino[I,J]);
WHILE Termino[I,J]<>CHR(13) DO (* Se compara con <RETORNO>.*)
 BEGIN
    WHILE (Termino[I,J]<>CHR(13)) AND (Termino[I,J]<>'+')  DO
         (* Se verifica si se introducirá otro término  *)
    BEGIN
      IF (ORD(Termino[I,J])>=97) AND (ORD(Termino[I,J])<=122)
         THEN
              (* El caracter introducido es una variable.*)
           BEGIN
             LOC:=ORD(Termino[I,J]) - 96;
             IF TablaIn[I,LOC]=BLANCO
                THEN
                     (* Esta variable no a sido introducida.*)
```

```
                BEGIN
                   TablaIn[I,LOC]:=Termino[I,J];
                   ValTemp[I]:=0;
                   Novar:=Novar+1
                     (* Incremento del contador de
                        variables e indicación de
                        existencia de temino.      *)
                END
            ELSE
                TablaIn[I,LOC]:=Termino[I,J]
                 (* Esta variable ya fué introducida.
                    Aplicación del teorema:
                                X  X  =    X.          *)
          END
        ELSE
          IF Termino[I,J]='''
           THEN
            BEGIN
                          (*  La variable es Testada,
                              se  cambia  su valor de
                              Minúscula  a  Mayúscula.*)
              LOC:=ORD(Termino[I,J-1])-96;
              K:=ORD(Termino[I,J-1])-32;
              TablaIn[I,LOC]:=CHR(K);
            END;
            IF (ORD(Termino[I,J])=49) AND (novar=0)
               THEN     (* La variables es un 1 *)
```

```
IF novar=0
 THEN
  BEGIN (* Teorema: 1 + X = 1 *)
       FOR R:=1 TO NUMTER DO
           FOR S:=1 TO NUMVAR DO
                BEGIN

                   Termino[R,S]:=CHR(32);
                   TablaIn[R,S]:=CHR(32);
                   ValTemp[R]:=4;
                   Cero:=1
                END
         ELSE
            IF (novar<>0) AND (CERO=3)
              THEN (* Caso  0 1 = 0 *)
                BEGIN
                   TablaIn[I,R]:=CHR(32);
                   Termino[I,R]:=CHR(32);
                   ValTemp[I]:=4;
                   Cero:=2
                END
  END;
  IF((ORD(Termino[I,J])=48) AND (novar<>0))
  THEN (*  La variable es 0 *)
  IF J<>1
   THEN (* Teorema  X 0 = 0 *)
       FOR R:=1 TO NumVar DO
         BEGIN
            TablaIn[I,R]:=CHR(32);
```

```
                                    Termino[I,R]:=CHR(32);
                                    Tablasal[I,R]:=3;
                                    ValTemp[I]:=4;
                                        CERO:=2
                                END;
                            ELSE
                                BEGIN
                                    CERO:=3;
                                    novar:=novar+1
                                END;

          READ(KBD,Termino[I,J]);
          WRITE(Termino[I,J])
        END; (* CICLO J*)
        Contemp[I]:=Novar; (* Almacenamiento del número de
                              variables que tiene el Término.*)
        ValTemp[I]:=0;
                           (* Indicación de que el término I
                            ya existe, levantando la bandera.*)
        Novar:=0;
        I:=I+1;
        Noter:=Noter+1;          (* Incremento del contador de
                                    términos.                 *)
        J:=1;
        READ(KBD,Termino[I,J]);  (*  Lectura de la variable  *)
        WRITE(Termino[I,J])
  END; (* CICLO I *)
END; (* Procedure Entrada. *)
```

En este procedimiento se introducirá la ecuación de entrada quedando el formato de la siguiente manera:

abc'd + abcd + a'b'cd + ab'c'd'

Y será guardada en el arreglo llamado TablaIn.

No se debe olvidar que en el caso de que un término contenga una constante, esta debe ser tecleada después de las variables.

Las variables Novar y Noter contendrán el número de variables y términos respectivamente.

PASO #3).- CAMBIO DE LA ECUACION DE FORMA LITERAL A NUMERICA.

Con el fin de facilitar el manejo de simplificación se convertirá la ecuación a números, ya que en la utilización de cualquier método gráfico se emplean 1's o 0's, los cambios serán de la siguiente manera:

Testadas ---------- 0
Normales ---------- 1
No presente ---------- 3

Debido a que solo se cambiarán los valores que se encuentran en las tablas ya definidas anteriormente su proceso no es muy complicado.

El procedimiento a desarrollar puede ser de la siguiente forma:

```
PROCEDURE LITCANON;
  (*  El siguiente procedimiento cambia la ecuación original de
forma literal a forma numérica, con el fin de facilitar el manejo
de comparación, así las variables testadas tomarán el valor de 0 y
las no testadas el valor de 1. En el caso de que una variable
no exista se cambiará su valor de BLANCO a 3. *)

  BEGIN
     WRITELN;
     FOR I:=1 TO NumTer DO
         BEGIN
              ContVar[I]:=Contemp[I];
              TablaVal[I]:=ValTemp[I]
         END;
          (*  Se pasan los valores originales de los términos
             existentes y la cantidad de variables que contiene
             cada término, a los arreglos que se manipulan en el
             siguiente proceso de simplificación.          *)
     WRITELN;
     IF PAS=1
        THEN WRITE(' PRESIONA <RETORNO> PARA INICIAR PROCESO.')
        ELSE WRITE(' PRESIONA <RETORNO> PARA CONTINUAR PROCESO.');
        READLN(S);
     CLRSCR;
```

```
WRITELN('**   CAMBIO A FORMA CANONICA DE LA ECUACION ORIGINAL
          **':45);
WRITELN;
IF PAS<>1
  THEN
   WRITELN('**    PARA  REALIZAR LA  SIMPLIFICACION  EN  OTRO
                                ORDEN. **':45);
WRITELN;
WRITE(' ':2);
FOR K:=97 TO 122 DO
         WRITE(CHR(K):2);
               (* Desplegado del valor de cada columna.*)
WRITELN;
FOR I:=1 TO Noter DO
  BEGIN
    WRITE(' ':2);
    FOR J:=1 TO NumVar DO
      BEGIN
        S:=ORD(TablaIn[I,J]);   (* Cambio  del  valor de  una
                                variable a su número  corres-
                                pondiente en ASCII.          *)
        IF (65<S) AND (S<=90)
          THEN
                           (*    La variable  es testada,    *)
                           (*  ya que pertenece al rango de *)
          S:=65            (*  las Mayúsculas.              *)

      ELSE
```

```
            IF(97<=S) AND (S<=122) (*La variable no es testada*)
               THEN          (* ya que esta dentro del rango de *)
                 S:=97;      (* las Minúsculas.                  *)
            CASE S OF
               65: TablaCanon[I,J]:=0;      (*  Cambio al número *)
               97: TablaCanon[I,J]:=1;      (*  correspondiente  *)
               88: TablaCanon[I,J]:=2;      (*  antes definido.  *)
               32: TablaCanon[I,J]:=3
            END;
                WRITE(TablaCanon[I,J]:2) (* impresión a pantalla.*)
         END;
         WRITELN
      END
   END;    (* DEL PROCEDIMIENTO *)
```

La descripción del PROCEDURE es la siguiente:

S .- Contiene el valor en ASCII de la variable dada,

Los rangos de conversión son los siguientes:

65 <= S < 90 implica un cambio a 0

97 <= S < 122 implica un cambio a 0

S = 88 implica un cambio a 2

S = 32 implica un cambio a 3

Estos cambios serán almacenados en TablaCanon, la cual manejará las ecuaciones en forma canónica aunque no este precisamente en las condiciones necesarias para ser canónica, ya

que requiere que la ecuación este expandida completamente.

PASO #4).- LLENADO DE TABLAS A MANEJAR.

Es conveniente llevar un control de todas las relaciones que existan al realizarse las simplificaciones con el fin de orientarse en la ejecución del programa, para ello se tenrdán dos variables dimensionales que actuarán como contadores.

Los contadores estarán en dos arreglos, uno llevará las relaciones de las mezclas de los tétminos en la simplificación y otro llevará el valor de los términos que desaparecen en una simplificación y los que continuan válidos.

El procedimiento correspondiente se ominte, dabido a su sencilles ya que solo conciste en poner en blanco dos arreglos.

PASO #5).- APLICACION DE LOS TEOREMAS DE SIMPLIFICACION.

El proceso mas importante de este programa es este paso, ya que aqui se realizará la simplificación de la ecuación, por lo que se le presta una mayor atención.

Se dividirá en 3 bloques este paso, ya que los primeros teoremas se pueden realizar simultáneamente en un solo bloque de comparación y los otros teoremas se realizarán por separado debido a que su proceso de simplificación es diferente:

así, para los teoremas restantes:

a).- x + x = x

b).- x + x' = 1

c).- x + xy' = x

d).- x + x'y = x + y

e).- xy + xy' = x

f).- xy + xy'z = xy + xz

Se aplicará todo este bloque de procedimientos, siendo por supuesto el más importante de este programa.

El primer bloque que se manejará será el referente a los teoremas:

a).- x + x = x

b).- x + x'= 1

Donde para el teorama (a), x puede se una variable o una cadena de variables, pero en el teorema (b), x tiene que ser exclusivamente una variable.

Se procederá a dar una idea del manejo de la programación mediante el siguiente procedimiento.

```
PROCEDURE SIMPLIFICACION;

(* El procedimiento SIMPLIFICACION aplica el teorema:

                A + A = A
   Donde indica que no pueden existir dos términos igua-
   les, siempre eliminandose uno de los dos.
```

```
        En este procedimiento se elimina el término localizado
      mas a la derecha para el caso en  que  la comparacion de
      de terminos sea de izquierda a derecha, y la eliminacion
      del termino mas a la izquierda se da en   caso de que la
      que el sentido de la comparacion sea de derecha a izquier-
      da.                                                    *)

BEGIN
  REPEAT                    (*  Hasta que no exista simplificacion
                            posible por  medio  de  este teorema.  *)
  CONT:=0;
  GOTOXY(5,3);
  WRITE('1a. SIMPLIFICACION ');
  FOR M:=1 TO NumTer-1 DO
                    (* Contador del ciclo de termino inicial.  *)
  BEGIN
   IF PAS>6
     THEN I:=NumTer+1-M    (* Sentido de comparacion izq-der. *)
     ELSE I:=M;            (* Sentido de comparacion der-izq. *)

   IF TablaVal[I]<>4 THEN  (* Solo se  comparan  terminos  que
                           existan, los cuales   son  iguales
                           a 1.                              *)
  BEGIN
    GOTOXY(1,5);
    WRITE('I:',I);
    FOR N:=M TO NumTer DO  (* Contador del termino con el cual
                            se compara el termino inicial.    *)
```

```
BEGIN
IF PAS>6
   THEN J:=NumTer-N       (*      Sentido izq-der.          *)
   ELSE
       IF N<NumTer
          THEN J:=N+1      (*    Sentido der-izq.          *)
          ELSE N:=NumTer;

IF TablaVal[J]<>4 THEN  (* Verificacion de que el
                              termino  J  existe.          *)
BEGIN
  GOTOXY(8,5);
  WRITE('J:',J);
  IGUAL:=0;          (* Bandera de igualdad de terminos. *)

  FOR K:=1 TO NumVar DO
                (* contador del ciclo de variables.  *)
   IF ((TablaCanon[I,K]<>3) AND (TablaCanon[J,K]<>3))
    THEN     (* Solo se comparan variables existentes. *)
      IF TablaCanon[I,K]=TablaCanon[J,K]
       THEN       (* Las variables K de los terminos J e I
                          son iguales.                    *)
         IGUAL:=IGUAL+1;  (* Incremento del contador
                              de igualdades.              *)
         IF ((IGUAL=ContVar[I])  AND (IGUAL=ContVar[J] ))
```

```
                THEN      (* Todas las variables de ambos terminos
                             deben de ser iguales y deben contener
                             igual numero de variables.          *)
                  BEGIN
                    TablaVal[J]:=4;     (* Eliminacion del termino
                                           con el cual se realiza  la
                                           comparacion.            *)
                    FOR K:=1 TO NumVar DO
                      BEGIN
                       TablaSal[J,K]:=3;
                       TablaCanon[J,K]:=3;
                       GOTOXY(18,5);
                       WRITE('K:',K);
                       CAMB:=1;       (* Bandera de Cambio *)
                       CONT:=1;       (* Bandera de simplificacion.*)
                       GOTOXY(40,5);
                       WRITE('CAMBIO =',CAMB)
                      END;
                    L:=L+1;
                    TablaCamb[PAS,L,1]:=I;
                    TablaCamb[PAS,L,2]:=J
                  END
          END
        END
    END;
  END;
UNTIL CONT=0    (* Hasta que no exista simplificacion posible por
```

```
                    el teorema:
                                A + A =  A *)
END;  (* FIN del procedimiento SIMPLIFICACION *)
```

Este teorema se repetirá hasta que no exista alguna simplificación posible, es decir que en el caso de que se realizé una vez, se dará otra vuelta con el fin de revizar de nuevo los términos generados.

En el siguiente bloque se analizarán los teoremas:

c).- $x + xy = x$

d).- $x + x'y = x + y$

e).- $xz + xz'y = xz + xy$

Donde x pudes ser una variable o una cadena de estas, al igual que y, en cambio z debe ser una variable.

El procedimiento referido en pascal, puede ser el siguiente:

```
PROCEDURE SEGUNSIMP;
  (* El procedimiento SEGUNSIMP se basa en los teoremas:
                        x + xy = x
                        x + x'y = x + y
                        xz + xz'y = xz + xy
   Donde x, y  puede ser una variable o cadena de  variables. *)
VAR
    ELIMIN1,
    ELIMIN2:  INTEGER;  (* Banderas de eleccion de terminos. *)
BEGIN
```

```
REPEAT
  CONT:=0;
  GOTOXY(25,3);
  WRITE('2a. SIMPLIFICACION');
  FOR M:=1 TO NumTer-1 DO     (* Contador del  ciclo  de  termino
                              inicial o primero para compararse.*)
  BEGIN
   IF PAS>6
     THEN I:=NumTer+1-M          (*    Sentido  izq-der.     *)
     ELSE I:=M;                  (*    Sentido  der-izq.     *)

  IF TablaVal[I]<>4 THEN     (*  Verificacion de existencia del
                                        Termino I.          *)
  FOR N:=M TO NumTer DO
   BEGIN                     (* Contador del ciclo del termino
                               con el cual se compara I.       *)
    IF PAS>6
       THEN J:=NumTer-N        (*       Sentido izq-der.      *)
       ELSE
         IF N<NumTer
               THEN J:=N+1
               ELSE N:=NumTer;   (*    Sentido der-izq.      *)

   IF TablaVal[J]<>4 THEN      (* Verificacion de existencia de*)
   IF TablaVal[I]<>4 THEN      (*   ambos termino, I y J.      *)
    BEGIN
     GOTOXY(1,5);
     WRITE('I:',I);            (*    Blanqueo de banderas    *)
```

```
ELIMIN1:=0;
ELIMIN2:=0;
GOTOXY(8,5);                (*              e              *)
WRITE('J:',J);
TRIDIF:=0;
ELIM:=0;                    (*   iniciacion de contadores. *)
Novar:=0;
FOR K:=1 TO NumVar DO
    VarDif[K]:=0;
FOR K:=1 TO NumVar DO
      BEGIN
        IF (TablaCanon[I,K]=3)
                AND
           ((TablaCanon[J,K]=1) OR (TablaCanon[J,K]=0))
          THEN
           BEGIN            (*  Deteccion de eliminacion *)
             TRIDIF:=TRIDIF+1;
                              (*  del Termino  I.     *)
             Novar:=Novar+1;
             ELIMIN1:=ELIMIN1+1
           END
          ELSE
             IF ((TablaCanon[I,K]=1) OR (TablaCanon[I,K]=0))
                              AND
                        (TablaCanon[J,K]=3)
                THEN
                   BEGIN    (* Deteccion de eliminacion *)
```

```
                    TRIDIF:=TRIDIF+1;
                    ELIMIN2:=ELIMIN2+1; (* del termino J. *)
                    Novar:=Novar+1;
                   END
                 ELSE
               IF (((TablaCanon[I,K]=1) AND (TablaCanon[J,K]=0
                                        OR
                   ((TablaCanon[I,K]=0) AND (TablaCanon[J,K]=1)
                   THEN
                       BEGIN
                            ELIM:=1;
                            VarDif[K]:=1;
                            novar:=novar+1
                       END
      (*  END  DIFERENCIA DE TERMINOS *)
   END; (* FOR k*)
IF ((TRIDIF<>0) AND (ELIMIN1=0) AND (ELIMIN2<>0))
  THEN
                    (* Eliminacion del termino I,
                    y llenado de tablas de terminos
                    que se combinaron en la simplifi-
                    cacion.                          *)
   BEGIN
    CAMB:=1;
    CONT:=1;
    L:=L+1;
    TablaCamb[PAS,L,1]:=I;
    TablaCamb[PAS,L,2]:=J;
```

```
        FOR K:=1 TO NumVar DO
          BEGIN
            IF ELIM=1
              THEN
                  IF VarDif[K]<>0
                     THEN
                         BEGIN
                                 TablaCanon[I,K]:=3;
                                 TablaSal[I,K]:=3;
                                 ContVar[I]:=ContVar[I]-1
                         END
                     ELSE
                  ELSE
                      BEGIN
                          TablaCanon[I,K]:=3;
                          TablaSal[I,K]:=3;
                          TablaVal[I]:=4
                      END
          END;
        gotoxy(20,5);
         writeln(' cambio ')
        END
      ELSE
        IF ((TRIDIF<>0) AND (ELIMIN2=0) AND (ELIMIN1<>0))
           THEN
                     (*  Eliminacion de termino J,
                     y llenado de la tabla de terminos
```

```
            que se combinaron en la simplifica-
            cion.                                    *)
BEGIN
L:=L+1;
TablaCamb[PAS,L,1]:=I;
TablaCamb[PAS,L,2]:=J;
CAMB:=1;
CONT:=1;
  FOR K:=1 TO NumVar DO
   BEGIN
        IF ELIM=1
           THEN
               IF VarDif[K]<>0
                  THEN
                    BEGIN
                           TablaCanon[J,K]:=3;
                           TablaSal[J,K]:=3;
                           ContVar[J]:=ContVar[J]-1
                    END
                  ELSE
           ELSE
               BEGIN
                     TablaCanon[J,K]:=3;
                     TablaSal[J,K]:=3;
                     TablaVal[J]:=4
               END
   END;
   gotoxy(25,6);
```

```
                    writeln(' CAMBIO  ')
                END;
            END;
        END; (* FOR I,J*)
    END
    UNTIL CONT=0
END; (* Fin de SEGUNSIMP *)
```

Por último se tiene el siguiente bloque, que simplifica en base al teorema restante.

El cual es:

$$xy + xy' = x$$

El procedimiento es el siguiente:

```
PROCEDURE TERSIMP;
 (* El procedimiento TERSIMP se basa en el teorema:
                        xy + xy' = x
    Donde x, y pueden ser una cadena de variables.
    La eliminacion del termino depende del sentido en el cual
 se este realizando la comparacion,  siempre siendo el termino
 J el que desaparece.                                          *)
BEGIN
  GOTOXY(50,3);
  WRITE('3a. SIMPLIFICACION');

  REPEAT
    FOR M:=1 TO NumTer-1 DO      (* Contador del termino I. *)
     BEGIN
         IF PAS>6
```

```
    THEN I:=NumTer+1-M   (*  Sentido  izq-der.      *)
    ELSE I:=M;           (*  Sentido  der-izq.      *)
GOTOXY(1,5);
WRITE('I:',I);
CONT:=0;
 IF TablaVal[I]<>4       (* Existencia del termino I *)
   THEN
     BEGIN
       FOR N:=M TO NumTer DO
        BEGIN              (* Contador del termino J.*)
          IF PAS>6
             THEN J:=NumTer-N  (* Sentido izq-der.   *)
             ELSE
                IF N<NumTer
                   THEN J:=N+1  (* Sentido der-izq.  *)
                   ELSE N:=NumTer;
          IF TablaVal[J]<>4
            THEN          (* Existencia del termino J *)
              BEGIN
                GOTOXY(8,5);
                WRITE('J:',J);
                S:=0;
                VADIF:=0; (* Contador de diferencias.*)
                FOR K:=1 TO NumVar DO
                   BEGIN
                     GOTOXY(18,5);
                     WRITE('K:',K);
```

```
IF ((TablaCanon[I,K]<>3)
                AND (TablaCanon[J,K]<>3))
  THEN
  (* Evita comparar terminos No existentes *)
    S:=S+1;
    IF ((TablaCanon[I,K]=1)
        AND (TablaCanon[J,K]=0))
              OR
        ((TablaCanon[I,K]=0)
        AND (TablaCanon[J,K]=1))
      THEN
            (* Aplicacion del teorema:
                  xy + xy' = x.          *)
        BEGIN
          VADIF:=VADIF+1;      (* Contador de
                              diferencias. *)

          LUDIF:=K             (* Lugar de la
                               diferencia. *)
        END;
      END;
    IF ((S=ContVar[I]) AND (S=ContVar[J]))
    THEN
    CASE VADIF OF
    0:BEGIN
       (*  PARA EL CASO DE QUE NO
          HAYA DIFERENCIAS YA SE A-
```

```
        PLICO LA PRIMERA SIMPLI-
        FICACION.                    *)
    END;
1:BEGIN   (* Para 1 o mas diferencias. *)
    CAMB:=1;

    IF ((ContVar[I]=1) AND (ContVar[J]=1))
    THEN      (* Aplicacion del teorema:
                        x + x' = 1
                Donde x es una variable.*)
     BEGIN
       FOR S:=1 TO NumTer DO
        FOR R:=1 TO NumVar DO
          BEGIN
            TablaCanon[S,R]:=3;
            TablaSal[S,R]:=3
          END;
       UNO:=1;
       CAMB:=1
     END
    ELSE
              (* Realización del teorema:
                      xy +xy' = A
                  Eliminandose el termino
              J, y la variable que di-
              fiere en I.
                Llenando la tabla de cam-
```

```
                                        bios y levantando banderas.*)
                                    BEGIN
                                     L:=L+1;
                                     TablaCamb[PAS,L,1]:=I;
                                     TablaCamb[PAS,L,2]:=J;
                                     TablaCanon[I,LUDIF]:=3;
                                     TablaSal[I,LUDIF]:=3;
                                     TablaVal[J]:=4;
                                     ContVar[I]:=ContVar[I]-1;
                                     FOR K:=1 TO NumVar DO
                                         BEGIN
                                             TablaCanon[J,K]:=3;
                                             TablaSal[J,K]:=3
                                         END;
                                     CONT:=1;
                                     GOTOXY(30,5);
                                     WRITE('CONT:',CONT);
                                     GOTOXY(40,5);
                                     WRITE('CAMBIO =',CAMB)
                                   END
                                  END
                                 END
                              END
                             END
                          END
              END;
    UNTIL  CONT=0
END;   (*  FIN del procedimiento TERSIMP  *)
```

La explicación del procedimiento es la siguiente:

Se cuentan el número de cambios que tienen dos términos, o sea el número de variables que tienen diferentes dos términos, por lo que si:

a).- No se tiene una ningun cambio entonces la simplificacioń ya se realizó en el primer procedimiento.

b).- Si solo se tiene un cambio se aplicará el teorema

$$xy + xy' = x$$

donde sa sabe de antemano que "x" puede ser una cadena de variables o una sola variable, pero "y" será una sola variable.

c).- Si se tiene un cambio, y los términos están compuestos por una sola variable, entonces se simplificará por el teorema:

$$x + x' = 1$$

d).- Si se tienen más de un cambio no se aplica ningún teorema.

Estos tres procedimientos se repetirán hasta que ya no exista ninguna simplificación, más se debe concicerar que la simplificación se desarrollará en este orden, por lo que si se hace

a lapiz y no se llega al mismo resultado es que el orden no fué el mismo de la computadora, la diferencia de resultados no implica error alguno en la simplificación.

PASO #6).- CAMBIO DE FORMA NUMERICA A LITERAL.

A continuación se devolverá la ecuación a su forma original de entrada que es la literal, para mejorar la interpretación del resultado.

El procedimiento a desarrollar tomará en cuenta la secuencia de las variables, y de igual manera se ordenarán de la A hasta la Z, ya sean mayúsculas o minúsculas.

La variable que desaparezca en un término debe aparecer como espacio en blanco.

El procedimiento correspondiente a este paso es:

```
 PROCEDURE CANONLIT;
    (*  El procedimiento CANONLIT convierte la ecuacion ya simpli-
      ficada de valor numerico a su literal  correspondiente,  con
      el fin de dar el resultado  en forma literal, para una mejor
      comprension  de la  solucion.  Cambiando los 0's a variables
      testadas y  los 1's a no testadas, las variables no existen-
      tes las cambia a espacios en blanco.                      *)
BEGIN
GOTOXY(10,10);
WRITELN(' ###  RESULTADO DE LA ECUACION BOOLEANA APLICADA  ### ':50
WRITELN('         EN EL PROCESO DE SIMPLIFICACIO NUM.':50,PAS);
WRITELN;
```

```
RET:=0;
IF UNO=1
   THEN
       WRITE(' #####  1  #### ')
   ELSE                          (*   Desplegado del resultado de
                                                  x + x' = 1       *)
   BEGIN
    FOR I:=1 TO NumTer DO
    IF TablaVal[I]<>4 THEN           (* No se desplegaran terminos
                                                  que no existan.  *)
    BEGIN
      RET:=RET+1;          (* Contador de terminos existentes *)
      R:=64;               (* Para cambios de 0's a Mayusculas *)

      FOR J:=1 TO NumVar DO
        BEGIN
         R:=R+1;
         S:=R+32;          (* Para cambios de 1's a normales. *)

         CASE TablaSal[I,J] OF
               1: BEGIN
                     TablaOut[RET,J]:=CHR(S);
                     TablaResul[PAS,RET,J]:=TablaOut[RET,J]
                  END;
               0: BEGIN
                     TablaOut[RET,J]:=CHR(R);
                      TablaResul[PAS,RET,J]:=TablaOut[RET,J]
                  END;
```

```
                2: BEGIN
                      TablaOut[RET,J]:=CHR(88);
                      TablaResul[PAS,RET,J]:=TablaOut[RET,J]
                   END;
                3: BEGIN
                      TablaOut[RET,J]:=CHR(32);
                      TablaResul[PAS,RET,J]:=TablaOut[RET,J]
                   END
                      (* Para desplegar el resultado de este pro-
                         ceso, y guardar en la tabla de almacena-
                         miento de resultados respectivamente. *)
        END
     END
END;

      (*  Desplegado del resultado obtenido  en el proceso de
         simplificacion numero PAS.                          *)
WRITE('****  ');
FOR I:=1 TO RET DO
    BEGIN
         IF I<>1 THEN WRITE(' + ');
         FOR K:=1 TO NumVar DO
            CASE TablaOut[I,K] OF
              'a','b'..'z': BEGIN
                               WRITE(TablaOut[I,K]);
                               R:=1
                            END;
             'A','B'..'Z':BEGIN
```

```
                                        TablaOut[I,K]:=
                                          CHR(ORD(TablaOut[I,K])+32);
                                        WRITE(TablaOut[I,K],'');
                                        R:=1
                                    END;
                END  (* CASE *)
        END;
        IF Cero=1
           THEN WRITE('1')
           ELSE
             IF Cero=2
               THEN WRITE('0');
        WRITELN('  **** ')
    END;
    CanTer[PAS]:=RET;

WRITELN;
WRITELN('MATRIZ DE CAMBIOS':40);
WRITELN;
I:=1;
REPEAT
    WRITE(' ':25);
    FOR J:=1 TO 2 DO
        WRITE(TablaCamb[PAS,I,J]:4);
    WRITELN;
    I:=I+1
 UNTIL TablaCamb[PAS,I,1]=0
```

```
END;  (* Fin del Procedimiento *)
```

Ahora, ya se tiene la ecuación de nuevo en forma literal, lo que facilitará la expresión de salida, debido a que estará en forma de las variables de entrada que se manejaron.

Cada vez que se genere un resultado, este se almacenará para posteriormente compararlo con todos los resultados de todos los procesos.

PASO #7).- SALIDA DE LA ECUACION SIMPLIFICADA.

Como último paso del programa se tiene la salida de la ecuación, junto con los valores de los términos que se compararon y de aquellos que desaparecieron en alguna simplificación, por lo que interesará desplegar estos tres arreglos, los cuales son:

TablaSal, TablaVal y TablaCamb.

El procedimiento correspondiente se omite debido a que simplemente es un desplegado a pantalla de el arreglo TablaOut.

Una vez ya desplegada la información se tienen completas todas las herramientas necesarias para realizar el programa, en el que solo se le dará la secuencia y ordenamiento a estas subrutinas, lo cual viene a ser una tarea demasiado sencilla, sin embargo se mostrará a continuación:

```
    BEGIN  (* PROGRAMA PRINCIPAL *)
BEGIN
     CLRSCR;
     WRITELN('  ********** INSTRUCCIONES DEL PROGRAMA **********  ');
     INSTRUCCIONES;
     PAS:=0;
     REPEAT

        REPEAT
           PAS:=PAS+1;    (* Contador de aplicacion de secuencias. *)
           L:=0;
           IF PAS=1
              THEN ENTRADA;
           LITCANON;
           LLENATABLAS;
           WRITELN(' ***** PROCESO DE SIMPLIFICACION  *****':50);
           REPEAT
             CAMB:=0;          (*    Aplicacion de la secuencia :  *)
             SIMPLIFICACION;   (*              1a                  *)
             SEGUNSIMP;        (*              2a                  *)
             TERSIMP;          (*              3a                  *)
           UNTIL CAMB=0;
           CANONLIT;

           PAS:=PAS+1;
           L:=0;
           LITCANON;
           LLENATABLAS;
           REPEAT
```

```
    CAMB:=0;
    SIMPLIFICACION;  (*          1a          *)
    TERSIMP;         (*          3a          *)
    SEGUNSIMP;       (*          2a          *)
  UNTIL CAMB=0;
  CANONLIT;

  PAS:=PAS+1;
  L:=0;
  LITCANON;
  LLENATABLAS;
  REPEAT
    CAMB:=0;
    SEGUNSIMP;       (*          2a          *)
    SIMPLIFICACION;  (*          1a          *)
    TERSIMP;         (*          3a          *)
UNTIL CAMB=0;
CANONLIT;

PAS:=PAS+1;
L:=0;
LITCANON;
LLENATABLAS;
REPEAT
  CAMB:=0;
    SEGUNSIMP;       (*          2a          *)
    TERSIMP;         (*          3a          *)
    SIMPLIFICACION;  (*          1a          *)
```

```
    UNTIL CAMB=0;
    CANONLIT;
    PAS:=PAS+1;
    L:=0;
    LITCANON;
    LLENATABLAS;
    REPEAT
      CAMB:=0;
      TERSIMP;          (*          3a          *)
      SEGUNSIMP;        (*          2a          *)
      SIMPLIFICACION;   (*          1a          *)
    UNTIL CAMB=0;
    CANONLIT;

    PAS:=PAS+1;
    L:=0;
    LITCANON;
    LLENATABLAS;
    REPEAT
      CAMB:=0;
      TERSIMP;          (*          3a          *)
      SIMPLIFICACION;   (*          2a          *)
      SEGUNSIMP;        (*          1a          *)
    UNTIL CAMB=0;
    CANONLIT;

  UNTIL PAS>=NumSimp;

    SELECRES;
```

```
    WRITELN;
    WRITE('******** DESEAS CONTINUAR ?  (y/n) ******');
    READ(KBD,CONTINUA);
    PAS:=0;
  UNTIL CONTINUA='n';
  CLRSCR;
  WRITELN(' ##### ESTAS FUERA DEL PROGRAMA #####')
END.      (* FIN DEL PROGRAMA PRINCIPAL *)
```

Después de haber dado el programa principal, el cual consiste en solo secuencias de manejo de procedimientos, viene el último procedimiento, en el cuál se elegirá el resultado más óptimo de los generados.

Las condiciones de la elección ya fueron previamente aclaradas.

El procedimiento encargado de la salección apropiada es el siguiente:

```
PROCEDURE SELECRES;
(* El procedimiento SELECRES, selecciona el resultado que contenga
 un menor numero de variables y terminos, de todas las  secuencias
 de simplificacion aplicadas en todo el programa.                *)
BEGIN
  WRITELN;
```

```
  WRITELN;
  WRITE(' PRESIONA <RETORNO> PARA SELECCIONA LA ECUACION  ');
  WRITE('MAS SIMPLIFICADA.');
  READLN(S);
  CLRSCR;
   (*  Muestreo para la seleccion del la ecuacion mas optima. *)
FOR PAS:=1 TO NumSimp DO
    BEGIN
        R:=0;     (* Iniciacion del contador de variables. *)
        FOR I:=1 TO CanTer[PAS] DO
          BEGIN
            IF I<>1
               THEN WRITE(' + ');
            FOR K:=1 TO NumVar DO
                IF TablaResul[PAS,I,K]<>CHR(32)
                   THEN
                       BEGIN
                          R:=R+1;  (* Incremento del contador
                                                de variables.*)
                          WRITE(TablaResul[PAS,I,K])
                       END;
          END;
        SelResul[PAS]:=R;
        WRITELN('No. de variables:':20,SelResul[PAS]);
    END;

      (* Seleccion de la ecuacion mas optima. *)
```

```
PAS:=1;
R:=SelResul[1];
FOR J:=1 TO NumSimp DO
  IF R>Selresul[J]
    THEN
        BEGIN
             R:=SelResul[J];
             PAS:=J
        END;
WRITELN;
WRITE('...La ec. selecionada es la #',PAS,'...');
WRITELN('..... tiene ',R,' variables.....');
WRITELN;
WRITELN;
WRITELN(' ***** "  SALIDA DEL RESULTADO  ELEGIDO. " *****');
WRITELN;
IF Cero=2
  THEN
    WRITE(' **** O ****')
  ELSE
    IF ((Cero=1) or (R=0))
      THEN
        WRITE(' ### 1 ###')
      ELSE
  BEGIN
  WRITE('  !!!!!--- ');
    FOR I:=1 TO CanTer[PAS] DO
        BEGIN
```

```
            IF I<>1
                    THEN
                        WRITE(' + ');
                FOR K:=1 TO NumVar DO
                    CASE TablaResul[PAS,I,K] OF
                        'a','b'..'z': BEGIN
                                          WRITE(TablaResul[PAS,I,K])
                                      END;
                        'A','B'..'Z': BEGIN
                                       J:=ORD(TablaResul[PAS,I,K])+32
                                       TablaResul[PAS,I,K]:=CHR(J);
                                       WRITE(TablaResul[PAS,I,K],'')
                                      END;
                    END (* CASE *)
            END; (* I *)
            WRITELN(' ---!!!!!')
        END
    END; (* SELECRES *)
```

Con esto se da por terminado el capitulo IV y para una mejor comprensión del objetivo de esta tesis se recomienda correr el programa y verificar los resultados, aplicandose a necesidades especificas con datos reales.

" C A P I T U L O I V "

IV).- APLICACIONES .

CAPITULO IV.

IV).- APLICACIONES.

El siguiente capítulo tiene fin dar una idea de lo importante que es el Algebra Booleana en el mundo electrónico.

Debido a que esta tesis es orientada hacia ingeniería se la dará el enfoque práctico a esta área, aclarando que existen muchos campos , los cuáles son posible mencionar.

No se profundizará demasiado, ya que se considera que el lector tiene conocimientos básicos de electrónica por lo que los conceptos se dejan a criterio del lector.

Todos los componentes electrónicos se pueden representar por medio de una ecuación matemática y dentro del Algebra Booleana esta no es la excepción, por lo que a continuación se presentarán diversos componentes y su ecuación booleana correspondiente.

*).- " BLOQUES LOGICOS ELECTRONICOS. "

Se ejemplificarán los bloques con solo 2 entradas por simple facilidad de manejo, pero debe considerarse que es posible tener un número mayor de estas.

Debido a que interesa más la lógica que la electrónica, se explicará solamente su relación con el Algebra Booleana.

Las entradas que se indiquen con (-), implica un nivel bajo de energía y por consiguiente las entradas que se indiquen con un (+) implican un nivel alto de energía.

A).- Bloques Lógicos con Diodos.

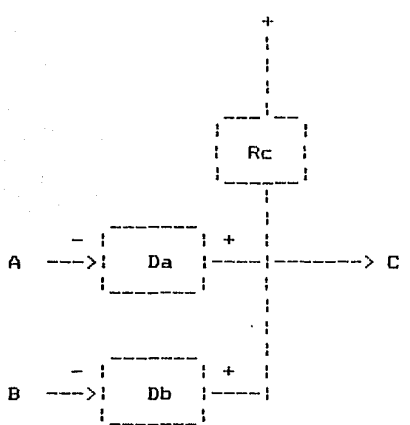
La siguiente figura muestra una compuerta AND con diodos.

FIGURA #1

Rc.- Resistencia de Carga.

Da.- Diodo A.

Db.- Diodo B.



CIRCUITO AND

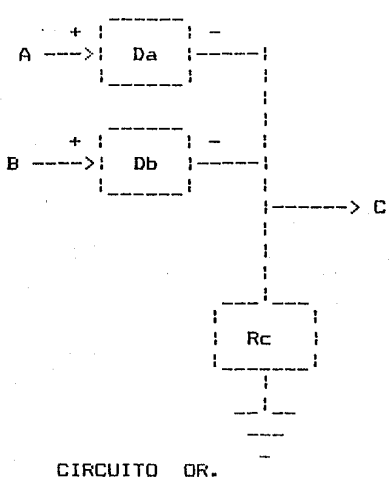
A	B	C
-	-	-
-	+	-
+	-	-
+	+	+

TABLA AND

Explicación del circuito:

La salida C estará al mismo nivel de voltaje que el nivel más bajo de voltaje de entrada, asi pues para que el nivel de voltaje de salida sea " + ", ambas entradas deben ser " + ", ya que si alguna entrada es " - ", toda la corriente se irá por esa entrada, quedando la salida C en " - ".

FIGURA #2



CIRCUITO OR.

Rc.- Resistencia de carga.

Da.- Diodo A.

Db.- Diodo B.

A	B	C
-	-	-
-	+	+
+	-	+
+	+	+

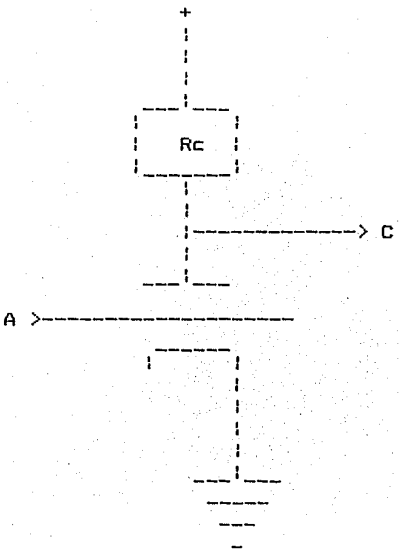
TABLA OR.

Explicación del circuito:

La salida estará al mismo nivel de voltaje que el nivel máximo de voltaje de entrada, por lo tanto la salida es " + " si cualquier entrada es " + ".

B).- Bloques lógicos con tubos al vacio.

FIGURA #3



Rc.- Resistencia de Carga.

A	C
-	+
+	-

TABLA NOT

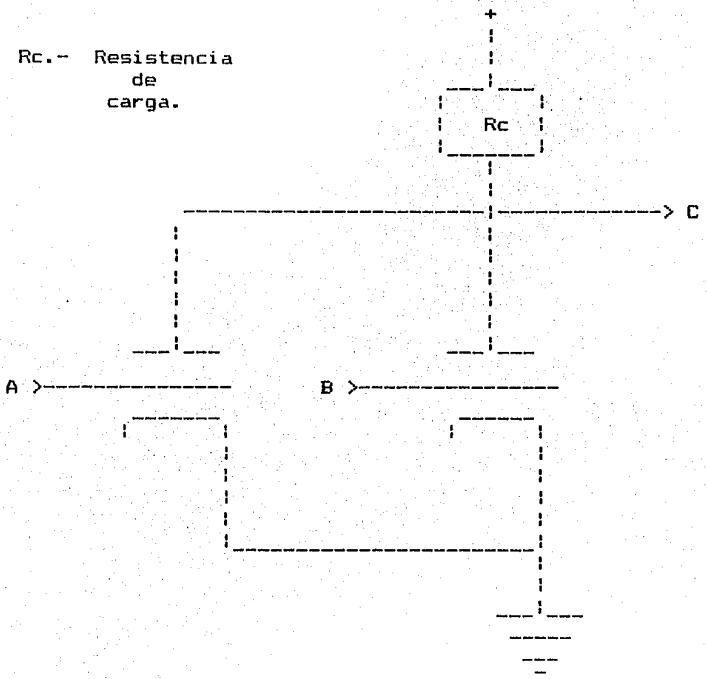
CIRCUITO NOT

Explicación del circuito:

Utilizando un solo triodo de tubo al vacio, si la entrada en la rejilla esta a nivel bajo, (-), se evita la conducción a travéz del triodo, no fluye corriente a travéz de la Resistencia de Carga y la salida está al nivel alto de el voltaje. Si la entrada de rejilla se encuentra en nivel de voltaje alto, ocurre conducción a travéz del triodo, fluye corriente por la resistencia de carga ocacionando una caida de voltaje a travéz de ella y la salida C esta a nivel bajo de voltaje.

FIGURA #4

Rc.- Resistencia de carga.



CIRCUITO NOR.

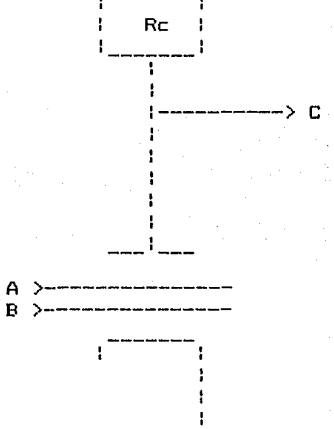
A	B	C
-	-	+
-	+	-
+	-	-
+	+	-

TABLA NOR

Explicación del circuito:

El triodo múltiple tiene sus placas conectadas a una Resistencia de carga común, Si toda la entrada de la reja se encuentra a nivel bajo de voltaje, ninguno de los triodos conduce, no hay corriente a través de la resistencia de carga y la salida está a un nivel alto. Si cualquier entrada se encuentra en un nivel alto, habrá conducción a través de ese triodo, la corriente fluirá a través de la resistencia de carga ocacionando una caída a travez de ella y la salida estará en nivel bajo de voltaje.

FIGURA #5 Rc.- Resistencia de Carga.



CIRCUITO NAND

A	B	C
-	-	+
-	+	+
+	-	+
+	+	-

TABLA NAND

Explicación del circuito:

Se usa un pentodo (este tubo al vacio tiene una tercera rejilla que no se muestra). Si cualquier entrada de rejilla se encuentra en nivel bajo de voltaje ; la conducción a travéz del pentodo no existe y la salida es a nivel alto de voltaje. Si ambas entradas de rejilla se encuentran a nivel alto de voltaje el pentodo conduce y la salida se encuentra en nivel bajo de voltaje.

Como se puede observar las funciones básicas booleanas se pueden implementar fácilmente con los elementos más comunes.

C).- Bloques lógicos con Transistores.

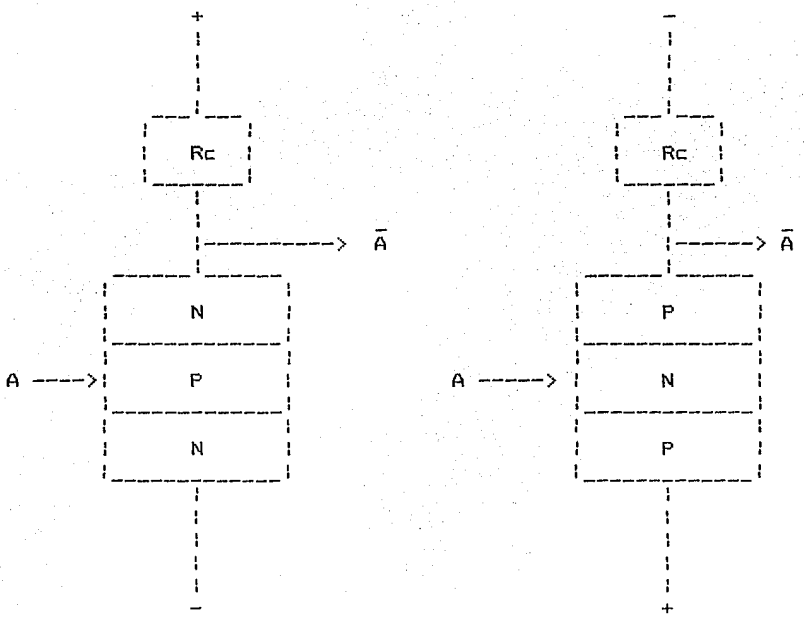
Con el invento de el transistor se facilitó mucho más la implementación de circuitos electrónicos a partir de las funciones booleanas debido a la gran versatilidad que posee este componente electrónico.

A continuación se muestrarán solamente los circuitos lógicos considerandose que el funcionamiento y clase de transistor ya es conocido por el lector. (En caso contrario consultar la referencia #5).

También se omitirán las tablas de verdad debido a que estas ya son familiares, en caso contrario verificarse en las figuras anteriores.

Un inversor se puede representar de la siguiente manera:

FIGURA #6



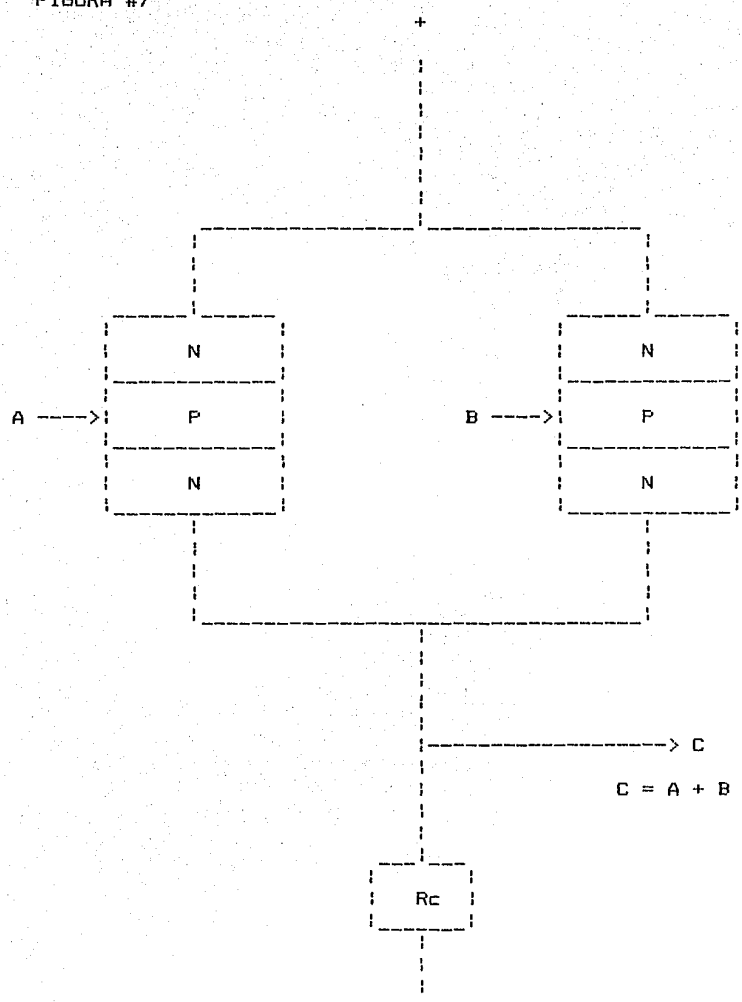
INVERSOR NPN INVERSOR PNP

Explicación del circuito:

En caso de que exista un nivel alto de energía en A, el transistor conduce, quedando C en un nivel bajo de energía, en caso contrario, al no estar polarizada la base del transistor, este no conduce, llendose toda la energía por C.

CIRCUITOS OR (Seguidores de emisor nPn).

FIGURA #7

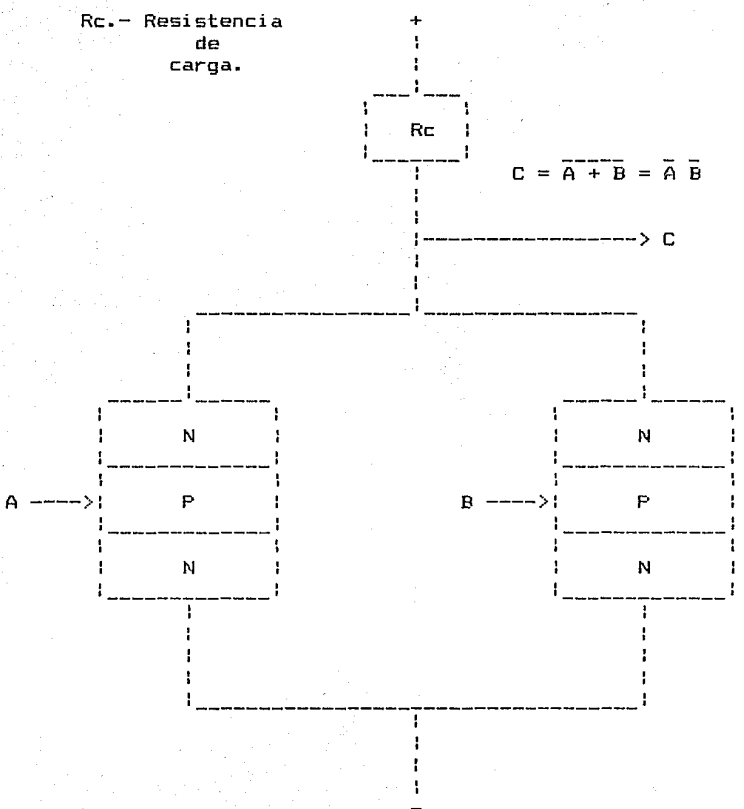


Su explicación se omite por considerarse ovbia, en caso

contrario consultar referencia #5.

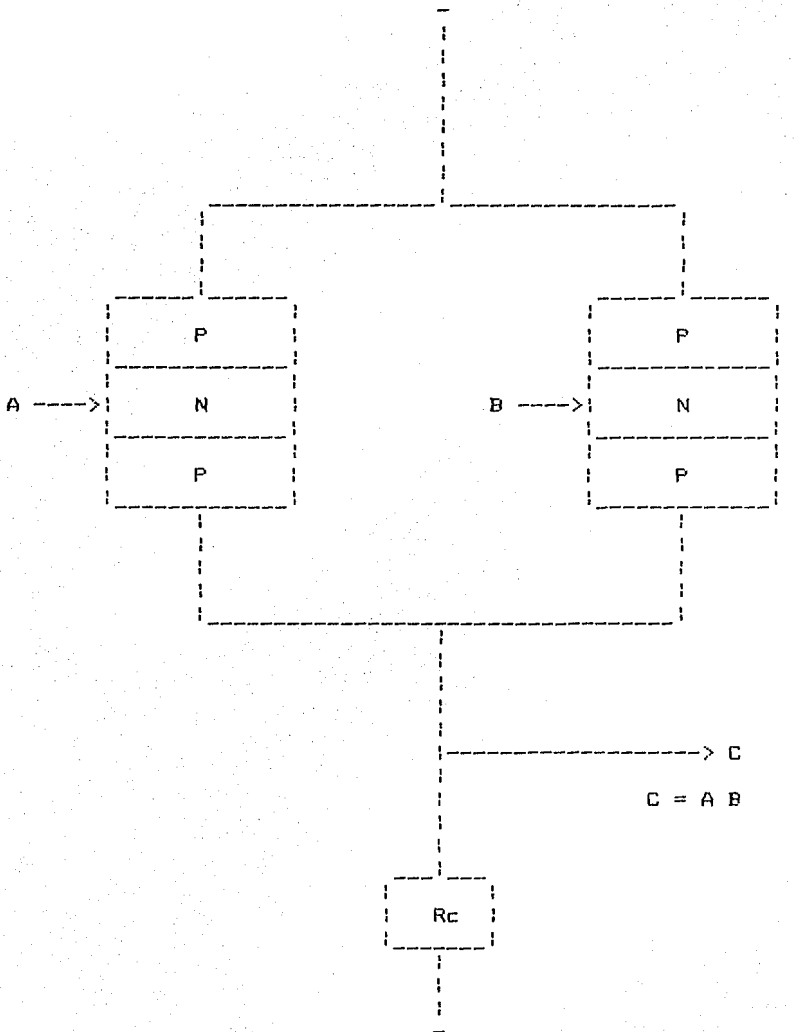
CIRCUITO NOR (Inversores nPn).

FIGURA #8



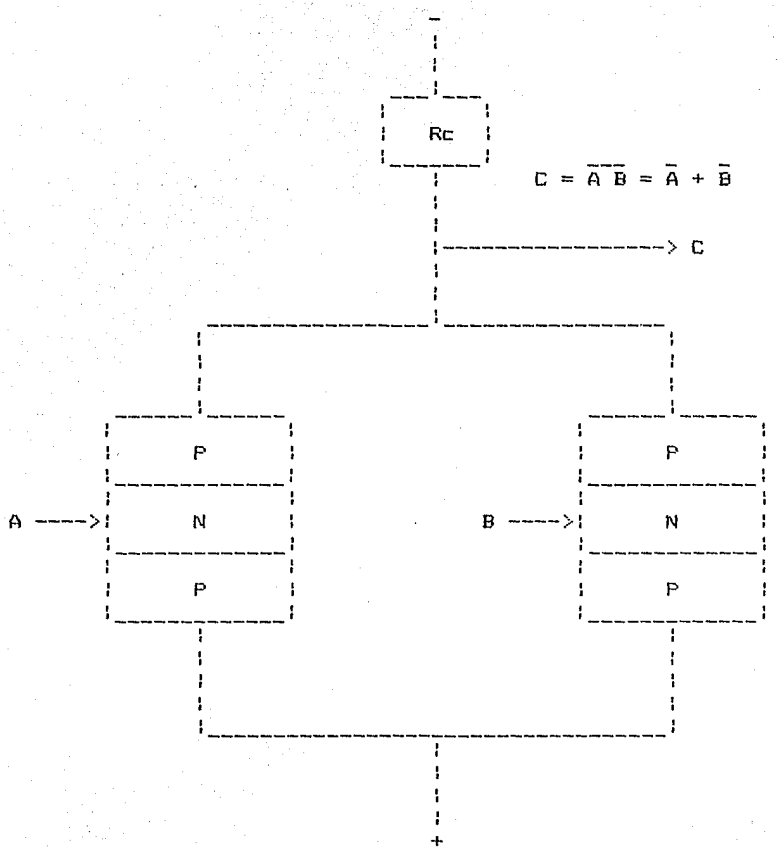
CIRCUITO AND (Seguidores de Emisor pNp).

FIGURA #9



CIRCUITO NAND (Inversores pNp).

FIGURA #10



Como se puede observar, por medio de los componentes electrónicos básicos es posible implementar las principales compuertas, de igual manera analizando cuidadosamente la construcción de los diagramas, es posible implementar circuitos que realicen las simplificaciones utilizadas como teoremas o postulados durante el desarrollo de la presente tesis. Más el objetivo principal de esta tesis es desarrollar un programa que realice la simplificaciones necesarias y no la implementación del circuito electrónico que desarrolle esta simplificación, por lo que si se realizara por medio de electrónica ya sería un trabajo de hardware y no software.

Pero se debe considerar que el uso que se le dé a este programa es totalmente independiente, lográndose aplicaciones útiles en cualquiera de los siguientes campos de la electrónica o electricidad:

* Redes de Contacto:
  - Redes no planares.
  - Circuitos puente.
  - Contactos de Transferencia.
  - Redes de contacto con salida Múltiple.

* Arbol: De Relevadores y Electrónicos:
  - Arbol de Relevadores.
  - Arboles Electrónicos.

* Sistemas Numéricos:

- Códigos de detección de error.
- Códigos de corrección de error.

* Circuitos Secuenciales.
* Circuitos Combinacionales.

Y si se deseara enumerar absolutamente todos los campos posible de aplicación en el area de la electrónica, esta lista sería bastante grande, más con esto se concidera suficiente para dar una idea de la gran importancia que tiene el Algebra Booleana en el campo de la electrónica.

Si se deseara mencionar otras áreas de aplicaciones posibles con tomar en cuenta lo importante que es saber los resultados precisos de la aplicación de un Test psicológico en el área de la industria, la medicina o la investigación, esta tesis puede dar resultados lo bastante confiables, seguros y rápidos como el mejor de los conocedores de el area que se desea aplicar el test. Por lo tanto, es posible enfocar esta tesis a cualquier rama que se desee o se considere factible su ayuda.

" C A P I T U L O V "

" CONCLUSIONES Y RECOMENDACIONES. "

" CONCLUSIONES Y RECOMENDACIONES. "

El amplio mundo electrónico y cientifico en que vivimos exige más, y de forma continua, del esfuerzo del estudiante para que este forme parte de la vida cotidiana de todo ser humano preparado profesionalemnte. Es por esto que la presente tesis viene a ser una arma nueva con la que cuenta el estudiante, tanto profesional como técnico para acelerar el aprendizaje y comprensión de las matéticas en el mundo electrónico.

Es por esto que el programa desarrollado se recomienda como ayuda para la solución de problemas booleanas demasiado largos y complejos, donde la confiabilidad de el resultado debe ser total y segura.

Para aquellos problemas que sean cortos y sencillos no es recomendable debido a que es talvéz más rápido resolverse a lápiz pero para aquellos que no tengan habilidad en el manejo de el Algebra Booleana pueden usarlo con absoluta confianza.

Muchas ecuaciones booleanas pueden ser sencillas pero largas entonces la obtención de un resultado confiable y óptimo viene a recaer en el estado emocional que se encuentre la persona a resolver la ecuación, ya que si se encuentra fatigado las probabilidades de caer en un error son mayores. Se esta hablando de ecuaciones que puedan tener de 7 a más variables y de 5 o más términos, por tan solo de imaginarse en resolver una ecuación de 15 variables con 10 términos prepara a la persona para enfrentarse a un problema tedioso y complicado de resolver,

independientemente del acomodo de las varibles o términos, ya que el problema esta en la cantidad de variables a manejar y no tanto en lo complicado de la combinación de las variables.

Es por esto que para casos como el anterior es recomendable la utilización del programa desarrollado en esta tesis.

REFERENCIAS BIBLIOGRAFICAS.

[1] Eldon Whitesitt. Algebra Booleana y sus Aplicaciones. México. C.E.C.S.A. Feb/1978.

[2] C. E. Strangio. Electrónica Digital. México, Interamericana, 1984.

[3] M. Morris Mano. Lógica Digital y diseño de Computadoras. Texas, E. U. Prentice Hall, Interamericana. Enero/1982.

[4] Mitchel P. Marcus. Circuitos de Conmutación para ingenieros. México. Diana, Feb/1980.

[5] Ibid. pp 214.

[6] Grogomo. Programación en Pascal. México, D. F. Fondo Educativo Interamericano,1980.

[7] Murray Lazo, Chicoriel Uziel. Aplicaciones de la Computación a la Ingenieria. México, Limusa, Abril/1984.

[8] Enrique Mandado. Problemas Digitales. España, 1977.