UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POST-GRADO

APLICACIONES DEL DISEÑO LOGICO PROGRAMABLE

POR

ING. MIGUEL ANGEL GUTIERREZ ZAMARRIPA

TESIS

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS
DE LA INGENIERIA ELECTRICA CON
ESPECIALIDAD EN ELECTRONICA

CD. UNIVERSITARIA, JUNIO DE 1999

APLICACIONES DEL DISEÑO LOGICO PROGRAMABLE

M.A.G.Z.

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POST-GRADO

APLICACIONES DEL DISEÑO LOGICO PROGRAMABLE

POR

ING. MIGUEL ANGEL GUTIERREZ ZAMARRIPA

TESIS

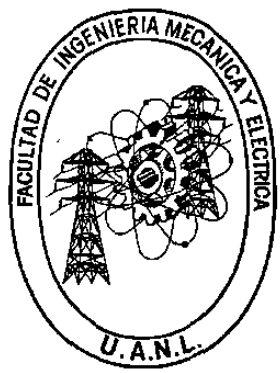
EN OPCION AL GRADO DE MAESTRO EN CIENCIAS
DE LA INGENIERIA ELECTRICA CON
ESPECIALIDAD EN ELECTRONICA

CD. UNIVERSITARIA, JUNIO DE 1999

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POST-GRADO



APLICACIONES DEL DISEÑO LOGICO PROGRAMABLE

POR

ING. MIGUEL ANGEL GUTIERREZ ZAMARRIPA

TESIS

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS DE LA

INGENIERIA ELECTRICA CON ESPECIALIDAD EN ELECTRONICA

CD.UNIVERSITARIA , JUNIO DE 1999

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POST-GRADO

Los miembros del comité de tesis recomendamos que la tesis APLICACIONES DEL DISEÑO LOGICO PROGRAMABLE realizada por el ING. MIGUEL ANGEL GUTIERREZ ZAMARRIPA sea aceptada para su defensa como opción al grado de MAESTRO EN CIENCIAS DE LA INGENIERIA ELECTRICA CON ESPECIALIDAD EN ELECTRONICA .

El Comité de Tesis

M.C. Juan Angel Garza Garza
Asesor

M.C. Sergio Martínez Luna
Coasesor

M.C. Luis Manuel Martínez Villarreal
Coasesor

Vo.Bo.
M.C. Roberto Villarreal Garza
División de Estudios de Post-grado

San Nicolás de los Garza, N.L. a 3 de Junio de 1999

AGRADECIMIENTO:

A mi esposa Glafira quien colaboró literalmente al escribir las primeras páginas de esta tesis, pero mi mayor agradecimiento es por el apoyo moral que recibí de su parte.

A mi asesor M.C. Juan Angel Garza Garza quien creyó en mí y me motivó a comenzar y a terminar este trabajo y que siempre invirtió parte de su valioso tiempo para atender todas las consultas así como todas las solicitudes de préstamo de material didáctico (libros, artículos de revistas, hojas de datos, etc.) , de sofware y hardware que le solicité.

A mis coasesores M.C. Luis Manuel Martínez Villarreal y M.C. Sergio Martínez Luna, quienes estuvieron siempre apoyándome y dispuestos a colaborar cuando fuese necesario.

A mis compañeros maestros M.C. Santiago Neira Rosales y M.C. Cesar Sordia Salinas quienes me apoyaron moral y tecnicamente al facilitarme el uso de su respectiva oficina incluyendo el equipo de la misma.

Al joven Roberto Augusto Flores Martínez , Becario de la coordinación de Ciencias Basicas por el apoyo técnico computacional que me brindó.

A todos ellos, ¡ muchas gracias!

DEDICATORIA

Con todo mi amor a mi esposa Glafira, a mis hijos Alejandra y Miguel Angel, También dedico esta obra a mi madre , Sra. Antonia Zamarripa Paz quien se esforzó para que yo realizara mis estudios, a mi padre , Juan Gutiérrez Colunga(✞) quien no pudo ver este trabajo, a mis hermanos Juan Antonio, Rodolfo y Rocío de la Luz, mis padres políticos , Lic Arturo Pàez P. y Sra. Glafira Garcìa de Pàez y hermanos políticos Jesús Arturo , Ana Laura y Clara Isabel.

Prólogo

Esta tesis es una aplicación directa del Diseño Lógico Programable, este concepto que consiste en empaquetar una gran cantidad de lógica dentro de un solo circuito integrado mediante la programación del mismo, fue tratado en la tesis "Diseño Lógico Programable" del M.C. Juan Angel Garza Garza catedrático de la F.I.M.E., quien ahora funge como Asesor en esta tesis elaborada por el Ing. Miguel Angel Gutiérrez Zamarripa.

Se aplica el concepto de Diseño Lógico Programable a dos problemas sugeridos, por otro lado se hace ver al alumno de Licenciatura que esté tomando un curso de Electrónica Digital (Electrónica Lógica) que hay otra alternativa para resolver los dos problemas arriba mencionados ya que en la materia "Eléctronica Lògica" que se imparte en las carreras de "Ingeniero en Electrónica y Comunicaciones" , "Ingeniero en Control y Computaciòn" e "Ingeniero Administrador de Sistemas" de la FIME se diseñan los sistemas digitales utilizando circuitos integrados de función fija, la propuesta es que el alumno diseñe a traves de PLDs. (DISPOSITIVOS LOGICOS PROGRAMABLES), esto es importante debido a que los PLDs se utilizan en muchas aplicaciones para reemplazar los circuitos SSI y MSI, ya que ahorran espacio y reducen el costo de los dispositivos en un determinado diseño. Para implementar diseños lógicos con PLDs se utilizan aplicaciones de software de programacion tales como ABEL y CUPL. En esta tesis se utiliza solamente el lenguaje ABEL solo para ilustrar los principios de programación para PLDs, los otros lenguajes aunque no se mencionan son en realidad muy similares y muy fáciles de aprender.

Otra opción que se propone es el de partir de un diagrama esquemático desarrollado en una aplicación de la computadora personal.

INDICE

CAPITULO 5 Programaciòn de los Dispositivos Lògicos Programables

CAPITULO 6

APLICACION COMBINACIONAL CON PLDs.

CAPITULO 7

APLICACIÓN SECUENCIAL CON PLDs

CAPITULO 1 Síntesis

1.1 SINTESIS

Este trabajo se enfoca a la solución de dos problemas de diseño digital, uno combinacional y otro secuencial, los cuales primero serán analizados bajo el punto de vista tradicional, es decir utilizando circuitos integrados de función fija. Después se utilizará el concepto de DISEÑO LOGICO PROGRAMABLE para solucionar los dos problemas mencionados. Esto implica la utilización de la P.C. así como software y hardware adecuados.

Se incluye además información adicional para que el concepto de DISEÑO LOGICO PROGRAMABLE sea manejado por la mayor cantidad posible de personas que tengan al menos conceptos básicos de Electrónica Digital.

Se comienza con una breve introducción al concepto de la Lógica Programable donde se analiza la situación actual acerca de la utilización de dicho concepto y se hace una comparación con las opciones que se tenían anteriormente. Se incluyen dos capítulos (el 6 y el 7) en uno de los cuales se utilizan los Dispositivos Lógicos Programables (Programmable Logic Device PLDs), donde se hace especial énfasis en la utilización de los PLDs para implementar Lógica Combinacional, en el otro capítulo, es decir el 7, se tratan las aplicaciones lógicas secuenciales de los PLDs y, en concreto, las

Matrices Lógicas Genéricas (Generic Array Logic. GAL) las cuales pueden emplearse para reemplazar dispositivos lógicos SSI y MSI.

Se incluye también un apéndice con información importante de los fabricantes de PLDs, además de un glosario con algunos términos digitales.

CAPITULO 2 Introducción

2.1 Descripción del problema

Anteriormente cuando se pensaba en el diseño lógico de sistemas de mediana y alta complejidad se debía pensar en la cantidad de componentes discretos que se tenían que incluir. Los prototipos eran alambrados, las pruebas tomaban demasiado tiempo y por supuesto, la salida al mercado del producto también tomaba mucho tiempo. Por otra parte, los costos por incluir en un diseño varios circuitos discretos integrados LSI o MSI eran elevados. Si se tenía que hacer algún cambio en el diseño se debían fabricar nuevos circuitos impresos o en su defecto realambrar, lo que representaba un gasto adicional. El espacio ocupado por los sistemas construidos bajo esta filosofía era grande y ya no se diga la dificultad que representaba transportarlos; por ejemplo si se abre una microcomputadora construida en 1982 se pueden identificar fácilmente todos sus componentes como el microprocesador, los circuitos integrados (CIs) que corresponden a la memoria, colocados en conjuntos de ocho, un CI controlador de disco flexible, y bloques físicos de CIs más pequeñitos.

Para evitar muchos de los problemas anteriores se introdujo el concepto de la lógica programable. Dicho concepto radica en empaquetar una gran cantidad de lógica dentro de un solo CI mediante la programación del mismo. Esto superó las barreras que se habían venido formando alrededor de los diseños lógicos.

A medida que se fueron encontrando nuevas tecnologías para la fabricación de CIs, los problemas anteriores se fueron superando. Por ejemplo, la *empresa "Monolithic Memories"* fue la primera en introducir la tecnología de fusibles bipolares utilizados en las memorias PROM *"Programmable Read Only Memory"*; así se pudo contar con una buena cantidad de lógica (en un principio solamente combinacional) dentro de un sólo CI que además era muy rápido.

Posteriormente se fabricaron los dispositivos lógicos programables o PLDs *"Programmable Logic Devices"*, con los que se obtendría una eficiencia superior a la lograda con los dispositivos PROMs. Además de empaquetar pura lógica combinacional, también se hizo posible empaquetar registros para utilizarlos en el diseño con lógica secuencial.

La complejidad de los diseños ha ido creciendo, de aquí la necesidad de empaquetar no solamente compuertas y registros en un sólo CI, sino también CIs completos de dichas compuertas y registros dentro de un mismo circuito integrado con mayor densidad; esto ha dado origen a los *CPLDs "Complex PLDs"* (Dispositivos Lógicos Programables Complejos) y a los FPGAs *"Field Programmable Gate Arrays"* (Arreglos de Compuertas Programables en el Campo de Trabajo).

Hacia la primera mitad de los 90´s, la industria electrónica experimentó una explosión en la demanda de computadoras personales, teléfonos celulares, y dispositivo de comunicaciones de datos de alta velocidad. Esto se debió a que hubo una considerable reducción del tamaño de los circuitos que contenían los equipos, las velocidades de los mismos se incrementaron, los costos por la disminución en la utilización de

circuitos discretos de uso específico disminuyeron, el consumo de potencia de los CIs disminuyó, la confiabilidad aumentó, lo que hasta la fecha significa mayor tiempo de garantía y menores precios al público consumidor. Actualmente, los dispòsitivos de lógica programable además de ser programables son borrables, ya sea por luz ultravioleta o por medios elèctricos, proporcionándole al usuario la ventaja de poder reutilizar el dispositivo en casi cualquier tipo de aplicaciòn digital. También se han sustituido los CIs. VLSI actuales por sus similares en lógica programable, ya que mediante el diseño con FPGAs se han logrado empaquetar hasta 125,000 compuertas utilizables.

Conforme avanza la tecnologìa de fabricaciòn de dispositivos lògicos programables, a la par se desarrollan lenguajes orientados al diseño de hardware tales como: AHDL,ABEL,CUPL,VERILOG,VHDL etc.Los cuales son herramientas de diseño y, consecuentemente, auxiliares en la programaciòn de dispositivos lògicos; estos se utilizan cada vez más debido a que son eficientes y amigables al usuario.Existe una especie de cadena circular, ya que estos lenguajes ayudan a optimizar cada vez mas el diseño de CIs, que a su vez también son utilizados en la fabricaciòn de computadoras con mejores caracterítiscas, que a su vez producirán mejores programas para el diseño de Cis.

Existen compañías dedicadas a la fabricación de estos Cis de Lógica Programable, entre las que podemos mencionar: Actel, Altera, Advanced Micro Devices (AMD), Atmel, AT&T microelectronics, Catalyst Semiciconductors, Cypress Semiconductors, Data I/O corporatión, Fujitsu, Gould AMI Semiconductors, International CMOS Techonology inc.,Intel Interstil, Lattice semiconductors Corporation, Mitsubishi, Plus Logic, Toshiba

National Semiconductors, NEC Corporation, GEC Pessey semiconductors inc., Phillips semiconductors(PML), Quick Logic, Raytheon, SGS Thomson Microelectronics, Texas instruments, Wafer Scale Integration y Xilinix. Esto nos puede dar una idea de qué tan importante es la Lógica Programable actualmente.

En un futuro las empresas que se dedican al diseño reduciran enormemente sus inventarios de dispositivos VLSI. Teniendo en existencia unos cuantos CIs de Lógica Programable y archivos para configurarlos.

2.2 Objetivos de la tesis

El objetivo principal de esta tesis es promover la utilización del concepto de Diseño Lógico Programable a los estudiantes de licenciatura que estén tomando un curso de Electrónica Digital; ya que hablando en forma general de tecnología VLSI (es decir el circuito integrado de muy grande escala) se tiene la alternativa de la utilización de los microprocesadores que son en efecto circuitos lógicos de estructura muy general que pueden aplicarse a una variedad de aplicaciones a través de la programación.

Sin embargo, aunque los microprocesadores pueden reemplazar ciertos circuitos lógicos, este camino no es siempre el mejor. En general debido a que los microporcesadores tienden a ser lentos como reemplazos lógicos, en esta tesis se estudia otro camino relativo al que le llamamos "Diseño Lógico Programable".

2.3 Justificación de la tesis

La justificación de la elaboración de esta tesis es la no utilización (desaprovechamiento) de un CI VLSI (Circuito Integrado de muy grande escala de integración) que puede reemplazar en un sistema digital a varios circuitos LSI (gran escala de integracion) y MSI (mediana escala de integración). Para hacer una analogía podemos pensar en un arreglo serie o paralelo de resistencias en un circuito eléctrico para obtener un cierto valor de ohms, en la práctica, en vez de pensar en un arreglo serie o paralelo se puede utilizar una sola resistencia (la llamada "resistencia equivalente") que tenga dicho valor. Una variación al utilizar esta analogía es que el valor de la resistencia equivalente la decide el fabricante y no el usuario, lo que no sucede con algunos PLDs en donde es el usuario el que decide por medio de la programación del mismo la función que se desea.Esto implica disminución de espacios y costos y justifica la utilización de los PLDs y esto a su vez justifica la información y aplicación que se da en esta tesis.

2.4 Metodología

La metodología que se va a seguir es simple, se proponen dos problemas de diseño digital los cuales primeramente serán analizados con la lógica tradicional que es la utilización de circuitos integrados de función fija, después serán resueltos con la utilización de herramientas de software para implementar en un solo circuito integrado de los llamados GALs (Arreglos Lógicos Genéricos), específicamente el GAL16V8 y el GAL20V8 la lógica de cada uno de los problemas sugeridos, esto se hará utilizando un programador universal y un lenguaje de descripción de hardware.

2.5 Límites del estudio

La principal limitación de esta tesis se debe principalmente a que es un tema relativamente nuevo y se esta implementando en el plan de estudio de muchas instituciones de nivel superior donde existan carreras afines a la electrónica digital, aún así en el internet se encuentran direcciones con artículos, información de cursos de lógica programable , fabricantes de PLDs etc. También existen algunos libros que tratan este tema exclusivamente y otros libros tradicionales de sistemas digitales que en sus últimas ediciones han agregado capítulos nuevos sobre Diseño Lógico Programable. (Ver resumen bibliográfico).

2.6 Revisión Bibliográfica

Los conceptos básicos de Electrónica Digital los tratan muchos libros en forma muy completa, dichos conceptos se dan por conocidos por el lector de esta tesis, aún así, se incluye un glosario de términos digitales, sin embargo esta tesis se puede decir que es una continuación de la tesis del M.C. Juan Angel Garza Garza en donde se hace un estudio de los circuitos programables desde un punto de vista más profundo dándoles el nombre más general de ASICs (Circuitos Integrados de Aplicación Específica)

CAPITULO 3 Diseño Lógico Programable.

3.1 Antecedentes

El concepto de Diseño Lógico Programable es simple, consiste en empaquetar una gran cantidad de lógica dentro de un solo circuito integrado mediante la programación del mismo. Estos circuitos integrados en su forma más general se denominan ASICs. (Circuitos Integrados de Aplicación Específica). En el mercado existen diferentes ASICs para aplicaciones sencillas, tales como decodificadores, multiplexores, controladores de motores de pasos, etc., los cuales la mayoría de los vendedores denominan PLDs (Dispositivos Lógicos Programables) también existen los CPLDs (Dispositivos Lógicos Programables Complejos) y los FPGAs (Arreglos de Compuertas Programables en el Campo de Trabajo) los cuales se utilizan para sustituir circuitos con densidades MSI y VLSI.

Los circuitos integrados CPLDs, se fabrican con el propósito de no incrementar el número de pines de los PLDs, sino de incrementar la densidad de los mismos empaquetando varios PLDs en un mismo circuito integrado, interconectándolos por medio de vías programables: estos integrados tienen mayor número de términos producto que los FPGAs pero menor número de flip-flops.

Los FPGAs son arreglos de celdas lógicas de alta densidad que se comunican entre sí y con los pines de entrada/salida por medio de vías que se encuentran dentro de los canales de ruta. El nombre que reciben está mal aplicado ya que el término arreglo de compuertas se refiere a una serie de compuertas lógicas arregladas en filas y columnas, relacionadas a una variedad de celdas entrada/salida, y los FPGAs raramente lo son.

Las celdas Lógicas, también conocidas como celdas generadoras de funciones, son bloques lógicos configurados para procesar cualquier función lógica. Dichas celdas tienen características de ser funcionalmente completas. Se dice que una celda es funcionalmente completa cuando se puede implementar cualquier función Booleana por medio de sumas de productos con el mismo tipo de celda. Como ejemplo de celdas lógicas de funcionalidad completa tenemos los multiplexores y las compuertas NAND.

En algunas ocasiones se puede llegar a confundir el concepto de FPGA con el CPLD ya que también se utilizan celdas lógicas en la fabricación de CPLDs. La diferencia, además de la cantidad de flip-flops, es que los arreglos de celdas lógicas en los FPGAs se encuentran interconectados por medio de vías horizontales y verticales, y en los CPLDs las salidas de las celdas se encuentran retroalimentadas con las entradas.

3.2 Clasificación de los Dispositivos Lógicos Programables.

En este punto se introducen los Dispositivos Lógicos Programables (Programmable Logic Device, PLD) y su clasificación. Para implementar diseños lógicos con PLDs. Se utilizan paquetes software de programación tales como ABEL y CUPL.

3.3 Matrices Programables.

Todos los PLDs están formados por matrices programables. Esencialmente, una matriz programable es una red de conductores distribuidos en filas y columnas con un fusible en cada punto de intersección. Las matrices pueden ser fijas o programables. El tipo más sencillo de matriz programable, que data de los años sesenta, era una matriz de diodos con un fusible en cada punto de intersección de la misma.

3.3.1 La Matriz OR.

La matriz de diodos primera evolucionó a la matriz integrada OR, que está formada por una serie de puertas OR conectadas a una matriz programable con fusibles en cada punto de intersección de una columna y una fila, como muestra la figura 3-1 (a). La matriz se programa fundiendo los fusibles para eliminar las variables seleccionadas de las funciones de salida, como ilustra la parte (b) de la misma figura, para un caso específico. Para cada una de las entradas de una puerta OR, solo queda intacto un fusible que conecta la variable deseada a la entrada de la puerta. Una vez que el fusible esta fundido, no se puede volver a conectar.

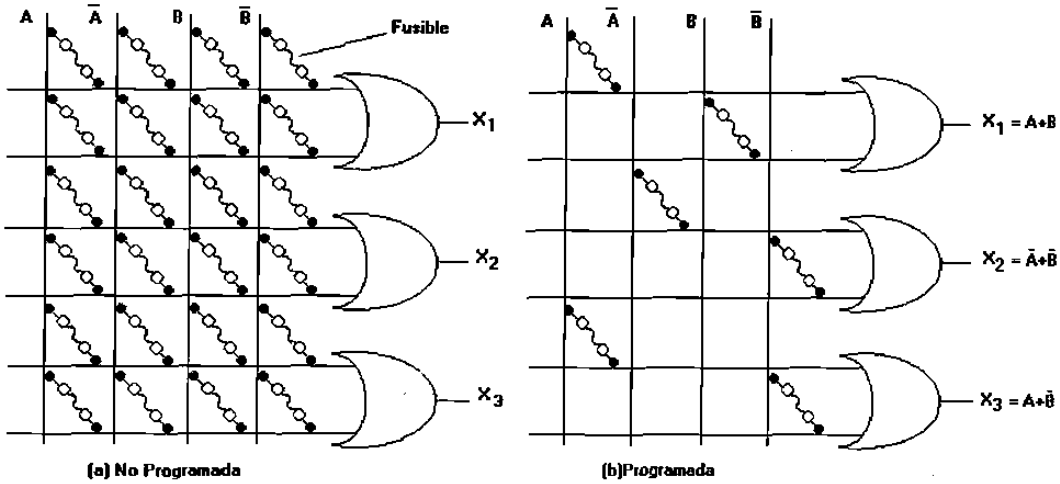
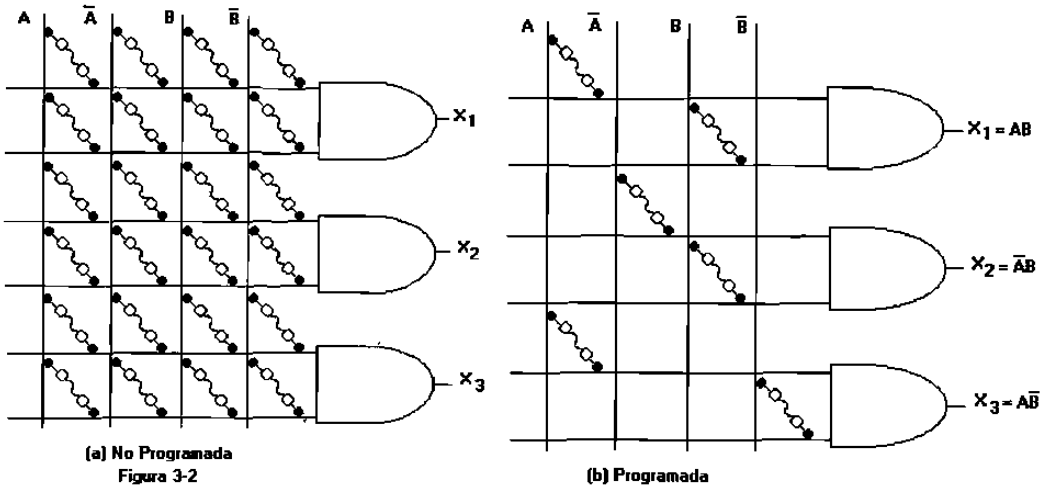


Figura 3-1

3.3.2 La Matriz AND.

Este tipo de matriz esta formado por puertas AND conectadas a una matriz programable con fusibles en cada punto de intersección, como muestra la figura 3-2(a). Al igual que la matriz OR, esta se programa fundiendo los fusibles para eliminar las variables de la función de salida, como ilustra la parte (b) de la figura. Para cada entrada de una puerta AND, solo queda intacto un fusible que conecta la variable deseada a la entrada de la puerta. Como para la matriz OR, la matriz AND con fusibles se puede programar una única vez.



(a) No Programada (b) Programada

Figura 3-2

3.3.3 Clasificación De Los PLDs.

Los PLDs se clasifican de acuerdo con su arquitectura, la cual es básicamente la ordenación funcional de los elementos internos que proporciona al dispositivo sus características específicas. Memoria Programable de Solo Lectura (Programable Read-Only Memory, PROM). La PROM está formada por un conjunto fijo (no programable) de puertas AND conectadas como decodificador y una matriz programable OR, como muestra el diagrama de bloques general de la figura 3-3. Fundamentalmente, la PROM se utiliza como una memoria direccionable y no como un dispositivo lógico, debido a las limitaciones que imponen las puertas AND fijas.

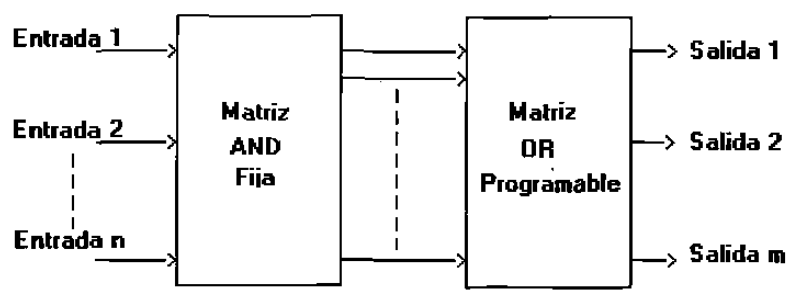


Figura 3.3

3.3.4 Matriz Lógica Programable PLA (Programmable Logic Array).

La PLA es un PLD formado por una matriz AND programable y una matriz OR programable, como muestra la figura 3-4. La PLA ha sido desarrollada para superar algunas de las limitaciones de las memorias PROM. La PLA también se denomina FPLA (Field-Programmable Logic Array, Matriz Lógica Programable en Campo) debido a que es el usuario y no el fabricante el que la programa.

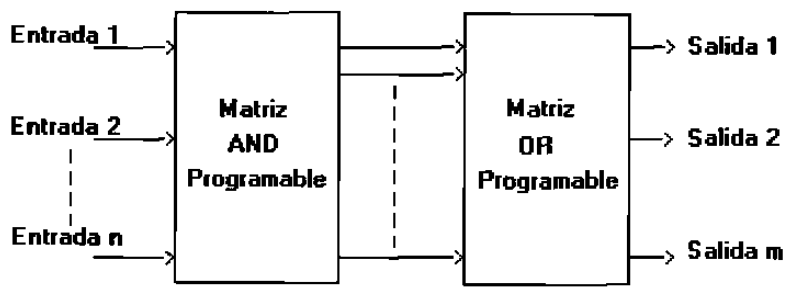


Figura 3.4

3.3.5 Matriz Lógica Programable PAL (Programmable Array Logic).

La PAL es un PLD que se ha desarrollado para superar ciertas desventajas de la PLA, tales como los largos retardos debidos a los fusibles adicionales que resultan de la utilización de dos matrices programables y la mayor complejidad del circuito. La PAL básica está formada por una matriz AND programable y una matriz OR fija con la lógica de salida, como se muestra en la figura 3-5. La PAL es el Dispositivo Lógico Programable una única vez más común, y se implementa con tecnología bipolar (TTL o ECL).

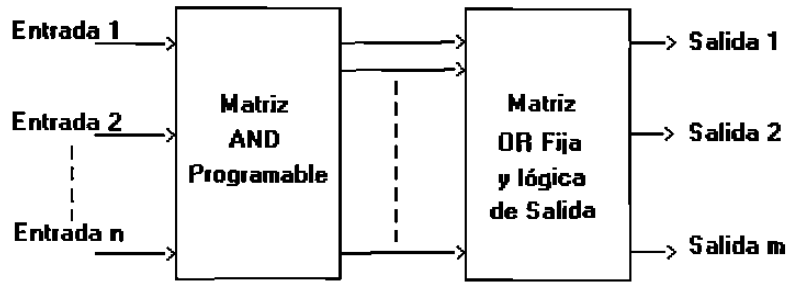


Figura 3.5

3.3.6 Matriz Lógica Genérica (Generic Logic Array, GAL).

El desarrollo más reciente de los PLDs es la GAL, al igual que la PAL, se forma con una matriz AND programable y una matriz OR fija, con una salida lógica programable. Las dos principales diferencias entre los dispositivos GAL y PAL son: (a) la GAL es programable y (b) la GAL tiene configuraciones de salida programables.

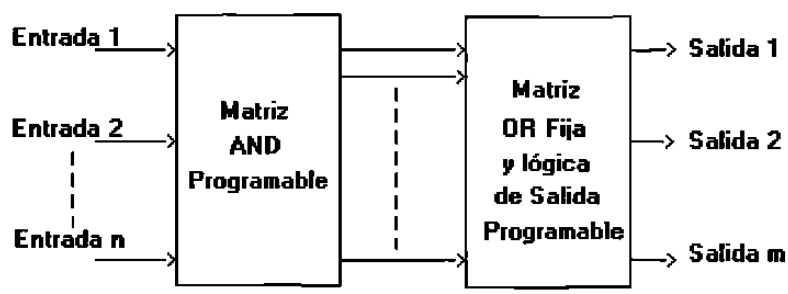


Figura 3.6

La GAL se puede reprogramar una y otra vez, porque se usa la tecnología EECMOS (Electrically Erasable Cmos, Cmos Borrable Eléctricamente), en lugar de tecnología bipolar y fusibles. En la figura 3-6 se muestra el diagrama de bloques de una GAL.

Capítulo 4 Matriz Genérica Programable (GAL)

4.1 Funcionamiento de una GAL

Básicamente, una GAL está formada por una matriz de puertas AND reprogramables conectadas a una matriz OR fija. Al igual que la PAL, esta estructura permite implementar cualquier función lógica como suma de productos con un número de variables definido.

En la figura 4-1 se ilustra la estructura Básica de una GAL con dos variables de entrada y una de salida, aunque la mayoría de los GALs pueden tener muchas entradas y muchas salidas. La matriz reprogramable es esencialmente una red de conductores ordenados en filas y columnas, con una celda CMOS eléctricamente borrable (EECMOS) en cada punto de intersección, en lugar de un fusible como en el caso de las PALs. En la figura 4-1 estas celdas se indican como bloques.

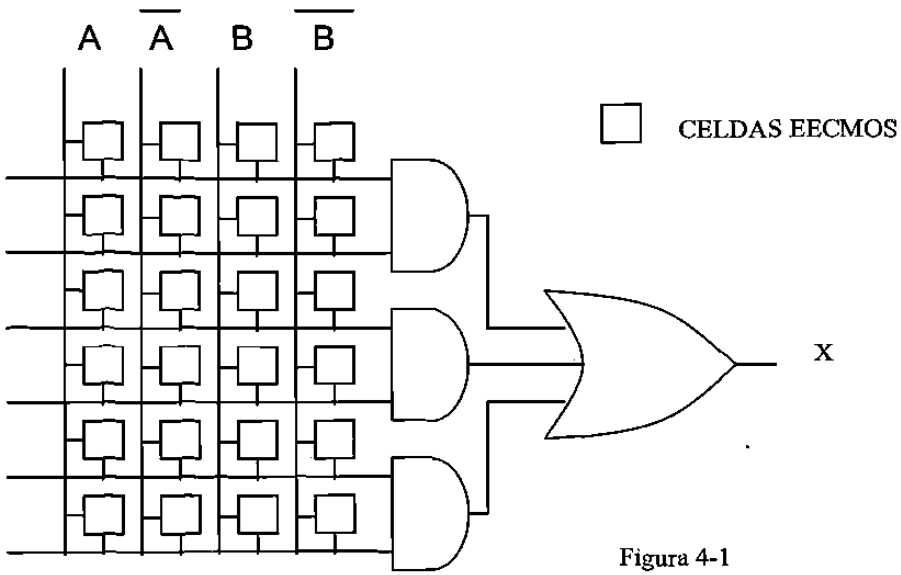


Figura 4-1

Cada fila esta conectada a la entrada de la puerta AND, y cada columna a una variable de entrada o a su complemento. Mediante la programación se activa o desactiva cada celda EECMOS, y se puede aplicar cualquier combinación de variables de entrada, o sus complementos, a una puerta AND para generar cualquier operación producto que se desee. Una celda activada conecta de forma efectiva su correspondiente fila y columna, y una celda desactivada desconecta la fila y la columna. Las celdas se pueden borrar y reprogramar eléctricamente. Una celda EECMOS típica puede mantener el estado en que se ha programado durante 20 años o más.

4.2 Implementación De Una Suma De Productos.

En la figura 4-2 se muestra un ejemplo de una sencilla matriz GAL, programada para obtener el producto $A\overline{B}$ en la puerta AND superior, AB en la puerta del centro y $\overline{A}\,\overline{B}$ en la puerta AND inferior. Como se indica, las celdas EECMOS activadas conectan las variables deseadas o sus complementos con las apropiadas entradas de las puertas AND. Las celdas EECMOS están desactivadas cuando una variable o su complemento no se utiliza en un determinado producto. La salida final de la puerta OR es una suma de productos.

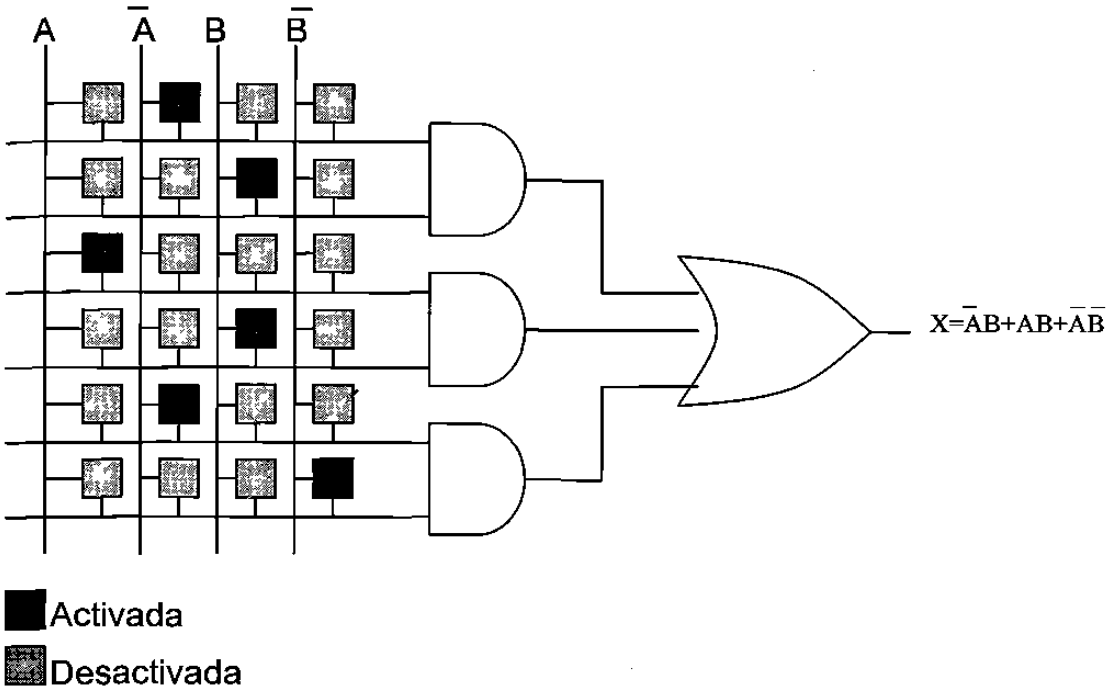


figura 4-2

4.3 Diagrama De Bloques De Una GAL.

En la figura 4-3 se presenta el diagrama de bloques de una GAL. Las salidas de la matriz AND se introducen en la macroceldas lógicas de salida (Output Logic Macrocells, OLMC) que contienen puertas OR y lógica programable. Una GAL típica puede tener ocho o más entradas, y ocho o más entradas/salidas de las OLMCs como se indica en la figura, siendo n>8 y m>8.

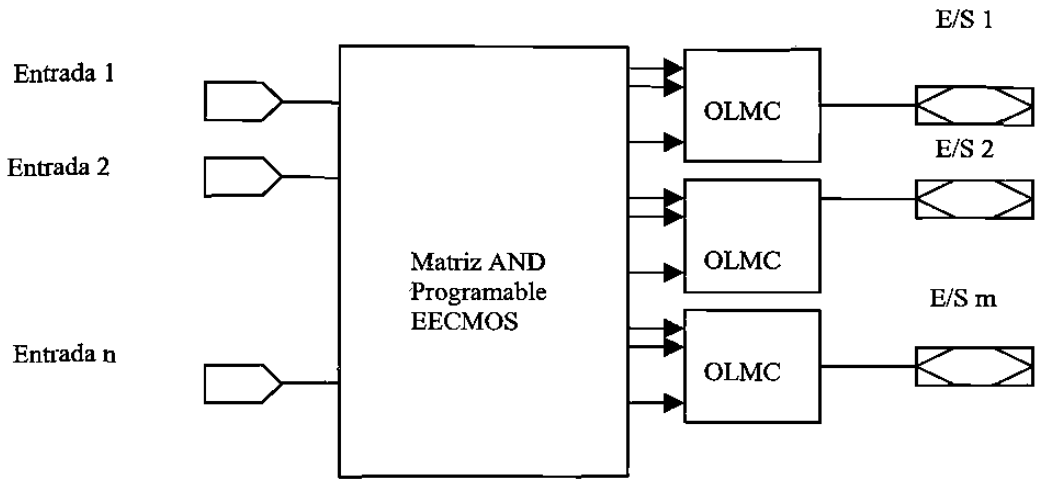


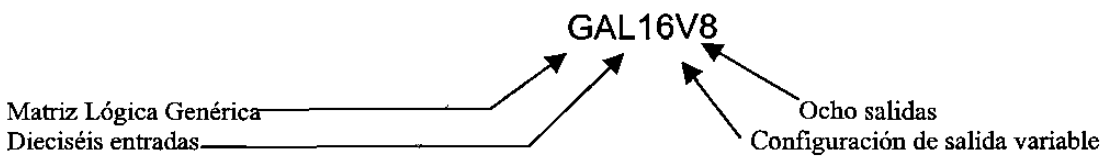
Figura 4-3

Las macroceldas lógicas de salida (OLMCs) están formadas por circuitos lógicos que se puede programar como lógica combinacional o como lógica secuencial. Las OLMCs proporcionan mucha más flexibilidad que la lógica de salida fija de una PAL.

4.4 Referencia Estándar De Una GAL.

Las GALs existen en una gran variedad de configuraciones, cada una de las cuales se identifica por una única referencia. Esta referencia siempre comienza por el prefijo GAL. Los dos primeros dígitos que siguen al prefijo indican el número de entradas, incluyendo las salidas que se pueden configurar como entradas. La letra V que sigue al número de entradas designa una configuración de salida variable. El, o los dos números

siguientes al tipo de salida son el numero de salidas. La siguiente referencia es un ejemplo:



4.5 Diagrama De Bloques De LA GAL16V8.

La GAL16V8 tiene ocho entradas dedicadas y ocho entradas/salidas, como muestra el diagrama de bloques de la figura 4-4 (a) en el que se indica la numeración de los pines del encapsulado. Este dispositivo esta disponible en encapsulador dip de 20 pines o en plcc también de 20 pines, como muestra la parte (b) de la misma figura.

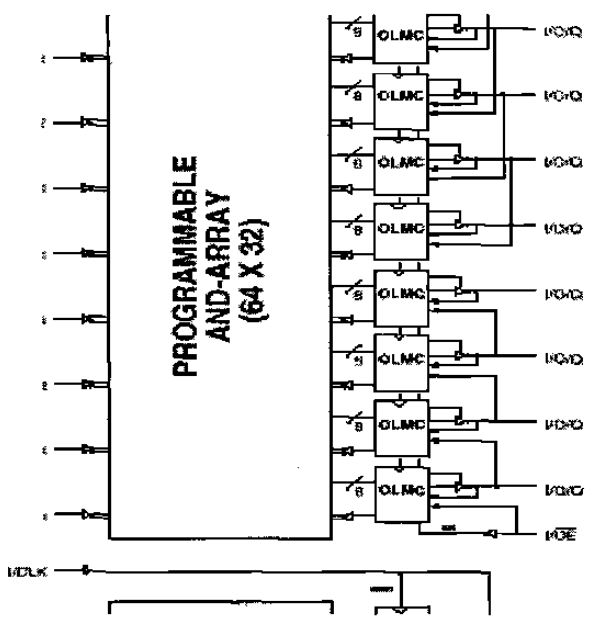


Figura 4-4(a)

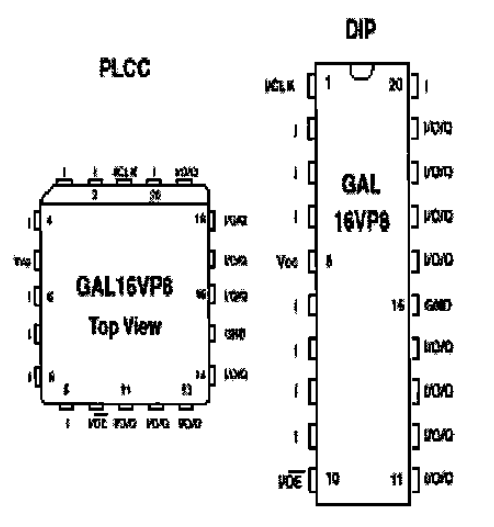


Figura 4-4(b)

4.6 Emulación De Una PAL.

Todas las posibles configuraciones de la OLMC de la GAL16V8 se clasifican en tres modos básicos: simple, complejo y secuencial. Cada uno de estos modos permite programar a la GAL16V8 para emular a un conjunto de PALs como las especificadas en la tabla 4-1 Los modos simples y complejos se asocian únicamente con salidas combinacionales.

GAL16V8 Modo simple	GAL16V8 Modo complejo	GAL16V8 Modo registro
PAL10L8	PAL16L8	PAL16R8
PAL12L6	PAL16H6	PAL16R6
PAL14L4	PAL16P8	PAL16R4
PAL16L2		PAL16RP8
PAL10H8		PAL16RP6
PAL12H6		PAL16RP4
PAL14H4		
PAL16H2		
PAL10P8		
PAL12P6		
PAL14P4		
PAL16P2		

Tabla 4-1 PALs que se pueden emular para cada modo de las OLMCs de la GAL16V8

4.7 Modo Simple.

En este modo, las OLMCs se configuran como salidas combinacionales dedicadas o como entradas dedicadas (como mucho seis). Las tres posibles configuraciones en este modo son:

*Salida combinacional.
*Salida combinacional con realimentación a La Matriz AND.
*Entrada dedicada.

Cada una de estas configuraciones se ilustran en la figura. 4-5 Observe que la línea de control de tres estados esta a nivel alto (vcc) activando el buffer para las configuraciones de salida combinacional, y a nivel bajo (a masa) dejando en abierto (alta impedancia) al buffer para la configuración de entrada dedicada.

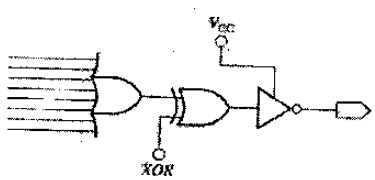


Figura 4.5(a) Salida combinacional

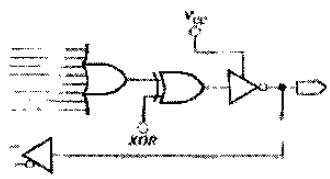


Fig. 4.5(b) Salida combinacional con realimentación a la matriz AND

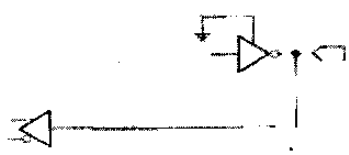


Fig. 4.5 (c) Entrada dedicada

La realimentación permite conectar la salida combinacional a la matriz AND. La entrada XOR de la puerta OR-EXCLUSIVA determina el estado activo de salida. Cuando XOR=0, la salida es activa a nivel bajo ya que la entrada procedente de la puerta OR no resulta invertida. Cuando XOR=1, la salida es activa a nivel alto, ya que la entrada procedente de la puerta OR resulta invertida.

Las ocho entradas de la puerta OR proceden de la matriz AND, e indican que la OLMC puede generar hasta ocho operaciones producto de una suma de productos.

4.8 Modo Complejo.

En este modo, la OLMC se puede configurar de dos formas:

*Salida Combinacional.
*Entrada/Salida Combinacional.

Estas dos configuraciones se ilustran en la figura. En este modo, las entradas/salidas (e/s) están limitadas a seis. Observe que solo existen siete entradas a la puerta OR procedentes de la matriz AND, ya que la octava entrada se utiliza para el control del buffer de tres estados. Esto significa que una OLMC puede producir hasta siete operaciones producto de una suma de productos.

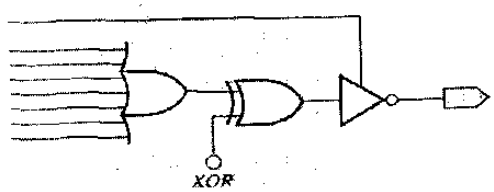


Fig. 4.6 (a) Salida combinacional

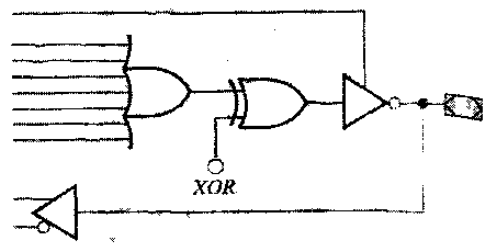


Fig. 4.5 (b) Entrada/Salida combinacional

Capitulo 5 Programación de PLDs.

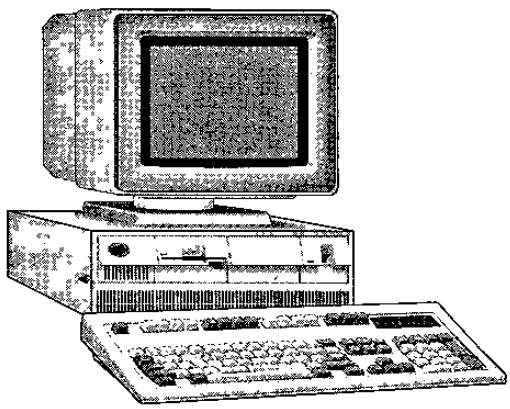
5.1 Elementos necesarios para programar un PLD.

Como ya hemos visto, las PALs se programan dejando intactos los fusibles especificados y fundiendo los restantes. Las GALs. Se programan de forma similar, pero son las celdas EECMOS las que se activan o desactivan. Las funciones lógicas que se implementan determinan las celdas que hay que activar. Para programar una PAL o una GAL se requieren los siguientes elementos: un computador, software de programación y un programador de PLDs.

Los tres componentes requeridos para programar los PLDs son:

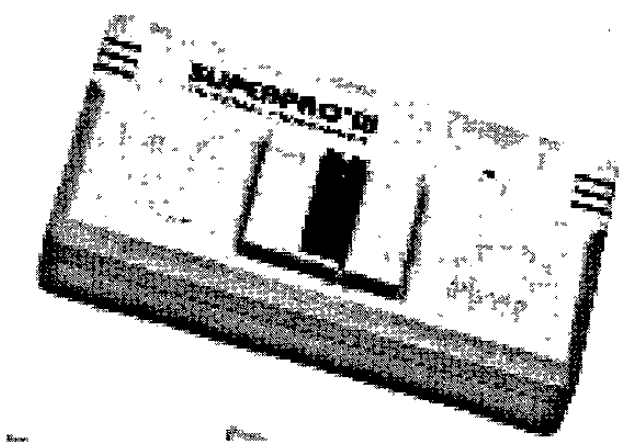
1.Software De Programación (Compilador Lógico).
2.Computador Personal Que Cumpla Los Requerimientos Software.
3. Programador Controlable Por Software.

El computador debe cumplir los requerimientos software. El programador es un dispositivo hardware que acepta datos de programación procedentes del computador e implementa un diseño lógico específico en el PLD, el cual se inserta en el zócalo en el programador. En la figura 5-1 se ilustran estos elementos.



La computadora

Software Requerido



Programador Universal

FIGURA 5-1

5.2 Computador.

Se puede emplear cualquier computador personal que cumpla las especificaciones software y del programador. Normalmente, estas especificaciones incluyen el tipo de microprocesador del que debe disponer el computador, la cantidad de memoria y el sistema operativo (dos, windows y Macintosh son algunos ejemplos).

5.3 Software.

Los paquetes software para la programación de PLDs se denominan compiladores lógicos. Existen diversos paquetes software disponibles, entre los que se incluyen los siguientes (existen otros además de estos): ABEL, CUPL, Orcad-Pld y Logic. Todos estos software realizan funciones similares: procesan y sintetizan el diseño lógico introducido mediante un método específico, convierten los datos introducidos en un archivo intermedio y luego generan un archivo de salida que se denomina archivo JEDEC (conocido también como mapa de celdas o mapa de fusibles) para el programador del dispositivo. También, con este software, se puede simular y depurar de forma completa un diseño lógico antes de proceder a fabricar el hardware.

Una característica fundamental del software para PLDs es el método para introducir el diseño lógico. Existen tres métodos básicos de introducir el diseño que son: ecuación booleana, tabla de verdad y máquina de estados. Existen otros métodos de tipo esquemático que son los diagramas de tiempo y la descripción hardware. Todos los paquetes software disponibles incluyen varios o todos los posibles métodos de introducción del diseño.

5.4 El Programador.

El dispositivo PLD se inserta en el zócalo del programador, que usualmente es un zócalo zif (zero insertion force, fuerza de inserción nula). Utilizando el archivo JEDEC generado por el compilador lógico, el programador programa el PLD por medio de la aplicación de las tensiones requeridas en los pines especificados, para alterar así las celdas específicas de la matriz de la forma que indique el mapa de fusibles. Todos los paquetes software y programadores de PLDs, independientemente de los fabricantes, son conformes a un estándar para la generación del archivo jedec establecido por el Joint Electronic Device Engineering Council (JEDEC).

5.5 El Proceso De Programación.

El organigrama general de la figura 5-2 muestra la secuencia para programar un PLD. En primer lugar, se diseña el circuito lógico y se expresa mediante ecuaciones booleanas, tablas de verdad, esquemas lógicos o cualquier otro formato aceptado, y se carga el software en el computador .

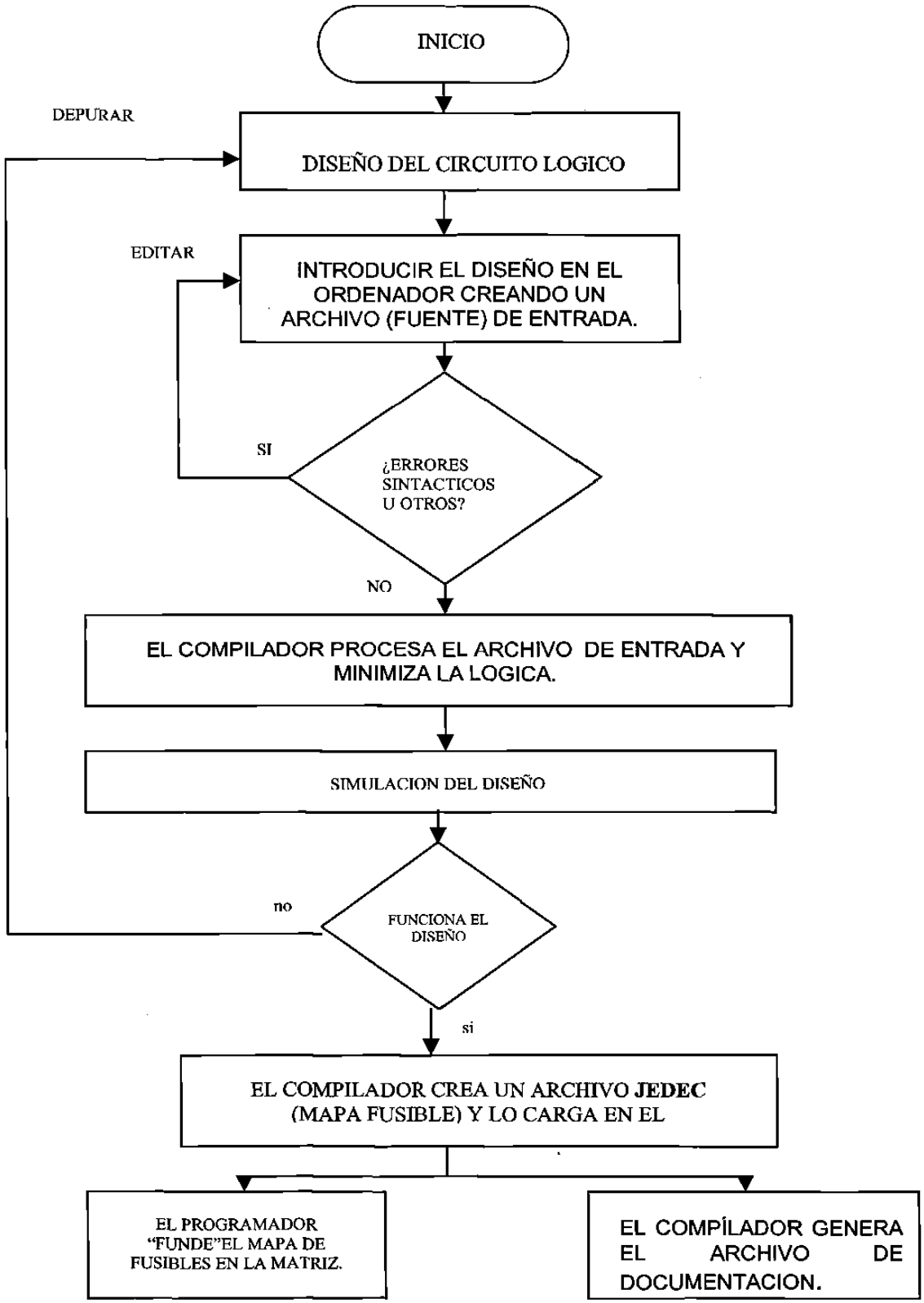

INICIO
DEPURAR
DISEÑO DEL CIRCUITO LOGICO
EDITAR
INTRODUCIR EL DISEÑO EN EL ORDENADOR CREANDO UN ARCHIVO (FUENTE) DE ENTRADA.
SI
¿ERRORES SINTACTICOS U OTROS?
NO
EL COMPILADOR PROCESA EL ARCHIVO DE ENTRADA Y MINIMIZA LA LOGICA.
SIMULACION DEL DISEÑO
no
FUNCIONA EL DISEÑO
si
EL COMPILADOR CREA UN ARCHIVO JEDEC (MAPA FUSIBLE) Y LO CARGA EN EL
EL PROGRAMADOR "FUNDE"EL MAPA DE FUSIBLES EN LA MATRIZ.
EL COMPÍLADOR GENERA EL ARCHIVO DE DOCUMENTACION.

FIGURA 5.2

5.6 Introducción Del Diseño.

El diseño lógico se introduce en el computador a través de un archivo puente o de entrada Típicamente, se introduce alguna información preliminar, tal como el nombre del usuario, la empresa, la fecha y la descripción del diseño. Luego se introduce el tipo de dispositivo PLD, la numeración de los pines de entrada y salida, y las variables. Por último, se introduce la función o funciones lógicas en forma de ecuaciones booleana, tabla de verdad, esquema o cualquier otro formato.

Cualquier error de sintaxis, u otro tipo de error cometido durante la introducción de datos en el archivo de entrada, será indicado y deberá corregirse. La sintaxis es el formato prescrito y la simbología que se usa describir una categoría de funciones.

5.7 Ejecución Del Software.

El compilador de software procesa y traduce el archivo de entrada y minimiza la lógica. Entonces, el diseño lógico se simula utilizando un conjunto de entradas hipotéticas conocidas como vectores de prueba. Este proceso "ejercita" el diseño de forma efectiva por software para determinar si trabaja correctamente, antes de programar realmente el PLD. Si se descubre cualquier defecto de diseño durante la simulación software, el diseño debe depurarse y modificarse para corregir dicho defecto.

Una vez terminado el diseño, el compilador genera un archivo de documentación, que incluye las ecuaciones lógicas finales, el archivo

JEDEC y, si se desea, un diagrama del patillaje del PLD. El archivo de documentación, esencialmente, facilita una copia en papel del diseño final.

5.8 Programación Del Dispositivo.

Cuando se ha terminado el diseño, el compilador crea un mapa de fusibles (archivo JEDEC) que se carga en el programador. Este archivo dice al programador que fusibles hay que fundir y que fusibles quedan intactos o, en el caso de una GAL, que celdas e CMOS se activan y cuales se desactivan. Luego el programador modela el dispositivo PLD de acuerdo con el mapa de fusibles.

En la sección se hace una introducción a la programación de los PLDs con el software ABEL.

5.9 Programación Interna Del Sistema.

Los PLDs estándar, incluyendo EEPROMS, PALs y GALs se programan utilizando un programador hardware, como acabamos de ver. Ciertos dispositivos programables, incluyendo las GALs, también pueden emplearse como dispositivos de programación dentro del sistema (isp, in-system programmable); por ejemplo, una versión para programación dentro del sistema de la GAL16V8 se denomina con la referencia ISPGAL16V8. El dispositivo isp es igual que el dispositivo interno, excepto que tiene una lógica adicional para la programación dentro del sistema.

Un dispositivo isp puede programarse como un dispositivo estándar antes de colocarse en una tarjeta de circuito impreso, o puede programarse de varias formas después de haberse montado en el circuito impreso. Los

dispositivos isp pueden programarse junto con otros circuitos dentro de un sistema de las formas siguientes:

*Utilizando un programador antes de montarlo en la tarjeta de circuito impreso.
*Utilizando un equipo automático de pruebas (Ate, Automatic Test Equipment) después de montarlo en la tarjeta.
*Utilizando un computador personal después de montarlo en la tarjeta.
*Utilizando un microprocesador interno del sistema (Incorporado) después del montaje en la tarjeta.

5.10 Software De Los PLDs.

5.10.1 El Lenguaje ABEL.

ABEL, que es el acrónimo de Advanced Boolean Expression Language, permite implementar diseños lógicos en dispositivos lógicos programables. ABEL se puede utilizar para programar cualquier tipo de PLD y, por tanto, es un lenguaje independiente del dispositivo. Como se explicó en la sección anterior, el lenguaje ABEL se ejecuta en un computador conectado a un programador de dispositivos, independiente del lenguaje, en el que se inserta el PLD.

5.10.2 Introducción Del Diseño Lógico.

ABEL proporciona tres diferentes formatos para describir e introducir un diseño lógico desde el teclado de un computador: ecuaciones, tablas de verdad y diagramas de estado. Las ecuaciones y las tablas de verdad se usan en los diseños lógicos combinacionales. El diagrama de estados, así como los otros dos formatos, se pueden utilizar para el diseño de lógica secuencial.

5.10.3. Simulación Del Diseño.

Una vez que se ha introducido el diseño lógico, se puede simular su funcionamiento utilizando vectores de prueba para asegurarse de que no existen errores en el diseño. Básicamente, una sección de vectores de prueba enumera todos los posibles valores de entrada y los correspondientes valores de salida. El software, esencialmente, "ejercita" el diseño lógico para asegurarse de que trabaja como se esperaba, tomando

todas las posibles combinaciones de entrada y comprobando los resultados de salida.

5.10.4 Síntesis Lógica.

El proceso software que convierte la descripción del circuito en formato de ecuaciones, tablas de verdad o diagrama de estado, en el fichero JEDEC estándar requerido para programar un PLD se denomina síntesis lógica. Usualmente, la síntesis lógica incluye la optimización y la minimización del diseño.

5.10.5 Operaciones Booleanas.

Como muestra la tabla 4-1 por orden de prioridad, las operaciones NOT (inversión), AND, OR y OR-EXCLUSIVA utilizan unos símbolos especiales en el lenguaje ABEL.

Operación Lógica	Simbolo ABEL
NOT	!
AND	&
OR	#
XOR	$

Tabla 5.1

A continuación se detallan ejemplos de las operaciones lógicas en notación estándar booleana y en lenguaje ABEL:

Notación booleana estándar		ABEL
$\overline{A}$	NOT	!A
A•B	AND	A&B
A+B	OR	A#B
$A \oplus B$	EXOR	A$B

5.10.6 Ecuaciones.

Uno de los métodos de entrada de ABEL son las ecuaciones lógicas. El signo igual (=) es el mismo que en las ecuaciones estándar y, cualquier letra o combinación de letras y números puede emplearse como identificadores de variables. Respecto a los identificadores de las variables, ABEL diferencia entre mayúsculas. Por ejemplo, en ABEL la letra mayúscula a y la minúscula a son dos variables diferentes.

Todas las ecuaciones en ABEL deben terminarse con un punto y coma (;) por ejemplo, la suma de productos x=ab+ab+ab se escribe en lenguaje ABEL del siguiente modo:

X=a&b#!a&b#a&!b;

Conjuntos. En algunos casos, las variables de entrada o salida múltiples se pueden agrupar como un conjunto utilizando corchetes, para simplificar una ecuación. Por ejemplo, un grupo de entradas denominadas

a3, a2, a1, a0, pueden definirse como una única variable a utilizando la notación de conjuntos de ABEL, del siguiente modo:
A=[a3,a2,a1,a0];

5.10.7 Tablas De Verdad.

Otro método de entrada de ABEL utiliza las tablas de verdad. Una función lógica especifica puede pasarse a ecuación, como hemos visto, o a tabla de verdad. En cuanto al resultado, ambos métodos son equivalentes. Usualmente, las tablas de verdad son mas adecuadas que las ecuaciones para describir ciertos tipos de circuitos, tales como decodificadores.

La tabla de verdad en ABEL se define mediante una cabecera y las entradas de la tabla. El formato de la cabecera se especifica como se indica en la siguiente expresión:

```
Truth_table ([a,b,c,d] -> [x1,x2])
```

Donde a, b, c y d son las entradas y x1 y x2 son las salidas. El símbolo -> es un operador que indica las entradas que producen las salidas combinacionales. Para ilustrar como se elabora una tabla de verdad en lenguaje ABEL, vamos a hacer un sencillo ejemplo de una puerta OR-EXCLUSIVA con entradas a y b y salida x:

```
Truth_table ([a,b] -> [x])
                    [1,0] -> [1];
                    [0,1] -> [1];
                    [0,0] -> [0];
                    [1,1] -> [0];
```

5.10.8 Vectores De Prueba.

Como ya sabe, los vectores de prueba se usan en ABEL para describir las señales de entrada y sus correspondientes salidas, con vistas a realizar la simulación lógica. En otras palabras, los vectores de prueba examinan el diseño lógico antes de programar el hardware, aplicando todas las posibles combinaciones de entrada y comprobando que las salidas son las correctas.

Esencialmente, los vectores de prueba son los mismos que las tablas de verdad y se introducen con el mismo formato. Por ejemplo, los vectores de prueba del decodificador bcd-7 segmentos son:

```
Test_vectors ([d,c,b,a] -> [a,b,c,d,e,f,g])
[0,0,0,0] -> [1,1,1,1,1,1,0];
[0,0,0,1] -> [0,1,1,0,0,0,0];
[0,0,1,0] -> [1,1,0,1,1,0,1];
[0,0,1,1] -> [1,1,1,1,0,0,1];
[0,1,0,0] -> [0,1,1,0,0,1,1];
[0,1,0,1] -> [1,0,1,1,0,1,1];
[0,1,1,0] -> [1,0,1,1,1,1,1];
[0,1,1,1] -> [1,1,1,0,0,0,0];
[1,0,0,0] -> [1,1,1,1,1,1,1];
[1,0,0,1] -> [1,1,1,1,0,1,1];
```

Los vectores se pueden simplificar utilizando conjuntos, al igual que las tablas de verdad.

5.10.9 El archivo de entrada ABEL.

Cuando se crea un archivo de entrada (fuente) en ABEL, se crea un modulo que contiene tres secciones básicas a continuación del nombre del módulo y el título: las declaraciones, las descripciones lógicas y los vectores de prueba.

5.10.10 Declaraciones.

Generalmente, la sección de declaraciones incluye la declaración del dispositivo, las declaraciones de los pines y las declaraciones de conjuntos. En los ejemplos se han utilizado las declaraciones de conjuntos.

La declaración del dispositivo se utiliza para especificar el PLD que se va a programar, al que algunas veces se denomina dispositivo objeto (target device). Un ejemplo de una sentencia de declaración de dispositivo es:

Decoder device 'p22v10';

La primera palabra es simplemente una descripción y puede ser la que se desee. La segunda palabra es una palabra clave y puede aparecer en mayúsculas o en minúsculas. El dispositivo debe especificarse de la forma que se indica, utilizando el prefijo p para las PALs y las GALs.

Un ejemplo de sentencia de declaración de pines es:

A0, a1, a2, a3, pin 1, 2, 3, 4;

La palabra clave pin debe escribirse en mayúsculas. Los números especificados detrás de esta palabra son los pines del dispositivo PLD que se va a programar. A0 corresponde al pin 1, a1 al pin 2, a2 al pin 3 y a3 al pin 4.

5.10.11 Descripciones lógicas.

Los métodos de entrada que se han descrito, ecuaciones y tablas de verdad, son dos tipos de descripciones lógicas. Existe un tercer tipo que es el diagrama de estados.

5.10.12 Vectores de prueba.

Como se ha dicho anteriormente, la sección de los vectores de prueba sirve para comprobar un diseño lógico antes de implementarlo en hardware. Los vectores de prueba tienen el mismo formato que las tablas de verdad.

5.10.13 El archivo de documentación.

Después de que el archivo de entrada se ha procesado, ABEL genera un archivo de documentación. Este archivo proporciona una copia en papel de las ecuaciones finales reducidas, un archivo JEDEC, y un diagrama del patillaje del dispositivo.

El archivo JEDEC. Como ya hemos comentado, esta parte del archivo de documentación. Tambien se denomina mapa de celdas o mapa de fusibles. El archivo JEDEC se carga en el programador para programar el

dispositivo objeto. Los archivos JEDEC están estandarizados para todos los lenguajes de programación.

El mapa de celdas muestra aquellas celdas de la matriz del dispositivo que están desconectadas (desactivadas o fundidas) y aquellas que están conectadas (activas o intactos). Un 0 en un mapa de celdas indica una celda que está conectada y un 1 indica una celda que está desconectada.

Capitulo 6 Aplicación Combinacional con PLD'S.

6.1 Ejemplo partiendo de un Diagrama esquemático.

Para desarrollar esta aplicación es necesario utilizar el programa Schematic Design Flow incluido en el programa ispEXPERT System para dibujar las compuertas del circuito incluyendo etiquetas de entradas y salida, en este ejercicio iniciaremos con el diseño tradicional hasta obtener el diagrama esquemático y posteriormente el diseño se capturara en el programa Schematic Design Flow para posteriormente compilarlo y obtener el archivo JEDEC necesario para programar el GAL.

Diseñar un comparador binario de dos números de dos bit's cada numero

a) Diagrama de Bloques

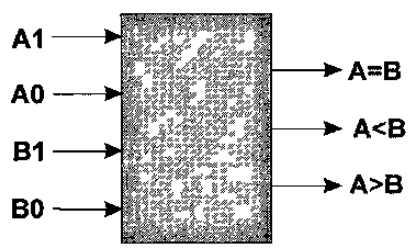


FIG. 6.1

b) Tabla de Verdad

	A1	A0	B1	B0	A > B	A < B	A = B
0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
2	0	0	1	0	0	1	0
3	0	0	1	1	0	1	0
4	0	1	0	0	1	0	0
5	0	1	0	1	0	0	1
6	0	1	1	0	0	1	0
7	0	1	1	1	0	1	0
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
10	1	0	1	0	0	0	1
11	1	0	1	1	0	1	0
12	1	1	0	0	1	0	0
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	0
15	1	1	1	1	0	0	1

FIG. 6.2

c) Ecuaciones

A>B = A0 B1'B0' + A1 B1' + A1 A0 B0'

A<B = A1'A0'B0 + A1'B1 + A0'B1 B0

A=B = A1'A0'B1'B0' + A1'A0 B1'B0 + A1 A0'B1 B0' + A1 A0 B1 B0

d) Diagrama Esquemático.

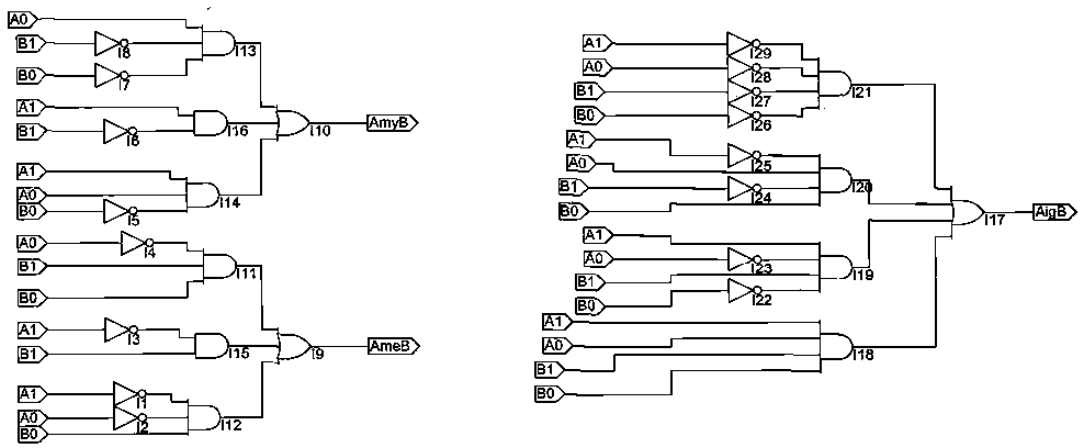


Figura 6.3

e) en el programa ispEXPERT System ejecutar los siguientes pasos:

1.-Crear un nuevo documento llamado comp2b

2.-Seleccionar el dispositivo GAL16V8ZD

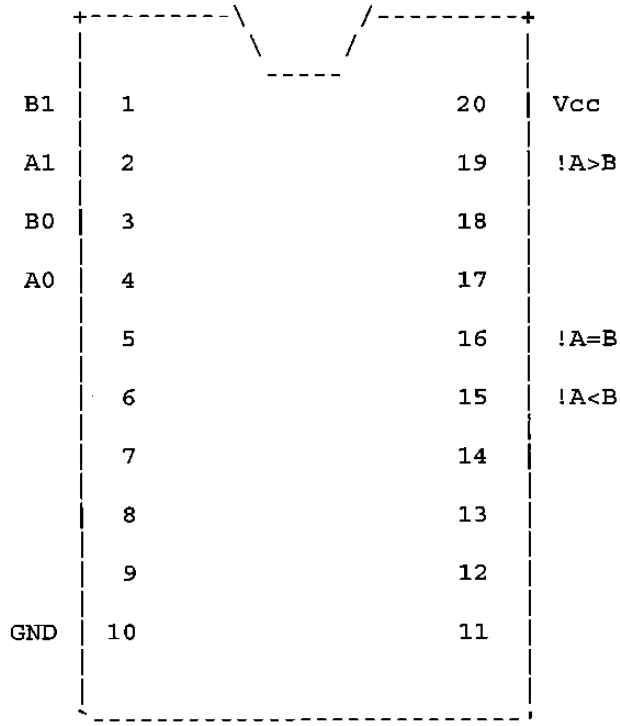
3.-Importar el archivo que contiene el diagrama esquematico.

4.- Compilar para obtener los archivos de reporte y JEDEC.

f) Archivo comp2b.rep

```
Page 1
ispEXPERT 7.1  -  Device Utilization Chart          Wed Dec 01 14:17:48
1999
Module                    : 'comp2b'
Input files:
    ABEL PLA file         : comp2b.tt3
    Device library        : P16V8AS.dev
Output files:
    Report file           : comp2b.rep
    Programmer load file  : comp2b.jed
```

```
Page 2
ispEXPERT 7.1  -  Device Utilization Chart    Wed Dec 01 14:17:48 1999
P16V8AS Programmed Logic:
AmyB   = (  !B1 & A1 #   !B1 & !B0 & A0 #   A1 & !B0 & A0 );
AmeB   = (  B1 & !A1   #   B1 & B0 & !A0  #   !A1 & B0 & !A0 );
AigB   = !( !B1 & A1   #   B1 & !A1   #   !B0 & A0 #   B0 & !A0 );

Page 3
ispEXPERT 7.1  -  Device Utilization Chart    Wed Dec 01 14:17:48 1999
P16V8AS Chip Diagram:
 P16V8AS
```



SIGNATURE: N/A

Page 4
ispEXPERT 7.1 - Device Utilization Chart Wed Dec 01 14:17:48 1999
P16V8AS Resource Allocations:

Device Resources	Resource Available	Design Requirement	Unused
Input Pins:			
Input:	10	4	6 (60 %)
Output Pins:			
In/Out:	6	1	5 (83 %)
Output:	2	2	0 (0 %)
Buried Nodes:			
Input Reg:	-	-	-
Pin Reg:	-	-	-
Buried Reg:	-	-	-

```
Page 5
ispEXPERT 7.1  -  Device Utilization Chart          Wed Dec 01 14:17:48
1999

c2n2b.bls

P16V8AS Product Terms Distribution:
------------------------------------------------------------------------
-------

            Signal                          |    Pin    | Terms | Terms |
Terms
             Name                           | Assigned  | Used  |  Max  |
Unused
===========================================|===========|=======|=======|====
===
A>B                                         |    19     |   3   |   8   |   5
A<B                                         |    15     |   3   |   8   |   5
A=B                                         |    16     |   4   |   8   |   4

     ==== List of Inputs/Feedbacks ====

Signal Name                                 | Pin       | Pin Type
============================================|===========|=========
B1                                          |     1     | INPUT
A1                                          |     2     | INPUT
B0                                          |     3     | INPUT
A0                                          |     4     | INPUT


Page 6
ispEXPERT 7.1  -  Device Utilization Chart          Wed Dec 01 14:17:48
1999

c2n2b.bls

P16V8AS Unused Resources:
------------------------------------------------------------------------

 Pin    |  Pin    |   Product     | Flip-flop
Number  |  Type   |   Terms       |   Type
========|=========|===============|==========
    5   |  INPUT  |       -       |     -
    6   |  INPUT  |       -       |     -
    7   |  INPUT  |       -       |     -
    8   |  INPUT  |       -       |     -
    9   |  INPUT  |       -       |     -
   11   |  INPUT  |       -       |     -
   12   |  BIDIR  | NORMAL  8     |     -
   13   |  BIDIR  | NORMAL  8     |     -
   14   |  BIDIR  | NORMAL  8     |     -
   17   |  BIDIR  | NORMAL  8     |     -
   18   |  BIDIR  | NORMAL  8     |     -
```

g) Archivo JEDEC

```
 ispEXPERT 7.1 Data I/O Corp.
JEDEC file for: P16V8AS V9.0
Created on: Wed Dec 01 14:17:48 1999
*
QP20* QF2194* QV0* F0*
 X0*
NOTE Table of pin names and numbers*
NOTE PINS AigB:16 AmeB:15 AmyB:19 B1:1 A1:2 A0:4 B0:3*
L0000 01101111111111111111111111111111*
L0032 11101011011111111111111111111111*
L0064 01111011011111111111111111111111*
L0768 01101111111111111111111111111111*
L0800 10011111111111111111111111111111*
L0832 11111011011111111111111111111111*
L0864 11110111101111111111111111111111*
L1024 10011111111111111111111111111111*
L1056 11010111101111111111111111111111*
L1088 10110111101111111111111111111111*
L2048 10001000*
L2128 11111111111111111111111111111111111111111111111111111111111111111111*
L2192 1*
C2F1C*
 9C8D
```

6.2 a partir de una tabla de verdad

es necesario elaborar un documento en el lenguaje ABEL PRIMER HDL que contenga la tabla de verdad

```
MODULE comp
DECLARATIONS
" variables de entrada
      A1,A0, B1, B0 pin 1,2,3,4;

" variables de salida
      X, Y, Z    pin 19,18,17;

TRUTH_TABLE ([A1, A0, B1, B0] -> [ X, Y, Z])
            [0,   0,  0,  0] ->  [ 0, 0,  1];
            [0,   0,  0,  1] ->  [ 0, 1, 0 ];
            [0,   0,  1,  0] ->  [ 0, 1, 0 ];
            [0,   0,  1,  1] ->  [ 0, 1, 0 ];
            [0,   1,  0,  0] ->  [ 1, 0, 0 ];
            [0,   1,  0,  1] ->  [ 0, 0, 1 ];
            [0,   1,  1,  0] ->  [ 0, 1, 0 ];
            [0,   1,  1,  1] ->  [ 0, 1, 0 ];
            [1,   0,  0,  0] ->  [ 1, 0, 0 ];
            [1,   0,  0,  1] ->  [ 1, 1, 0 ];
            [1,   0,  1,  0] ->  [ 0, 0, 1 ];
            [1,   0,  1,  1] ->  [ 0, 1, 0 ];
            [1,   1,  0,  0] ->  [ 1, 0, 0 ];
            [1,   1,  0,  1] ->  [ 1, 0, 0 ];
            [1,   1,  1,  0] ->  [ 1, 0, 0 ];
            [1,   1,  1,  1] ->  [ 0, 0, 1 ];

END comp
```

6.3 a partir de ecuaciones

A continuación se muestra el archivo HDL conteniendo las ecuaciones del diseño del comparador de dos numeros binarios de dos bit´s cada numero.

```
MODULE comp

DECLARATIONS

" variables de Entrada
     A1,A0, B1, B0 pin 1,2,3,4;

" variables de Salida
     A=B, A>B, A<B   pin 19,18,17;

EQUATIONS

     A=B = !A1 & !A0 & !B1 & !B0 # !A1 & A0 & !B1 &B0
           # A1 & !A0 & B1 & !B0 # A1 & A0 & B1 & B0;
     A>B= A0& ¡B1 &!B0 # A1 & !B1 # A1 & A0 !B0;
     A<B=!A1 & !A0 & B0 # !A1 & B1 # !A0 & B1 & B0

END comp
```

En las tres formas descritas a partir de Diagrama Esquemático, Tabla de Verdad y Ecuaciones se obtienen los mismos resultados queda a elección del diseñador elegir el método mas apropiado para su diseño.
Una limitante seria la de que en el diseño se requieran mas términos productos de los que soporta el dispositivo, esto lo indicaría en el proceso de compilación y para estos casos se recomienda utilizar un dispositivo con mas capacidad, y si aun no es suficiente el dispositivo mas grande existente en el mercado se sugiere utilizar otra tecnología de dispositivos lógicos programables como los ISP (In System Program) en los cuales se utilizarían los mismos archivos HDL.

Capitulo 7 Aplicación Secuencial.

7.1 Diseño de un Sistema Secuencial para un concurso.

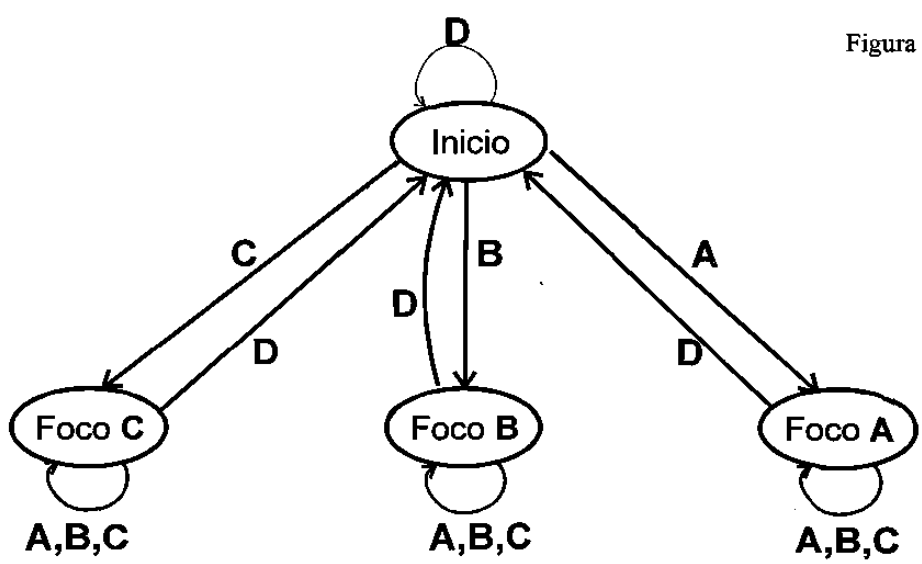
Introducción: En este capítulo se va a aplicar el concepto de Diseño Lógico Programable a un concurso llamado Jeopardi, en este concurso los participantes, al terminar de escuchar una pregunta del conductor deben oprimir un botón para tener la opción de responder a la pregunta. El concursante que oprima primero el botón, y por tanto encienda su foco, tiene la oportunidad de contestar la pregunta.

7.2 Descripción del problema

Lo que se va a diseñar es un sistema Secuencial que indique por medio de tres focos (Foco A, Foco B y Foco C) cual de los tres participantes oprimiò primero el botòn(hay un Botón A un Botón B y un Botón C), además debe haber un cuarto botón (Botón D) para que el conductor del programa una vez terminada la respuesta regrese al sistema a condiciones iniciales(Focos apagados).

7.3 Diagrama de estados.

Figura 7.1



Partiendo del Estado de Inicio:

- a). Si se oprime el botón A el estado próximo será **Foco A** que tiene como finalidad encender solo el foco A.
- b) Si se oprime el botón B el estado próximo será **Foco B** que tiene como finalidad encender solo el foco B.
- c) Si se oprime el botón C el estado próximo será **Foco C** que tiene como finalidad encender solo el foco C.
- d) Si se oprime el botón D el sistema permanecerá en el mismo estado.
  - Una vez estando en cualquiera de los estados **Foco A, Foco B** o **Foco C**, si se oprimen los botones A , B o C, el sistema permanecerá en el mismo estado.
  - Pero si se oprime el botón D el sistema regresará al estado de **Inicio.**

7.4 Archivo en formato ABEL-PRIMER Jeopardy.abl

Este archivo debe contener las siguientes partes:

Modulo: Aquì se incluye la identificaciòn del archivo y el PLD que se va a Utilizar.

Ecuaciones: Aquì aparecen las ecuaciones en lenguaje ABEL.

Descripciòn de los pines (pin out): Aquì aparece un diagrama del chip con los pines que se van a utilizar como entradas y como salidas.

Descripciòn de la utilizaciòn del chip:Aquì aparecen los pines que se utilizaron (en cuanto a cantidad) como entradas y salidas y el porcentaje de utilizaciòn del chip.

Los reportes aparecen a partir de la siguiente página.

```
MODULE Jeopardy

TITLE 'se tienen 3 concursantes A,B,C y el conductor D'

DECLARATIONS
"clock name
        CLK PIN 1;
"Input variables
        A,B,C,D PIN 2,3,4,5;
"Output variables
        FB PIN 12 ISTYPE 'com';
        FC PIN 13 ISTYPE 'com';
        FA PIN 14 ISTYPE 'com';
"State variables
        S0  PIN ISTYPE 'reg';
        S1  PIN ISTYPE 'reg';

"State Register assignment
DECLARATIONS
        sreg=[ S0,S1];

EQUATIONS
        sreg.clk=CLK;
DECLARATIONS
        CI=[0, 0];
        FOCO_A=[0, 1];
        FOCO_B=[1, 0];
        FOCO_C=[1, 1];

state_diagram sreg;
        state FOCO_A:
        FA=1;
        FB=0;
        FC=0;
        IF D THEN CI Else FOCO_A;
        state FOCO_B:
        FA=0;
        FB=1;
        FC=0;
        IF D THEN CI Else FOCO_B;
        state FOCO_C:
        FA=0;
        FB=0;
        FC=1;
        IF D THEN CI Else FOCO_C;
        state CI:
        FA=0;
        FB=0;
        FC=0;
        IF A Then FOCO_A;
        IF B Then FOCO_B;
        IF C Then FOCO_C;
        IF D THEN CI;

END
```

7.5 Archivo de reporte Jeopardy.rep

Page 1
ispEXPERT 7.1 - Device Utilization Chart Fri Oct 01 07:46:52 1999

se tienen 3 concursantes A,B,C y el conductor D

Module : 'jeopardy'

Input files:

ABEL PLA file : jeopardy.tt3
Device library : P16V8R.dev

Output files:

Report file : jeopardy.rep
Programmer load file : jeopardy.jed

---

ispEXPERT 7.1 - Device Utilization Chart Fri Oct 01 07:46:52 1999

se tienen 3 concursantes A,B,C y el conductor D

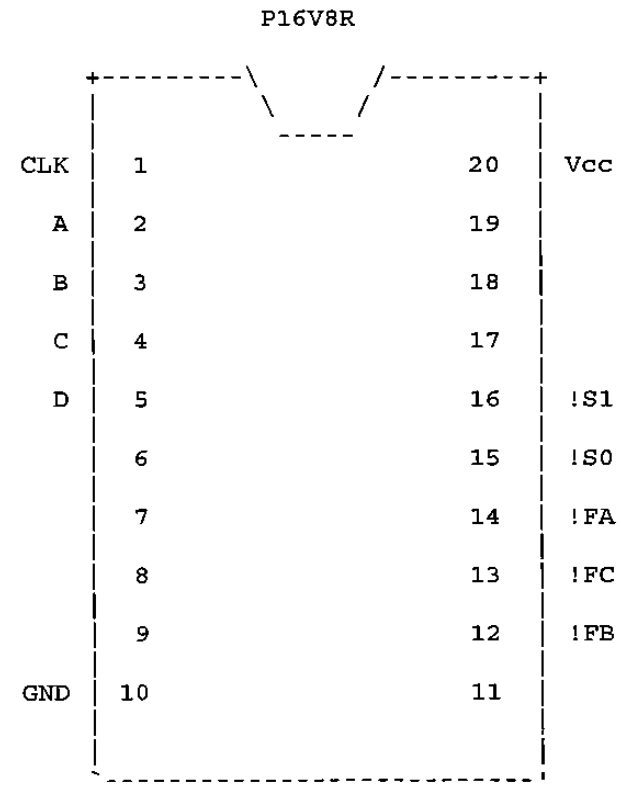
P16V8R Programmed Logic:

```
FB    = (  !S0.Q & S1.Q );

FC    = (  !S0.Q & !S1.Q );

FA    = (  S0.Q & !S1.Q );

S0.D  = (  S0.Q & !S1.Q
       #   !S0.Q & D
       #   S0.Q & !B & !C ); " ISTYPE 'INVERT'
S0.C  = (  CLK );

S1.D  = (  !S0.Q & S1.Q
       #   !S1.Q & D
       #   S1.Q & !A & !C ); " ISTYPE 'INVERT'
S1.C  = (  CLK );
```

Page 3
ispEXPERT 7.1 - Device Utilization Chart Fri Oct 01 07:46:52 1999

se tienen 3 concursantes A,B,C y el conductor D

P16V8R Chip Diagram:



SIGNATURE: N/A

Page 4
ispEXPERT 7.1 - Device Utilization Chart Fri Oct 01 07:46:52 1999

se tienen 3 concursantes A,B,C y el conductor D

P16V8R Resource Allocations:

Device Resources	Resource Available	Design Requirement	Unused
Input Pins:			
Input:	10	5	5 (50 %)
Output Pins:			
In/Out:	8	5	3 (37 %)
Output:	-	-	-
Buried Nodes:			
Input Reg:	-	-	-
Pin Reg:	8	2	6 (75 %)
Buried Reg:	-	-	-

Page 5
ispEXPERT 7.1 - Device Utilization Chart Fri Oct 01 07:46:52 1999

se tienen 3 concursantes A,B,C y el conductor D

P16V8R Product Terms Distribution:

Signal Name	Pin Assigned	Terms Used	Terms Max	Terms Unused
FB	12	1	7	6
FC	13	1	7	6
FA	14	1	7	6
S0.D	15	3	8	5
S1.D	16	3	8	5

==== List of Inputs/Feedbacks ====

Signal Name	Pin	Pin Type
CLK	1	CLK
D	5	INPUT
A	2	INPUT
B	3	INPUT
C	4	INPUT

Page 6
ispEXPERT 7.1 - Device Utilization Chart Fri Oct 01 07:46:52 1999

se tienen 3 concursantes A,B,C y el conductor D

P16V8R Unused Resources:

Pin Number	Pin Type	Product Terms	Flip-flop Type
6	INPUT	-	-
7	INPUT	-	-
8	INPUT	-	-
9	INPUT	-	-
17	BIDIR	NORMAL 7	D
18	BIDIR	NORMAL 7	D
19	BIDIR	NORMAL 7	D

7.6 archivo JEDEC Jeopardy.jed

```
lspEXPERT 7.1 Data I/O Corp.
JEDEC file for: P16V8R V9.0
Created on: Fri Oct 01 07:46:52 1999

se tienen 3 concursantes A,B,C y el conductor D
*
QP20* QF2194* QV0* F0*
 X0*
NOTE Table of pin names and numbers*
NOTE PINS CLK:1 A:2 B:3 C:4 D:5 FB:12 FC:13 FA:14 S0:15 S1:16*
L0768 11111111111111101101111111111111*
L0800 11111111111101011111111111111111*
L0832 10111111101111101111111111111111*
L1024 11111111111111011110111111111111*
L1056 11111111111101111101111111111111*
L1088 11111011101111111110111111111111*
L1280 11111111111111111111111111111111*
L1312 11111111111111011110111111111111*
L1536 11111111111111111111111111111111*
L1568 11111111111111011101111111111111*
L1792 11111111111111111111111111111111*
L1824 11111111111111101101111111111111*
L2048 00000111*
L2120 00000111*
L2128 11111111111111111111111111111111111111111111111111111111111111111111*
L2193 1*
C369C*
BC55
```

7.7 Programación del Circuito integrado GAL18V8D.

Para nuestro caso utilizaremos el programador Super Pro III de la compañía XELTEC, en el que el programador se conecta al puerto paralelo de una computadora personal a través de un cable de conectores DB25 M DB25 H, se instala un programa para ejecutar la aplicación llamado SP3, una vez conectado el programador e instalado el software se recomienda seguir los siguientes pasos:

a) Alimentar la energía del programador y encenderlo.

b) Ejecutar desde la computadora el programa SP3. en ese momento se realiza una inicialización del programador y aparece una ventana con opciones.

c) En el programa seleccionar el circuito integrado que se va a programar.

d) Cargar el archivo Jeopardy.jed en el menú de FILE OPEN.

e) Ejecutar el comando programar. La aplicación automáticamente borra el contenido anterior del circuito integrado posteriormente graba el archivo JEDEC , efectúa una verificación y concluye con un mensaje que indica que la programación se realizó con éxito.

Una vez programado el circuito integrado se efectúan las conexiones necesarias basadas en el reporte de IPIN-OUT y el diseño queda terminado.

CAPITULO 8

Conclusiones y recomendaciones

Los PLDs. Que han estado disponibles desde hace algún tiempo son los PLA y los PAL. Los GALs, son innovaciones recientes que prometen un gran impacto en el diseño de sistemas digitales. Un GAL se conoce también como arreglo lógico inconcluso, es básicamente un circuito VLSI que contiene un gran numero de compuertas que no están conectadas para formar un circuito lógico, mas bien, las interconexiones se forman metalizando el circuito utilizando una PC y hardware y software adecuado para generar un arreglo y se realice la función que se desea. El objetivo de esta tesis ha sido el de dar una presentación de la nueva era de los sistemas digitales, que es el Diseño Lógico Programable. Los retos y oportunidades que ésta tecnología ofrece deben haberse hecho evidentes.

En conclusión, vivimos en una era en la que el Diseño Lógico Programable, como parte de la Electrónica Digital, se esta extendiendo cada vez mas en esta área. No es necesario decir que esto presenta grandes oportunidades a aquellas personas que han elegido esta área de estudio. Espero haber transmitido al lector un entusiasmo por el Diseño Lógico Programable.

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN BIBLIOGRAFICO

BIBLIOGRAFIA

Floyd Martin
Sistemas Digitales
Editorial Interamericana
Sexta edición

DIRECCIONES DE INTERNET

http://intranet.uacj.mx/hochoa
http://clotho.ujavcali.edu.co/L.o7

APENDICE A

Glosario de términos.

ABEL. **Advanced Boolean Expressión Languaje. Lenguaje de compilación para la programación de descripción de Hardware.**

Alfanumérico. **Que contiene números, letras y otros caracteres.**

AND. Operacion Logica básica en la que se obtiene una salida verdadera (ALTO) solo si todas las condiciones de entrada son verdaderas.

Buffer. Circuito que evita la carga de una entrada o salida.

Buffer triestado. Circuito lógico que posee tres estados de salida: ALTO, BAJO y de alta impedancia (abierto)

Celda global. Celda programable de una matriz PLD que afecta A TODAS LAS OLMCs cuando se programa.

Celda Local. Celda programable de una matriz PLD que afecta a las OLMCs individuales cuando se programan.

Celda. Una zona de un mapa de Karnaugh que representa una única convinación de variables en forma de productos; elemento de almacenamiento básico en una memoria

Circuito. Disposición de componentes eléctricos y/o electrónicos interconectados de manera que realicen una función específica.

Circuito integrado. (CI) Un tipo de circuito en el que todos sus componentes se encuentran integrados en un único chip semiconductor de muy pequeño tamaño.

Circuito Secuencial. Circuito digital cuyos estados lógicos dependen de una determinada secuencia temporal.

CMOS. Complementary Metal-Oxide semiconductor, semiconductor complementario de metal-óxido: un tipo de circuito de transistores que utiliza transistores de efecto de campo.

Codificador. Circuito (dispositivo) digital que convierte la información a un formato codificado.

Codigo ASCII. American Standart Code for Information Interchange, codigo utilizado por los fabricantes de equipo para intercambio de información.

Codigo Binario. Código en el cual mediante UNOS Y CEROS se representa información en los sitemas digitales.

Codigo BCD. Binary Coded Decimal , código en el cual los numeros decimales del 0 al 9 se representan en codigo binario para representar información.

Código EBCDIC. Extended Binary Coded Decimal Interchange Code, código en el cual todos los caracteres estan representados por ocho bits o dos números hexadecimales,

Código TTY. Tele Type, también llamado código BAUDOT, codigo que utiliza 5 bits por caracter.

Codigo 2421. Codigo BCD en el cual la posicion del bit de mayor peso es 2.

Código. Un conjunto de bits ordenados según un patrón único y utilizados para representar información tal como números, letras y otros símbolos.

CUPL. Compiler for Universal Programmable Logic, compilador de lógica universal programable; Un tipo de lenguaje de descripción hardware.

Decodificador. Circuito (dispositivo) digital que convierte la información en algún otro formato (mas familiar)

Digital. Relacionado con los dígitos o con cantidades discretas; que posee un conjunto de valores discretos, en oposición a tener valores continuos.

EECMOS. Electrecally Erasable CMOS (EECMOS), CMOS eléctricamente borrable. Tecnología de circuitos utilizada para las celdas reprogramables de un GAL.

EEPROM. Electrically Erasable Programmable Read-Only Mémory. Memoria programable de solo lectura electricamente borrable.

EPROM. Erasable Programmable Read_Only Memory, memoria programable de solo lectura borrable.

Exceso 3. Codigo digital sin peso en el que cada uno de los digitos decimales se representa mediante un código de 4 bits sumando 3 al digito decimal y convirtiendolo a binario.
Archivo de documentación. La información de un ordenador que documenta el diseño final posterior al procesamiento del fichero de entrada.

Archivo de entrada. La información introducida en un ordenador que escribe un diseño lógico, utilizando un lenguaje de programación PLD como ABEL.

Archivo JEDEC. Fichero software estandar generado a partir de un software de compilación que se emplea en un dispositivo de programación para implementar un diseño lógico de un PLD; tambien se denomina mapa de fusibles o mapa de celdas.

Flip-flop. Circuito básico de almacenamiento que puede almacenar solo un bit a un tiempo; dispositivo biestable síncrono.

Fusible. Elemento programable en ciertos tipos de PLDs.

GAL. Generic Array Logic, matriz lógica genérica. Dispositivo que tiene una matriz AND reprogramable, una matriz OR fija y macroceldas lógicas programables de salida.

HDL. Hardware Descriptión Languaje, lenguaje de descripción hardware. ABEL y CUPL son ejemplos de este tipo de lenguaje.

JEDEC. Join Electronic Device Engineering Council.

Latch. Dispositivo digital biestable utilizado para almacenar un bit.

Lógica combinacional. Combinacion de puertas lógicas interconectadas para producir una determinada función bbleana sin capacidad de almacenamiento de memoria.

LSI. Large-Scale Integratión, integración a gran escala; un nivbel en la complejidad de Cis que tienen entre 100 y 9999 puertas equivalentes por chip.

Mapa de Karnaugh. Disposición de celdas que representa la combinación de literales de una expresión booleana y que se utiliza para la simplificación sistematica de la expresión.

Maquina de estados. Sistema Lógico que exhibe una secuencia de estados condicionada por la lógica interna y las entradas externas.Cualquier circuito secuencial que exhibe una determinada secuencia de estados.

Matriz de memoria. Matriz formada por las celdas de memoria.

Matriz. En un PLD, una matriz formada por filas de terminos productos y columnas de l´pineas de entrada con una celda programable en caDA UNIÓN.

NOT. Operación lógica básica que realiza inversiones.

OLMC. Output-Logic Macrocell, macrocelda lógica de salida. Salñida programable de un GAL.

OR. Operación lógica básica en la que se obtiene una salida verdadera (ALTO) cuando una o mas de las condiciones de entrada son verdaderas (ALTO)

PAL .Programmable Array Logic, matriz lógica programable. Dispositivo con una matriz AND programmable y una matriz OR fija.

PLA. Programmable Logic Array, matriz lógica programable. Dispositivo con matrices AND y OR programables.

PLD. Programmable Logic Device, dispositivo lógico programable.

Programador. Instrumento que programa PLDs utilizando un fichero JEDEC obtenido de un ordenador en el que esté corriendo software HDL (lenguaje de descripción de hardware).

PROM. Programmable Read-Only semiconductor Mémory, memoria semiconductora programable de solo lectura. Dispositivo con una matriz AND y una matriz OR programmable; se utiliza como dispositivo de memoria, pero, generalmente, no como circuito lógico.

Sintesis. Proceso software por el que se convierte una descripción de circuito a un fichero JEDEC estandar.

Software. Programa de ordenador. Programas que dicen al ordenador que es lo que tiene que hacer para póder realizar un determinado número de tareas.

Triestado. Un tipo de salida en un circuito lógico que posee tres estados: ALTO; BAJO y abierto (desconectado). El estado abierto se denomina "estado de alta impedancia".

TTL. Transistor-Transistor-Logic. Logia transistor-transistor; un tipo de circuito integrado que utiliza transistores bipolares.

ULSI. Ultra Large_Scale Integratión, integracion a ultra gran escala. Un nivel en la complejidad de los Cis que tienen mas de 100,000 puertas equivalentes por chip.

Vectores de prueba. Sección de un programa para PLD que consiste en un conjunto de niveles de entrada y de salida resultantes, y que se utilizan para comprobar un determinado diseño lógico antes de convertirlo en hardware.

VLSI. Very Large Scale Integratión, integración a muy gran escala. Un nivel en la complejidad de los Cis que tienen entre 10,000 y 99,999 puertas equivalentes por chip.

Zócalo ZIF. Zocalo de fuerza inserción cero. Tipo de Zócalo.

Resumen Autobiográfico

Nombre	Miguel Angel Gutiérrez Zamarripa
Nombre de los padres	Juan Gutiérrez Colunga (†) Antonia Zamarripa de Gutiérrez
Lugar y fecha de Nacimiento	León Gto. 2 de octubre de 1961
Grado de escolaridad	Ingeniero Electricista Ingeniero en Electrónica y Comunicaciones
Campo profesional	Catedrático de la Facultad de Ingeniería Mecánica y Eléctrica, Universidad Autónoma de Nuevo León, impartiendo clases de, Química, Algebra, Matemáticas 2, Matemáticas1 y Física 4 Desde 1991 ha la fecha. Auxiliar en el Departamento de Recursos Humanos desde 1986 hasta 1997.
Titulo que desea obtener	Maestro en Ciencias de la Ingeniería Eléctrica con especialidad en Electrónica.
Nombre de la tesis	Aplicaciones del Diseño Lógico Programable.