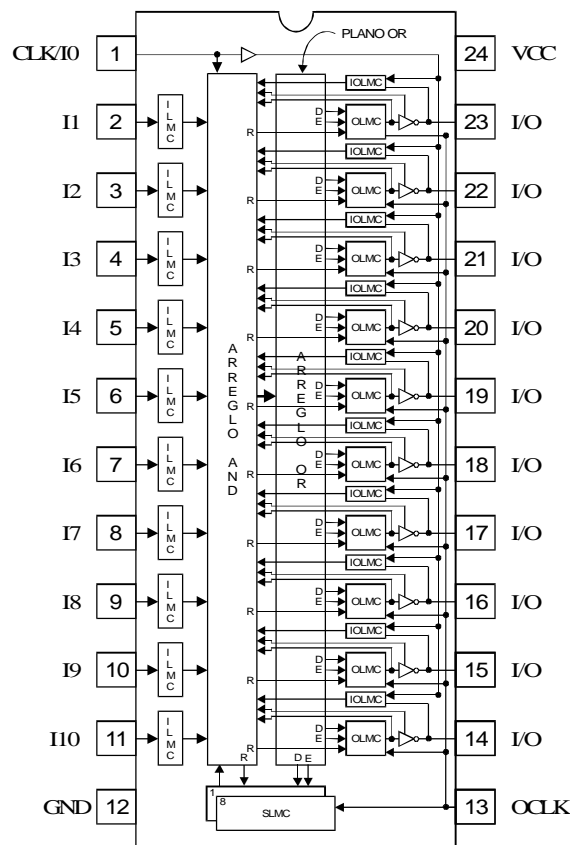


UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS
EXACTAS E INGENIERIAS

PLD ´S

“DISPOSITIVOS LOGICOS
PROGRAMABLES”



EXTRACTO ELABORADO POR: ING. JOSE MIGUEL MORAN LOZA

¿ QUE ES UN PLD ?

Las iniciales PLD vienen del inglés *Programmable Logic Device*, que traducido a nuestro idioma significa Dispositivo Lógico Programable y son circuitos integrados que ofrecen a los diseñadores en un solo chip, un arreglo de compuertas lógicas y flip-flop's, que pueden ser programados por el usuario para implementar funciones lógicas; y así, una manera más sencilla de reemplazar varios circuitos integrados estándares o de funciones fijas.

Las ventajas que trae con respecto a los circuitos integrados de funciones fijas (series 74XX y 40XX) son variadas, entre ellas las que considero más importantes son:

- Los PLD's representan menor costo para los fabricantes.
- Pueden reemplazar funciones de otros dispositivos lógicos.
- Reducción de espacio en las tarjetas de circuito impreso.
- Simplificación del alambrado entre unos chips y otros.
- Disminución en los requerimientos de potencia (por consiguiente menor consumo de energía)
- Realización de aplicaciones especiales no encontradas en circuitos integrados de funciones fijas.
- Puede reflejarse menor costo para el usuario al ver las ventajas de tener menor cantidad de circuitos integrados; por consiguiente, procesos de ensamblado más rápidos, menor probabilidad de que puedan ocurrir fallas, así como menores procedimientos en la detección de fallas cuando estas se presenten.

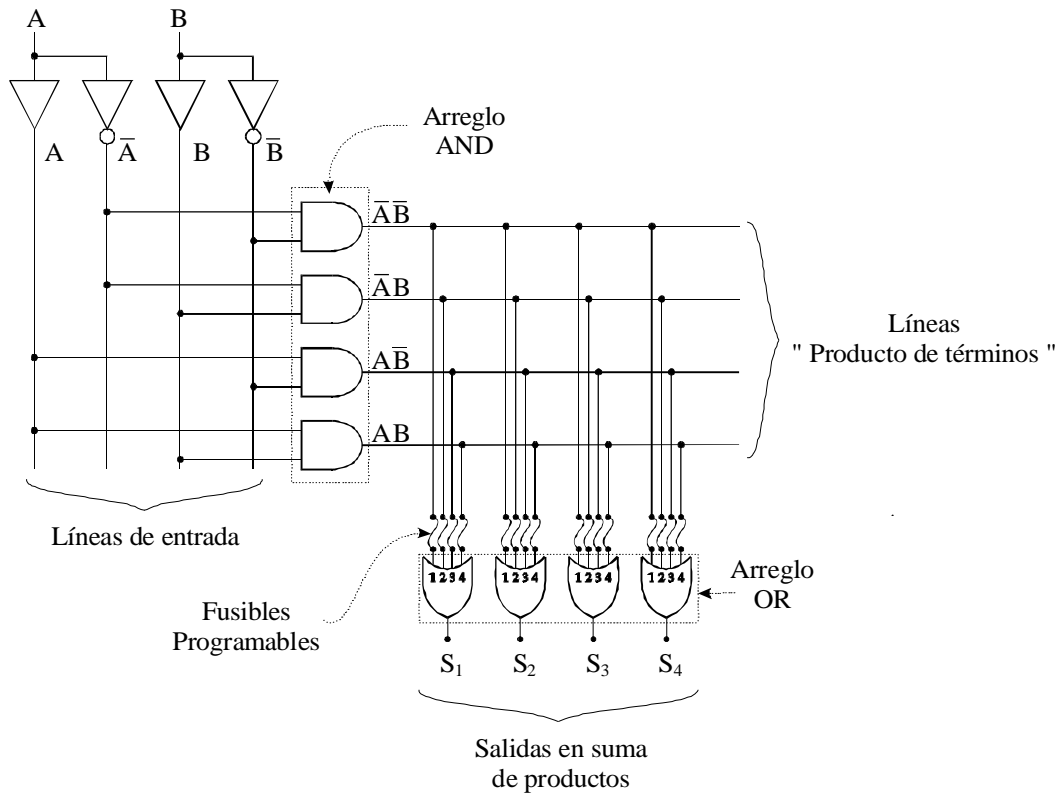
Un PLD típico está compuesto de arreglos de compuertas lógicas, uno de ellos a base de compuertas AND al que se le denomina **Plano AND** y el otro de compuertas OR, denominado **Plano OR**; estos pueden ser programables y dependiendo del plano o los planos que lo sean, será la clasificación que reciba el PLD.

Las variables de entrada (que vienen de las terminales externas del dispositivo) tienen interconexiones hacia uno de los planos, a través de compuertas con salidas complementarias (es decir con una salida inversora y una no-inversora); y salidas de los planos, conectadas a las terminales externas del dispositivo, por elementos lógicos como pueden ser: inversores, compuertas OR y flip-flop's; además, en algunos casos existe retroalimentación de las salidas hacia uno de los planos, para tomarlas como entradas nuevamente (aplicación utilizada frecuentemente en el caso de lógica secuencial).

La programación se lleva a cabo por medio de conexiones fusibles; de tal forma que en una compuerta OR, una entrada con conexión fusible " Fundida o Quemada " (fusible abierto) funcione como un cero lógico y una conexión intacta como el valor de la(s) variable(s) de entrada.

Un ejemplo de un PLD sencillo se muestra a continuación:

Se tienen dos variables de entrada, etiquetadas como A y B, en donde cada una se conecta a dos compuertas, a un inversor y a un no-inversor, las salidas de dichas compuertas van directamente conectadas al Plano AND y las salidas de las compuertas del Plano AND, van conectadas a las entradas de las del Plano OR y las salidas de este plano, hacia las terminales externas del dispositivo como se muestra en la figura siguiente.



Sin quemar ningún fusible la salida de cada compuerta OR es igual a 1.

Demostración:

$$\begin{aligned}
 S_1 &= \bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB \\
 S_1 &= \bar{A}(\bar{B} + B) + A(\bar{B} + B) \quad \text{y si } \bar{B} + B = 1 \\
 S_1 &= \bar{A} + A \\
 S_1 &= \bar{A} + A = 1
 \end{aligned}$$

Las salidas S_1, S_2, S_3 y S_4 se pueden programar en forma individual para lograr cualquier función posible con solo "Quemar los fusibles". Por ejemplo para obtener la operación de una compuerta NOR-Exclusiva en la salida S_1 , se necesitan quemar los fusibles 2 y 3. Recordando que en este ejemplo un fusible quemado es igual a un cero lógico.

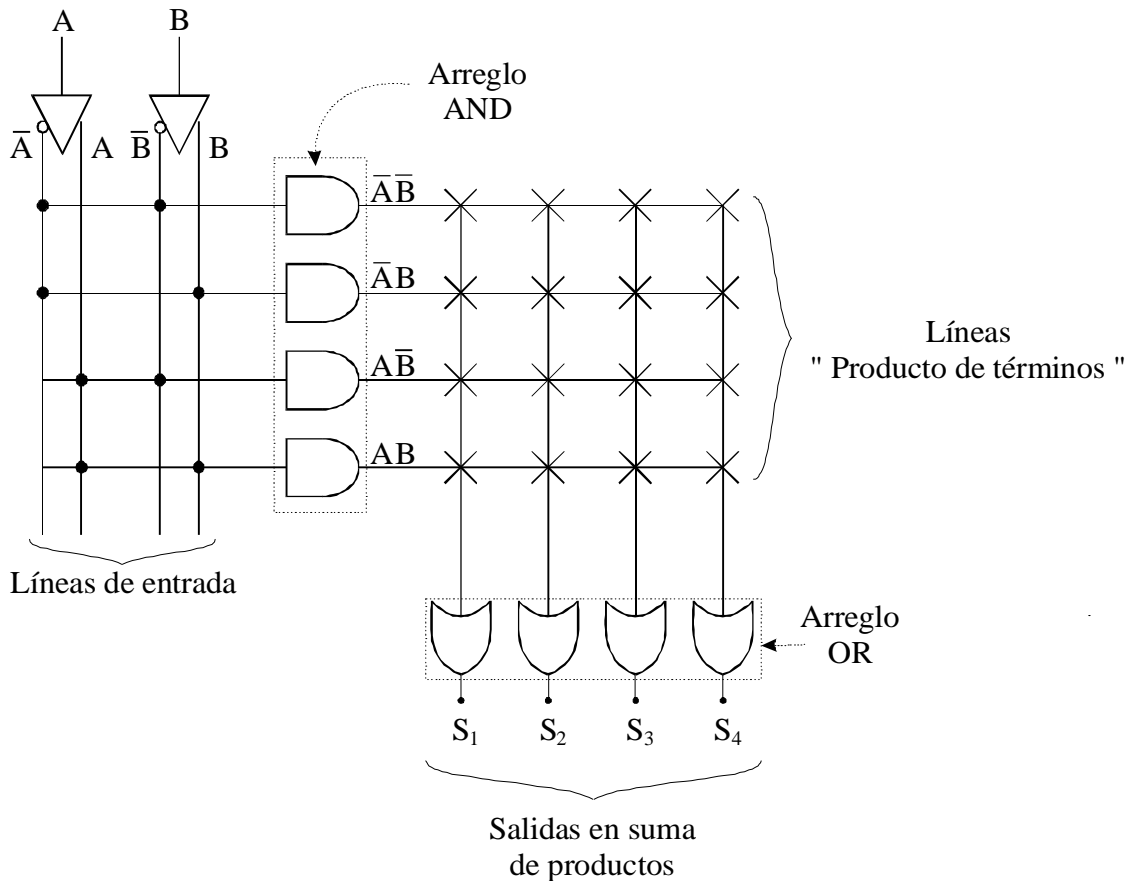
$$\begin{aligned}
 S_1 &= \bar{A}\bar{B} + 0 + 0 + AB \\
 S_1 &= \bar{A}\bar{B} + AB \\
 S_1 &= \overline{A \oplus B}
 \end{aligned}$$

SIMBOLOGIA ADOPTADA EN LOS PLD's

Como se pudo apreciar en la figura anterior solo se tienen dos variables de entrada, habría que imaginar cuan grande sería uno de cuatro, seis u ocho variables; para ver que ya es bastante complejo para poder representarlo. Afortunadamente los fabricantes han adoptado una simbología simplificada; para poder así, describir la circuitería interna del dispositivo.

Los fabricantes han sustituido el símbolo del inversor y del no-inversor en uno solo; pero, con dos salidas complementadas. Han simplificado las líneas de entrada a una compuerta AND u OR, por medio de una sola línea. Las conexiones entre compuertas se representan mediante una "X" o un punto. Las "X" se encuentran en el Plano programable y describen una conexión fusible intacta. En el Plano fijo, un punto representa una conexión fija y que por supuesto, ya no puede cambiarse. La ausencia de estos dos símbolos en un cruce de líneas significa que no existe conexión entre ellas.

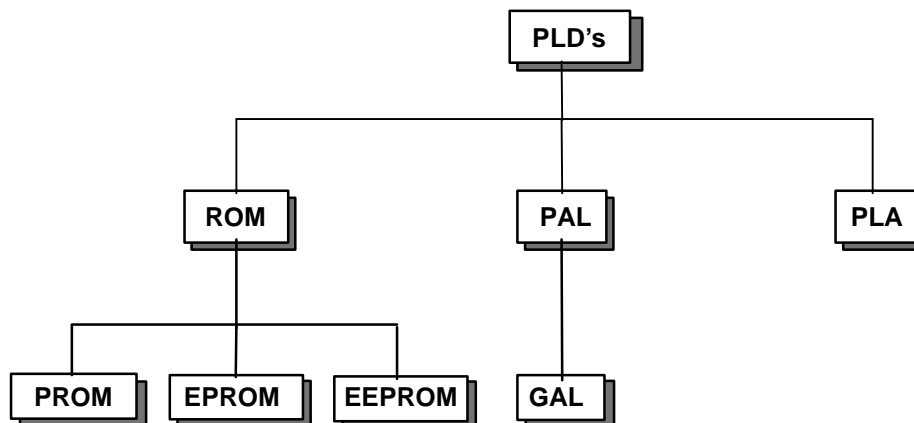
Ejemplo:



CLASIFICACION ENTRE ARQUITECTURAS DE LOS PLD's

La clasificación de los PLD's, como se mencionó anteriormente, dependerá básicamente del plano o los planos que sean programables.

La clasificación se hace en tres grupos:



ROM *Mask Read-Only Memory* (Memoria de Máscara Programable de Solo Lectura), Dispositivo programado solamente por el fabricante y como se muestra en el esquema anterior este se subdivide en tres partes que son:

PROM *Programmable Read-Only Memory* (Memoria Programable de Solo Lectura), Dispositivo programado por el usuario y no borrable o reprogramable.

EPROM *Erasable Programmable Read-Only Memory* (Memoria Programable y Borrable de Solo Lectura); este tipo de Memorias se borran Mediante Luz ultravioleta; con la ventaja de que puede ser programada por el usuario.

EEPROM *Electrically Erasable Programmable Read-Only Memory* (Memoria Programable y Borrable Eléctricamente de Solo Lectura); al igual que la anterior está puede ser programada por el usuario.

Y pueden ser utilizados como PLD's, debido a que las entradas de direccionamiento pueden ser manejadas como variables de entrada en las ecuaciones y las salidas de la memoria, como salidas de las mismas.

El número de productos es igual a:

$$2^n \times S = C$$

Donde:

n es igual al número de variables de Entrada.

S es la cantidad de funciones de Salida.

C es la capacidad de la memoria en bits.

De tal forma que, para una ecuación de cuatro variables de entrada y cuatro funciones distintas de salida será necesario una memoria de:

$$2^4 \times 4 = 16 \times 4 = 64 \text{ bits}$$

en caso de que fuera una de ocho variables de entrada y de cuatro funciones de salida sería necesario una memoria de:

$$2^8 \times 4 = 256 \times 4 = 1\text{K bits, una 74S287 por ejemplo}$$

y en caso de que fuese necesario manejar doce entradas y ocho salidas se necesitaría una memoria de:

$$2^{12} \times 8 = 4\text{K} \times 8 = 32\text{K bits, una 27C32 por ejemplo.}$$

Desgraciadamente estas se vuelven imprácticas cuando se contemplan grandes números de entradas, debido a que por cada variable que se anexe, el arreglo de fusibles se duplica. Muchas aplicaciones requerirán de un número mayor de entradas, pero no tendrán la flexibilidad que puede ofrecer una PROM como decodificador completo. Desde el punto de vista del fabricante usar una PROM como PLD representa un uso ineficiente del silicio y por lo tanto se incrementa su costo.

En este tipo de PLD's el plano AND es fijo y el OR es programable.

PLA

Programmable Logic Array (Arreglo Lógico Programable), este tipo de dispositivos resuelve el problema de las PROM; debido a que, tiene tanto el plano AND como el OR programables. De forma que solo se seleccionan los productos de términos necesarios para las diferentes aplicaciones; esto hace mucho más eficiente la matriz programable y al dispositivo más versátil. A este tipo de dispositivos, también se les conoce como *Field Programmable Logic Array* (Arreglos Lógicos Programables de Campo). Los FPLA o PLA aceptan más variables de entrada con mucho menor producto de términos que 2^n . Estos PLD's incluyen además la capacidad de programar la polaridad de salida, lo que permite trabajar con max-términos si se requieren; esto se logra a través de una OR-Exclusiva.

Un FPLA es el TIFPLA840 de Texas Instruments, el cual es especificado como un FPLA de 14 x 32 x 6. Es decir que, tiene 14 variables como entradas, 32 compuertas AND para generar los productos lógicos de las variables, y 6 compuertas OR que pueden formar cualquier combinación de las salidas de las compuertas AND.

Un ejemplo más es la serie " MAPL " *Multiple Array Programmable Logic* (Lógica Programable en Arreglo Múltiple), de National Semiconductor; que no son, más que arreglos de FPLAS como son: El MAPL128 y el MAPL144, algunos incluyen un arreglo PAL; como lo es el MAPL244.

No obstante, los fusibles adicionales (debido a que hay dos planos programables), agregan un retardo mayor que los de un solo plano programable y una circuitería más compleja y al mismo tiempo la programación se vuelve más elaborada. Debido a la tecnología que utilizan también aumenta su costo.

PAL

Programmable Array Logic (Lógica en un Arreglo Programable), la arquitectura de éste PLD esta compuesta por un Plano AND programable y el Plano OR fijo. Este dispositivo es el intermedio entre una PROM y un PLA; debido a que, por cada entrada que se agregue no será necesario duplicar la cantidad de fusibles y el tener un plano fijo conduce a un menor retardo en la circuitería interna. También incluye la capacidad de programar la polaridad de salida. Este PLD puede incluir una serie de componentes a la salida del plano OR, como pueden ser: Inversores y Flip-Flops, que permitirán hacer del dispositivo, un PLD versátil.

Existen dos tipos de PAL's, uno de los cuales puede ser programado solamente una vez, por ejemplo: El PAL16R8 el cual es un dispositivo de 16 posibles entradas y con 8 salidas; todas con Flip-Flops. El otro PAL mejor conocido como GAL de *Generic Array Logic* (Lógica en Arreglo Genérico), combina las características de un PAL; pero además, agrega tecnología especial para ser borrado y programado eléctricamente. Este dispositivo que es el que nos ocupa, será descrito y analizado detalladamente en las páginas subsecuentes.

DIFERENCIA ENTRE LAS ARQUITECTURAS DE LOS PLD'S

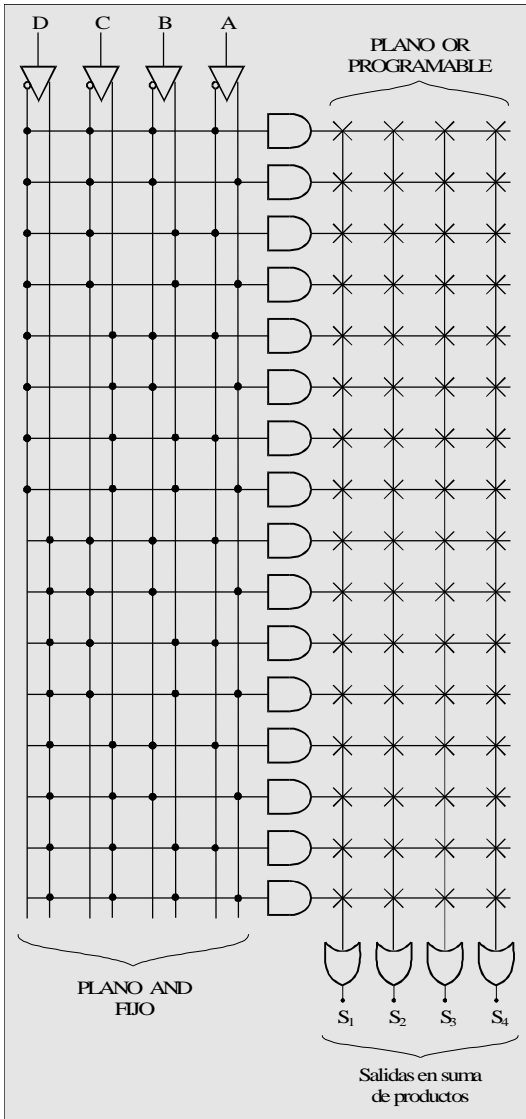


DIAGRAMA ESQUEMATICO DE UN PLD TIPO PROM

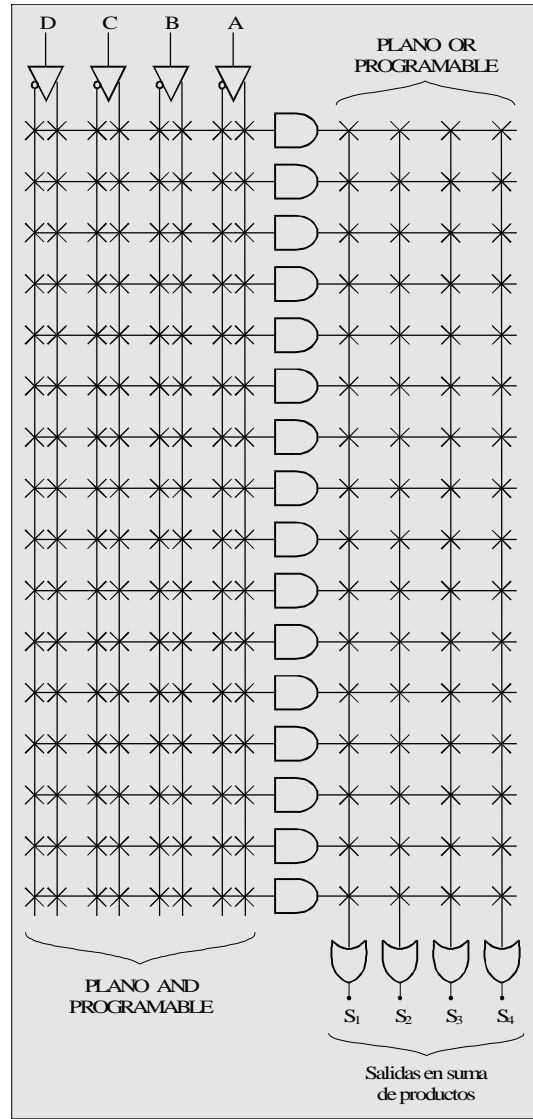


DIAGRAMA ESQUEMATICO DE UN PLD TIPO PLA

DIFERENCIA ENTRE LAS ARQUITECTURAS DE LOS PLD's
(CONTINUACION)

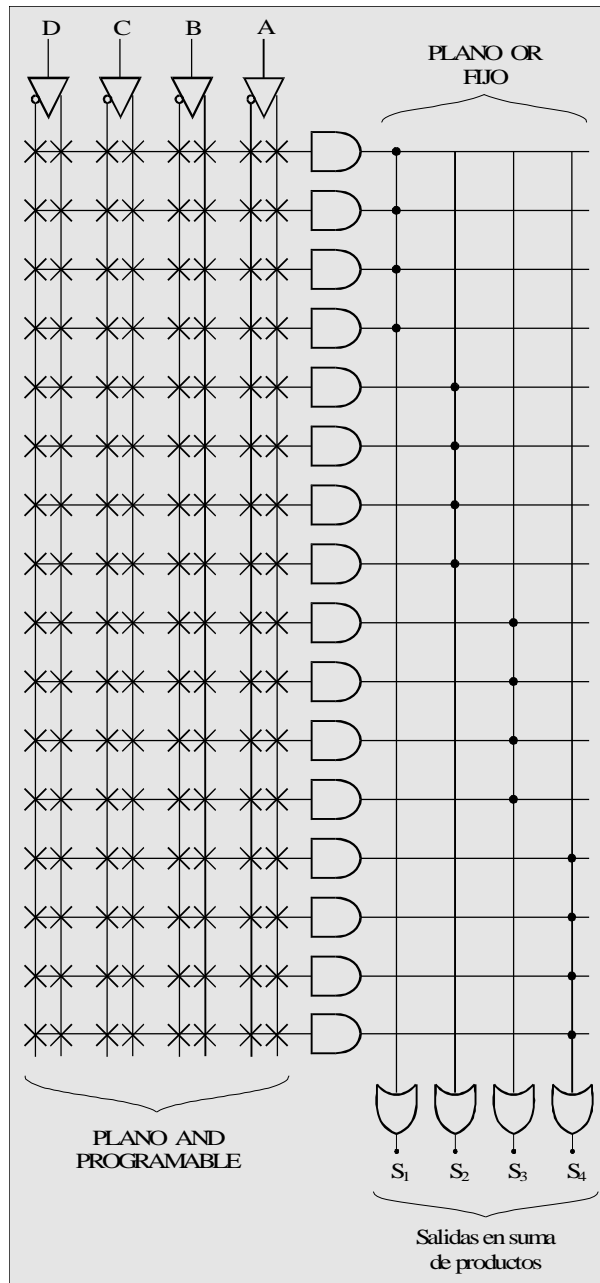


DIAGRAMA ESQUEMATICO DE
UN PLD TIPO PAL

CRONOLOGIA DE LOS PLD'S

- 1948** Se inventa el transistor de contacto puntual en los Laboratorios Bell Telephone en Estados Unidos, resultado de una investigación sobre semiconductores llevada a cabo por Walter Brattain, John Bardeen y William Shockley, quienes recibirían el premio Nobel por su enorme contribución en 1956.
- 1951** Se logra un transistor con una estructura como la que se conoce actualmente.
- 1957** John Wallmark de RCA patenta el FET (Field Effect Transistor).
- 1959** Se concibe el primer Circuito Integrado Digital en la compañía Texas Instruments y es Jack Kilby quién desarrolla un Flip-Flop sobre una base de substrato de Germanio y contenía solo cuatro transistores.
- 1961** Se presenta la primera familia de Circuitos Integrados Digitales comerciales, denominada R.T.L. (Resistor - Transistor - Logic) y que fue introducida por Fairchild Semiconductor bajo la serie 900, está familia operaba a 3.2 Voltios, poco tiempo después nace otra familia denominada D.T.L. (Diode - Transistor - Logic).
- 1962** Aparece la familia T.T.L. (Transistor - Transistor - Logic), con características como el de ser más rápida que sus predecesoras, los primeros trabajos hechos en TTL los realizó James Buie de Pacific Semiconductor (hoy subsidiaria de TRW). En ese mismo año Steven Hofstein y Frederick Heiman de RCA, desarrollan el MOSFET y a finales del mismo, fabrican el primer Circuito Integrado MOS (Metal - Oxide - Silicon) que contenía 16 transistores sobre una pastilla de silicio de 0.063 mm por lado.
- 1963** La compañía RCA producía un Circuito Integrado con cientos de MOSFET'S en un área muy reducida, al mismo tiempo nacían familias como la MOS de canal N y de canal P, NMOS y PMOS respectivamente y así como la CMOS (Complementary MOS). La CMOS se impuso con el tiempo bajo la serie 40XX lanzada por RCA. Y poco tiempo después la 74CXX de National Semiconductor.
- A mediados de los 60's surge el primer PLD, una matriz de diodos configurables y fusibles desarrollado por Harris Semiconductor (conocida en ese tiempo como Radiation, Inc.).
- 1967** Fairchild lanza al mercado una ROM de 64 bits con tecnología MOS.
- 1969** Nace el primer PLA, desarrollado por IBM y descrito como ROAM (Read - Only Associative - Memory).
- 1970** La compañía Harris crea la PROM, que combinaba la tecnología de fusibles de nicromo con una simplificación en la estructura de la ROM. En este mismo año Texas Instruments fabrica el TMS 200 y era un Circuito Integrado de máscara programable basado en el ROAM de IBM, este manejaba diecisiete entradas y ocho salidas, contenía ocho Flip - Flops JK como elementos de memoria.
- 1971** Collins Radio ofrece otro PLA de máscara programable denominado CRC 3506/7, similar al TMS 200. Intel hace una innovación tecnológica al introducir la EPROM borrable con rayos UV. General Electric abre una puerta más con una nueva tecnología PROM, desarrollada por David Greer, donde la estructura consistía de un Plano-Or y señales que van hacia un Plano-And; permitiendo el uso de lógica de multinivel sin desperdicio de pines I/O.

- 1971** Al mismo tiempo General Electric hace experimentos con PLD's de tecnología MOS, usando las características de los PLA y con la tecnología de borrado con rayos UV. En junio Intel ofrece al mercado el primer microprocesador MOS (el 4004, de 4 bits) que contenía 2300 transistores.
- 1972** MOSTEK Corporation lanza la primera Memoria de Alta Densidad (una RAM dinámica de 1024 bits e Intel ofrece los primeros microprocesadores de 8 bits (el 8008 y el 8080).
- 1973** National Semiconductor crea su propio PLA de máscara programable similar al TMS 200 pero con catorce entradas y ocho salidas sin elementos de memoria. El mérito a este dispositivo consistió en su menor complejidad en el diseño, mostrando así un avance en la nueva tecnología. Este dispositivo fue conocido como DM7575/DM8575.
- 1974** Monolithic Memories fabrica un dispositivo denominado PALA (Programable Associative Logic Array) bajo el número de parte MMT 5760/6760 implementaba multiniveles y circuitos secuenciales de más de 100 compuertas e incorporó bajo acuerdo de General Electric innovaciones en el dispositivo de máscara programable.
- 1975** Intersil anuncia el IM5200 un FPLA; poco después, Signetics hace lo mismo con el 82S100 que logró encabezar la carrera de los PLD's durante un tiempo.
- 1978** En el verano de este año nace el dispositivo PAL, como un proyecto de MMI encabezado por John Birkner, en el que se pretendían satisfacer varias necesidades del mercado, entre ellas las de reemplazar la lógica estándar, mejorar los tamaños y la velocidad de los ya existentes; bajo esta idea los PAL invaden el mercado. El PAL que conocemos actualmente se basa en un diseño de H.T. Chua. MMI ofrece soporte para el manejo de los nuevos dispositivos en el " PAL Handbook " escrito por John Birkner y que en el mismo se acompañaba de un programa hecho en Fortran para ayudar a programar los dispositivos.
- 1980** Se propone y presenta el primer formato JEDEC para los PLD's.
- 1981** Signetics registra FPLA's con aplicaciones para máquinas de estados.
- 1982** En el verano de este año Bill Wiley Smith de Signetics crea una muestra de lo que sería el soporte para la programación de PLD's, llamado BEE (Boolean Equation Entry); cuyas características eran las Ecuaciones Booleanas, notación de estados, tablas de verdad, minimización lógica en forma automática así como la simulación de los diseños. En diciembre de este año se anuncia el proyecto ABEL (Advanced Boolean Expression Language) para un número limitado de PLD's de diferentes manufacturas y que fue un Software muy bien recibido por el mercado. Se crea otra herramienta denominada CUPL (Common Universal tool for Programable Logic) desarrollado por Bob Osann de Assited Technology.

- 1983** En Marzo se crea una segunda versión de CUPL soportando a todos los PAL's soportados por PALASM y un número limitado de FPLAS combinacionales de Signetics con características similares al BEE. Poco después surge la segunda versión de ABEL que soportaba virtualmente a todos los PLD's de esa época y esto toma por sorpresa a los diseñadores de otras herramientas de Software. En seguida National Semiconductor lanza el Software llamado PLAN (Programable Logic Analysis by National). Cypress Semiconductor crea un PAL que se hace popular por su alta velocidad. Lattice Semiconductor compañía especializada en tecnología borrable CMOS crea un PAL borrable eléctricamente al que llamaron Generic Array Logic o GAL, pero esta compañía tuvo problemas legales con MMI hoy parte de AMD quién obtuvo el derecho de producir el GAL pero bajo otro nombre, en seguida Lattice crea el GAL 39V18 conocido hoy como Lattice 6001.
- 1983** International CMOS Technology (I.C.T.) desarrolla un dispositivo llamado PEEL (Programable Electrically Erasable Logic) con tecnología de Lattice y fue llevado a primera producción en 1986.
- 1984** Se anuncia un nuevo concepto en cuanto a la tecnología de los PLD's y es encabezado por Xilinx Corporation, el dispositivo desarrollado es el LCA (Logic Cell Array) compuesto de pequeñas celdas lógicas, similares a la arquitectura de una PROM, donde cada celda es capaz de crear cuatro o cinco funciones de entrada y dos de salida. Poco después Exel Microelectronic's crece el XL78C800 Erasic, este dispositivo creado bajo arreglo de multiniveles de lógica y tardo aún más el diseño del Software y programación en estar disponible.
- 1988** Actel Corporation introduce un FPGA diferente al de los dispositivos de Xilinx. El Act 1 de densidad comparable al arreglo de compuertas de máscara programable al igual que el LCA requiere de el trazado de rutas de funciones lógicas para ser usado efectivamente.
- 1989** Plessey Semiconductor introduce un FPGA con características similares, pero con una mejor arquitectura.
- 1995** Lattice Semiconductor Corporation, anuncia el 7 de Agosto, la introducción de World's fastest 3.3 Volt 22V10, cuya máxima velocidad de operación es de 7.5 ns (133.33 Mhz). Que permite la utilización del dispositivo con baterías.

Lattice Semiconductor Corporation, anuncia una actualización para los Programadores Universales en septiembre de ese año. La actualización responde a que sus nuevos productos, el GAL16LV8D-3LJ y el GAL16V8D-5LJ son lanzados al mercado. Las nuevas cualidades de estos dispositivos son : velocidades desde 3.5 ns (286 Mhz); además de operar a un voltaje de alimentación de 3.3 Volts.

El presente capítulo esta destinado en forma muy especial a los **GAL**, que es el tema principal de esta tesis y es aquí donde se descubrirán sus secretos , en donde también se hará notar lo que los hace unos PLD's muy versátiles. La principal misión de la investigación realizada, será reflejada en este capítulo en su mayor parte y que es la de facilitar el Diseño Lógico Digital. Y que estoy seguro se logrará, después de describir a cada uno de los dispositivos que me ocupan.

¿ Qué es un GAL ?.

Primeramente diré que significa " GAL ". GAL son las iniciales de **Generic Array Logic** y que en nuestro Idioma significa *Arreglo Lógico Genérico*. Y se trata de la 4ª generación de PAL's, capaces de funcionar en modo combinacional y/o secuencial; además, de superar a sus antecesores en cuanto a tecnología programable se refiere, ya que estos son capaces de reprogramarse hasta un mínimo de 100 veces; aunque, esto depende también del fabricante.

Les llamo la 4ª generación de PAL's debido a que:

La **1ª Generación** corresponde a los PAL's comunes creados por AMD (Advanced Micro Devices), y que son programables una sola vez y que emplean tecnología PROM de fusible Titanio-Tungsteno.

La **2ª Generación** correspondería a los PAL creados con arquitectura " V " (Variable); pero, programables una sola vez. Esta designación es apoyada por Texas Instruments.

La **3ª Generación** será aquella que permite la ventaja de la arquitectura " V ", con tecnología EPROM y borrado con rayos U.V.

La **4ª Generación** es la propia del GAL que conocemos actualmente, arquitectura " V "; pero, con tecnología EEPROM. Creada en forma casi simultánea por AMD y LATTICE.

Se sabe cual es la diferencia entre arquitecturas *ROM, PLA, PAL* y sus planos programables, ya que se habló de esto en la introducción, pero no será suficiente para formarse una idea de lo que es un PAL a nivel comercial, mucho menos un GAL.

Y Para tener una imagen precisa del funcionamiento del GAL, se iniciará dando un vistazo a los PAL; esto permitirá entender las distintas configuraciones que cada GAL puede adoptar y así comprender el ¿ Por qué ? algunos de ellos como el GAL16V8 y el GAL20V8, son capaces de emular a casi todos los PAL; además, de obtener una visión completa y bien formada de lo que se puede lograr con estos dispositivos que nos ocupan, los GAL.

La razón principal de revisar la arquitectura de los PAL, antes que la del GAL es la siguiente:

" Los GAL, conservan algunas características propias de los PAL. Tan es así que los GAL's 16V8, 20V8, y posteriormente el 20RA10 y el 22V10 fueron creados para reemplazar la mayoría de los PAL's existentes ya, en la época de los 80's y no solo eso; sino que los superaron ".

Es natural que se piense en que se explicarán todos los PAL; pero, no es la finalidad primordial de esta tesis y solo se tomarán aquellos que de alguna manera, estén más interrelacionados en cuanto a comportamiento de los GAL se refiere.

ETIQUETAS

La lógica estándar (TTL's y CMOS) tiene etiquetas que nos ayudan a identificar el tipo de Circuito Integrado con el cual estamos trabajando, por ejemplo:

La etiqueta 74LS04 nos dice que se trata de un Circuito Integrado de la familia TTL y de la sub-familia Schottky de baja potencia, y que está compuesto de seis INVERSORES.

La etiqueta 4011 nos dice que se trata de un Circuito Integrado de la familia CMOS, y que está compuesto por cuatro compuertas NAND.

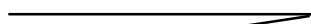




De la misma forma los dispositivos PAL y GAL tendrán etiquetas que nos ayudaran a identificar los Circuitos Integrados fácil y rápidamente. La nomenclatura general de estos dispositivos se muestra en la siguiente página.

Cabe hacer las siguientes aclaraciones:




- El orden de las etiquetas en la nomenclatura mostrada, puede verse afectada según del fabricante que se trate.
- Pueden aparecer etiquetas que se presten a confusión.
- Habrán etiquetas que no aparezcan dentro de la nomenclatura.

Por ejemplo:

EL **TIBPAL16L8-15CN**; es un PAL de Texas Instruments, donde:

TI = Texas Instruments.  *Aclaración c).*
B = Tecnología de Fusible bipolar. 
PAL = Programmable Array Logic.
16 = Con 16 entradas al Arreglo.
L = Con salidas activas en bajo.
8 = Con 8 salidas de configuración " L ".
-15 = Tiempo de retardo " 15 ns ".  *Aclaración c).*
C = Rango de Temperatura " Comercial ".  *Aclaración a).*
N = Tipo de Empaque " Plástico tipo DIP ". 

El **PALCE16V8**; es un GAL de AMD/MMI, donde:

PAL = Programmable Logic Array.  *Aclaración c) y b).*
C = Tecnología de Fusible CMOS. 
E = Erasable " Borrable ". 
16 = Con 16 entradas al Arreglo.
V = Con salidas Variables " Programables ".
8 = Con 8 salidas de configuración " V ".

En el caso del PAL de Texas Instruments, quedan claras las notas; pero, en el caso del PALCE de AMD/MMI se puede prestar a confusión; debido a que, la primer etiqueta nos habla de un PAL. Sin embargo, la " V " nos dice que se trata de un GAL y está es la forma más eficaz hasta el momento, de identificar dicho dispositivo como tal. El mismo caso se presenta con el fabricante Texas Instruments debido a que utiliza las siguientes etiquetas para describir a un GAL de 22 entradas y 10 salidas variables y programables como lo es el TIBPAL22VP10-20C. La mayoría de las ocasiones la primer etiqueta hace mención a PAL o un GAL, pero cada fabricante podrá tener su propia etiqueta, y donde de seguro se imprimirá el sello de la compañía que lo fabrique.

NOMENCLATURA

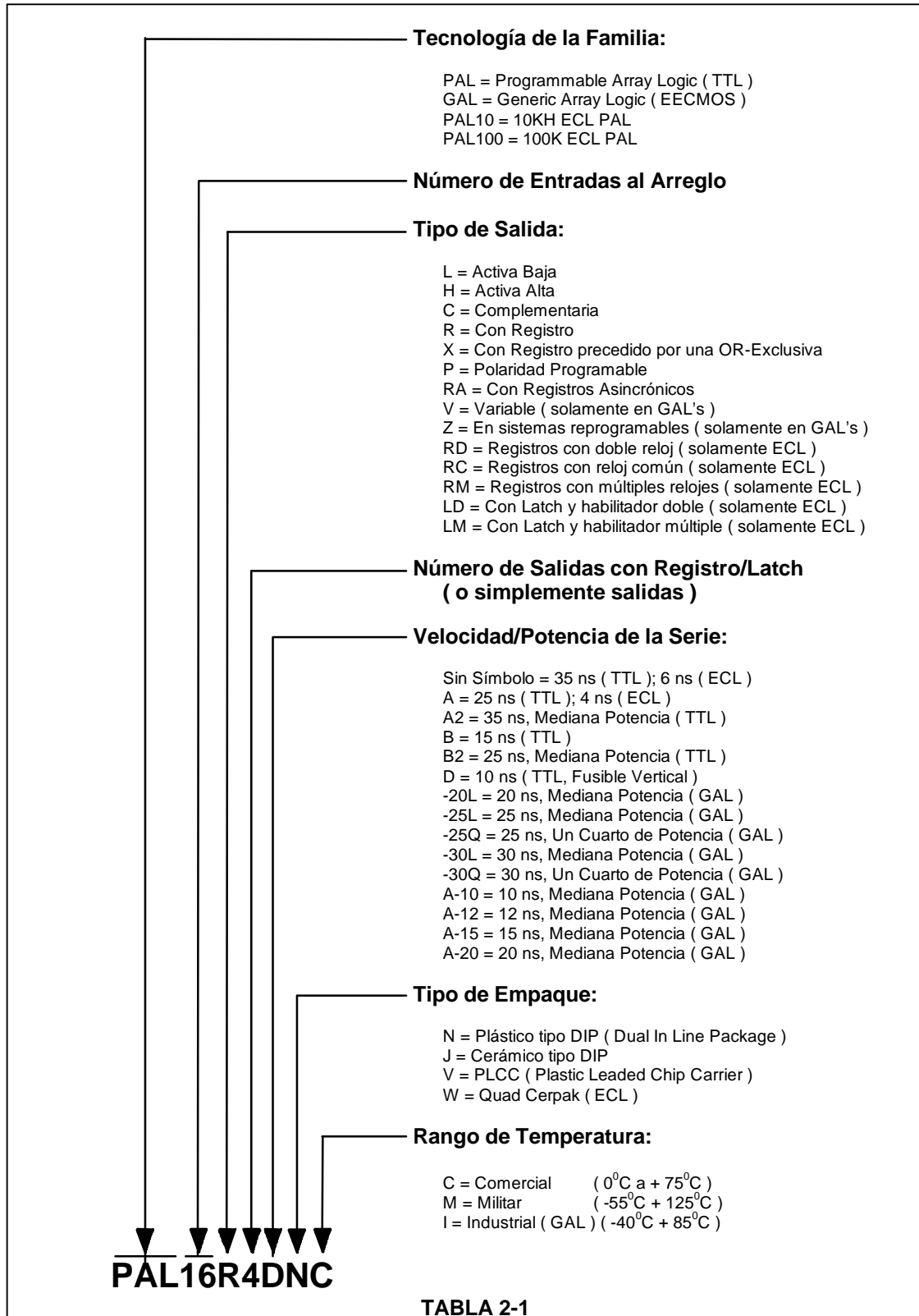


TABLA 2-1

ARQUITECTURA DEL PAL16L8

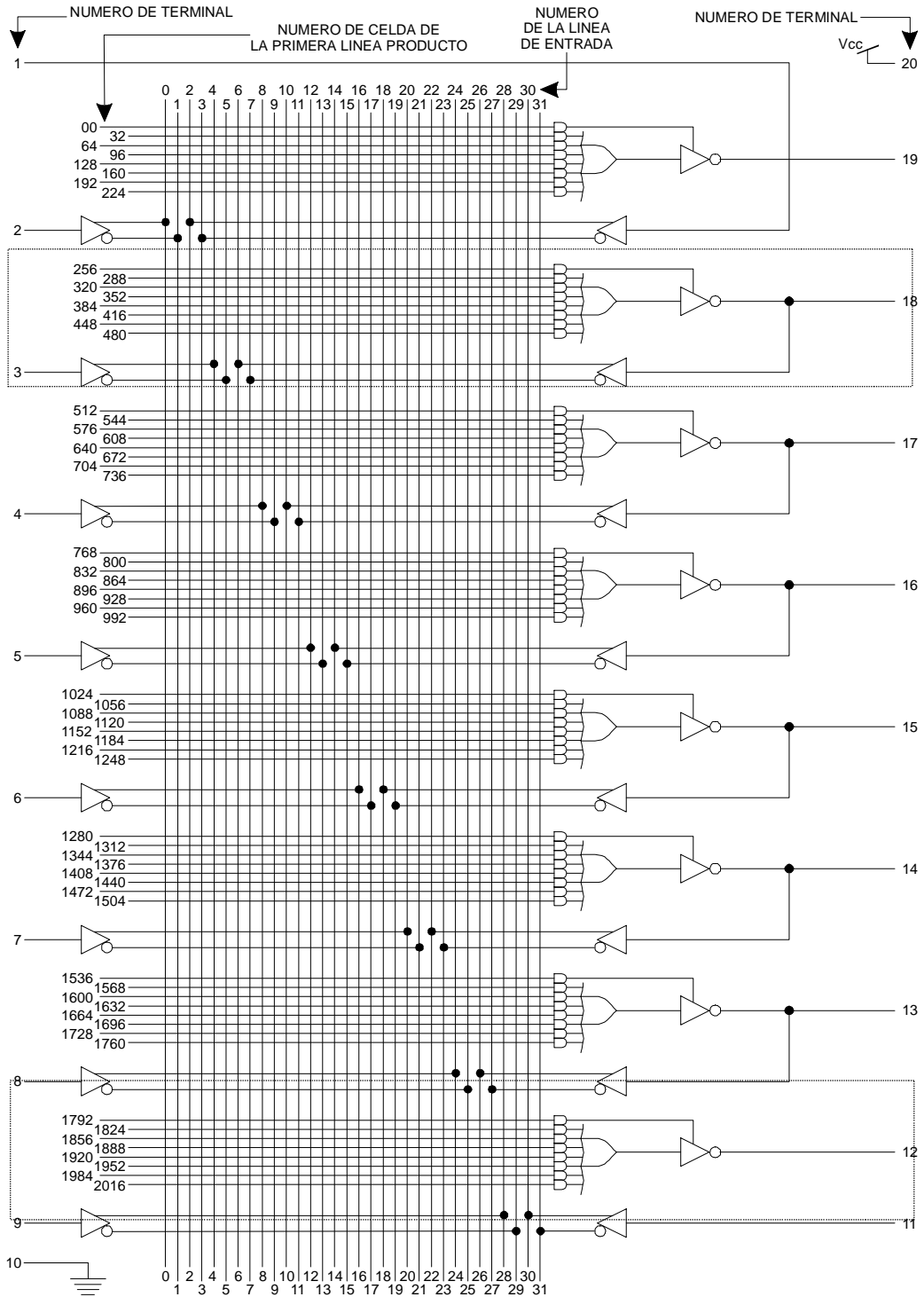


Figura 2-1

ARQUITECTURA DEL PAL16R4

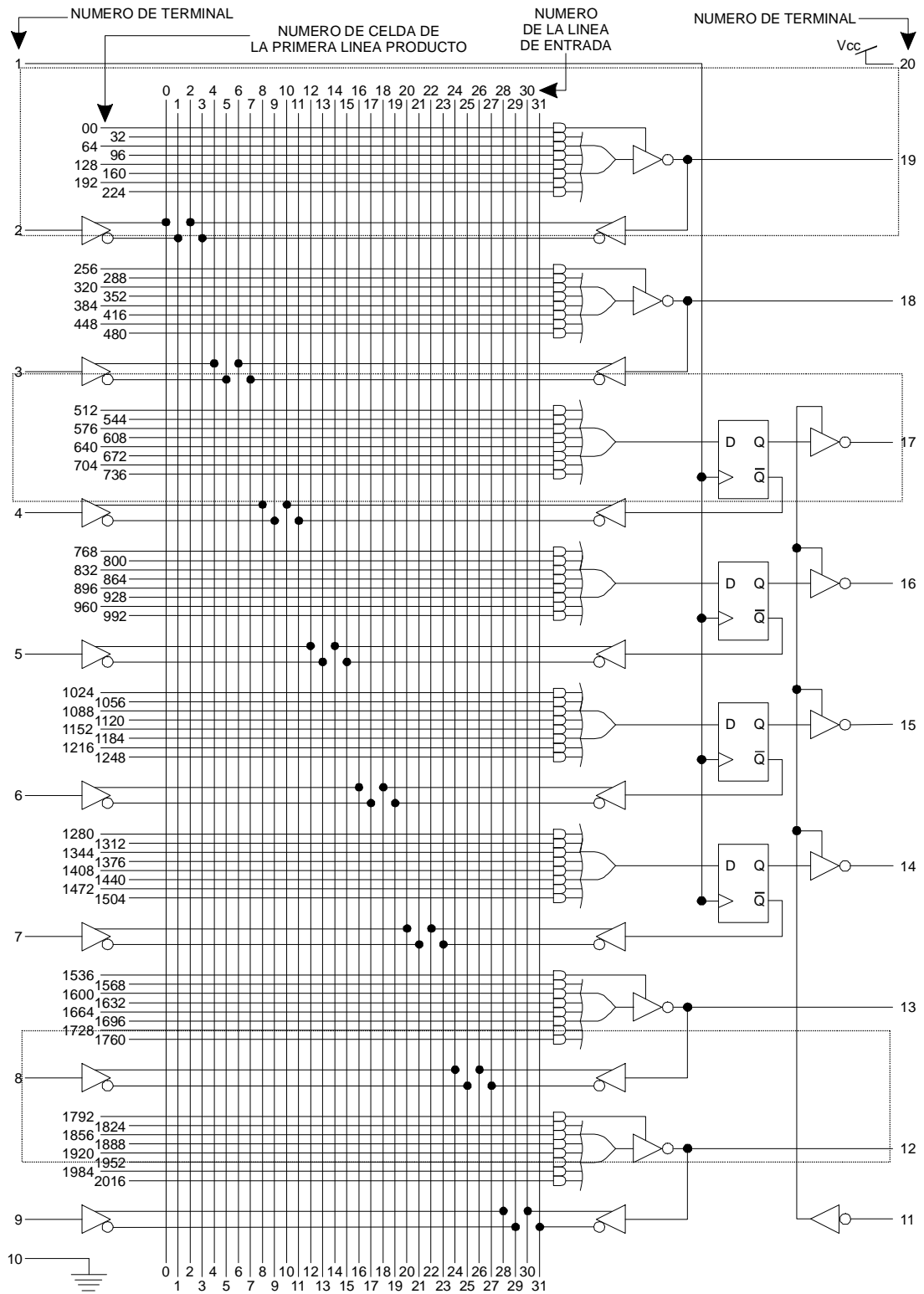


Figura 2-8

ARQUITECTURA DEL PAL16H2

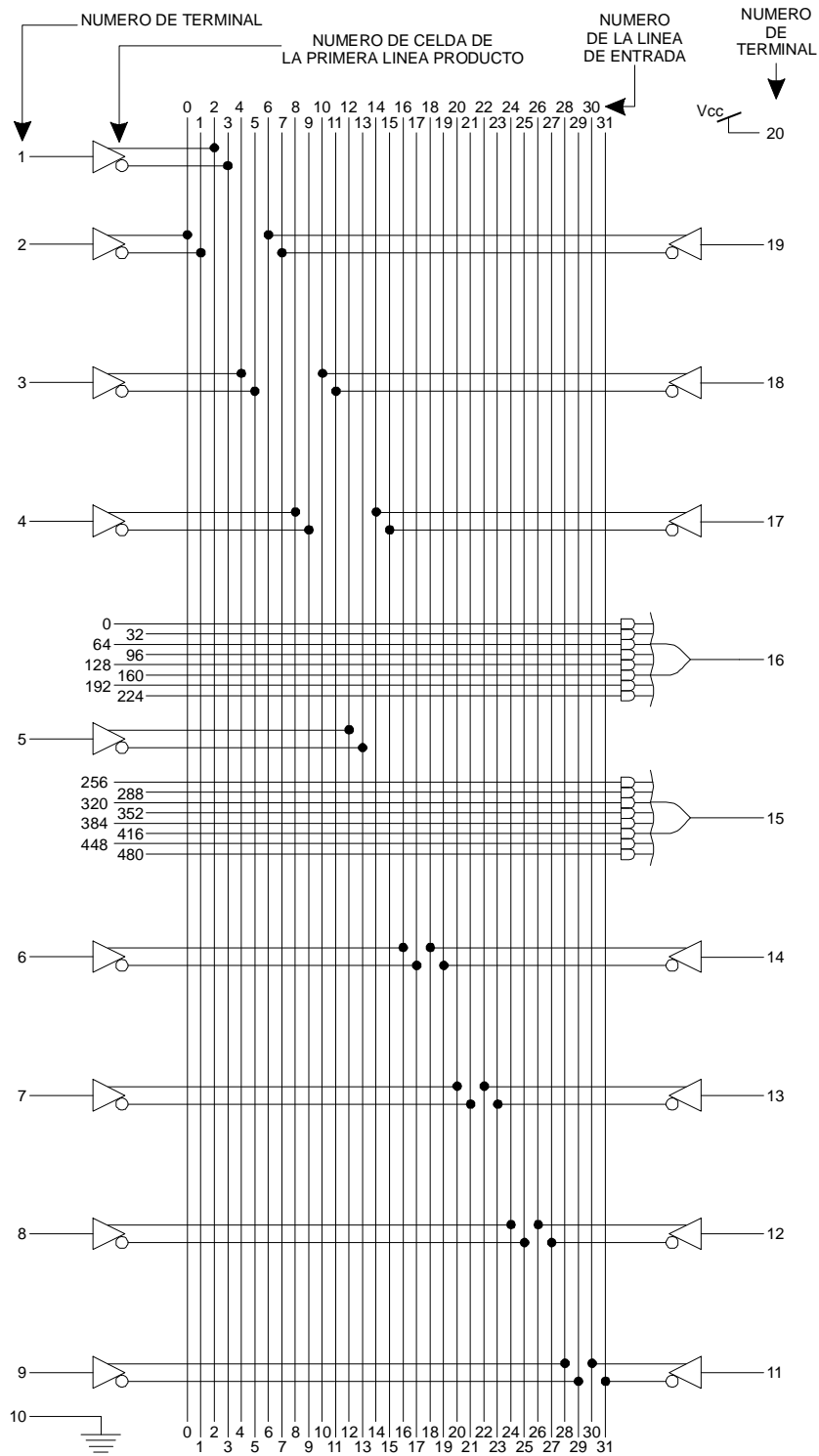


Figura 2-14

GAL16V8

A continuación se muestra un diagrama a bloques de la Estructura Interna del GAL16V8 en presentación DIP (*Dual In Line Package*).

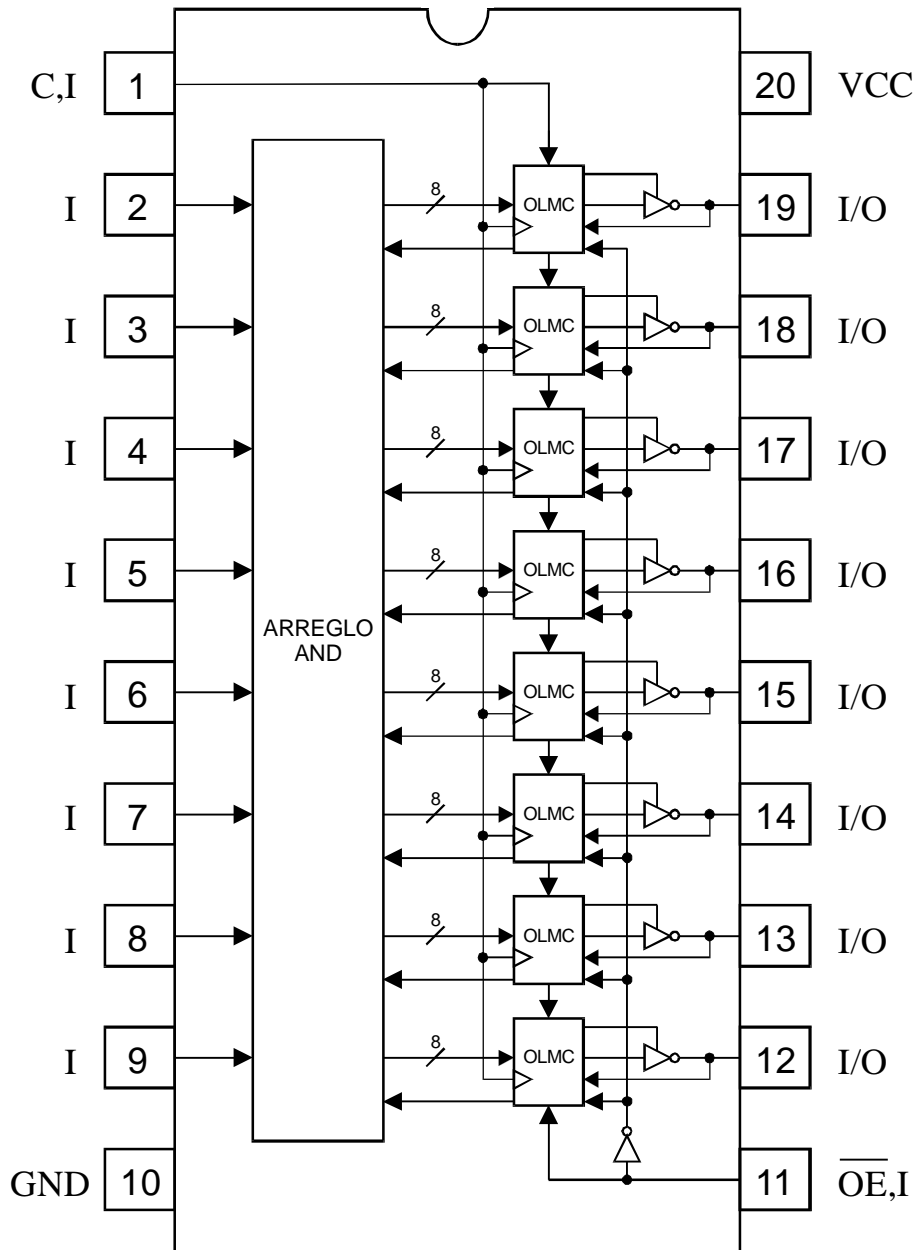


Figura 2-15

DESCRIPCION DEL GAL16V8

La arquitectura interna del GAL16V8 contiene un **PLANO AND** programable compuesto por 64 compuertas y un **PLANO OR** fijo compuesto por 8 compuertas, en forma similar a la arquitectura bipolar de un PAL. El arreglo lógico está organizado con 16 líneas de entrada complementarias hacia el plano programable y que se cruzan con 64 líneas denominadas " términos-producto " ; en cada cruce o intersección de líneas existe una celda EEPROM programable y debido a que son 16 entradas con sus respectivos 16 complementos, hablamos de 32 entradas en forma total y de 64 líneas de " términos-producto " dándonos un total de 2048 celdas programables o sea 32 líneas de entrada x 64 líneas de " términos-producto " = 2048 cruces de líneas. Cada celda programable puede establecer una conexión entre una línea de entrada y un " término-producto ".

Los 64 " términos-producto " están organizados dentro de 8 grupos de salida, con 8 " términos-producto " cada uno . Siete u ocho de los " términos-producto " de cada grupo, son conectados a una compuerta OR, para producir una función lógica de salida; uno de los " términos-producto ", puede ser utilizado para el control del tercer estado en la salida. La función de transferencia fundamental de cada salida del GAL es la ya familiar **suma de productos Booleanos o los productos de sumas** según sea el caso.

Hasta este punto, la similitud entre un PAL y un GAL es tal, que podríamos afirmar que se trata de la misma arquitectura; tan es así, que los Planos AND del PAL16L8 (ver figura 2-1) y del PAL16R4 (ver figura 2-8) son idénticos e incluso poseen el mismo número de terminales externas que el GAL16V8 (ver figuras 2-15 y 2-16) ; por lo tanto se diría que la única diferencia estriba en la capacidad de su reprogramabilidad; siendo esto así, tal vez esta tesis no tendría sentido. Sin embargo, no se explica como es que el GAL16V8 puede entonces emular al PAL16L8 y el PAL16R4, entre otros.

Pues resulta que la verdadera versatilidad de los dispositivos GAL así como su " *magia* " se encuentran en la " **Macroelda Lógica de Salida** " u **OLMC** (por sus siglas en inglés *Output Logic MacroCell*). *¡¡* Sí, sí, ese recuadro adornado con flechas negras que entran y salen, con líneas que van y vienen *!!*.

Como se puede apreciar en la figura 2-16 todas las funciones de salida AND se dirigen hacia una Macroelda en cada grupo de salida, de igual forma se introducen líneas que vienen de la terminal externa # 1 y # 11 así como las líneas de retroalimentación de las terminales externas; así como salidas como son el inversor de salida con capacidad para Tercer-Estado, de la misma forma sale la línea que activa dicha capacidad y también una salida que se dirige de la Macroelda hacia el Plano AND. Cada una de las funciones lógicas AND/OR son alimentadas hacia dentro de una OLMC.

Por esta razón se comenzará por la descripción de la Arquitectura de la **Macroelda** y posteriormente a la descripción de su funcionamiento y en seguida el GAL en forma completa.

ARQUITECTURA DEL GAL16V8

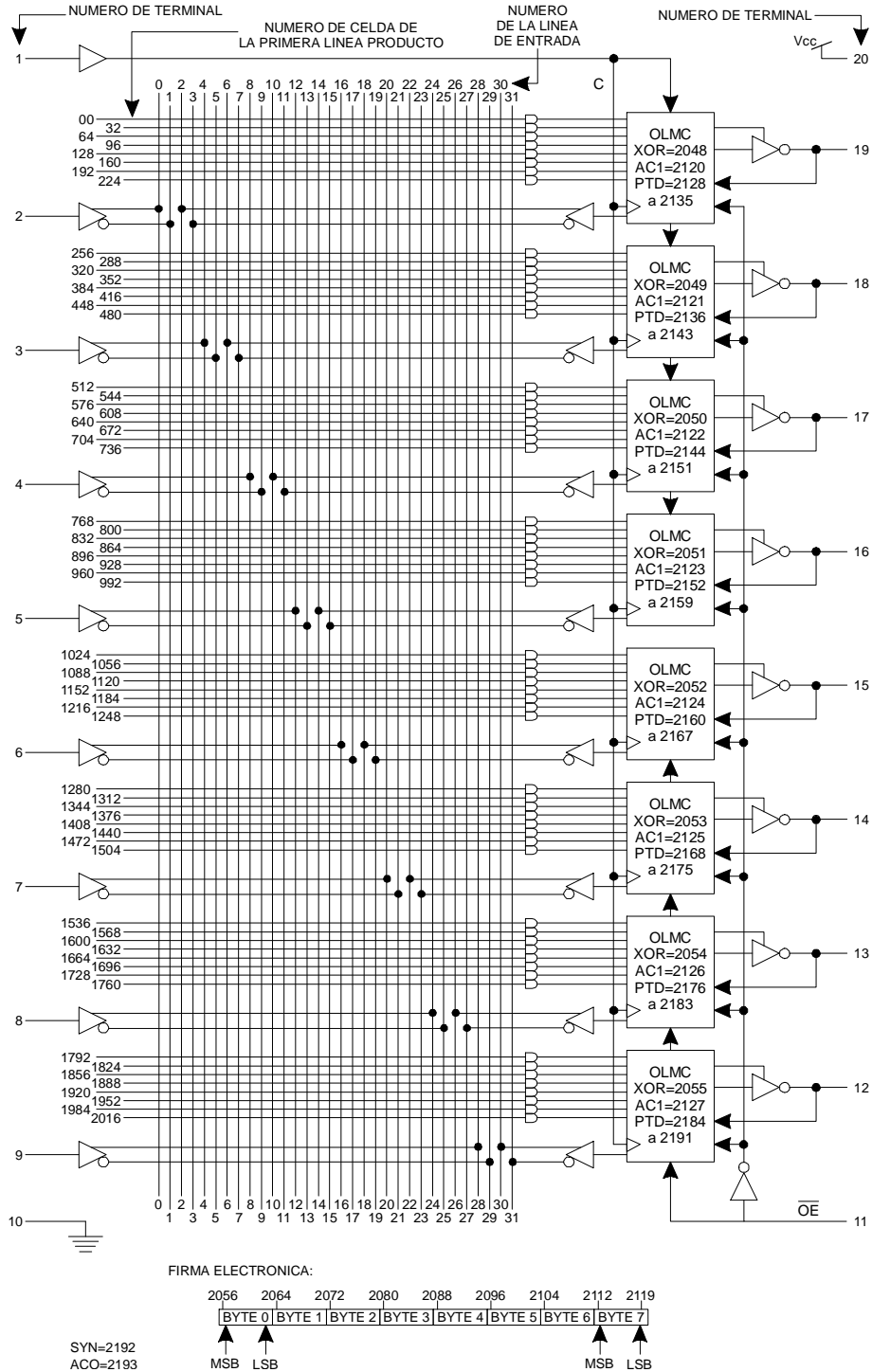


Figura 2-16

ARQUITECTURA Y DESCRIPCION DE LA OLMC

La OLMC esta formada por un conjunto de dispositivos tales como:

- Multiplexores.
- Flip-Flop tipo " D ".
- XOR.
- Buffer inversor , con opción para operar en " Tercer Estado ".
- Dentro de la OLMC se encuentra también la compuerta OR.

Tal y como se aprecia en la siguiente figura.

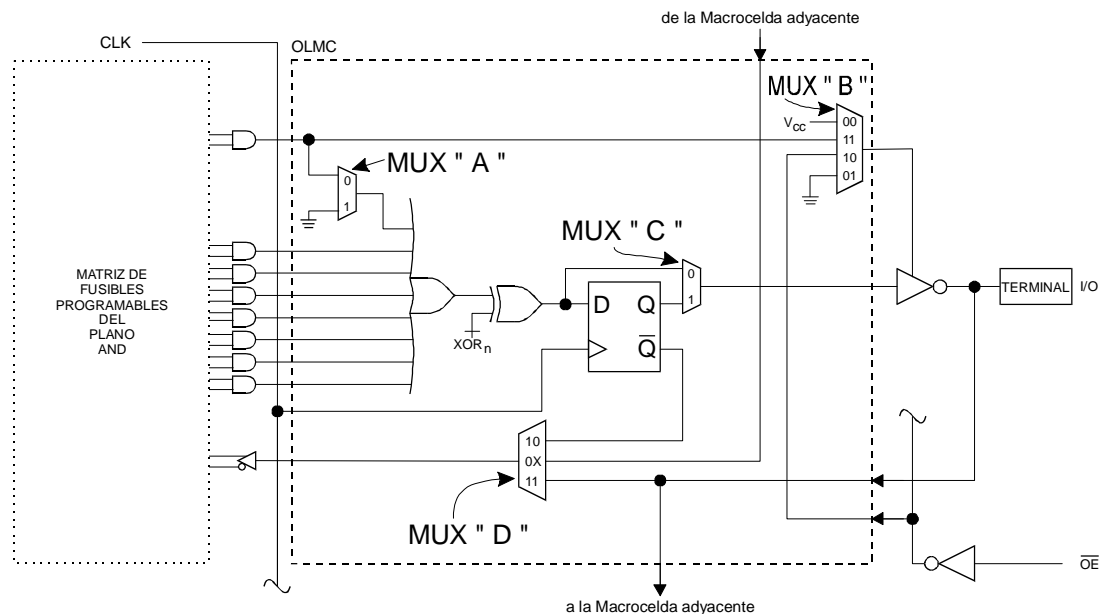


Figura 2-17

En la Macrocella observamos como una de las ocho compuertas del PLANO AND no entra en forma directa hacia la compuerta del PLANO OR, sino que su salida, se conecta a la entrada de dos multiplexores (que en el dibujo están etiquetados como " MUX "); una al **Multiplexor " A "** y otra al **Multiplexor " B "**.

La salida del **Multiplexor " A "**, tiene acceso a una entrada de la compuerta OR y este puede ejecutar dos operaciones que son:

- a) Selecciona la salida de la compuerta AND.
- b) Selecciona GND que equivale a un " 0 " lógico.

y que realizará una a la vez, según el modo de trabajo del dispositivo.

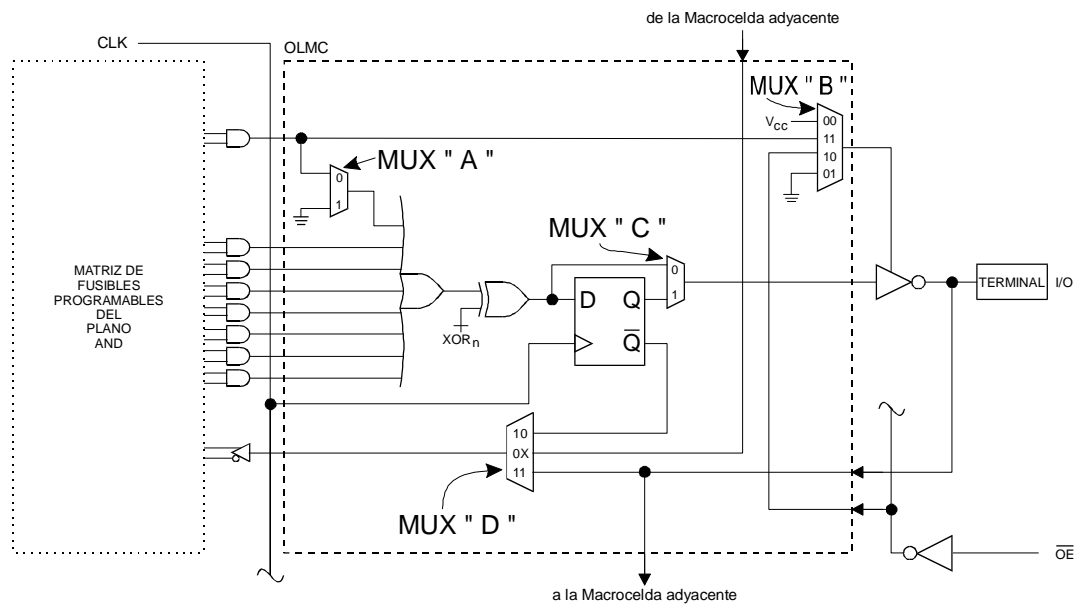


Figura 2-18

El **Multiplexor " B "** determina el Tercer Estado en la terminal de salida a través del buffer inversor, dependiendo de las necesidades que se requieran y para esto tiene cuatro opciones seleccionables:

- Selecciona Vcc que equivale a un " 1 " lógico.
- Selecciona la salida de la compuerta AND.
- Selecciona la salida del inversor de la entrada /OE de la terminal 11 del dispositivo.
- Selecciona GND que equivale a un " 0 " lógico.

El **Multiplexor " C "** selecciona entre dos formas de trabajo del GAL *combinacional* o *secuencial* dependiendo de la opción que se tome:

- Selecciona la salida de la compuerta XOR (*combinacional*).
- Selecciona el flip-flop Tipo " D " (*secuencial*).

El **Multiplexor " D "** se encarga a través de su salida de proporcionar una retroalimentación hacia el PLANO AND, teniendo tres opciones:

- Selecciona la salida /Q del flip-flop Tipo " D ".
- Selecciona la salida del Buffer Inversor.
- Selecciona la salida del Buffer Inversor de una OLMC adyacente.

Bajo el control de una OLMC, cada salida puede ser designada en forma general en:

- **Modo Secuencial** (con registros).
- **Modo Combinacional** (sin registros).

En **Modo Secuencial**, la función lógica de salida pasa a través de un flip-flop tipo " D " disparado por flancos de subida por la señal de reloj.

En **Modo Combinacional**, la función lógica de salida es el resultado de una suma de productos Booleanos en donde no interviene ningún flip-flop.

Adicionalmente, a las funciones lógicas de salida invariablemente del modo, se les puede asignar una polaridad activa-baja o activa-alta (que es ajustada antes del registro); la polaridad activa-baja equivaldría a trabajar con maxi-terminos. Las opciones de las OLMC son seleccionadas usando un cambio en la arquitectura de las celdas de control. Esta arquitectura de las celdas son configuradas automáticamente por el software o hardware de programación.

La **Macrocelda** controla internamente el comportamiento que tendrán las entradas y salidas de las señales entre el arreglo lógico y las **terminales I/O** del dispositivo. Las **terminales I/O** son terminales que pueden ser usadas como entradas **I** (input) o como salidas **O** (output). Estas terminales están comprendidas en el GAL16V8 a partir de la terminal # 12 hasta la # 19, ver figura 2-15.

Para observar con mejor detalle la forma en que se comporta la **Macrocelda** así como sus alcances, en forma particular y posteriormente el conjunto de las ocho; se describirá con detalle el funcionamiento de cada una de las operaciones o configuraciones posibles que la **Macrocelda** puede adoptar.

Y basado en la siguiente figura se procederá a la descripción de las distintas configuraciones de la **Macrocelda**.

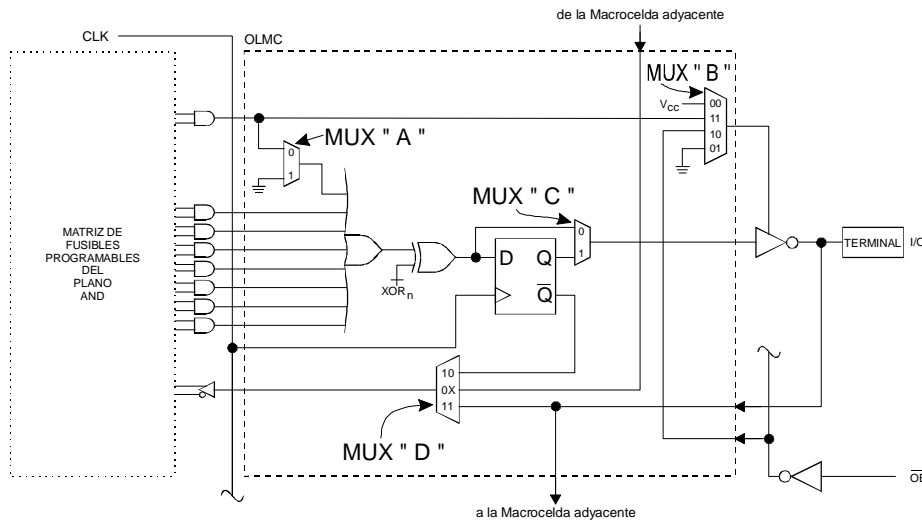


Figura 2-19

Con las siguientes indicaciones para las figuras que representan el diagrama de la OLMC:

Las líneas más oscuras determinan el camino hacia cada una de las compuertas y multiplexores, dependiendo de la selección que se haga con los multiplexores (en cuyo caso se muestran indicando la selección mediante un bloque gris) dicho camino se continuará, a través de los dispositivos necesarios, que se encuentran en la OLMC para lograr cada una de las diferentes funciones que a continuación se ilustran y describen para la OLMC.

CONFIGURACIONES ADOPTADAS POR LA MACROCELDA

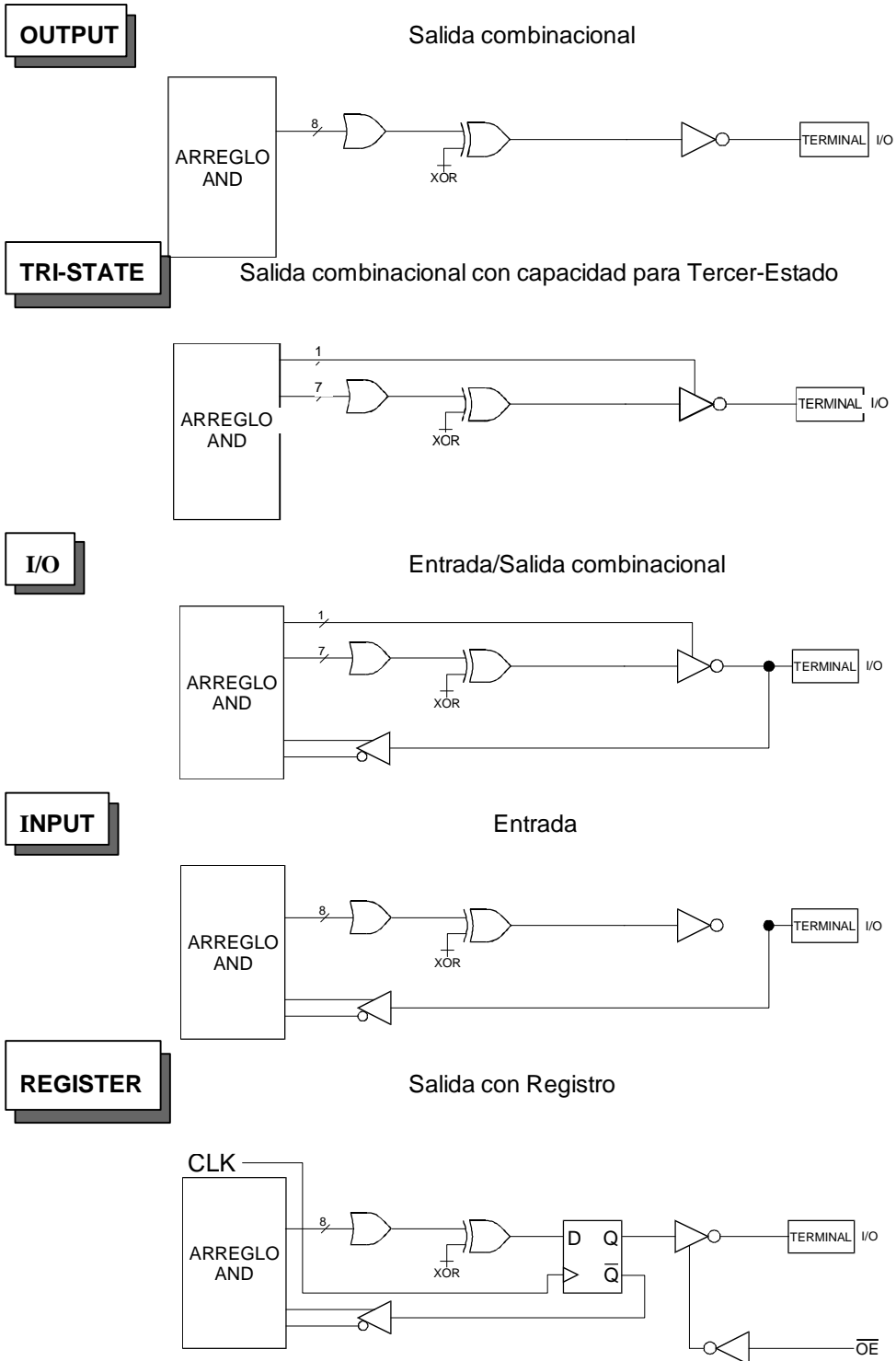


Figura 2-26

SOFTWARE

El Software que se eligió para la programación de los dispositivos GAL es un Software de National Semiconductor Corporation denominado " OPAL " en su versión 1.02 de 1991. Este Software tiene un ambiente de trabajo mediante ventanas, que hace más fácil el manejo del mismo, proporcionando a la vez una organización de las operaciones que se pueden lograr con este paquete.

DESCRIPCION GENERAL

Este paquete de Software ofrece el soporte para algunos PLD's como son los dispositivos MAPL's, GAL's y ECL PAL's. Este paquete es capaz de interpretar Ecuaciones de Algebra de Boole, Máquinas de Estados y Tablas de Verdad para posteriormente en archivos individuales Ensamblarlos y convertirlos en su correspondientes formatos JEDEC. El programa " OPAL " posee un editor de texto con funciones especiales que hacen más versátil el vaciado de datos sobre el mismo, contiene un protocolo de programación accesible al programador de PLD's.

CARACTERISTICAS ESPECIALES

- Minimización de Ecuaciones de Algebra Booleana.
- Asignación automática de terminales.
- Simulación mediante Diagramas de Tiempos.
- Listado con información referente al porcentaje de aprovechamiento del dispositivo así como también la modalidad en que se están utilizando cada una de las terminales.
- Conversión de formatos PAL a GAL.

REQUERIMIENTOS DEL SOFTWARE OPAL

Estos son los requerimientos del sistema para poder operar en una PC.

- Computadora compatible con IBM², ya sea AT o XT con por lo menos 350K de RAM.
- Monitor VGA, EGA, Hercules² (también funcionará con CGA pero no se visualizarán los diagramas de tiempos)
- MS - DOS² 2.1 o superior.

ARCHIVOS DE QUE SE COMPONE ESTA VERSION:

DISCO ? 1

README
ADDENDUM.MAN
INSTALL.EXE
OPL2PLA.EXE
PLA2EQN.EXE
EQN2JED.EXE
ESPRESSO.EXE
FITMAPLXEXE
JED2CKT.EXE
DEVICE.LIB

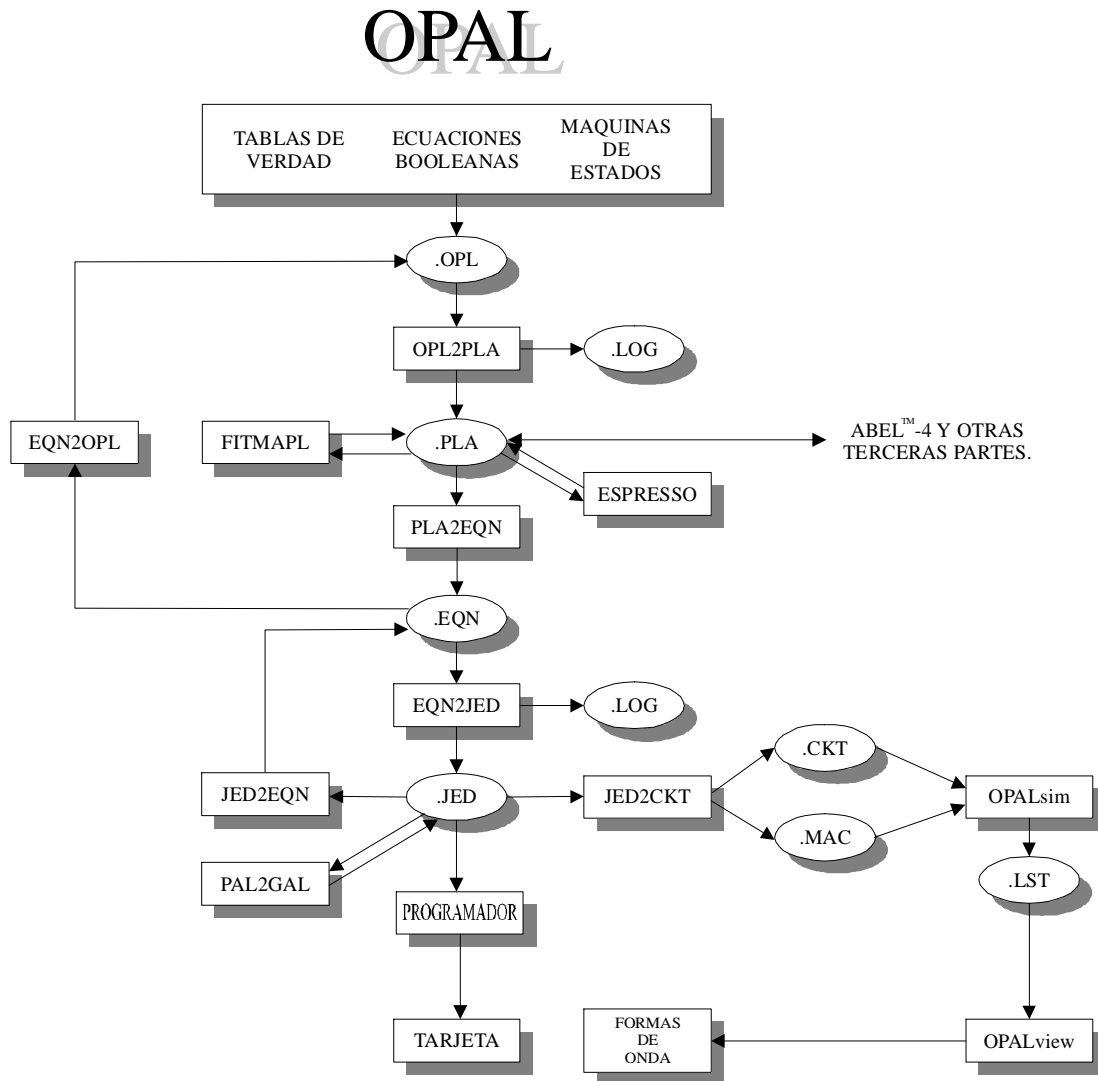
DISCO ? 2

OPAL.EXE
OPALSIM.EXE
OPALVIEW.EXE
OPAL.HLP
OPALVIEW.HLP
EPS.COM
HPS.COM

DISCO ? 3

JED2EQN.EXE
EQN2OPL.EXE
PAL2GAL.EXE
DEMO*. *
EXAMPLES\EQN*. *
EXAMPLES\OPL*. *

DIAGRAMA DE FLUJO QUE DESCRIBE EL FUNCIONAMIENTO COMPLETO DEL SOFTWARE

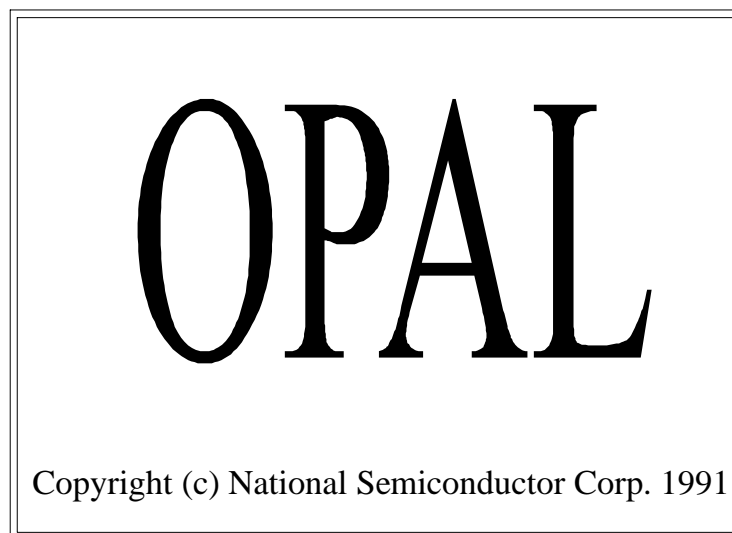


DESCRIPCION DEL SOFTWARE Y DE SUS FUNCIONES

La instalación del programa se realiza mediante el archivo INSTALL.EXE, al igual que muchos otros paquetes de Software. Una vez que el programa ha sido instalado corre con la instrucción OPAL seguido de un ENTER después del prompt ejemplo:

```
C:\?OPAL (ENTER)
```

y a continuación se mostrará una presentación como la siguiente:



VENTANA PRINCIPAL.

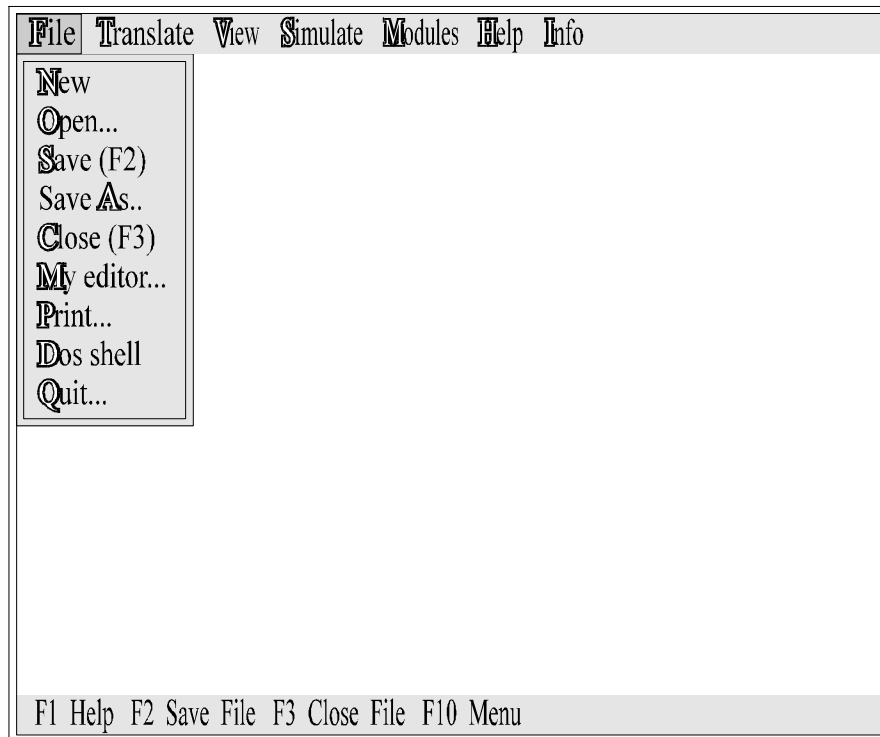
Ya que el programa ha sido cargado, estará listo para trabajar dentro de él cuando se muestre la pantalla principal:



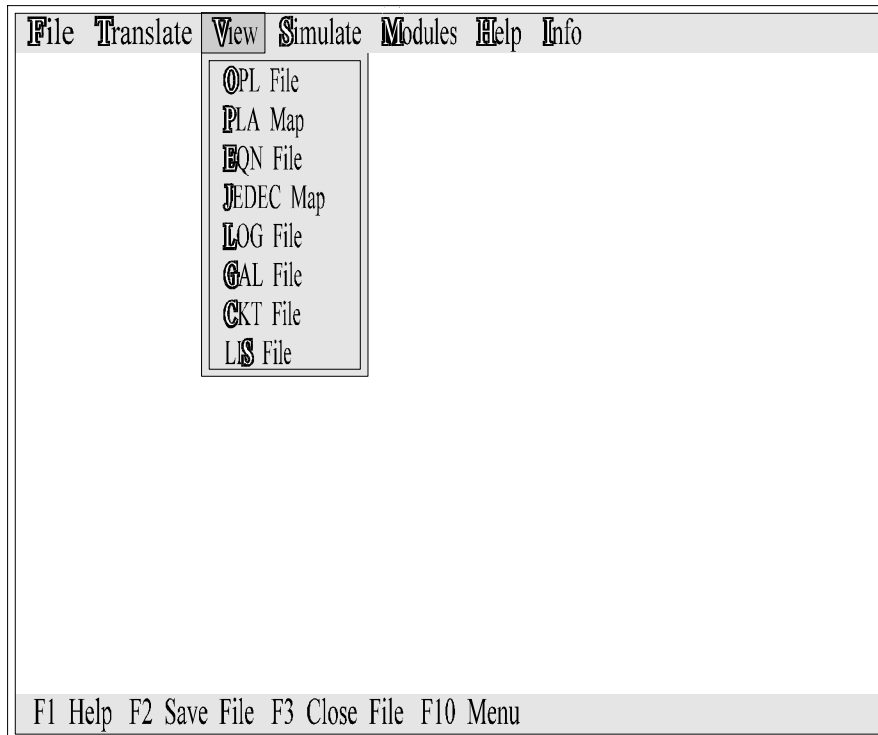
En la parte superior se encuentran los “ Títulos de las Ventanas ”, que contienen las opciones de trabajo del programa. Cada uno de los “ Títulos de las Ventanas ”, tiene una letra resaltada; lo que indica, que dicha ventana podrá ser seleccionada, con solo teclear la letra resaltada del “ Título ”.

En la parte inferior se encuentran teclas rápidas para algunas funciones y que ejecutaran la operación señalada con solo oprimir la tecla de función, por ejemplo:

F1 HELP nos permitirá visualizar la ayuda en cada opción con solo teclear F1.

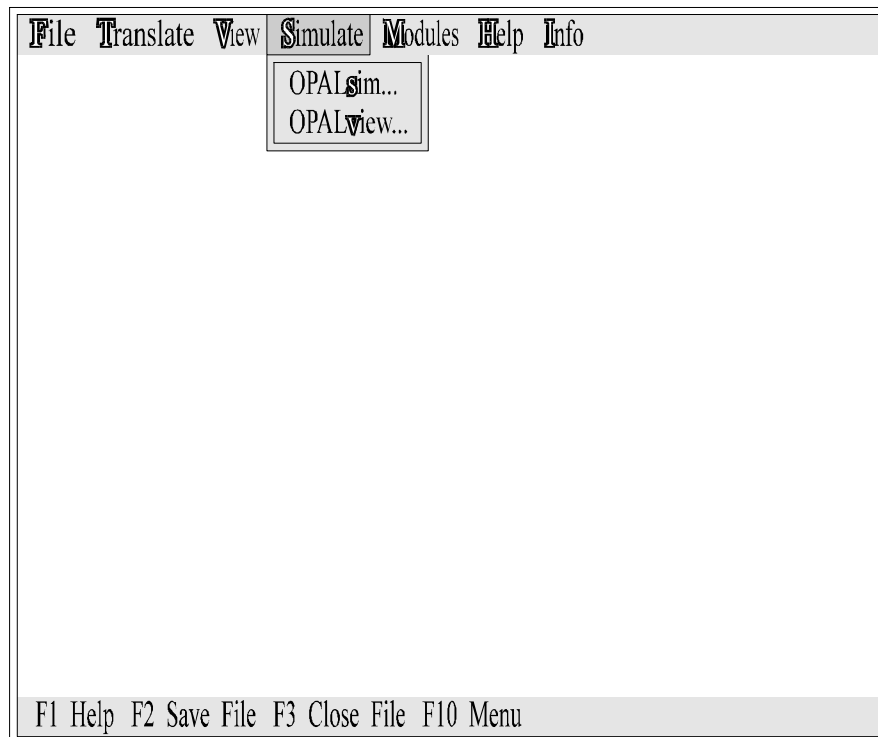
VENTANA FILE.

La ventana **File** contiene nueve opciones y al igual que los " Títulos de las Ventanas " cada una de las opciones tiene también una letra resaltada y al oprimir alguna de ellas se seleccionara la función correspondiente, otra forma de seleccionar las opciones se hace mediante el cursor (flechas del teclado) seguidas de un ENTER; las teclas ?? seleccionan la opción de la ventana y las teclas ? ? seleccionan otra ventana.

VENTANA VIEW:

La ventana **View** mediante la *ventana de búsqueda* (ver la opción **Open...** de la ventana **File**) se hace la selección rápida de archivos según su extensión:

- OPL File: ? busca archivos con cualquier nombre; pero, con extensión OPL (?.opl).
- PLA Map: ? busca archivos con cualquier nombre; pero, con extensión PLA (?.pla).
- EQN File: ? busca archivos con cualquier nombre; pero, con extensión EQN (?.eqn).
- JEDEC MAP: ? busca archivos con cualquier nombre; pero, con extensión JEDEC (?.jed).
- LOG File: ? busca archivos con cualquier nombre; pero, con extensión LOG (?.log).
- GAL File: ? busca archivos con cualquier nombre; pero, con extensión GAL (?.gal).
- CKT File: ? busca archivos con cualquier nombre; pero, con extensión CKT (?.ckt).
- LIS File: ? busca archivos con cualquier nombre; pero, con extensión LIS (?.lis).

VENTANA SIMULATE:

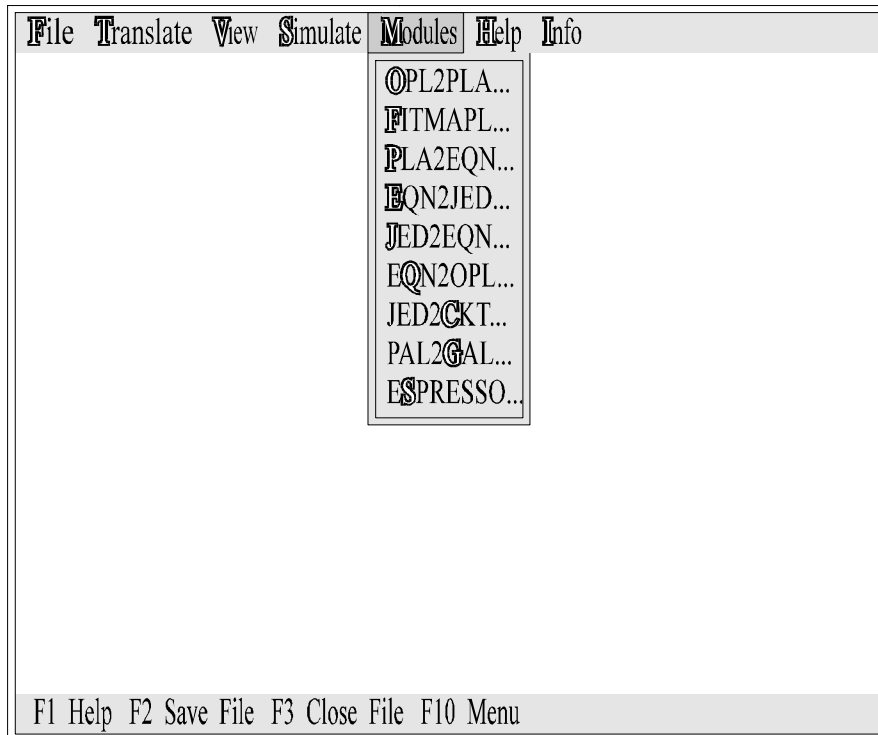
La ventana **Simulate** permite realizar la simulación del circuito que se acaba de diseñar dentro de OPAL, dentro de esta ventana se encuentran dos opciones y que con la ayuda de la *ventana de búsqueda* se podrán efectuar más rápidamente la selección de los archivos de simulación.

Opalsim...

Después de la búsqueda del archivo de simulación (?.ckt), se procederá a la generación de un archivo (?.Lst) basado en la Tabla de Estados anexada al archivo (?.ckt) donde se podrán visualizar los diagramas de tiempos de la Tabla mencionada.

Opalview...

Mediante esta opción se efectúa la visualización en pantalla de los diagramas de tiempos, para esto es necesario que exista el archivo que se creó con la opción anterior (?.Lst).

VENTANA MODULES:

La ventana **Modules** posee nueve opciones, cabe hacer mención que es una de las ventanas más importantes dentro del paquete (Software) ya que dentro de ella se realiza la conversión (Ensamble y compilación) entre archivos y que serán descritas a continuación:

FORMATO Y PROTOCOLO DE PROGRAMACION PARA EL SOFTWARE OPAL USANDO ECUACIONES DE ALGEBRA DE BOOLE.

El archivo que se ha de crear bajo el modo de Ecuaciones de Algebra de Boole, deberá tener la extensión EQN (abreviación de Equations), para hacer esto se debe hacer lo siguiente:

- a) Abrir ventana **File**, seleccionar la opción **New**.
- b) Seleccionar la opción salvar como " **Save As...** ".
- c) Darle nombre al archivo, no exceder 8 caracteres y agregar la extensión EQN (Ejemplo1.EQN). Teclear < Enter >.
- d) Para actualizar nuestro archivo conforme se realiza el vaciado de los datos basta con teclear F2.

El formato del archivo EQN, está dividido en 3 secciones que son:

- I. Bloque de Encabezado.
- II. Bloque de Declaraciones y
- III. Bloque de Ecuaciones.

Antes de comenzar con la descripción del Bloque de Encabezado; será necesario, conocer algunas palabras clave y el protocolo para los comentarios.

Existen solo 3 palabras clave, que no deberán ser empleadas como Etiquetas o identificadores en este archivo y son:

- | | |
|--------------------|---|
| ? CHIP | Que señala el final del Bloque de Encabezado y el principio del Bloque de Declaraciones; además de dar de alta al dispositivo de trabajo. |
| ? EQUATIONS | Que indica el final del Bloque de Declaraciones y el principio del bloque de Ecuaciones. |
| ? GLOBAL | Que marca las aplicaciones de una operación " Global o General ", que ha sido, declarada por otros identificadores. |

Comentarios:

Estos nos sirven para hacer alguna nota, explicación o descripción dentro del archivo. Los comentarios pueden ser colocados en cualquier parte o lugar del archivo EQN; sin importar, el Bloque donde se coloquen.

Existen dos formatos para señalar los comentarios dentro del archivo y son:

- ? Los Comentarios por Líneas; en cuyo caso se deberá iniciarse con un punto y coma (;) antes del comentario. Ejemplo:

```
;De esta forma se colocan comentarios en una línea.
```

- ? Los Comentarios por Multilíneas, que deberá iniciarse con una llave ({) y finalizar con otra (}). Ejemplo:

```
{ En este tipo de comentarios
  se pueden emplear
  varias líneas }
```

Bloque de Encabezado.

El encabezado es cualquier texto antes de la palabra clave "CHIP" y este es colocado como encabezado en los archivos generados posteriormente; siempre y cuando este exista. Ejemplo:

```

CIUDAD          Guadalajara, Jalisco, México.
LUGAR           Universidad de Guadalajara.
CAMPUS          Centro Universitario de Ciencias Exactas e Ingenierías.
CARRERA         Ingeniería en Comunicaciones y Electrónica.
AUTOR           José Miguel Morán Loza
DISEÑO          Decodificador de Binario de BCD a 7 Segmentos.

```

Bloque de Declaraciones.

Este Bloque inicia con la palabra clave "CHIP", seguido por una etiqueta (opcional) y el nombre del dispositivo.

A continuación (en la siguiente línea) se colocan la lista de terminales (opcional) y Directivas. Formato:

- **CHIP** Etiqueta Nombre-del-Dispositivo.
- **Lista de Terminales.**
- **Directivas.**

CHIP - Palabra clave que asigna y da de alta el dispositivo, con el cual se pretenden trabajar las ecuaciones y la cual tiene dos formatos:

```

10.-                               20.-
CHIP EJEMPLO11 GAL16V82           CHIP GAL16V8

```

A continuación debe hacerse una asignación de terminales mediante un listado que se puede efectuar de dos formas diferentes:

1⁰.- Caso de asignación de terminales.

```

1 2 3 4 5 6 7 8 9 GND
11 12 13 14 15 16 17 18 19 VCC

```

En donde los números, serán sustituidos por las etiquetas asignadas por el usuario, para las variables de entrada y salida respetando los espacios entre ellas; por ejemplo:

```

1 2 3 4 A B C 8 9 GND
11 Q1 13 Q2 15 ACARREO 17 18 19 VCC

```

¹Se recomienda tomar esta opción; en cuyo caso, la etiqueta deberá ser idéntica, al nombre del archivo sin la extensión. Esto evitará algunos problemas que podría presentarse al momento de convertir el archivo a otro formato.

² Si no se desea especificar el nombre del dispositivo colocar la palabra "UNKNOWN" (desconocido) en su lugar.

A las variables, se les pueden asignar con minúsculas, mayúsculas, acompañadas si se desea de números, para distinguir variables similares entre sí; por ejemplo:

a ? A a ? a1 A ? A1 a1? A1

³De esta forma:

```
1 2 3 4 a a1 b b1 9 GND
11 12 13 A 15 16 A1 18 19 VCC
```

2⁰ Caso de asignación de terminales.

La asignación de las variables se hace igualando la etiqueta con la terminal, ejemplo:

```
A=5 B=6 C=7
Q1=12 Q2=13 ACARREO=16
```

La ventaja que ofrece esta opción es la de no anotar las terminales, que son innecesarias; así como omitir, las terminales de alimentación (VCC y GND).

DIRECTIVAS @

Son directivas que van inmediatamente, después de la lista de terminales y estas son:

- @ DEFINE Que se utiliza para agrupar expresiones que se repetirán, en el bloque de ecuaciones, ejemplo:
 - @ DEFINE Topo " A1*/B2*C *D1 "
- @ UES Sirve para colocar la Firma Electrónica (User Electronic Signature) en el dispositivo⁴, ejemplo:
 - @ UES MORAN L.
- @ iLCH Declara variables (identificadores) como entradas con LATCH.
 - @ iLCH 4 i1 i2 i3 i4.
- @ iREG Señala variables de entrada con registro.
 - @ iREG 2 i5 i6

Las últimas dos solo son aplicables en el caso del GAL6001.

³Cabe hacer mención que no es muy recomendable hacer asignación, de minúsculas y mayúsculas, como en el primer y cuarto caso; debido a que el simulador, no distingue entre minúsculas y mayúsculas, lo cual podría causar confusión, durante la interpretación del Diagrama de Tiempos.

⁴La cual se puede colocar en código ASCII (8 caracteres), en Hexadecimal o Binario.

BLOQUE DE ECUACIONES.

El Bloque de Ecuaciones inicia con la palabra clave " EQUATIONS " y a continuación se declaran las ecuaciones del usuario.

El vaciado de las ecuaciones es muy similar al que estamos acostumbrados a manejar, en el Algebra de Boole; aunque, con los respectivos símbolos para el archivo EQN, que se muestran a continuación:

TABLA 3-1

OPERADOR	EJEMPLO	OPERACION	PRECEDENCIA
/	/A	INVERSOR	4 (MAYOR)
!	!A	INVERSOR	4 (MAYOR)
*	A*B	AND	3
&	A&B	AND	3
 	A B	OR	2
+	A+B	OR	2
^	A^B	XOR	1
\$	A\$B	XOR	1
::+	A::+B	XOR	1
=	A=B	ASIGNACION (COM)	0
:=	A:=B	ASIGNACION (SEC)	0 (MENOR)

Los operadores en negritas son los que se usarán de aquí en adelante y el resto serán considerados como símbolos alternativos.

El orden de precedencia, nos sirve para identificar y tener presente, la prioridad de un operador al realizar una operación; por ejemplo:

$$D = /A*B$$

Según el orden de precedencia el programa efectuará la operación de la ecuación como sigue:

- a) Inversor /A
- b) Operación AND /A*B
- c) Asignación D = /A*B

- a) Realiza la operación de Inversor en la variable A
- b) Realiza la operación AND en las variables /A*B
- c) Realiza la operación de asignación " = " (combinacional) a las operaciones realizadas anteriormente.

Es obvio que los operadores se repetirán; en cuyo caso, el orden de precedencia será tomado por el programa, de izquierda a derecha en una ecuación, por ejemplo:

$$D = /A*B*/C + A*C$$

Se efectuará la operación de inversor sobre A, en seguida la AND con /A*B, después de la operación del inversor con C, continuando con la operación AND /A*B*/C, según el orden de precedencia, se realiza la operación AND con las variables A*C, con la posterior operación OR /A*B*/C + A*C y finalmente la operación de asignación (combinacional en este caso), D = /A*B*/C + A*C.

POLARIDAD DE UNA SEÑAL VARIABLE (ENTRADA Ó SALIDA)

Recordemos que una función de salida puede tener polaridad negativa(activa-baja) o positiva (activa-alta), ejemplo:

$/D = A + B$ Polaridad negativa (activa-baja)

$D = A + B$ Polaridad positiva (activa-alta)

Dentro del programa existen dos factores que afectan la polaridad de una ecuación y son:

- 1) La lista de terminales (en el Bloque de Declaraciones).
- 2) La polaridad asignada en la ecuación (Bloque de Ecuaciones).

La polaridad será determinada según la Tabla 3-2.

TABLA 3-2

		Ecuaciones Booleanas	
		S	/S
Lista de terminales	S	Positiva	Negativa
	/S	Negativa	Positiva

Ejemplos:

? Se define una variable de salida " S ". En la lista de terminales se asigna como S; pero, en la ecuación se asigna como /S= Según la Tabla 3-2, la polaridad real de salida es negativa.

? Un ejemplo más sería, en el cual la variable de salida es asignada como /S, en la lista de terminales y como /S en la ecuación. Revisando la Tabla 3-2 observaremos que la polaridad real es positiva.

INSTRUCCIONES ESPECIALES

Todas las instrucciones mostradas en esta sección sirven para activar algunas funciones de carácter especial, que poseen algunos dispositivos PAL y GAL; por ejemplo:

Después de haber analizado a los PAL's y GAL's, recordaremos que en las terminales asignadas como O e I/O, son terminales en las que se puede activar el OE (Output Enable), para aprovechar la condición de TRI-STATE (Tercer estado) disponible en dichas terminales. Cabe hacer mención que existen terminales en tales dispositivos, que activan el OE en todas las terminales de salida con registro; tal es el caso de la terminal # 11 en el PAL16R4, PAL16R6, PAL16R8 y GAL16V8. Pero esto solo en Modo Secuencial; debido al arreglo lógico en la arquitectura de algunos dispositivos, como en el caso de los antes mencionados.

Sin embargo nos preguntaríamos ¿ Cómo podré activar el OE en Modo Combinacional ?, la respuesta ha esta pregunta y a otras que pudieran generarse serán saciadas en esta parte.

Para entender de una forma sencilla, el formato para estas instrucciones especiales, aprovecharemos el siguiente ejemplo:

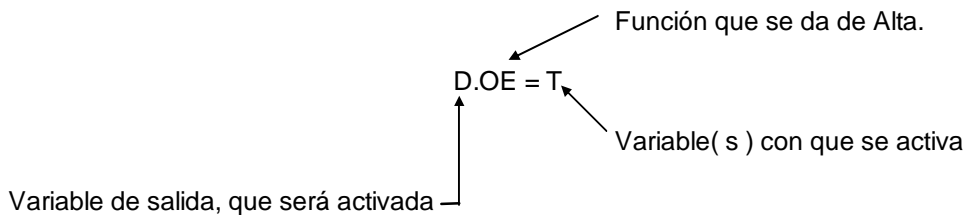
$$D = A*/B + /A*B$$

La salida D realiza una operación XOR (en la práctica la función de salida puede ser cualquier operación) “ Combinacional ”.

$$D.OE = T$$

Esta Instrucción, declara la salida D con habilitación para OE, cuando la variable sea igual a T = 1, así cuando /T entonces tendremos TRI-STATE.

Formato para la Instrucción:



Lo que significa que la función que se desea activar, se encuentra inmediatamente después del punto. Algunas funciones extras son mostradas en la Tabla 3-3.

TABLA 3-3

FUNCION		EJEMPLO	DISPOSITIVOS GAL DONDE ES FACTIBLE EMPLEAR LA FUNCION
OE	OUTPUT ENABLE	X.OE = Z	GAL16V8, GAL20V8, GAL22V10, GAL20RA10 Y GAL6001
AR	ASYNCHRONOUS RESET	X.AR = Z	GAL22V10, GAL20RA10 Y GAL6001
SP	SYNCRONOUS PRESET	X.SP = Z	GAL22V10 Y 6001
C	CLOCK	X.C = Z	GAL20RA10
PR	PRESET	X.PR = Z	GAL20RA10
RE	RESET	X.RE =Z	GAL20RA10

Después de la descripción del formato y protocolo de programación se anexa un ejemplo de un archivo con extensión EQN, el cual complementará la idea de lo descrito en las páginas anteriores. Sin embargo, no será suficiente para comprenderlo todo; por lo cual, se deberá completar con los archivos EQN presentados en el Capítulo IV.

Archivo EQN :

AUTOR: José Miguel Morán Loza.
 DISEÑO: Contador BCD de 7 Segmentos Ascendente/Descendente con Acarreo
 COMENTARIOS: Display Anodo Común
 LUGAR: Centro Universitario de Ciencias Exactas e Ingenierías.
 CARRERA: Ingeniería en Comunicaciones y Electrónica.

CHIP CONT_97A GAL16V8

CLK=1 Up=2 Reset=5 c=12 d=13 e=14 g=15
 f=16 a=17 b=18 Carry=19

EQUATIONS

```

a := /Up * /Reset * /a * b * /c * /d * e * /g * /f
    + /Up * /Reset * /a * /b * /c * /d * /e * g * /f
    + /Up * /Reset * /a * /b * c * /d * /e * /g * f
    + /Up * /Reset * /a * /b * /c * /d * e * /g * f

b := /Up * /Reset * /a * /b * /c * d * e * g * f
    + /Up * /Reset * /a * b * /c * /d * /e * /g * /f
    + /Up * /Reset * a * /b * /c * d * e * /g * /f
    + /Up * /Reset * /a * b * /c * /d * e * /g * /f

c := /Up * /Reset * /a * /b * /c * /d * e * /g * f
    + /Up * /Reset * a * /b * /c * d * e * g * f

d := /Up * /Reset * /a * /b * c * /d * /e * /g * f
    + /Up * /Reset * /a * b * /c * /d * e * /g * /f
    + /Up * /Reset * /a * /b * /c * /d * /e * g * /f
    + /Up * /Reset * /a * /b * /c * /d * e * /g * f
    + /Up * /Reset * /a * b * /c * /d * /e * /g * /f
    + /Up * /Reset * /a * /b * /c * /d * /e * /g * /f

e := /Reset * a * /b * /c * d * e * /g * /f
    + /Reset * /a * /b * c * /d * /e * /g * f
    + /Up * /Reset * /a * b * /c * /d * /g * /f
    + /Up * /Reset * /a * /b * /c * /d * e * /g * f
    + /Reset * /a * /b * /c * /d * /e * /f
    + /Reset * /a * /c * /d * /e * /g * /f

g := /Up * /Reset * /a * b * /c * /d * /e * /g * /f
    + /Up * a * /b * /c * d * e * g * f
    + /Up * /a * /b * c * /d * /e * /g * f
    + /Up * /a * /b * /c * /d * e * /g * /f
    + /Up * /a * /b * /c * /d * /e * g * /f
    + /Up * /a * /b * /c * /d * /e * /g * /f
    + Reset

f := /Up * /Reset * a * /b * /c * d * e * g * f
    + /Up * /Reset * a * /b * /c * d * e * /g * /f
    + /Up * /Reset * /a * b * /c * /d * /e * /g * /f
    + /Up * /Reset * /a * /b * /c * /d * /e * g * /f
    + /Reset * /a * /b * c * /d * /e * /g * f
    + /Up * /Reset * /a * /b * /c * /d * e * /g * f
    + /Up * /Reset * /a * /b * /c * /d * /e * /g * /f

Carry := /Up * /a * /b * /c * /d * e * /g * /f
    + /Up * /a * /b * /c * /d * /e * g * /f

Carry.C = CLK

g.C = CLK

f.C = CLK

e.C = CLK

d.C = CLK

c.C = CLK

b.C = CLK

a.C = CLK

```

1. Decodificador de Binario a Hexadecimal para Display de 7 segmentos utilizando el GAL16V8.

Esta es una aplicación ya implementada y en uso en nuestra comunidad. Sin embargo deseo retomarla para hacer algunas anotaciones posteriores.

Es sabido que esta aplicación se a vuelto bastante popular, debido a que los dispositivos DM9368 y DM8374 han sido descontinuados. Motivo por el cual es muy difícil, si no es que imposible; encontrarlos en el Mercado De Componentes Electrónicos.

? Planteamiento

Se deberá realizar un decodificador de binario a hexadecimal para Display de 7 segmentos.

? Diseño desarrollo:

Es obvio que se tienen que manejar cuatro variables de entrada en binario; para poder así, cubrir los 16 caracteres que presenta el sistema hexadecimal. Se requieren de 7 salidas para manejar los 7 segmentos del display.

Para el desarrollo de este diseño se empleará un Display de Cátodo Común.

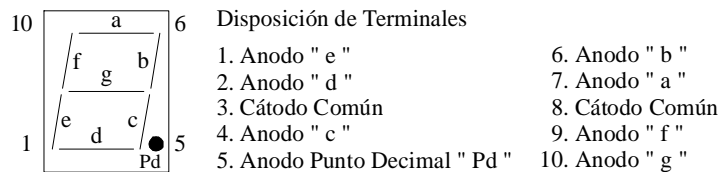


Figura 41

Tabla de Verdad											
Variables de Entrada				Variables de Salida para Display							
MSB			LSB	Segmentos							
D	C	B	A	a	b	c	d	e	f	g	#
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9
1	0	1	0	1	1	1	0	1	1	1	A
1	0	1	1	0	0	1	1	1	1	1	B
1	1	0	0	1	0	0	1	1	1	0	C
1	1	0	1	0	1	1	1	1	0	1	D
1	1	1	0	1	0	0	1	1	1	1	E
1	1	1	1	1	0	0	0	1	1	1	F

Mediante Mapas de Karnaugh y usando lógica negativa (se realiza la reducción y obtención de ecuaciones del álgebra de Boole:

		BA			
		00	01	11	10
DC	00		0		
	01	0			
	11		0		
	10			0	

a

		BA			
		00	01	11	10
DC	00				
	01		0		0
	11	0		0	0
	10			0	0

b

		BA			
		00	01	11	10
DC	00				0
	01				
	11	0		0	0
	10				

c

		BA			
		00	01	11	10
DC	00		0		
	01	0		0	
	11			0	0
	10				0

d

		BA			
		00	01	11	10
DC	00		0	0	
	01	0	0	0	
	11				
	10		0		

e

		BA			
		00	01	11	10
DC	00		0	0	0
	01			0	
	11		0		
	10				

f

		BA			
		00	01	11	10
DC	00	0	0		
	01			0	
	11	0			
	10				

g

Obteniendo:

$$\bar{a} ? \bar{D}\bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{B}\bar{A}$$

$$\bar{b} ? \bar{D}\bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{A} ? \bar{C}\bar{B}\bar{A} ? \bar{D}\bar{B}\bar{A}$$

$$\bar{c} ? \bar{D}\bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{A} ? \bar{D}\bar{C}\bar{B}$$

$$\bar{d} ? \bar{D}\bar{C}\bar{B}\bar{A} ? \bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{B}\bar{A}$$

$$\bar{e} ? \bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{B} ? \bar{D}\bar{A}$$

$$\bar{f} ? \bar{D}\bar{C}\bar{A} ? \bar{D}\bar{C}\bar{B} ? \bar{D}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{B}\bar{A}$$

$$\bar{g} ? \bar{D}\bar{C}\bar{B} ? \bar{D}\bar{C}\bar{B}\bar{A} ? \bar{D}\bar{C}\bar{B}\bar{A}$$

Se puede incluir una variable más, que nos permita comprobar el correcto funcionamiento del Display después de hacer las conexiones pertinentes del GAL16V8 al Display.

Esta variable será P, la cual encenderá todos los Segmentos del Display al colocarle un 1 lógico a su entrada. Por lo que se deberá agregar a cada producto de los minterminos la variable “/P”.

El siguiente paso consistirá en vaciar dichas ecuaciones al programa OPAL como se muestra a continuación:

Archivo EQN:

```
LUGAR:          Universidad de Guadalajara.
CAMPUS:        Centro Universitario de Ciencias Exactas e Ingenierías.
CARRERA:       Ingeniería en Comunicaciones y Electrónica.
AUTOR:         José Miguel Morán Loza.

APLICACION #1  Decodificador de Binario a Hexadecimal a 7 Segmentos
                Display Sencillo de Cátodo Común.

CHIP BINHEX71 GAL16V8

P D1 C1 B1 A1 6 7 8 9 GND
11 12 a b f g d c e VCC

EQUATIONS

/a=/P*/D1*/C1*/B1*A1 + /P*/D1*C1*/B1*/A1 + /P*D1*/C1*B1*A1 +
/P*D1*C1*/B1*A1

/b=/P*/D1*C1*/B1*A1 + /P*D1*C1*/A1 + /P*C1*B1*/A1 + /P*D1*B1*A1

/c=/P*/D1*/C1*B1*/A1 + /P*D1*C1*/A1 + /P*D1*C1*B1

/d=/P*/D1*/C1*/B1*A1 + /P*C1*B1*A1 + /P*D1*/C1*B1*/A1 + /P*/D1*C1*/B1*/A1

/e=/P*/C1*/B1*A1 + /P*/D1*C1*/B1 + /P*/D1*A1

/f=/P*/D1*/C1*A1 + /P*/D1*/C1*B1 + /P*/D1*B1*A1 + /P*D1*C1*/B1*A1

/g=/P*/D1*/C1*/B1 + /P*/D1*C1*B1*A1 + /P*D1*C1*/B1*/A1
```

Archivo JEDEC :

GAL16V8

EQN2JED - Boolean Equations to JEDEC file assembler (Version V024)

Copyright (c) National Semiconductor Corporation 1990,1991

Assembled from "C:\OPAL\TESIS\BINHEX71.EQN". Date: 9-19-97

LUGAR: Universidad de Guadalajara.

CAMPUS: Centro Universitario de Ciencias Exactas e Ingenierías.

CARRERA: Ingeniería en Comunicaciones y Electrónica.

AUTOR: José Miguel Morán Loza.

APLICACION #1 Decodificador de Binario a Hexadecimal a 7 Segmentos
Display Sencillo de Cátodo Común.

*

NOTE PINS P:1 D1:2 C1:3 B1:4 A1:5 6:6 7:7 8:8 9:9 GND:10 11:11*

NOTE PINS 12:12 a:13 b:14 f:15 g:16 d:17 c:18 e:19 VCC:20*

QF2194*QP20*F0*

L0000	L1216
11101011101101111111111111111111	00000000000000000000000000000000
10100111101111111111111111111111	00000000000000000000000000000000*
10101111111011111111111111111111	L1280
00000000000000000000000000000000	10100111101101111111111111111111
00000000000000000000000000000000	01100111111110111111111111111111
00000000000000000000000000000000	11100111011110111111111111111111
00000000000000000000000000000000	01101111011101111111111111111111
00000000000000000000000000000000*	00000000000000000000000000000000
L0256	00000000000000000000000000000000
10101011011110111111111111111111	00000000000000000000000000000000
01100111111110111111111111111111	00000000000000000000000000000000*
01100111011111111111111111111111	L1536
00000000000000000000000000000000	10101011101101111111111111111111
00000000000000000000000000000000	10100111101110111111111111111111
00000000000000000000000000000000	01101011011101111111111111111111
00000000000000000000000000000000	01100111101101111111111111111111
00000000000000000000000000000000*	00000000000000000000000000000000
L0512	00000000000000000000000000000000
10101011101101111111111111111111	00000000000000000000000000000000
11100111011101111111111111111111	00000000000000000000000000000000*
01101011011110111111111111111111	L1792
10100111101110111111111111111111	00000000000000000000000000000000
00000000000000000000000000000000	00000000000000000000000000000000
00000000000000000000000000000000	00000000000000000000000000000000
00000000000000000000000000000000	00000000000000000000000000000000
00000000000000000000000000000000*	00000000000000000000000000000000
L0768	00000000000000000000000000000000
10101011101111111111111111111111	00000000000000000000000000000000
10100111011101111111111111111111	00000000000000000000000000000000*
01100111101110111111111111111111	L2048
00000000000000000000000000000000	00000000*
00000000000000000000000000000000	L2056
00000000000000000000000000000000	00000000000000000000000000000000
00000000000000000000000000000000	00000000000000000000000000000000*
00000000000000000000000000000000*	L2120
L1024	00000000*
10101011111101111111111111111111	L2128
10101011011111111111111111111111	11100000111000001111000011100000
10101111011101111111111111111111	11110000111100001111000000000000*
01100111110111011111111111111111	L2192
00000000000000000000000000000000	10*
00000000000000000000000000000000*	C5F23*

Archivo LOG :

EQN2JED - Boolean Equations to JEDEC file assembler (Version V024)
 Copyright (c) National Semiconductor Corporation 1990,1991

Document file for C:\OPAL\TESIS\BINHEX71.EQN
 Device: 16V8

\$LABELS 20 P D1 C1 B1 A1 6 7 8 9 GND 11 12 a b f g d c e VCC

Pin	Label	Type
1	P	pos,com input
2	D1	pos,com input
3	C1	pos,com input
4	B1	pos,com input
5	A1	pos,com input
6	6	unused
7	7	unused
8	8	unused
9	9	unused
10	GND	ground pin
11	11	unused
12	12	unused
13	a	neg,com output
14	b	neg,com output
15	f	neg,com output
16	g	neg,com output
17	d	neg,com output
18	c	neg,com output
19	e	neg,com output

EQN2JED - Boolean Equations to JEDEC file assembler (Version V024)
 Copyright (c) National Semiconductor Corporation 1990,1991

Device Utilization:

No of dedicated inputs used : 5/10 (50.0%)
 No of dedicated outputs used : 2/2 (100.0%)
 No of feedbacks used as dedicated outputs : 5/6 (83.3%)

Pin	Label	Terms	Usage
19	e	3/8	(37.5%)
18	c	3/8	(37.5%)
17	d	4/8	(50.0%)
16	g	3/8	(37.5%)
15	f	4/8	(50.0%)
14	b	4/8	(50.0%)
13	a	4/8	(50.0%)
Total		25/64	(39.1%)

EQN2JED - Boolean Equations to JEDEC file assembler (Version V024)
 Copyright (c) National Semiconductor Corporation 1990,1991

Chip diagram (DIP)

P	1	20	VCC
D1	2	19	e
C1	3	18	c
B1	4	17	d
A1	5	16	g
6	6	15	f
7	7	14	b
8	8	13	a
9	9	12	12
GND	10	11	11

Archivo CKT :

```

$ BINHEX71
$ JED2CKT -- JEDEC File to OPALSIM Circuit/Macro Translator (Version V022)
$ Copyright (c) National Semiconductor Corporation 1990,1991
$ Translated from BINHEX71.jed. Date: 9-19-97
$ DEVICE GAL16V8

.LIB BINHEX71.mac

* U1 BINHEX71 2 P D1 C1 B1 A1 6 7 8 9 GND 11 12 a b f g d c e
+ VCC

P INPUT =00000000000000001111111111111111
D1 INPUT =00000000111111110000000011111111
C1 INPUT =00001111000011110000111100001111
B1 INPUT =00110011001100110011001100110011
A1 INPUT =01010101010101010101010101010101

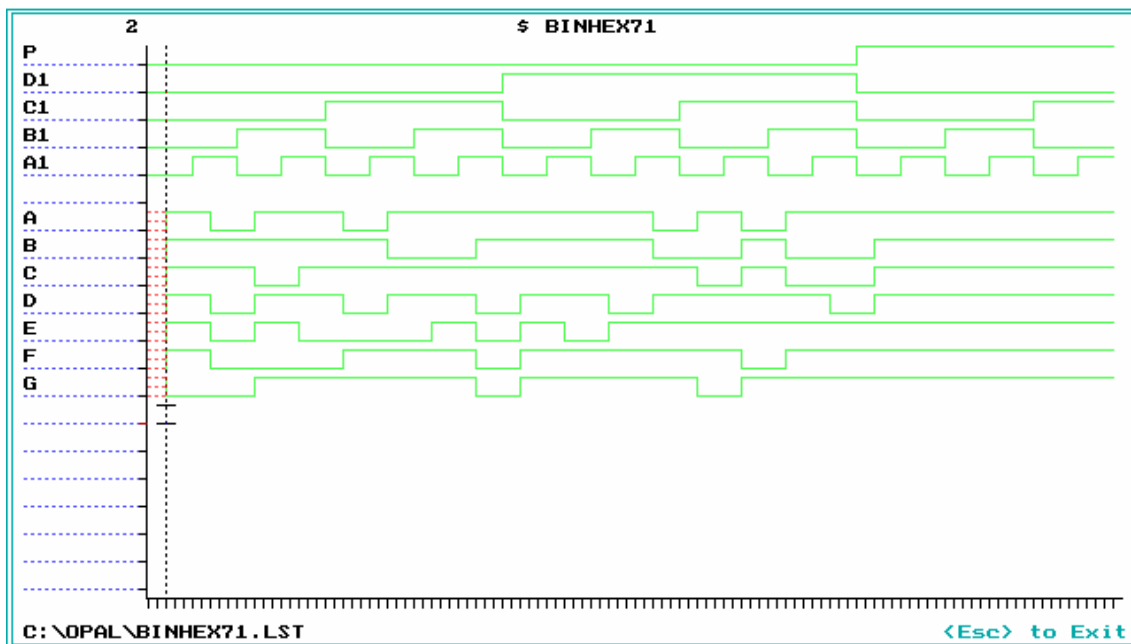
.PROBE P D1 C1 B1 A1 . a b f g d c e

.TIME 1000

.END

```

Del archivo anterior mediante su simulación obtendremos el siguiente

Archivo LST :

En la aplicación anterior se observa como el dispositivo cumple una función especial; sin embargo, también nos damos cuenta que el dispositivo es subutilizado. ¿Por qué? La razón es muy palpable; el dispositivo solo utiliza un 39.1 % de sus terminos y un 68.1 % del total de su arquitectura y con esta aplicación el dispositivo podría aprovecharse doblemente.