84
2 Ejem.

Universidad Nacional Autónoma de México

FACULTAD DE INGENIERIA

CONVERTIDOR DE CODIGO ASCCI-MORSE REVERSIBLE (USANDO FPLA'S)

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A

JOSE L. MAGAÑA VILLASEÑOR

DIRECTOR DE TESIS:

ING. JORGE E. LAVIN MARTINEZ

CIUDAD UNIVERSITARIA, MEXICO, D. F. 1985

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

C O N T E N I D O

I N T R O D U C C I O N

El propósito que se persigue al desarrollar este trabajo, del cual no se pretende agotar el tema, es mostrar las características y funcionamiento de un componente novedoso como una alternativa en la solución de los diseños de sistemas digitales y se pone de manifiesto con el diseño de un convertidor de código ASCII-MORSE-ASCII, el cual puede ser utilizado dentro de sistemas telegráficos en los medios rurales de nuestro país, en donde aún utilizan la "llave Morse", como un medio de transmisión de mensajes.

Aunque siempre tendremos la necesidad de una lógica que ofrezca el mejor funcionamiento posible -- sin tomar en cuenta el costo, la mayoría de los diseños correponden a una lógica que nos brinda:

a) Funcionamiento aceptable a bajo costo.
b) Buen funcionamiento a un costo óptimo.

Esto es, un aspecto importante en el trabajo de diseño de lógica, es la búsqueda constante del mejor compromiso posible entre el funcionamiento y el costo del diseño. El tipo de curva que determina la decisión entre éstos, se muestra en la Fig. 1--- en la que se destacan dos puntos importantes:

(B) Que corresponde a la base o costo mínimo, y que

traslada la curva a la derecha o a la izquierda.

(C) Que corresponde al codo de la curva.

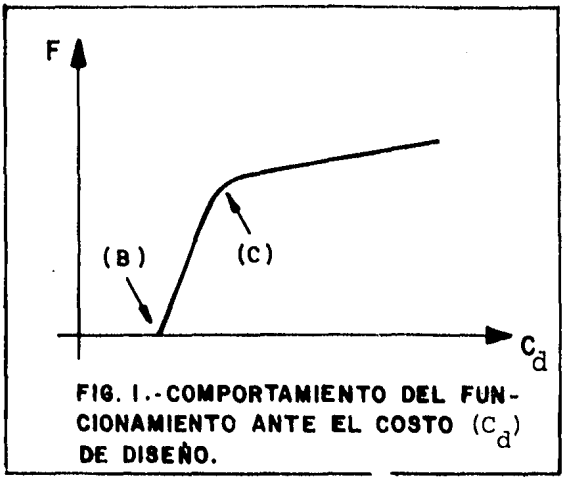


FIG. 1.-COMPORTAMIENTO DEL FUNCIONAMIENTO ANTE EL COSTO (C_d) DE DISEÑO.

Anteriormente, cuando los circuitos lógicos se construían con componentes discretos o incluso con pequeños bloques de unidades lógicas tales como compuertas NAND o NOR, el codo de la curva se determinaba fácilmente; el costo dependía directamente del número de componentes usados.

Dado que en un diseño con tres a cinco niveles se tiene un número adecuado de circuitos, menos niveles implicarían un incremento significativo del costo, en tanto que un aumento de niveles, mejoraría el costo insignificantemente.

La base de la curva (B), estaba también bien definida y era determinada por el tiempo total utilizado en el diseño y en menor grado por el número de aplicaciones diferentes de un grupo de lógica dado. Los costos de ensamble, fueron esencialmente una -- función del número de circuitos, ya que los compo-- nentes individuales costaban lo mismo en cualquier circuito en que fueran usados.

Con la llegada de la Gran Escala de Integración (LSI), los factores que determinaban esta curva - - "Costo-funcionamiento", comenzaron a cambiar drásticamente; en primer lugar, el costo tuvo que ser considerado en términos del área total de silicón, en comparación con el número de componentes agrupados dentro de esta área; este factor determinaba la forma de la curva de la Fig. 1, pero no el desplazamiento de (B). Este último es afectado por un grupo de cantidades que incluyen el tiempo de diseño - manual, costos de diseño automatizados, etc. En particular, el mayor número de componentes fabricados actualmente para un diseño dado, dan una (B) -- menor.

Por otro lado, una vez que los componentes han - sido fabricados, se debe considerar el costo de - - prueba. Finalmente, la posibilidad de falla durante su uso también se incrementa con la complejidad del componente.

Los resultados de los comentarios anteriores, son que actualmente podemos obtener un buen rendimiento de nuestro gasto, solo si hacemos grandes cantidades de un componente dado; en la práctica, relativamente pocos componentes hacen su costo competitivo, ya que su uso no llega a ser lo suficientemente amplio; esto se origina a causa de los cambios de ingeniería que estan asociados con errores de diseño, así como cambios en la filosofía del desarrollo de la máquina. El problema aumenta después que la máquina se entrega al cliente; debido a nuevos adelantos, requerimientos específicos del cliente, cambios de filosofía, etc.

Es aconsejable entonces, que para determinar si nuestro diseño se encuentra dentro del rango óptimo de "funcionalidad-costo", hagamos una estimación de acuerdo a la gráfica de la Fig. 1.; sobre todo en aquellos proyectos de construcción a gran escala.

Los diseñadores lógicos advirtieron un aparente vacío entre la lógica TTL y la LSI. Los productos diseñados en base a componentes discretos TTL, consumen una cantidad no aceptable de espacio físico y de potencia eléctrica, y los productos diseñados con componentes LSI programados con software (microprocesadores), aunque ofrecen una alta densidad y necesitan relativamente poca potencia para realizar casi cualquier cosa imaginable, el diseñador paga

un alto precio en la realización de software, y todavía tendrá que usar componentes discretos para acoplarlos con el mundo exterior.

Hasta recientemente, no se encontraban componentes que proporcionaran un camino real y efectivo para salvar ese vacío; los fabricantes, al ver esta necesidad, crearon un nuevo componente. El PLA (Programmable Logic Array), el cual marca una nueva era de circuitos integrados, comparable a la introducción de componentes de MSI, en los días cuando las compuertas y los Flip-Flops eran los únicos circuitos integrados disponibles.

Los arreglos lógicos programables (PLA's), como su nombre lo indica, son arreglos de elementos lógicos tales que, para una función de entrada dada, producen una función de salida conocida; en este sentido, dichos componentes podrían ser tan simples como una compuerta o tan complejos como una memoria ROM.

Aunque cualquier código de entrada puede ser decodificado a cualquier código de salida, en el PLA no todas las combinaciones posibles de las entradas son reconocidas por un mismo componente; el número de entradas en los PLAs, va desde cuatro hasta veinte, mientras que el número de palabras o productos parciales, se ha normalizado en 96 para los pro

gramados en fábrica y en 48 para los programados por el usuario, aunque con toda seguridad en un futuro próximo podrán ser encontrados en otras proporciones.

Cada producto se puede describir como una función lógica AND, y puede ser programado con cualquier grado de complejidad, mientras el límite de entradas lo permita. Es posible combinar cualquiera de los productos en cualquiera de las diferentes salidas, estableciendo las combinaciones del código de salida deseado; lógicamente, cualquiera o todos los productos (términos AND) pueden combinarse en cada salida (términos OR).

Un PLA no necesita más de una salida, pero en general, es más eficiente construirlos con más de una; los más comunes, tienen 8 terminales de salida y éstas pueden presentarse como verificadas activas altas o verificadas activas bajas según convenga en el diseño; para usar al PLA como almacenamiento de memoria, las ecuaciones deben ser escritas o tabuladas de tal manera que puedan ser programadas en él.

Al diseñar el sistema, se debe buscar una minimización óptima de los productos, ya que los términos comunes pueden ser combinados dentro del mismo paquete; generalmente, la agrupación de productos comunes es una de las ventajas ofrecidas por los arreglos lógicos programables.

CAPITULO I

E S T R U C T U R A

El número de terminales que comunmente tiene un arreglo lógico programable, depende básicamente del número de entradas, el número de salidas y aquellas terminales que sirven para polarizarlo y programarlo, lo cual es establecido por el fabricante. Así, tenemos que el DM7575/DM7576 de la National Semiconductor, tiene 14 terminales de entrada, 8 de salida y 2 de polarización; el IM5200 de la Intersil, tiene 14 terminales de entrada, 8 de salida y 2 para su polarización; el 82S100/82S101 de Signetics, tiene 16 terminales de entrada, 8 de salida, 2 de polarización, 1 habilitadora del componente y 1 de programación; el 5775A/5776A de Monolithic Memories, tiene 12 terminales de entrada, 2 habilitadoras del componente, 8 de salida y 2 de polarización.

Sin embargo, desde el punto de vista de funcionamiento, todos ellos pueden ser divididos en cuatro bloques fundamentales (Fig. 1.1) y son:

1)- El decodificador de entrada.
2)- La matriz "AND".
3)- La matriz "OR".
4)- El arreglo de salida.

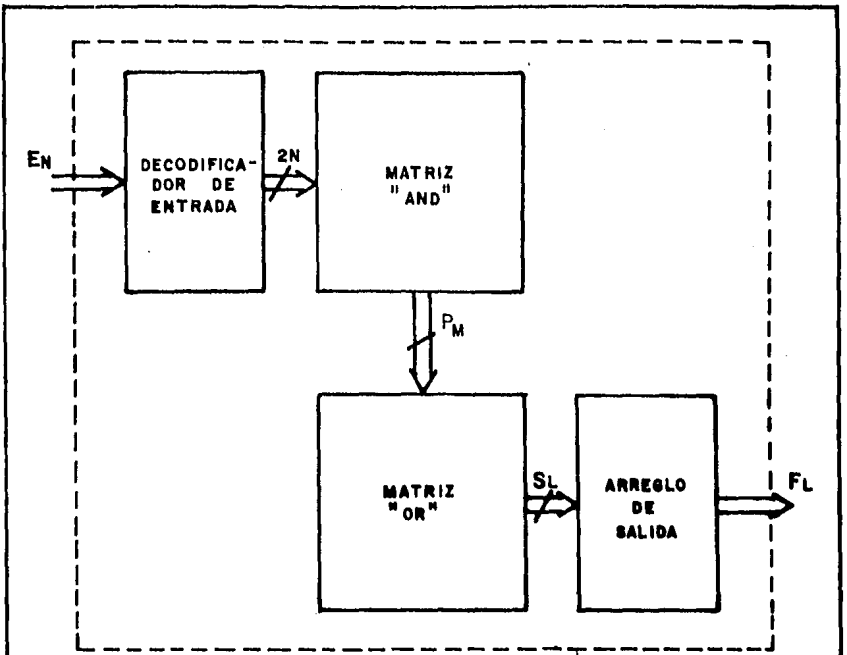


FIG. I.I EXTRUCTURA DE UN PLA.

1.1 ESQUEMA GENERAL

El PLA es una memoria de solo lectura con características muy particulares y la manera en que procesa la información es la siguiente: Una vez que se ha colocado una dirección a la entrada (como se - - muestra en la Fig. 1.2) el decodificador se encarga de transmitir el estado verdadero y el complemento de cada una de las "N" variables de entrada "E" a - la matriz "AND", que está formada por "M" compuer--

tas "P" de "2N" entradas cada una y podrán o no reconocer la dirección (dependiendo de las especificaciones), dando una salida verdadera en caso de serlo, o falsa en caso contrario; éstas a su vez, estan conectadas a la entrada de la matriz "OR", la cual está formada por "L" compuertas "S" de "M" entradas cada una, que permitirán la transmisión al bloque de salida, dependiendo de la programación realizada; una vez en el bloque de salida, cada una de las funciones "F" podrán ser verificadas altas o bajas según lo desee el diseñador.

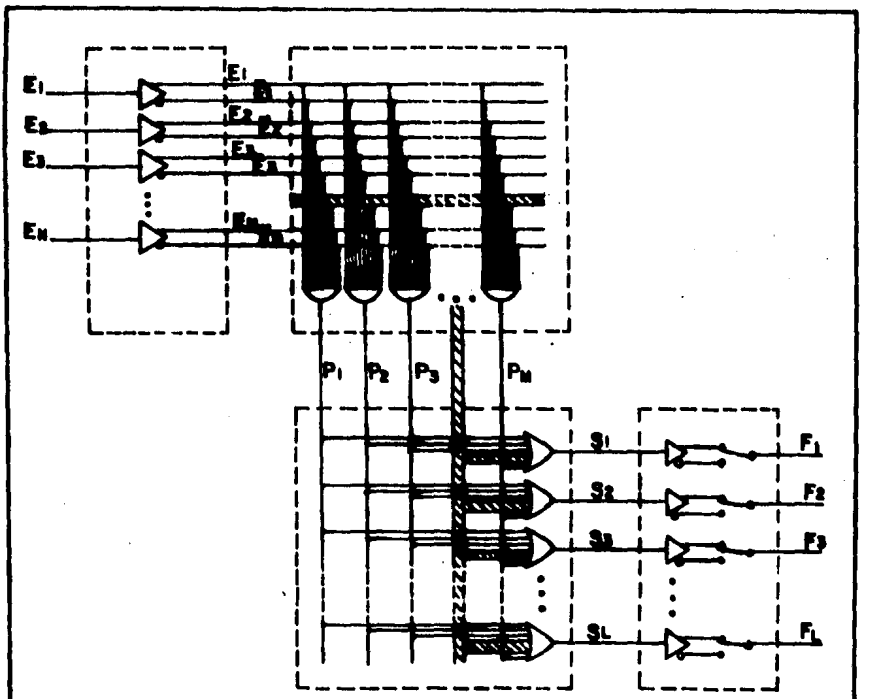


FIG. I.2 DIAGRAMA LOGICO DE UN PLA.

1.2 DECODIFICADOR DE ENTRADA

La decodificación a la entrada de los PLA's, no se realiza con circuitos convencionales, es decir, aquellos en los cuales se generan internamente todos los mintérminos de las variables de entrada y solo dan una salida por cada combinación posible de éstas. Si se tiene presente que el número de entradas a un PLA es grande, con un decodificador convencional, el número de salidas y el área ocupada en el componente por su circuito, sería también bastante grande; por ejemplo, si utilizáramos un decodificador convencional en un PLA con 14 entradas, se tendrían 16,384 salidas y una cantidad ligeramente mayor de compuertas que ocuparían un espacio prohibitivo. Por otro lado, los PLA's tienen la ventaja de poder escoger un grupo de productos de las variables de entrada de entre todas las posibles combinaciones, y esta ventaja se perdería.

La decodificación de las variables de entrada es realizado por un grupo de "N" decodificadores de una variable (Fig.13), llamados también generadores de verdadero-complemento (Phase splitter), de manera que a la salida de éstos, se tiene el estado verdadero y complementado de cada una de las variables, con lo cual, en el caso del ejemplo antes mencionado, se tendrían solo 28 salidas en lugar de 16,384 y se conservaría la propiedad de poder escoger un

pequeño grupo de combinaciones de entre todas las - posibles que pueden presentarse.

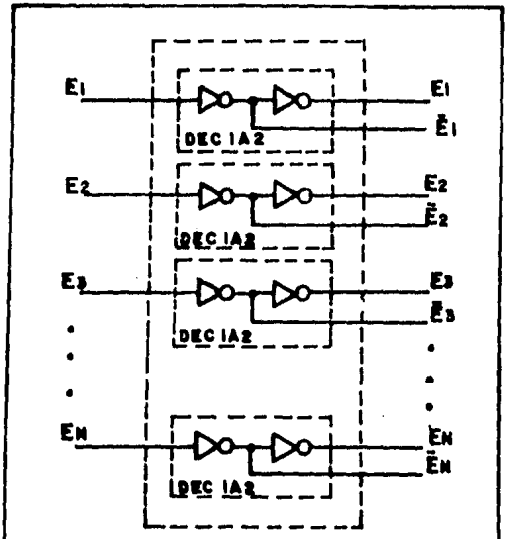


FIG. 1.3 DECODIFICADOR DE ENTRADA.

1.3 MATRIZ "AND"

En este arreglo son combinados los estados verdaderos y/o complementados de las variables de entrada. En su estado original, cada compuerta "AND" - del arreglo, tiene en sus entradas el estado verdadero y complementado de cada una de las variables - de entrada; es decir, si tenemos 14 variables de entrada, las compuertas "AND" deberán tener 28 entradas, de las que solo se dejarán conectadas aquellas que formen el producto deseado; por ejemplo, si que

remos obtener un producto $E_1\bar{E}_2\bar{E}_3E_N$ (de solo cuatro variables) en un PLA donde N=6 entradas, se tendrían que desconectar de las entradas de la compuerta "AND" correspondiente, las variables: $\bar{E}_1$, E_2, E_3, E_4, $\bar{E}_4$, E_5, $\bar{E}_5$, $\bar{E}_6$. Nótese que el producto deseado es equivalente a la suma Booleana de los mintérminos: $E_1\bar{E}_2\bar{E}_3E_4E_5E_6 + E_1\bar{E}_2\bar{E}_3\bar{E}_4E_5E_6 + E_1\bar{E}_2\bar{E}_3E_4\bar{E}_5E_6 + E_1\bar{E}_2\bar{E}_3\bar{E}_4\bar{E}_5E_6$ mostrando así la gran capacidad de compresión lógica de los PLA's que es una de las ventajas con las que cuenta; ésta es posible ya que en el PLA no se requiere almacenar los estados falsos, pues cuando se presenta una dirección inválida, el PLA no la reconoce y dá salidas falsas.

En el caso de los arreglos lógicos programables por el usuario, la implementación física de la estructura "AND" es realizada con diodos del tipo zener, en cuyo ánodo se aplica un nivel alto permanente a través de un fusible y una resistencia (Fig. 1.4) y su cátodo acepta el estado verdadero o complementado de una de las variables de entrada; así, si nosotros queremos obtener los productos $P_1 = E_1\bar{E}_2\bar{E}_3E_n$, $P_2 = \bar{E}_1\bar{E}_2$, $P_3 = 0$ y $P_M = 1$: para P_1 los fusibles correspondientes a los diodos en cuyo cátodo se encuentren las variables E_1, $\bar{E}_2$, $\bar{E}_3$ y E_n, deberán dejarse intactos y deben quemarse todos los demás; para P_2, los fusibles correspondientes a los diodos en cuyo cátodo se encuentren las variables $\bar{E}_1$ y $\bar{E}_2$, deberán dejarse intactos y deben de -

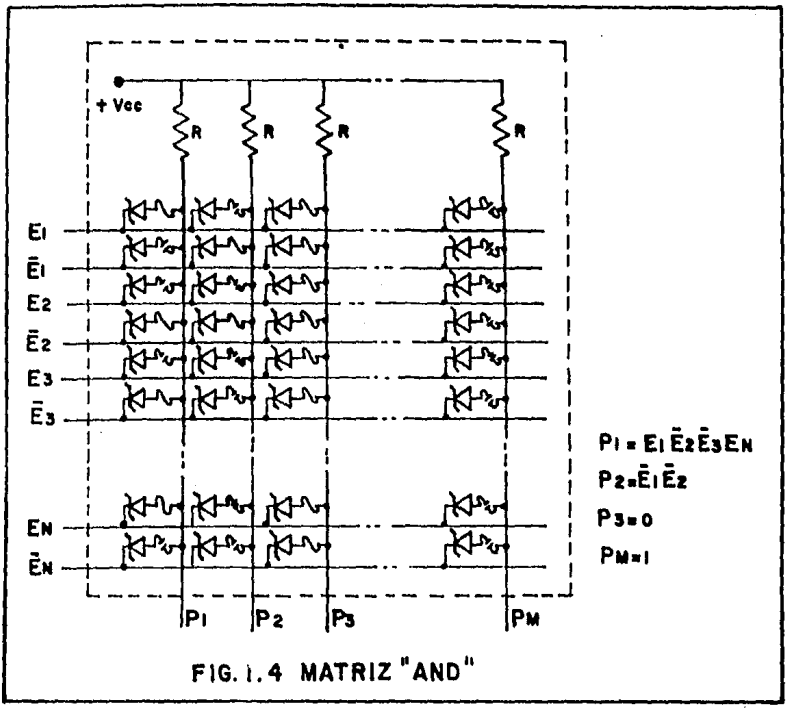


FIG. I.4 MATRIZ "AND"

quemarse todos los demás; si todos los fusibles de la columna P_3 se dejan intactos, se tendrá un cero permanentemente -bastará con que los fusibles co--- rrespondientes al estado verdadero y complementado de una variable sean dejados intactos, para que el producto tenga siempre un nivel bajo-; en caso de que todos los fusibles de una columna sean quemados, el producto tendrá siempre un nivel alto, que es el caso que presenta P_M.

1.4 MATRIZ "OR"

Este arreglo, está formado por un número de compuertas "OR" igual al número de salidas del componente, que como ya se había mencionado, en la mayoría de los casos es de ocho; cada compuerta a su vez, tiene un número de entradas igual al número de productos permitidos por el PLA, el cual puede ser de 48 ó 96, y solo se conectarán aquellas cuyos productos esten asociados con la función de salida.

Es aquí donde se realiza la suma Booleana, por lo que sólo bastará que una de las entradas sea verdadera, para que su salida lo sea también.

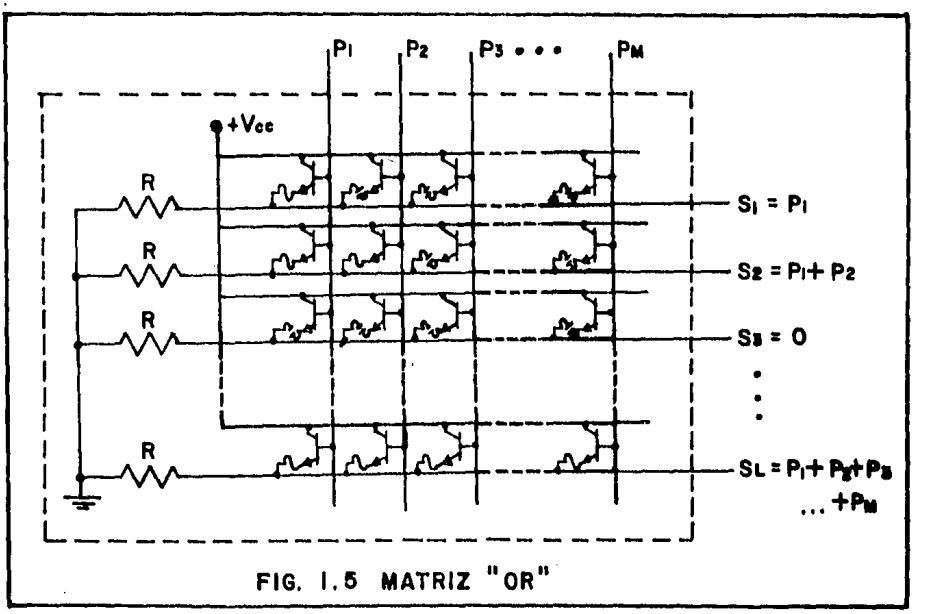


FIG. 1.5 MATRIZ "OR"

En un arreglo lógico programable por el usuario, la implementación física de esta estructura (Fig. - 1.5) se realiza mediante transistores en configuración emisor-seguidor; las bases de los transistores están comandadas por los productos obtenidos de la matriz "AND"; así, si nosotros queremos obtener los sumandos $S_1 = P_1$, $S_2 = P_1 + P_2$, $S_3 = 0$, $S_L = P_1 + P_2 + P_3 + \ldots + P_M$: Para S_1 el fusible correspondiente al emisor del transistor cuya base está comandada - por P_1, deberá estar intacto, de tal manera que - - cuando P_1 tenga un nivel alto, el transistor conduzca fijándose un nivel alto en la resistencia conectada en el emisor y todos los demás deberán de ser quemados; para S_2, los fusibles correspondientes al emisor de los transistores cuyas bases están comandadas por P_1 y P_2, deberán conservarse intactos y todos los demás deberán de ser quemados; si todos - los fusibles correspondientes al renglón de la salida S_3 son quemados, tendremos un nivel bajo permanente; y si todos los fusibles del renglón correspondiente a S_L se dejan intactos, se podría pensar que tendríamos un nivel alto permanente, pero no es así, pues en realidad, esta salida corresponde a la suma de todos los productos y tomará un nivel alto, si al menos uno de ellos tiene un nivel alto.

1.5 ARREGLO DE SALIDA

Cada una de las salidas son programadas indivi--

dualmente como verificadas altas o como verificadas bajas. Esta característica podría parecer redundante, pero por ejemplo, si tuviéramos una función cuyo estado dependiera de cinco productos, y el complemento de esta misma función dependiera solo de tres, el PLA podría ser programado utilizando los productos de la función negada y la salida verificada baja, obteniéndose así la función deseada y logrando con esto un ahorro de memoria sin necesidad de componentes extras. Por otro lado, habrá ocasiones en que la etapa siguiente al PLA utilice lógica positiva o lógica negativa, lo cual puede ser solucionado fácilmente; simplemente se programan cada una de las salidas como verificadas altas o verificadas bajas; o una combinación de las dos, en los casos en que la etapa siguiente utilice lógica mixta.

En el caso de un arreglo lógico programable por el usuario, la implementación física de esta estructura, se realiza mediante una compuerta "OR-EX" de dos entradas (Fig. 1.6), en la que una de ellas se encuentra conectada al sumando S_1, y la otra se encuentra conectada a tierra a través de un fusible, de tal manera que si nosotros deseamos obtener a la salida una función verificada alta, es menester dejar el fusible intacto, y en el caso de que se desee obtener a la salida la función verificada baja, se deberá quemar el fusible de la compuerta corres-

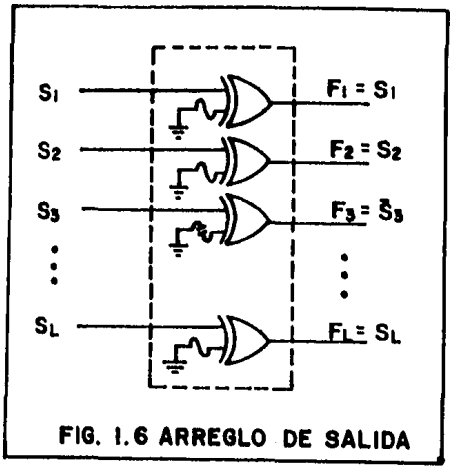


FIG. 1.6 ARREGLO DE SALIDA

pondiente; así tenemos el caso de la Fig. 1.6, en - el que se requiere el estado verdadero de las fun-- ciones F_1, F_2, F_L, casos en los que se encuentran - los fusibles intactos y para la función F_3, donde - se requiere una función verificada baja, el fusible se encuentra quemado.

CAPITULO 2

INFORMACION TECNICA

Un aspecto importante dentro del diseño de circuitos digitales, es el conocimiento de la descripción funcional de cada una de las terminales del componente, de los niveles de voltajes de polarización y operación, velocidad de operación, consumo de potencia, etc., que utilizan los circuitos integrados que formarán parte del diseño, pues como es sabido, en el desarrollo de un circuito, éste ocupa alrededor del 60% del trabajo total. El saber acerca de las características de operación, nos permite ser homogéneos con respecto a los tipos de componentes utilizados, aunque habrá ocasiones en que se tengan que mezclar componentes de familias lógicas diferentes, para lo que ya existen circuitos de acoplamiento.

2.1. CLASIFICACION

Los grupos o clases de Pla's están formados de acuerdo a:

I) Su estructura.
II) Su evolución y desarrollo técnico.

De acuerdo a su estructura, se clasifican según

el número de entradas, salidas y número de productos de la siguiente manera:

$N_E \times N_p \times N_S$ donde:

N_E = número de entradas.

N_p = número de productos.

N_S = número de salidas.

De acuerdo a su evolución y desarrollo técnico, se pueden dividir en dos grupos:

I) PLA's programados por el fabricante.
II) PLA's programados por el usuario.

Ambos grupos pueden ser del tipo: Open-Collector Tri-State, Pull-Up activo y Pull-Up pasivo.

Los PLA's programados por el usuario se sub-dividen a su vez en:

a) PLA's con matriz AND y matriz OR programable totalmente por el usuario.
b) PLA's con matriz AND programable por el usuario y matriz OR pre-establecida por el fabricante.

2.2. CARACTERISTICAS DE OPERACION

A continuación se muestran los arreglos de terminales y las tablas de características de operación y programación; así como las formas de onda de los componentes bajo prueba. Estas están referidas a dispositivos de las marcas Signetics y National.

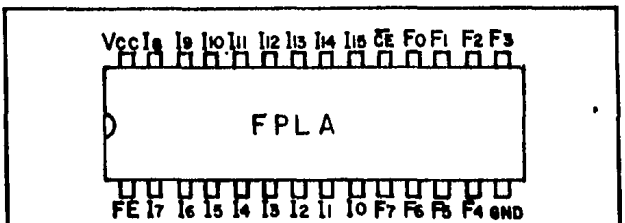


FIG. 2.1 CONFIGURACION DEL 82S100/101.

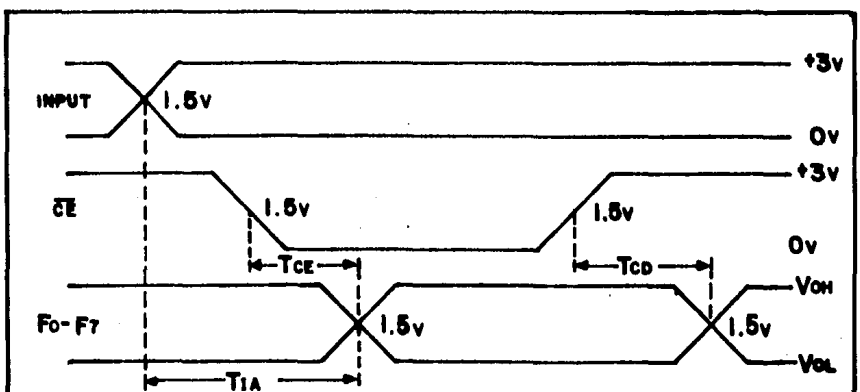


FIG. 2.2 FORMAS DE ONDA DEL 82S100/101 BAJO PRUEBA.

CARACTERISTICAS ELECTRICAS.						
SIMBOLO (UNIDAD)	PARAMETRO	CONDICIONES DE PRUEBA		MIN	TIP	MAX
V_{IH} (V)	HIGH LEVEL INPUT VOLTAGE	V_{CC}=5.25v			2	
V_{IL} (V)	LOW LEVEL INPUT VOLTAGE	V_{CC}=4.75v				0.8
V_{IC} (V)	INPUT CLAMP VOLTAGE	V_{CC}=4.75v, I_{IN}=-18mA			0.8	1.2
V_{OH} (V)	HIGH LEVEL OUTPUT VOLTAGE (82S100).	V_{CC}=4.75v, I_{OH}=-2mA		2.4		
V_{OL} (V)	LOW LEVEL OUTPUT VOLTAGE	V_{CC}=4.75v, I_{OL}=9.6mA			0.35	0.45
I_{OLK} (μA)	OUTPUT LEAKAGE CURRENT (82S101)	V_{CC}=5.25v	V_{OUT}=5.25v		1	40
I_{OO} (μA)	HI-Z STATE OUTPUT CURRENT (82S100)	V_{CC}=5.25v	V_{OUT}=5.25v V_{OUT}=0.45v		1 -1	40 -40
I_{IH} (μA)	HIGH LEVEL INPUT CURRENT	V_{IN}=5.5v			1	25
I_{IL} (μA)	LOW LEVEL INPUT CURRENT	V_{IN}=0.45v			-10	-100
I_{OS} (mA)	SHORT CIRCUIT OUTPUT CURRENT (82S100)	V_{CC}=5.25v, V_{OUT}=0v		-20		-70
I_{CC} (mA)	V_{CC} SUPPLY CURRENT	V_{CC}=5.25v			120	170
C_{IN} (pF)	INPUT CAPACITANCE	V_{CC}=5v	V_{IN}=2v		5	
C_O (pF)	OUTPUT CAPACITANCE	V_{CC}=5v	V_{OUT}=2v		8	
RETRASO DE PROPAGACION						
T_{IA} (nS)	INPUT TO OUTPUT	C_L=30pF			35	50
T_{CD} (nS)	CHIP DISABLE TO OUTPUT	R_1=270ohm			15	30
T_{CE} (nS)	CHIP ENABLE TO OUTPUT	R_2=600ohm			15	30

FIG. 2.3 CARACTERISTICAS DE OPERACION DEL 82S100/101.

SIMBOLO (UNIDAD)	PARAMETRO	MAGNITUD
V_{CC} (V)	POWER SUPPLY VOLTAGE	+7
V_{IN} (V)	INPUT VOLTAGE	+5.5
V_{OH} (V)	HIGH LEVEL OUTPUT VOLTAGE (82S101)	+5.5
V_{O} (V)	OFF STATE OUTPUT VOLTAGE (82S100)	+5.5
T_{A} (°C)	OPERATING TEMPERATURE RANGE	0°a +75°
T_{STG} (°C)	STORAGE TEMPERATURE RANGE	-65°a +150°

FIG. 2.4 LIMITES DE OPERACION DEL 82S100/101.

SIMBOLO (UNIDAD)	PARAMETRO	CONDICIONES DE PRUEBA	MIN	TIP	MAX
V_{CCS} (V)	V_{CC} SUPPLY (PROGRAM "OR")	I_{CCS}=550mA,min (TRANSIENT OR STEADY STATE)	8.5	8.75	9
V_{CCL} (V)	V_{CC} SUPPLY (PROGRAM OUTPUT POLARITY)		0	0.4	0.8
I_{CCS} (mA)	I_{CC} LIMIT (PROGRAM "OR")	V_{CCS}=8.75±0.25v	550		1000
V_{OPH} (V)	OUTPUT VOLTAGE (PROGRAM OUTPUT POLARITY)	I_{OPH}=300±25mA	16	17	18
V_{OPL} (V)	OUTPUT VOLTAGE (IDLE)		0	0.4	0.8
I_{OPH} (mA)	OUTPUT CURRENT LIMIT (PROGRAM OUTPUT POLARITY)	V_{OPH}=17± 1v	275	300	325
T_{R} (µS)	OUTPUT PULSE RISE TIME		10		50
t_{p} (mS)	PROGRAMMING PULSE WIDTH		1		1.5
t_{D} (µS)	PULSE SEQUENCE DELAY				
T_{PR} (mS)	PROGRAMMING TIME			2	
$\frac{T_{PR}}{T_{PR}+T_{PS}}$ %	PROGRAMMING DUTY CYCLE				50
F_{L} (CYCLE)	FUSING ATTEMPTS PER LINK				3
V_{S} (V)	VERIFY THRESHOLD		0.9	1	1.1

FIG. 2.5A VARIABLES DE PROGRAMACION DEL 82S100/101.

SIMBOLO (UNIDAD)	PARAMETRO	CONDICIONES DE PRUEBA	MIN	TIP	MAX
V_{IH} (V)	INPUT VOLTAGE (LOGIC "1")		2.4		5.5
V_{IL} (V)	INPUT VOLTAGE (LOGIC "0")		0	0.4	0.8
I_{IH} (μA)	INPUT CURRENT LOGIC "1")	V_{IH}=5.5v			50
I_{IL} (μA)	INPUT CURRENT (LOGIC "0")	V_{IL}=0v			-500
V_{OHF} (V)	FORCED OUTPUT (LOGIC "1")		2.4		5.5
V_{OLF} (V)	FORCED OUTPUT (LOGIC "0")		0	0.4	0.8
I_{OHF} (μA)	OUTPUT CURRENT (LOGIC "1")	V_{OHF}=5.5v			100
I_{OLF} (mA)	OUTPUT CURRENT (LOGIC "0")	V_{OLF}=0v			-1
V_{IX} (V)	CE PROGRAM ENABLE LEVEL		9.5	10	10.5
I_{IX1} (mA)	INPUT VARIABLES CURRENT	V_{IX}=10v			2.5
I_{IX2} (mA)	CE INPUT CURRENT	V_{IX}=10v			5
V_{FEH} (V)	FE SUPPLY (PROGRAM)	I_{FEH}=300±25mA (TRANSIENT OR STEADY STATE)	16	17	18
V_{FEL} (V)	FE SUPPLY (IDLE)		0	0.4	0.8
I_{FEH} (mA)	FE SUPPLY CURRENT LIMIT	V_{FEH}=17±1v	275	300	325
V_{CCP} (V)	V_{CC} SUPPLY (PROGRAM "AND")	I_{CCP}=550mA,min (TRANSIENT OR STEADY STATE)	4.75	5	5.25
I_{CCP} (mA)	I_{CC} LIMIT (PROGRAM "AND")	V_{CCP}=5±0.25v	550		1000
V_{OPF} (V)	FORCED OUTPUT (PROGRAM)		9.5	10	10.5
I_{OPF} (mA)	OUTPUT CURRENT (PROGRAM)				10

FIG. 2.5B (CONTINUACION) VARIABLES DE PROGRAMACION DEL 82S100/101.

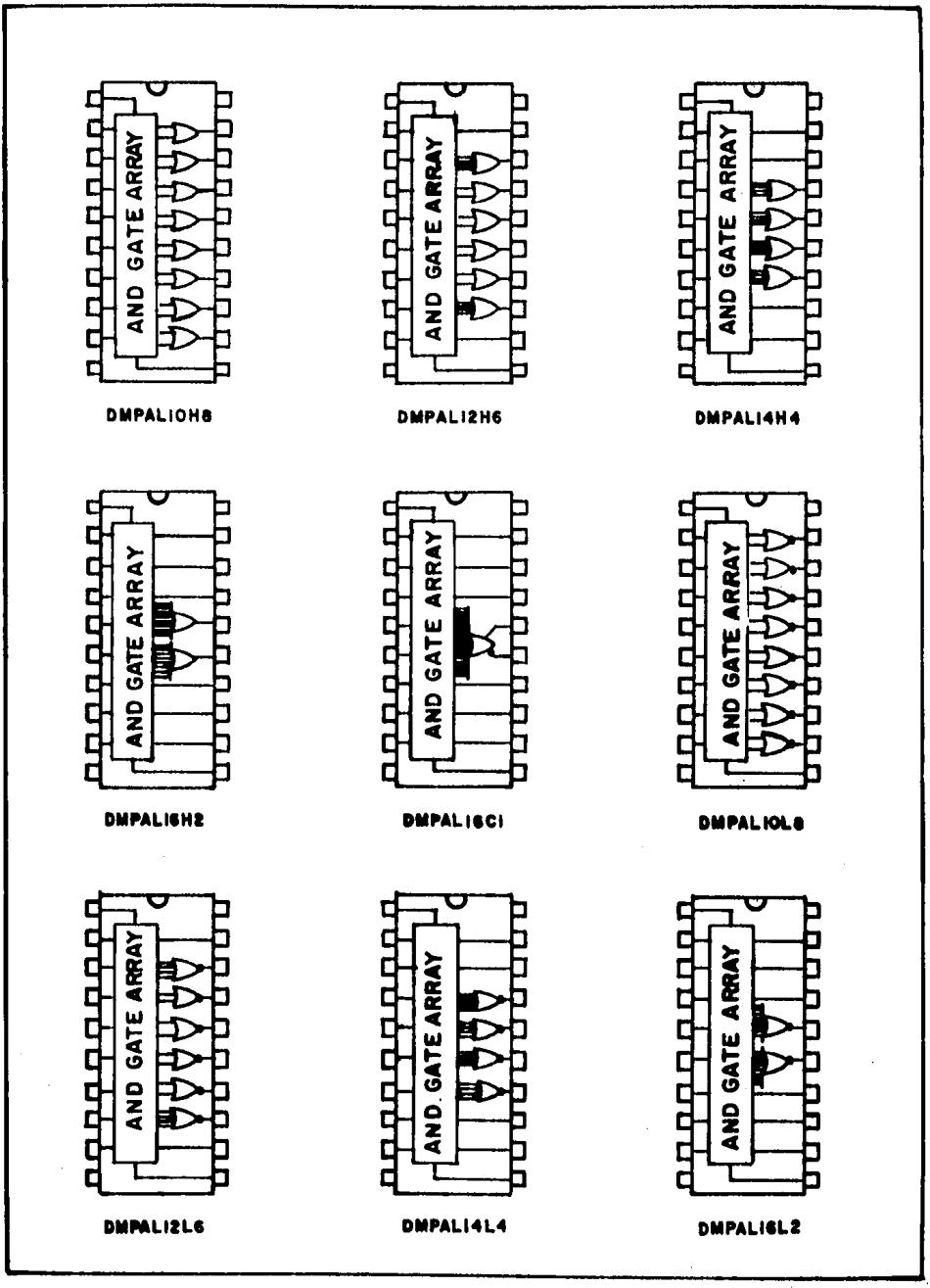
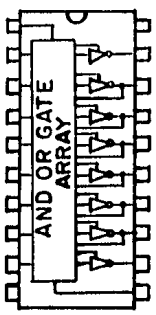
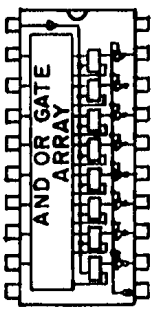


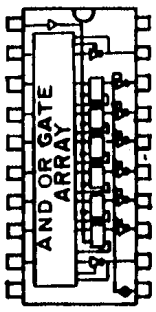
FIG. 2.6A CONFIGURACION DE ALGUNOS PAL'S DE LA NATIONAL.



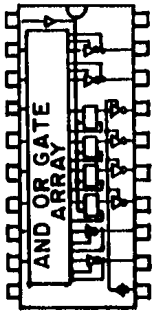
DMPAL16L8



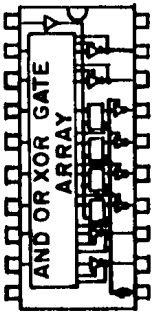
DMPAL16R8



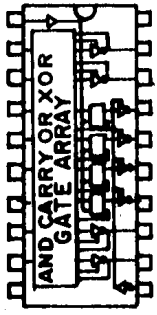
DMPAL16R6



DMPAL16R4



DMPAL16x4



DMPAL16A4

FIG. 2.6B (CONTINUACION) CONFIGURACION DE ALGUNOS PAL'S DE LA NATIONAL

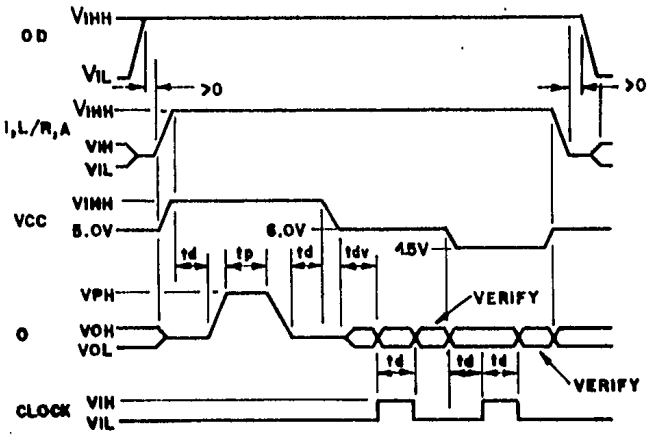


FIG.2.7 TIPO DE ONDA EN LA PROGRAMACION DE PAL's.

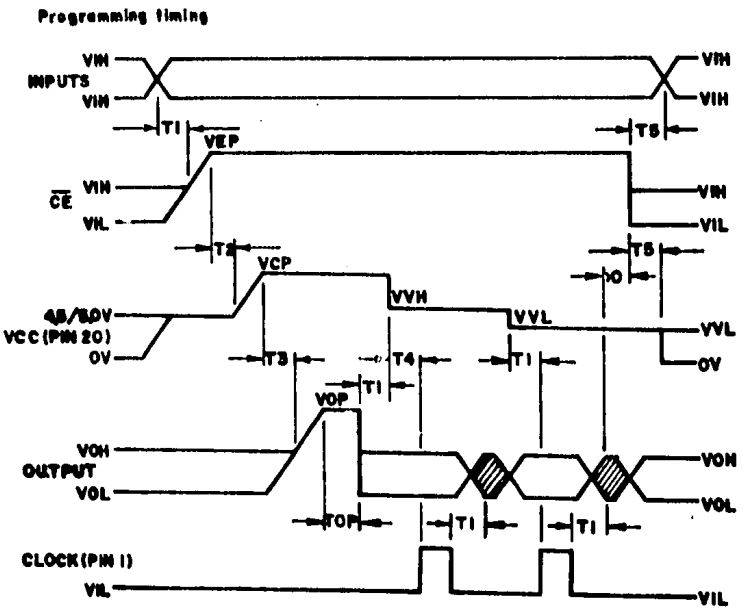


FIG. 2.8 DIAGRAMA DE TIEMPO DE LA PROGRAMACION DE LOS PAL's.

SIMBOLO (UNIDAD)	PARAMETRO	DURANTE LA OPERACION	DURANTE LA PROGRAMACION
V_{CC} (V)	SUPPLY VOLTAGE	7	12
V_I (V)	INPUT VOLTAGE	5.5	12
V_O (V)	OFF-STATE OUTPUT VOLTAGE	5.5	12
T (°C)	STORAGE TEMPERATURE RANGE	-65 a 150	-65 a 150

(A)

SIMBOLO (UNIDAD)	PARAMETRO		MIN	TIP	MAX
V_{CC} (V)	SUPPLY VOLTAGE		4.75	5	5.25
I_{OH} (mA)	HIGH-LEVEL OUTPUT CURRENT				-3.2
I_{OL} (mA)	LOW-LEVEL OUTPUT CURRENT	10H8,12H6,14H4, 16H2,16C1,10L8, 12L6,14L4,16L2, 10H8A,12H6A,14H4A, 16H2A,16C1A,10L8A, 12L6A,14L4A,16L2A.			8
		16L8A.16R8A,16R6A, 16R4A.			24
		16L8,16R8,16R6, 16R4,16X4,16A4, 16L8A1,16R8A1, 16R6A1,16R4A1.			21
T_A (°C)	OPERATING FREE AIR TEMPERATURE		0		75

(B)

FIG. 2.9 (A) LIMITES DE OPERACION Y (B) CONDICIONES DE OPERACION RECOMENDADAS;PARA LOS PAL's DE LA NATIONAL.

SIMBOLO (UNIDAD)	PARAMETRO		CONDICIONES DE PRUEBA	MIN	TIP	MAX
V_{IH} (V)	HIGH LEVEL INPUT VOLTAGE			2		
V_{IL} (V)	LOW LEVEL INPUT VOLTAGE					0.8
V_{IC} (V)	INPUT CLAMP VOLTAGE		V_{CC}=MIN, I_I=-18mA			-1.5
V_{OH} (V)	HIGH LEVEL OUTPUT VOLTAGE		V_{CC}=MIN, V_{IH}=2v V_{IL}=0.8v, I_{OH}=MAX	2.4		
V_{OL} (V)	LOW LEVEL OUTPUT VOLTAGE		V_{CC}=MIN, V_{IH}=2v V_{IL}=0.8v, I_{OL}=MAX			0.5
I_I (mA)	INPUT CURRENT AT MAXIMUN INPUT VOLTAGE		V_{CC}=MAX, V_I=5.5v			1
I_{IH} (µA)	HIGH LEVEL INPUT CURRENT		V_{CC}=MAX, V_I=2.4v			25
I_{IL} (µA)	LOW LEVEL INPUT CURRENT		V_{CC}=MAX, V_I=0.4v			-250
I_{OS} (mA)	SHORT CIRCUIT OUTPUT CURRENT		V_{CC}=5v	-30		-130
I_{OZH} (µA)	OFF STATE OUTPUT CURRENT HIGH LEVEL VOLTAGE APPLIED	16L8,16R8, 16R6,16R4, 16X4,16A4, 16L8A,16R8A, 16R6a,16R4A, 16L8A1,16R8A1, 16R6A1,16R4A1.	V_{CC}=MAX, V_I=2v V_O=2.4v, V_{IH}=0.8v			100
I_{OZL} (µA)	OFF STATE OUTPUT CURRENT LOW LEVEL VOLTAGE APPLIED		V_{CC}=MAX, V_I=2v V_O=0.4v, V_{IH}=0.8v			-100
I_{CC} (mA)	SUPPLY CURRENT	10H8,12H6,14H4, 16H2,16C1,10L8, 12L6,14L4,16L2, 10H8A,12H6A, 14H4A,16H2A, 16C1A,10L8A, 12L6A,14L4A, 16L2A.	V_{CC}=MAX		55	90
		16L8A,16R8A, 16R6A,16R4A.	V_{CC}=MAX			150
		16L8.	V_{CC}=MAX		140	210
		16R4,16R6,16R8.	V_{CC}=MAX		150	225
		16X4,16A4.	V_{CC}=MAX		160	
		16L8A1,16R8A1, 16R4A1,16R6A1,	V_{CC}=MAX		140	180

FIG. 2.10 CARACTERISTICAS ELECTRICAS DE LOS PAL's.

SIMBOLO (UNIDAD)	PARAMETRO		CONDICIONES DE PRUEBA	MIN	TIP	MAX
t_{PD} (nS)	FROM ANY INPUT TO ANY OUTPUT.	*1	R_L=2Kohm C_L=15pF		25	40
		*2			18	25
	INPUT TO OUTPUT.	*3	R_L=667ohm C_L=45pF		25	40
		*4			18	35
		*5			15	25
	CLOCK TO OUTPUT.	*3			15	25
		*4			15	25
		*5			12	17
t_{PZX}(nS)	PIN 11 TO OUTPUT ENABLE.	*3			15	25
		*4			12	
		*5			15	25
	INPUT TO OUPUT ENABLE.	*3			25	40
		*4			18	
		*5			18	35
t_{PXZ}(nS)	PIN 11 TO OUPUT DISABLE.	*3	R_L=667ohm C_L=5pF		15	25
		*4			12	
		*5			15	25
	INPUT TO OUTPUT DISABLE.	*3			25	40
		*4			18	
		*5			18	35
t_W (nS)	WIDTH OF CLOCK.	HIGH *3		25		
		*4		17		
		*5		17		
		LOW *3		25		
		*4		17		
		*5		17		
t_{SU} (nS)	SET UP TIME.	*3		40		
		*4		35		
		*5		25		
t_h (nS)	HOLD TIME.	*3,*4,*5		0	-15	

*1:10H8,12H6,14H4,16H2,16C1,10L8,12L6,14L4,16L2.
*2:10H8A,12H6A,14H4S,16H2A,16C1A,10L8A,12L6A,14L4A,16L2A.
*3:16L8,16R8,16R6,16R4,16X4,16A4.
*4:16L8A,16R8A,16R6A,16R4A.
*5:16L8A-1.16R8A-1,16R6A-1,16R4A-1.

FIG. 2.11 TIEMPOS DE PROPAGACION DE LOS PAL's.

SIMBOLO (UNIDAD)	PARAMETRO		MIN	TIP	MAX
V_{IHH} (V)	PROGRAM-LEVEL INPUT VOLTAGE		11	11.5	12
I_{IHH} (mA)	PROGRAM-LEVEL INPUT CURRENT	OUTPUT PROGRAM PULSE			50
		OD,L/R			25
		ALL OTHER INPUTS			5
I_{CCH} (mA)	PROGRAM SUPPLY CURRENT				400
T_P (µS)	PROGRAM PULSE WIDTH		10		50
t_d (nS)	DELAY TIME		100		
t_{dc} (%)	PROGRAM PULSE DUTY CYCLE				25
V_P (V)	PROGRAM /VERIFY-PROTECT-INPUT VOLTAGE			20	
I_P (mA)	PROGRAM/VERIFY-PROTECT-INPUT CURRENT				400
t_{dv} (µS)	DELAY TIME TO VERIFY		100		
V_{PH} (V)	PROGRAMMING PULSE		11	11.5	12
V_{EP} (V)	CE (ENABLE) PROGRAMMING VOLTAGE		9.5	10	10.5
V_{CP} (V)	ELEVATED V_{CC} (WHILE PROGRAMMING)		9.5	10	10.5
V_{OP} (V)	OUTPUT PIN PROGRAMMING VOLTAGE		9.5	10	10.5
T_{OP} (µS)	OUTPUT PIN PROGRAMMING PULSE		9	10	11
V_{VH} (V)	HIGH VERIFY (V_{CC}) VOLTAGE			6	6
V_{VL} (V)	LOW VERIFY (V_{CC}) VOLTAGE		4.5	4.5	
T_r (V/µS)	SLEW RATE; V_{EP}, V_{CP}, AND V_{OP}		0.4	5	10
T_1 (nS)	INPUT, V_{VH}, OUTPUT SET-UP TIME		100		
T_2 (nS)	ENABLE SET-UP TO V_{CP} TIME		500		
T_3 (nS)	V_{CP} SET-UP TO PROGRAMMING PULSE		200		
T_4 (nS)	V_{VH} SET-UP TO CLOCK		10		
T_5 (nS)	INPUT HOLD TIME (FROM V_{EP})		50		
CLOCK (nS)	PAL REGISTER CLOCK		25		

FIG. 2.12 PARAMETROS DE PROGRAMACION DE LOS PAL's.

2.3 EXPANSION DE LA CAPACIDAD DE LOS PLA's

Habrá ocasiones, en que para implementar una función, un solo PLA no sea suficiente, dado que el número de productos, entradas y salidas de éste es fijo, por lo tanto, deberán realizarse combinaciones de PLA's que nos den uno equivalente de mayor capacidad y existen basicamente tres casos que se nos pueden presentar: la necesidad de incrementar el número de productos, incrementar el número de entradas ó incrementar el número de salidas.

2.3.1 EXPANSION DE PRODUCTOS.

La expansión de productos, se obtiene de la combinación de PLA's con sus entradas y salidas conectadas en paralelo (Fig. 2.13) en componentes del tipo Open-Collector (y pull-Up pasivo), conexión que no puede ser realizada con los del tipo Pull-Up activo, ya que se presentarían corrientes excesivas que dañarían al componente cuando se presentara el caso en el que la salida de uno sea alta y la del otro baja (enseguida se explicará la manera en que pueden ser conectados éstos); en los componentes del tipo Tri-State la expansión de los productos puede ser realizada pero con dificultad, a causa de los conflictos lógicos resultantes de las salidas, puesto que comparten el mismo bus, por lo tanto, la expansión debe incluir la entrada habili-

tadora del componente.

En la Fig. 2.13, se muestra un arreglo realizado con componentes del tipo Pull-Up pasivo, utilizando el 5775A de Monolithic Memories, el cual cuenta con una resistencia Pull-Up pasiva interna en cada una de sus salidas, lo que hace innecesario el uso de una externa, a no ser que se deseen mayores velocidades de propagación. Las resistencias internas reducen el rango de I_{OL} de 12 a 9.6 mA y además limitan esta configuración a un máximo de 5 PLA's (o 480 productos), tomando cada una de las 8 salidas un valor de I_{OL}=2.4 mA a 0.45 V. Si se requieren mas de 5 PLA's, las salidas deberán conectarse a la

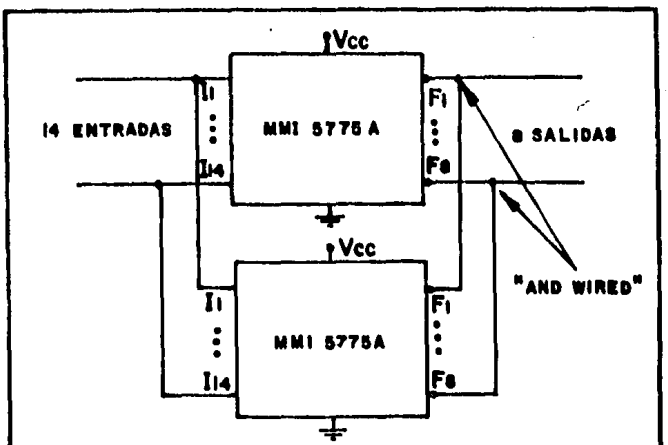


FIG. 2.13 EXPANSION DE PRODUCTOS CON ALAMBRADO AND.

entrada de compuertas AND, como se muestra en la Fig. 2.14. En este caso, los PLA's del tipo Pull-Up activo pueden utilizarse, con la ventaja de que proporcionan una mayor velocidad de propagación que los del tipo pull-up pasivo; debe hacerse notar que el arreglo de salida está programado en lógica verificada baja, por lo tanto, los PLA's no direccionados tendrán salidas altas y no afectarán a las 8 compuertas. Así, si nosotros tenemos dos PLA's de 4x2x1 y deseamos implementar la función $F = AC + \bar{B}\bar{C}D + \bar{A}B\bar{D} + BC$, primero será necesario encontrar la función complementada y presentarla como una suma mínima de productos, misma que al aplicarle el teorema de Morgan, quedará como un producto de dos sumandos complementados y los productos contenidos en cada sumando son los que se programarán en cada uno de los PLA'S y el arreglo de salida se deberá programar como inversor, (ver Fig. 2.15) Para así obtener finalmente F.

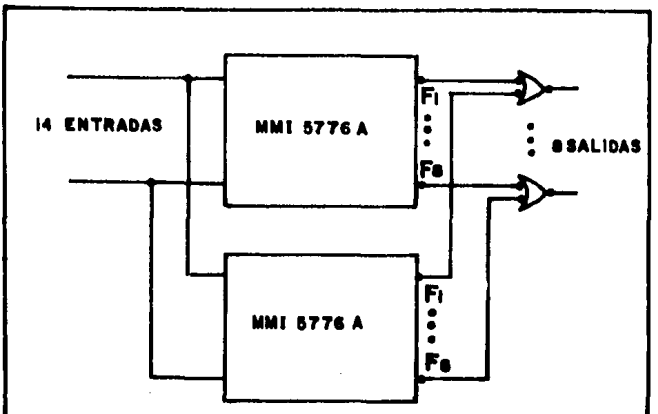
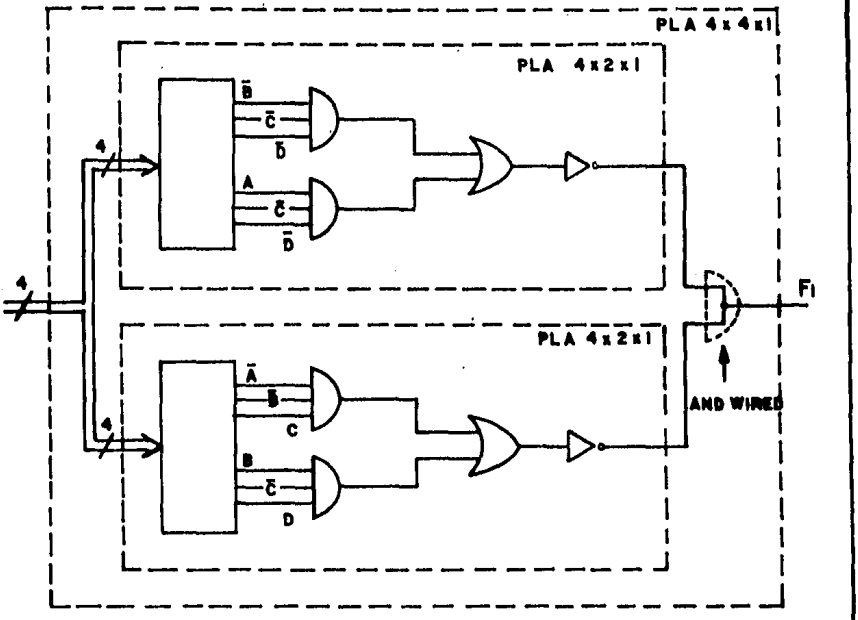


FIG. 2.14 EXPANSION DE PRODUCTOS CON COMPUERTA "AND" EQUIVALENTE.



$F = AC + \bar{B}C\bar{D} + \bar{A}B\bar{D} + BC$ ∴
$F = \bar{B}C\bar{D} + AC\bar{D} + \bar{A}\bar{B}C + BC\bar{D} \Rightarrow F = \overline{(\bar{B}C\bar{D} + AC\bar{D})}\,\overline{(\bar{A}\bar{B}C + BC\bar{D})}$

FIG. 2.15 PLA DE 4 x 4 x 1 EQUIVALENTE.

En circuitos con FPLA's del tipo Tri-State, donde se requieran mas de 48 productos, la realización es obtenida con cierta dificultad, pues para hacerlo, es necesario repartir la tabla programa en dos - o mas sub-tablas, cada una conteniendo menos de 48 productos que deberán ser acomodados en FPLA's separados. Esta repartición es ejecutada segmentando la tabla de programa original sobre los "1"s y "0"s

P_N	ENTRADAS				SALIDAS							
	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0	X	X	X	1	0	0	0	0	0	0	0	1
1	X	X	1	0	0	0	0	0	0	1	0	0
2	X	1	0	1	0	0	0	0	1	0	0	0
3	X	0	1	1	0	0	0	0	1	0	0	0
4	X	1	0	0	0	0	0	1	0	0	0	0
5	0	1	X	1	0	0	0	1	0	0	0	0
6	1	0	X	1	0	0	0	1	0	0	0	0
7	1	0	1	X	0	0	1	0	0	0	0	0
8	1	1	X	1	0	0	1	0	0	0	0	0
9	0	1	1	X	0	0	1	0	0	0	0	0
10	1	0	X	X	0	1	0	0	0	0	0	0
11	1	X	1	X	0	1	0	0	0	0	0	0
12	1	1	X	X	1	0	0	0	0	0	0	0

FIG. 2.16 TABLA DE VERDAD DE UN SISTEMA DE 4 ENTRADAS Y 8 SALIDAS.

P-terms		ENTRADAS				SALIDAS							
P_n	P_n	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0a	0	x	0	x	1	0	0	0	0	0	0	0	1
1a	1	x	0	1	0	0	0	0	0	0	1	0	0
3	2	x	0	1	1	0	0	0	0	1	0	0	0
6	3	1	0	x	1	0	0	0	1	0	0	0	0
7	4	1	0	1	x	0	0	1	0	0	0	0	0
10	5	1	0	x	x	0	1	0	0	0	0	0	0
11a	6	1	0	1	x	0	1	0	0	0	0	0	0

P-terms		ENTRADAS				SALIDAS							
P_n	P_n	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0b	0	x	1	x	1	0	0	0	0	0	0	0	1
1b	1	x	1	1	0	0	0	0	0	0	1	0	0
2	2	x	1	0	1	0	0	0	0	1	0	0	0
4	3	x	1	0	0	0	0	0	1	0	0	0	0
5	4	0	1	x	1	0	0	0	1	0	0	0	0
8	5	1	1	x	1	0	0	1	0	0	0	0	0
9	6	0	1	1	x	0	0	1	0	0	0	0	0
11b	7	1	1	1	x	0	1	0	0	0	0	0	0
12	8	1	1	x	x	1	0	0	0	0	0	0	0

FIG. 2.17 TABLA DE VERDAD SEGMENTADA, DEL SISTEMA DE 4 ENTRADAS Y 8 SALIDAS.

de las variables de entrada seleccionadas y aquellos productos Pn que contengan Don't Care en la variable seleccionada, darán origen a dos productos -

(P_{na} y P_{nb}), por lo que en general, el número final de productos usados es mayor, debido a esta expansión. Las variables seleccionadas se deberán quitar del grupo de dirección de entrada al FPLA, para usarse en la entrada habilitadora de los mismos, haciendo la decodificación apropiada. Así, por ejemplo, si contamos con FPLA's Tri-State 4x10x8 y se desea programar el contenido de la tabla de la Fig. 2.16, es necesario realizar una segmentación de ésta sobre la variable I_2 (por ser la que tiene menor número de Don't Care), con lo que se obtendrán dos sub-tablas (Fig. 2.17), donde se debe notar que el número de productos en cada una es menor que 10 y corresponden a FPLA's separados, los cuales son operados en paralelo y controlados por I_2 a través de sus entradas habilitadoras como se muestra en la Fig. 2.18.

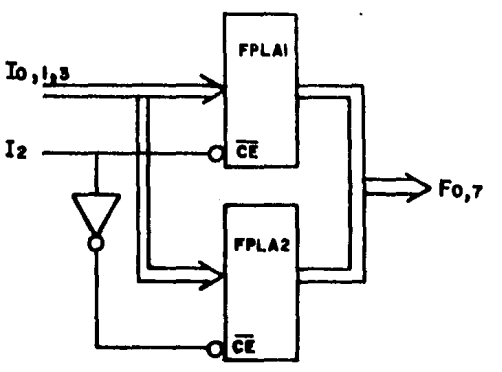


FIG.2.18 EXPANSION CON FPLAS POR SEGMENTACION DE UNA VARIABLE DE ENTRADA.

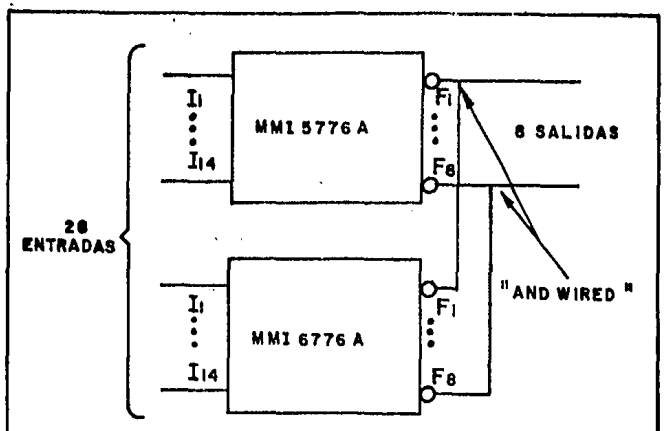


FIG. 2.19 EXPANSION A 28 VARIABLES DE ENTRADA CON DOS PLA DE 14 ENTRADAS

El uso de este método, depende del contenido de la tabla de programa original, ya que habrá ocasiones en que no será posible emplearlo.

2.3.2 EXPANSION DE VARIABLES DE ENTRADA.

Aunque las salidas y los productos pueden expanderse tanto como se deseen, la expansión de las entradas es mucho mas incómoda. En la Fig. 2.19 se muestra la interconexión de PLA's utilizando el - - 5776A de Monolithic Memories para la expansión a 28 entradas, donde cualquiera de las 8 salidas puede - contener hasta las 28 variables de entrada en su totalidad; sin embargo, una entrada existente en un -

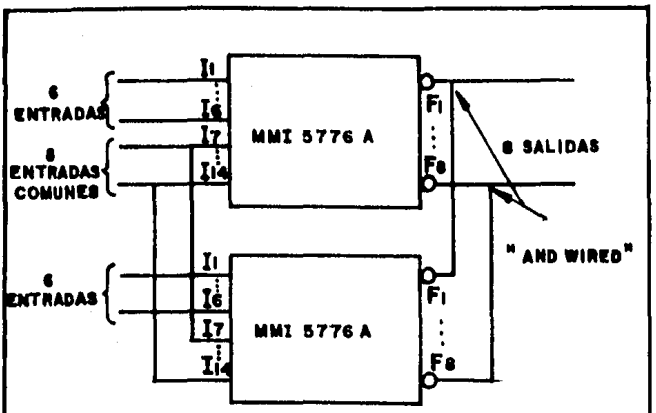


FIG. 2.20 EXPANSION A 20 VARIABLES DE ENTRADA CON DOS PLA's DE 14 ENTRADA.

PLA, no puede ser usada en los productos del otro. En otras palabras, las ecuaciones de salida están - restringidas a una simple función AND de dos expresiones, cuyas variables no pueden ser combinadas.

La Fig. 2.20 muestra una expansión práctica de - 20 entradas, de las cuales 8 son comunes a ambos -- componentes. Para hacer uso de este tipo de expansión, es necesario seleccionar las variables de entrada que formarán parte del grupo común y poste--- riormente determinar como repartir las otras 12 variables de manera que ningún producto direccionado

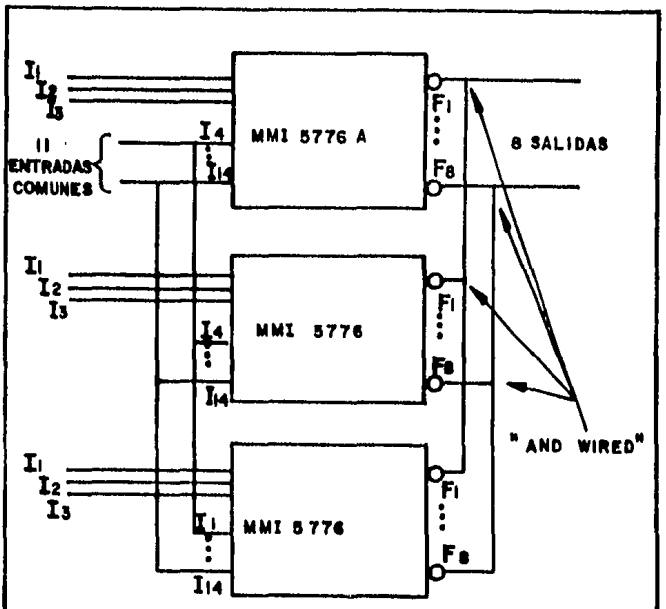


FIG. 2.21 EXPANSION A "20 VARIABLES DE ENTRADA" CON TRES PLAS DE 14 ENTRADAS.

requiera entradas de ambos grupos. Si 8 entradas comunes no son suficientes, con un tercer PLA se pueden incrementar éstas a 11 (Fig. 2.21). En este caso, el número total de entradas sigue siendo de 20, y el problema de variables comunes insuficientes, podría quedar resuelto. La flexibilidad de programación puede también ser mejorada si se separan las entradas en 7 grupos, en lugar de 4 y cada una de los 3 grupos adicionales sean comunes solo a 2 PLA's.

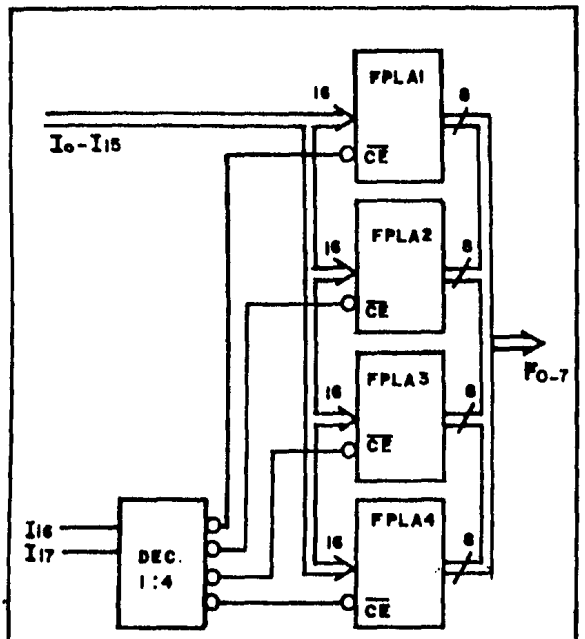


FIG. 2.22 FPLA EQUIVALENTE DE 18 VARIABLES DE ENTRADA.

Con las configuraciones anteriores, podrán ser - solucionados muchos de los problemas donde sea necesario utilizar mas entradas de las que se pueden encontrar en un PLA, pero no es una solución general, ya que si se requiere un PLA equivalente de mas entradas de las que se disponen con uno solo -hablando estrictamente- cada producto deberá tener la posibilidad de ser formado hasta con el número total de las variables de entrada; así, cuando se dice --

que un PLA es de 30 entradas, es porque se pueden - realizar productos de hasta 30 variables diferentes. Por ejemplo, en la Fig. 2.22, se muestra un arreglo con el 82S100 de Signetics, donde efectivamente se obtiene un FPLA equivalente de 18 variables de en-- trada y puede ser realizado con componentes del - - tipo Open-Collector o Tri-State. El número total de FPLA's necesarios para la expansión, es igual a 2^n, donde (n) es el número de variables que se de-- sean incrementar; en componentes que no cuentan con entrada habilitadora, el número de FPLA's será - - igual a 2^{n+1} ya que será necesario utilizar una de sus entradas de dirección como selectora del mismo dentro de la programación. Con mas de 20 entradas, esta solución podrá llegar a ser muy costosa y será necesario encontrar una solución particular utili zando los métodos del primer caso.

2.3.3. EXPANSION DE SALIDAS.

En la Fig. 2.23 se combinan dos PLA's de 14 en-- tradas, 8 salidas y 96 productos -cada uno- para -- proporcionar la expansión de las salidas; ésta ex-- pansión produce el equivalente de un PLA de 14 en-- tradas, 96 productos y 16 salidas en lugar de 8. - A simple vista, se podría pensar que el PLA equiva- lente es de 192 productos, dado que en una aplica-- ción particular, los 96 productos del PLA(a), po--- drían ser diferentes de los 96 productos del PLA(b),

obteniéndose así un total de 192 productos diferentes; pero no se debe perder de vista, que la capacidad de los productos en un PLA está definida por el número de combinaciones posibles que pueden ser - - agrupadas en cada una de las salidas. En la Fig.2. 23, se ve que en cada una de las salidas del PLA(a) pueden ser coleccionados hasta los 96 productos de éste, pero ninguno de los pertenecientes al PLA(b)- pueden ser agrupados en ellas; y de la misma manera, en las salidas del PLA(b), no pueden ser agrupados productos del PLA(a), por lo tanto, este arreglo es en realidad de 96 y no de 192 productos.

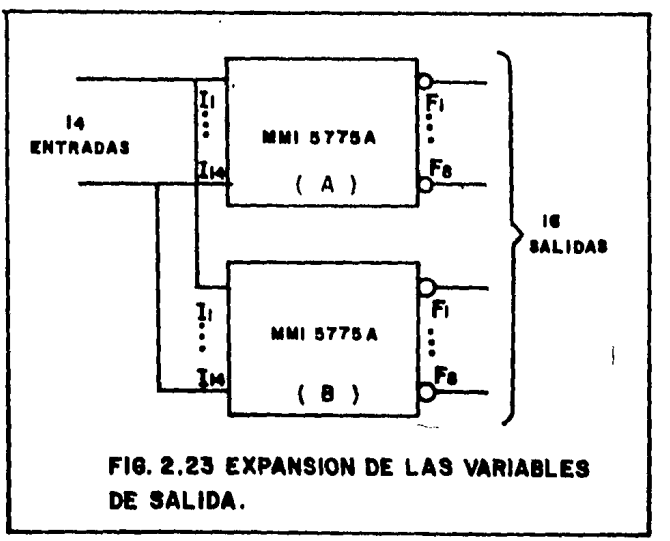


FIG. 2.23 EXPANSION DE LAS VARIABLES DE SALIDA.

En algunos casos, podrá ser mas económico realizar una implementación codificando la tabla de salida almacenandola en un solo componente y entonces descifrar los estados de salida deseados a través de un PROM de 32x8, o un decodificador 1:N según se requiera. Ambos métodos se muestran en las Figs. 2.24 y 2.25.

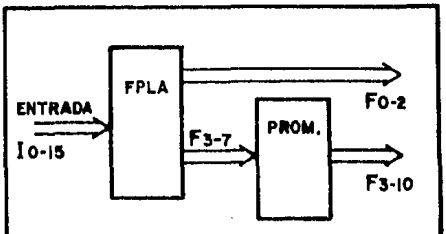


FIG. 2.24 EXPANSION DE SALIDAS POR MEDIO DE UN PROM.

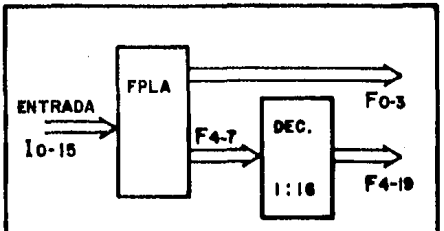


FIG. 2.25 EXPANSION DE SALIDAS POR MEDIO DE UN DEC.

CAPITULO 3

PROCEDIMIENTO DE PROGRAMACION

Una vez que se ha concluído la etapa de diseño, habiéndose obtenido las funciones de salida deseadas, -si ha de utilizarse un arreglo lógico programable- será necesario decidir entre un arreglo lógico programable por el fabricante (PLA), un arreglo lógico con matriz AND programable por el usuario --(PAL), o un arreglo lógico con matriz AND, matriz - OR y polaridad de salida programable por el usuario (FPLA).

Si se elige el PLA, deberá proporcionarse al fabricante una tabla con el contenido del programa, - la cual incluye las variables y los productos co--- rrespondientes a cada una de las funciones de salida con sus polaridades deseadas, el nombre del solicitante, número de orden de compra, número de componente y cantidad requerida, como se muestra en la - Tabla de la fig. 3.1.

Si se elige un PAL, se tendrá que determinar de entre los existentes, cual es el que mejor se adapta a nuestro diseño, ya que como se mostró antes, - existen PAL's de 8, 10, 12, 14 y 16 entradas, de 1, 2, 4, 6 y 8 salidas ciertas y/o complementadas, con o sin registro de salidas y/o realimentación. Cada

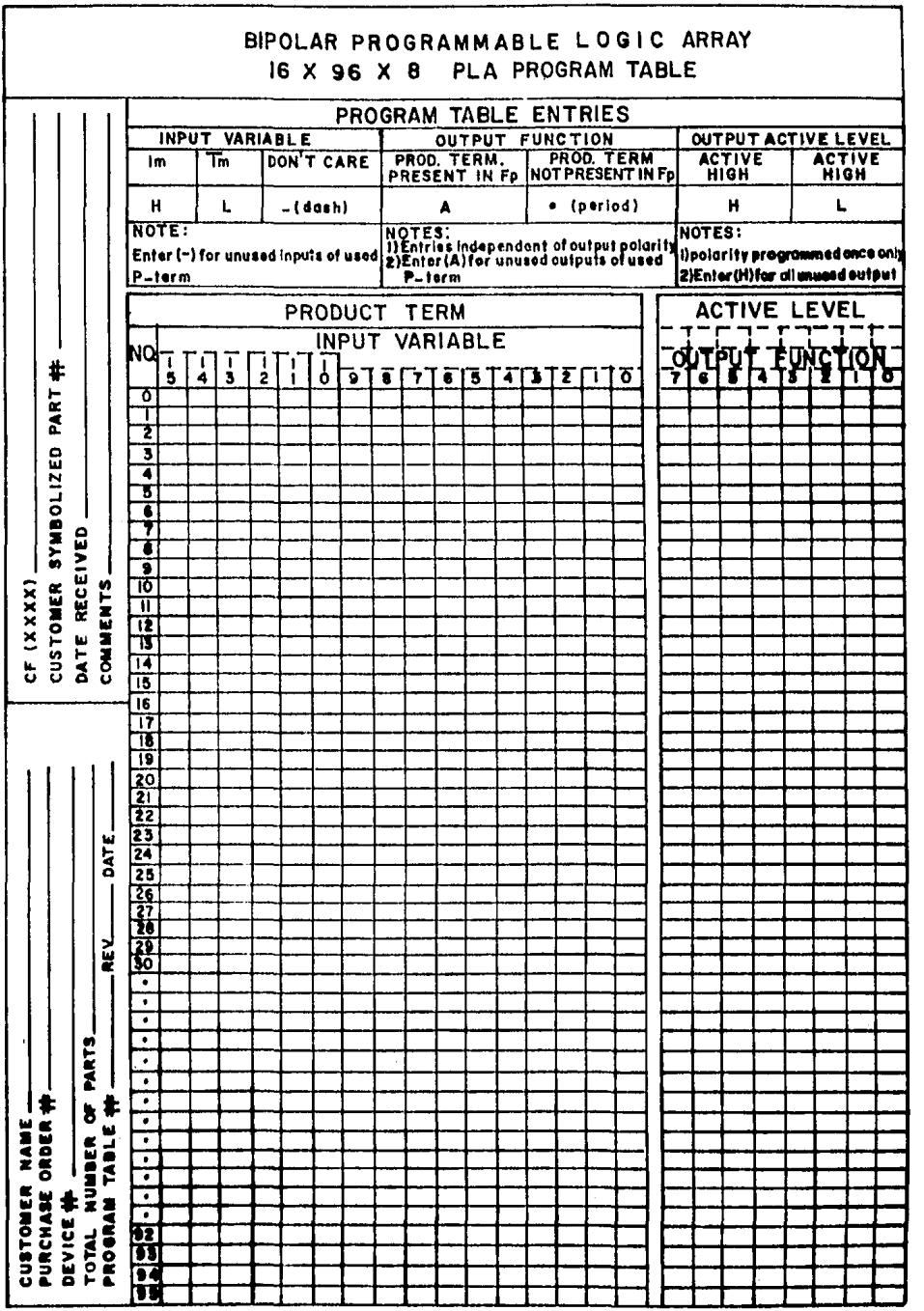


FIG. 3.1 FORMATO DE LA TABLA PROGRAMA DEL PLA.

uno de los PAL's, tiene su tabla de programa par--- cialmente especificada y deberá completarse de mane ra que se realicen las funciones deseadas; la infor mación de esta tabla, es la que se utiliza para gra bar en forma automática los PAL's, utilizando pro-- gramadores de PROM's con un adaptador. En la Fig. 3.2A se muestra la tabla que le corresponde al PAL 12H6 y en la fig. 3.2B el diagrama lógico del mismo en su estado virgen.

En el caso de que se elija un FPLA, se deberán - especificar los productos existentes, cuales estan contenidos en cada una de las salidas y las polaridades de éstas, en una tabla similar a la de los -- PLA's (Fig. 3.3). No existen adaptadores para gra bar los FPLA's utilizando programadores de PROM's, por lo que cuentan con uno especial, que además sir ve para verificar su contenido una vez programados. Si no se dispone de este programador, proporcionándole al distribuidor la tabla programa, es capaz de programarnos el FPLA.

3.1 GENERACION DE LA TABLA DE PROGRAMA.

En los PLA's y FPLA's, el término "tabla de programa", es usado en lugar de "tabla de verdad", ya que la primera permite Don't Care (X) como entradas directas, lo que la hace mas general y va de acuerdo con sus estructuras. Idealmente, la tabla de

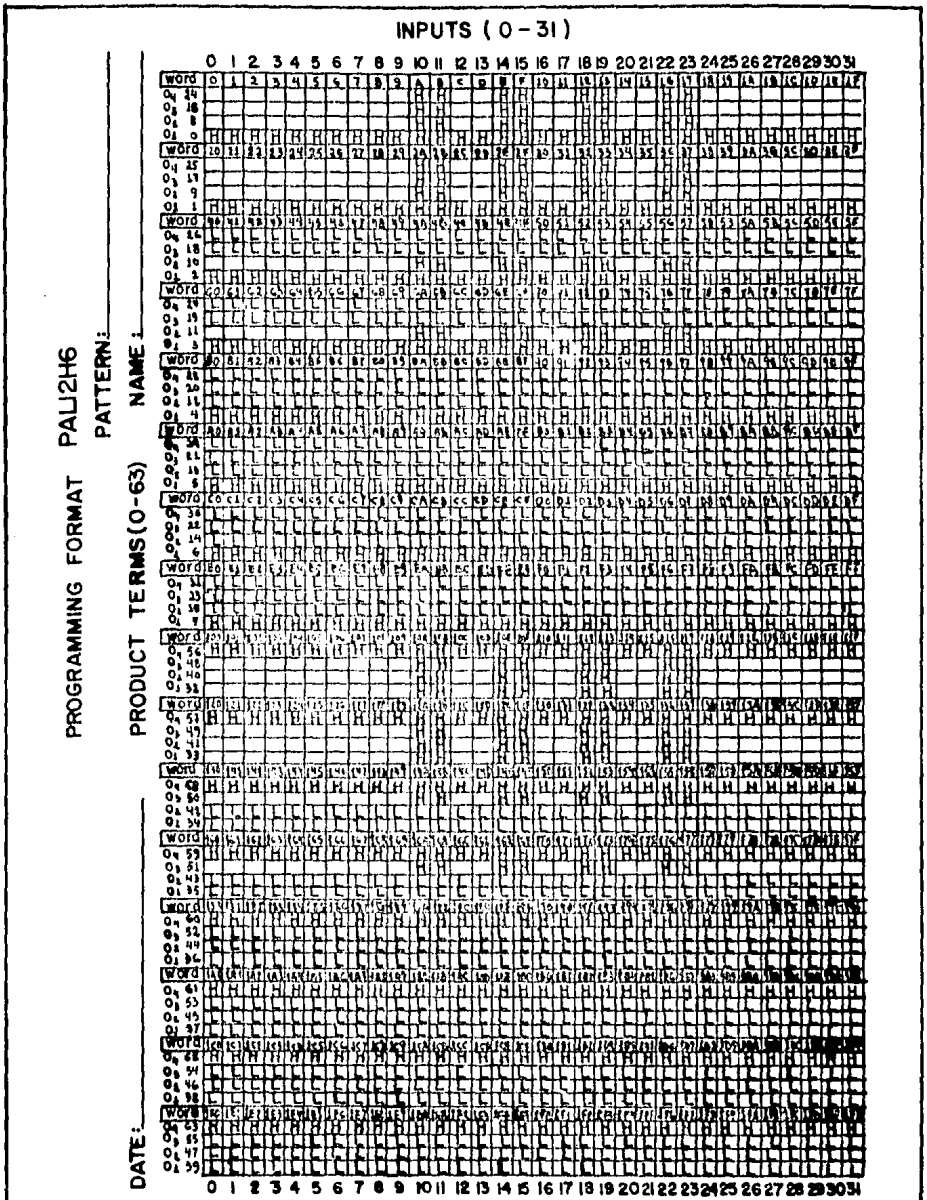
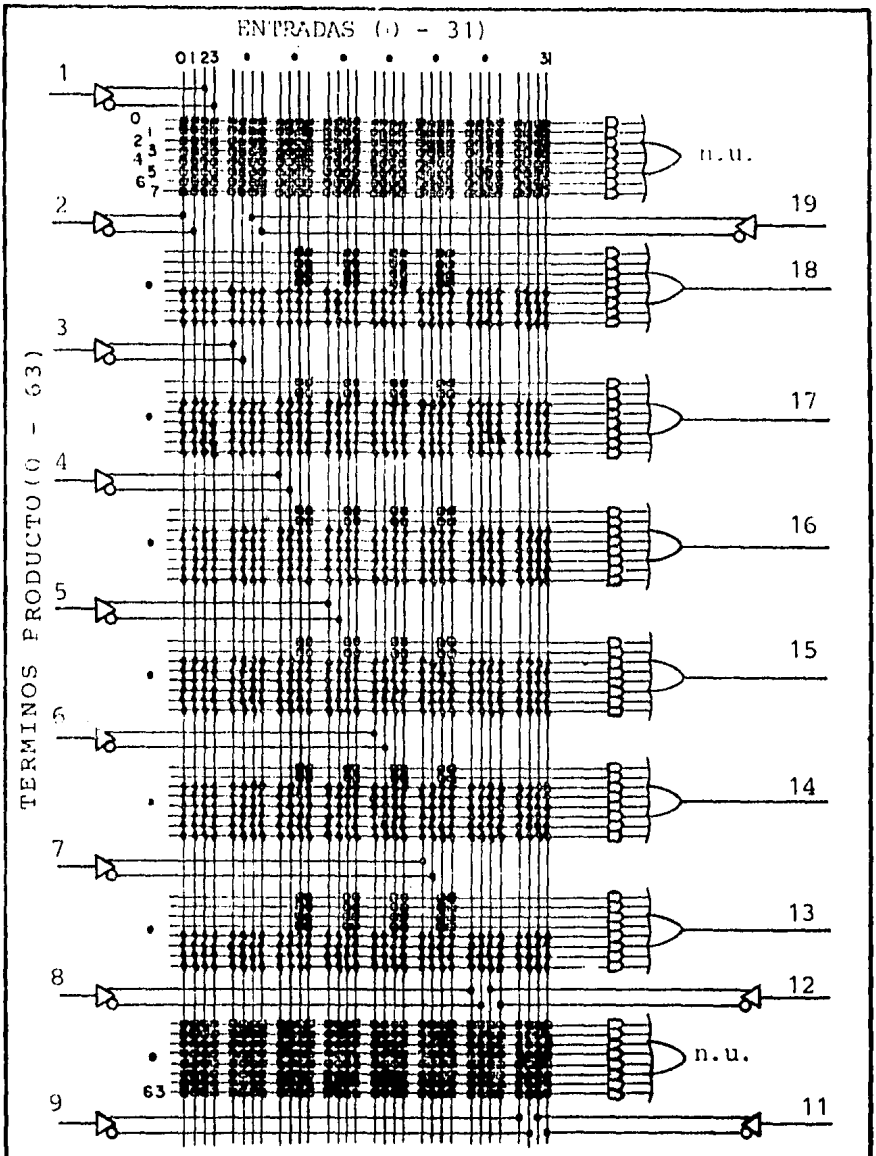


FIG. 3.2A TABLA PROGRAMA DEL PAL12H6.



SIMBOLOGIA:
- • FUSIBLES NO EXISTENTES (L,N,0).
- □ FUSIBLES NO EXISTENTES (H,P,1).
- • CONEXION DEL FABRICANTE.
- x CONEXION DEL USUARIO.
- n.u. NO SE USA.

FIG. 3.2B DIAGRAMA LOGICO DEL PAL12H6.

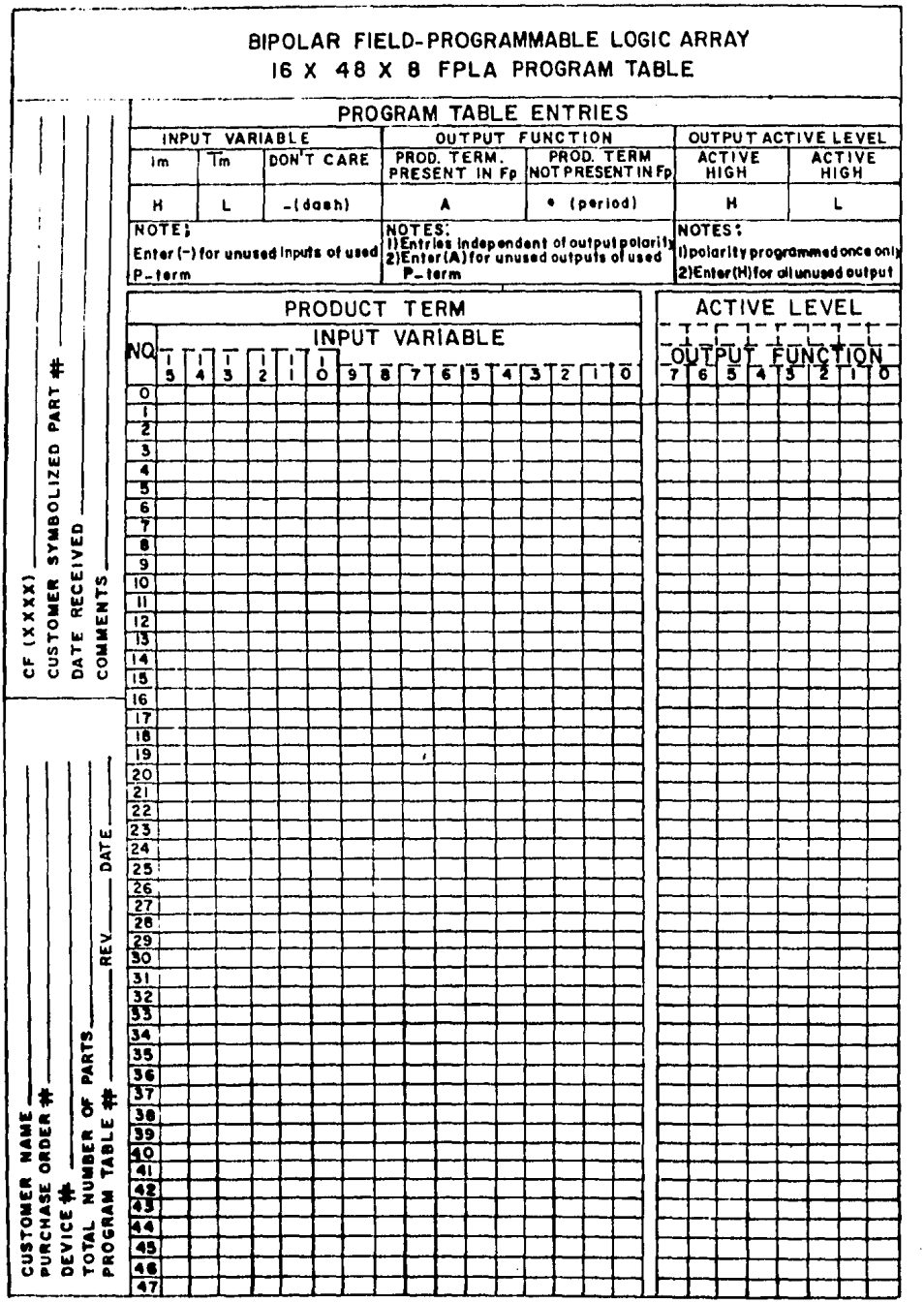


FIG. 3.3 FORMATO DE LA TABLA PROGRAMA DE UN FPLA.

programa, deberá contener entradas construídas con un código que no solamente evite ambigüedades en la programación, sino que también proporcione un sistema claro, el cual muestre fácilmente el estado lógico actual de las salidas; la mayoría de los diseñadores lógicos, usan uno de los dos símbolos, ya sea (1/0) o (H/L). La (X) es generalmente usada para estados de entrada "Don't Care". Para obtener sistemas de programación claros, el estado de todas las variables de entrada en cada producto, se debe codificar como se muestra en la tabla de la Fig. 3.4 -- donde todas las variables de entrada indican claramente el estado lógico que activa un producto dado. Se requiere un símbolo adicional para codificar el estado en el que ambos listones E_m y $\bar{E}_m$ estan intactos; para facilidad del indicador, se utiliza el cero (0), ya que ocurre solamente en componentes -- virgen, o en variables de productos programados -

$P_n \stackrel{?}{=} f(I_m)$	Entrada	Asignación en la tabla programa	Estado del fusible.
Si	I_m	H	Fusible $\bar{I}_m$ quemado
	$\bar{I}_m$	L	Fusible I_m quemado.
No	Indeterminado.	-	Fusibles $\bar{I}_m$ y I_m quemados.

Fig. 3.4 Simbología usada para las entradas de la tabla Programa.

parcialmente o sin uso; éste es el estado inicial - de todas las variables de entrada, y significa su - estado lógico nulo. Si algún producto contiene -- por lo menos una variable en el estado nulo, el producto nunca será seleccionado por ninguna combina-- ción lógica de entrada. La entrada de un cero en - la tabla de programa, no tiene sentido y no se permite, por lo tanto, éste se utiliza en sistemas de programación para indicar resultados verificados en blanco o condiciones de falla en el programa.

Aunque estos símbolos son apropiados para codificar los diferentes estados de las variables de en-- trada para cada producto, así como para la polari-- dad del nivel activo de salida, cuando son utilizados para codificar éstas, dan lugar a algunas ambigüedades, y esto es debido a la elección que se haga del nivel activo. Para codificar las salidas, se tienen diferentes alternativas; en cualquier caso, la selección de cada entrada, incluye la inspección del mapa de actividades para determinar si -- una función de salida contiene un producto particular; sin tomar en cuenta la polaridad de salida elegida, un producto activa la función F_p, si está contenido en ella, por lo tanto, cualquier F_p será -- puesta alta y $\bar{F}_p$ será puesta baja; lo que significa, que si P_n no está contenido en una salida, todas -- las funciones F_p y $\bar{F}_p$ permanecerán en su estado lógico de invalidéz (bajo y alto respectivamente). --

Un método particularmente conveniente para la codificación de la tabla de salida, se muestra en la tabla de la Fig. 3.5

$F_p \stackrel{?}{=} f(P_n)$	En el mapa de actividades.	En la tabla de salidas.	Estado del fusible.
Si	P_n	A A	Fusible de P_n intacto en la matriz "oR".
No	/	• •	Fusible de P_n quemado en la matriz "OR".
Nivel activo		H L	
		F_p $\bar{F}_p$	Polaridad de la funcion.

Fig. 3.5 SIMBOLOGIA USADA POR LAS SALIDAS DE LA TABLA PROGRAMA.

Este sistema de codificación, utiliza una A (activa), para indicar la presencia de P_n en cualquiera de las dos funciones (F_p o $\bar{F}_p$), y un " ." (punto), - para indicar ausencia, y tiene la ventaja de que la salida puede ser construída directamente del mapa - de actividades. También, cuando es recuperada la tabla de salida almacenada de un componente programado, la presencia o ausencia de un producto en una función F_n de salida, es fácilmente detectado, ha-- ciendo el procedimiento de verificación del arreglo mas fácil; sin embargo, para relacionar las salidas

lógicas actuales con las entradas anteriores (especialmente cuando se relacionan con la conversión de códigos o translación de direcciones), es necesario referirse a la tabla de la Fig. 3.6.

En la tabla de salidas.	Salida lógica del FPLA.		
A	H	L	
	L	H	
	H	L	Nivel activo
Polaridad de la función.	F_p	$\bar{F}_p$	

FIG. 3.6 SIMBOLOGIA USADA PARA INDICAR EL NIVEL -- ACTIVO DE LAS SALIDAS EN LA TABLA PROGRAMA.

En los PAL's la tabla de programa no resulta -- tan clara, ni muestra fácilmente el estado lógico - actual de las salidas y se debe a que su construc-- ción está encaminada principalmente a facilitar la transferencia de su contenido a un programador de - PROM's con un adaptador y no a uno especialmente -- para ellos, como en el caso de los PLA's y FPLA's. Otro aspecto de esta tabla, es que para poderla llenar, primero se debe seleccionar en el diagrama lógico -particular del componente elegido- las entra-

das y salidas que van a ser utilizadas (ver Fig. -- 3.2B), así como las conexiones que deben marcarse - en él, para formar las funciones de salida. En el diagrama lógico, hay algunas líneas horizontales y verticales marcadas, en las que no se pueden hacer conexiones, ya que físicamente no existen fusibles (en ese componente en particular). Las conexiones que se hagan sobre una misma línea horizontal po--- niendo una "X", son equivalentes al producto de todas las variables de entrada ahí conectadas; las -- conexiones realizadas en diferentes líneas horizontales -correspondientes a una misma salida- son - - equivalentes a la suma Booleana de los productos de cada línea. Si alguna de las líneas horizontales -contenida en una de las salidas deseadas- no es -- utilizada, se deberán marcar conexiones en todos -- sus puntos de cruce (variables verdaderas y complementadas) con las líneas verticales, para que no -- influya en la función de salida y simbólicamente se puede hacer, poniendo una "X" en el interior de la compuerta "AND" que esté al final de dicha línea. - La numeración de las líneas horizontales y verticales, corresponde con la de los renglones y columnas respectivamente, de la tabla de programa. Una vez que el diagrama lógico está especificado, se puede proceder a llenar la tabla poniendo una "L" en los cuadros correspondientes a las intersecciones del diagrama lógico que tenga una "X"; si alguna de las líneas horizontales no fué utilizada, entonces el -

renglón correspondiente deberá llenarse con "L's" (excepto aquellos cuadros que ya estén ocupados); - una vez hecho ésto, a todos los cuadros que queden vacíos, se les pone una "H". Un aspecto importante de la construcción de ésta tabla, es que una vez llena, queda formada por 512 ($1FF_{HEX}$) palabras de 4 bit's, con la dirección de cada palabra y el peso de cada bit marcados.

El papel que juega la tabla de programa, durante el proceso de grabado de los arreglos lógicos programables, es muy importante, por lo que no debe -- prescindirse de ella.

Con el objeto de ilustrar la manera en que deben ser llenadas las tablas de programa, en la Fig.3.7A se muestra una tabla de verdad, a partir de la cual se obtubieron las ecuaciones de salida mostrada en la Fig. 3.7B y es en base a éstas, que se obtiene - el mapa de actividades, simplemente asignando un número a cada uno de los productos diferentes dentro de las ecuaciónes y posteriormente, poner las salidas en función de los números asignados, como se -- muestra en la Fig. 3.8 donde se debe notar, que se puso una diagonal "/" en la posición de un producto que no esta contenido en la salida dada. Es a partir de esta información con la que son llenadas las tablas de programa del PLA (Fig.3.9) y del FPLA --- (Fig.3.10) y la parte que no se ocupa, se deja en - blanco, para poder permitir cambios futuros en com-

ponentes grabados, si fuera necesario; para llenar la tabla de programa del PAL (Fig.3.12) primero se marcó sobre el diagrama lógico (Fig.3.11) las co-- nexiónes correspondientes a las ecuaciónes de salida.

ENTRADAS				SALIDAS						
I_3	I_2	I_1	I_0	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	1	0	0	1
0	1	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	1	0	0	1
0	1	1	0	0	1	0	0	1	0	0
0	1	1	1	0	1	1	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	1
1	0	1	0	1	1	0	0	1	0	0
1	0	1	1	1	1	1	1	0	0	1
1	1	0	0	0	0	1	0	0	0	0
1	1	0	1	0	1	0	1	0	0	1
1	1	1	0	1	0	0	0	1	0	0
1	1	1	1	1	1	0	0	0	0	1

(A)

$F_0=XXXI_0$

$F_1=0$

$F_2=XXI_1\bar{I}_0$

$F_3=XI_2\bar{I}_1I_0+X\bar{I}_2I_1I_0$

$F_4=XI_2\bar{I}_1\bar{I}_0+\bar{I}_3I_2XI_0+I_3\bar{I}_2XI_0$

$F_5=I_3\bar{I}_2I_1X+I_3I_2XI_0+\bar{I}_3I_2I_1X$

$F_6=I_3\bar{I}_2XX+I_3XI_1X$

(B)

Fig. 3.7 (A) Tabla de verdad y (B) Ecuaciónes de salida para un sistema dado.

$P_0 = XXXI_0$ $P_4 = XI_2\bar{I}_1\bar{I}_0$ $P_8 = I_3I_2XI_0$

$P_1 = XXI_1\bar{I}_0$ $P_5 = \bar{I}_3I_2XI_0$ $P_9 = \bar{I}_3I_2I_1X$

$P_2 = XI_2\bar{I}_1I_0$ $P_6 = I_3\bar{I}_2XI_0$ $P_{10} = I_3\bar{I}_2XX$

$P_3 = X\bar{I}_2I_1I_0$ $P_7 = I_3\bar{I}_2I_1X$ $P_{11} = I_3XI_1X$

$F_0 = P_0 + / + / + / + / + / + / + / + / + / + / + /$

$F_1 = / + / + / + / + / + / + / + / + / + / + /$

$F_2 = / + P_1 + / + / + / + / + / + / + / + / + /$

$F_3 = / + / + P_2 + P_3 + / + / + / + / + / + / + /$

$F_4 = / + / + / + / + P_4 + P_5 + P_6 + / + / + / + / + /$

$F_5 = / + / + / + / + / + / + P_7 + P_8 + P_9 + / + /$

$F_6 = / + / + / + / + / + / + / + / + / + P_{10} + P_{11}$

FIG. 3.8 MAPA DE ACTIVIDADES CORRESPONDIENTE AL SISTEMA DE 4 ENTRADAS Y 7 SALIDAS.

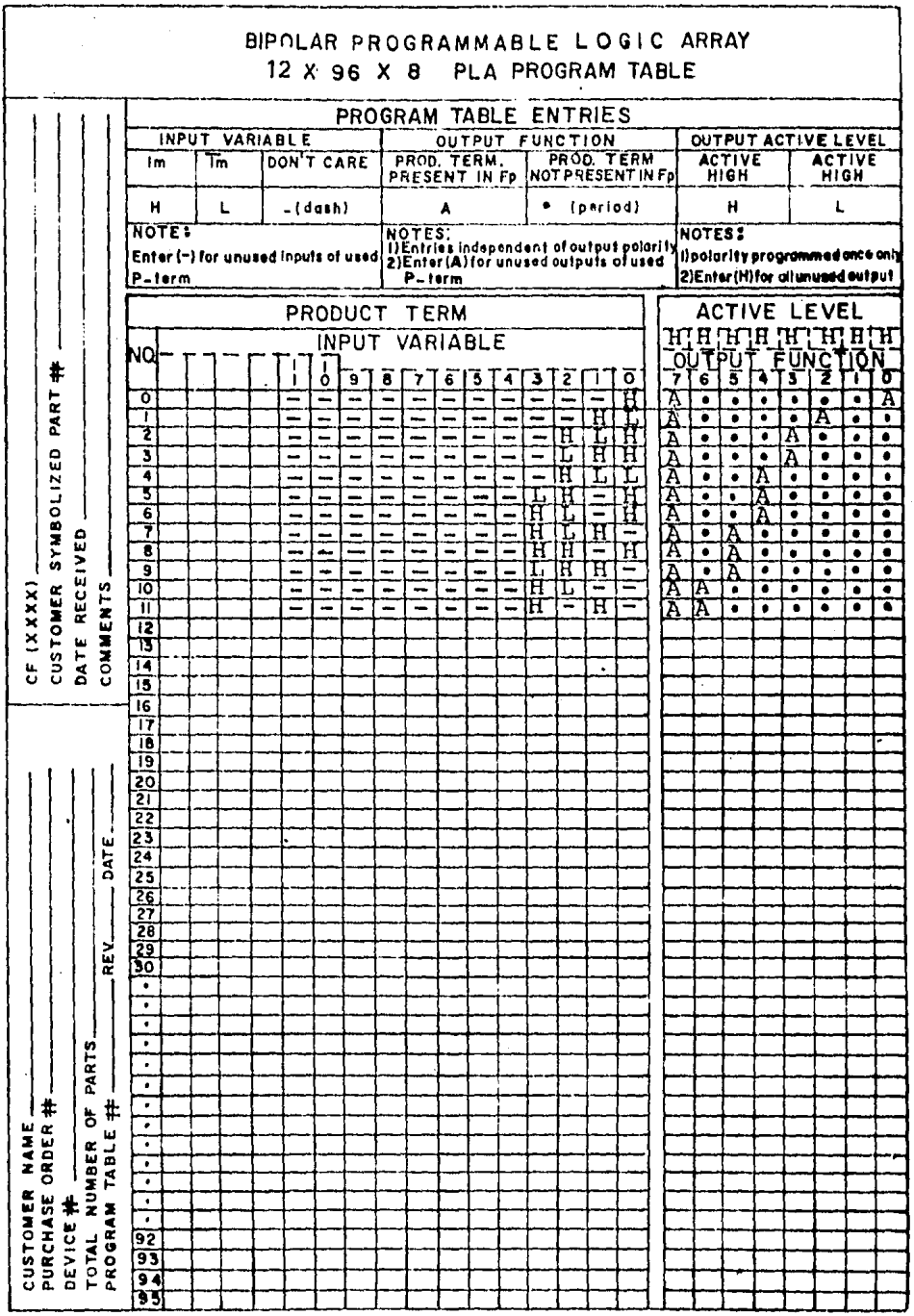


FIG.3.9 TABLA DE PROGRAMA PARA UN PLA.

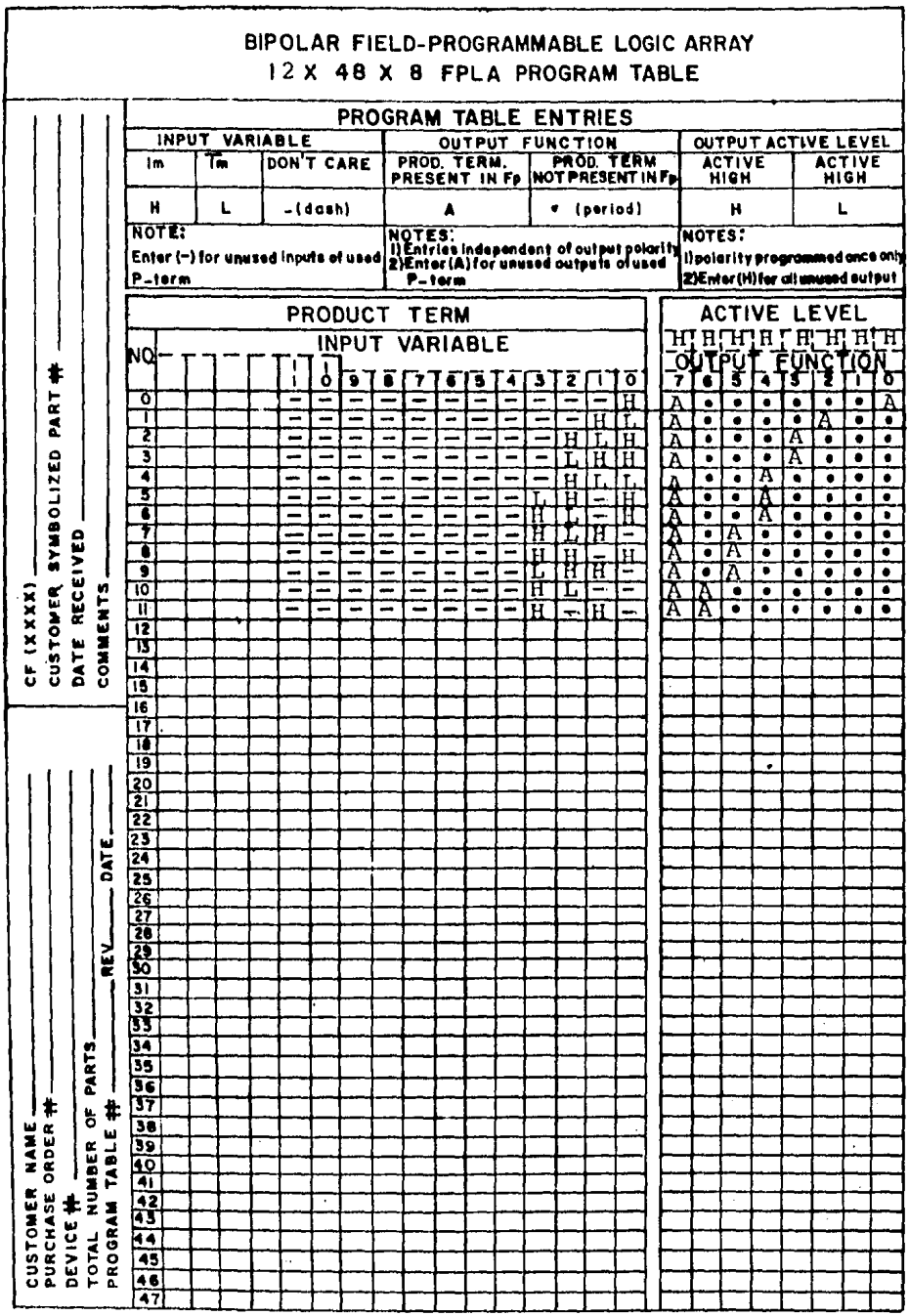
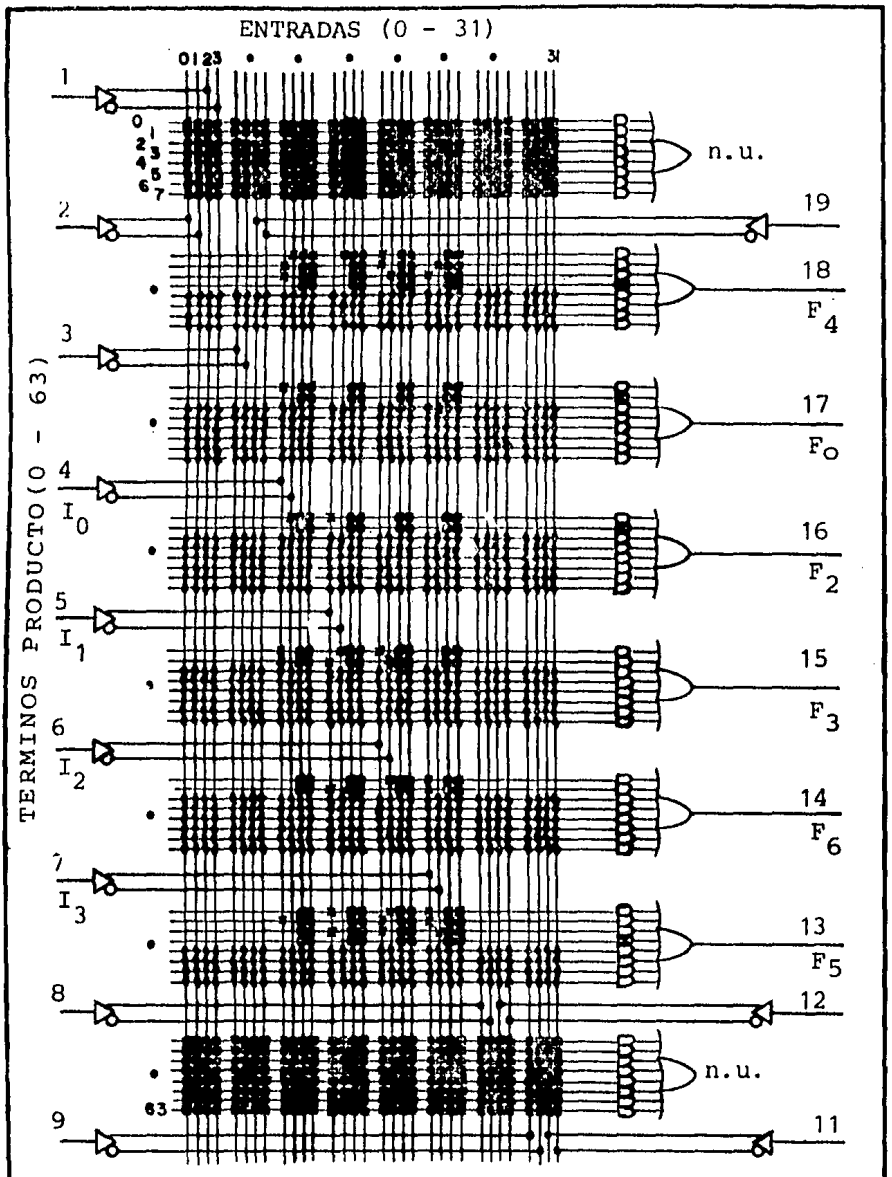


FIG.3.10 TABLA DE PROGRAMA PARA UN FPLA.



SIMBOLOGIA:

♦	FUSIBLES NO EXISTENTES (L,N,0).
◻	FUSIBLES NO EXISTENTES (H,P,1).
•	CONEXION DEL FABRICANTE.
x	CONEXION DEL USUARIO.
n.u.	NO SE USA.

FIG. 3.11 DIAGRAMA LOGICO DEL PAL12H6.

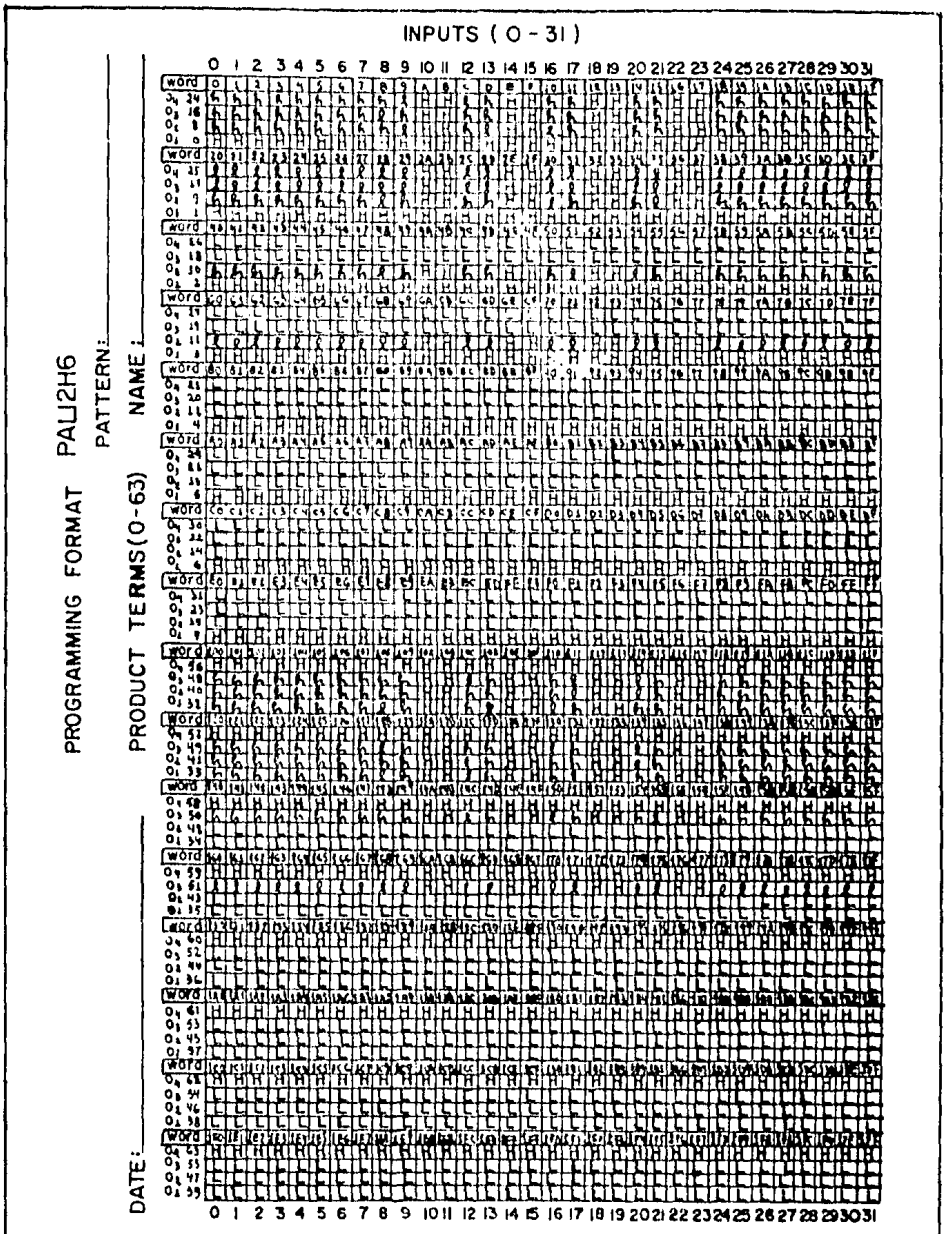


FIG. 3.12 TABLA DE PROGRAMA PARA UN PAL12H6.

3.2. PROGRAMACION DE LA POLARIDAD DE SALIDA.

La polaridad de las salidas de los PLA's, es programada por el fabricante, de acuerdo con las especificaciones de la tabla de programa que se le proporciona. En los PAL's, la polaridad de salida ya está fija, por lo que solo se elige el de la salida adecuada al diseño. La programación de la polaridad de las salidas de los FPLA's, es realizada por el usuario y el procedimiento indicado por la Signetics para el caso del 82S100/101 es el siguiente: Si alguna de las salidas se desea verificada baja, primero se ponen las terminales 1 (FE) y 14 (GND) a tierra (0 volts), después, la terminal 28 (V_{cc}) a V_{CCL}, la 29 ($\overline{CE}$) y las terminales 9, 8, 7, 6, 5, 4, 3, 2, 27, 26, 25, 24, 23, 22, 21 y 20 (entradas I_0 a I_{15}) a V_{IH}, por último, se aplica un voltaje V_{OPH} durante un tiempo t_p en la salida F_L (F_o, F_1, ..., F_6 ó F_7) cuyo fusible vaya a ser quemado. La salida sin uso no necesitan ser programadas, lo mismo - que las salidas que se deseen verificadas altas, ya que el componente en su estado virgen, las tiene de esta manera. En la Fig. 3.13, se muestra la se--- cuencia de programación.

3.3 PROGRAMACION DE LA MATRIZ "AND"

La programación de las variables de entrada ó -- programación de la matriz "AND", en los PLA's es -

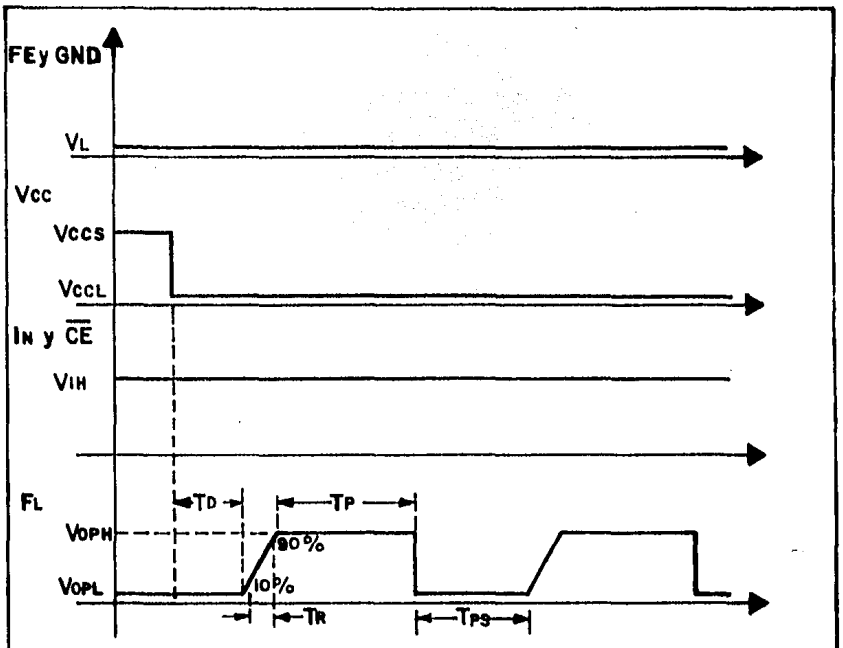


FIG. 3.13 SECUENCIA DE PROGRAMACION DE LA POLARIDAD DE SALIDA DEL PLA

efectuada por el fabricante a partir de la tabla de programa que se le proporciona. En los PAL's, esta programación es efectuada por el usuario, auxiliándose de un programador de PROM's con el adaptador apropiado (Ver Tabla de la Fig. 3.14) y el procedimiento es el mismo que se utiliza para grabar PROM's, se introduce al programador la información que cada dirección debe contener y en forma automática se graba el PAL.

FABRICANTE (MARCA)	ADAPTADOR PARA PAL STANDART DE LA NATIONAL S.C.
NATIONAL SEMI STARPLEX	DM 9068
CYBERNETIC PROG. SYSTEMS	CYMPC -1
PRO LOG	DM 9068
STRUCTURED DESIGN	SD-20/24

FIG. 3.14 ADAPTADORES DE PROGRAMADORES DE PROM's PARA PROGRAMACION DE PAL'S.

En los FPLA's, la programación es más detallada y en el 82S100/101 de la Signetics es la siguiente. Se programan una por una, las variables que deben - quedar contenidas en cada uno de los productos, poniendo primero la terminal 14 (GND) a tierra (0 -- volts), la 28 (V_{cc}) a V_{CCP}, la 19 ($\overline{CE}$) a V_{IH}, las - terminales 9, 8, 7, 6, 5, 4, 3, 2, 27, 26, 25, 24, 23, 22, 21 y 20 (entradas I_o a I_{15}) a V_{IX}, después se selecciona el producto P_M que será programado -- (del 0 al 47) alimentando un código binario en las salidas F_o a F_5 (siendo F_o el bit menos significativo), utilizando los niveles de voltaje V_{OHF} y V_{OLF} según se requiera. Si el producto P_m contiene a una variable verdadera I_N, entonces se quema el fusible de la variable complementada $\bar{I}_N$, bajando el - voltaje de la entrada I_N desde V_{IX} hasta V_{IH}, se --

deja pasar un tiempo T_D y se aumenta el voltaje de FE desde V_{FEL} a V_{FEH}, después de otro tiempo T_D se aumenta el voltaje de $\overline{CE}$ desde V_{IH} a V_{IX} durante un período T_p y se vuelve a dejar en V_{IH}; se deja pasar un tiempo T_D y se regresa FE a V_{FEL} y la entrada I_N a V_{IX}. Si la variable que contiene el producto P_M está complementada ($\bar{I}_N$), se sigue el procedimiento anterior, pero ahora bajando el voltaje de la entrada I_N desde V_{IX} hasta V_{IL}. Si una variable de entrada I_N no esta contenida en el producto P_M en su estado verdadero ni complementado (Don't - Care), entonces se deberán quemar los fusibles de - I_N y $\bar{I}_N$. Los fusibles de las variables de entrada, de los productos no usados, no necesitan quemarse. La Fig. 3.15 muestra la secuencia de programación.

3.4 PROGRAMACION DE LA MATRIZ "OR"

La programación de los productos que deben estar contenido en cada una de las salidas o programación de la matriz "OR" en los PLA's, es efectuada por el fabricante a partir de la tabla de programa que se le proporciona. En los PAL's, ya se encuentra grabada y lo único que se hace es seleccionar el más - adecuado al diseño. En los FPLA's, debe ser realizada por el usuario y el procedimiento en el 82S100 /101 de la Signetics es el siguiente: Se programa - uno de los términos contenidos en cada salida a la vez, poniendo primero la terminal 14 (GND) a tierra

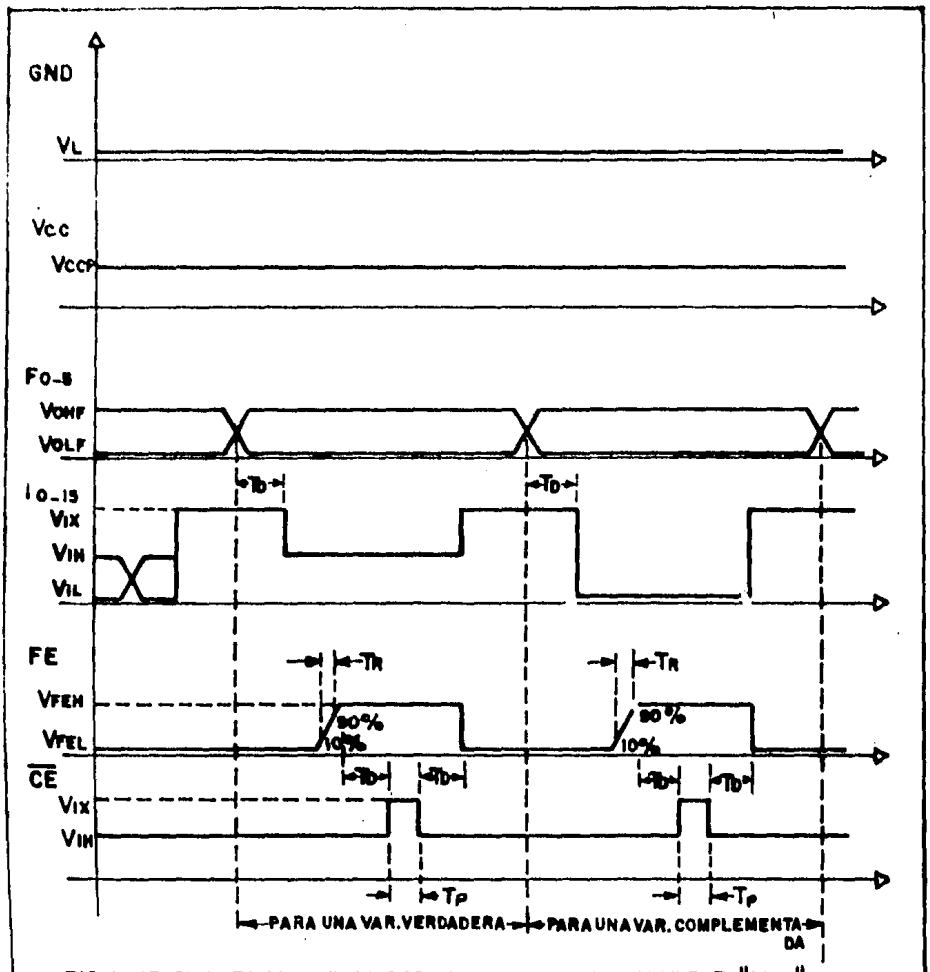


FIG. 3.15 SECUENCIA DE PROGRAMACION DE LA MATRIZ "AND"

(0 volts), la 28 (V_{cc}) a V_{CCS}, la 19 ($\overline{CE}$) a --- V_{IH}, las terminales 9, 8, 7, 6, 5, 4, 3, 2, 27, 26, 25, 24, 23, 22, 21, 20 (entradas I_o a I_{15}) a V_{IH} o V_{IL}; después se direcciona el producto P_M (del 0 - al 47) que no este contenido en la salida F_L, alimentando el código binario correspondiente, en las

entradas I_0 a I_5 (donde I_0 corresponde al bit menos significativo) y así quemar el fusible de este producto P_M, forzando la salida F_L a V_{OPF} y después de un tiempo T_D se aumenta el voltaje de FE desde -- V_{FEL} a V_{FEH}, se espera otro tiempo T_D se aumenta el voltaje de $\overline{CE}$ desde V_{IH} a V_{IX} durante el período T_p

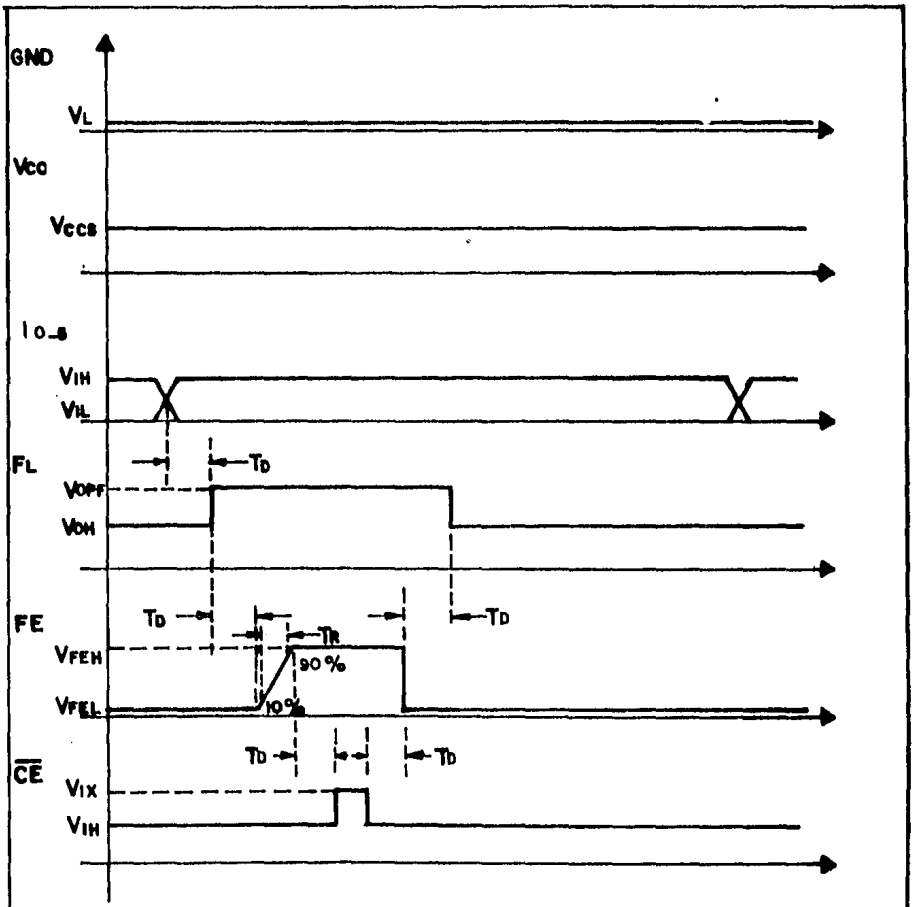


FIG. 3.16 SECUENCIA DE PROGRAMACION DE LA MATRIZ "OR"

y se vuelve a dejar en V_{IH}, se deja pasar un tiempo T_D y se regresa FE a V_{FEL}, se deja pasar otro tiempo T_D y se quita el voltaje V_{OPF} aplicado a la salida F_L. Inicialmente, los FPLA's en su estado virgen, contienen en sus salidas a todos los productos, de manera que si se desea que algún producto quede contenido en una salida, su fusible se deja intacto. Los fusibles de los productos P_M -en la matriz OR- correspondientes a salidas no usadas y los de productos P_M no usados, no necesitan quemarse. La Fig. 3.16 muestra la secuencia de programación.

3.5 PROGRAMADOR

Así como la grabación de los FPLA's difiere de - la de los PROM's, el equipo con el cual se realiza, también es diferente y esto es debido principalmente a que en los PROM's, la matriz AND ya viene grabada por el fabricante (en los PAL's viene grabada la matriz OR) mientras que en los FPLA's, el arreglo de salida, la matriz AND y la matriz OR son grabadas por el usuario; en las Figs. 3.17, 3.18 y - - 3.19 se muestra este aspecto, en donde las conexiones realizadas por el fabricante se presentan por (o) y las realizadas por el usuario, por (■).

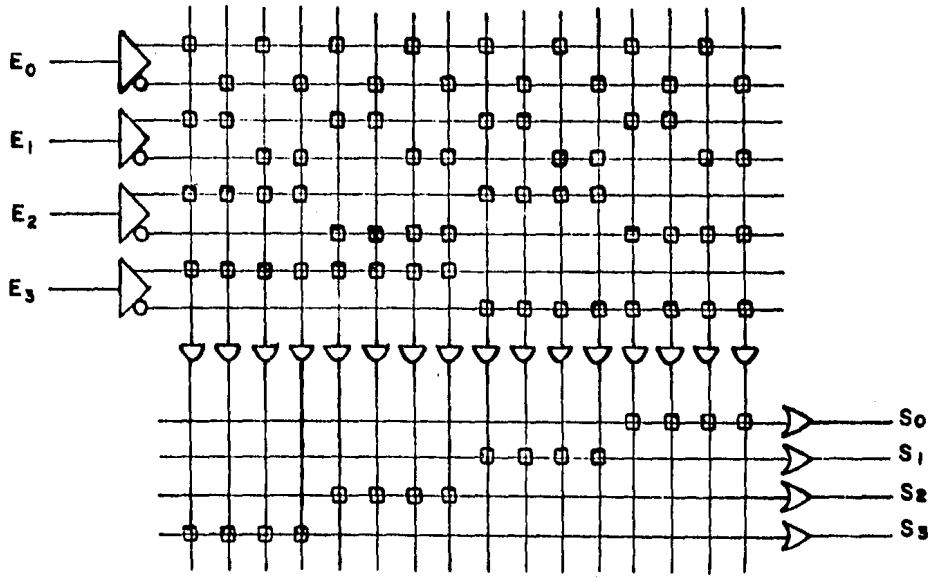


FIG.3.17 REPRESENTACION ESQUEMATICA DE UN FPLA MOSTRANDO LA PROGRAMACION DEL USUARIO (□).

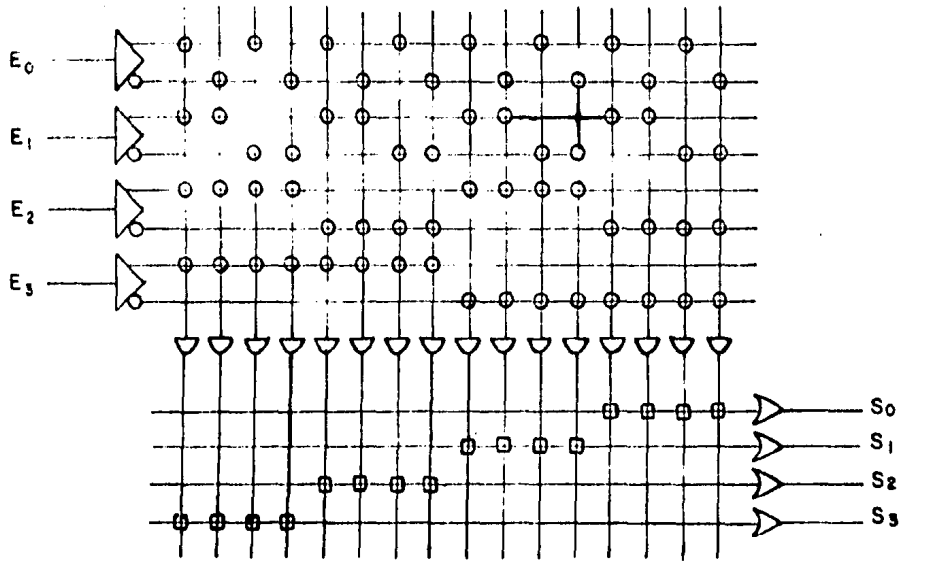


FIG. 3.18 REPRESENTACION ESQUEMATICA DE UN PROM MOSTRANDO LA PROGRAMACION DEL USUARIO (□) Y LA DEL FABRICANTE (○).

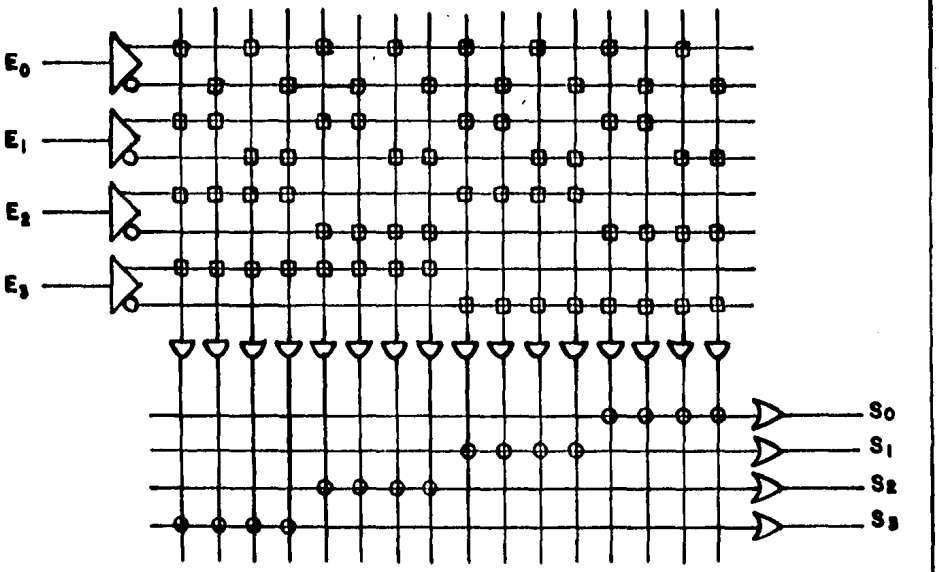


FIG. 3.19 REPRESENTACION ESQUEMATICA DE UN PAL MOSTRANDO LA PROGRAMACION DEL USUARIO (□) Y LA DEL FABRICANTE (○).

El principio de los progamadores es general y es implementado por cada fabricante, de acuerdo con -- las características de sus componentes; en la Fig.3.20 A y B, se muestra el diagrama del programador manual del 82S100/101 de la Signetics, realizado con reguladores de 5 Volts (7805) y uno de 2 a 7 Volts -- (A723); cuenta con cinco medidores para comprobar - las salidas (17 V, 10 V, 8.75 V y dos de 5 V) utilizadas durante la programación y verificación del -- componente por V_{CCS}(8.75), V_{CCP} y V_{IH} (5), V_{OPF} y V_{IX}(10), V_{OPH} y V_{FEH}(17); además, cuenta con led's indicadores del estado de las salidas y entradas -- del componente para verificar su programación; con un pulsador (PB1) conectado a un Flip-flop SR (para evitar falsos disparos) el cual está conectado a un multivibrador monoestable que aplica un voltaje VIX a $\overline{CE}$ durante un tiempo T_p -fijado por R_4 y C_3-; -- con un pulsador (PB2) que aplica un pulso con un frente de subida T_R determinando por R2, R3, y C2, y un -- voltaje V_{OPH} el cual permanece durante un período - de tiempo T_p determinando por R3 y C2; y 6 interruptores con los que se selecciona la programación ya sea de la matriz AND, matriz OR o la polaridad de - salida, así como la verificación de éstas.

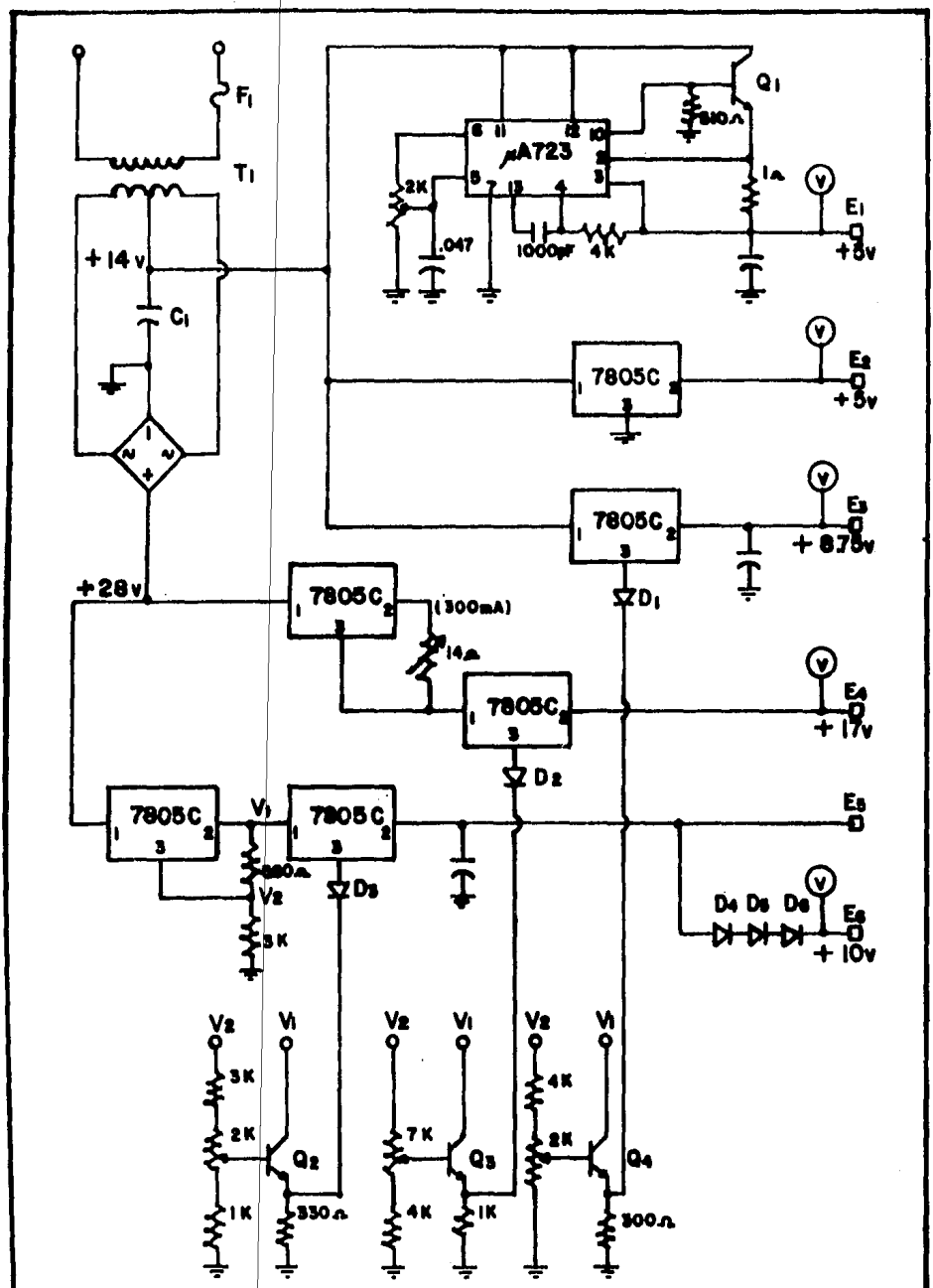


FIG.3.20A FUENTE DE ALIMENTACION DEL PROGRAMADOR MANUAL.

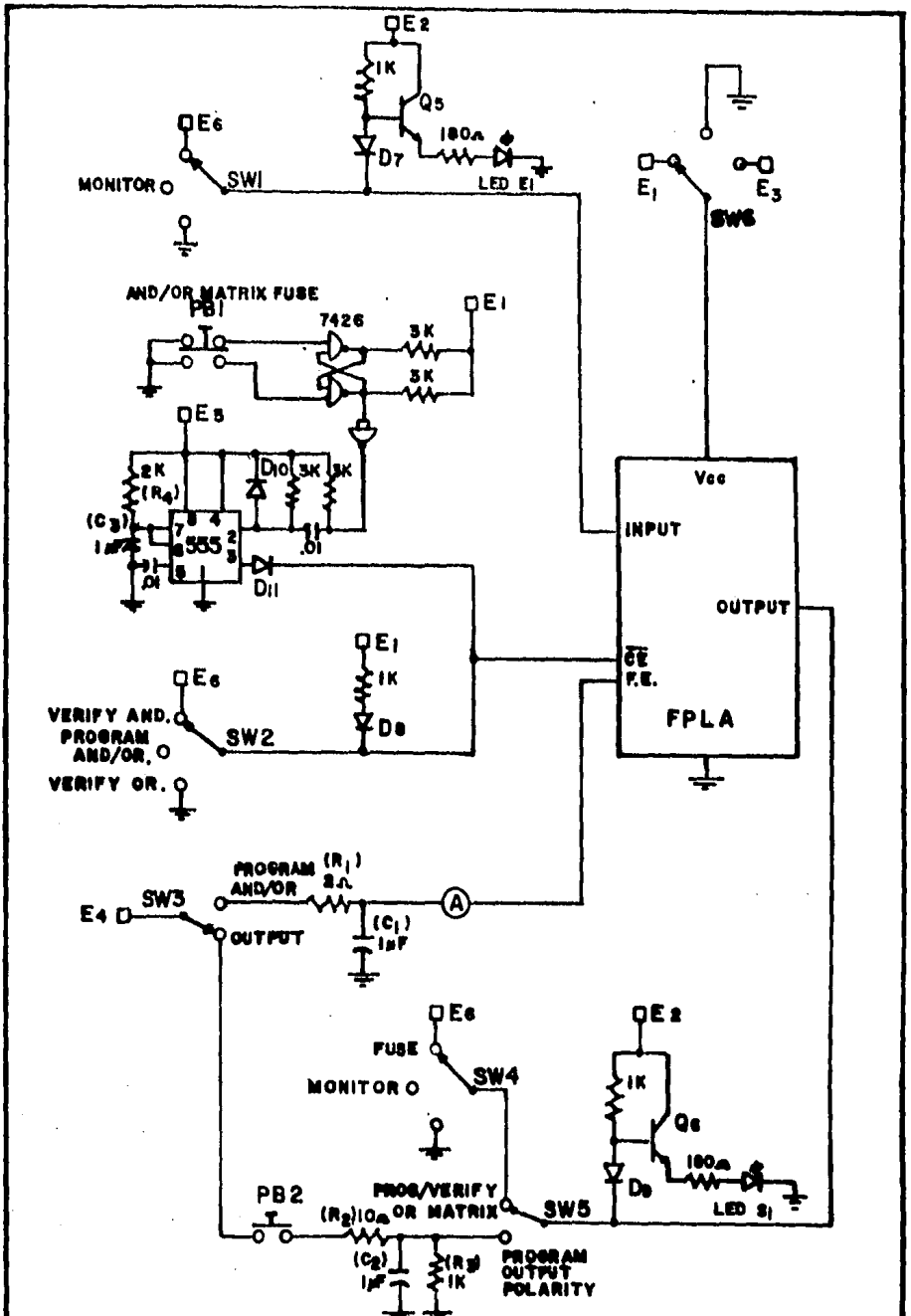


FIG.3.20B PROGRAMADOR MANUAL DE FPLA.

CAPITULO 4

CRITERIOS DE DECISION ENTRE EL USO DE UN FPLA Y UN PROM.

El FPLA es una estructura lógica de propósito general, cuya flexibilidad le permite compararse con los PROM's, ya que en algunas aplicaciones específicas, es una alternativa que ofrece mayor economía y rapidez de solución, obteniéndose un mejor funcionamiento del sistema de procesamiento de datos; tanto el FPLA como el PROM son memorias de solo lectura - programables por el usuario y cada uno posee sus -- propias características, de tal manera que para una aplicación determinada, las de uno pueden ser más - convenientes que las del otro, pues como se ha di-- cho en capítulos anteriores, la principal limita--- ción de los FPLA's, es el número de productos (o palabras) que puede almacenar; aunque hay aplicacio-- nes que no requieren de una gran combinación de las variables de entrada en sus funciones de salida, -- también ayuda mucho la imaginación que se tenga a - la hora de hacer el diseño.

Los principales aspectos que influyen para decidir cual de ellos debemos utilizar en la solución - de nuestro problema son: La forma en que ambos se - direccionan, el número de productos (o capacidad de almacenamiento), el número de variables de entrada, voltaje de operación, costo del componente, etc.

4.1 DIRECCIONAMIENTO

La capacidad de compresión lógica de los FPLA's, es una consecuencia de la matriz AND de direccionamiento programable, la cual desempeña la función del necesario decodificador interno del PROM; la matriz AND, elige un sub-grupo finito de entre todos los posibles estados de entrada (como por ejemplo un directorio telefónico, en el que se encuentran los nombres de los que estan inscritos y no todas las posibles combinaciones de los nombres que existen), por eso se dice que un FPLA es un PROM con direcciones programables.

Cada columna de la matriz de direccionamiento del FPLA, funciona como un comparador lógico, el cual es programado para reconocer la presencia simultánea de cada entrada, y el estado de éstas puede ser cierto, falso, ambos (Don't Care) o lógica nula; esto es, cuando cualquier combinación programada aparece en la entrada, la columna correspondiente a la matriz de direccionamiento se pone alta (lógica activa), forzando a todas las salidas conectadas a esa columna a sus estados lógicos ciertos; a la inversa, en todas las combinaciones lógicas no programadas a las entradas, las columnas permanecen bajas (lógica nula), forzando a todas las salidas conectadas a la columna a su estado falso por incumplimiento. Esto no sucede con los PROM's, ya que

en ellos es necesario programar todas las posibles combinaciones de las entradas -aunque haya muchas que no se utilicen-, mientras que en los FPLA's podemos prescindir de hasta n-1 variables de entrada para direccionar alguna palabra, además una combinación de entrada, puede direccionar varias palabras (direccionamiento coincidente), o también, diferentes combinaciones, pueden direccionar una sola palabra (direccionamiento múltiple).

La tabla de verdad de la fig. 4.1A, muestra el comportamiento de un sistema digital, y es la única información que se requiere para grabar las direcciones de cada palabra en un PROM, por otro lado, en la Fig. 4.1B aparecen las funciones minimizadas, en la Fig. 4.1C el mapa de actividades y en la Fig. 4.1D la tabla de programa, en la que se puede apreciar la compresión lógica obtenida, al reducirse el número de direcciones de 16 a 4; finalmente, en la Tab. de la Fig. 4.2, se puede ver que las palabras P_1, P_2 y P_4, son direccionadas por 1111 (direccionamiento coincidente), y que P_1 es direccionado por 0101, 0111, 1101 y 1111 (que corresponde a un direccionamiento múltiple).

D	C	B	A	F_1	F_2	F_3
0	0	0	0	0	1	1
0	0	0	1	0	1	1
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	1	1	0
0	1	1	0	0	0	0
0	1	1	1	1	1	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	1
1	1	0	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	0	0	1
1	1	1	1	1	1	1

(A)

$F_1 = CA + DBA$

$F_2 = CA + \bar{D}\bar{C}\bar{B}$

$F_3 = DC + DBA + \bar{D}\bar{C}\bar{B}$

(B)

$F_1 = P_1 + P_2$ Para: $P_1 = CA$

$F_2 = P_1 + / + P_3$ $P_2 = DBA$

$F_3 = / + P_2 + P_3 + P_4$ $P_3 = \bar{D}\bar{C}\bar{B}$

$P_4 = DC$

(C)

D	C	B	A	F_1	F_2	F_3
-	H	-	H	A	A	·
H	-	H	H	A	·	A
L	L	L	-	·	A	A
H	H	-	-	·	·	A

(D)

FIG. 4.1 (A) TABLA DE VERDAD, (B) FUNCIONES DE SALIDA, (C) MAPA DE ACTIVIDADES Y (D) TABLA DE PROGRAMA, DE UN SISTEMA DIGITAL.

D	C	B	A	PRODUCTO
0	0	0	0	P_3
0	0	0	1	P_3
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	P_1
0	1	1	0	
0	1	1	1	P_1
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	P_2
1	1	0	0	P_4
1	1	0	1	P_1, P_4
1	1	1	0	P_4
1	1	1	1	P_1, P_2, P_4

FIG. 4.2 DIRECCIONAMIENTO COINCIDENTE Y MULTIPLE.

En general, un producto puede ser direccionado - por diferentes combinaciones de acuerdo a la si---- guiente relación:

$$(M_n)_T = 2^{m-r}$$

donde: m = número de variables de entrada.
r = número de entradas activas (cierta o complemento) contenidas en el producto.

Por ejemplo, para P_o=XXXI$_o$; m= 4 y r = 1, el número de combinaciones que pueden seleccionar al producto P_o es $(M_n)_T = 8$.

Además de la compresión sustancial de las tablas de verdad, los FPLA's tienen una capacidad de redacción, la cual les permite cambios posteriores en el diseño del programa, y esto se puede hacer con los productos que no se usaron y que fueron dejados intactos después de la programación inicial, ya que pueden ser programados posteriormente para combinarse a cualquier función de salida; asimismo, un producto cualquiera puede ser eliminado de una función de salida después de la programación inicial.

4.2 NUMERO DE COMBINACIONES

El número de combinaciones diferentes realizadas con todas las variables de entrada (verdaderas y/o complementadas) que son reconocidas por un FPLA, -- depende de su matriz AND y es de 48 en uno de 14 x 48 x 8, mucho menos de las que puede reconocer un PROM que es de 512 (2^9) en uno de 512 x 8 bits, el cual depende de la capacidad de su decodificador, por lo tanto, cuando se desee implementar una función -

que necesite menos de 48 productos diferentes, será mas conveniente utilizar un FPLA. Considerese por ejemplo, la tabla de verdad de la Fig. 4.3 con la función "S" de 9 variables; la función es cierta solo cuando todas las variables son ciertas, y falsa de otra manera. Un PROM necesitaría que los 2^9 min-- términos -en su totalidad- fueran representados en memoria, mientras que un FPLA solo necesitaría utilizar uno de sus 48 productos disponibles, en el -- que se tendría que grabar el estado verdadero de -- las variables, por supuesto, la respuesta lógica -- puede ser fácilmente suministrada con una compuerta AND de 9 entradas, pero si el número de mintérminos para que "S" sea cierta se aumentara a 30, ya no bastaría una sola compuerta AND, el PROM tendría de -- nuevo que usar toda su memoria y el FPLA todavía le sobraría parte útil que podría ser utilizada por -- otra función.

E_9	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	S
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	1	1	1

FIG. 4.3 TABLA DE VERDAD DE LA FUNCION DE SALIDA "S".

A continuación se dan tres reglas basadas en el número de combinaciones, que sirven de mucha ayuda en la decisión de selección entre un FPLA y un PROM:

1a. Si menos del 8% de las combinaciones de entrada de un problema son utilizadas para obtener las funciones de salida, los FPLA's son mas económicos.

2a. Si más del 20% de las combinaciones de entradas de un problema son utilizadas para obtener las funciones de salida, los PROM's son mas convenientes.

3a. Si el número de combinaciones se encuentra comprendido entre el 8% y el 20%, la decisión depende del precio de los componentes y de la programación de ellos.

El uso de los FPLA's en aplicaciones de funciones estandard tales como el convertidor Hollerith - ASCII, tienden a ser la excepción mas bien que la - regla, pues por ejemplo, en éste, solo se utiliza - el 2.3% de las posibles combinaciones de entrada; - muchas funciones usan mas del 20% de sus combinaciones de entrada y es por ello que los FPLA's no pueden desplazar a los PROM's.

4.3 NUMERO DE VARIABLES DE ENTRADA.

El número de variables de entrada, es uno de los factores mas importantes en la selección entre un PROM y un FPLA; cuando se necesitan mas de nueve, - el FPLA en general, es la elección mas económica, y esto es en parte, debido a la organización de los - PROM's comunes, ya que uno de los de mayor capaci-- dad, es de 8K bits (10 entradas y 1024 salidas de 8 bits). Así, si contamos con un PROM de 8K bits -- (mostrado en la Fig. 4.4) y se tubiera la necesidad de uno de 14 entradas en lugar de 10 entradas de -- dirección, se tendría que realizar un arreglo de -- 16,384x8 bits, utilizando para ello 16 PROM's de - 8K bits y un decodificador 1 de 16 (Fig. 4.5A), y esto sería demasiado grande para estar contenido en un - solo componente. Ahora bien, si la función que se fuera a grabar en el PROM requiriera mucho menos -

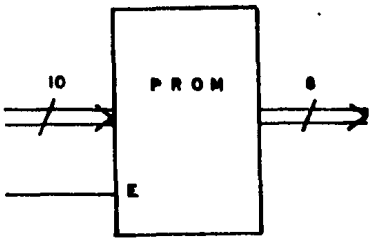


FIG. 4.4 CONFIGURACION DE UN PROM DE 2^{10} PALABRAS.

de las 2^{14} palabras, contando aún con las 14 entradas, podría ser mas económico usar una clase especial de PROM, es decir, un sistema con "n" FPLA's - (Fig. 4.5B) para suministrar la capacidad de memoria requerida por la función, es decir, si el número de productos diferentes está entre 48 y 96 dos - FPLAS serán suficientes, pero si es mayor este número

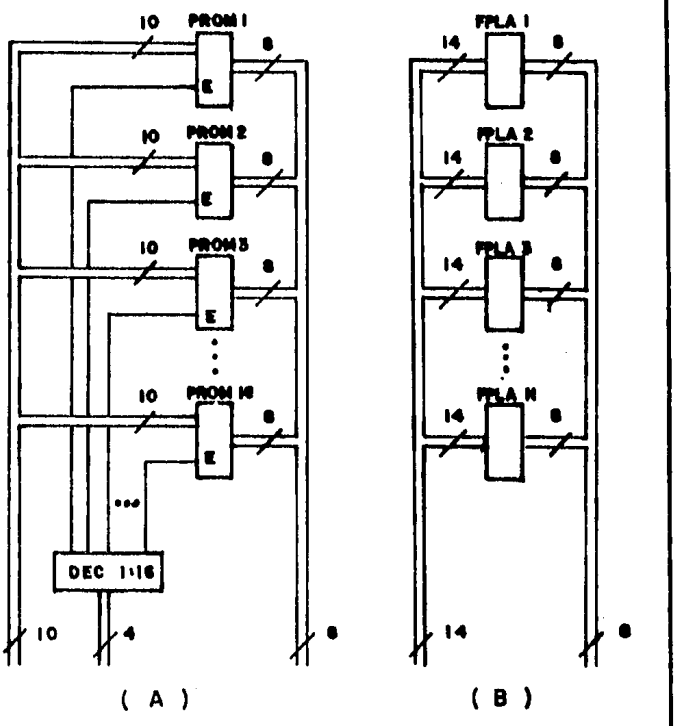


FIG. 4.5 (A) PROM EQUIVALENTE DE 2^{14} PALABRAS DE 8 BITS, (B) FPLA EQUIVALENTE 14 X 48N X 8.

ro, entonces serán necesarios más FPLAS.

4.4 PROPIEDADES FISICAS.

Un aspecto importante durante el análisis entre el uso de un FPLA y un PROM, es el tamaño de cada - uno de los componentes, las fuentes y voltajes necesarios para su operación, la disipación de potencia, la velocidad de acceso y la compatibilidad con - - otros elementos de diseño.

Cuando se está diseñando el circuito impreso, es de gran importancia conocer el tamaño de los componentes que se van a montar en éste, ya que son estos los que determinan el tamaño de la tablilla y - en la mayoría de los casos se encuentra restringido a un tamaño máximo o forma determinada. Los FPLA's y PROM's, varían de acuerdo al número de entradas y salidas que manejan y en general son mayores las dimensiones de los primeros, así, tenemos por ejemplo que el FPLA82S100 (16x48x8) tiene 28 terminales y - viene encapsulado en un paquete de cerámica cuyas - dimensiones son 1.58cm x 3.78cm, mientras que el -- PROM 82S181 (1024x8 BITS) tiene 24 terminales y - - también se encuentra encapsulado en un paquete de - cerámica cuyas dimensiones son 1.58cm x 3.28cm. -- Resulta más práctico construir sistemas con componentes que utilicen una sola fuente de alimentación, que construirlos con componentes que necesiten mas de una, el 82S100 requiere una sola fuente de +5 -- Volts, así como también el 82S181; en este caso, no

hay ninguna ventaja entre cualquiera de ellos, a no ser que sea necesario utilizar mayores voltajes, - para lo cual se cuenta con PROM's de la familia MOS (aunque son mas lentos); la disipación de potencia de los componentes, toma importancia cuando son utilizados varios de ellos, o cuando forman parte de - un gran número de componentes, ya que una gran disipación, además del calor generado, representa una - fuente de alimentación de gran capacidad; el 82S100, tiene una disipación de potencia típica de 600 mW, y puede trabajar en un medio cuya temperatura se encuentre entre 0 - 75°C, mientras que el 82S181, tiene una disipación de potencia de 622 mW y puede trabajar a una temperatura entre 0-75°C. Cuando la - velocidad de operación del sistema es un factor importante para su óptimo funcionamiento, cada componente representa un retrazo, y los de pequeño tiempo de operación, son los mas requeridos; el 82S100, tiene un tiempo de operación del orden de 50 nseg., menor que el de 82S181 que tiene un tiempo de operación de 70 nseg. El contar con elementos que puedan ser usados en diferentes familias digitales o que exista variedad de componentes para cada una de las familias, es uno de los aspectos que los hace - mas versátiles y en este caso, los PROM's llevan la delantera sobre los FPLA's, pues cuentan con elementos para las familias MOS y TTL; los FPLA's solo -- son utilizados con familias TTL (claro que es posible construirlos con matrices de diodos y de esta -

manera podrían trabajar a diferentes voltajes).

4.5 CAMPOS DE APLICACION

El FPLA, puede usarse como un PROM, pero en lugar de construir lógica con PROM's, el FPLA permite al PROM ser construido con lógica; sus principales aplicaciones son aquellas en las que se requiere gran velocidad de acceso, organización de palabras no estandard, capacidad de bits no usuales, o decodificadores de esquemas que no siguen una secuencia binaria directa y hasta varias de éstas al mismo tiempo.

Una aplicación natural de los FPLA's, la representa el convertidor de código Hollerith -ASCII mostrado en la Fig. 4.6, el cual, tiene 12 variables de entrada y representan la posibilidad de 2^{12} palabras, pero el código Hollerith contiene solo 96 caracteres gráficos del total de estados codificados. La matriz AND del FPLA, alojaría el código Hollerith y la matriz OR, almacenaría el código ASCII, los elementos de salida se programan como inversores, para que las salidas se pongan altas cuando el componente sea deshabilitado; las terminales 10 y 11 son habilitadoras del componentes y pueden ser usadas para deshabilitarlo cuando así se desee.

En la aplicación de un circuito contador decodi-

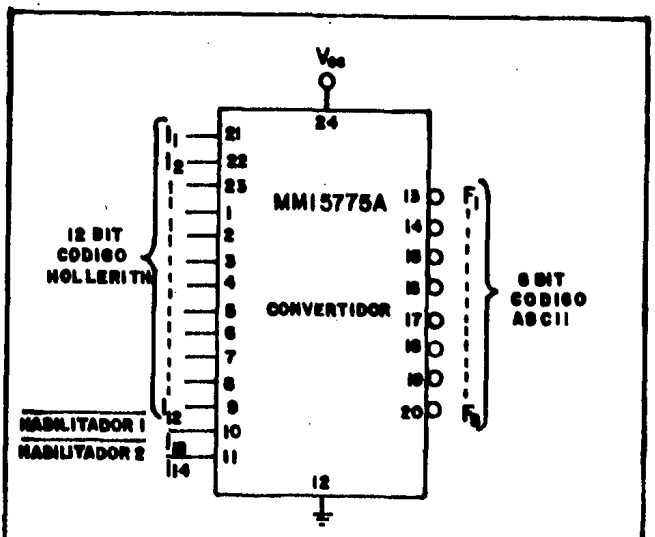


FIG.4.6 CONVERTIDOR DE CODIGO HOLLERITH A CODIGO ASCII.

ficador (Fig. 4.7), el tiempo entre 96 eventos separados, pueden ser seleccionados de entre 2^{14} intervalos de tiempo del contador compuesto de cuatro -- componentes; la selección se lleva a cabo en el - - arreglo de entrada (matriz AND) del FPLA. El arreglo de salida (matriz OR), se divide en dos partes:

1. Las salidas F_8 a F_5 seleccionan uno de los 6 decodificadores 74154 (1 de 16).

2. Las salidas F_4 a F_1, direccionan a todos los decodificadores en paralelo, pero solo el decodificador seleccionado (por F_8 y F_5) produce una indicación de evento.

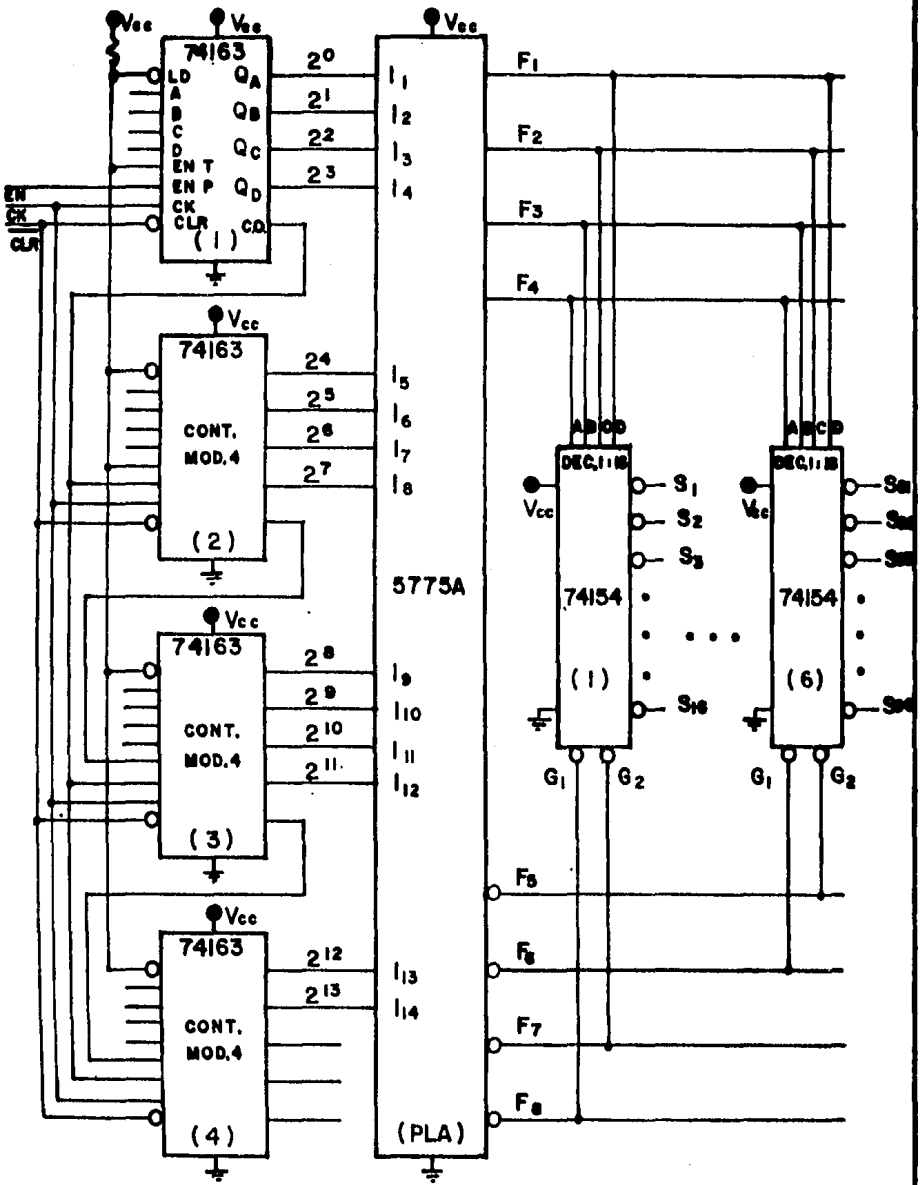


FIG. 4.7 CIRCUITO CONTADOR-DECODIFICADOR DE 96 EVENTOS.

NUMERO DE EVENTO (S_{96}) / SALIDA DEL CONTADOR	↓	I_{14} I_{13} I_{12} I_{11} I_{10} I_9 I_8 I_7 I_6 I_5 I_4 I_3 I_2 I_1	F_8	F_7	F_6	F_5	F_4	F_3	F_2	F_1	DEC. SELECCIONADO ↓	
			L	L	L	L	H	H	H	H		
170	1	L L L L L L H L H L H L H L	A	A	·	·	·	·	·	·	1	0
365	2	L L L L L L H L H H L H H L H	A	A	·	·	·	·	·	A	1	1
512	3	L L L L L H L L L L L L L L L	A	A	·	·	·	·	A	·	1	2
≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
2560	15	L L H L H L L L L L L L L L L	A	A	·	·	A	A	A	·	1	14
2728	16	L L H L H L H L H L H L L L L	A	A	·	·	A	A	A	A	1	15
2820	17	L L H L H H L L L L L H L L L	A	·	A	·	·	·	·	·	2	0
2884	18	L L H L H H L H H L H H L L L	A	·	A	·	·	·	·	A	2	1
≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
5440	32	L H L H L H L H L L L L L L L	A	·	A	·	A	A	A	A	2	15
5473	33	L H L H L H L H H L L L L H	A	·	·	A	·	·	·	·	3	0
5489	34	L H L H L H L H H H L L L H	A	·	·	A	·	·	·	A	3	1
≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
8118	48	L H H H H H H L H H L H H L L	A	·	·	A	A	A	A	A	3	15
8300	49	H L L L L L L H H L H H L L	·	A	A	·	·	·	·	·	4	0
8428	50	H L L L L L H H H L H H L L	·	A	A	·	·	·	·	A	4	1
≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
10801	64	H L H L H L L L H H L L L H	·	A	A	·	A	A	A	A	4	15
10811	65	H L H L H L L L H H H L H H	·	A	·	A	·	·	·	·	5	0
10939	66	H L H L H L H L H H H L H H	·	A	·	A	·	·	·	A	5	1
≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
13631	80	H H L H L H L L H H H H H H H	·	A	·	A	A	A	A	A	5	15
13831	81	H H L H H L L L L L L H H H	·	·	A	A	·	·	·	·	6	0
13895	82	H H L H H L L H L L L H H H	·	·	A	A	·	·	·	A	6	1
≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
16383	96	H H H H H H H H H H H H H H H	·	·	A	A	A	A	A	A	6	15
PESO → BINARIO DEL CONTADOR		8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1	PESO → DE LA DIR. DEL DECODIFICADOR				8	4	2	1	↑ DIRECCION DEL DECODIFICADOR	

FIG. 4.8 TABLA DE PROGRAMA DEL CIRCUITO CONTADOR-DECODIFICADOR DE 96 EVENTOS.

Las salidas F_8 a F_5, se programan como inversores; si el contador no está en ningún evento de tiempo particular (ningún término es direccionado y las salidas están en el estado apagado), las cuatro salidas se pondrán altas y deshabilitarán los seis decodificadores, las salidas F_4 a F_1 se programan como no-inversores, así, la mitad derecha del arreglo de salida puede ser programado en secuencia binaria directa (Fig. 4.8).

La mayor demanda de terminales y el acoplamiento hombre/máquina, requieren caracteres mas legibles y de generadores mas rápidos, tareas en las que el FPLA puede también ser utilizado.

Los generadores alfanuméricos y de símbolos de propósito especial, están basados generalmente en patrones de información binaria almacenados en ROM's y son obtenidos a sus salidas de acuerdo con los códigos presentados a sus entradas, y se utilizan para modular el eje "z" en las pantallas TRC (Tubo de rayos catódicos). El haz de electrones se conecta y desconecta en sincronía con la señal de deflexión, para presentar el patrón de puntos y formar el caracter deseado; los formatos de caracter de mayor uso, se presentan en cinco columnas y siete renglones (5 x 7) ó en siete columnas y nueve renglones (7 x 9); Los puntos/caracter adicionales mejoran la legibilidad, pero los requerimientos de almacena

miento también aumentan. La escritura de símbolos de calidad, está limitada en los PROM's debido al - costo por aumento de la capacidad de almacenamiento, el costo de circuitos mas rápidos y la desviación - de formatos convencionales; los FPLA's, cuentan con una velocidad de operación adecuada y los caracte-- res no se almacenan como patrones de puntos de tama ño fijo, sino que se almacenan como estructuras que son descritas por sumas de productos dentro de ecua ciones lógicas, y el mejoramiento de los generado-- res de caracter, no incrementa el requerimiento de almacenamiento tanto como en el PROM. Considere-- mos por ejemplo un formato de puntos de 13 x 17 con el que puede obtenerse muy buena legibilidad, como puede apreciarse con la letra "Q", la cual se muestra en la Fig. 4.9 usando un PROM, necesitaríamos 221 - bits por cada caracter, ahora bien, utilizando un - FPLA, el número de bits sería variable dependiendo del caracter mostrado, en el caso de la letra "Q", se necesitarían 12 productos del FPLA, y si usaramos - un (.) punto, solo utilizaríamos un producto.

La Fig. 4.10 muestra el diagrama lógico de un -- generador de caracteres utilizando FPLA's, Al ini cio de operación el contador módulo 13 está en 0000 y el contador módulo 17 está en 00000 y en las sali das de los FPLA's se encuentra el contenido del pri mer renglón, mientras que a la salida del multiplex or aparece el primer Bit de este renglón y son re--

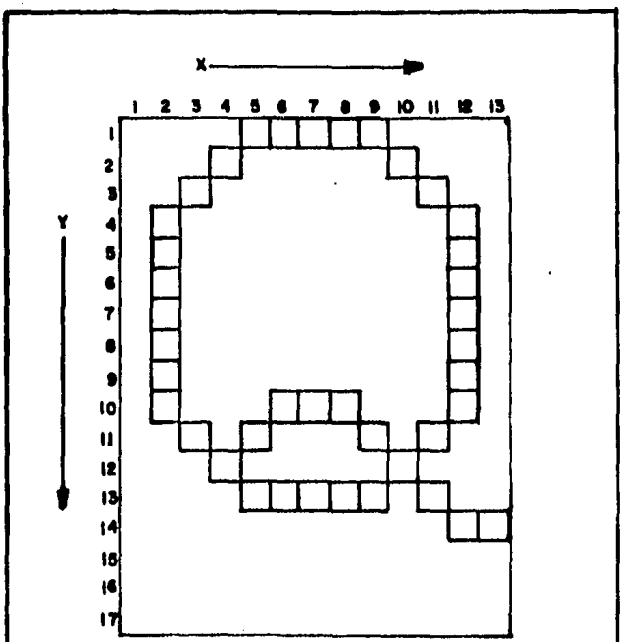


FIG.4.9 REPRESENTACION DE LA LETRA "Q" CON UN FORMATO DE 13 x 17 PUNTOS.

presentados los siguientes bit's de este mismo renglón en cada pulso del reloj, hasta que se presenta el último, momento en el cual se incrementa el contador módulo 13 y el contador módulo 17 se pone de nuevo en 00000 presentado así el primer bit del siguiente renglón y así sucesivamente hasta que son presentados todos los bit's de la matriz 17x13.

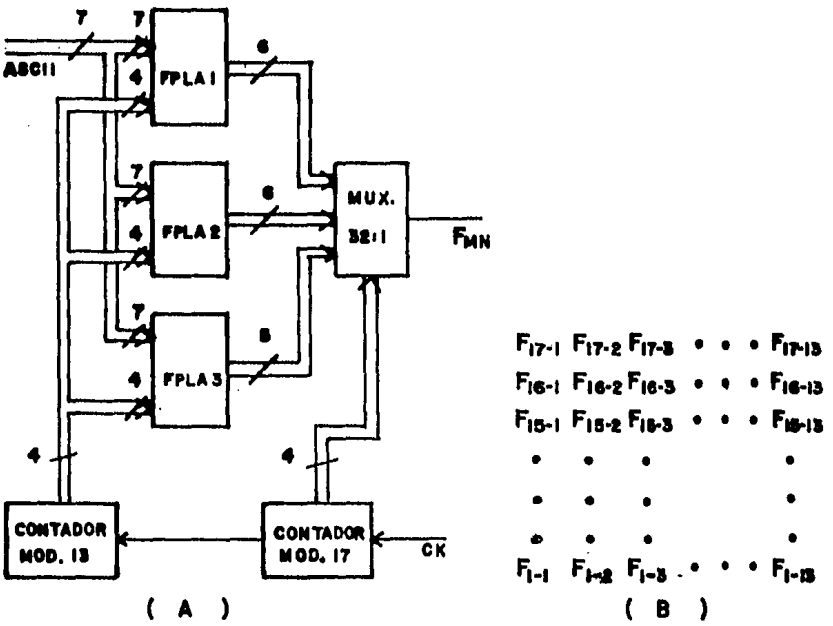


FIG. 4.10 (A) DIAGRAMA LOGICO DE UN GENERADOR DE CARACTERES PARA UNA MATRIZ DE 17 x 13. (B) MATRIZ DE PUNTOS DE 17 x 13.

Los PROM's tienen una gran aplicación en sistemas decodificadores standard, en convertidores de código de uso común, en generador de caracteres y en menor escala en circuitos secuenciales (realizando la función del circuito combinacional).

A P L I C A C I O N

La aplicación que aquí se presenta tiene como -- principal objetivo, mostrar el camino que se sigue durante el diseño de un circuito implementado con - FPLA's.

El convertidor de código ASCII/MORSE reversible fué escogido de entre diferentes posibles aplicaciones, ya que, es en el campo de los convertidores - donde los PLA's exhiben sus mejores características.

Este convertidor de código viene a ser una parte importante dentro de un sistema telegráfico sencillo que podría ser usado en lugares donde no se requieren equipos muy sofisticados, ni muy caros. La primera ventaja que se trasluce, al utilizar el convertidor de código, es que la persona que deba tener a su cargo la trasmisión y recepción de mensajes por la línea telegráfica, no tendrá la necesidad de saber el código MORSE y verse obligado a manejar el manipulador (llave MORSE) para el envío de mensajes o interpretar el cambio de tonos, que representan a los puntos y rayas del código MORSE, en la recepción de los mismos, si no más bien, mediante un teclado en donde estén las letras y símbolos de uso común, podrá escribir de una manera simple -

los mensajes que desee enviar, para lo cual el convertidor se encargará de traducir en código MORSE y de esta forma puede ser transmitido por la línea o durante la recepción el convertidor puede traducir, en código ASCII, la información que se esté recibiendo y alimentar una impresora donde los mensajes sean escritos. Es claro que el convertidor de código, por sí solo, no puede desempeñar esta función, sino que, necesitará de un controlador que además de manejarlo se encargue de sincronizar las señales del receptor con las del transmisor y viceversa.

5.1. DISEÑO DE UN CONVERTIDOR DE CODIGO ASCII/ MORSE REVERSIBLE.

Como primer paso debe quedar clara y perfectamente especificada la función que va a realizar el circuito que se ha de diseñar. En este caso el convertidor deberá ser capaz de presentar a su salida el código ASCII equivalente del MORSE presentado a la entrada o el Código MORSE equivalente del ASCII presentado a la entrada, para lo cual el circuito contará:

1) Con una entrada de control, mediante la cual pueda ser seleccionada la función que va a realizar -convertir código ASCII a MORSE ó MORSE a ASCII- que la podemos llamar "SELECTOR" y representar mediante una "S" que tendrá un nivel lógico alto

en el primer caso y bajo en el segundo.

2) Con una entrada de control, mediante la cual pueda ser puesto en un estado activo cuando se está haciendo uso de él ó inactivo cuando no está siendo usado (cuando no se está transmitiendo información) y que podemos llamar "RESET" y representar mediante una "R", la cual al tomar un estado lógico alto permite trabajar al convertidor y cuando toma un nivel bajo lo inhibe y manda a todas sus salidas a un estado lógico bajo.

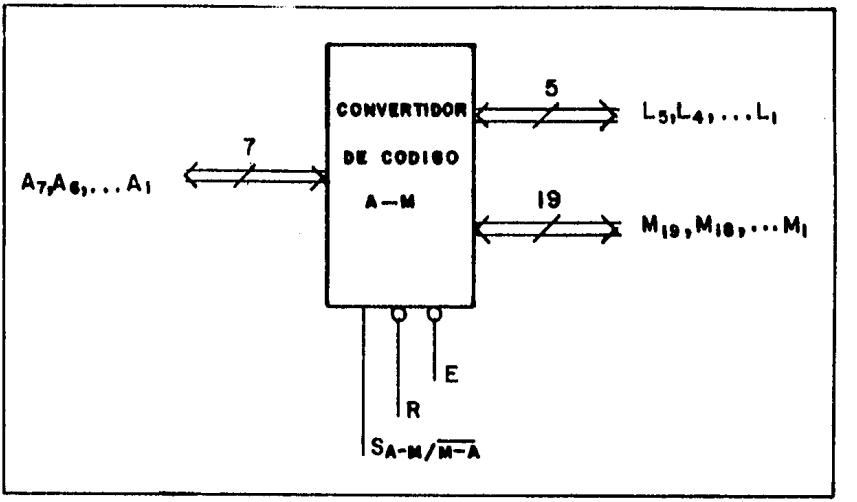
3) Con una entrada de control, mediante la cual pueda ser habilitado o deshabilitado, para que pueda ser usado en circuitos de bus compartido, que podemos llamar "ENABLE" y representar con una "E", la cual al tomar un estado lógico alto, pondrá todas las salidas del convertidor en "tri-state" y al tomar un estado bajo lo habilitará permitiéndole operar.

4) Con siete líneas (A_7, A_6,...A_1) donde podrá obtenerse o depositarse el código ASCII.

5) Con diez y nueve líneas (M_{19},M_{18},...M_1) donde podrá obtenerse o depositarse el código MORSE (la palabra MORSE de mayor longitud consta de 19 Bit's).

6) Con cinco líneas (L_5, L_4,...L_1) en donde podrá ser indicada la longitud del código MORSE (la longitud máxima es de 19, es decir, 10011) ya que éste es de longitud variable.

En la Fig. 5.1.A mostramos el convertidor como una caja negra con todas sus entradas y salidas y en la Fig.5.1.B su tabla de verdad.



(A)

	ASCII					MORSE									
E	R	S	A_7	A_6	A_5 ...	A_1	↔	M_{19}	M_{18}	M_{17} ...	M_1	L_5	L_4	L_3 ...	L_1
0	0	0	0	0	0	0	-	M_{19}	M_{18}	M_{17}	M_1	L_5	L_4	L_3	L_1
0	0	1	A_7	A_6	A_5	A_1	-	0	0	0	0	0	0	0	0
0	1	0	A_7	A_6	A_5	A_1	←	M_{19}	M_{18}	M_{17}	M_1	L_5	L_4	L_3	L_1
0	1	1	A_7	A_6	A_5	A_1	→	M_{19}	M_{18}	M_{17}	M_1	L_5	L_4	L_3	L_1
1	X	0	Z	Z	Z	Z	-	M_{19}	M_{18}	M_{17}	M_1	L_5	L_4	L_3	L_1
1	X	1	A_7	A_6	A_5	A_1	-	Z	Z	Z	Z	Z	Z	Z	Z

(B)

FIG. 5.1. (A) CONVERTIDOR DE CODIGO ASCII-MORSE REVERSIBLE Y (B) SU TABLA DE VERDAD.

Una vez que se ha especificado completamente la función que va a desempeñar el circuito -como caja negra- el siguiente paso consiste en mostrar con -- detalle, como es que lo va a llevar a cabo, para lo cual, es necesario hablar del comportamiento de las señales de entrada, señales de salida y de las in-- terrelaciones que existen entre éstas.

El Alfabeto MORSE se compone de trazos largos y cortos llamados rayas y puntos; por convención, el punto debe ser tres veces más corto que la raya. - Los elementos que constituyen las diferentes combinaciones que se pueden formar con las rayas y puntos, dando lugar a las letras y demás signos, se separan entre sí, por un intervalo que equivale a un trazo corto o sea un punto; el espacio entre dos letras equivale a tres puntos y la separación entre - dos palabras, a cinco. En la Fig. 5.2 se muestra el código MORSE ordenado en forma mnemotécnica, empezando con letras, seguidas de símbolos, señales - de control y por último, números.

El código ASCII (American Standard Code For In-- formation Interchange) está formado por 128 combinaciones de 7 bit's cada una (2^7) como se muestra en la Tabla de la Fig. 5.3. La idea de obtener una - equivalencia entre el código MORSE y ASCII nació -- del hecho de que el primero es el precursor de los códigos utilizados para la transmisión de informa--

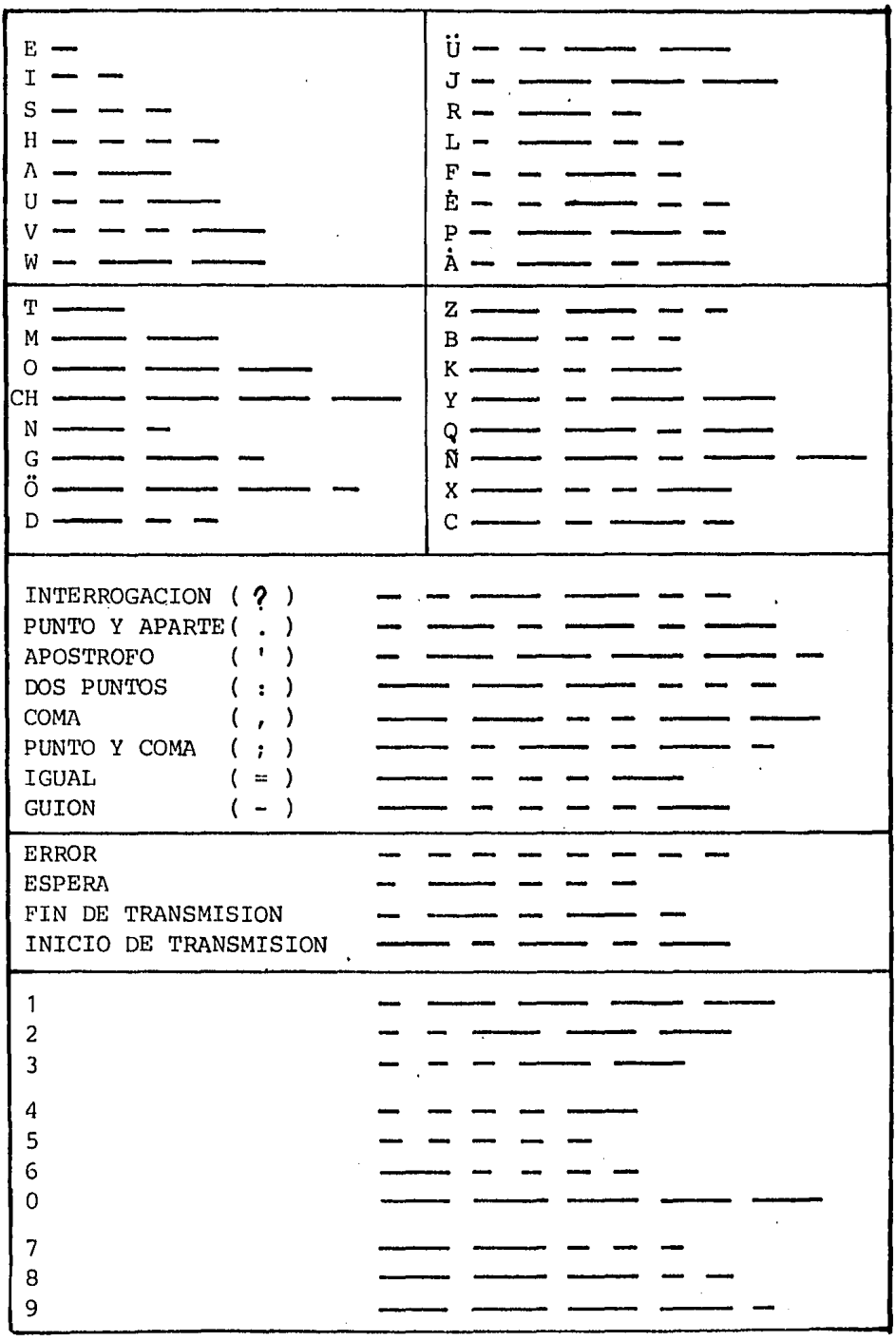


FIG. 5.2 ESTRUCTURA DEL CODIGO MORSE.

$b_4b_3b_2b_1$ \ b_7 b_6 b_5	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	|
1 1 0 1	CR	GS	-	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	_	o	DEL

NUL. NULL.
SOH. START OF HEADING.
STX. START OF TEXT.
ETX. END OF TEXT.
EOT. END OF TRANSMISSION.
ENQ. ENQUIRY.
ACK. ACKNOWLEDGE.
BEL. BELL.
BS. BACK SPACE.
HT. HORIZONTAL TABULATION.
LF. LINE FEED.
VT. VERTICAL TABULATION.
FF. FORM FEED.
CR. CARRIAGE RETURN.
SO. SHIFT OUT.
SI. SHIFT IN.
SP. SPACE.

DLE. DATA LINK ESCAPE.
DC1. DEVICE CONTROL 1.
DC2. DEVICE CONTROL 2.
DC3. DEVICE CONTROL 3.
DC4. DEVICE CONTROL 4.
NAK. NEGATIVE ACKNOLEDGE.
SYN. SINCHRONOUS IDLE.
ETB. END OF TRANSMISSION BLOCK.
CAN. CANCEL.
EM. END OF MEDIUN.
SUB. SUBSTITUTE.
ESC. ESCAPE.
FS. FILE SEPARATOR.
GS. GROUP SEPARATOR.
RS. RECORD SEPARATOR.
US. UNIT SEPARATOR.
DEL. DELETE.

FIG. 5.3 CODIGO ASCII.

ción y el segundo es de los que más aceptación tiene actualmente por tratarse de -un código de lenguaje universal-como ya lo dijo el "American National Standard Institute" de los Estados Unidos.

En la tabla de la Fig. 5.4A y B, se muestra la - equivalencia entre el Código MORSE y ASCII, en donde el Código MORSE está representado mediante "1's" y "0's"; un "punto" es representado con un "1", una "raya" con tres "1" (ya que como se mencionó antes, el punto debe ser tres veces mas corto que una raya) y la separación entre un punto y una raya con un -- "0" (intervalo que equivale a un punto). En esta tabla solo son representados aquellos símbolos que son reconocidos por ambos códigos sin ambigüedades.

En la tabla de las Fig. 5.5 A y B se muestra - - también la equivalencia entre los códigos ASCII y - MORSE, pero aquí es usado el mintérmino correspon-- diente a cada código ASCII en lugar de éste. Además los bit's impares del código Morse ya no son -- presentados, ya que en todos los casos es una "1" (excepto para m_{32}) y a todos ellos se representan - mediante MBI "salida Morse de los bit's impares". Por último, aquí también se incluye el número de -- Bit's -en binario- por los que esta formada la palabra MORSE correspondiente, pues como ya se mostró en la - Fig. 5.2 la longitud de ésta es variable y se deberá indicar para saber cuantos de los 19 Bit's presentados a la

	"ASCII"	$A_7A_6A_5A_4A_3A_2A_1$		"MORSE" $M_{19}M_{18}M_{17}M_{16}M_{15}M_{14}M_{13}M_{12}M_{11}M_{10}M_9M_8M_7M_6M_5M_4M_3M_2M_1$
ESPERA	(NUL)	0 0 0 0 0 0 0	(m_0)	1 0 1 1 1 0 1 0 1 0 1
INICIO DE TRAS.	(STX)	0 0 0 0 0 1 0	(m_2)	1 1 1 0 1 0 1 1 1 0 1 0 1 1 1
FIN DE TRAS.	(ETX)	0 0 0 0 0 1 1	(m_3)	1 0 1 1 1 0 1 0 1 1 1 0 1
ERROR	(CAN)	0 0 1 1 0 0 0	(m_{24})	1 0 1 0 1 0 1 0 1 0 1 0 1
ESPACIO	(SP)	0 1 0 0 0 0 0	(m_{32})	0 0 0 0 0*
APOSTROFO	(')	0 1 0 0 1 1 1	(m_{39})	1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1
COMA	(,)	0 1 0 1 1 0 0	(m_{44})	1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1
GUION	(-)	0 1 0 1 1 0 1	(m_{45})	1 1 1 0 1 0 1 0 1 0 1 0 1 1 1
PUNTO	(.)	0 1 0 1 1 1 0	(m_{46})	1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1
0		0 1 1 0 0 0 0	(m_{48})	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
1		0 1 1 0 0 0 1	(m_{49})	1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
2		0 1 1 0 0 1 0	(m_{50})	1 0 1 0 1 1 1 0 1 1 1 0 1 1 1
3		0 1 1 0 0 1 1	(m_{51})	1 0 1 0 1 1 1 0 1 1 1
4		0 1 1 0 1 0 0	(m_{52})	1 0 1 0 1 0 1 0 1 1 1
5		0 1 1 0 1 0 1	(m_{53})	1 0 1 0 1 0 1 0 1
6		0 1 1 0 1 1 0	(m_{54})	1 1 1 0 1 0 1 0 1 0 1
7		0 1 1 0 1 1 1	(m_{55})	1 1 1 0 1 1 1 0 1 0 1 0 1
8		0 1 1 1 0 0 0	(m_{56})	1 1 1 0 1 1 1 0 1 1 1 0 1 0 1
9		0 1 1 1 0 0 1	(m_{57})	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
DOS PUNTOS	(:)	0 1 1 1 0 1 0	(m_{58})	1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1
PUNTO Y COMA	(;)	0 1 1 1 0 1 1	(m_{59})	1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1
IGUAL	(=)	0 1 1 1 1 0 1	(m_{61})	1 1 1 0 1 0 1 0 1 0 1 1 1
INTERROGACION	(?)	0 1 1 1 1 1 1	(m_{63})	1 0 1 0 1 1 1 0 1 1 1 0 1 0 1

FIG. 5.4 A EQUIVALENCIA ENTRE LOS CODIGOS "ASCII" Y "MORSE".

	"ASCII"								"M O R S E"																			
	A_7	A_6	A_5	A_4	A_3	A_2	A_1		M_{19}	M_{18}	M_{17}	M_{16}	M_{15}	M_{14}	M_{13}	M_{12}	M_{11}	M_{10}	M_9	M_8	M_7	M_6	M_5	M_4	M_3	M_2	M_1
A	1	0	0	0	0	0	1	(m_{65})															1	0	1	1	1
B	1	0	0	0	0	1	0	(m_{66})											1	1	1	0	1	0	1	0	1
C	1	0	0	0	0	1	1	(m_{67})								1	1		1	0	1	0	1	1	1	0	1
D	1	0	0	0	1	0	0	(m_{68})													1	1	1	0	1	0	1
E	1	0	0	0	1	0	1	(m_{69})																			1
F	1	0	0	0	1	1	0	(m_{70})											1	0	1	0	1	1	1	0	1
G	1	0	0	0	1	1	1	(m_{71})											1	1	1	0	1	1	1	0	1
H	1	0	0	1	0	0	0	(m_{72})													1	0	1	0	1	0	1
I	1	0	0	1	0	0	1	(m_{73})																	1	0	1
J	1	0	0	1	0	1	0	(m_{74})						1	0	1	1	1	0	1	1	1	0	1	1	1	
K	1	0	0	1	0	1	1	(m_{75})											1	1	1	0	1	0	1	1	1
L	1	0	0	1	1	0	0	(m_{76})											1	0	1	1	1	0	1	0	1
M	1	0	0	1	1	0	1	(m_{77})													1	1	1	0	1	1	1
N	1	0	0	1	1	1	0	(m_{78})															1	1	1	0	1
O	1	0	0	1	1	1	1	(m_{79})								1	1		1	0	1	1	1	0	1	1	1
P	1	0	1	0	0	0	0	(m_{80})								1	0		1	1	1	0	1	1	1	0	1
Q	1	0	1	0	0	0	1	(m_{81})						1	1	1	0		1	1	1	0	1	0	1	1	1
R	1	0	1	0	0	1	0	(m_{82})													1	0	1	1	1	0	1
S	1	0	1	0	0	1	1	(m_{83})															1	0	1	0	1
T	1	0	1	0	1	0	0	(m_{84})																	1	1	1
U	1	0	1	0	1	0	1	(m_{85})													1	0	1	0	1	1	1
V	1	0	1	0	1	1	0	(m_{86})											1	0	1	0	1	0	1	1	1
W	1	0	1	0	1	1	1	(m_{87})											1	0	1	1	1	0	1	1	1
X	1	0	1	1	0	0	0	(m_{88})								1	1		1	0	1	0	1	0	1	1	1
Y	1	0	1	1	0	0	1	(m_{89})						1	1	1	0		1	0	1	1	1	0	1	1	1
Z	1	0	1	1	0	1	0	(m_{90})								1	1		1	0	1	1	1	0	1	0	1

FIG. 5.4 B (CONTINUACION) EQUIVALENCIA ENTRE LOS CODIGOS "ASCII" Y "MORSE"

	M_{18}	M_{16}	M_{14}	M_{12}	M_{10}	M_8	M_6	M_4	M_2	L_5	L_4	L_3	L_2	L_1	MBI
m_o					0	1	0	0	0	0	1	0	1	1	1
m_2			1	0	0	1	0	0	1	0	1	1	1	1	1
m_3				0	1	0	0	1	0	0	1	1	0	1	1
m_{24}			0	0	0	0	0	0	0	0	1	1	1	1	1
m_{32}								0	0	0	0	1	0	1	0
m_{39}	0	1	0	1	0	1	0	1	0	1	0	0	1	1	1
m_{44}	1	0	1	0	0	0	1	0	1	1	0	0	1	1	1
m_{45}			1	0	0	0	0	0	1	0	1	1	1	1	1
m_{46}		0	1	0	0	1	0	0	1	1	0	0	0	1	1
m_{48}	1	0	1	0	1	0	1	0	1	1	0	0	1	1	1
m_{49}		0	1	0	1	0	1	0	1	1	0	0	0	1	1
m_{50}			0	0	1	0	1	0	1	0	1	1	1	1	1
m_{51}				0	0	0	1	0	1	0	1	1	0	1	1
m_{52}					0	0	0	0	1	0	1	0	1	1	1
m_{53}						0	0	0	0	0	1	0	0	1	1
m_{54}					1	0	0	0	0	0	1	0	1	1	1
m_{55}				1	0	1	0	0	0	0	1	1	0	1	1
m_{56}			1	0	1	0	1	0	0	0	1	1	1	1	1
m_{57}		1	0	1	0	1	0	1	0	1	0	0	0	1	1
m_{58}		1	0	1	0	1	0	0	0	1	0	0	0	1	1
m_{59}		1	0	0	1	0	0	1	0	1	0	0	0	1	1
m_{61}				1	0	0	0	0	1	0	1	1	0	1	1
m_{63}			0	0	1	0	1	0	0	0	1	1	1	1	1

Fig.5.5A Equivalencia entre los códigos ASCII y MORSE tomando en consideración la longitud del Código Morse.

	M_{18}	M_{16}	M_{14}	M_{12}	M_{10}	M_8	M_6	M_4	M_2	L_5	L_4	L_3	L_2	L_1	MBI
m_{65}							0	1		0	0	1	0	1	1
m_{66}						1	0	0	0	0	1	0	0	1	1
m_{67}					1	0	0	1	0	0	1	0	1	1	1
m_{68}							1	0	0	0	0	1	1	1	1
m_{69}										0	0	0	0	1	1
m_{70}						0	0	1	0	0	1	0	0	1	1
m_{71}						1	0	1	0	0	1	0	0	1	1
m_{72}							0	0	0	0	0	1	1	1	1
m_{73}									0	0	0	0	1	1	1
m_{74}				0	1	0	1	0	1	0	1	1	0	1	1
m_{75}						1	0	0	1	0	1	0	0	1	1
m_{76}						0	1	0	0	0	1	0	0	1	1
m_{77}							1	0	1	0	0	1	1	1	1
m_{78}								1	0	0	0	1	0	1	1
m_{79}					1	0	1	0	1	0	1	0	1	1	1
m_{80}					0	1	0	1	0	0	1	0	1	1	1
m_{81}				1	0	1	0	0	1	0	1	1	0	1	1
m_{82}							0	1	0	0	0	1	1	1	1
m_{83}								0	0	0	0	1	0	1	1
m_{84}									1	0	0	0	1	1	1
m_{85}							0	0	1	0	0	1	1	1	1
m_{86}						0	0	0	1	0	1	0	0	1	1
m_{87}						0	1	0	1	0	1	0	0	1	1
m_{88}					1	0	0	0	1	0	1	0	1	1	1
m_{89}				1	0	0	1	0	1	0	1	1	0	1	1
m_{90}					1	0	1	0	0	0	1	0	1	1	1

Fig. 5.5.B (Continuación) Equivalencia entre los códigos ASCII y Morse tomando en consideración la longitud del código Morse.

salida corresponden a dicha palabra MORSE en particular.

El número de funciones de salidas presentadas en la tabla de la Fig.5.5A y B es de "15", mientras que el número de variables de entrada es de "7", por lo que para encontrar las ecuaciones lógicas que determinan el comportamiento de las salidas en función de las entradas, se debería recurrir al método de Quine-McKlusky y hacer uso de una computadora para encontrar el conjunto óptimo de productos en cada caso, sin embargo aquí la minimización se hizo por el método gráfico de los mapas de Karnaugh, para mostrar mas claramente, como en el caso de los FPLA'S la reducción de el número de productos no está condicionada a que el número de variables que forma cada producto sea mínimo también, como sería en el caso de circuitos implementados por ejemplo, con elementos "TTL" (compuertas, inversores, etc.).

Durante la minimización deberá tomarse en cuenta que son necesarios dos FPLA's para obtener todas las salidas (con FLAS de 8 salidas), por lo que se deberán hacer dos grupos, buscando que en cada grupo queden las funciones de salidas cuyos productos sean mas afines y además que el número de ellos no rebase los 48 que pueden ser usados en cada uno de los FPLA's. Si dejamos a $M_2, M_4, M_6, M_8, M_{10}, M_{12}, M_{14}$ y M_{16} en el primer grupo y a $M_{18}, L_1, L_2, L_3, L_4, L_5$ y

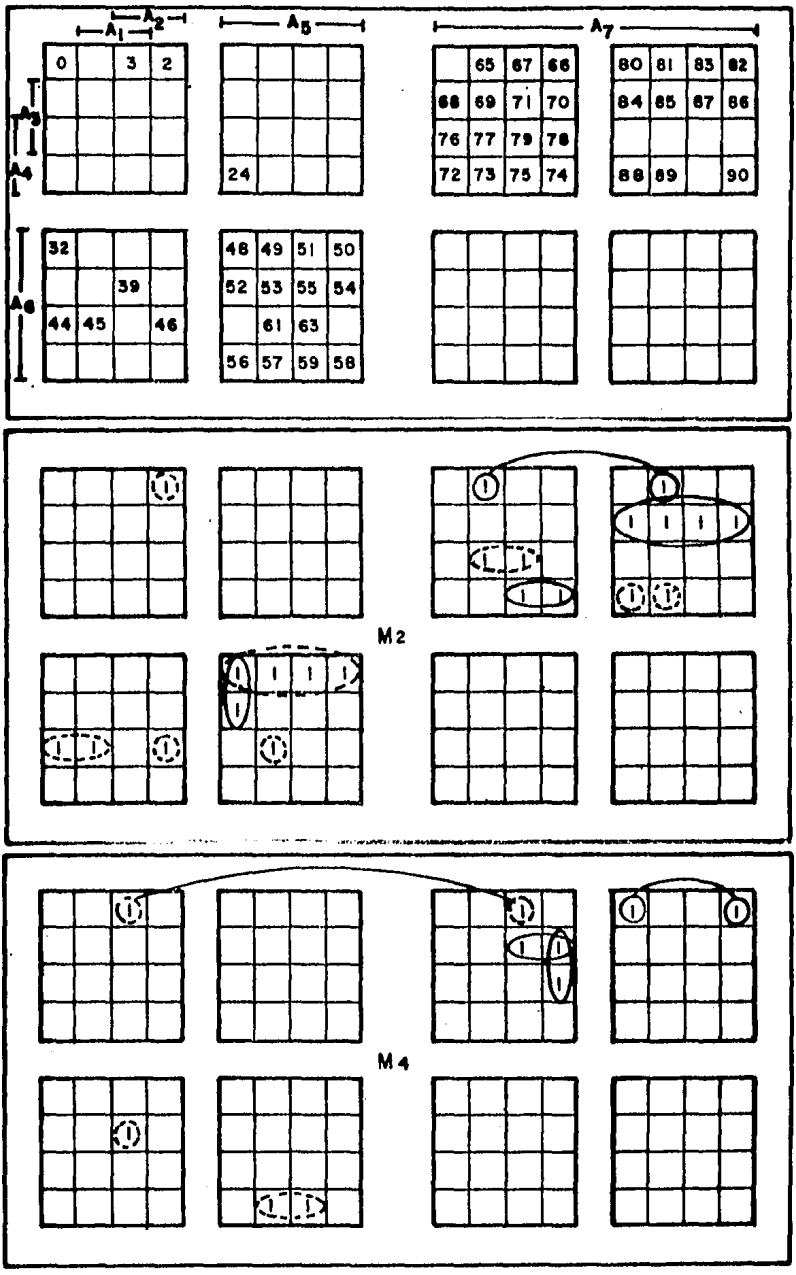


FIG. 5.6A MAPA PATRON DE KARNAUGH, DE M_2 Y M_4.

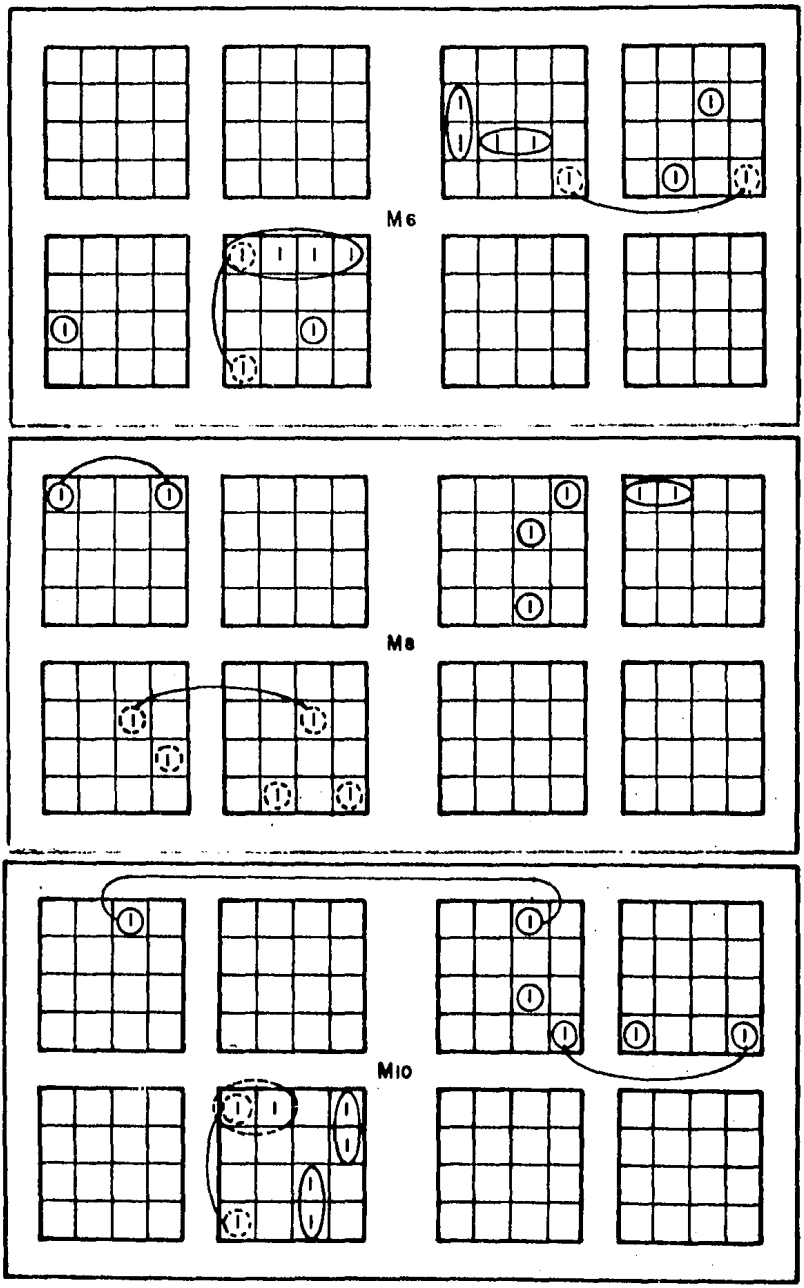


FIG. 5.6B (CONTINUACION) MAPAS "K" DE M_6, M_8 Y M_{10}.

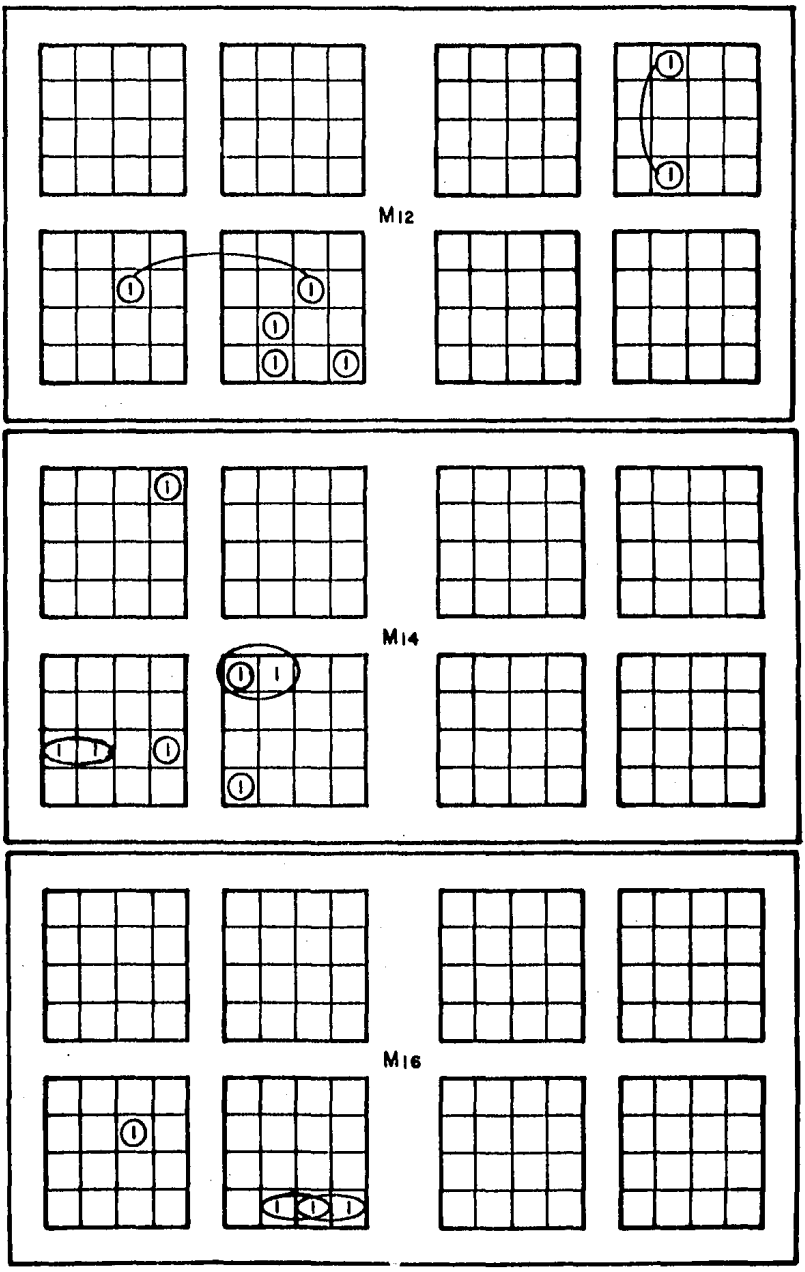


FIG. 5.6C (CONTINUACION) MAPAS "K" DE M_{12},M_{14} Y M_{16}.

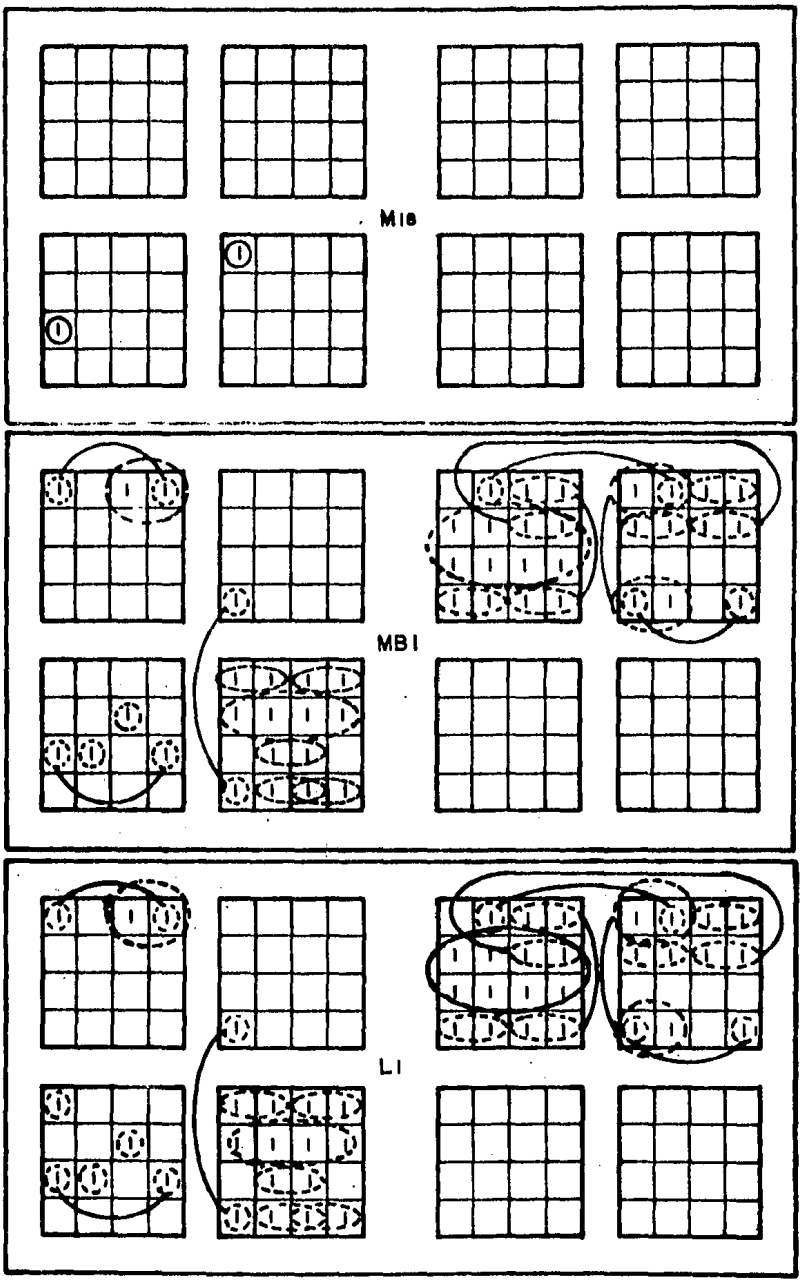


FIG. 5.6D (CONTINUACION) MAPAS "K" DE M_{18}, MBI Y L_1.

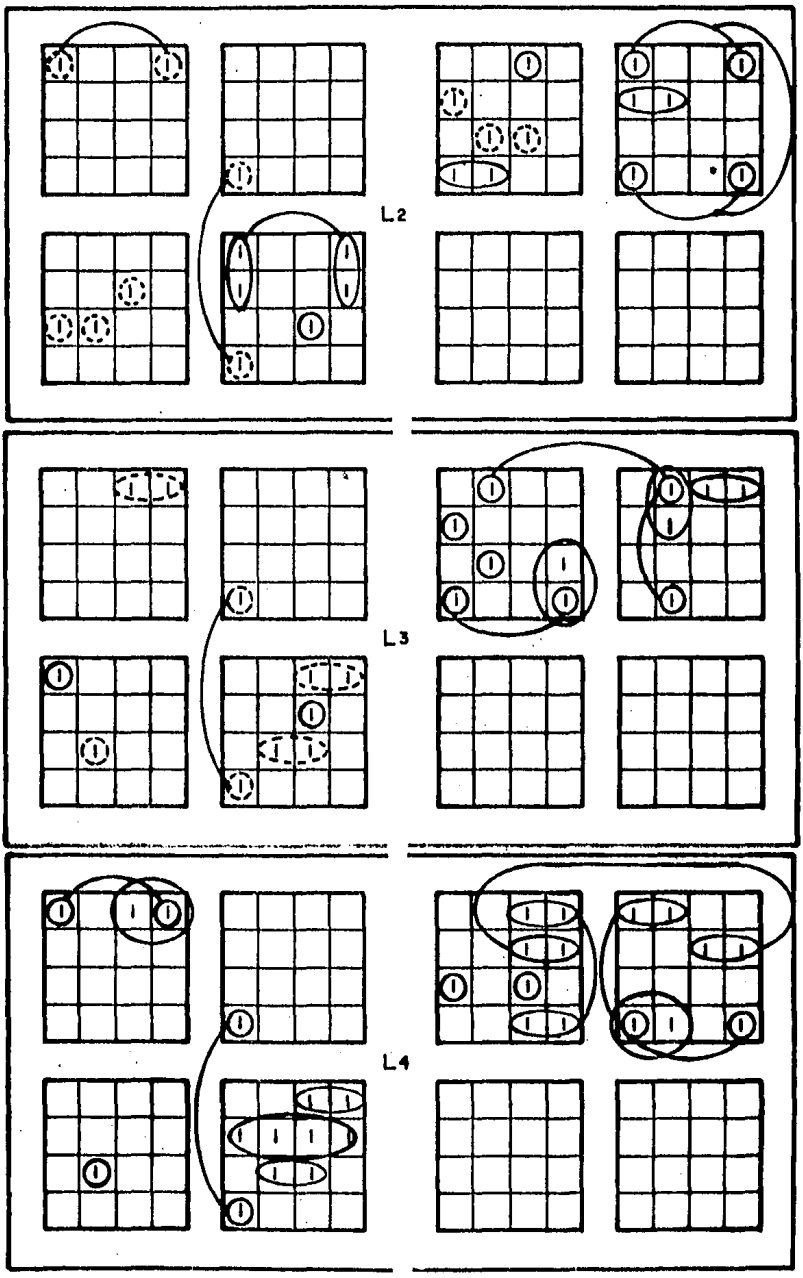


FIG. 5.6E (CONTINUACION) MAPAS "K" DE L_2, L_3 Y L_4.

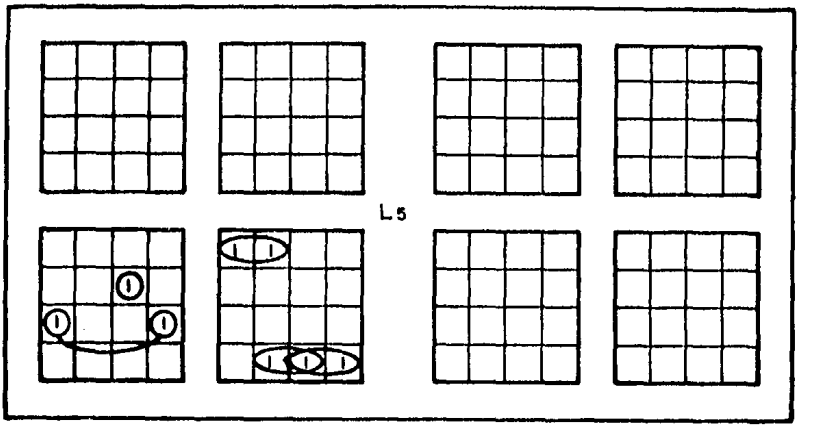


FIG. 5.6F (CONTINUACION) MAPA "K" DE L_5.

MBI en el segundo grupo, encontraremos que es posible lograr una minimización que nos permita utilizar dos FPLA's sin exceder su capacidad de memoria.

Una vez que se han llenado los mapas "K" de cada uno de los grupos, a partir de la información de las tablas de la Fig. 5.5 A y B, primero deben ser seleccionados aquellos productos que contengan a todas las variables en un estado cierto o falso, es decir, los mintérminos que no pueden ser combinados con otros mintérminos, dando lugar a minimizaciones. Después que un producto se a seleccionado en uno de los mapas del grupo (marcados con línea llena), se puede considerar dentro de los otros mapas (marcados con línea punteada) sin que esto incremente el uso de memoria. La selección de productos que no dependen de todas las variables, deberá ser posterior, hasta que por último solo queden aquellos que dependan de una sola variable (si los hubiera).

En los mapas de las Fig. 5.6 A, B, C, D, E y F, se encuentran las presentaciónes mínimas en forma -

global y no individual, es decir, en un mapa en particular el número de agrupaciones puede no ser el mínimo, sin embargo, el número total de éstas se -- puede ver reducido al utilizarse las mismas agrupaciones en diferentes mapas, tal situación es mas -- apreciable en el mapa de Karnaugh correspondiente a la función de salida "L_1"donde las agrupaciones son hechas en base a las realizadas en los mapas posteriores, e incluso, un mintérmino es agrupado mas de una sola vez, cosa que no se hace normalmente, pero aquí el beneficio obtenido es que la función "L_1" se logra, incrementando el número total de productos diferentes, en solo uno.

En seguida se muestra una lista de los diferentes productos obtenidos en los mapas, correspondientes a cada una de las agrupaciones hechas y ordenados en forma binaria empezando por aquellos que no contienen "Don't Care".

$P_1 = \bar{A}_7 \bar{A}_6 \bar{A}_5 \bar{A}_4 \bar{A}_3 A_2 \bar{A}_1$

$P_2 = \bar{A}_7 A_6 \bar{A}_5 \bar{A}_4 \bar{A}_3 \bar{A}_2 \bar{A}_1$

$P_3 = \bar{A}_7 A_6 \bar{A}_5 \bar{A}_4 A_3 A_2 A_1$

$P_4 = \bar{A}_7 A_6 \bar{A}_5 A_4 A_3 \bar{A}_2 \bar{A}_1$

$P_5 = \bar{A}_7 A_6 \bar{A}_5 A_4 A_3 \bar{A}_2 A_1$

$P_6 = \bar{A}_7 A_6 \bar{A}_5 A_4 A_3 A_2 \bar{A}_1$

$P_7 = \bar{A}_7 A_6 A_5 \bar{A}_4 \bar{A}_3 \bar{A}_2 \bar{A}_1$

$P_8 = \bar{A}_7 A_6 A_5 \bar{A}_4 \bar{A}_3 A_2 A_1$

$P_9 = \bar{A}_7 A_6 A_5 A_4 \bar{A}_3 \bar{A}_2 A_1$

$P_{10} = \bar{A}_7 A_6 A_5 A_4 \bar{A}_3 A_2 \bar{A}_1$

$P_{11} = \bar{A}_7 A_6 A_5 A_4 A_3 \bar{A}_2 A_1$

$P_{12} = \bar{A}_7 A_6 A_5 A_4 A_3 A_2 A_1$

$P_{13} = A_7\ \bar{A}_6\ \bar{A}_5\ \bar{A}_4\ \bar{A}_3\ A_2\ \bar{A}_1$

$P_{14} = A_7\ \bar{A}_6\ \bar{A}_5\ \bar{A}_4\ \bar{A}_3\ A_2\ A_1$

$P_{15} = A_7\ \bar{A}_6\ \bar{A}_5\ \bar{A}_4\ A_3\ \bar{A}_2\ \bar{A}_1$

$P_{16} = A_7\ \bar{A}_6\ \bar{A}_5\ \bar{A}_4\ A_3\ A_2\ A_1$

$P_{17} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ \bar{A}_3\ A_2\ A_1$

$P_{18} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ A_3\ \bar{A}_2\ \bar{A}_1$

$P_{19} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ A_3\ \bar{A}_2\ A_1$

$P_{20} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ A_3\ A_2\ A_1$

$P_{21} = A_7\ \bar{A}_6\ A_5\ \bar{A}_4\ A_3\ A_2\ A_1$

$P_{22} = A_7\ \bar{A}_6\ A_5\ A_4\ \bar{A}_3\ \bar{A}_2\ \bar{A}_1$

$P_{23} = A_7\ \bar{A}_6\ A_5\ A_4\ \bar{A}_3\ \bar{A}_2\ A_1$

$P_{24} = \bar{A}_7\ \bar{A}_6\ \bar{A}_5\ \bar{A}_4\ \bar{A}_3\ X\ \bar{A}_1$

$P_{25} = \bar{A}_7\ \bar{A}_6\ \bar{A}_5\ \bar{A}_4\ \bar{A}_3\ A_2\ X$

$P_{26} = X\ \bar{A}_6\ \bar{A}_5\ \bar{A}_4\ \bar{A}_3\ A_2\ A_1$

$P_{27} = \bar{A}_7\ X\ A_5\ A_4\ \bar{A}_3\ \bar{A}_2\ \bar{A}_1$

$P_{28} = \bar{A}_7\ A_6\ X\ \bar{A}_4\ A_3\ A_2\ A_1$

$P_{29} = \bar{A}_7\ A_6\ \bar{A}_5\ A_4\ A_3\ \bar{A}_2\ X$

$P_{30} = \bar{A}_7\ A_6\ \bar{A}_5\ A_4\ A_3\ X\ \bar{A}_1$

$P_{31} = \bar{A}_7\ A_6\ A_5\ \bar{A}_4\ \bar{A}_3\ \bar{A}_2\ X$

$P_{32} = \bar{A}_7\ A_6\ A_5\ \bar{A}_4\ X\ \bar{A}_2\ \bar{A}_1$

$P_{33} = \bar{A}_7\ A_6\ A_5\ X\ \bar{A}_3\ \bar{A}_2\ \bar{A}_1$

$P_{34} = \bar{A}_7\ A_6\ A_5\ \bar{A}_4\ \bar{A}_3\ A_2\ X$

$P_{35} = \bar{A}_7\ A_6\ A_5\ \bar{A}_4\ X\ A_2\ \bar{A}_1$

$P_{36} = \bar{A}_7\ A_6\ A_5\ A_4\ \bar{A}_3\ X\ A_1$

$P_{37} = \bar{A}_7\ A_6\ A_5\ A_4\ \bar{A}_3\ A_2\ X$

$P_{38} = \bar{A}_7\ A_6\ A_5\ A_4\ X\ A_2\ A_1$

$P_{39} = \bar{A}_7\ A_6\ A_5\ A_4\ A_3\ X\ A_1$

$P_{40} = A_7\ \bar{A}_6\ X\ \bar{A}_4\ \bar{A}_3\ \bar{A}_2\ A_1$

$P_{41} = A_7\ \bar{A}_6\ \bar{A}_5\ X\ A_3\ \bar{A}_2\ \bar{A}_1$

$P_{42} = A_7\ \bar{A}_6\ \bar{A}_5\ \bar{A}_4\ A_3\ A_2\ X$

$P_{43} = A_7\ \bar{A}_6\ \bar{A}_5\ X\ A_3\ A_2\ \bar{A}_1$

$P_{44} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ \bar{A}_3\ \bar{A}_2\ X$

$P_{45} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ \bar{A}_3\ X\ \bar{A}_1$

$P_{46} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ \bar{A}_3\ A_2\ X$

$P_{47} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ X\ A_2\ \bar{A}_1$

$P_{48} = A_7\ \bar{A}_6\ X\ A_4\ \bar{A}_3\ A_2\ \bar{A}_1$

$P_{49} = A_7\ \bar{A}_6\ \bar{A}_5\ A_4\ A_3\ X\ A_1$

$P_{50} = A_7\ \bar{A}_6\ A_5\ \bar{A}_4\ \bar{A}_3\ \bar{A}_2\ X$

$P_{51} = A_7\ \bar{A}_6\ A_5\ \bar{A}_4\ \bar{A}_3\ X\ \bar{A}_1$

$P_{52} = A_7\ \bar{A}_6\ A_5\ \bar{A}_4\ X\ \bar{A}_2\ A_1$

$P_{53} = A_7\ \bar{A}_6\ A_5\ X\ \bar{A}_3\ \bar{A}_2\ A_1$

$P_{54} = A_7\ \bar{A}_6\ A_5\ \bar{A}_4\ \bar{A}_3\ A_2\ X$

$P_{55} = A_7\ \bar{A}_6\ A_5\ \bar{A}_4\ A_3\ \bar{A}_2\ X$

$P_{56} = A_7\ \bar{A}_6\ A_5\ A_4\ \bar{A}_3\ X\ \bar{A}_1$

$P_{57} = \bar{A}_7\ A_6\ A_5\ \bar{A}_4\ \bar{A}_3\ X\ X$

$P_{58} = \bar{A}_7\ A_6\ A_5\ \bar{A}_4\ X\ X\ \bar{A}_1$

$P_{59} = \bar{A}_7\ A_6\ A_5\ \bar{A}_4\ A_3\ X\ X$

$P_{60} = A_7\ \bar{A}_6\ \bar{A}_5\ X\ \bar{A}_3\ A_2\ X$

$P_{61} = A_7\ \bar{A}_6\ X\ \bar{A}_4\ A_3\ A_2\ X$

$P_{62} = A_7\ \bar{A}_6\ A_5\ X\ \bar{A}_3\ \bar{A}_2\ X$

$P_{63} = A_7\ \bar{A}_6\ A_5\ X\ \bar{A}_3\ X\ \bar{A}_1$

$P_{64} = A_7\ \bar{A}_6\ A_5\ \bar{A}_4\ A_3\ X\ X$

$P_{65} = A_7\ \bar{A}_6\ \bar{A}_5\ X\ A_3\ X\ X$

De la lista anterior se puede concluir que el número total de productos diferentes es de 65, sin - embargo, solo 38 de éstos son utilizados en uno de los FPLA's y otros 37 diferentes son utilizados en el otro FPLA, es decir, en ninguno de lo dos se -- excede el máximo permitido, que es de 48.
Las ecuaciones obtenidas a partir de los mapas de - Karnaugth son las siguientes:

FPLA 1

$M_2 = P_1+P_6+P_{11}+P_{22}+P_{23}+P_{29}+P_{32}+P_{40}+P_{46}+P_{49}+P_{57}+P_{64}$

$M_4 = P_3+P_{26}+P_{36}+P_{42}+P_{43}+P_{51}$

$M_6 = P_4+P_{12}+P_{21}+P_{23}+P_{33}+P_{41}+P_{48}+P_{49}+P_{57}$

$M_8 = P_6+P_9+P_{10}+P_{13}+P_{16}+P_{17}+P_{24}+P_{28}+P_{50}$

$M_{10} = P_{20}+P_{22}+P_{26}+P_{31}+P_{33}+P_{35}+P_{38}+P_{48}$

$M_{12} = P_9+P_{10}+P_{11}+P_{28}+P_{53}$

$M_{14} = P_1+P_6+P_{29}+P_{31}+P_{33}$

$M_{16} = P_3+P_{36}+P_{37}$

FPLA 2.

$M_{18} = P_4 + P_7 \cdot$

$MBI = P_3 + P_5 + P_{24} + P_{25} + P_{27} + P_{30} + P_{31} + P_{34} + P_{36} + P_{37} + P_{39} + P_{40} + P_{44} + P_{54} + P_{55} + P_{56} + P_{59} + P_{60} + P_{61} + P_{62} + P_{65} \cdot$

$L_1 = P_2 + P_3 + P_5 + P_{24} + P_{25} + P_{27} + P_{30} + P_{31} + P_{34} + P_{36} + P_{37} + P_{39} + P_{40} + P_{44} + P_{54} + P_{55} + P_{56} + P_{59} + P_{60} + P_{61} + P_{62} + P_{65} \cdot$

$L_2 = P_3 + P_4 + P_5 + P_{12} + P_{14} + P_{15} + P_{19} + P_{20} + P_{24} + P_{27} + P_{44} + P_{55} + P_{58} + P_{63} \cdot$

$L_3 = P_2 + P_5 + P_8 + P_{15} + P_{19} + P_{25} + P_{27} + P_{34} + P_{39} + P_{40} + P_{45} + P_{47} + P_{52} + P_{53} + P_{54} \cdot$

$L_4 = P_5 + P_{18} + P_{20} + P_{24} + P_{25} + P_{27} + P_{34} + P_{39} + P_{56} + P_{59} + P_{60} + P_{61} + P_{62} \cdot$

$L_5 = P_3 + P_{30} + P_{31} + P_{36} + P_{37} \cdot$

En las tablas de programa de las Fig. 5.7 y 5.8 se vació el contenido de las ecuaciones lógicas, en donde, las salidas 7, 6, 5, 4, 3, 2, 1 y 0 de la primera tabla corresponden a M_{16}, M_{14}, M_{12}, M_{10}, M_8, M_6, M_4, M_2, respectivamente como verificada altas y sus entradas 7, 6, 5, 4, 3, 2 y 1 corresponden a A_7, A_6, A_5, A_4, A_3, A_2, A_1, respectivamente; la entrada "0" se destinó para "RESET" como una variable más, en cada uno de los productos que son reconocidos por el FPLA, de manera que, éste podrá reconocer un producto dado, dependiendo de si "RESET" toma un nivel alto, ya que en la tabla aparece como "H" en todos los casos y ningún producto lo contiene con un nivel "L".

En la segunda tabla tenemos que las salidas 7, 6, 5, 4, 3, 2, 1 y 0 corresponden a MBI, MBI, M_{18}, L_5, L_4, L_3, L_2, L_1 respectivamente como verificadas altas y donde debe notarse que MBI está presente en dos de las salidas (para aumentar el "FAN OUT" de ésta). Las entradas 7, 6, 5, 4, 3, 2, 1 y 0 de nuevo corresponden a A_7, A_6, A_5, A_4, A_3, A_2, A_1, y R respectivamente. Aquí "R" también representa la entrada "RESET" y para poder permitir que el FPLA reconozca una entrada tendrá que tener un nivel alto, ya que en caso contrario, todas sus salidas tomarán un nivel bajo, dado que ninguna de las palabras que puede reconocer el FPLA contiene a "R" en un nivel bajo.

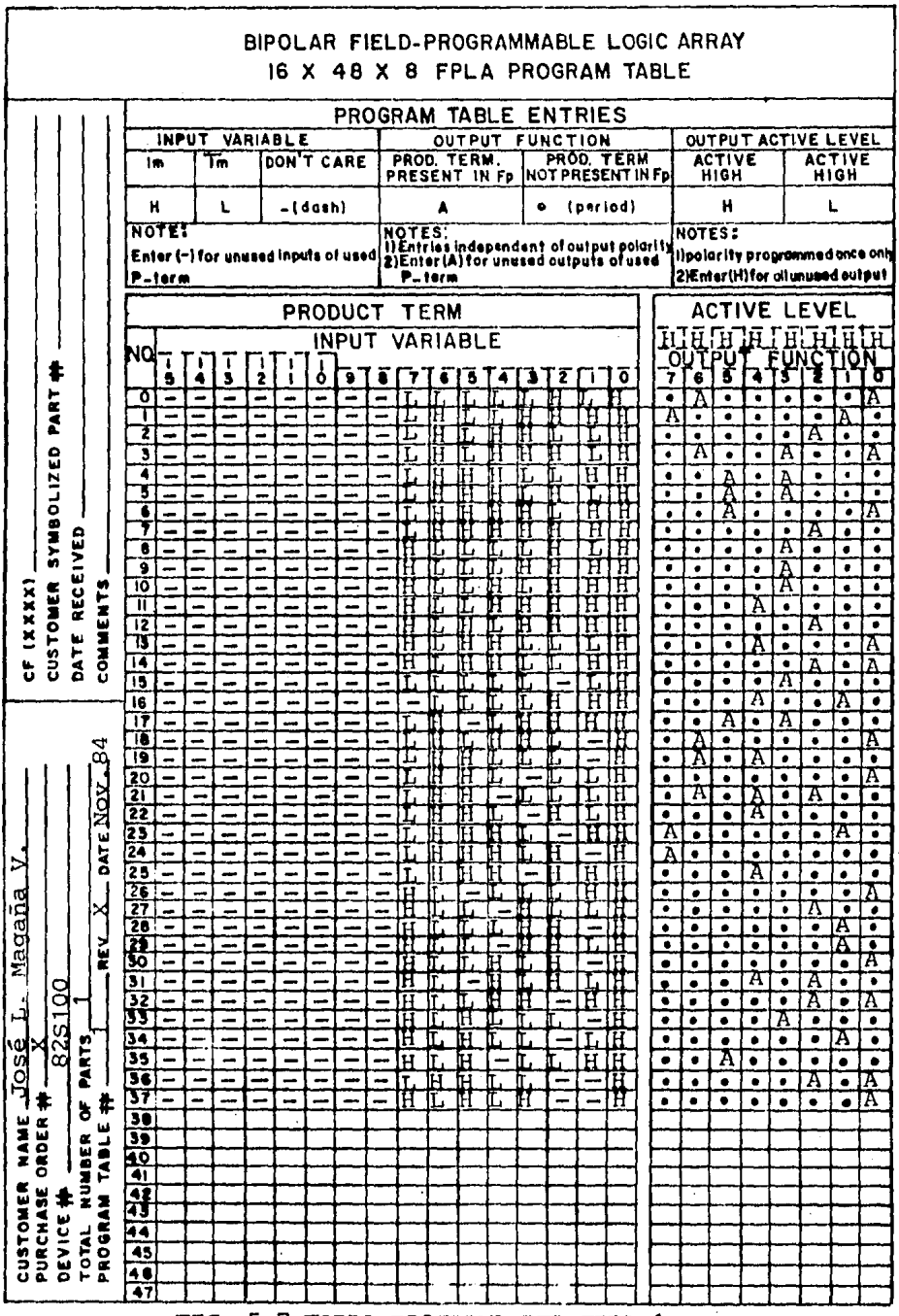


FIG. 5.7 TABLA PROGRAMA DEL FPLA 1.

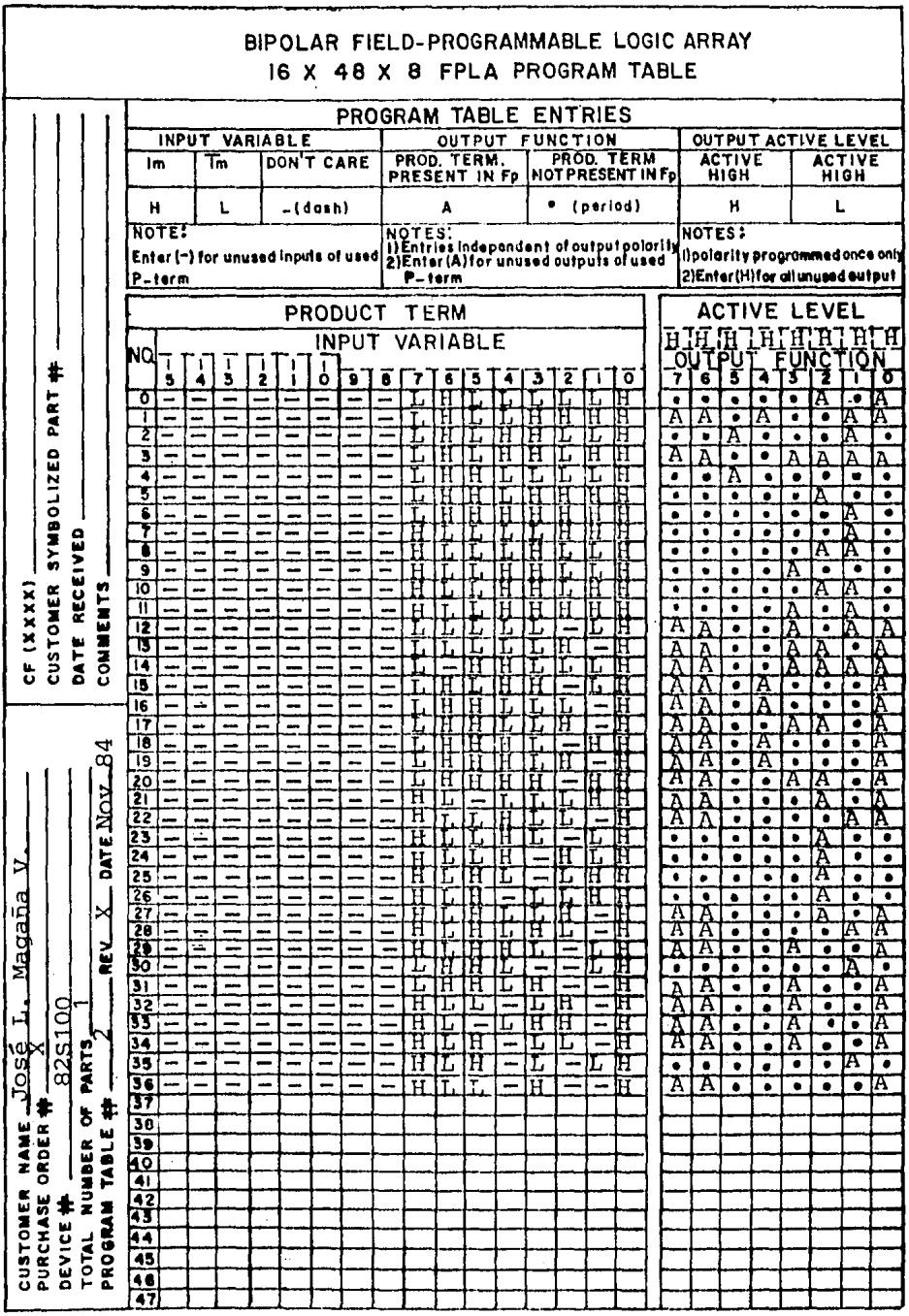


FIG. 5.8 TABLA PROGRAMA DEL FPLA 2.

En la Fig. 5.9 se muestra la manera en que se deberán interconectar los FPLA's 1 y 2, y como son obtenidos los BIT impares de la palabra Morse, haciendo posible la conversión de código ASCII a MORSE.

Con los "BUFFER's" se logra aumentar el número de salidas del FPLA 2 de 8 a 16, aunque el precio que se paga con este arreglo, es un mayor tiempo de propagación. Se pusieron del tipo tri-state, para -- poder cumplir con las especificaciones de diseño.

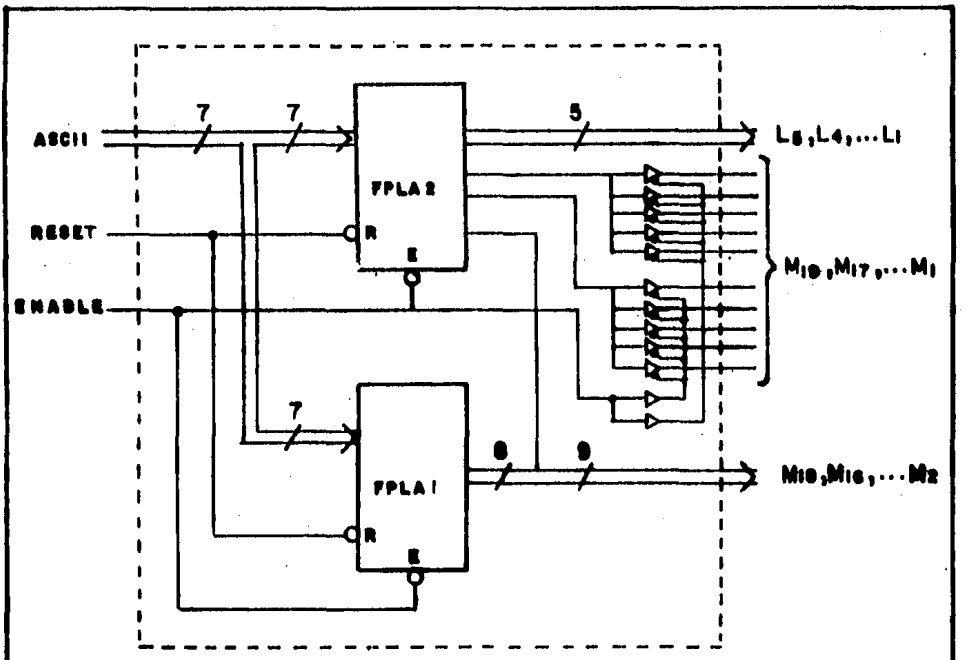


FIG. 5.9 ETAPA DEL CONVERTIDOR DE CODIGO DESTINADA A REALIZAR LA CONVERSION DE CODIGO ASCII A MORSE.

Ya quedó establecido, como es que el circuito va a ser la conversión de ASCII A Morse, por lo que -- ahora se especificará como se hará la conversión en sentido inverso.

Para hacer la conversión de código Morse a código ASCII, paresería suficiente examinar solo aque-- llos Bit's pares de la palabra Morse indicados por la longitud de esta misma, ya que los Bit's impares son siempre "1" (excepto en un caso), sin embargo, po-- dría presentarse por error que alguno de estos no - fuera un "1" y no sería correcto que se realizara - la conversión, si no mas bien, se debe dar por desconocida esta palabra y dar una salida nula. En - consecuencia, para poder ejecutar la conversión, se tendría que contar con un FPLA de 24 entradas (19 - de la palabra y 5 de la longitud de la misma) el -- cual se podría implementar utilizando varios FPLAS de 16 entradas, lo que no sería práctico; otra solución sería, asignar un FPLA de 16 entradas para examinar los bits impares y proporcionar los resulta-- dos a otro FPLA, que además examine los bits pares; para poder hacer ésto, es necesario construir una tabla, tomando en consideración que la longitud de la palabra MORSE es la que determina el número de - bits, que se deben de tomar en cuenta. En la ta-- bla de la Fig. 5.10 se muestran todas aquellas palabras que serán reconocidos por el primer FPLA y también las salidas que le indicarán al segundo FPLA -

$L_5L_4L_3L_2L_1$	$M_{19}M_{17}M_{15}M_{13}M_{11}M_9M_7M_5M_3M_1$	R	A B
0 0 0 0 1	X X X X X X X X X 1	1	1 1
0 0 0 1 1	X X X X X X X X 1 1	1	1 1
0 0 1 0 1	X X X X X X X 0 0 0	1	1 0
0 0 1 0 1	X X X X X X X 1 1 1	1	1 1
0 0 1 1 1	X X X X X X 1 1 1 1	1	1 1
0 1 0 0 1	X X X X X 1 1 1 1 1	1	1 1
0 1 0 1 1	X X X X 1 1 1 1 1 1	1	1 1
0 1 1 0 1	X X X 1 1 1 1 1 1 1	1	1 1
0 1 1 1 1	X X 1 1 1 1 1 1 1 1	1	1 1
1 0 0 0 1	X 1 1 1 1 1 1 1 1 1	1	1 1
1 0 0 1 1	1 1 1 1 1 1 1 1 1 1	1	1 1

FIG. 5.10 PALABRAS QUE DEBERAN SER RECONOCIDAS POR EL FPLA3.

las condiciónes de las entradas impares.

Cuando la longitud de la palabra Morse es de 1 (00001 en binario) el único, dentro de los 10 bit's impares que debe ser tomado en cuenta, es M_1 ya que los otros no forman parte de la palabra. Si la -- longitud de la palabra es de 3 (en la Fig. 5.5 AyB se puede ver que la longitud del código MORSE es siempre impar) los únicos bits impares que se deberán considerar son M_1 y M_3; y asi sucesivamente.

La entrada de "RESET" es incluida aquí y deberá tener un nivel lógico alto, para que el FPLA reco--

nozca la palabra Morse, de no ser así, la salida se hace nula.

Dos salidas son necesarias para indicar cada uno de los tres posibles casos que se pueden presentar: "11" que indica que la palabra Morse es reconocida y además todos los Bit's impares asociados son "1s"; "0X" que indica que la palabra Morse no es reconocida y que al menos uno de los bit's impares asociados es "0"; y 10 que indica que la palabra Morse es reconocida y además todos los bit's impares asociados son "0s" caso que se presenta cuando la palabra Morse corresponde a un "espacio" (00000).

El número de productos que deberá almacenarse en este FPLA es de solo **11**, pues como se puede ver en de la Fig. 5.10, ya no es posible hacer una minimización. Para que esto fuera posible, tendría que haber cuando menos dos productos que presentaran la misma salida y cuando más, fueran diferentes en uno de sus bits, cosa que no sucede.

En la Fig. 5.11 son mostrados cada uno de los productos con su correspondiente número asignado, así como también el mapa de actividades y es a partir de éste, que se llena la tabla de programa de la Fig. 5.12, donde las entradas 15,14,13...0, corresponden a $M_{19}, M_{17} M_{15} \ldots L_5 \ldots L_1$ y R respectivamente. Aquí "R" también corresponde a la entrada de "RESET"; y las salidas 2 y 3 corresponden a B y A respectivamente.

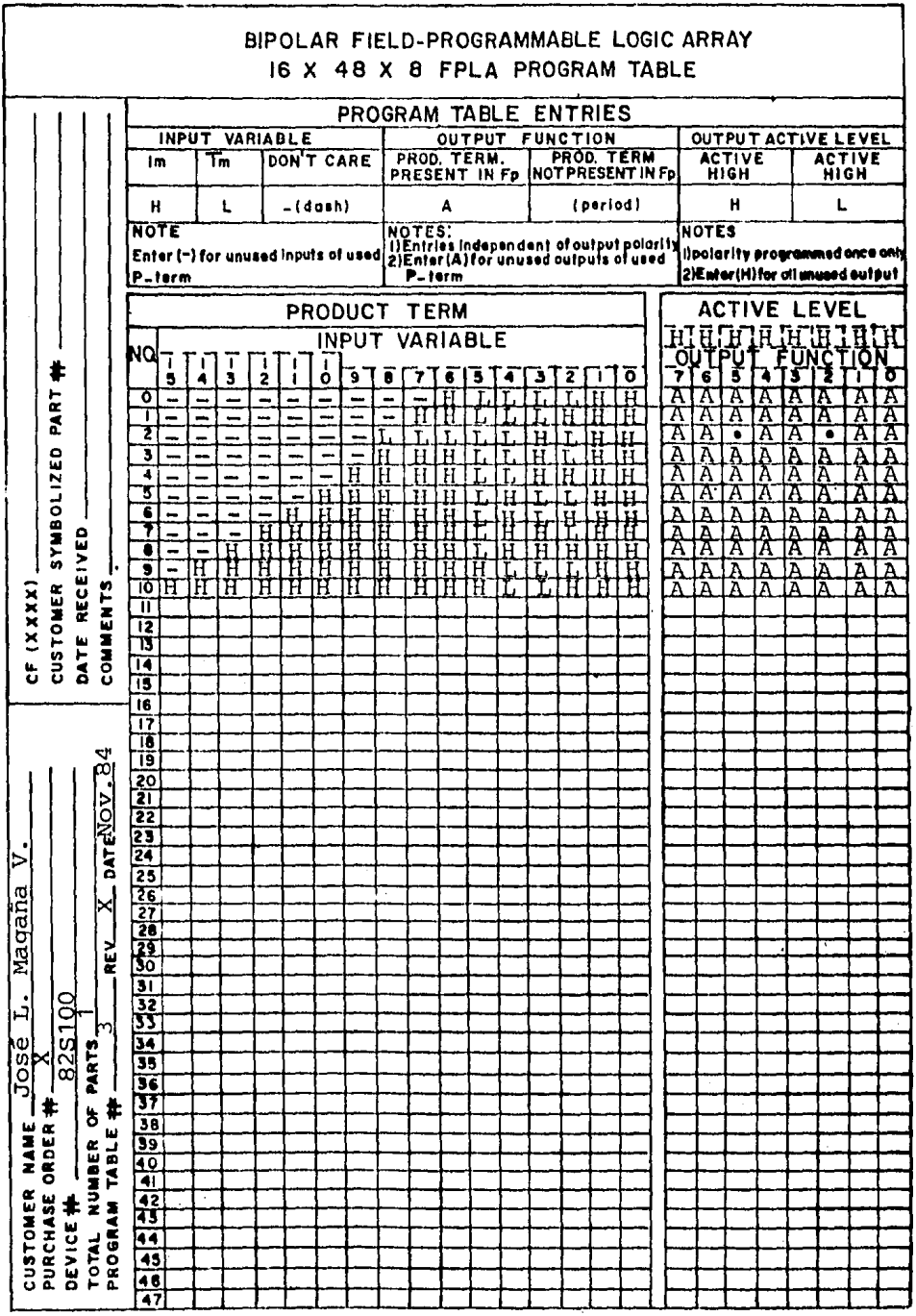
$$
\begin{array}{lcccccccccccccccc}
P_1 & = X & X & X & X & X & X & X & X & X & M_1 & \bar{L}_5 & \bar{L}_4 & \bar{L}_3 & \bar{L}_2 & \bar{L}_1 & R \\
P_2 & = X & X & X & X & X & X & X & X & M_3 & M_1 & \bar{L}_5 & \bar{L}_4 & \bar{L}_3 & L_2 & L_1 & R \\
P_3 & = X & X & X & X & X & X & X & \bar{M}_5 & \bar{M}_3 & \bar{M}_1 & \bar{L}_5 & \bar{L}_4 & L_3 & \bar{L}_2 & L_1 & R \\
P_4 & = X & X & X & X & X & X & X & M_5 & M_3 & M_1 & \bar{L}_5 & \bar{L}_4 & L_3 & \bar{L}_2 & L_1 & R \\
P_5 & = X & X & X & X & X & X & M_7 & M_5 & M_3 & M_1 & \bar{L}_5 & \bar{L}_4 & L_3 & L_2 & L_1 & R \\
P_6 & = X & X & X & X & X & M_9 & M_7 & M_5 & M_3 & M_1 & \bar{L}_5 & L_4 & \bar{L}_3 & \bar{L}_2 & L_1 & R \\
P_7 & = X & X & X & X & M_{11} & M_9 & M_7 & M_5 & M_3 & M_1 & \bar{L}_5 & L_4 & \bar{L}_3 & L_2 & L_1 & R \\
P_8 & = X & X & X & M_{13} & M_{11} & M_9 & M_7 & M_5 & M_3 & M_1 & \bar{L}_5 & L_4 & L_3 & \bar{L}_2 & L_1 & R \\
P_9 & = X & X & M_{15} & M_{13} & M_{11} & M_9 & M_7 & M_5 & M_3 & M_1 & \bar{L}_5 & L_4 & L_3 & L_2 & L_1 & R \\
P_{10} & = X & M_{17} & M_{15} & M_{13} & M_{11} & M_9 & M_7 & M_5 & M_3 & M_1 & L_5 & \bar{L}_4 & \bar{L}_3 & L_2 & L_1 & R \\
P_{11} & = M_{19} & M_{17} & M_{15} & M_{13} & M_{11} & M_9 & M_7 & M_5 & M_3 & M_1 & L_5 & \bar{L}_4 & \bar{L}_3 & L_2 & L_1 & R
\end{array}
$$

$$A = P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8 + P_9 + P_{10} + P_{11}$$

$$B = P_1 + P_2 +/ + P_4 + P_5 + P_6 + P_7 + P_8 + P_9 + P_{10} + P_{11}$$

Fig. 5.11 Productos que deberá almacenar el FPLA3 Y ecuaciones lógicas que representa el mismo.

Haciendo uso de las funciones de salida "A y B" obtenidas del FPLA anterior y tomando en consideración los bit's pares y longitud de la palabra Morse, se puede construir una tabla, Fig. 5.13 A y B misma en la que queda representada la equivalencia entre el código MORSE y ASCII, la cual a sido ordenada a partir de la palabra Morse más pequeña (mayor núme-



BIPOLAR FIELD-PROGRAMMABLE LOGIC ARRAY
16 X 48 X 8 FPLA PROGRAM TABLE

CF (XXXX) _____
CUSTOMER SYMBOLIZED PART # _____
DATE RECEIVED _____
COMMENTS _____

CUSTOMER NAME Jose L. Magaña V.
PURCHASE ORDER # X
DEVICE # 82S100
TOTAL NUMBER OF PARTS 1
PROGRAM TABLE # 3 REV. X DATE NOV. 84

PROGRAM TABLE ENTRIES

INPUT VARIABLE			OUTPUT FUNCTION		OUTPUT ACTIVE LEVEL	
Im	$\overline{Im}$	DON'T CARE	PROD. TERM. PRESENT IN Fp	PROD. TERM NOT PRESENT IN Fp	ACTIVE HIGH	ACTIVE HIGH
H	L	-(dash)	A	(period)	H	L

NOTE
Enter (-) for unused inputs of used P-term

NOTES:
1) Entries independent of output polarity
2) Enter (A) for unused outputs of used P-term

NOTES
1) polarity programmed once only
2) Enter (H) for all unused output

NO	PRODUCT TERM: INPUT VARIABLE I5	I4	I3	I2	I1	I0	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	-	-	-	-	-	-	-	-	-	H	L	L	L	L	H	H	A	A	A	A	A	A	A	A
1	-	-	-	-	-	-	-	-	H	H	L	L	L	H	H	H	A	A	A	A	A	A	A	A
2	-	-	-	-	-	-	-	L	L	L	L	L	H	L	H	H	A	A	•	A	A	•	A	A
3	-	-	-	-	-	-	-	H	H	H	L	L	H	L	H	H	A	A	A	A	A	A	A	A
4	-	-	-	-	-	-	H	H	H	H	L	L	H	H	H	H	A	A	A	A	A	A	A	A
5	-	-	-	-	-	H	H	H	H	H	L	H	L	L	H	H	A	A	A	A	A	A	A	A
6	-	-	-	-	H	H	H	H	H	H	L	H	L	H	H	H	A	A	A	A	A	A	A	A
7	-	-	-	H	H	H	H	H	H	H	L	H	H	L	H	H	A	A	A	A	A	A	A	A
8	-	-	H	H	H	H	H	H	H	H	L	H	H	H	H	H	A	A	A	A	A	A	A	A
9	-	H	H	H	H	H	H	H	H	H	H	L	L	L	H	H	A	A	A	A	A	A	A	A
10	H	H	H	H	H	H	H	H	H	H	H	L	L	H	H	H	A	A	A	A	A	A	A	A
11																								
12																								
13																								
14																								
15																								
16																								
17																								
18																								
19																								
20																								
21																								
22																								
23																								
24																								
25																								
26																								
27																								
28																								
29																								
30																								
31																								
32																								
33																								
34																								
35																								
36																								
37																								
38																								
39																								
40																								
41																								
42																								
43																								
44																								
45																								
46																								
47																								

FIG. 5.12 TABLA PROGRAMA DEL FPLA3.

A	B	L_5	L_4	L_3	L_2	L_1	M_{18}	M_{16}	M_{14}	M_{12}	M_{10}	M_8	M_6	M_4	M_2	A_7	A_6	A_5	A_4	A_3	A_2	A_1
1	1	0	0	0	0	1	X	X	X	X	X	X	X	X	X	1	0	0	0	1	0	1
1	1	0	0	0	1	1	X	X	X	X	X	X	X	X	0	1	0	0	1	0	0	1
1	1	0	0	0	1	1	X	X	X	X	X	X	X	X	1	1	0	1	0	1	0	0
*1	0	0	0	1	0	1	X	X	X	X	X	X	X	0	0	0	1	0	0	0	0	0
*1	1	0	0	1	0	1	X	X	X	X	X	X	X	0	0	1	0	1	0	0	1	1
1	1	0	0	1	0	1	X	X	X	X	X	X	X	0	1	1	0	0	0	0	0	1
1	1	0	0	1	0	1	X	X	X	X	X	X	X	1	0	1	0	0	1	1	1	0
1	1	0	0	1	1	1	X	X	X	X	X	X	0	0	0	1	0	0	1	0	0	0
1	1	0	0	1	1	1	X	X	X	X	X	X	0	0	1	1	0	1	0	1	0	1
1	1	0	0	1	1	1	X	X	X	X	X	X	0	1	0	1	0	1	0	0	1	0
1	1	0	0	1	1	1	X	X	X	X	X	X	0	0	1	1	0	0	0	1	0	0
1	1	0	0	1	1	1	X	X	X	X	X	X	1	0	1	1	0	0	1	1	0	1
1	1	0	1	0	0	1	X	X	X	X	X	0	0	0	0	0	1	1	0	1	0	1
1	1	0	1	0	0	1	X	X	X	X	X	0	0	0	1	1	0	1	0	1	1	0
1	1	0	1	0	0	1	X	X	X	X	X	0	0	1	0	1	0	0	0	1	1	0
1	1	0	1	0	0	1	X	X	X	X	X	0	1	0	0	1	0	0	1	1	0	0
1	1	0	1	0	0	1	X	X	X	X	X	0	1	0	1	1	0	1	0	1	1	1
1	1	0	1	0	0	1	X	X	X	X	X	1	0	0	0	1	0	0	0	0	1	0
1	1	0	1	0	0	1	X	X	X	X	X	1	0	0	1	1	0	0	1	0	1	1
1	1	0	1	0	0	1	X	X	X	X	X	1	0	1	0	1	0	0	0	1	1	1
1	1	0	1	0	1	1	X	X	X	X	0	0	0	0	1	0	1	1	0	1	0	0
1	1	0	1	0	1	1	X	X	X	X	0	1	0	0	0	0	0	0	0	0	0	0*
1	1	0	1	0	1	1	X	X	X	X	0	1	0	1	0	1	0	1	0	0	0	0
1	1	0	1	0	1	1	X	X	X	X	1	0	0	0	0	0	1	1	0	1	1	0
1	1	0	1	C	1	1	X	X	X	X	1	0	0	0	1	1	0	1	1	0	0	0

FIG. 5.13A EQUIVALENCIA ENTRE EL CODIGO MORSE Y EL ASCII.

A B	L L L L L L 5 4 3 2 1	M M M M M M M M M 18 16 14 12 10 8 6 4	A A A A A A A 7 6 5 4 3 2 1
1 1	0 1 0 1 1	X X X X 1 0 0 1 0	1 0 0 0 0 1 1
1 1	0 1 0 1 1	X X X X 1 0 1 0 0	1 0 1 1 0 1 0
1 1	0 1 0 1 1	X X X X 1 0 1 0 1	1 0 0 1 1 1 1
1 1	0 1 1 0 1	X X X 0 0 0 1 0 1	0 1 1 0 0 1 1
1 1	0 1 1 0 1	X X X 0 1 0 0 1 0	0 0 0 0 0 1 1
1 1	0 1 1 0 1	X X X 0 1 0 1 0 1	1 0 0 1 0 1 0
1 1	0 1 1 0 1	X X X 1 0 0 0 0 1	0 1 1 1 1 0 1
1 1	0 1 1 0 1	X X X 1 0 0 1 0 1	1 0 1 1 0 0 1
1 1	0 1 1 0 1	X X X 1 0 1 0 0 0	0 1 1 0 1 1 1
1 1	0 1 1 0 1	X X X 1 0 1 0 0 1	1 0 1 0 0 0 1
1 1	0 1 1 1 1	X X 0 0 0 0 0 0 0	0 0 1 1 0 0 0
1 1	0 1 1 1 1	X X 0 0 1 0 1 0 0	0 1 1 1 1 1 1
1 1	0 1 1 1 1	X X 0 0 1 0 1 0 1	0 1 1 0 0 1 0
1 1	0 1 1 1 1	X X 1 0 0 0 0 0 1	0 1 0 1 1 0 1
1 1	0 1 1 1 1	X X 1 0 0 1 0 0 1	0 0 0 0 0 1 0
1 1	0 1 1 1 1	X X 1 0 1 0 1 0 0	0 1 1 1 0 0 0
1 1	1 0 0 0 1	X 0 1 0 0 1 0 0 1	0 1 0 1 1 1 0
1 1	1 0 0 0 1	X 0 1 0 1 0 1 0 1	0 1 1 0 0 0 1
1 1	1 0 0 0 1	X 1 0 0 1 0 0 1 0	0 1 1 1 0 1 1
1 1	1 0 0 0 1	X 1 0 1 0 1 0 0 0	0 1 1 1 0 1 0
1 1	1 0 0 0 1	X 1 0 1 0 1 0 1 0	0 1 1 1 0 0 1
1 1	1 0 0 1 1	0 1 0 1 0 1 0 1 0	0 1 0 0 1 1 1
1 1	1 0 0 1 1	1 0 1 0 0 0 1 0 1	0 1 0 1 1 0 0
1 1	1 0 0 1 1	1 0 1 0 1 0 1 0 1	0 1 1 0 0 0 0

FIG. 5.13B (CONTINUACION) EQUIVALENCIA ENTRE EL CODIGO MORSE Y CODIGO ASCII.

ro de "DON'T CARE") hasta la de mayor longitud (ningun "DON'T CARE"); la longitud de la palabra MORSE nos indica cuales bit's deben ser considerados de la misma palabra y aquellos que no forman parte de ésta son indicados como "DON'T CARE" pues al no tomarse en cuenta lo mismo dá que sea un "1" ó un "0".

Existe solo un caso, donde no es posible encontrar el código ASCII equivalente, a partir -únicamente- de la longitud y bit's pares de la palabra morse y corresponde precisamente a el "espacio" y a la "S" que son la cuarta y quinta palabra de la lista de la Fig. 5.13A misma donde se puede ver que lo único que las hace diferente es el bit "B" obtenido del primer FPLA.

El número de palabras diferentes enlistadas en la tabla de la Fig. 5.13 A y B, -correspondientes al cuarto FPLA-, es de 49, pero en realidad solo 48 deberán ser consideradas, ya que, para la palabra Morse "espera" las salidas se hacen nulas, pues la palabra ASCII equivalente es 0000000. A partir de éstas, es que se debería proceder a la minimización, acudiendo -como ya se dijo- al método de QUINE-MCKLUSKY por computadora, sin embargo, para los fines aquí perseguidos, el hecho de usar toda la capacidad de la memoria, no nos afecta en lo absoluto.

En la FIg. 5.14 AyB son mostrado cada uno de los

P 1 = X X X X X X X X X $\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P 2 = X X X X X X X X $\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P 3 = X X X X X X X X $M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P 4 = X X X X X X X $\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P 5 = X X X X X X X $\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P 6 = X X X X X X X $\bar{M}_4M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P 7 = X X X X X X X $M_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P 8 = X X X X X X $\bar{M}_6\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P 9 = X X X X X X $\bar{M}_6\bar{M}_4M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P10 = X X X X X X $\bar{M}_6M_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P11 = X X X X X X $M_6\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P12 = X X X X X X $M_6\bar{M}_4M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P13 = X X X X X $\bar{M}_8\bar{M}_6\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P14 = X X X X X $\bar{M}_8\bar{M}_6\bar{M}_4M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P15 = X X X X X $\bar{M}_8\bar{M}_6M_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P16 = X X X X X $\bar{M}_8M_6\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P17 = X X X X X $\bar{M}_8M_6\bar{M}_4M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P18 = X X X X X $M_8\bar{M}_6\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P19 = X X X X X $M_8\bar{M}_6\bar{M}_4M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P20 = X X X X X $M_8\bar{M}_6M_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P21 = X X X X $\bar{M}_{10}\bar{M}_8\bar{M}_6\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P22 = X X X X $\bar{M}_{10}\bar{M}_8\bar{M}_6\bar{M}_4M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P23 = X X X X $M_{10}\bar{M}_8\bar{M}_6\bar{M}_4\bar{M}_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P24 = X X X X $M_{10}\bar{M}_8\bar{M}_6\bar{M}_4M_2\bar{L}_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

FIG. 5, 14 A, PRODUCTOS QUE DEBERAN SER RECONOCIDOS POR EL FPLA 4.

P25 = X X X X $M_{10}\bar{M}_8\bar{M}_6M_4\bar{M}_2\bar{L}_5L_4\bar{L}_3L_2L_1$ B A

P26 = X X X X $M_{10}\bar{M}_8M_6\bar{M}_4\bar{M}_2\bar{L}_5L_4\bar{L}_3L_2L_1$ B A

P27 = X X X X $M_{10}\bar{M}_8M_6\bar{M}_4M_2\bar{L}_5L_4\bar{L}_3L_2L_1$ B A

P28 = X X X $\bar{M}_{12}\bar{M}_{10}\bar{M}_8M_6\bar{M}_4M_2\bar{L}_5L_4L_3\bar{L}_2\bar{L}_1$ B A

P29 = X X X $\bar{M}_{12}M_{10}\bar{M}_8\bar{M}_6M_4\bar{M}_2\bar{L}_5L_4L_3\bar{L}_2\bar{L}_1$ B A

P30 = X X X $\bar{M}_{12}M_{10}\bar{M}_8M_6\bar{M}_4M_2\bar{L}_5L_4L_3\bar{L}_2\bar{L}_1$ B A

P31 = X X X $M_{12}\bar{M}_{10}\bar{M}_8\bar{M}_6\bar{M}_4M_2\bar{L}_5L_4L_3\bar{L}_2\bar{L}_1$ B A

P32 = X X X $M_{12}\bar{M}_{10}\bar{M}_8M_6\bar{M}_4M_2\bar{L}_5L_4L_3\bar{L}_2\bar{L}_1$ B A

P33 = X X X $M_{12}\bar{M}_{10}M_8\bar{M}_6\bar{M}_4\bar{M}_2\bar{L}_5L_4L_3\bar{L}_2\bar{L}_1$ B A

P34 = X X X $M_{12}\bar{M}_{10}M_8\bar{M}_6\bar{M}_4M_2\bar{L}_5L_4L_3\bar{L}_2\bar{L}_1$ B A

P35 = X X $\bar{M}_{14}\bar{M}_{12}\bar{M}_{10}\bar{M}_8\bar{M}_6\bar{M}_4\bar{M}_2\bar{L}_5L_4L_3L_2L_1$ B A

P36 = X X $\bar{M}_{14}\bar{M}_{12}M_{10}\bar{M}_8M_6\bar{M}_4\bar{M}_2\bar{L}_5L_4L_3L_2L_1$ B A

P37 = X X $\bar{M}_{14}\bar{M}_{12}M_{10}\bar{M}_8M_6\bar{M}_4M_2\bar{L}_5L_4L_3L_2L_1$ B A

P38 = X X $M_{14}\bar{M}_{12}\bar{M}_{10}\bar{M}_8\bar{M}_6\bar{M}_4M_2\bar{L}_5L_4L_3L_2L_1$ B A

P39 = X X $M_{14}\bar{M}_{12}M_{10}\bar{M}_8\bar{M}_6\bar{M}_4M_2\bar{L}_5L_4L_3L_2L_1$ B A

P40 = X X $M_{14}\bar{M}_{12}M_{10}\bar{M}_8M_6\bar{M}_4\bar{M}_2\bar{L}_5L_4L_3L_2L_1$ B A

P41 = X $\bar{M}_{16}M_{14}\bar{M}_{12}\bar{M}_{10}M_8\bar{M}_6\bar{M}_4M_2L_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P42 = X $\bar{M}_{16}M_{14}\bar{M}_{12}M_{10}\bar{M}_8M_6\bar{M}_4M_2L_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P43 = X $M_{16}\bar{M}_{14}\bar{M}_{12}M_{10}\bar{M}_8\bar{M}_6M_4\bar{M}_2L_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P44 = X $M_{16}\bar{M}_{14}M_{12}\bar{M}_{10}M_8\bar{M}_6\bar{M}_4\bar{M}_2L_5\bar{L}_4\bar{L}_3\bar{L}_2L_1$ B A

P45 = X $M_{16}\bar{M}_{14}M_{12}\bar{M}_{10}M_8\bar{M}_6M_4\bar{M}_2L_5\bar{L}_4L_3\bar{L}_2L_1$ B A

P46 = $\bar{M}_{18}M_{16}\bar{M}_{14}M_{12}\bar{M}_{10}M_8\bar{M}_6M_4\bar{M}_2L_5\bar{L}_4L_3\bar{L}_2L_1$ B A

P47 = $M_{18}\bar{M}_{16}M_{14}\bar{M}_{12}M_{10}\bar{M}_8M_6\bar{M}_4M_2L_5\bar{L}_4L_3\bar{L}_2L_1$ B A

P48 = $M_{18}\bar{M}_{16}M_{14}\bar{M}_{12}M_{10}\bar{M}_8M_6\bar{M}_4M_2L_5\bar{L}_4\bar{L}_3L_2L_1$ B A

FIG. 5, 14 B, CONTINUACION PRODUCTOS QUE DEBERAN SER RECONOCIDOS POR EL FPLA.

productos con su número correspondiente y basado en éstos, se obtiene el mapa de actividades mostrado en la Fig. 5.15y también la tabla programa mostrada en la Fig. 5.16.

$$A_7 = P_1 + P_2 + P_3 + P_5 + P_6 + P_7 + P_8 + P_9 + P_{10} + P_{11} + P_{12} + P_{14} + P_{15} + P_{16} + P_{17} + P_{18} + P_{19} + P_{20} + P_{23} + P_{25} + P_{26} + P_{27} + P_{28} + P_{31} + P_{33} + P_{35}.$$

$$A_6 = P_4 + P_{13} + P_{21} + P_{24} + P_{29} + P_{32} + P_{34} + P_{37} + P_{38} + P_{39} + P_{41} + P_{42} + P_{43} + P_{44} + P_{45} + P_{46} + P_{47} + P_{48} + P_{49}.$$

$$A_5 = P_3 + P_5 + P_9 + P_{10} + P_{13} + P_{14} + P_{17} + P_{21} + P_{23} + P_{24} + P_{25} + P_{27} + P_{29} + P_{32} + P_{33} + P_{34} + P_{35} + P_{36} + P_{37} + P_{38} + P_{41} + P_{43} + P_{44} + P_{45} + P_{46} + P_{49}.$$

$$A_4 = P_2 + P_7 + P_8 + P_{12} + P_{16} + P_{19} + P_{25} + P_{27} + P_{28} + P_{31} + P_{32} + P_{33} + P_{36} + P_{37} + P_{39} + P_{41} + P_{42} + P_{44} + P_{45} + P_{46} + P_{48}.$$

$$A_3 = P_1 + P_3 + P_7 + P_9 + P_{11} + P_{12} + P_{13} + P_{14} + P_{15} + P_{16} + P_{17} + P_{20} + P_{21} + P_{24} + P_{28} + P_{32} + P_{34} + P_{37} + P_{39} + P_{42} + P_{47} + P_{48}.$$

$$A_2 = P_5 + P_7 + P_{10} + P_{14} + P_{15} + P_{17} + P_{18} + P_{19} + P_{20} + P_{24} + P_{26} + P_{27} + P_{28} + P_{29} + P_{30} + P_{31} + P_{34} + P_{37} + P_{38} + P_{40} + P_{42} + P_{44} + P_{45} + P_{47}.$$

$$A_1 = P_1 + P_2 + P_5 + P_6 + P_9 + P_{12} + P_{13} + P_{17} + P_{19} + P_{20} + P_{26} + P_{28} + P_{29} + P_{30} + P_{32} + P_{33} + P_{34} + P_{35} + P_{37} + P_{39} + P_{43} + P_{44} + P_{46} + P_{47}.$$

FIG. 5.15 ECUACIONES LOGICAS REALIZADAS EN EL FPLA 4.

BIPOLAR FIELD-PROGRAMMABLE LOGIC ARRAY
16 X 48 X 8 FPLA PROGRAM TABLE

CUSTOMER NAME José L. Magaña V.
PURCHASE ORDER # X
DEVICE # 82S100
TOTAL NUMBER OF PARTS 1
PROGRAM TABLE # 4 REV. X DATE Nov. 84

CF (XXXX) _____
CUSTOMER SYMBOLIZED PART # _____
DATE RECEIVED _____
COMMENTS _____

PROGRAM TABLE ENTRIES

INPUT VARIABLE			OUTPUT FUNCTION		OUTPUT ACTIVE LEVEL	
Im	$\overline{Tm}$	DON'T CARE	PROD. TERM. PRESENT IN Fp	PROD. TERM NOT PRESENT IN Fp	ACTIVE HIGH	ACTIVE HIGH
H	L	-(dash)	A	(period)	H	L

NOTE
Enter (-) for unused inputs of used P_term

NOTES:
1) Entries independent of output polarity
2) Enter (A) for unused outputs of used P_term

NOTES
1) polarity programmed once only
2) Enter (H) for all unused output

NO.	PRODUCT TERM: INPUT VARIABLE I5	I4	I3	I2	I1	I0	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	ACTIVE LEVEL: OUTPUT FUNCTION O7	O6	O5	O4	O3	O2	O1	O0
0	-	-	-	-	-	-	-	-	-	L	L	L	L	H	H	H	A	•	•	•	A	•	A	A
1	-	-	-	-	-	-	-	-	-	L	L	L	L	H	H	H	A	•	•	A	•	•	A	A
2	-	-	-	-	-	-	-	-	H	L	L	L	H	H	H	H	A	•	A	•	A	•	•	A
3	-	-	-	-	-	-	-	L	L	L	L	L	H	L	H	H	•	A	•	•	•	•	•	A
4	-	-	-	-	-	-	-	L	L	L	L	H	L	H	H	H	A	•	A	•	•	A	A	A
5	-	-	-	-	-	-	-	L	H	L	L	H	L	H	H	H	A	•	•	•	•	•	A	A
6	-	-	-	-	-	-	-	H	L	L	L	H	L	H	H	H	A	•	•	A	A	A	•	A
7	-	-	-	-	-	-	L	L	L	H	L	H	L	H	H	H	A	•	•	A	•	•	•	A
8	-	-	-	-	-	-	L	L	H	L	L	H	L	H	H	H	A	•	A	•	A	•	A	A
9	-	-	-	-	-	-	L	H	L	L	L	H	L	H	H	H	A	•	A	•	•	A	•	A
10	-	-	-	-	-	-	H	L	L	L	L	H	L	H	H	H	A	•	•	•	A	•	•	A
11	-	-	-	-	-	-	H	L	H	L	L	H	L	H	H	H	A	•	•	A	A	•	A	A
12	-	-	-	-	-	L	L	L	L	H	L	H	L	L	H	H	•	A	A	•	A	•	A	A
13	-	-	-	-	-	L	L	L	H	L	L	H	L	L	H	H	A	•	A	•	A	A	•	A
14	-	-	-	-	-	L	L	H	L	L	L	H	L	L	H	H	A	•	•	•	A	A	•	A
15	-	-	-	-	-	L	L	H	H	L	L	H	L	L	H	H	A	•	•	A	A	•	•	A
16	-	-	-	-	-	L	H	L	H	L	L	H	L	L	H	H	A	•	A	•	A	A	A	A
17	-	-	-	-	-	H	L	L	L	L	H	L	L	L	H	H	A	•	•	•	•	A	•	A
18	-	-	-	-	-	H	L	L	H	L	H	L	L	L	H	H	A	•	•	A	•	•	A	A
19	-	-	-	-	-	H	L	H	L	L	H	L	L	L	H	H	A	•	•	•	A	A	A	A
20	-	-	-	-	L	L	L	L	H	L	H	L	H	L	H	H	•	A	A	•	A	•	•	A
21	-	-	-	-	L	L	L	H	L	L	H	L	H	L	H	H	A	•	•	•	•	•	•	A
22	-	-	-	-	H	L	L	L	L	L	H	L	H	L	H	H	•	A	A	•	A	A	•	A
23	-	-	-	-	H	L	L	L	H	L	H	L	H	L	H	H	A	•	A	A	•	•	•	A
24	-	-	-	-	H	L	L	H	L	L	H	L	H	L	H	H	A	•	•	•	•	A	A	A
25	-	-	-	-	H	L	H	L	L	L	H	L	H	L	H	H	A	•	A	•	A	•	A	A
26	-	-	-	-	H	L	H	L	H	L	H	L	H	L	H	H	A	•	•	A	A	A	A	A
27	-	-	-	-	H	L	H	H	L	L	H	H	L	L	H	H	•	A	A	•	•	A	A	A
28	-	-	-	-	H	L	H	H	H	L	H	H	L	L	H	H	•	•	•	•	•	A	A	A
29	-	-	-	-	H	H	L	L	L	L	H	H	L	L	H	H	A	•	•	A	•	A	•	A
30	-	-	-	-	H	H	L	L	H	L	H	H	L	L	H	H	•	A	A	A	A	•	A	A
31	-	-	-	-	H	H	L	H	L	L	H	H	L	L	H	H	A	•	A	A	•	•	A	A
32	-	-	-	-	H	H	L	H	H	L	H	H	L	L	H	H	•	A	A	•	A	A	A	A
33	-	-	-	-	H	H	H	L	L	L	H	H	L	L	H	H	A	•	A	•	•	•	A	A
34	-	-	-	L	L	L	L	L	L	L	H	H	H	H	H	H	•	•	A	A	•	•	•	A
35	-	-	-	L	L	H	L	H	L	L	H	H	H	H	H	H	•	A	A	A	A	A	A	A
36	-	-	-	L	L	H	L	H	H	L	H	H	H	H	H	H	•	A	A	•	•	A	•	A
37	-	-	-	H	L	L	L	L	L	H	L	H	H	H	H	H	•	A	•	A	A	•	A	A
38	-	-	-	H	L	L	H	L	L	H	L	H	H	H	H	H	•	•	•	•	•	A	•	A
39	-	-	-	H	L	H	L	H	L	L	H	H	H	H	H	H	•	A	A	A	•	•	•	A
40	-	-	L	H	L	H	L	L	H	L	H	L	L	L	H	H	•	A	•	A	A	A	•	A
41	-	-	L	H	L	H	L	H	L	H	H	L	L	L	H	H	•	A	A	•	•	•	A	A
42	-	-	H	L	L	H	L	L	L	H	L	L	L	L	H	H	•	A	A	A	•	A	•	A
43	-	-	H	L	L	H	L	L	H	L	H	L	L	L	H	H	•	A	A	A	•	•	A	A
44	-	-	H	L	H	L	H	L	L	H	L	L	L	L	H	H	•	A	A	A	•	•	A	A
45	L	-	H	L	H	L	H	L	H	L	H	L	L	H	H	H	•	A	•	•	A	A	A	A
46	H	L	H	L	L	L	H	L	L	H	H	L	L	H	H	H	•	A	•	A	A	•	•	A
47	H	L	H	L	H	L	H	L	H	H	L	L	L	H	H	H	•	A	A	•	A	•	•	A

FIG. 5.16 TABLA PROGRAMA DEL FPLA 4.

En la Fig. 5.17 se muestra la manera en que se - deberá interconectar los FPLAS 3 y 4 para que el -- convertidor sea capaz de convertir el código Morse a ASCII. Observese que a pesar que el FPLA4 no tiene entrada para "RESET", cuando la entrada de "RESET" de el FPLA3 toma un nivel bajo, sus salidas A y B se ponen bajas, haciendo que el FPLA4 ya no pueda - reconocer ningún código.

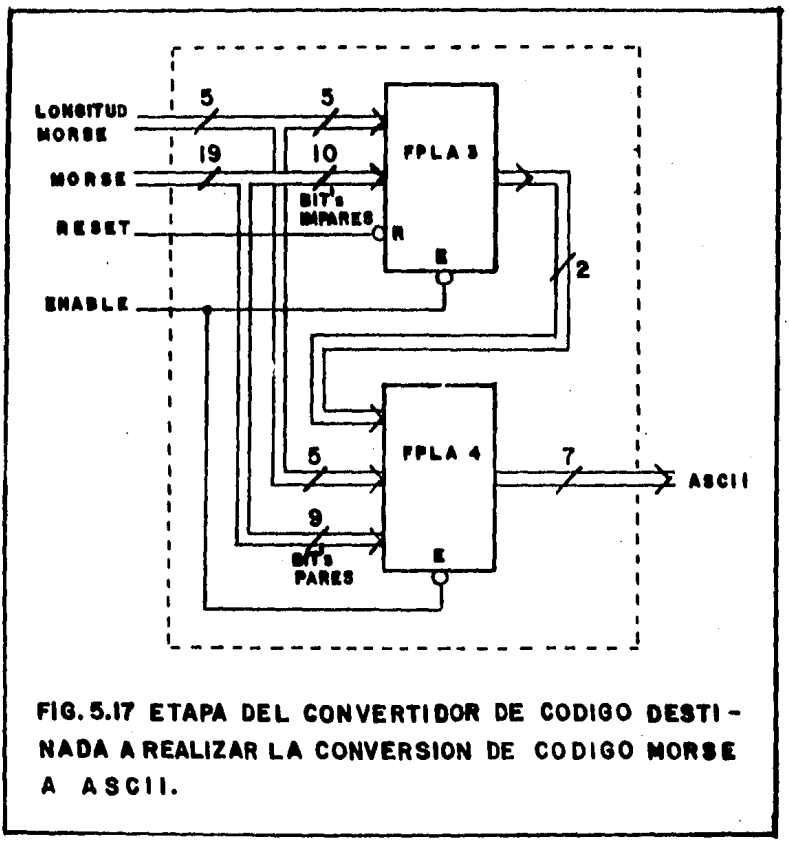


FIG. 5.17 ETAPA DEL CONVERTIDOR DE CODIGO DESTINADA A REALIZAR LA CONVERSION DE CODIGO MORSE A ASCII.

Ya han sido establecidas las dos etapas del convertidor y ahora es necesario determinar la manera en que deben ser interconectadas, para lo cual, antes se procederá a especificar la operación de las entradas de control "S" y "E".

En la Fig. 5.18 se muestra la tabla de verdad, los mapas "K", ecuaciónes de salida y circuito lógico que cumplen con las condiciónes de operación establecidas en un principio.

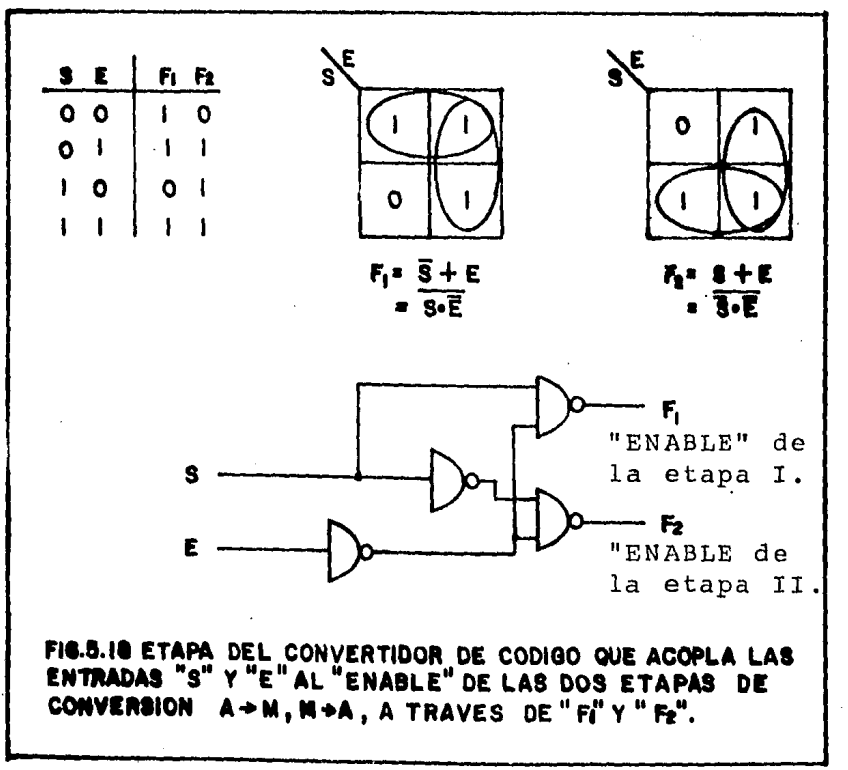


FIG.5.18 ETAPA DEL CONVERTIDOR DE CODIGO QUE ACOPLA LAS ENTRADAS "S" Y "E" AL "ENABLE" DE LAS DOS ETAPAS DE CONVERSION A→M, M→A, A TRAVES DE "F1" Y "F2".

En la Fig. 5.19 se presenta a cada una de las -- etapas que constituyen al convertidor y la interconexión entre éstas. La implementación física del convertidor de código es llevada a cabo, haciendo - uso de cuatro "FPLA's" 82S100, dos "HEX BUFFER's" - 74LS365 y un "QUADRUPLE TWO INPUT's NAND GATE" - - 74LS00. Son utilizados componentes "TTL" del tipo "LS" por que son compatibles con los FPLA's y por - que corresponden a un bajo consumo de potencia y -- una alta velocidad de operación, características -- muy importantes dentro de un diseño. Se puede ver en el diagrama que en ningún momento pueden estar - los cuatro FPLA's activos, pero dependiendo de los- niveles lógicos de "S" y "E", los FPLA's 1 y 2 pueden estar activos, mientras los FPLA's 3 y 4 están en "Tri-State"; , los FPLA's 3 y 4 estar activos, - mientras los FPLA's 1 y 2 están en "Tri-State"; o - los FPLA's 1, 2, 3, y 4 estar inactivos al mismo -- tiempo. Por otro lado tenemos que la señal de control "RESET" es aplicada directamente a cada uno de los FPLA's con excepción del último, el cual,en forma indirecta lo recibe del FPLA3.

De acuerdo con el diseño obtenido, se puede de-- terminar sus características de operación como sigue: Todos los componentes utilizados en el circuito trabajan con el mismo voltaje de alimentación y es de +5v; El consumo de corriente del circuito es obtenido a partir de los consumos individuales de - cada uno de los componentes y es de 270mA; El consu

En la Fig. 5.19 se presenta a cada una de las -- etapas que constituyen al convertidor y la interconexión entre éstas. La implementación física del convertidor de código es llevada a cabo, haciendo - uso de cuatro "FPLA's" 82S100, dos "HEX BUFFER's" - 74LS365 y un "QUADRUPLE TWO INPUT's NAND GATE" - - 74LS00. Son utilizados componentes "TTL" del tipo "LS" por que son compatibles con los FPLA's y por - que corresponden a un bajo consumo de potencia y -- una alta velocidad de operación, características -- muy importantes dentro de un diseño. Se puede ver en el diagrama que en ningún momento pueden estar - los cuatro FPLA's activos, pero dependiendo de los- niveles lógicos de "S" y "E", los FPLA's 1 y 2 pueden estar activos, mientras los FPLA's 3 y 4 están en "Tri-State"; los FPLA's 3 y 4 estar activos, - mientras los FPLA's 1 y 2 están en "Tri-State"; o - los FPLA's 1, 2, 3, y 4 estar inactivos al mísmo -- tiempo. Por otro lado tenemos que la señal de control "RESET" es aplicada directamente a cada uno de los FPLA's con excepción del último, el cual,en forma indirecta lo recibe del FPLA3.

De acuerdo con el diseño obtenido, se puede de-- terminar sus características de operación como si-- gue: Todos los componentes utilizados en el circuito trabajan con el mísmo voltaje de alimentación y es de +5v; El consumo de corriente del circuito es obtenido a partir de los consumos individuales de - cada uno de los componentes y es de 270mA; El consu

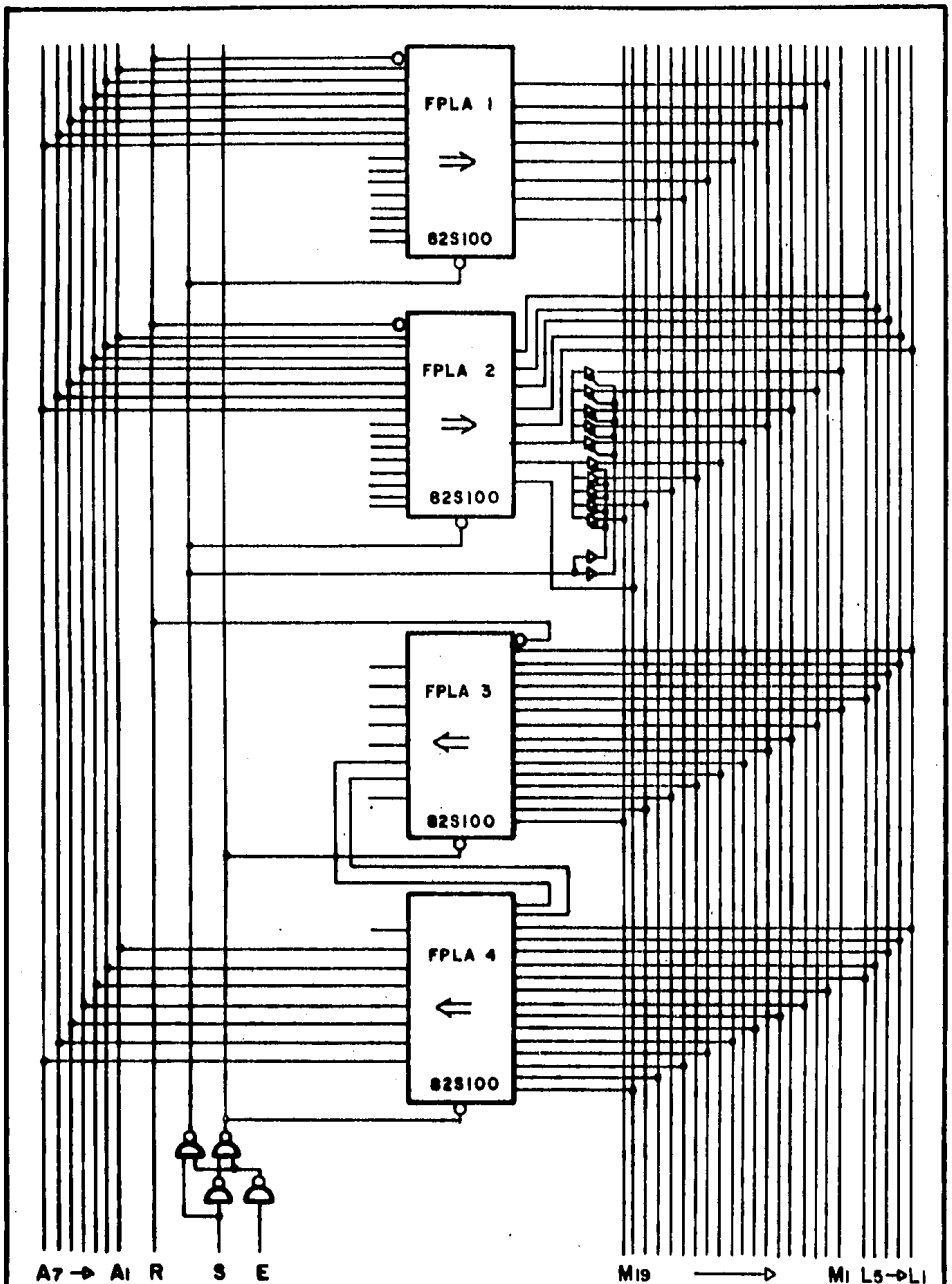


FIG 5.19. CIRCUITO LOGICO DEL CONVERTIDOR DE CODIGO ASCII-MORSE REVERSIBLE

mo de potencia es la debida a todos los componentes y es de 1.35w; y el tiempo de acceso es el obtenido en el peor de los casos, o sea, cuando se esta realizando una conversión MORSE a ASCII, debido a que los FPLA's 3 y 4 están en serie, por lo que el tiempo se duplica y es de 100nS; observese que cuando se esta realizando una conversión de código ASCII a MORSE el tiempo de propagación es el debido al ocacionado por el FPLA2 y a los Buffers, dado que estan en serie y es de 60nS.

Son éstos, los parámetros que definen el funcionamiento del circuito y los que determinarán sus ventajas frente a otros en el mercado, al hacer su evaluación.

La aplicación del convertidor de código ASCII - Morse muestra como el FPLA se adapta fácilmente en su diseño, haciendo resaltar las ventajas que resultan al utilizarlo, como son: El buen desenvolvimiento durante la operación, como resultado del uso de menor número de componentes, de niveles de lógica - y de interconexiones, la reducción del costo de - - construcción, del tamaño de la tablilla de circuito impreso y de los gastos ocasionados por el mantenimiento al tener un inventario reducido; estas ventajas se obtienen utilizando FPLA's en lugar de compuertas TTL exclusivamente en su implementación; la ventaja obtenida con el uso de FPLA's en lugar de - PROM's, es la simplicidad de la programación, ya -- que solo es necesario grabar algunas combinaciones; o la sencillez de un componente que no requiere de una sofisticada programación software como es el -- caso de los microprocesadores, pues no se trata de un problema lo suficientemente complejo como para - hacer uso de éstos.

Hay muchas otras ideas de diseño en las que se - hará aparente el uso de los FPLA's, una vez estudiado el comportamiento del sistema, pues por tratarse de estructuras lógicas de propósito general, resultan una proposición flexible y económica. Por -- otro lado, debe quedar claro que no en cualquier --

diseño, los FPLA's son siempre la mejor solución -- pero deben ser considerados antes de poder determinarlo.

B I B L I O G R A F I A

* AN INTRODUCTION TO ARRAY LOGIC (pag. 98 a 109)
  H. FLEISHER AND L. I. MAISSEL
  IBM JOURNAL OF RESEARCH AND DEVELOPMENT (MARCH 1975)

* PLA A UNIVERSAL LOGIC ELEMENT (pag. 67 a 75)
  JOE MAGGIORE (ROCKWELL MICROELECTRONICS)
  ELECTRONIC PRODUCTS MAGAZINE (APRIL 15, 1974)

* THE PLA: A "DIFFERENT KIND" OF ROM (pag.78 a 84)
  ALBERT HEMEL (MONOLITHIC MEMORIES)
  ELECTRONIC DESIGN (JANUARY 5, 1976)

* USER PROGRAMS LOGIC ARRAY (pag. 177)
  INTERSIL AND SIGNETICS BOPOLAR
  ELECTRONICS (MARCH 20, 1975)

* FIELD-PLA's SIMPLIFY LOGIC DESIGNS (pag.84 a 90).
  NAPOLEONE CAVLAN (ADVANCED PRODUCTS)AND RON CLINE (SIGNETICS)
  ELECTRONIC DESIGN (SEPTEMBER 18, 1975)

* HOW TO DESIGN WITH PROGRAMMABLE LOGIC ARRAY (pag. 1 a 8)
  DALE MRAZEK AND MEL MORRIS
  NATIONAL SEMICONDUCTOR

* SIGNETIC's FIELD PROGRAMMABLE LOGIC ARRAY (pag.1a38)
NAPOLEONE CAVLAN
SIGNETICS (FEBRUARY 1976)

* PAL DATABOOK (pag. 1-1 a 4-63)
NATIONAL SEMICONDUCTOR CORPORATION (1980)

* THE ART OF DIGITAL DESIGN (pag. 134 a 137)
DAVID WINKEL AND FRANKLIN PROSSER
PRENTICE HALL (1980)

* GRAN ENCICLOPEDIA PRACTICA DE LA ELECTRICIDAD (pag. 430a469)
TOMO II
HENRI DESARCES
EDITORIAL LABOR S. A.

* OSCILADOR PARA CODIGO MORSE (pag. 211 a 212)
LA ELECTRONICA AL ALCANCE DE TODOS (1962)

* THE DESIGN OF DIGITAL SYSTEMS (pag. 294)
JOHN B. PEATMAN
McGRAW HILL (1972)

* TEORIA DE CONMUTACION Y DISEÑO LOGICO (pag. 121 a 148)
FREDERICK J. HILL, GERALD R PETERSON
LIMUSA (1982)