UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA
Y ELECTRICA

DIVISION DE ESTUDIOS DE POST-GRADO

DISEÑO LOGICO PROGRAMABLE

POR

ING. JUAN ANGEL GARZA GARZA

TESIS

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

CD. UNIVERSITARIA, JUNIO DE 1998

GARZA GARZA

DISEÑO LOGICO PROGRAMABLE


1020122998

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA
Y ELECTRICA

DIVISION DE ESTUDIOS DE POST-GRADO

DISEÑO LOGICO PROGRAMABLE

POR

ING. JUAN ANGEL GARZA GARZA

TESIS

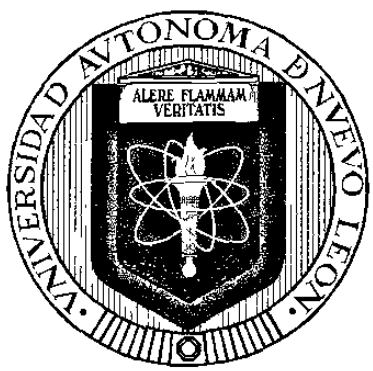
EN OPCION AL GRADO DE MAESTRO EN
CIENCIAS DE LA INGENIERIA ELECTRICA
CON ESPECIALIDAD EN ELECTRONICA

CD. UNIVERSITARIA, JUNIO DE 1998

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POST-GRADO



DISEÑO LOGICO PROGRAMABLE

TESIS

EN OPCIÓN AL GRADO DE MAESTRO EN CIENCIAS DE LA INGENIERÍA ELÉCTRICA CON ESPECIALIDAD EN ELECTRÓNICA

QUE PRESENTA EL

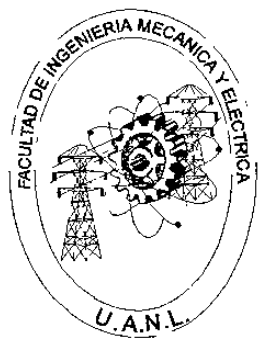
ING. JUAN ANGEL GARZA GARZA

CD. UNIVERSITARIA JUNIO DE 1998

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POST-GRADO



DISEÑO LOGICO PROGRAMABLE

TESIS

EN OPCIÓN AL GRADO DE MAESTRO EN CIENCIAS DE LA INGENIERÍA ELÉCTRICA CON ESPECIALIDAD EN ELECTRÓNICA

QUE PRESENTA EL

ING. JUAN ANGEL GARZA GARZA

CD. UNIVERSITARIA JUNIO DE 1998

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVSION DE ESTUDIOS DE POS-TGRADO

Los miembros del Comité de Tesis recomendamos que la Tesis **DISEÑO LOGICO PROGRAMABLE,** realizada por el Ing. Juan Angel Garza Garza sea aceptada para su defensa como opción al grado de Maestro en Ciencias de la Ingeniería Eléctrica con especialidad en Electrónica.

El Comité de Tesis

M.C. Luis Manuel Camacho Velázquez
Asesor

M.C. Juan Diego Garza González
Coasesor

M.C. Luis Manuel Martínez Villarreal
Coasesor

Vo.Bo.
M.C. Roberto Villarreal Garza
División de Estudios de Pos-tgrado

San Nicolás de los Garza, N.L., Mayo de 1998

DEDICATORIA

A Marina mi Esposa, por su valiosa colaboración en todos los proyectos de mi vida, quien sin su gran ayuda jamas hubiéramos logrado lo que hasta ahorita tenemos.

A mis Hijos Juan Angel, Eduardo Nicolás y Jesús Daniel, que son el proyecto más hermoso de mi vida.

A mi Madre por contar siempre con su cariño y sus consejos.

A la memoria de mi Padre un maravilloso ejemplo de quien tengo gratos recuerdos y siempre estará entre nosotros.

A mis Hermanos María Guadalupe, Nicolás, Martha Alicia, Blanca Elena, Rocío y a mi Tío Juan Diego, por contar siempre con ellos en las buenas y en las malas.

Agradecimientos

Al Ing. Castulo E. Vela Villarreal por el apoyo en la realización de esta tesis.

A mis Asesores M. en C. Luis Manuel Camacho Velázquez, M. en C. Luis Manuel Martínez Villarreal y M. en C. Juan Diego Garza González, por las sugerencias y observaciones realizadas a este trabajo.

A mi compañero y amigo Ing. Víctor Manuel Calderón González por su paciencia y valiosas aportaciones a este proyecto.

Al Lic. Julio Cesar Méndez, Prof. Román Guerrero, Diana Gallegos y Cristina E. García por sus contribuciones constructivas en la realización de este trabajo.

Prólogo

Esta tesis trata sobre un tema que está en constante evolución y existe actualmente gran diversidad y una amplia información de los Dispositivos Lógicos Programables así como también acerca de las herramientas de programación.

El propósito de desarrollar esta tesis es aportar un documento que contenga la información sobre los Dispositivos Lógicos Programables, Herramientas y Metodología de Diseño, que actualmente se utilizan como tecnología de punta los desarrolladores en la fabricación de equipos electrónicos y también pueda ser usada como material didáctico en el desarrollo de proyectos en el área de Electrónica Digital.

Los temas fueron desarrollados de tal manera que se pueda conocer la evolución del diseño lógico programable para conocer todas las etapas, desde los archivos de información de diseño generados a partir de un editor, hasta los Lenguajes de Descripción de Hardware.

Ing. Juan Angel Garza Garza

INDICE

CAPITULO 5

CAPITULO 6

CAPITULO 7

CAPITULO 8

CAPITULO 1 Síntesis

1.1 Síntesis

Esta tesis contiene una descripción y aplicación de las actuales herramientas del Diseño Digital usando aplicaciones en la Computadora Personal (software) y el equipo periférico de programación y lectura (hardware).

La metodología propuesta consiste en que a partir de las ecuaciones, Diagrama de Transición o de una tabla de verdad, auxiliado de una aplicación de la Computadora Personal (software) se obtengan archivos de Lenguaje de Descripción de Hardware HDL en formatos como VHDL (VHSIC Very High Speed Integrated Circuit, Hardware Descripción Language) o ABEL (Advanced Boolean Expression Language), que contienen las ecuaciones con la finalidad de modelar y facilitar su simulación, partiendo de esos formatos de archivos y con la ayuda de un compilador se genera el archivo conocido como JEDEC (Joint Electron Device Engineering Concil) que proporciona las especificaciones en la comunicación con los equipos de programación de dispositivos y con la ayuda de un programador es posible obtener un dispositivo con las especificaciones del diseño y posteriormente implementarlo.

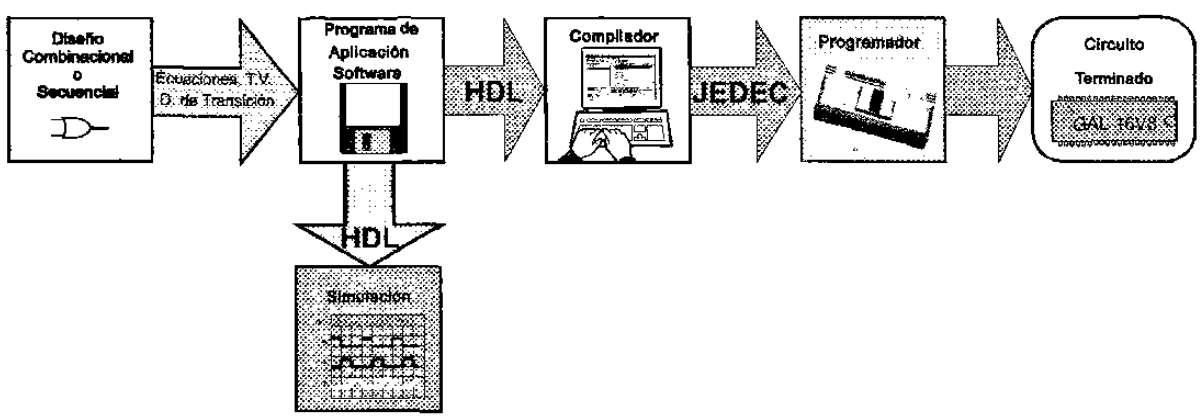


Fig. 1.1 Secuencia de programación de los Dispositivos Lógicos

CAPITULO 2 Introducción

2.1 Introducción

Vivimos en la era de la Informática, la cual en los últimos diez años ha tenido un gran crecimiento, junto con la electrónica y las comunicaciones, desarrollos que nos facilitan nuestras tareas intelectuales en una forma que puede compararse a la aplicación de la máquina de vapor a la tecnología rudimentaria de las máquinas mecánicas, en aligerar el esfuerzo físico del trabajo humano, que hizo su aparición en la revolución industrial del siglo XIX.

El Incremento de popularidad de los dispositivos lógicos programables o PLDs está siguiendo un proceso solamente comparable al que hace algunos años acompañó a los microprocesadores.

Los PLDs se utilizan en casi todos los nuevos equipos electrónicos de control, industriales, de consumo, de oficina, de comunicaciones, tarjetas para computadora, etc.

En fin, los PLDs facilitan el proceso de diseño, eliminan los problemas en la producción de equipo y reducen el tiempo en los nuevos desarrollos.

Con la gran ayuda de la computadora Personal y sus programas de aplicación es posible llevar a cabo todo el proceso de diseño en poco tiempo y en un escritorio.

2.2 Objetivos

El objetivo principal de esta tesis es el dar a conocer y aplicar las herramientas de diseño hardware, software y metodología a ser usadas en la enseñanza de la electrónica digital y que los alumnos tengan al alcance la tecnología de punta usada en la industria en la fabricación de sistemas digitales.

Este trabajo está dirigido a todas aquellas personas que deseen iniciarse en el ámbito de la lógica programable y a los usuarios de lógica programable interesados en obtener mayores beneficios de esta tecnología. También puede resultar adecuado para profesores y profesionales de área que no han tenido oportunidad de recibir información acerca de los PLDs o que deseen aumentar los conocimientos sobre este tema.

2.3 Metodología

La metodología usada en esta tesis radica en la aplicación del conocimiento en forma inmediata (teórico práctico), con la finalidad de facilitar el diseño de Sistemas Digitales utilizando los Dispositivos Lógicos Programables, partiendo de los conocimientos básicos en el diseño de la Electrónica Lógica en los Sistemas Combinacionales y Sistemas Secuenciales y aprovechando las herramientas de desarrollo de aplicaciones en la Computadora Personal como auxiliar en el diseño reduce considerablemente el tiempo en obtener el producto terminado, así como aumenta la capacidad de solucionar problemas más complejos.

2.4 Resumen Bibliográfico

Con la gran ayuda del Internet se puede encontrar información sobre la tecnología de punta en el diseño Lógico Programable, los manuales de los fabricantes de los PLDs como Lattice Semiconductor que proporciona en un CDROM la información técnica y metodología de diseño.

Además de una aplicación de la computadora personal llamada Synario en la que a partir de un lenguaje de programación como ABEL-HDL se puede obtener el archivo JEDEC necesario en la programación de Circuitos Integrados, AMD (Advanced Micro Devices) con su manual PAL Devices Data Book and Desing Guide 1996, donde se propone una metodología más básica en la que a partir de un editor de texto se generan los archivos en formato PLD y mediante un compilador se obtienen los archivos JEDEC.

Los Libros como Digital System Design with Programable Logic del autor Martin Bolton del Editorial Addison Wesley, que describe la arquitectura básica de los Dispositivos Lógicos Programables y el Libro Lógica Programable del autor Mariano Barrón Ruiz, Editorial Mc Graw Hill, que contiene una muy completa información acerca de los diferentes tipos de PLDs y las técnicas de la programación de Dispositivos Lógicos Programables, así como una explicación amplia de los formatos usados en el proceso de diseño y propone una metodología basada en la aplicación comercial de diseño auxiliado por computadora llamada OrCAD/PLD.

CAPITULO 3 ASICS

3.1 Antecedentes

El diseño de Sistemas Digitales desde sus inicios se ha enfocado a la automatización buscando el menor costo en la implementación de estos Sistemas, uno de los factores principales para lograrlo es usar la menor cantidad de Circuitos Integrados "CI o chips" ya que independientemente del costo del circuito integrado, los costos más significativos son los costos de producción, control de calidad, la Ingeniería, el tamaño del diseño, el consumo de potencia y la facilidad para cambiar el diseño.

3.2 Introducción a los ASIC

Desde los finales de la década de los sesenta, los equipos electrónicos digitales se han construido utilizando Circuitos Integrados (CI o CHIPS) de función lógica fija, realizados en pequeña o mediana escala de integración (LSI, MSI).

Los circuitos integrados de función fija llamados "Off the Shell" (fuera del entorno) se diseñan para cumplir las necesidades de una gran variedad de aplicaciones. Por ejemplo los circuitos TTL (Transistor Transistor Logic) de la serie 7400 o la serie 4000 de los circuitos integrados CMOS y también los microprocesadores.

Para la implementación de aplicaciones muy complejas que requerirían una gran cantidad de circuitos de función fija es conveniente usar los circuitos a la medida que son llamados ASICS Application Specific Integrated Circuits.

Los avances en la tecnología VLSI (Very Large Scale Integration) "Integración a gran escala", trajeron consigo Circuitos Integrados (CI chips) que incluyen millones de transistores. Con el objetivo de ser competitivos, en el desarrollo de nuevos productos y mejorar los existentes incorporando las últimas técnicas VLSI, y aún más, diseñar circuitos integrados que sean especializados para sus propias aplicaciones. Estos chips diseñados para cumplir aplicaciones muy específicas, y por supuesto, son los llamados Circuitos Integrados de Aplicación Específica, o ASICs (Application Specific Integrated Circuits).

En la actualidad, los ASICs ofrecen muchas ventajas sobre los dispositivos estándar, aquí se mencionan algunas de ellas:

- Espacio reducido en el circuito impreso.
- Menor número de terminales en el circuito impreso.
- Bajo consumo de potencia.
- Menor Calentamiento.
- Bajos costos.
- Más tolerancia a radiaciones (esto es especial para misiones espaciales).
- Mejor facilidad en la verificación (Control de calidad).
- Mejor confiabilidad.
- Posibilidad de implementar seguridad contra copia.

Durante los 80's, se recurría a los ASICs por dos razones principales:

- El diseño ASIC se enfocaba en producir en un corto tiempo un diseño.
- Cumplir los requerimientos específicos del consumidor.

Existe un pequeño problema de precisión sobre el uso actual del término ASIC.

Los chips ASIC no son en general solamente de aplicación específica, sino que también son de aplicación específica según el consumidor.

Se puede decir que sólo los dispositivos de lógica programable por el consumidor son los verdaderos ASIC.

El término ASIC es usado por diferentes personas en diversos contextos con diferentes significados.

Es importante observar una comparación en cuanto a costos entre las diferentes tecnologías de CIs para tener una mayor claridad sobre su comercialización. Los costos de una producción ASIC pueden dividirse en costos fijos y costos variables.

Dentro de los costos fijos están los costos de ingeniería no recurrentes (NRE) y los costos de documentación. Los costos NRE son todos aquellos costos en que se incurre durante el diseño de un chip y que no son distribuibles en las diferentes interacciones que se puedan presentar, como los costos de ingeniería, costos de personal, costos de soporte y costos de fabricación de prototipos.

Los costos de documentación están asociados a todos los catálogos, manuales, ayudas didácticas, documentos técnicos, etc., que facilitan el desarrollo y mantenimiento de un proyecto.

Los costos variables están asociados a los costos de producción, como por ejemplo los costos de fabricación en serie, costos de programación y costos de inventario.

Los componentes estándar y los componentes discretos presentan continuamente una reducción en sus ventas, siendo claramente desplazados por soluciones basadas en arquitecturas con microprocesadores. Los ASIC conservan una porción significativa del mercado.

3.3 Opciones de diseño ASIC

Para la implementación de ASICs deben conocerse muy bien las diferentes tecnologías y sus correspondiente metodología de diseño para poder tomar la mejor opción para un desarrollo.

A continuación se realiza una descripción de las diferentes alternativas ASIC.

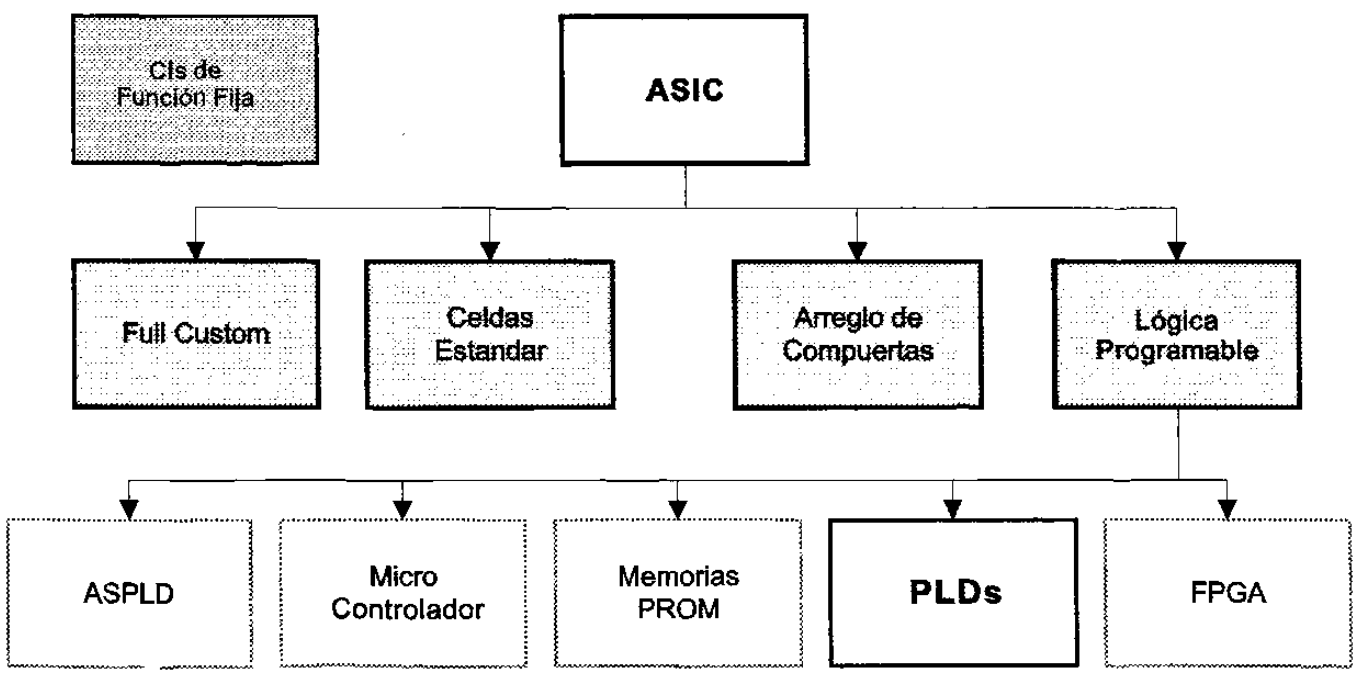


Fig. 3.1 Clasificación de los Circuitos Integrados

3.3.1 Full-Custom (Diseño Sobremedida)

En este clásico diseño, cada función lógica o transistor es manualmente diseñado y optimizado.

Esto tiene como resultado un chip más compacto con la mayor velocidad posible y el menor consumo de potencia. Sin embargo, la inversión inicial en el costo de ingeniería no recurrente (NRE, Non-Recurring Engineering) es mucho mayor comparada con cualquier otro estilo de diseño.

El diseñador debe manipular las formas geométricas individuales que representan los aspectos de cada transistor en el chip.

Un diseño relativamente simple de 300 compuertas puede requerir el manejo de 300,000 rectángulos por chip.

El conjunto de herramientas que se utilizan en la realización de este tipo de diseño se agrupa bajo la denominación de compiladores de silicio. Estos tienen una plataforma básica en la composición del circuito integrado, la cual se configura para soportar la tecnología de fabricación seleccionada.

Además de las funciones lógicas básicas, la mayoría de compiladores de silicio permiten generar estructuras complejas en una forma de integración muy compacta. Por ejemplo, pueden generarse RAMs, ROMs y funciones más complejas como multiplicadores, procesadores, etc., a partir de sus parámetros (longitud de palabra, tamaño en bytes, etc.). Igualmente es posible incluir funciones analógicas para aumentar la integración del dispositivo y cubrir un mayor rango de aplicaciones.

En esta tecnología se hace uso óptimo del silicio, lo que se traduce en reducción de costos.

Pero a pesar de esto, los costos y riesgos de desarrollo son bastante altos, y los volúmenes de producción deben ser significativamente bastante grandes.

Por lo anterior, este tipo de metodología es usada para los ASIC actuales sólo cuando se tienen prototipos semicustom muy bien optimizados, y demandas de producción relativamente altas.

3.3.2 Celdas Estándar (Standard Cells).

En esta metodología, bloques lógicos y funcionales predefinidos están disponibles en una biblioteca de celdas. Típicamente las bibliotecas comienzan con un conjunto de compuertas básicas AND, OR, NAND, NOR, XOR, inversores, Flip-Flops, registros y similares.

Generalmente las bibliotecas contienen también funciones más complejas como sumadores, registros de corrimiento, decodificadores multiplexores, ALUs y memorias (RAM, ROM, FIFOs, etc.).

En algunos casos las bibliotecas de celdas estándar incluyen funciones como: multiplicadores, divisores, microcontroladores, microprocesadores y soportes para microprocesadores (puertos, controladores de DMA, contadores, reloj de tiempo real, etc.).

Las celdas estándar son solamente máscaras de elementos predefinidas para cada etapa del proceso de fabricación del dispositivo.

Los diseños son creados usando herramientas de captura esquemática o vía síntesis de algún lenguaje de descripción de hardware.

Esto hace esta metodología mucho más sencilla, pues ya no debe realizarse un diseño manual y geométrico de los transistores y compuertas, sino que simplemente se dibuja a través de símbolos esquemáticos o se describe en algún lenguaje de descripción de hardware el circuito funcional (y no el circuito físico como en full-custom).

3.3.3 Arreglo de Compuertas (Gate Arrays).

Las metodologías full-custom y de celdas estándar requieren una fabricación del chip usando un conjunto completo de máscaras únicas que definen el procesamiento del semiconductor. Así, los costos de ingeniería no recurrente en el conjunto de máscaras y el tiempo del diseño es un poco alto.

Como alternativa, un chip puede ser creado usando un patrón de interconexión usual en un arreglo de compuertas lógicas sin conexiones.

Las obleas de chips que contienen arreglos de compuertas libres pueden ser prefabricados hasta el punto antes de la etapa final de metalización, la cual crea la configuración lógica.

Comparando con celdas estándar o full-custom, los costos y el tiempo de diseño se reducen debido a que sólo debe aplicarse la última capa de interconexión y contacto.

Los costos de las obleas para arreglos de compuertas son menores debido a que se pueden producir en gran volumen, ya que muchos diseños diferentes pueden ser creados a partir de la misma oblea base.

Similarmente, los costos de las pruebas se reducen debido al uso de estándares en la ubicación de terminales y que pueden ser usados en múltiples diseños.

Usualmente, el diseño con estos dispositivos se lleva a cabo mediante captura esquemática a partir de compuertas lógicas básicas o Macrocelda de mayor nivel con funcionamiento similar a los dispositivos de lógica estándar.

3.4 Lógica Programable. FPIC

Un dispositivo de lógica programable en campo es un Circuito Integrado cuya estructura lógica final es directamente configurada por el usuario, sin necesidad de llevar ningún proceso de fabricación.

Los FPICs (Field Programmable Integrated Circuits) son CI programados por el usuario mediante programadores comerciales. El término FPIC también incluye a los CI no destinados a las operaciones Lógicas.

Eliminando los ciclos de fabricación, tanto el tiempo de comercialización como el riesgo financiero son reducidos notoriamente. Como FPICs podemos nombrar a:

- PLDs
- ASPL
- Microcontroladores
- Memorias
- FPGA

3.4.1 PLDs

Las clases más sobresalientes de circuitos integrados programables en campo son los Dispositivos Lógicos Programables PLDs (Programmable Logic Device) y los GALs (Generic Array Logic) los PLDs, y GALs han emergido como las soluciones ASIC de mejor relación costo-beneficio porque proporcionan la creación de prototipos a bajo costo con producción casi instantánea.

Este tipo de dispositivos consiste de un arreglo de elementos lógicos sin ligar, cuya estructura lógica o de interconexión puede ser personalizada en campo de acuerdo a las especificaciones del usuario.

3.4.2 ASPLD

Los ASPLDs (Application Specific Programmable Logic Devices) son PLDs diseñados para realizar funciones específicas como: Decodificadores de alta velocidad, secuenciadores, periféricos programables de microprocesadores, etc.

Partes del ASPLD son programables permitiendo la adaptación del circuito a una aplicación determinada, pero manteniendo la función básica, así por ejemplo, un decodificador lo personaliza el usuario, pero sigue siendo un decodificador.

3.4.3 Microcontroladores

Los Microcontroladores son microprocesadores que además del listado de instrucciones, tienen integrados puertos en los que se pueden programar como entradas o salidas y contienen internamente memoria RAM y ROM.

3.4.4 Memorias

Las Memorias PROM, EPROM y EEPROM son consideradas PLDs, y en ocasiones se excluyen de esta denominación debido a que su contenido se define usando elementos de desarrollo propio de microprocesadores, tales como ensambladores, emuladores y lenguajes de programación de alto nivel, también se usan para realizar una función lógica almacenando una tabla de verdad y con Flip Flops a la salida se puede implementar un Sistema Secuencial.

3.4.5 FPGA

FPGA (Field Programmable Gate Array): Consiste de un arreglo de bloques lógicos, rodeado de bloques de entrada/salida programables, y conectados a través de interconexiones programables

CAPITULO 4 PLDs

4.1 Clases de PLDs:

Los PLDs están orientados a reemplazar circuitos TTL para reducir el espacio de la tarjeta, realizar rápidos prototipos de sistemas digitales, explotando la flexibilidad y confiabilidad de los elementos actualmente disponibles.

Los Dispositivos Lógicos Programables reducen el tiempo de desarrollo y comercialización del circuito gracias a los programas de diseño (CAD), los cuales permiten describir el funcionamiento del circuito a través de ecuaciones, tablas de verdad, diagramas de estado y entrada esquemática; pueden optimizar el diseño y simularlo, y una vez programados físicamente realizar una prueba del dispositivo con los mismos vectores de simulación, con lo que la posibilidad de falla en la tarjeta es prácticamente nula.

4.1.1 PAL (Programmable Array Logic):

- PAL (And programable y Or fija).

Es un PLD en el que se pueden programar las uniones de la matriz de compuertas AND, pero siguen siendo fijas las de la matriz de compuertas OR. Son los más utilizados y en los cuales se centrará nuestro interés. Es un Sistema Combinacional incompleto, ya que teniendo n entradas, dispone de menos de 2 términos producto.

4.1.2 PROM (Programmable read only memory):

- PROM (Or programable, And fija)

Es el dispositivo donde permanecen fijas las uniones en la matriz de compuertas AND, siendo las de compuertas OR programables. Es un sistema combinacional completo que permite realizar cualquier función lógica con las n variables de entrada, ya que dispone de 2 términos producto. Están muy bien adaptadas para aplicaciones como: tablas, generadores de caracteres, convertidores de código, etc.

4.1.3 FPAL (Fiel Programmable Logic Array):

- FPLA (And y Or programables)

Son los que se pueden programar uniones tanto en la matriz de compuertas AND como en la matriz de compuertas OR. Son más flexibles pero más lentos y grandes. Se utilizan para construir máquinas de estados. Para otras aplicaciones las PAL resultan más efectivas. Es un sistema combinacional incompleto.

4.1.4 GAL (Generic logic Array):

Es una marca registrada de Lattice Semiconductor que tienen la ventaja de programarlas, y si se desea, borrarlas eléctricamente para volver a programar,

Mantienen una compatibilidad con la mayoría de los PAL en la que pueden ser un reemplazo porque su configuración terminal a terminal es de entradas y salidas es idéntica, sólo faltaría verificar la programación.

La arquitectura del GAL es muy compleja como resultado de la diversidad de PALs a los que puede reemplazar, cada terminal de entrada-salida dispone de una Macrocelda, como se muestra en el diagrama de bloques, en donde los fusibles de las Macrocelda permiten configurar al GAL de 3 siguientes tres modos:

- **SIMPLE**

El modo simple proporciona ocho compuertas AND para cada salida que no tienen la condición tri-state, este modo está pensado para emular a los PALs sencillos como el PAL14H4, ideal en la implementación de sistemas combinacionales.

- **COMPLEJO**

El modo Complejo se usa para lógica combinacional como la que realiza la PAL16L8; siete compuertas AND por salida y una compuerta AND para habilitación del tri-state, las terminales reloj (clock) y enable están disponibles como entradas normales.

- **REGISTRO**

El modo Registro se utiliza para reemplazar a las PAL16R8, PAL16R6, y PAL16R4 que poseen registros en donde su Reloj (clock) es activado por transición positiva, común a todos los registros y cuyas salidas están controladas por medio de manejador (driver) tri-state activado con otra terminal llamada Enable (habilitador) común a todas las salidas de los registros, Ideal en el diseño de Sistemas Secuenciales.

El GAL16V8 es un dispositivo de 20-terminales en el que se cuenta con ocho entradas dedicadas y ocho que pueden ser usadas como entrada-salida.

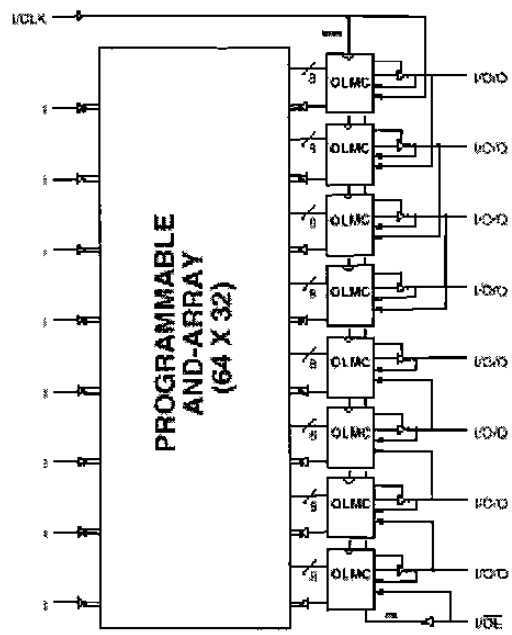


Fig 4.1 Diagrama de Bloques del GAL 16V8

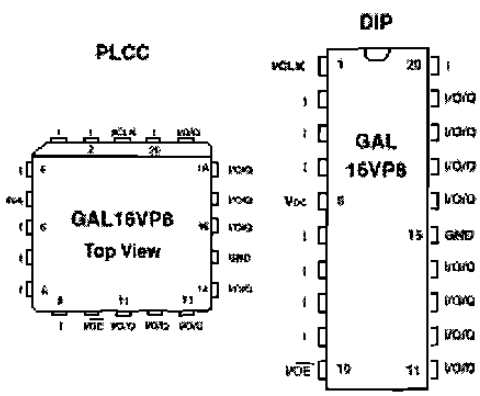


Fig 4.2 Distribución de las terminales del GAL 16V8

El GAL20V8 de 24-terminales contiene doce entradas dedicadas y ocho que pueden ser usados como entrada-salida.

El GAL16V8 y el GAL20V8 proporcionan la más alta velocidad de los PLDs actualmente existente 3.5 y 5.0 ns respectivamente.

4.2 Seguridad

Los PLDs tienen fusibles de seguridad que impiden la lectura de los dispositivos programados, protegiendo los diseños frente a copias.

Permiten realizar modificaciones posteriores del diseño y en ocasiones hacen posible la reutilización de circuitos impresos con algunas fallas, mediante una reasignación de las terminales de los PLDs.

Algunos PLDs ofrecen las dos formas de retroalimentación a la vez, es decir, que disponen de retroalimentación desde la terminal de salida y de retroalimentación desde la salida del registro. Si en estos dispositivos se selecciona una salida provista de registro y al mismo tiempo se deja en tercer estado al Driver de salida, la terminal asociada a la Macrocelda puede actuar como entrada. De esta forma se puede disponer simultáneamente de un registro interior enterrado (buried) o sin acceso al exterior y de una entrada.

Hay muchas aplicaciones (por ejemplo, registros de desplazamiento) que utilizan registros enterrados y también hay PLDs que incluyen registros interiores sin posibilidad de acceder a una con objeto de atender a este tipo de aplicaciones.

4.3 Fusibles de Seguridad

Con los PLDs, al igual que sucede con las memorias PROM o EPROM, el equipo de programación puede utilizarse para leer el contenido de los dispositivos programados. Sin embargo, prácticamente todos los PLDs disponen de un fusible de seguridad, que una vez fundido impide la lectura del PLD programado proporcionando gran seguridad frente a la piratería.

En los PLDs bipolares, la programación destruye ciertos fusibles, Los PLDs CMOS borrables eléctricamente o mediante luz ultravioleta son mucho más seguros, y el fusible de seguridad ofrece suficientes garantías de protección contra el acceso al contenido.

4.4 Ventajas del uso de PALs.

- Reducción de retardos de propagación de las compuertas lógicas.
- Reducción del número de puntos de soldadura y de cables o pistas. Con ello se logra aumentar la confiabilidad.
- Facilidad para cambiar el diseño.
- Menor número de circuitos integrados, disminución del consumo de energía y de las fuentes posibles de ruido y aumento de la confiabilidad del circuito.
- Menor superficie ocupada en la placa.
- Protección del diseño (fusible de seguridad).
- Su tiempo de propagación ronda los 5 a 10 ns, equivalente al de las compuertas que puede reemplazar; pero con una sola PAL se puede reemplazar varios niveles o retardos de compuertas.
- Facilidad de diseño y montaje.
- Reducción de la variedad de circuitos integrados usados en el circuito, disminuyendo su costo de producción seriada.
- Diseño a la medida con bajo costo.

4.5 Limitaciones de los PALs.

- Las PALxx son OTP, es decir, sólo se pueden programar una vez.
- Dado que el número de minitérminos es limitado, en algunas aplicaciones se necesitarían varios PALs en la implementación y quizás resulta más simple y menos costoso implementarla con compuertas.

CAPITULO 5 Herramientas Para El Diseño Con Pals

5.1 Introducción

Las herramientas de diseño se basan en aplicaciones de software y hardware usados en la Computadora Personal y se clasifican en tres grupos:

- Archivos con la información del diseño
- Compiladores
- Programadores

5.2 Archivos con la Información del Diseño

Los archivos se pueden obtener de diversas fuentes que van desde usar un editor de textos siguiendo la sintaxis que es requerida en los compiladores, hasta lenguajes como VHDL o a partir de un software del tipo CAD que automáticamente generan los archivos necesarios que posteriormente serán compilados.

5.2.1 Símbolos y Operadores

En el desarrollo de un archivo de información del diseño los símbolos más frecuentemente usados son:

´	Not
!	Not
&	And
#	Or
##	ExOr OR-Exclusivo
&´	Nand
#´	Nor
##´	ExNor o And Coincidence

A continuación se muestra un archivo cuyo propósito es implementar una compuerta NAND de tres entradas en un PAL 16R4, realizado en un editor de textos con el nombre de NAND3.PLD.

```
include p16r4;
!pin19 = pin2 & pin3 & pin4;
pin19.oe = 1;
test_vectors {
   pin2 pin3 pin4 !pin19;
    0    0    0     L   ;
    0    0    1     L   ;
    0    1    0     L   ;
    0    1    1     L   ;
    1    0    0     L   ;
    1    0    1     L   ;
    1    1    0     L   ;
    1    1    1     H   ;
}
```

5.3 Lenguajes de Descripción de Hardware

- HDL Hardware Description Languages

Los lenguajes de programación se han desarrollado desde hace casi 50 años, pero desde sus inicios se comenzó con el desarrollo de lenguajes de programación de software. Algunos de estos fueron diseñados para cálculos matemáticos, otros se basaron en lógica, otros son de tipo funcional basándose en las matemáticas, hasta llegar a los lenguajes orientados a objetos. Cada lenguaje se forma según las aplicaciones que soportan.

Existen otros tipos de lenguajes de programación diferente a los de software y son los Lenguajes de Descripción de Hardware (HDLs), los cuales son una forma de notación para describir la estructura y el comportamiento de un circuito eléctrico.

Se busca que los HDLs sean estándares, para que no dependan de una familia en especial, ni de una compañía, con lo cual se facilite su uso.

Los HDLs son utilizados para describir el hardware desde diferentes niveles de abstracción. Al diseñar en alto nivel se describe el comportamiento del circuito, y en niveles bajos se describe el circuito mediante sus componentes internos e interconexiones, por ejemplo mediante la conformación de compuertas y sus uniones para crear una función lógica que determine una aplicación.

Los HDLs permiten utilizar módulos ya creados en nuevos diseños, por ejemplo para realizar conexiones en cascada, lo que facilita y optimiza el diseño.

Una parte importante al utilizar HDLs es la síntesis. En esta etapa se traduce el programa diseñado en una descripción mediante ecuaciones booleanas (compuertas), para que una interfaz pueda configurar el hardware y este cumpla con el funcionamiento del programa, o se pueda construir un circuito integrado.

Algunos de los HDLs más conocidos, y que son estándares, son:

- VHDL
- ABEL HDL
- Verilog HDL

5.3.1 Introducción al VHDL

VHDL, significa VHSIC (Very High Speed Integrated Circuit) Hardware Descripción Language. VHDL es un lenguaje de descripción y modelado diseñado para describir (en una forma que los humanos y las máquinas puedan leer y entender) la funcionalidad y la organización de Sistemas Digitales, hardware, circuitos impresos y componentes.

VHDL fue desarrollado como un lenguaje de modelado y simulación lógica dirigida por eventos de sistemas digitales, y actualmente se utiliza también en la síntesis automática de circuitos.

Se trata de un lenguaje con una sintaxis amplia y flexible que permite el modelado estructural, en flujo de datos y de comportamiento del hardware. VHDL permite el modelado preciso, en distintos estilos, del comportamiento de un sistema digital conocido y el desarrollo de modelos de simulación.

5.3.2 Historia de VHDL

El VHDL se comenzó a desarrollar en 1980 por el departamento de defensa de los Estados Unidos. Este nació debido a las necesidades presentadas de estandarizar un lenguaje con amplias capacidades, que funcionarán igual en cualquier simulador, y que fuera independiente de la tecnología, de una empresa y de la metodología de diseño.

Antes de este estándar, el diseño resultaba costoso y difícil, ya que se encontraban gran cantidad de simuladores, lenguajes y diversos tipos de herramientas, dependiendo de las tecnologías, de las empresas, y cualquier cambio en el diseño, llevaba a grandes inversiones, y retardos de tiempo.

Así se decidió crear un estándar para satisfacer estas necesidades y se comenzó la creación de VHDL.

En el proceso de creación del estándar, participaron varias industrias. La base de VHDL fue desarrollada por Intermetrics, IBM y Texas Instruments quienes ganaron el contrato para desarrollarlo. Este desarrollo se presentó entre los años 1983 y 1985. Fue publicado como: VHDL versión 7.2.

En 1986 el Departamento de Defensa transfiere los derechos al IEEE, para que se aceptara fácilmente por la industria. Con algunas modificaciones fue publicado en 1987 por la IEEE como: IEEE Standard 1076-1987 En 1988 es aprobado por la ANSI.

En 1993 y en 1994 fue revisada y se publicó la ultima versión como: IEEE Standard 1076-1993

Además el Departamento de Defensa exige una descripción en VHDL para cada ASIC que es enviado a éste. Una de las aplicaciones iniciales más complejas utilizando VHDL desarrolladas por el Gobierno de los Estados Unidos fue el diseño del avión de combate F22. En éste, varios contratistas realizaron subsistemas. Mediante el uso del estándar, los subsistemas electrónicos fueron de fácil acople debido a que se utilizó VHDL en la descripción de todos los dispositivos electrónicos del proyecto.

5.3.3 Características del VHDL

Las principales características del lenguaje VHDL son:

El VHDL puede describir el funcionamiento y la estructura de sistemas electrónicos, pero está especialmente diseñado para circuitos electrónicos digitales.

El VHDL se creó buscando gran flexibilidad, por lo que permite manejo de diseños análogos en casos limitados. El VHDL no está contemplado para el diseño de circuitos analógicos pero se puede modelar un circuito analógico, lo que suele llevar bastante trabajo. Para mejores resultados en este campo se creó el VHDL análogo, que aún se viene desarrollando.

Los principales beneficios que posee VHDL son:

- Permite diversas alternativas de diseño.
- Se puede simular el diseño al nivel de programa.
- El funcionamiento no depende de la implementación.
- Puede manejar gran complejidad.
- Las herramientas realizan automáticamente la síntesis.
- Se disminuyen los tiempos en que el producto se comercializa.
- Describe una gran variedad de hardware digital.
- Provee un lenguaje muy flexible.
- Permite utilizar funciones ya creadas (bloques) en varios programas.
- Es un estándar a nivel mundial. La mayoría de tecnologías de diseño, y empresas soportan VHDL
- El lenguaje es independiente de la tecnología y de las herramientas, lo que lleva a grandes ahorros económicos si son necesarios cambios. Esto permite proteger grandes inversiones.

5.3.4 Aplicaciones comerciales

Podemos encontrar en algunas aplicaciones que a partir de una tabla de verdad o de un diagrama de transición se pueden generar los códigos de los archivos VHDL, por mencionar algunos, StateCAD, Orcad/PLD etc.

A continuación se muestran el Diagrama de Transición y el correspondiente archivo de diseño llamado ALARMA.ABL en formato ABEL-HDL generado en el programa StateCAD, consiste en una Llave electrónica, que cuenta con cuatro botones de entrada llamados A,B,C,D y dos salidas una para abrir la puerta y la otra para activar una alarma.

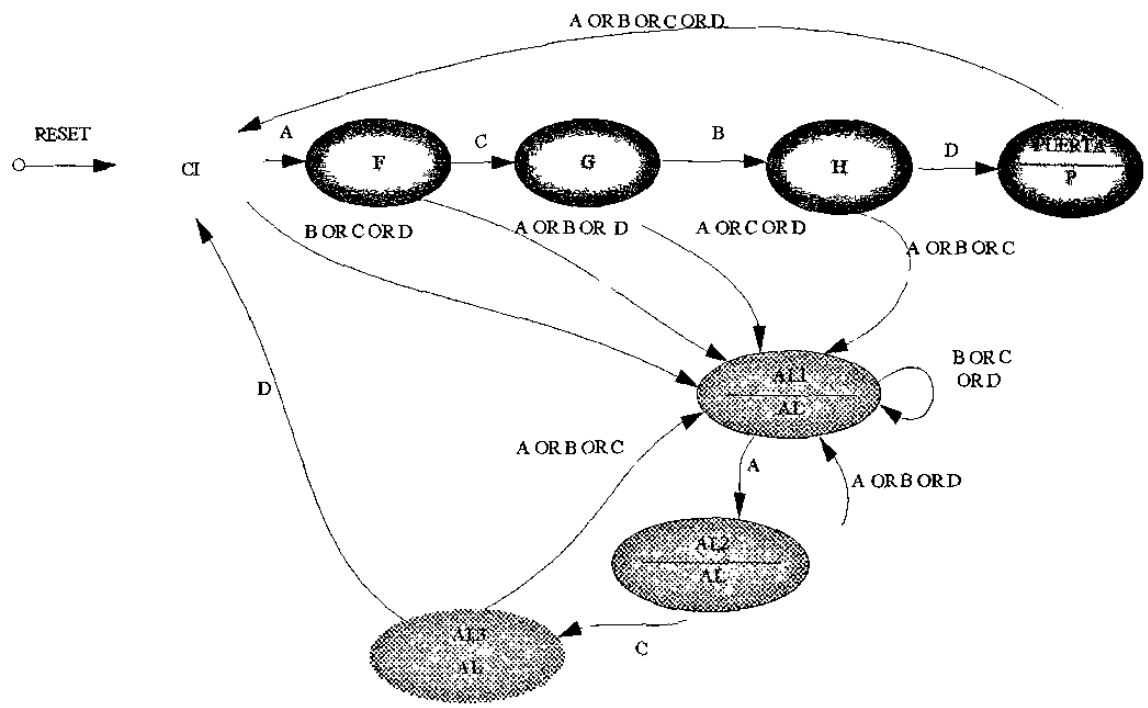


Fig. 5.1 Diagrama de Transición de la Llave Electrónica.

Archivo 5.1 **ALARMA.ABL** en formato ABEL-HDL.

```
MODULE ALARMA


DECLARATIONS
"CLOCK NAME
        CLK PIN 1;


"Variables de Entrada
        A PIN 2;
        B PIN 3;
        C PIN 4;
        D PIN 5;
        RESET PIN 6;


"Variables de Salida
        AL PIN 12 ISTYPE 'COM';
        P PIN 13 ISTYPE 'COM';


"STATE VARIABLES
        SV0  PIN 14 ISTYPE 'REG';
        SV1  PIN 15 ISTYPE 'REG';
        SV2  PIN 16 ISTYPE 'REG';


"STATE REGISTER ASSIGNMENT
DECLARATIONS
        SREG=[ SV0,SV1,SV2];


EQUATIONS
        SREG.CLK=CLK;


DECLARATIONS
        AL1=[0, 0, 0];
        AL2=[0, 0, 1];
        AL3=[0, 1, 0];
        CI=[0, 1, 1];
        F=[1, 0, 0];
        G=[1, 0, 1];
        H=[1, 1, 0];
        PUERTA=[1, 1, 1];


STATE_DIAGRAM SREG;

STATE AL1:
        P=0;
        AL=1;
        IF ( RESET ) THEN CI;
        ELSE
                IF ( !B & !C & !D & !A ) THEN
AL1;
                IF ( B  #  C  #  D ) THEN AL1;
                IF ( A ) THEN AL2;


STATE AL2:
        P=0;
        AL=1;
        IF ( RESET ) THEN CI;
        ELSE
                IF ( !C & !A & !B & !D ) THEN
AL2;
                IF ( C ) THEN AL3;
                IF ( A  #  B  #  D ) THEN AL1;
```

```
STATE AL3:
        P=0;
        AL=1;
        IF ( RESET ) THEN CI;
        ELSE
                IF ( !D & !A & !B & !C ) THEN
AL3;
                IF ( D ) THEN CI;
                IF ( A  #  B  #  C ) THEN AL1;


STATE CI:
        AL=0;
        P=0;
        IF ( RESET ) THEN CI;
        ELSE
                IF ( !A & !B & !C & !D ) THEN
CI;
                IF ( A ) THEN F;
                IF ( B  #  C  #  D ) THEN AL1;


STATE F:
        AL=0;
        P=0;
        IF ( RESET ) THEN CI;
        ELSE
                IF ( !C & !A & !B & !D ) THEN
F;
                IF ( C ) THEN G;
                IF ( A  #  B  #  D ) THEN AL1;
```

```
STATE G:
        AL=0;
        P=0;
        IF ( RESET ) THEN CI;
        ELSE
                IF ( !A & !C & !D & !B ) THEN
G;
                IF ( A  #  C  #  D ) THEN AL1;
                IF ( B ) THEN H;


STATE H:
        AL=0;
        P=0;
        IF ( RESET ) THEN CI;
        ELSE
                IF ( !D & !A & !B & !C ) THEN
H;
                IF ( D ) THEN PUERTA;
                IF ( A  #  B  #  C ) THEN AL1;

STATE PUERTA:
        AL=0;
        P=1;
        IF ( RESET ) THEN CI;
        ELSE
                IF ( A  #  B  #  C  #  D ) THEN
CI;
                IF ( !A & !B & !C & !D ) THEN
PUERTA;

END AL
```

5.4 Compiladores

La función de los compiladores es a partir del archivo de descripción de lenguaje HDL generar el archivo en formato JEDEC que contiene el mapa de fusibles del PLD, en la siguiente tabla se muestra el archivo JEDEC correspondiente al diseño de la Llave Electrónica:

Archivo 5.2 **ALARMA.JED** en formato JEDEC.

```
JEDEC FILE FOR: P16V8R V
QP20* QF2194* QV0* F0*  X0*
NOTE TABLE OF PIN NAMES AND NUMBERS*
NOTE PINS CLK:1 A:2 B:3 C:4 D:5 AL:12 P:13 SV0:14 SV1:15 SV2:16*
L0768 11111111111111011101110111111111*
L0800 11111110111111101110110111111111*
L0832 11111111111101101101111111111111*
L0864 01111111111111101110111011111111*
L0896 10111011101110011111111111111111*
L1024 11111111111111011101110111111111*
L1056 11110111111111011111110111111111*
L1088 11111110111111011111011011111111*
L1120 11111111111101101101111111111111*
L1152 10111011101111101101111111111111*
L1184 10111011101110111101111111111111*
L1280 10111011101110111111110111111111*
L1312 11110111111111011110110111111111*
L1344 11111110111111011101101111111111*
L1376 11110111011111111110110111111111*
L1408 11111111111101101101110111111111*
L1440 01111111111111011101111011111111*
L1536 11111111111111111111111111111111*
L1568 11111111111111011101110111111111*
L1792 11111111111111011101110111111111*
L1824 11111011101110011101111111111111*
L1856 10111011101101101101111111111111*
L1888 10111011111110101110110111111111*
L1920 10111111101110011111110111111111*
L1952 10111011101110111111110111111111*
L2048 00011110*
L2120 00000010*
L2128 11111111111111111111111111111111111111111111111111111111111111111111*
L2193 1*
C5D30*
0 E D2
```

5.4.1 JEDEC

JEDEC son las siglas de Joint Electron Device Engineering Council. Es un consejo que crea, aprueba, dirige y promueve estándares de la industria de dispositivos electrónicos de estado sólido. La dirección de pagina de Internet del consejo es http://www.eia.org/jedec/.

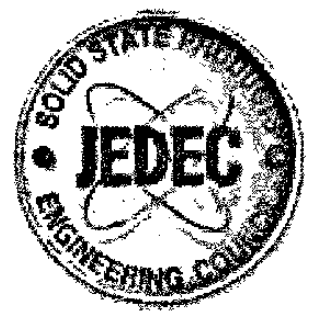


Fig. 6.2 Logotipo JEDEC

En lógica programable se refiere al formato de un archivo de software que contiene información acerca de la configuración de un dispositivo; esta información es enviada hacia el programador de dispositivos.

El formato de este archivo es un estándar aprobado por el JEDEC.

Este formato de archivo JEDEC permite transferir los datos de programación para los PLDs, independientemente de cualquiera que fuera el fabricante del dispositivo o el software de programación.

También se normalizaron las asignaciones de terminales, para diversos paquetes de CIs, a fin de hacer los programadores universales menos complicados (un programador universal es un equipo que permite realizar la configuración de diversos dispositivos PIC).

En consecuencia, los aparatos para programación pueden configurar numerosos tipos de PLDs.

El software de desarrollo que permite al diseñador especificar una configuración para un PLD, solo necesita producir un archivo de salida que cumpla con la norma JEDEC. Luego, este archivo JEDEC se pude cargar en cualquier programador de PLD compatible con JEDEC, que tenga capacidad para programar el tipo deseado de PLD.

El formato JEDEC no es el único formato de archivo de intercambio de información para PICs.

Existen varios formatos más, los cuales pueden o no ser aprobados por las comisiones de estandarización.

En la actualidad casi todo el software de desarrollo de PLDs genera los archivos base para enviar al programador de dispositivos en forma automática.

El formato de archivo JEDEC tiene en general la siguiente información:

- Referencia del dispositivo a utilizar.
- Número de terminales del dispositivo.
- Número total de elementos básicos de programación.
- Número de vectores de prueba que contiene el dispositivo.
- Estado por defecto para los elementos de programación no especificados.
- Asignación de terminales a señales.
- Información de la configuración de los elementos de programación (unos y ceros que representan si una intersección va a estar cerrada o abierta). Esta información va precedida de la letra L.
- Información sobre los vectores de prueba (precedidos por la letra V).

A continuación mencionaremos algunos compiladores comerciales.

- PROLOGIC
- SYNARIO
- FMPL – FastMap PLUS, Programmable Logic, para PC.
- OrCAD/PLD
- CUPL Universal Compiler for Programmable Logic.

5.4.2 Descripción del Compilador CUPL

CUPL es una aplicación de la computadora software que permite compilar diseños lógicos, y posteriormente programar elementos ó dispositivos lógicos programables, PLDs. El paquete CUPL consta de los siguientes programas :

CUPL : Universal Compiler for Programmable Logic.,**CSIM** : CUPL Simulator , **CBLD** : CUPL Build., **PTOC** : PALASM to CUPL Translator.

CUPL permite escribir y compilar las descripciones lógicas que se quieren asignar a los PLDs ó elementos lógicos programables.

CSIM permite simular el diseño después de compilado. Debe crearse un archivo con el funcionamiento esperado del PLD para algunos valores de entrada. CSIM compara los valores esperados con los calculados.

CBLD permite utilizar las bibliotecas que contienen descripciones de los PLDs que se encuentran en la corriente versión de CUPL.

PTOC convierte diseños PALASM a formato CUPL.

En el procedimiento, primero se crea un archivo fuente con la descripción lógica del diseño que se quiere asignar al PLD con lenguaje CUPL. El archivo fuente se crea en un editor de texto estándar, como el editor del DOS.

Posteriormente se compila el archivo generado en CUPL y crea el archivo de mapa de fusibles para llevar el elemento al programador del PLD. Durante la operación de CUPL se puede especificar la creación de un archivo absoluto que es utilizado después por CSIM.

Opcionalmente, puede crearse un archivo de prueba, especificando valores de entrada y salida, para verificar el diseño. Corriendo CSIM se comparan los valores esperados en el archivo de prueba con los valores actuales del archivo absoluto. Cuando la simulación se completa sin obtener errores se puede especificar que los vectores de prueba verificados sean adicionados al archivo JEDEC que se llevará al programador.

5.5 Programadores

Los programadores tienen la función de que a partir del archivo JEDEC y auxiliados con la Computadora Personal grabar el mapa de fusibles en el Circuito Integrado, algunos programadores son universales que pueden programar diversos dispositivos tales como:

- Memorias:

  PROM: N/CMOS E(E)PROM y FLASH Memory Bipolar PROM:

- Dispositivos Lógicos Programables:

  PAL,EPLD, EEPLD, EPM, GAL, MACH, MAX, PEEL, FPGA, IFL, FPL

- Microcontroladores:

  Microchip, Motorola, NEC, NSC, Signetics, WSI.

La mayoría de los dispositivos programadores se conectan a una computadora personal mediante una tarjeta interface o mediante los puertos serie o paralelo, así el archivo de diseño puede ser Editado, Ensamblado, Cargado al programador y entonces el dispositivo es programado, todo en un mismo lugar. A continuación listaremos algunos programadores:

- Dataman-48
- PALASM (AMD). A+PLUS,
- MAX+PLUS,SAM+PLUS(ALTERA).
- PLAN (National Semiconductor).
- AMAZE(Phillips-Signetics).
- CUPL (Logical Devices).
- LOG/IC (ISDATA).
- SUPERPRO-III Universial Programmer de XELTEC

5.5.1 Características del programador SUPERPRO-III

- Soporta más de 3,500 dispositivos: PROM: N/CMOS E(E)PROM y FLASH Memory Bipolar PROM: Signetics, TI, NSC, AMD/MMI ,PLD: PAL, EPLD, EEPLD, EPM, GAL, MACH, MAX, PEEL, FPGA, IFL, FPL, Microcontroller: Atmel, Hitachi, Intel, Microchip, Motorola, NEC, NSC, Signetics, WSI, Zilog, etc.
- Compatible con Computadoras Portatil, Personal o PS / 2 a través del puerto de la impresora.
- Acepta formatos de archivos: JEDEC, INTEL (Extendido)) HEX, Motorola S record
- Soporta a la mayoría de los compiladores en formato JEDEC incluyendo ABEL, CUPL, PALASM, SYNARIO, TANGO PLD, OrCAD PLD, PLD Designer e ISDATA
- Maneja palabras de 16 y 32 bit´s
- Adaptador de 48 terminales ZIF (Zero Insertion Force).

- Software SP3

Requiere de Procesador 386/486/pentium, Puerto Paralelo con las direcciones LPT1 (278H), LPT2 (378H), LPT3 (3BCH), 2 Mb de memoria RAM, 7 Mb de espacio en disco, Sistema Operativo MS-DOS o PC-DOS versión 2.1 o más actualizada, Monitor con capacidad de gráficas.

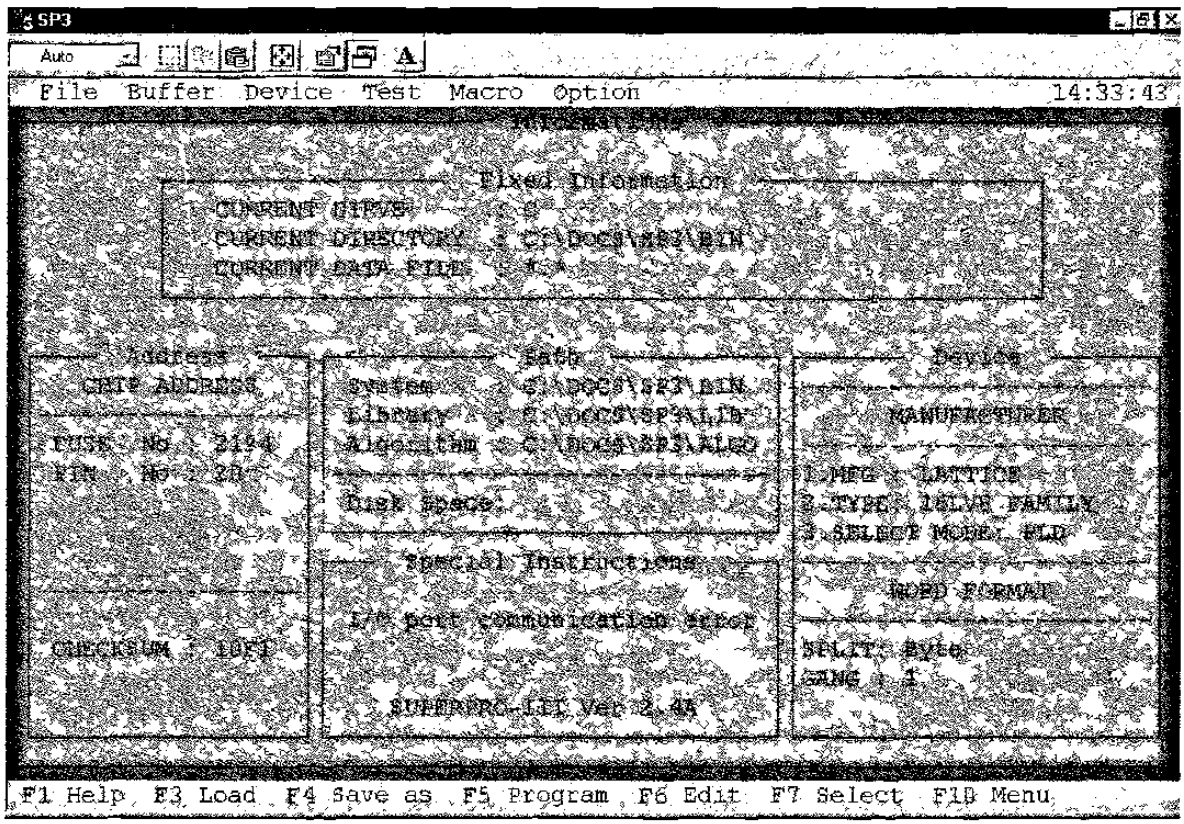


Fig. 5.2 Menú del programa SP3.EXE

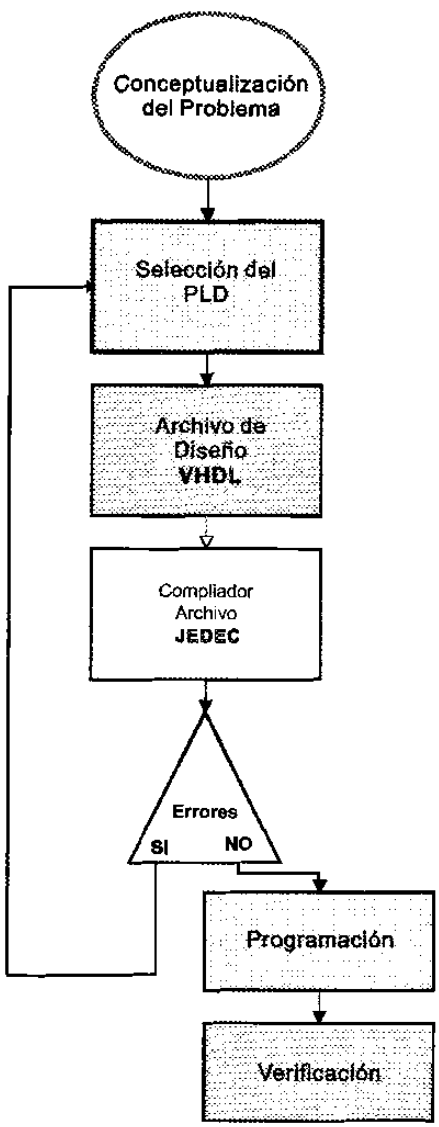
Características Físicas y Eléctricas

- Cable DB-25 (uno a uno) de 3Ft de largo
- Dimensiones 11.3" x 6.8" x 1.7"
- Peso 5 Lbs.
- Consumo de potencia (máximo): 28v 1.5 Amp.

CAPITULO 6 Metodología del Diseño

6.1 Metodología de Diseño

En el diseño de Sistemas Combinacionales y Secuenciales se puede usar la misma metodología que a continuación se propone la diferencia es solamente el punto de partida que puede ser de las Ecuaciones, tabla de verdad o Diagrama de Transición.



- Conceptualización del problema. La primer etapa en el proceso de diseño con PLDs consiste en especificar el diseño.

- La selección del dispositivo lógico adecuado y hacer mas practico y rentable nuestro diseño.

- Generar un archivo con la información del diseño el cual también incluye la simulación con vectores de prueba.

- Compilar y obtener el archivo de formato JEDEC.

- Programar el Dispositivo lógico Programable.

- Verificación y Prueba del diseño.

Fig. 6.1 Metodología del Diseño

6.2 Conceptualización de un Problema de Diseño

En este primer paso es en donde se definen las variables del proceso y su comportamiento así como la completa descripción del diseño y el número de entradas y salidas.

El primer paso en el proceso de diseño con PLDs, es el mismo que para cualquier Diseño Combinacional o Secuencial con pequeña y mediana escala de integración SSI/MSI.

Para el caso de Sistemas Combinacionales con la Tabla de Verdad o las ecuaciones son suficientes para describirlo completamente, mientras que en los Sistemas Secuenciales es necesario un Diagrama de Transición.

Una ventaja de los PLDs es que en esta etapa al diseñador sólo le interesa la función lógica requerida, sin preocuparse de la cantidad de Circuitos Integrados, el tiempo utilizado en reducir las ecuaciones, el diseño del circuito impreso y el costo en la implementación etc.

Con SSI o MSI varias limitaciones de los dispositivos lógicos deben tomarse en cuenta antes de que el diseño pueda ser iniciado.

Un diseñador necesita desarrollar una breve y completa descripción funcional basado sobre los requerimientos de diseño del sistema.

6.3 La selección del dispositivo lógico adecuado

En la selección del PLD hay que tomar en cuenta los siguientes factores:

- Número de Entradas.
- Número de Salidas.
- Número de líneas de Entrada/Salida.
- Polaridad de Salida.
- Salidas Combinacionales.
- Salidas EXOR.
- Salidas de Registro (R-S, J-K, D ó T; en general. D).
- Salidas TRI-STATE.
- Número de Minitérminos.
- Retroalimentación de las Salidas.
- Número de Macroceldas.

Si seleccionamos un dispositivo no adecuado no concluiríamos el proceso de compilación, y en este mismo proceso se genera un reporte de errores indicando las causas por las cuales no fue aceptado.

Para el caso de Salidas de Registro, el Flip-Flop más usual por utilizar una sola entrada de control. es el del tipo D.

6.4 Archivo de Diseño

En la creación de un Archivo de Diseño o Archivo de Descripción del Hardware HDL, se especifica la función deseada, ésta es típicamente representada en su forma de suma de productos, y puede ser directamente obtenida de las ecuaciones, Diagrama de Transición o tabla de verdad. Ocasionalmente Tablas de Estado pueden ser utilizadas.

El archivo generado comúnmente tiene la extensiones PLD, ABL o HDL, estos archivos contienen la siguiente información:

- Variables de Entrada.
- Variables de Salida,
- Las Ecuaciones Lógicas
- La asignación de valores a los Estados.

El archivo de diseño en algunas aplicaciones es posible simular el funcionamiento y comprobar si es el comportamiento esperado o detectar errores y corregirlos.

6.5 COMPILAR

Este proceso se lleva a cabo en la computadora personal con la ayuda de un programa de aplicación. Al compilar dependiendo del compilador se efectúan varios procesos como:

- Verificación de la sintaxis.
- Generación de un archivo indicando los errores.
- Determinar si la arquitectura del dispositivo seleccionado es correcta.
- Ensamble de las ecuaciones.
- Generación del archivo JEDEC especificando el estado de cada conexión sobre el dispositivo.

6.6 Programar

En la programación del dispositivo se deben de seguir los pasos que indica cada programador, por ejemplo en el SUPERPRO-III.

- Ejecute el programa ejecutable SPIII con el programador encendido, conectado al puerto de la computadora personal y el dispositivo fuera de su conexión.
- Seleccione la marca del fabricante del dispositivo.
- Seleccione el dispositivo.
- Inserte el dispositivo en el socket
- Cargue el archivo JEDEC
- En algunos casos como en el GAL de Latticce es necesario borrar la información grabada anteriormente en el dispositivo
- Grabe la nueva información en el dispositivo y asegúrese de que en este proceso de carga no tenga errores.

6.7 Verificación y Prueba

El dispositivo está listo y puede entonces ser insertado en la tarjeta o en un maletín de pruebas donde se recomienda elaborar una rutina para la simulación y verificar que efectúe completamente la función y cumpla con los requerimientos de diseño.

CAPITULO 7 Aplicaciones

7.1 EJEMPLO DE DISEÑO COMBINACIONAL

A continuación evaluaremos un ejemplo de Sistema Combinacional usando la metodología sugerida en el capítulo anterior, la aplicación Combinacional consiste en que se implementan en un solo dispositivo las operaciones básicas del álgebra booleana.

7.1.1 Conceptualización del problema,

Efectuar las operaciones And, Or, Exor Nand y Nor en donde se aplican diferentes variables a cada una de las entradas y salidas:

V0 = A And B

V1 = C Or D

V2 =E Exor F

V3 = G Nand H

V4 = I Nor J

7.1.2 Selección del PLD

Se requiere de un dispositivo con mínimo 10 entradas y cinco salidas, seleccionamos el GAL16V8 de Lattice Semiconductor por ser un dispositivo de 20 terminales en el que se cuenta con ocho entradas dedicadas y ocho terminales que pueden ser usadas como entrada-salida, este dispositivo cumple sobradamente con los requerimientos de este diseño.

7.1.3 Archivo de Diseño

A continuación mostramos el archivo de diseño generado con el programa de aplicación StateCAD y conociendo la distribución de terminales de entrada salida del GAL16V8 asignamos la terminal correspondiente a cada variable.

Variable	Terminal
A	2
B	3
C	4
D	5
E	6
F	7
G	8
H	9
I	11
J	12
V0	13
V1	14
V2	15
V3	16
V4	17

Tabla 7.1 Asignación de Terminales

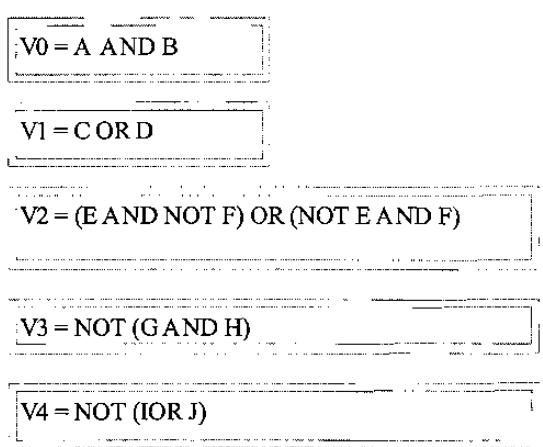


Fig. 7.1 Ecuaciones planteadas en StateCAD.

Archivo 7.1 **"OR.ABL"** generado en formato ABEL-HDL.

```
" C:\SC40E\OR.abl
" ABEL code created by Visual Software Solution's StateCAD Version 4.10
" Sun Apr 19 09:58:59 1998

" This Abel code was generated using:
" binary encoded state assignment with structured code format.
" Minimization is enabled,  implied else is enabled,
" and outputs are manually optimized.

MODULE OR

DECLARATIONS

"Input variables
        A PIN 2;
        B PIN 3;
        C PIN 4;
        D PIN 5;
        E PIN 6;
        F PIN 7;
        G PIN 8;
        H PIN 9;
        I PIN 11;
        J PIN 12;

"Logic variables
        V0 PIN 13 ISTYPE 'com';
        V1 PIN 14 ISTYPE 'com';
        V2 PIN 15 ISTYPE 'com';
        V3 PIN 16 ISTYPE 'com';
        V4 PIN 17 ISTYPE 'com';

"Logic Equations
EQUATIONS
V0 =  A & B ;
V1 =  C  #  D ;
V2 =  E & !F  #  !E & F ;
V3 =  !H  #  !G ;
V4 =  !I & !J ;
END OR
```

7.1.4 Compilar

Los pasos en la generación de los archivos de programación JEDEC y Chip Report necesarios en el compilador SYNARIO se detallan a continuación:

- Generar un nuevo proyecto, en nuestro caso se llamará OR.SYN, esto implica generar un subdirectorio con el nombre del proyecto.
- Seleccionar el Dispositivo Lógico Programable, en nuestro caso es el GAL16V8.
- Importar el archivo de diseño generado por el programa de aplicación en StateCAD llamado OR.ABL.
- Compilar el diseño hasta obtener los archivos con información del Circuito integrado (Chip Report) y el archivo en formato JEDEC llamado OR.JED.
- Archivo en formato JEDEC.

ARCHIVO 7.2**OR.JED.**

```
SYNARIO 2.10 DATA I/O CORP.
JEDEC FILE FOR: P16V8AS V
CREATED ON: SUN APR 19 15:41:50 1998

*
QP20* QF2194* QV0* F0* X0*
NOTE TABLE OF PIN NAMES AND NUMBERS*
NOTE PINS A:2 B:3 C:4 D:5 E:6 F:7 G:8 H:9 I:11 J:12 V0:13 V1:14 V2:15*
NOTE PINS V3:16 V4:17*
L0512 11111111111111111111111111011101*
L0768 11111111111111111111111101110111*
L1024 11111111111111110111011111111111*
L1056 11111111111111111011101111111111*
L1280 11111111101110111111111111111111*
L1536 01110111111111111111111111111111*
L2048 00000010*
L2127 1*
L2128 11111111111111111111111111111111111111111111111111111111111111*
L2192 1*
C1FE6*
8 3C8
```

El archivo Chip Report consta de seis páginas que contienen la información del resultado correspondiente al proceso de la compilación del archivo JEDEC se describen a continuación:

- Página 1 **Module** (Módulo)**'OR'**, en esta se incluyen los nombres de los archivos de entrada utilizados en el proceso de compilar y los nombres de los archivos salida generados.
- Página 2 **Programmed Logic** (Lógica Programada), aquí se muestran las ecuaciones implementadas en forma de Suma de Productos.
- Página 3 **Chip Diagram**, aquí se describe el diagrama del Circuito Integrado indicando la terminal correspondiente a cada una de las variables de entrada y salida.
- Página 4 **Resource Allocations,** aquí se describe el porcentaje de utilización del Circuito Integrado.
- Página 5 **Product Terms Distribution**, es en donde se indica la cantidad de términos usados por cada una de las variables.
- Página 6 **Unused Resources**, esta página nos muestra las terminales que no son usadas.

Archivo 7.3 Chip Report **OR.REP**

```
PAGE 1
SYNARIO 2.10  -  DEVICE UTILIZATION CHART        SUN APR 19 16:07:12 1998
--------------------------------------------------------------------------------
MODULE                  : 'OR'
--------------------------------------------------------------------------------

INPUT FILES:

    ABEL PLA FILE         : OR.TT3
    DEVICE LIBRARY        : P16V8AS.DEV

OUTPUT FILES:

    REPORT FILE           : OR.REP
    PROGRAMMER LOAD FILE  : OR.JED
```

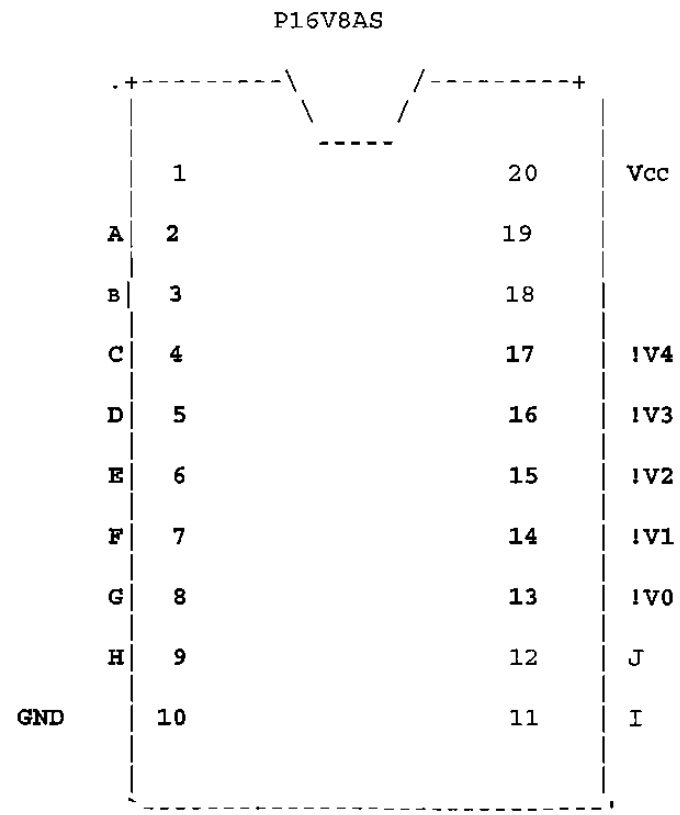
SYNARIO 2.10 - DEVICE UTILIZATION CHART SUN APR 19 16:07:12 1998

P16V8AS PROGRAMMED LOGIC:

```
V0      = (  A & B );

V1      = !(  !C & !D );

V2      = !(  E & F #   !E & !F );

V3      = !(  H & G );

V4      = (  !I & !J );
```


SYNARIO 2.10 - DEVICE UTILIZATION CHART SUN APR 19 16:07:12 1998

P16V8AS CHIP DIAGRAM:



SIGNATURE: N/A

PAGE 4

SYNARIO 2.10 - DEVICE UTILIZATION CHART SUN APR 19 16:07:12 1998

P16V8AS RESOURCE ALLOCATIONS:

DEVICE RESOURCES	RESOURCE AVAILABLE	DESIGN REQUIREMENT	UNUSED
INPUT PINS:			
INPUT:	10	9	1 (10 %)
OUTPUT PINS:			
IN/OUT:	6	4	2 (33 %)
OUTPUT:	2	2	0 (0 %)
BURIED NODES:			
INPUT REG:	-	-	-
PIN REG:	-	-	-
BURIED REG:	-	-	-

PAGE 5

SYNARIO 2.10 - DEVICE UTILIZATION CHART SUN APR 19 16:07:12 1998

P16V8AS PRODUCT TERMS DISTRIBUTION:

SIGNAL NAME	PIN ASSIGNED	TERMS USED	TERMS MAX	TERMS UNUSED
V0	13	1	8	7
V1	14	1	8	7
V2	15	2	8	6
V3	16	1	8	7
V4	17	1	8	7

==== LIST OF INPUTS/FEEDBACKS ====

SIGNAL NAME	PIN	PIN TYPE
A	2	INPUT
B	3	INPUT
C	4	INPUT
D	5	INPUT
E	6	INPUT
F	7	INPUT
H	9	INPUT
G	8	INPUT
I	11	INPUT
J	12	BIDIR

PAGE 6

SYNARIO 2.10 - DEVICE UTILIZATION CHART SUN APR 19 16:07:12 1998

P16V8AS UNUSED RESOURCES:

PIN NUMBER	PIN TYPE	PRODUCT TERMS	FLIP-FLOP TYPE
1	INPUT	-	-
18	BIDIR	NORMAL 8	-
19	BIDIR	NORMAL 8	-

7.1.5. Programar

Contamos con el programador Superpro III de la compañía XELTEC y en la programación efectuaremos los siguientes pasos:

- Ejecute el programa SPIII con el programador encendido, conectado al puerto de la computadora y el dispositivo fuera de su conexión.
- Seleccione el tipo de dispositivo programable.

  PAL

- En el menú seleccione el fabricante del dispositivo.

  Lattice.

- Seleccione el dispositivo.

  GAL16V8.

- Inserte el dispositivo en el socket, con precaución ya que estos dispositivos son CMOS y con una descarga estática se pueden dañar, de preferencia utilize una pulsera antiestatica conectada a tierra para asegurar un camino de descarga.
- Cargue el archivo JEDEC.

  OR.JED

- Grabe la nueva información en el dispositivo con la función programar y asegúrese de que en este proceso de carga no contenga errores.
- Apague el programador antes de retirar el dispositivo.

7.1.6 Verificación y Prueba

Se recomienda para una culminación satisfactoria del diseño lo siguiente:

- Elabore un plan de verificación de funcionamiento, auxiliado con el diagrama de entradas y salidas, y funciones deseadas que asegure la prueba de todas las funciones implementadas en el dispositivo.
- Efectuar todas las conexiones con el dispositivo desenergizado
- Asegurese de conectar correctamente la polaridad de la alimentación del dispositivo ya que un error puede dañar irreversiblemente al dispositivo.

7.2 Ejemplo de Sistema Secuencial

Este ejemplo consiste en un sistema de llenado de un tanque en el que contamos con tres bombas llamadas "A, B y C" que se usarán alternadamente, además como entrada tiene un sensor de nivel.

7.2.1 Conceptualización del problema

El Sistema de Llenado del Tanque deberá de trabajar con la siguiente secuencia:

Partiendo de que el tanque se encuentra vacío NIVEL=0 (!NIVEL) el sistema deberá encender la BOMBA A, hasta llenar el tanque NIVEL=1 (NIVEL) y entonces desconectarla manteniendo las tres bombas apagadas.

En el momento que se vacíe (¡NIVEL), el sistema cambiará de estado y encenderá la BOMBA B, hasta llenar el tanque NIVEL=1 (NIVEL) y entonces desconectarla.

Si de nuevo se vacía el tanque entonces deberá de encender la BOMBA C hasta llenar el tanque y entonces desconectarla, y así sucesivamente, el propósito del sistema es que las bombas se alternen en su funcionamiento.

En un Sistema Secuencial mediante un Diagrama de Transición se puede especificar claramente el funcionamiento.

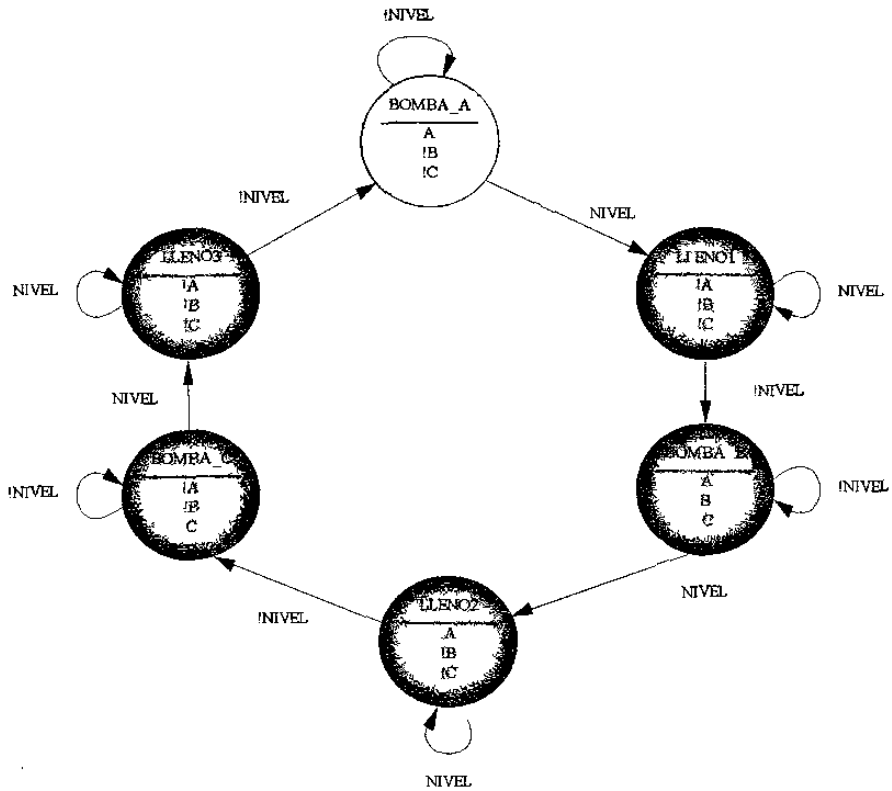


Fig 7.2 Diagrama de Transición del Sistema Secuencial

7.2.2 Selección del PLD

Se requiere de un dispositivo con 2 entradas una de reloj y la otra para el NIVEL, y seis salidas tres para las BOMBAS y las otras tres para los estados, seleccionamos el GAL16V8 de lattice.

7.2.3 Archivo de Diseño

El archivo de diseño se obtiene con el programa de aplicación StateCAD a partir del Diagrama de Transición, se compila y se genera el archivo B3.ABL en formato ABEL-HDL.

ARCHIVO 7.4 **"B3.ABL"** en formato ABEL-HDL.

```
MODULE B3



DECLARATIONS

"CLOCK NAME

        CLK PIN 1;



"Input variables

        NIVEL PIN 2;

"OUTPUT VARIABLES

        A PIN 12 ISTYPE 'COM';

        B PIN 13 ISTYPE 'COM';
        C PIN 14 ISTYPE 'COM';

"STATE VARIABLES
        SV0  PIN ISTYPE 'REG';
        SV1  PIN ISTYPE 'REG';
        SV2  PIN ISTYPE 'REG';

"STATE REGISTER ASSIGNMENT
DECLARATIONS
        SREG=[ SV0,SV1,SV2];

EQUATIONS
        SREG.CLK=CLK;

DECLARATIONS
        BOMBA_A=[0, 0, 0];
        BOMBA_B=[0, 0, 1];
        BOMBA_C=[0, 1, 0];
        LLENO1=[0, 1, 1];
        LLENO2=[1, 0, 0];
        LLENO3=[1, 0, 1];
STATE_DIAGRAM SREG;
STATE BOMBA_A:
        A=1;
        B=0;
        C=0;
        IF ( NIVEL ) THEN LLENO1;
        IF ( !NIVEL ) THEN BOMBA_A;
STATE BOMBA_B:
        A=0;
        B=1;
        C=0;
        IF ( !NIVEL ) THEN BOMBA_B;
        IF ( NIVEL ) THEN LLENO2;
STATE BOMBA_C:
        A=0;
        B=0;
        C=1;
        IF ( !NIVEL ) THEN BOMBA_C;
        IF ( NIVEL ) THEN LLENO3;
STATE LLENO1:
        A=0;
        B=0;
        C=0;
        IF ( NIVEL ) THEN LLENO1;
        IF ( !NIVEL ) THEN BOMBA_B;
STATE LLENO2:
        A=0;
        B=0;
        C=0;
        IF ( NIVEL ) THEN LLENO2;
        IF ( !NIVEL ) THEN BOMBA_C;
STATE LLENO3:
        A=0;
        B=0;
        C=0;
        IF ( NIVEL ) THEN LLENO3;
        IF ( !NIVEL ) THEN BOMBA_A;

END B3
```

7.2.4 Compilar

Los pasos en la compilación (Link Design) para obtener los archivos B3.JED JEDEC File y B3.REP Chip Report se representan en la siguiente figura:

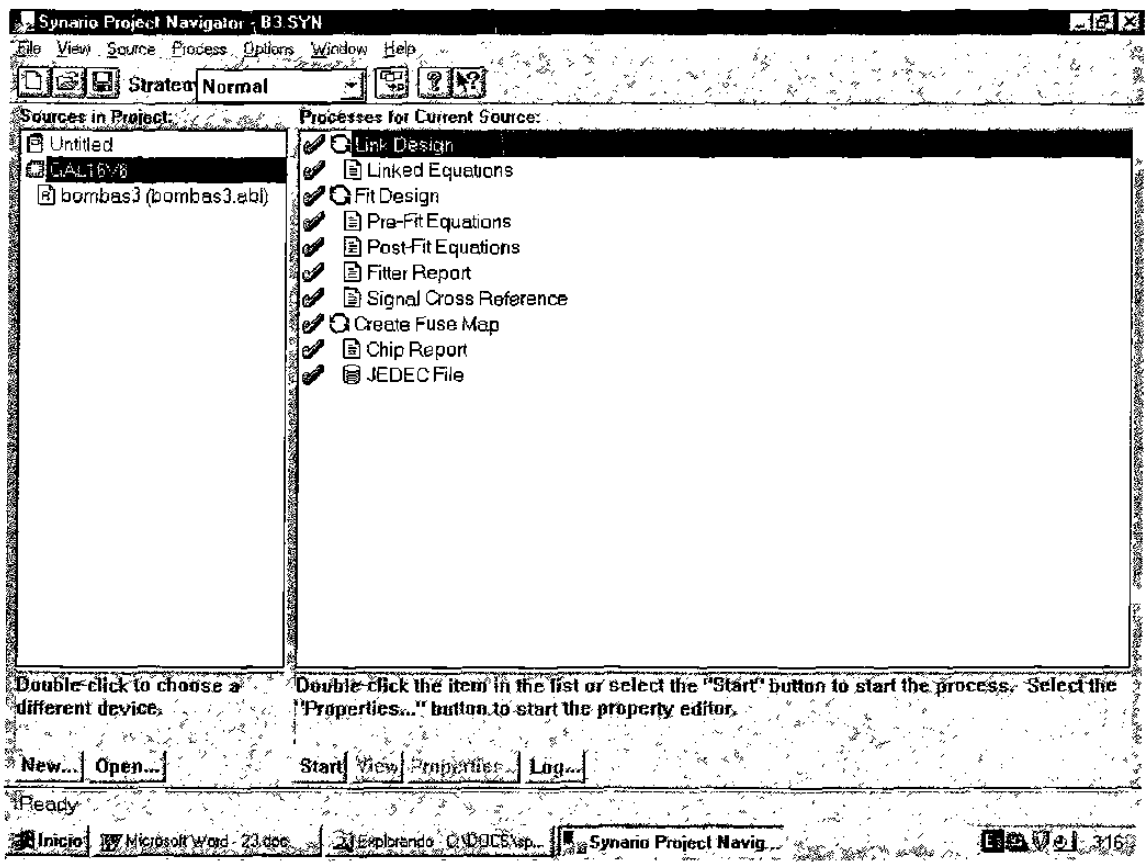


Fig. 7.3 Proceso de compilación en el programa SYNARIO.

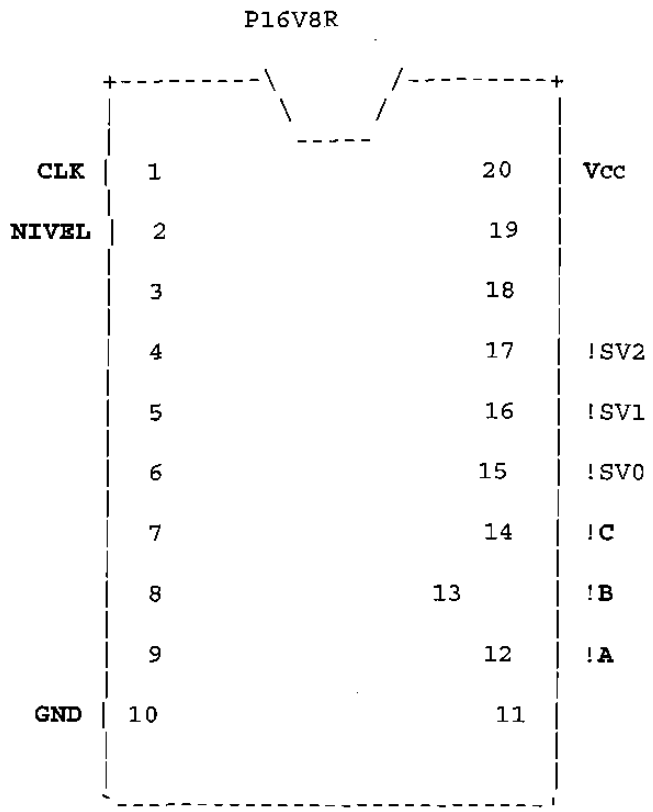
ARCHIVO 7.5 **B3.JED.**

```
S Y NARIO 2.10 DATA I/O CORP.
JEDEC FILE FOR: P16V8R V
CREATED ON: MON APR 20 17:23:25 1998
*

QP20* QF2194* QV0* F0* X0*

NOTE TABLE OF PIN NAMES AND NUMBERS*

NOTE PINS CLK:1 NIVEL:2 A:12 B:13 C:14 SV0:15 SV1:16 SV2:17*

L0512 11111111110111101110111111111111*
L0544 01111111111111101111011111111111*
L0576 01111111110111101111111111111111*
L0608 10111111111011101101111111111111*
L0768 11111111111011011110111111111111*
L0800 01111111111111011110111111111111*
L0832 10111111110111101101111111111111*
L1024 01111111111111101111111111111111*
L1280 11111111111111111111111111111111*
L1312 11111111110110111101111111111111*
L1536 11111111111111111111111111111111*
L1568 11111111110111101110111111111111*
L1792 11111111111111111111111111111111*
L1824 11111111110110111011011111111111*
L2048 00111111*
L2120 00000111*
L2128 11111111111111111111111111111111111111111111111111111111111111111111*
L2193 1*
C3CA6*
B A E2
```

ARCHIVO 7.6 **B3.REP.**

```
                                                                    PAGE 1
SYNARIO 2.10  -  DEVICE UTILIZATION CHART        MON APR 20 17:23:25 1998


--------------------------------------------------------------------------------

MODULE                  : 'B3'

--------------------------------------------------------------------------------

INPUT FILES:

   ABEL PLA FILE        : B3.TT3
   DEVICE LIBRARY       : P16V8R.DEV

OUTPUT FILES:

   REPORT FILE          : B3.REP
   PROGRAMMER LOAD FILE : B3.JED

--------------------------------------------------------------------------------


                                                                    PAGE 2
SYNARIO 2.10  -  DEVICE UTILIZATION CHART        MON APR 20 17:23:25 1998

P16V8R PROGRAMMED LOGIC:
--------------------------------------------------------------------------------


A      = (  SV0.Q & SV1.Q & SV2.Q );

B      = (  SV0.Q & SV1.Q & !SV2.Q );

C      = (  SV0.Q & !SV1.Q & SV2.Q );

SV0.D    = (  SV1.Q & NIVEL ); " ISTYPE 'INVERT'
SV0.C    = (  CLK );

SV1.D    = (  SV0.Q & !SV1.Q & SV2.Q
          #   SV0.Q & !SV1.Q & NIVEL
          #   !SV0.Q & SV1.Q & !SV2.Q & !NIVEL ); " ISTYPE 'INVERT'
SV1.C    = (  CLK );

SV2.D    = (  SV0.Q & SV1.Q & !SV2.Q
          #   SV0.Q & !SV1.Q & NIVEL
          #   SV1.Q & !SV2.Q & NIVEL
          #   !SV0.Q & SV1.Q & SV2.Q & !NIVEL ); " ISTYPE 'INVERT'
SV2.C    = (  CLK );
```

P16V8R CHIP DIAGRAM:



SIGNATURE: N/A

PAGE 4

SYNARIO 2.10 - DEVICE UTILIZATION CHART MON APR 20 17:23:25 1998

P16V8R RESOURCE ALLOCATIONS:

DEVICE RESOURCES	RESOURCE AVAILABLE	DESIGN REQUIREMENT	UNUSED
INPUT PINS:			
INPUT:	10	2	8 (80 %)
OUTPUT PINS:			
IN/OUT:	8	6	2 (25 %)
OUTPUT:	-	-	-
BURIED NODES:			
INPUT REG:	-	-	-
PIN REG:	8	3	5 (62 %)
BURIED REG:	-	-	-

PAGE 5

SYNARIO 2.10 - DEVICE UTILIZATION CHART MON APR 20 17:23:25 1998

P16V8R PRODUCT TERMS DISTRIBUTION:

SIGNAL NAME	PIN ASSIGNED	TERMS USED	TERMS MAX	TERMS UNUSED
A	12	1	7	6
B	13	1	7	6
C	14	1	7	6
SV0.D	15	1	8	7
SV1.D	16	3	8	5
SV2.D	17	4	8	4

==== LIST OF INPUTS/FEEDBACKS ====

SIGNAL NAME	PIN	PIN TYPE
CLK	1	CLK
NIVEL	2	INPUT

PAGE 6

SYNARIO 2.10 - DEVICE UTILIZATION CHART MON APR 20 17:23:25 1998

P16V8R UNUSED RESOURCES:

PIN NUMBER	PIN TYPE	PRODUCT TERMS	FLIP-FLOP TYPE
3	INPUT	-	-
4	INPUT	-	-
5	INPUT	-	-
6	INPUT	-	-
7	INPUT	-	-
8	INPUT	-	-
9	INPUT	-	-
18	BIDIR	NORMAL 7	D
19	BIDIR	NORMAL 7	D

7.2.5 Programar

Los pasos para Programar son los mismos que en el ejemplo del Diseño Combinacional.

7.2.6 Verificacion y Prueba.

En los Sistemas Secuenciales se requieren de pulsos de reloj, elabore un plan de verificación de funcionamiento (Diagrama de Tiempos), auxiliado con el diagrama de entradas y salidas, y funciones deseadas que asegure la prueba de todas las funciones implementadas en el dispositivo

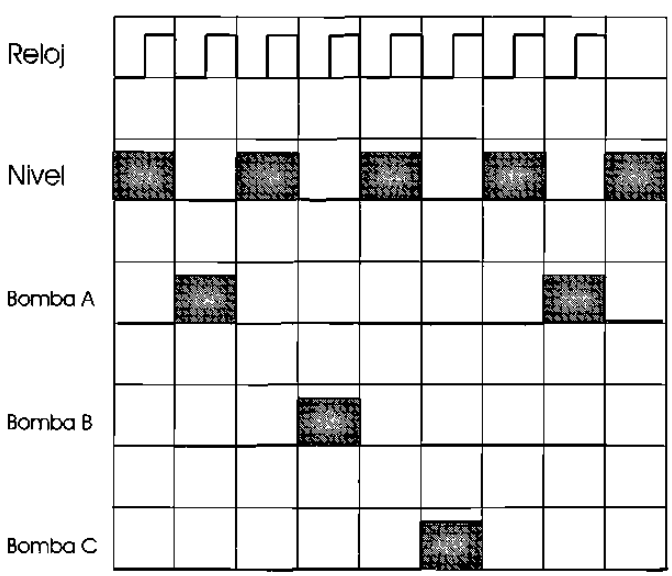


Fig. 7.4 Diagrama de Tiempos

No olvide polarizar adecuadamente el GAL16V8 en donde la terminal 10 se conecta a tierra GND y la terminal 20 a voltaje VCC= 5VCD.

CAPITULO 8 Conclusiones y recomendaciones

8.1 Conclusiones y Recomendaciones

Teniendo todos los elementos como Computadora Personal, programas de aplicación (software) y Programador, todo el procedimiento completo se puede llevar a cabo en un tiempo muy corto y en un escritorio sin que el diseñador tenga que ir a otro sitio.

La metodología propuesta ha sido probada con los diferentes diseños que se aplican normalmente en el curso de Sistemas Digitales (Electrónica Lógica I), en donde en los diseños más complejos se demuestra la ventaja de usar Dispositivos Lógicos Programables por la rapidez de programación, el menor número de circuitos integrados, además de su bajo costo.

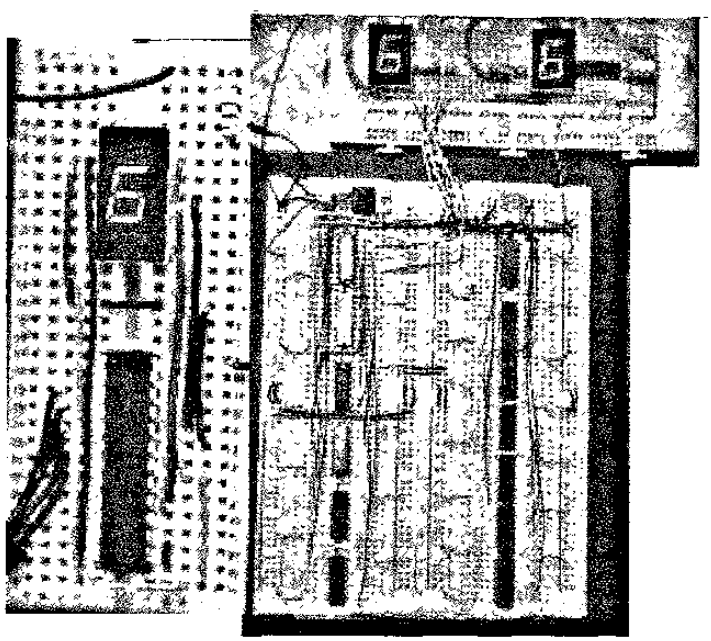


Fig. 8.1 Comparación de la implementación con PLDs vs el método tradicional.

Es recomendable que los alumnos en las prácticas iniciales utilicen las compuertas tradicionales TTL para demostrar los fundamentos del Algebra Booleana y se familiaricen en el manejo y conexión de circuitos integrados.

Se recomienda proponer proyectos para la evaluación del curso de Sistemas Digitales usando los Dispositivos Lógicos Programables ya que son económicos y no son difíciles de conseguir, el GAL16V8 y GAL20V8 son comerciales y su precio actual fluctúa entre los $15.00 y $50.00 M.N. y su capacidad (Entradas, Salidas y Registros) para los proyectos es aceptable, además estos dispositivos son reprogramables de modo que el mismo dispositivo se puede usar en varios proyectos.

El manejo de los Programas de aplicación, la Computadora Personal, el Programador y los Dispositivos Lógicos Programables se recomienda hacerlo con precaución siguiendo las indicaciones del fabricante de cada uno de los elementos, ya que el uso inapropiado fácilmente puede dañarlos.

Indice de Figuras, Archivos y Tablas

FIGURAS

ARCHIVOS

TABLAS

Bibliografía:

Bibliografía:

1 Barrón Ruiz Mariano, Lógica Programable, Mc Graw Hill 1994.

2 Bolton Martin, Digital System Design with Programable Logic, ADDISON WESLEY 1990.

3 PAL Devices Data Book and Design Guide, Advanced Micro Device 1996

4 Programmable Logic Devices Databook & Design Guide", National Semiconductor.

5 Roth Charles H., Logic Design, WEST, 1992.

6 The proLogic Compiler User´s Guide, Texas Instruments 1993.

7 ISP Synario Starter Software & ISP Encyclopedia, Lattice Semiconductor Corporation, 1996.

Enlaces a sitios de Internet relacionados con esta tecnología.

- Programmable Logical Devices –CUPL Software
  http://www.logicaldevices.com/CUPL.html
- Homepage de Lattice Semiconductor
  http://www.latticesemi.com/
- Universidad Pontificia Bolivariana
  http://sansua.upb.edu.co/microelectronica/
- Visual Software Solutions
  http://www.statecad.com
- Facultad de Ingeniería Universidad de Antioquia
  http://atenea.udea.edu.co

Apendice A

NOMENCLATURA GENERALIZADA DE LAS PALS.

Las PALCExx son CMOS y las PALxx son TTL. Sus tiempos de propagación son comparables. PALs programables varias veces: serie PALCExxx y GALs.

Las GALs tienen macroceldas programables. Son más caras que las PALs pero pueden implementar diseños hechos para varios tipos de PALs (ver figura a insertar, revista CEKIT). No necesitan ser borradas previamente a su programación y permiten la reprogramación en campo de pruebas. PALCExxx son PLDs de AMD equivalentes a las GALs de Lattice. Las PALxxVxx son reprogramables .Altera tiene EEPLDs.

FABRICANTES DE PALS Y GALS.

- AMD Advanced Micro Devices
- PHILIPS
- SAMSUNG
- Altera
- Actel
- Atmel
- Lattice Semiconductor
- Texas Instruments.
- XILINX
- INTEL

Lattice Semiconductor Corporation

Apendice B

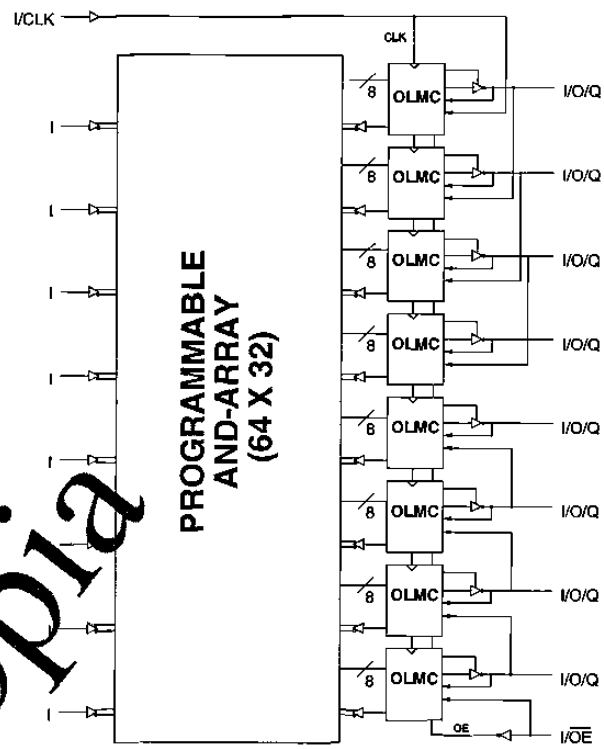
GAL16LV8

Low Voltage E²CMOS PLD Generic Array Logic™

FEATURES

- HIGH PERFORMANCE E²CMOS® TECHNOLOGY
  - 3.5 ns Maximum Propagation Delay
  - Fmax = 250 MHz
  - 2.5 ns Maximum from Clock Input to Data Output
  - UltraMOS® Advanced CMOS Technology
- 3.3V LOW VOLTAGE 16V8 ARCHITECTURE
  - JEDEC-Compatible 3.3V Interface Standard
  - Interfaces with Standard 5V TTL Devices (GAL16LV8C)
- ACTIVE PULL-UPS ON ALL PINS (GAL16LV8D Only)
- E² CELL TECHNOLOGY
  - Reconfigurable Logic
  - Reprogrammable Cells
  - 100% Tested/Guaranteed 100% Yields
  - High Speed Electrical Erasure (<100ms)
  - 20 Year Data Retention
- EIGHT OUTPUT LOGIC MACROCELLS
  - Maximum Flexibility for Complex Logic Designs
  - Programmable Output Polarity
- PRELOAD AND POWER-ON RESET OF ALL REGISTERS
  - 100% Functional Testability
- APPLICATIONS INCLUDE:
  - Glue Logic for 3.3V Systems
  - DMA Control
  - State Machine Control
  - High Speed Graphics Processing
  - Standard Logic Speed Upgrade
- ELECTRONIC SIGNATURE FOR IDENTIFICATION

FUNCTIONAL BLOCK DIAGRAM



DESCRIPTION

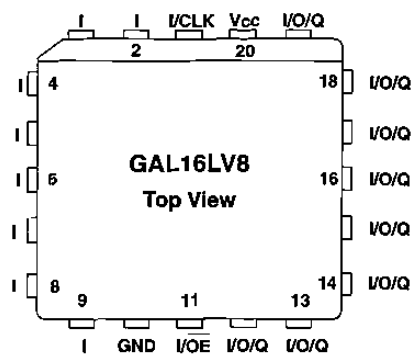
The GAL16LV8D, at 3.5 ns maximum propagation delay time, provides the highest speed performance available in the PLD market. The GAL16LV8C can interface with both 3.3V and 5V signal levels. The GAL16LV8 is manufactured using Lattice Semiconductor's advanced 3.3V E²CMOS process, which combines CMOS with Electrically Erasable (E²) floating gate technology. High speed erase times (<100ms) allow the devices to be reprogrammed quickly and efficiently.

The 3.3V GAL16LV8 uses the same industry standard 16V8 architecture as its 5V counterpart and supports all architectural features such as combinatorial or registered macrocell operations.

Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, Lattice Semiconductor guarantees 100% field programmability and functionality of all GAL products. In addition, 100 erase/write cycles and data retention in excess of 20 years are guaranteed.

PIN CONFIGURATION

PLCC



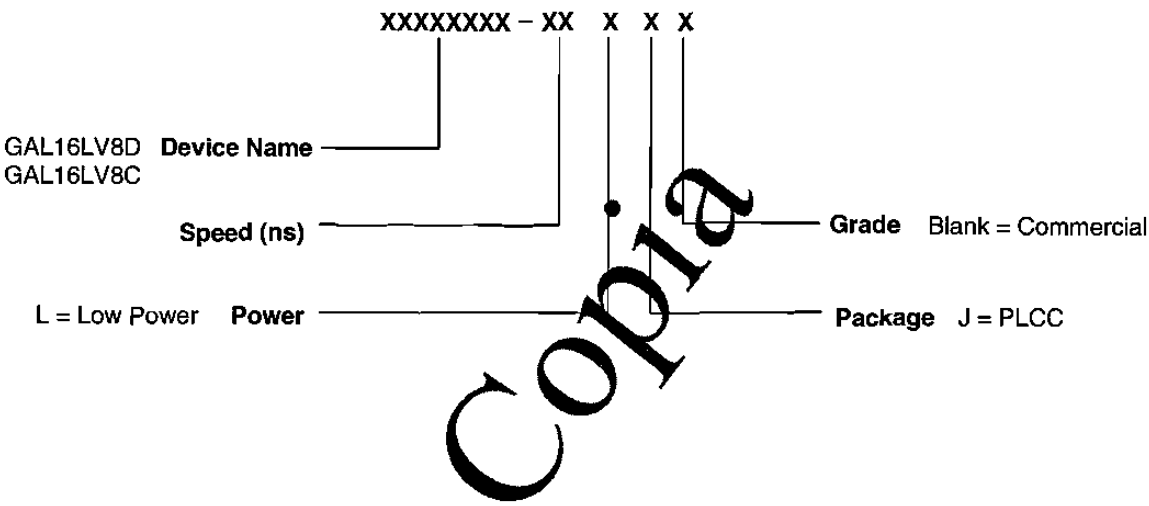
LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A.
Tel. (503) 681-0118; 1-800-FASTGAL; FAX (503) 681-3037; http://www.latticesemi.com

GAL16LV8 ORDERING INFORMATION

Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
3.5	3	2.5	70	GAL16LV8D-3LJ	20-Lead PLCC
5	4	3	70	GAL16LV8D-5LJ	20-Lead PLCC
7.5	6	5	65	GAL16LV8C-7LJ	20-Lead PLCC
10	7	7	65	GAL16LV8C-10LJ	20-Lead PLCC
15	12	10	65	GAL16LV8C-15LJ	20-Lead PLCC

PART NUMBER DESCRIPTION



OUTPUT LOGIC MACROCELL (OLMC)

The following discussion pertains to configuring the output logic macrocell. It should be noted that actual implementation is accomplished by development software/hardware and is completely transparent to the user.

There are three global OLMC configuration modes possible: **simple**, **complex**, and **registered**. Details of each of these modes are illustrated in the following pages. Two global bits, SYN and AC0, control the mode configuration for all macrocells. The XOR bit of each macrocell controls the polarity of the output in any of the three modes, while the AC1 bit of each of the macrocells controls the input/output configuration. These two global and 16 individual architecture bits define all possible configurations in a GAL16LV8. The information given on these architecture bits is only to give a better understanding of the device. Compiler software will transparently set these architecture bits from the pin definitions, so the user should not need to directly manipulate these architecture bits.

The following is a list of the PAL architectures that the GAL16LV8 can emulate. It also shows the OLMC mode under which the GAL16LV8 emulates the PAL architecture.

PAL Architectures Emulated by GAL16LV8	GAL16LV8 Global OLMC Mode
16R8	Registered
16R6	Registered
16R4	Registered
16RP8	Registered
16RP6	Registered
16RP4	Registered
16L8	Complex
16H8	Complex
16P8	Complex
10L8	Simple
12L6	Simple
14L4	Simple
16L2	Simple
10H8	Simple
12H6	Simple
14H4	Simple
16H2	Simple
10P8	Simple
12P6	Simple
14P4	Simple
16P2	Simple

COMPILER SUPPORT FOR OLMC

Software compilers support the three different global OLMC modes as different device types. These device types are listed in the table below. Most compilers have the ability to automatically select the device type, generally based on the register usage and output enable (OE) usage. Register usage on the device forces the software to choose the registered mode. All combinatorial outputs with OE controlled by the product term will force the software to choose the complex mode. The software will choose the simple mode only when all outputs are dedicated combinatorial without OE control. The different device types listed in the table can be used to override the automatic device selection by the software. For further details, refer to the compiler software manuals.

When using compiler software to configure the device, the user must pay special attention to the following restrictions in each mode.

In **registered mode** pin 1 and pin 11 are permanently configured as clock and output enable, respectively. These pins cannot be configured as dedicated inputs in the registered mode.

In **complex mode** pin 1 and pin 11 become dedicated inputs and use the feedback paths of pin 19 and pin 12 respectively. Because of this feedback path usage, pin 19 and pin 12 do not have the feedback option in this mode.

In **simple mode** all feedback paths of the output pins are routed via the adjacent pins. In doing so, the two inner most pins (pins 15 and 16) will not have the feedback option as these pins are always configured as dedicated combinatorial output.

	Registered	Complex	Simple	Auto Mode Select
ABEL	P16V8R	P16V8C	P16V8AS	P16V8
CUPL	G16V8MS	G16V8MA	G16V8AS	G16V8
LOG/iC	GAL16V8_R	GAL16V8_C7	GAL16V8_C8	GAL16V8
OrCAD-PLD	"Registered"[1]	"Complex"[1]	"Simple"[1]	GAL16V8A
PLDesigner	P16V8R[2]	P16V8C[2]	P16V8C[2]	P16V8A
TANGO-PLD	G16V8R	G16V8C	G16V8AS[3]	G16V8

1) Used with **Configuration** keyword.
2) Prior to Version 2.0 support.
3) Supported on Version 1.20 or later.

REGISTERED MODE

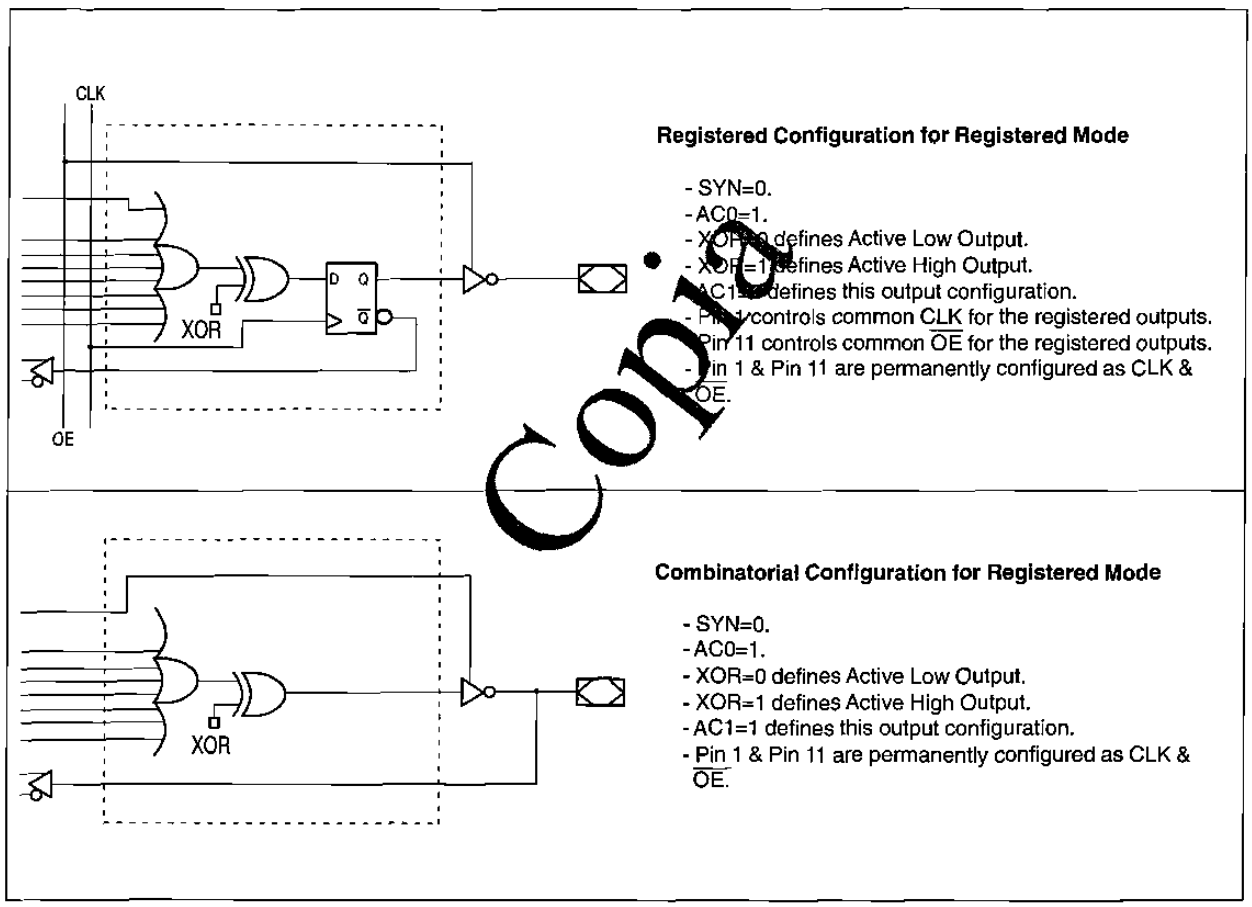
In the Registered mode, macrocells are configured as dedicated registered outputs or as I/O functions.

Architecture configurations available in this mode are similar to the common 16R8 and 16RP4 devices with various permutations of polarity, I/O and register placement.

All registered macrocells share common clock and output enable control pins. Any macrocell can be configured as registered or I/O. Up to eight registers or up to eight I/O's are possible in this mode. Dedicated input or output functions can be implemented as subsets of the I/O function.

Registered outputs have eight product terms per output. I/O's have seven product terms per output.

The JEDEC fuse numbers, including the User Electronic Signature (UES) fuses and the Product Term Disable (PTD) fuses, are shown on the logic diagram on the following page.



Registered Configuration for Registered Mode

- SYN=0.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1 defines this output configuration.
- Pin 1 controls common CLK for the registered outputs.
- Pin 11 controls common $\overline{OE}$ for the registered outputs.
- Pin 1 & Pin 11 are permanently configured as CLK & $\overline{OE}$.

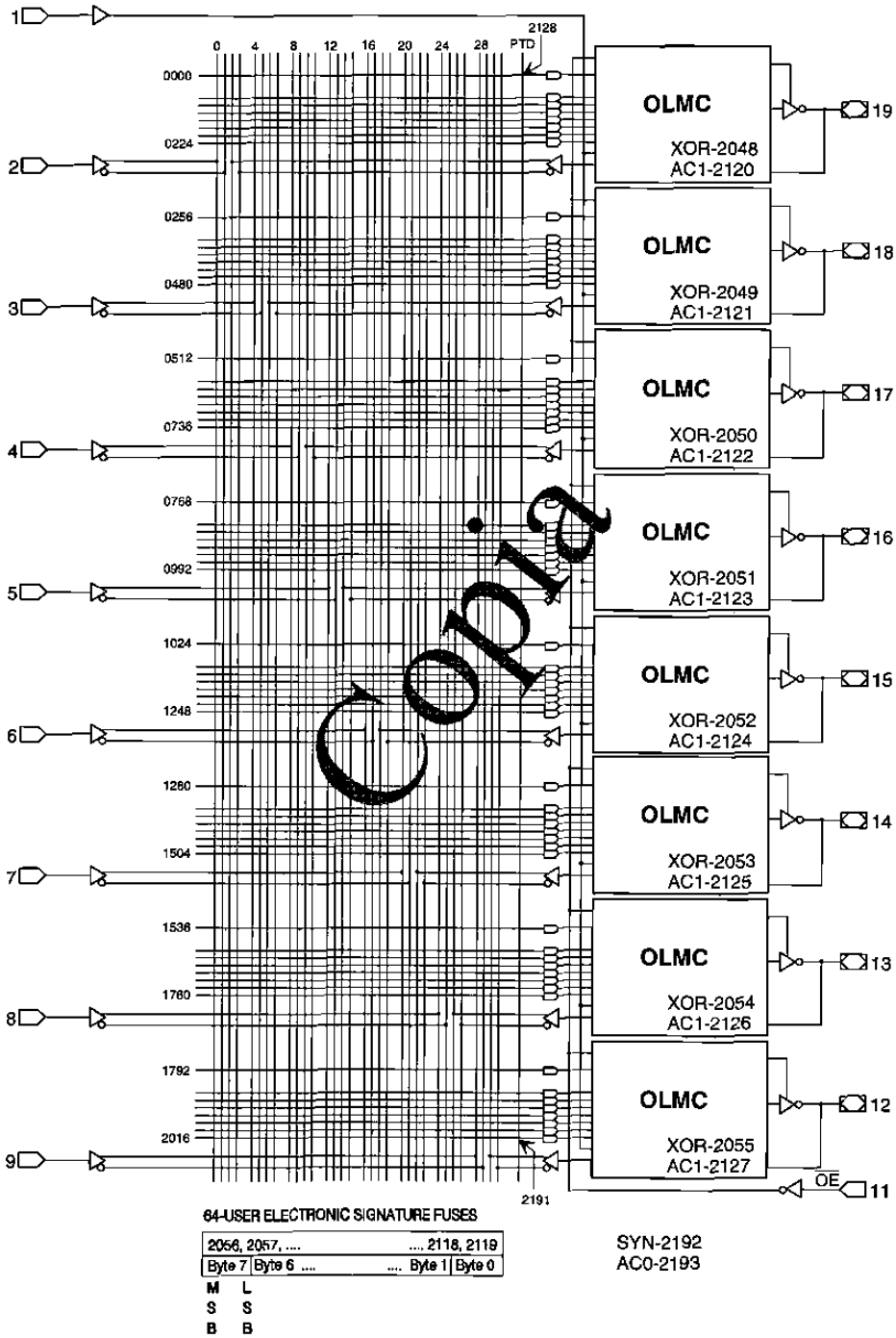
Combinatorial Configuration for Registered Mode

- SYN=0.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1 defines this output configuration.
- Pin 1 & Pin 11 are permanently configured as CLK & $\overline{OE}$.

Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

REGISTERED MODE LOGIC DIAGRAM

PLCC Package Pinout



COMPLEX MODE

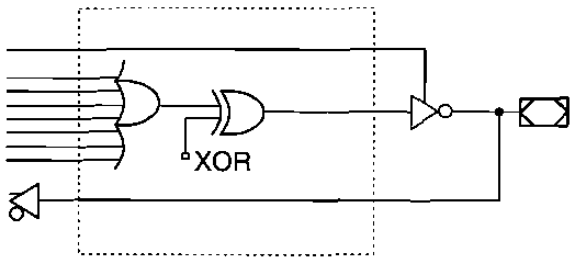
In the Complex mode, macrocells are configured as output only or I/O functions.

Architecture configurations available in this mode are similar to the common 16L8 and 16P8 devices with programmable polarity in each macrocell.

Up to six I/O's are possible in this mode. Dedicated inputs or outputs can be implemented as subsets of the I/O function. The two outer most macrocells (pins 12 & 19) do not have input capability. Designs requiring eight I/O's can be implemented in the Registered mode.

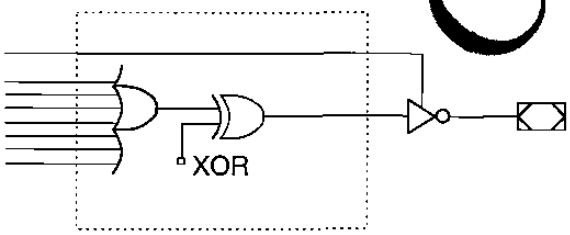
All macrocells have seven product terms per output. One product term is used for programmable output enable control. Pins 1 and 11 are always available as data inputs into the AND array.

The JEDEC fuse numbers including the UES fuses and PTD fuses are shown on the logic diagram on the following page.



Combinatorial I/O Configuration for Complex Mode

- SYN=1.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1.
- Pin 13 through Pin 18 are configured to this function.



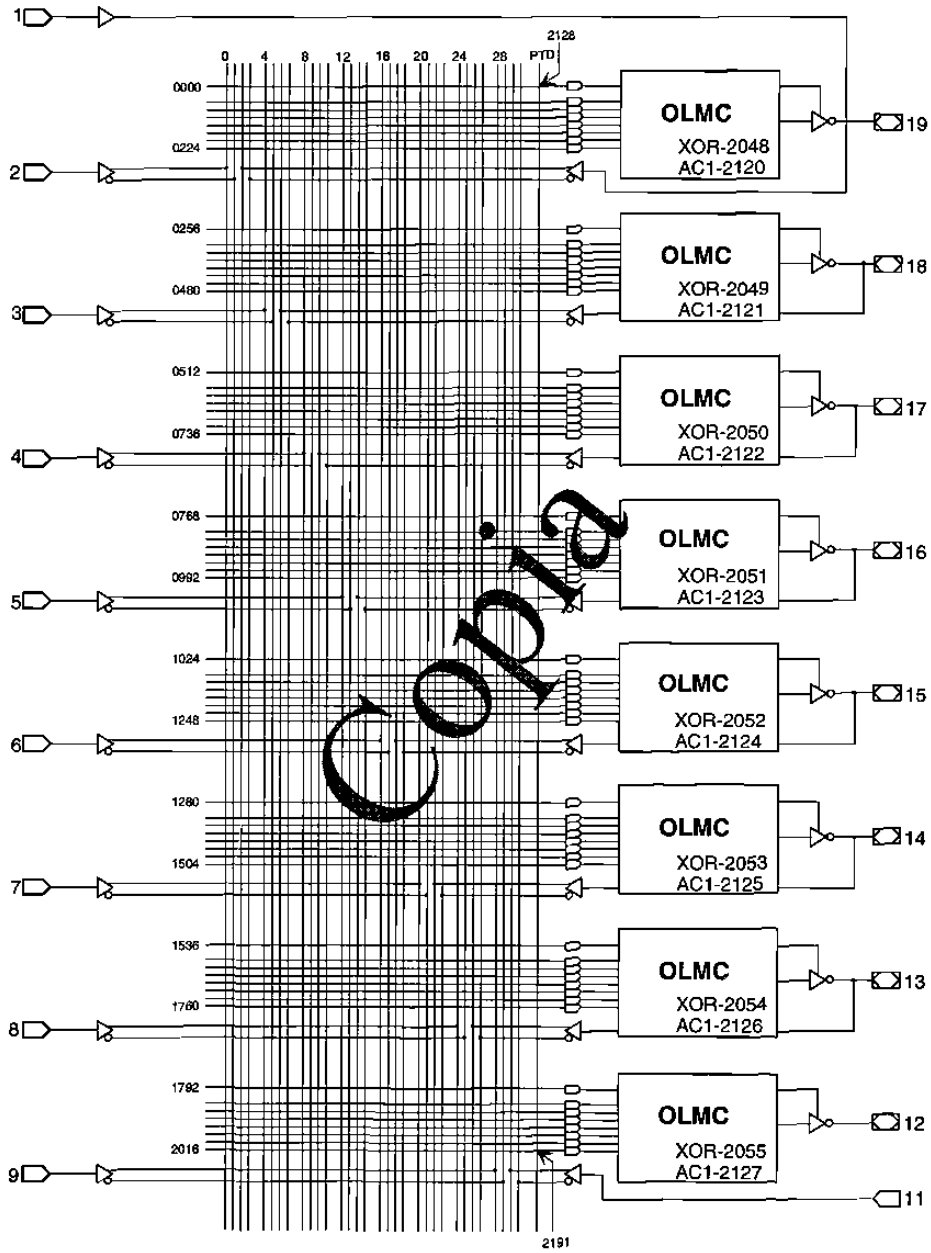
Combinatorial Output Configuration for Complex Mode

- SYN=1.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1.
- Pin 12 and Pin 19 are configured to this function.

Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

COMPLEX MODE LOGIC DIAGRAM

PLCC Package Pinout



SIMPLE MODE

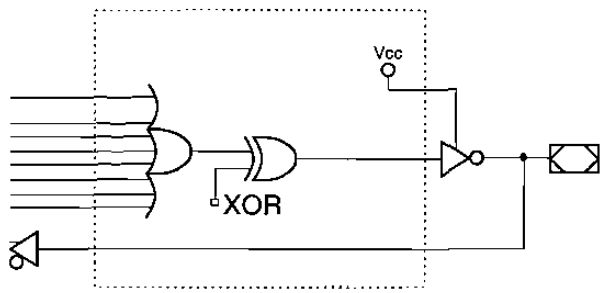
In the Simple mode, macrocells are configured as dedicated inputs or as dedicated, always active, combinatorial outputs.

Architecture configurations available in this mode are similar to the common 10L8 and 12P6 devices with many permutations of generic output polarity or input choices.

All outputs in the simple mode have a maximum of eight product terms that can control the logic. In addition, each output has programmable polarity.

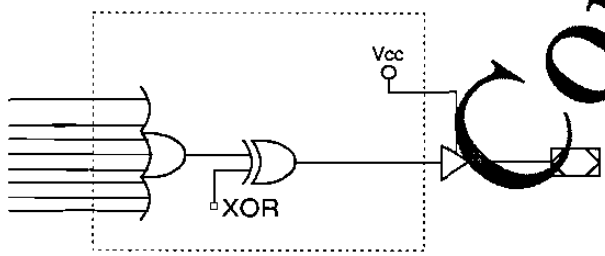
Pins 1 and 11 are always available as data inputs into the AND array. The center two macrocells (pins 15 & 16) cannot be used as input or I/O pins, and are only available as dedicated outputs.

The JEDEC fuse numbers including the UES fuses and PTD fuses are shown on the logic diagram.



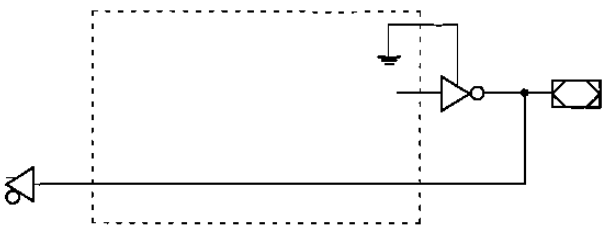
Combinatorial Output with Feedback Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=0 defines this configuration.
- All OLMC **except** pins 15 & 16 can be configured to this function.



Combinatorial Output Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=0 defines this configuration.
- Pins 15 & 16 are permanently configured to this function.



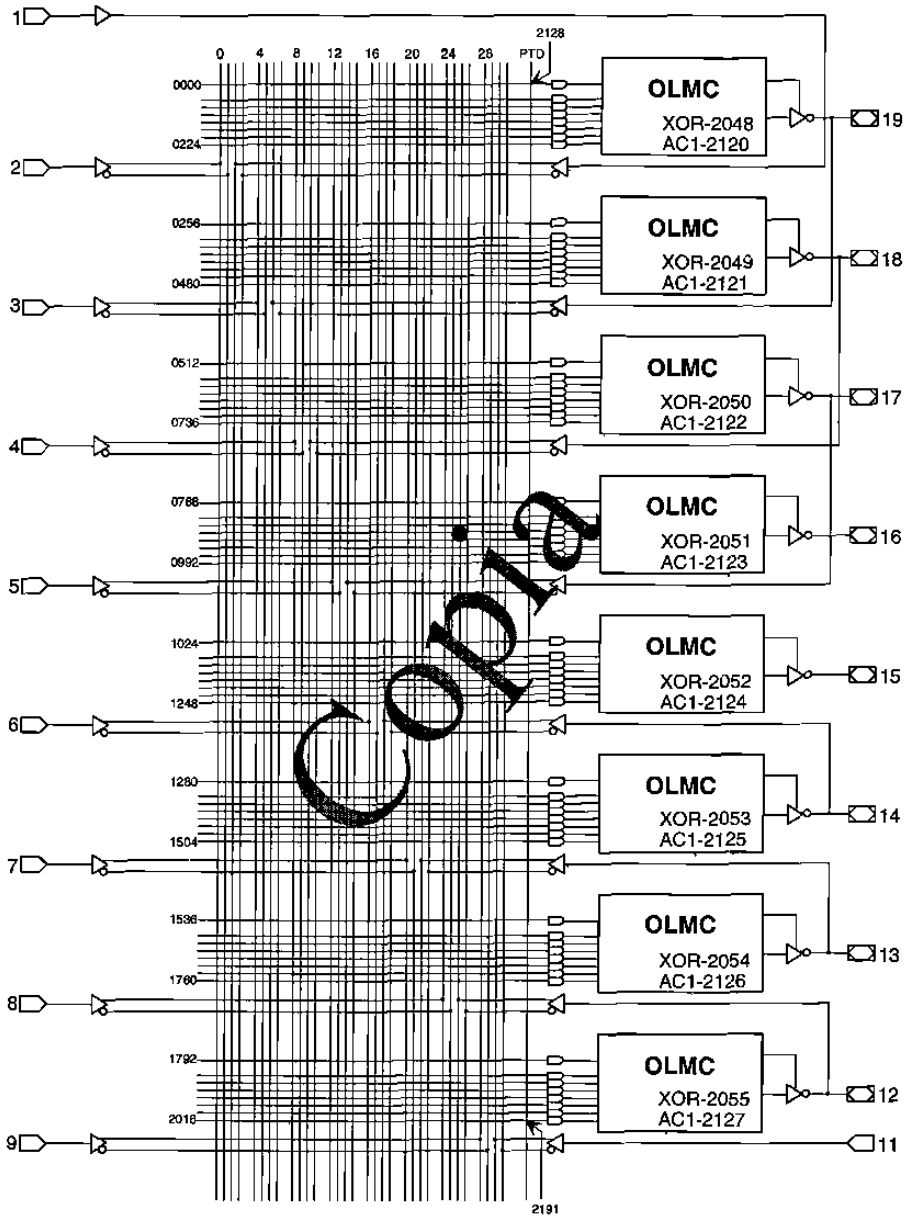
Dedicated Input Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1 defines this configuration.
- All OLMC **except** pins 15 & 16 can be configured to this function.

Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

SIMPLE MODE LOGIC DIAGRAM

PLCC Package Pinout



64-USER ELECTRONIC SIGNATURE FUSES

2056, 2057, ...	.... 2118, 2119
Byte 7 | Byte 6	.... Byte 1 | Byte 0

MSB LSB

SYN-2192
AC0-2193

Glosario de Términos

- **ABEL** (Advanced Boolean Expression Language): Es un lenguage universal para el diseño con PLD.
- **AIM** (Avalanche Induced Migration): Es una tecnología que utiliza un transistor como fusible, donde una corriente elevada une la base con el emisor.
- **Antifusible**: Es lo contrario del fusible, es un circuito abierto que se puede programar para ser una baja impedancia. Es, al igual que el fusible, OTP.
- **Arreglo de Compuertas**: Es un grupo de transistores que se configura por el usuario en los niveles de conexión metálicos, formando funciones lógicas.
- **ASICS** (Application Specific Integrated Circuits.) Circuitos Integrados de aplicación especifíca.
- **Asynchronous Clocks**: En algunos CPLDs se utilizan productos internos para generar el clock del flip-flop.
- **Bloque**: Un bloque es una parte de un PLD, el cual está formado por varios elementos lógicos con interconexión programable entre sí. Varios bloques interconectados forman el dispositivo.
- **Bus global**: Son unas rutas dentro de un integrado que permite conectar todos los elementos lógicos.
- **Circuitos estándar de celdas**: Dispositivo donde los bloques se encuentran en áreas con dimensiones específicas, usualmente con una altura fija y un ancho variable para cada celda.
- **Circuitos para aplicaciones específicas** (Full Custom Circuits): Es un dispositivo en el cual se diseña el tamaño, la interconexión y todas las características para cumplir una aplicación específica.
- **CLB** (Complex Logic Block): Bloque lógico complejo. Es un arreglo de celdas de múltiples entradas y múltiples salidas programables.
- **Conectividad Global**: Es una arquitectura que poseen algunos CPLD que permite usar todas las entradas y salidas en los productos de todas las macroceldas al mismo tiempo. Entrada dedicada: Es una terminal de entrada que no puede ser usado como salida.

- **CPLD** (complex programmable logic device): Es un integrado donde se tienen varios PLDs con una red de rutas que permite interconectarlos y ejecutar funciones lógicas más complejas.
- **Diseño Jerárquico**: Un diseño organizado comenzando desde los niveles más simples hasta los más complejos.
- **Dispositivo**: Se refiere a un circuito integrado CI.
- **DPGA:** Dynamically Programmable Gate Array, otro nombre para DR FPGA
- **DR FPGA** (FPGA reconfigurable dinámicamente): Es un FPGA que puede ser reprogramado durante la operación del sistema. Algunos permiten reconfigurar algunas partes y otros deben ser reprogramados completamente.
- **DRL** (Lógica reconfigurable dinámicamente): Son circuitos lógicos digitales que pueden ser reconfigurados durante la operación del sistema.
- **E2CMOS**: (Electrically Erasable Complementary Metal Oxide Semiconductor)
- **EDA** (Electronic Design Automation) es el nombre que se le da a todas las herramientas (tanto hardware como software) para la ayuda al diseño de sistemas electrónicos
- **EEPLD**: Es un PLD que utiliza celdas de memoria EEPROM para guardar la lógica programada. Es mucho más complejo que un PLD simple.
- **EEPROM** o E2PROM (Electrically Erasable Programmable Read-Only Memory).
- **EPLD**: Es un PLD que utiliza celdas de memoria EPROM en vez de fusibles para guardar la lógica programada.
- **EPROM** (Erasable Programmable Read-Only Memory).
- **EPROM**: Es una ROM (Read Only Memory) que es programada inyectando cargas en una compuerta MOS flotante, y borrada exponiendola a rayos ultravioleta.
- **ERA** (Electrically Reconfigurable Array).
- **ERASIC** (Erasable Reprogrammable ASIC).
- **Flash**: Es una tecnología de memorias no volátiles, que permite bajos costos y altos desempeños. Los dispositivos con esta tecnología son borrados y programados eléctricamente.
- **FPCB** (Field Programmable Circuit Board).

- **FPGA** (Field Programmable Gate Array): Consiste de un arreglo de bloques lógicos, rodeado de bloques de entrada/salida programables, y conectados a través de interconexiones programables.
- **FPID** (Field Programmable Interconnect Device).
- **FPLA** (Field Programmable Logic Array): Es un PLD que posee tanto las AND como las OR programables, pero con la complejidad de un PLD simple.
- **FPLS** (Field Programmable Logic Sequencer): Son FPLA que poseen registros.
- **Fusible** (fuse): Es un elemento de baja resistencia que puede ser modificado en un circuito abierto. La programación del fusible se denomina "quemar" el fusible y suele ser térmicamente mediante corrientes elevadas para este. Es OTP (sólo se puede programar una vez).
- **HCPLD** (High Capacity Programmable Logic Device) : En esta categoría se encuentran los CPLDs y FPGAs.
- **HDL** (Hardware Description Language): Es un lenguaje que permite describir un diseño lógico usando ecuaciones Booleanas, tablas de verdad y estados.
- **JEDEC** (Joint Electron Device Engineering Concil)**:** Los archivos JEDEC contienen el mapa de fusibles del PLD listo a ser programado.
- **Lógica Cache**: Término para denotar DRL, debido a las similitudes entre la operación de DRL y la de la memoria cache.
- **LUT** (Look-Up Table): Es una implementación alternativa de un CLB, donde las múltiples entradas generan las salidas complejas. Básicamente es una memoria que contiene una tabla de verdad.
- **Macrocelda**: Es un circuito en bloque que contiene compuertas OR para sumar los productos (resultados del arreglo de AND. Además contiene flip-flops, un buffer tres estados, y varios multiplexores para seleccionar las señales de control.
- **Mask-programmable**: Dispositivos, por lo general arreglos de compuerta que son programados en fábrica, poniendo conexiones de metal entre los elementos lógicos.
- **Maxitérminos**: termino Or que contiene todas las variables de la función ya sea en su forma normal o complementada.

- **Minimización de lógica**: Es un proceso en el cual una expresión Booleana se simplifica para que requiera menos compuertas (espacio).
- **Minitérminos** termino producto (And) que contiene todas las variables de la función ya sea en su forma normal o complementada.
- **MOS** (metal-oxide semiconductor): Es una tecnología para crear transistores controlados por voltaje.
- **No-Volátil**: Se refiere a una memoria que no necesita estar alimentada para conservar la información programada.
- **OTP** (one time programmable): Solo puede es programado una vez.
- **PAL** (Programmable Array Logic): Es una arquitectura que simplifica la de los PLAs. En esta los arreglos de OR son fijos y los de AND son programables.
- **PIC** (Programmable Integrated Circuit): Es cualquier circuito integrado que puede ser programado después de la fabricación de las capas de silicio.
- **PLA** (Programmable Logic Array): Es una arquitectura que utiliza un arreglo de AND programable, en serie con un arreglo de OR programable.
- **PLD** (programmable logic device): Es un circuito que puede ser configurado por el usuario para que realice una función lógica. Estos suelen estar constituidos por un arreglo de compuertas ANDs seguidos por un arreglo de compuertas ORs. Normalmente se utiliza para pequeños PLDs como PALs y FPLAs.
- **Producto de sumas**: Es una expresión lógica igual a la salida de un arreglo de compuertas OR seguido por un arreglo de compuertas AND.
- **Producto de términos**: Es igual a la salida de un arreglo de compuertas AND.
- **Programable**: Que se puede configurar de una manera deseada.
- **Programación basada en Memoria**: Es un tipo de programación donde la forma de interconexión es guardada en una memoria, la cual es reprogramable.
- **Registro de entrada**: Es un flip-flop o un Latch en algunos CPLDs que mantiene las señales de entrada, utilizado cuando se multiplexa el bus.
- **Retroalimentación**: Es un camino el cual conecta una señal generada internamente a una entrada. Suele ser programada y permite funciones lógicas.

- **Sistemas Combinacionales** son aquellos en donde los valores de salida únicamente dependen de las combinaciones de entrada.
- **Sistemas Secuenciales** : Son aquellos en donde los valores de salida no dependen únicamente de las combinaciones de entrada sino también de las salidas mismas.
- **SPLD** (Simple Programmable Logic Device).
- **SSI** (Small Scale Integration). Una medida de complejidad de un circuito integrado que es equivalente a 10 compuertas.
- **Standard Cell** Un método de diseño de circuitos semicustom o full custom en el cual se juntan las células predefinidas para obtener una función predeterminada.
- **Three State** Es un tipo de salida de un dispositivo lógico que puede tomar el valor de uno, cero y alta impedancia.
- **TTL** Transistor Transistor Logic: Familia de dispositivos lógicos bipolares mas usada.
- **VERILOG**: Lenguaje de diseño donde se introduce la descripción de hardware de alto nivel.
- **VHDL** (VHSIC Hardware Description Language): Es uno de los lenguajes de programación de dispositivos lógicos más utilizados. Creado por el Departamento de Defensa de Estados Unidos.
- **VHSIC** (Very High Speed Integrated Circuit): Circuito integrado de muy alta velocidad. Se comenzó a desarrollar por el Departamento de Defensa de los Estados Unidos (1979).
- **ZIF**(Zero Insertion Force Socket).
- **ZPAL** (A zero-power PAL): Es un PAL de bajo consumo. Nombre utilizado por AMD para sus dispositivos.

INDICE ALFABÉTICO

Resumen Autobiográfico

Nombre:	Juan Angel Garza Garza
Nombre de los padres:	Nicolás Garza y Garza + Blanca Amelia Garza González
Lugar y fecha de nacimiento:	Monterrey N.L. México 28 de Noviembre de 1953
Grado de Escolaridad:	Ingeniero Mecánico Electricista (1972-1977) Facultad de Ingeniería Mecánica y Eléctrica Universidad Autónoma de Nuevo León
Especialización técnica:	Servicios a la Instrumentación, SECIL A.C. (CONACYT y ANUIES) Servicios Centrales de Instrumentación y Laboratorios. México D.F. (1978-1979)
Campo Profesional:	Catedrático de la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León, Impartiendo las clases de Electrónica Lógica I desde 1979 a la fecha. Coordinador del Departamento de Informática y Servicios de Cómputo desde 1996 a la fecha.
Asociaciones:	ANIEI Asociación Nacional de Instituciones de Educación en Informática A.C.
Grado que desea obtener:	Maestro en Ciencias de la Ingeniería Eléctrica con especialidad en Electrónica.
Nombre de la tesis:	Diseño Lógico Programable.