UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"ARAGON"

SÍNTESIS DE DISPOSITIVOS DIGITALES MEDIANTE EL USO DEL LENGUAJE AVANZADO DE EXPRESIONES BOOLEANAS "ABEL"

T E S I S

Que para obtener el Título de:

INGENIERO EN COMPUTACIÓN

P R E S E N T A N

CELIA ROSA FIERRO SANTILLÁN

JOSÉ FILIBERTO LULE FLORES

México 1998

Sabiduría ante todo; adquiere sabiduría;
y sobre todas tus posesiones adquiere inteligencia.
Ella te honrará cuando tú la hayas abrazado.
Cuando anduvieres no se estrecharán tus pasos,
y si corrieres no tropezarás.

PROVERBIOS

Tengo. vamos a ver,
que ya aprendí a leer, a contar;
tengo que ya aprendí a escribir
y a pensar
y a reir.
Tengo, vamos a ver,
que ya tengo
lo que tenía que tener.

NICOLÁS GILLÉN

Hoy llego a la culminación de un sueño largamente anhelado. Ha sido un camino largo y difícil; si he podido llegar al final es porque no estuve sola.

Dedico este trabajo de tesis:

A Dios, mi creador, por el infinito amor con que me depositó en el vientre de mi madre.

A Jesucristo, mi salvador, porque siempre me ha llevado en sus brazos y en los momentos difíciles acarició mi corazón entristecido.

Al Espíritu Santo, por regalarme los dones de la fortaleza y la intuición.

A Israel Fierro Cruz, mi papá, por enseñarme con su ejemplo que se debe luchar hasta el final por lo que se ama, por las carencias y privaciones que pasó para darme una carrera y por enseñarme que la posesión más valiosa es el conocimiento.

A María de la Luz Santillán Sagala, mi mamá, porque estuvo siempre conmigo apoyándome y guiándome, porque ha sido como un manantial de luz en mi vida que ha iluminado mi camino cuando parecía demasiado oscuro, por las noches que pasó en vela conmigo para que yo terminara mis tareas, por haber cuidado a mi hija con el mismo amor que lo hubiera hecho yo, por los incontables detalles de amor y de aliento que ha tenido para conmigo. Mamá tu obra es la de un gigante, yo no creo ser capaz de superarla.

A Guadalupe Sagala, mi abuelita, por haberme hecho creer que yo era una niña inteligente y que podría llegar al final de una carrera universitaria.

A Filiberto Lule Flores, mi amado esposo y compañero, porque cuando creí que yo no podría terminar mi carrera el me impulsó a seguir adelante y me enseñó lo importante de la disciplina y la constancia, por haberme enseñado cálculo diferencial e integral con infinito amor y paciencia lo que hizo posible el milagro, creo que ninguna otra persona lo habría logrado.

A Laura Celia Lule Fierro, mi amada hija, por su comprensión en las noches que se desveló sentada en una silla porque tenía miedo de dormir sola pero comprendía que mamá debía seguir trabajando. Hija yo no puedo pedirte que hagas lo que yo misma no he sido capaz de hacer, hoy puedo decirte que si eres fiel a tu verdad puedes llegar a realizar tus sueños.

A Alma Delia Fierro Santillán, mi hermana, por el amor fraternal que siempre me ha demostrado, por cuidar de mi hija, por su ayuda económica, por escucharme durante incontables horas y ayudarme a comprender el amasijo de contradicciones que soy.

A María Teresa Fierro Santillán, mi hermana, por ser mi amiga y compañera de la infancia y mi confidente en la adolescencia.

A mis compañeras y compañeros que escucharon durante años mi eterna letanía "Quiero ser Ingeniero", en especial a Luchita, Lupita y Leo quienes me dieron la sugerencia correcta en el momento correcto inspirados por un poder superior para guiarme en la hora más oscura hasta llegar a la luz; a Arturo en memoria de los tiempos en que fuimos "un trío de tres" y nos ayudamos mutuamente en la carrera y en la vida.

A mis hermanos en Cristo que me han tenido presente en sus oraciones.

A mis tíos: Celso Rafael Santillán Sagala, por las miradas de ternura que ha tenido para con mi hija, por alimentar mis sueños de niña y porque sé que desde el fondo de su corazón se siente orgulloso de mí y me ama; a Juan Ciro Santillán Sagala, por el apoyo que nos brindó cuando papá ya no estaba.

A la memoria del Ing. Juan Méndez Moreno y de mi compañera Paty.

A todos los familiares y amigos que me apoyaron en este difícil camino

Celia Rosa Fierro Santillán

Dedico este trabajo de tesis:

Al todopoderoso por ser el autor de todas las cosas y porque en el confío.

A mis padres Jesús Lule Guzmán y María Transito Flores por el apoyo que me dieron a través de la vida.

A mi esposa Celia Rosa Fierro Santillán por su insustituible apoyo y comprensión que con nada se puede pagar.

A mi hija Laura Celia Lule Fierro por su paciencia y sacrificio en las horas que no le dediqué.

A mis hermanos por ser mi familia y porque los quiero a todos y cada uno de ellos.

A este pueblo mexicano que son mi raza y que sufren y aguantan porque tienen alma de acero y corazón de bronce.

José Filiberto Lule Flores

Agradecemos:

A todos nuestros maestros, porque de cada uno de ellos aprendimos. En particular al M. en Ing. David González Maxinez. quien nos asesoró en esta tesis. por sus consejos y sugerencias, así como la amistad que nos ha brindado.

Al Ing. Francisco Javier González Torres, por su invaluable amistad y la información acerca de la máquina cortadora de cartón. las variables que en ella intervienen, su funcionamiento, etc.

A los Ingenieros: José Isabel Cárdenas . Antonia Navarro González. Juan Raúl Rea Pérez , Antonio Ortiz Peña y a La Lic. Guadalupe Almánzar Vázquez por sus comentarios y sugerencias en la revisión de este trabajo. las cuales fueron de gran valor para darle su forma final.

A Martha Eugenia Lule Flores por su ayuda para contactar a las empresas fabricantes de PLDs, lo cual fue esencial para obtener la bibliografía y los circuitos de esta tesis.

Celia Rosa Fierro Santillán
y José Filiberto Lule Flores

LISTA DE MARCAS REGISTRADAS QUE SE MENCIONAN EN ESTE TRABAJO

ABEL es una marca registrada por Data I/O Corporation.

Altera es una marca registrada por Altera Corporation.

ATGEN es una marca registrada por ACUGEN Software, Inc.

CUPL es una marca registrada por Logical Devices.

GAL es una marca registrada por Lattice Semiconductor.

IBM PC es una marca registrada por International Business Machines Corp.

MACH es una marca registrada por Advanced Micro Devices Inc.

MAPL es una marca registrada por National Semiconductor Corporation.

MS-DOS y **Windows** son marcas registradas por Microsoft Corporation.

OPAL y **OPALjr** son marcas registradas por National Semiconductor Corporation.

PAL es una marca por Advanced Micro Devices.

PALASM es una marca registrada por Advanced Micro Devices, Inc.

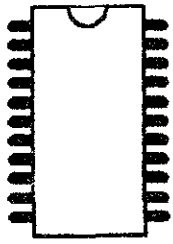
PGADesigner es una marca registrada por Minc Incorporated.

PLDesigner es una marca registrada por AT & Bell Laboratories.

VAX es una marca registrada por View Logic Systems. Inc.

ViewPLD es una marca registrada por Viewlogic Systems, Inc.

ÍNDICE



INTRODUCCIÓN

El Lenguaje Avanzado de Expresiones Booleanas (**ABEL**) fue creado por **Data I/O Corporation** en 1984, actualmente es muy utilizado en varios países, sin embargo, en México es poco conocido. En esta tesis se utilizó la versión 5 de **ABEL-EDU**, la cual puede trabajar con circuitos PAL (Arreglo lógico programable), GAL (Arreglo lógico genérico), y el circuito MACH215 (Macro - Arreglo CMOS de Alta Densidad). El trabajo se enfocó a los dos últimos debido a que los circuitos PAL actualmente han sido desplazados en el mercado nacional por los GALs que son más versátiles; mientras que los MACH son prácticamente desconocidos en nuestro país.

En este trabajo se supone una familiaridad con los conceptos básicos de programación (menú, comando, compilar, software, hardware etc.) y diseño de circuitos de electrónica digital (métodos de reducción de álgebra booleana, de Quine Mc. Cluskey y mapas de Karnaught). Razón por la cual no se dan explicaciones extensivas acerca de ellos.

En el capítulo uno, se explica la lógica programable y los tipos de diagramas que en ella se utilizan. También se hace una descripción de los distintos tipos de PLDs (Dispositivos Lógicos Programables) que existen actualmente, configuración interna, tecnología que se utiliza en su elaboración al igual que las ventajas y desventajas de cada uno con respecto a los otros. Además se explican brevemente las características de los lenguajes de programación de PLDs más utilizados actualmente.

En el capítulo dos, se hace una breve introducción al lenguaje **ABEL** y la forma de programar en él, secciones principales, tipos de descripción lógica que utiliza, palabras reservadas y reglas para el uso de variables y constantes.

En el capítulo tres se elabora una descripción minuciosa del ambiente de programación **ABEL**: comandos principales, subcomandos, opciones de compilación y trazado de los vectores de prueba en la simulación, etc.

En el capítulo cuatro se describe una metodología para el diseño de circuitos combinacionales utilizando dispositivos GAL. Los circuitos que aquí se describen no son comerciales, con la finalidad de demostrar que por medio de **ABEL** y los circuitos GAL podemos tener circuitos lógicos para propósitos específicos.

En el capítulo cinco se establece una metodología para diseñar circuitos combinacionales o máquinas de estado, se describen las diferentes maneras de realizar estos circuitos como son las tablas de transición de estados y los diagramas

de estado. También se explican algunas técnicas para optimizar las máquinas de estado, tales como la partición de diagramas de estado.

En el capítulo seis se describe la metodología para realizar controladores mediante el uso de **ABEL**. A lo largo del capítulo se desarrolla un controlador para una máquina cortadora de cartón. Se decidió usar circuitos MACH, debido a la gran cantidad de señales que se utilizan en los circuitos auxiliares del controlador, concretamente se utilizó el MACH 215 para realizar un contador, un comparador y el propio controlador.

Finalmente se incluyen tres apéndices para complementar la información desarrollada en los seis capítulos, el primero de ellos se refiere a las instrucciones del lenguaje **ABEL**, el segundo trata de los circuitos GAL y el tercero del MACH 215.

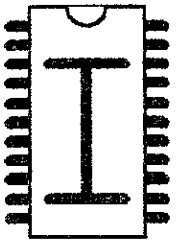
La herramienta esencial para la programación y simulación de los circuitos aquí expuestos fue la computadora. Se utilizó el paquete original de **ABEL - EDU** versión 5 con la finalidad de tener información de primera mano acerca de su funcionamiento. Como bibliografía básica se consultaron varios manuales de PLDs, principalmente de circuitos GAL y MACH. Los cuales, al igual que el software **ABEL** fueron importados de Estados Unidos, solicitándolos directamente a las empresas fabricantes de PLDs: Texas Instruments, National Semiconductors y Advanced Micro Devices.

Este trabajo tiene como finalidad dar a conocer una metodología de programación de circuitos GAL y MACH para aplicaciones no comerciales utilizando el lenguaje **ABEL**, describiendo en detalle el uso del mismo mediante ejemplos tradicionales en electrónica digital (comparadores, sumadores, codificadores, contadores, etc.) y otros menos populares (como el controlador que se describe en el capítulo seis). Demostrando que esta metodología es aplicable para diseñar la mayoría de los circuitos en los que se aplica la electrónica digital.

La mayor parte de la información contenida en esta tesis no existe en ninguna otra fuente escrita en español dentro de nuestro país y a pesar de que parte de esta puede ser consultada en inglés dentro de los manuales correspondientes en ninguno de ellos se explica la metodología que aquí se expone. En el caso particular de los diagramas y modos de funcionamiento de los distintos tipos de macroceldas[1] de los circuitos GAL descritas en el apéndice B se trata de información que no se encuentra en ninguna otra fuente de consulta y que fue desarrollada por los autores mediante el método deductivo aplicado a sus conocimientos de electrónica digital. Los ejemplos desarrollados también son propios de los autores. Por todo lo anterior la presente tesis es una aportación importante en el área de la electrónica digital en nuestro país.

---

[1] Las macroceldas constituyen el componente fundamental de los circuitos GAL (ver apéndice B).



CAPITULO I. INTRODUCCIÓN A LOS DISPOSITIVOS LÓGICOS PROGRAMABLES

I.1 DISPOSITIVOS LÓGICOS PROGRAMABLES

El termino dispositivo lógico programable (PLD) engloba a los circuitos integrados que pueden ser programados internamente para desempeñar una función especifica en lógica digital. Dicha programación es posible gracias a que los PLDs están formados por arreglos de compuertas conectadas a través de fusibles, que al romperse, determinan un estado lógico.

Aunque se habla de programación los cambios que se realizan son de hardware (dentro de los circuitos), ya que se modifican las características físicas internas del dispositivo rompiendo fusibles para darle la configuración adecuada. Esto es posible gracias al uso de celdas programables que permiten la creación de circuitos lógicos que pueden ser configurados en diseños específicos.

Desde la creación de los dispositivos LSI (Alta Escala de integración) existía un vacío entre los circuitos de alta densidad de propósito general como los microprocesadores y los de baja densidad de propósito especifico como los TTL [2]. Los PLDs lo cubren ya que pueden ser programados para la función especifica que se requiera.

Los PLDs pueden ser usados para reducir la cantidad de circuitos requeridos en un diseño sustituyendo a varios circuitos SSI (baja escala de integración) o MSI (mediana escala de integración), lo que incrementa la funcionalidad y elimina los errores causados por un cableado excesivo en la aplicación, además de economizar espacio y costo.

Un PLD típico puede tener cientos de compuertas interconectadas a través de fusibles internos. Un diagrama que mostrara todas esas interconexiones seria muy complejo, por eso se utiliza una simbología especial para las compuertas que componen los arreglos internos de un PLD. En la figura 1.1 se muestran los símbolos convencional y de arreglo para compuertas AND y OR. Como se puede observar, un símbolo convencional muestra cada entrada a la compuerta como una

[2] TTL (Lógica Transistor - Transistor) Familia de circuitos integrados de propósito especifico que es la más utilizada en diseño digital.

línea separada. En el símbolo de arreglo se muestra una sola línea de entrada a la compuerta, en la cual están conectadas todas las entradas.

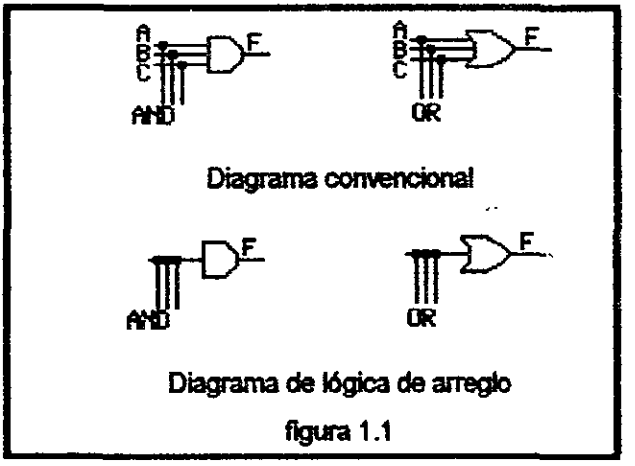


figura 1.1

La configuración interna de un PLD se representa a través de diagramas de lógica de arreglo. En cada intersección existe un fusible que puede romperse o permanecer intacto después de la programación. Un fusible intacto se representa por una X en la intersección de las líneas mientras que la ausencia de una X representa un fusible abierto (fig. 1.2).

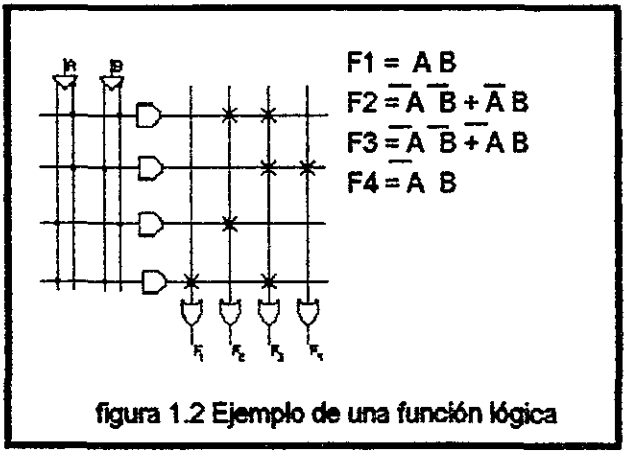


figura 1.2 Ejemplo de una función lógica

I.2 MARCO HISTÓRICO

En la década de los sesentas se empezaron a utilizar memorias programables de solo lectura (PROM) las cuales contenían internamente un arreglo de compuertas AND formando un decodificador y un arreglo de diodos con fusibles de aluminio en los puntos de conexión. Por medio de estos fusibles era posible programar la memoria una sola vez. En aplicaciones que no utilizan todos los minitérminos[3] posibles una memoria PROM se desperdicia. El problema fue resuelto introduciendo una segunda matriz de fusibles en el decodificador, con lo cual se pueden seleccionar solo los minitérminos que requiere el diseño. Este dispositivo se realizó usando fusibles configurables en campo, por lo que se le llamo arreglo de lógica programable en campo (FPLA o solo PLA).

Al tener dos arreglos de compuertas configurables se tiene un dispositivo de mayor complejidad, por lo tanto su fabricación es más cara. Una solución fue el crear un circuito el cual tiene una matriz de compuertas AND programable dejando el arreglo OR fijo. A este dispositivo se le llamo lógica de arreglo programable (PAL).

Todos los circuitos anteriormente mencionados se pueden programar una sola vez, por lo que no son funcionales al hacer modificaciones en un diseño determinado. Se requerían dispositivos que pudieran ser grabados varias veces, los cuales se lograron gracias al desarrollo de la celda de memoria eléctricamente borrable que fue utilizada primero en memorias EEPROM (Memorias Programables Eléctricamente Borrables) y posteriormente en un nuevo PLD, el GAL (Arreglo Lógico Genérico). Este dispositivo puede ser reprogramado varias veces y además puede sustituir a casi cualquier PAL.

Con el desarrollo de la tecnología en semiconductores se lograron mayores densidades en los circuitos integrados y fue posible crear PLDs de alta densidad, es decir, con un mayor número de compuertas, tales como las familias MAPL (Arreglo Múltiple de Lógica Programable) de National Semiconductores, y las familias MACH 1, 2, 3 y 4 (Macro Arreglo CMOS de Alta Densidad) de Advanced Micro Devices.

I.3 LÓGICA PROGRAMABLE

La lógica programable involucra a las herramientas de software (programas) y hardware(circuitos integrados) con las cuales se pueden realizar diseños de lógica digital grabando directamente en los fusibles internos de un PLD. El resultado

---

[3] Se le llama minitérmino a un producto de variables en el cual ninguna de ellas aparece más de una vez. Las funciones en electrónica digital se describen a través de minitérminos.

es una aplicación específica con un mínimo de circuitos integrados y una gran flexibilidad para hacer correcciones sobre el mismo dispositivo.

En electrónica digital el diseñador tiene varias opciones para resolver un problema: Usar circuitos LSI, microcontroladores, microprocesadores y PLDs. Los dispositivos LSI ofrecen una solución universal pero si el sistema a diseñar requiere una solución más simple y/o funciones especiales no disponibles en estos circuitos, es más eficiente el uso de lógica programable. Los microcontroladores y microprocesadores pueden ser programados en software, lo que los hace más funcionales, pero en casos en los que se utiliza un mínimo porcentaje de la capacidad de estos circuitos o se requiere de un controlador específico (como el controlador descrito en el capítulo seis) , puede resultar mas sencillo y económico el utilizar PLDs.

Las ventajas de utilizar lógica programable son las siguientes:

1. Reducción de costos: La aplicación se puede grabar usando un mínimo de dispositivos, los circuitos impresos son de menor tamaño, el cableado es prácticamente inexistente, se pueden hacer correcciones sobre el mismo PLD. Todo lo anterior da como resultado menores gastos.
2. Reducción de espacio: Un diseño puede ser grabado en un solo circuito.
3. Menor tiempo en el diseño: Con el uso del software adecuado es posible realizar un diseño en poco tiempo (desde algunos minutos hasta alguna horas).
4. Menor cantidad de errores: El uso de la simulación permite eliminar los errores antes de grabar el circuito, también se eliminan los errores debidos a falsos contactos en el cableado.

Las posibles desventajas de la lógica programable radican únicamente en no emplearla adecuadamente. Por ejemplo, si hay una aplicación específica disponible en circuitos TTL es innecesario diseñar un PLD que realice la misma función; si la cantidad de memoria que requiere un diseño es muy grande es preferible utilizar una memoria programable; si el algoritmo[4] de diseño es complejo y utiliza subrutinas seria mejor el uso de un microprocesador.

[4] Secuencia de pasos para resolver un problema.

I.4 TECNOLOGÍAS DE FABRICACIÓN DE PLDs

Los PLDs abarcan un gran numero de circuitos con diferencias en su fabricación de acuerdo al tipo de dispositivo del que se trate. La celda básica que puede ser programada se realiza de acuerdo a diversas tecnologías que se analizan a continuación.

TECNOLOGÍA DE TRANSISTOR CMOS DE COMPUERTA FLOTANTE

Un transistor CMOS (Metal - Óxido Semiconductor Complementario) esta formado por la combinación de dos transistores MOS, uno de canal tipo P y otro tipo N[5], en un mismo sustrato. Es por esto que se les llama complementarios. A continuación se describe como funciona un transistor MOS de compuerta flotante del tipo N (fig. 1.3).

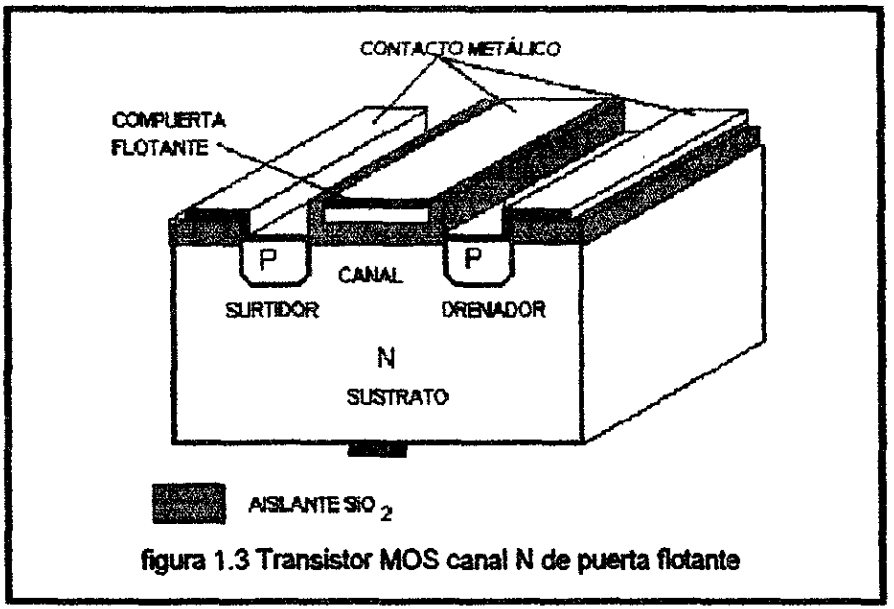


figura 1.3 Transistor MOS canal N de puerta flotante

El transistor MOS canal N consta de un substrato ligeramente dopado[6] de material de silicio de tipo N. Existen dos regiones muy dopadas por difusión con impurezas de tipo P para formar el surtidor y el drenador. La región entre las dos secciones P sirve como canal, la compuerta es una placa de metal separada del

[5] Canal P , positivo; canal N, negativo.

[6] En los semiconductores como el germanio y el silicio, con valencia 4, las moléculas se agrupan con un enlace covalente que hace imposible la transmisión de electrones y es necesario contaminarlos con otra sustancia cuyo número de valencia sea diferente de 4, a esta contaminación se le llama dopado.

canal por un dieléctrico aislado de dióxido de silicio (SiO_2). La compuerta flotante se encuentra entre la compuerta normal y el substrato. La compuerta externa se conecta a la línea de selección correspondiente del decodificador y la compuerta flotante actúa como elemento de almacenamiento de cargas. La célula de memoria se programa poniendo a cero (voltaje a tierra) su substrato y aplicando una tensión elevada (típica de 21 V) que provoca una inyección por avalancha de electrones los cuales quedan atrapados en la compuerta flotante. En su estado inicial (sin electrones en la compuerta), el transistor posee un voltaje de umbral[7] bajo que permite su activación cuando se aplican 5 V a la compuerta externa con respecto al surtidor. La presencia de electrones en la compuerta flotante eleva la tensión de umbral y hace que el transistor no se active aunque se apliquen 5V a la compuerta externa.

Las células de memoria se programan inyectando o no electrones en las correspondientes compuertas aisladas. Cuando se quiere cambiar el contenido del dispositivo programable se aplica luz ultravioleta al circuito para que los electrones atrapados en la compuerta flotante sean liberados y cada célula de memoria vuelva al estado inicial.

TECNOLOGÍA DE TRANSISTORES CMOS ELÉCTRICAMENTE BORRABLES

Este tipo de transistores tienen un funcionamiento similar a los descritos en el apartado anterior. La única diferencia es que pueden ser borrados eléctricamente y no requieren de luz ultravioleta. Los PLDs con tecnología de transistores CMOS eléctricamente borrables (EECMOS) están constituidos por transistores de compuerta flotante en los que se ha reducido el espesor de la capa aislante que separa la compuerta flotante de la compuerta externa y el drenador, de tal manera que tanto la inyección de electrones como su eliminación dentro de la compuerta flotante puede ser realizada únicamente con impulsos eléctricos. Para lograr un dispositivo confiable con este tipo de transistores es necesario que la célula básica de memoria este formada por dos transistores: uno de compuerta flotante y otro normal que se conectan en serie. La función del transistor normal es la de seleccionar el transistor que se borra o graba mediante impulsos eléctricos. En la figura 1.4 se muestra la célula básica de memoria con tecnología EECMOS.

El transistor de compuerta flotante esta en corte si su compuerta no posee electrones y esta saturado en caso contrario. La grabación consiste en inyectar dichos electrones mediante la aplicación de una tensión elevada a la compuerta flotante con respecto al surtidor, para lo cual es necesario que el transistor normal conduzca. El borrado del PLD se realiza aplicando una tensión en sentido contrario. Cuando se lee el circuito si un transistor de compuerta flotante esta grabado a la salida aparece un cero, en caso contrario aparece un uno.

---

[7] Voltaje mínimo de operación del transistor.

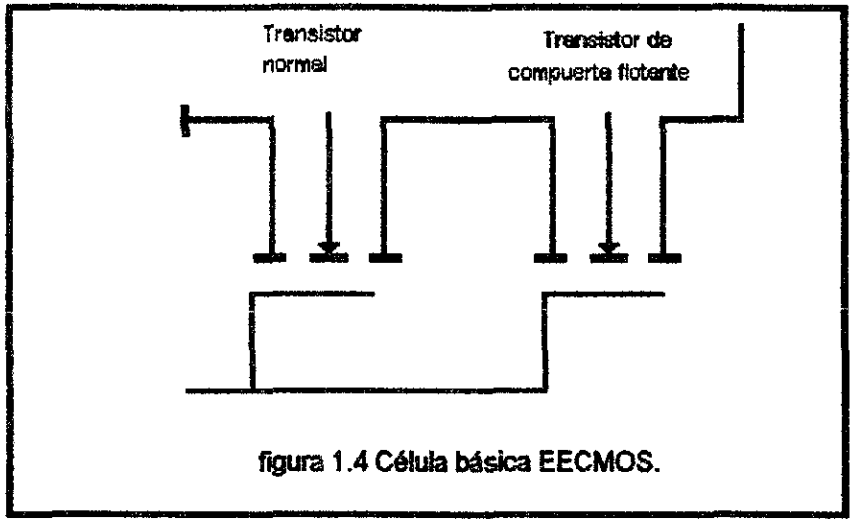


figura 1.4 Célula básica EECMOS.

TECNOLOGÍA DE FUSIBLE BIPOLAR

El fusible bipolar de Titanio -Tungsteno (TiW) es usado como elemento básico de programación en algunos PLDs. Esta formado por una banda de Titanio - tungsteno sobre la cual se encuentra una capa de aluminio (Al) y de dióxido de silicio (SiO_2), como se muestra en la figura 1.5 El tamaño ideal del fusible de Titanio - tungsteno es de alrededor de 500 angstroms. El TiW funciona también como una barrera para prevenir el contacto directo del aluminio con el silicio. Para prevenir la difusión del aluminio durante el proceso de alta temperatura, el grosor de la barrera de metal es de alrededor de 2000 Angstroms.

Para programar un fusible bipolar se hace pasar por el una temperatura de aproximadamente 2 100° C. La migración del metal que resulta del calentamiento hace que se abra. Simultáneamente el TiW se oxida a ambos lados y el dióxido se silicio ocupa el lugar que anteriormente ocupaba este (figura 1.6).

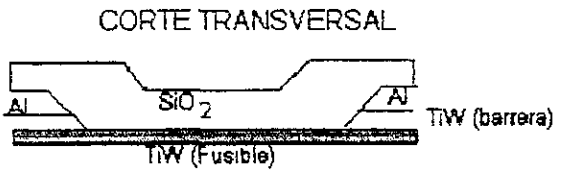
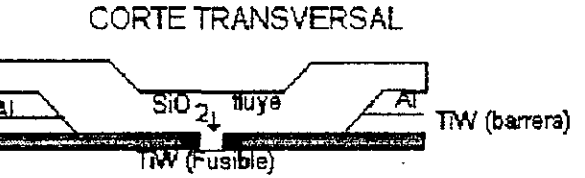


figura 1.5 Fusible de Titanio-Tungsteno sin programar



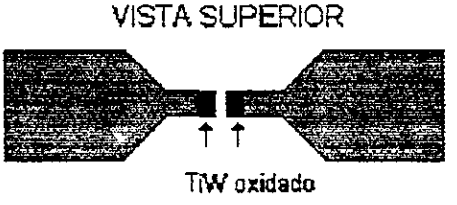


figura 1.6 Fusible de TiW después de la programación

I.5 ARQUITECTURA DE LOS DISPOSITIVOS PROGRAMABLES PLDs Y FPGAs

Existen varios tipos de PLDs que se agrupan de acuerdo con sus características internas, a continuación presentamos una clasificación de PLDs y su correspondiente diagrama interno básico.

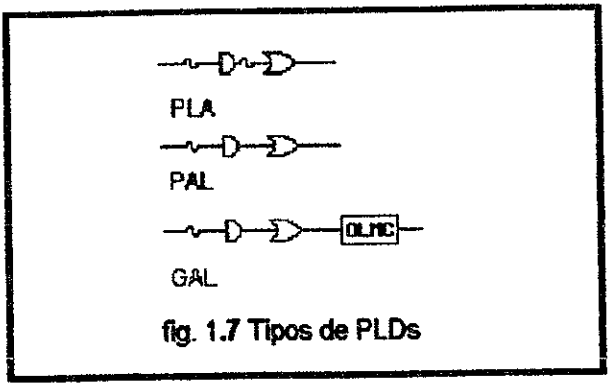


fig. 1.7 Tipos de PLDs

LÓGICA DE ARREGLO PROGRAMABLE (PLA)

Este tipo de circuito está constituido internamente por dos matrices programables, una de compuertas AND y otra de OR. La matriz AND es equivalente al decodificador de una memoria programable, solo que el número total de minitérminos generados es menor que 2^n, donde n representa el numero de entradas. Como no se generan todas las combinaciones posibles se dice que el PLA es un sistema programable incompleto. La matriz OR tiene la función de realizar las sumas de minitérminos necesarias para una función lógica determinada.

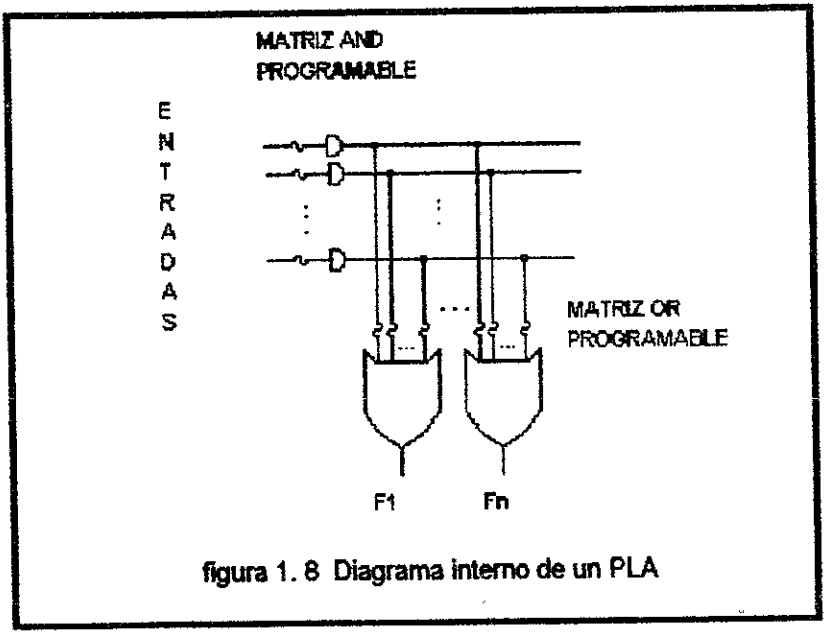


figura 1. 8 Diagrama interno de un PLA

El PLA es muy flexible ya que permite implementar funciones lógicas utilizando solo los minitérminos indispensables y por medio de la matriz OR programable se puede utilizar el mismo minitérmino en mas de una función, optimizando los recursos disponibles. Un PLA es útil para diseños en los cuales las funciones de salida solo toman el valor de uno lógico para algunas de las combinaciones posibles de las variables de entrada, las otras combinaciones son ceros o no importa. El PLA se vuelve mas eficiente cuando existe una gran interdependencia entre las funciones de salida, es decir, que varios minitérminos se utilizan en más de una función.

ARREGLO LÓGICO PROGRAMABLE (PAL)

Los PAL están constituidos internamente por una matriz de compuertas AND programable y una matriz de compuertas OR fija. Desde el punto de vista funcional es menos flexible que un PLA, ya que si existen minitérminos que se utilizan en dos o más funciones estos se deben programar en la matriz AND tantas veces como sea necesario. Al igual que el PLA se trata de dispositivos de lógica incompleta que no generan todas las combinaciones posibles con n variables de entrada.

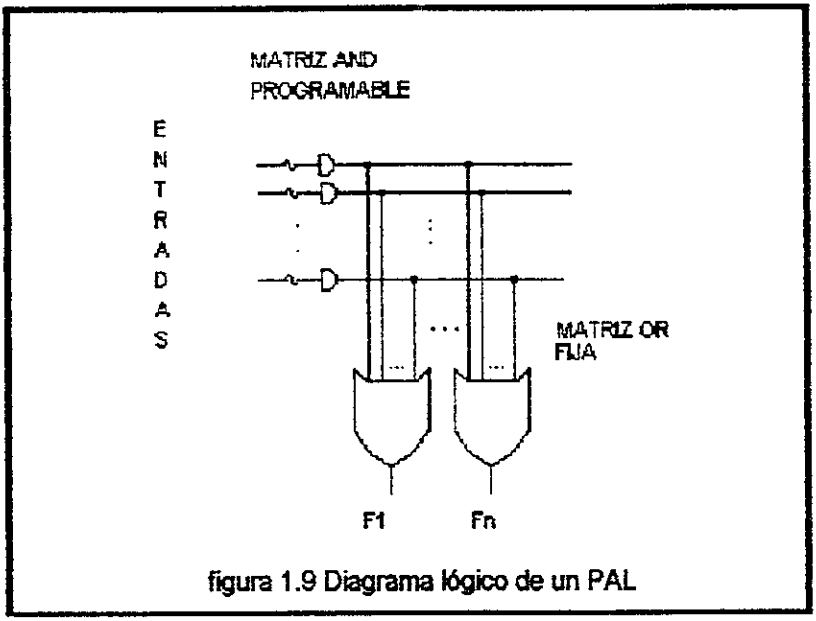


figura 1.9 Diagrama lógico de un PAL

Existen varios tipos de PAL, los hay para implementar aplicaciones puramente combinacionales y también con flip-flops que permiten el uso de lógica secuencial, son muy utilizados y permiten crear diseños que no existen en dispositivos estándar. Son recomendables en los casos en que solo se utilizan algunas de las 2^n posibles combinaciones con n variables de entrada. Un PAL puede ser grabado una sola vez, debido a la tecnología con la que están fabricados.

ARREGLO LÓGICO GENÉRICO (GAL)

Debido a que existen muchos tipos de PAL se pensó en tener un dispositivo que pudiera sustituir a varios de ellos. El GAL tiene, al igual que un PAL, una matriz AND programable, una matriz OR fija y además una Macrocelda Lógica de Salida (OLMC), la cual se explica detalladamente en el apéndice B. La OLMC hace posible que el pin al que esta conectada pueda funcionar como entrada, salida combinacional, salida de registro o salida triestado. Un GAL de 20 pines puede sustituir a cualquier PAL de 20 pines.

Los dispositivos GAL están fabricados con tecnología CMOS eléctricamente borrable. Gracias a esto es posible utilizar un GAL en una configuración determinada y posteriormente volver a reconfigurarlo si fuera necesario.

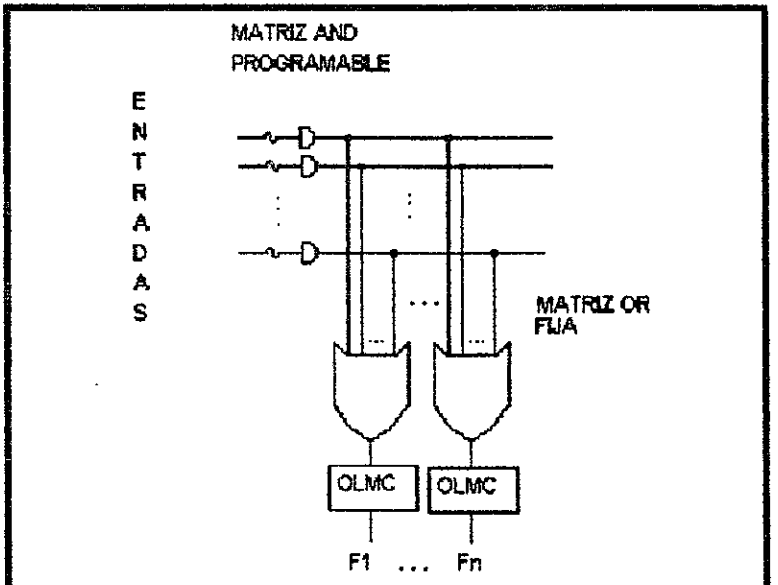


figura 1.10 Diagrama lógico de un GAL

Los GALs son útiles para el desarrollo de prototipos en los cuales se están haciendo cambios constantes y cuando no se encuentra el PAL apropiado siempre que las aplicaciones no requieran de todas las combinaciones de las variables de entrada, ya que también son dispositivos de lógica incompleta. En este trabajo la mayoría de los diseños fueron grabados en circuitos GAL.

FPGAs (Arreglo de Compuertas Programables en Campo)

Los FPGAs son circuitos que contienen múltiples arreglos de bloques lógicos en un mismo circuitos, lo que permite que en un solo chip se tenga el equivalente a varios dispositivos PLA, PAL o GAL.

FAMILIA MAPL

La familia MAPL (Lógica Múltiple de Arreglo Programable) agrupa circuitos integrados de alta densidad construidas con tecnología EECMOS, lo que hace que puedan ser reconfigurados varias veces solo con impulsos eléctricos. Un circuito MAPL contiene varios PLAs pero solo se utiliza una parte de la lógica en un tiempo determinado. Lo anterior se logra particionando lógicamente el dispositivo en páginas, de tal manera que solo la página actual esta activa en un tiempo determinando; las otras se encontrarán en estado inactivo.

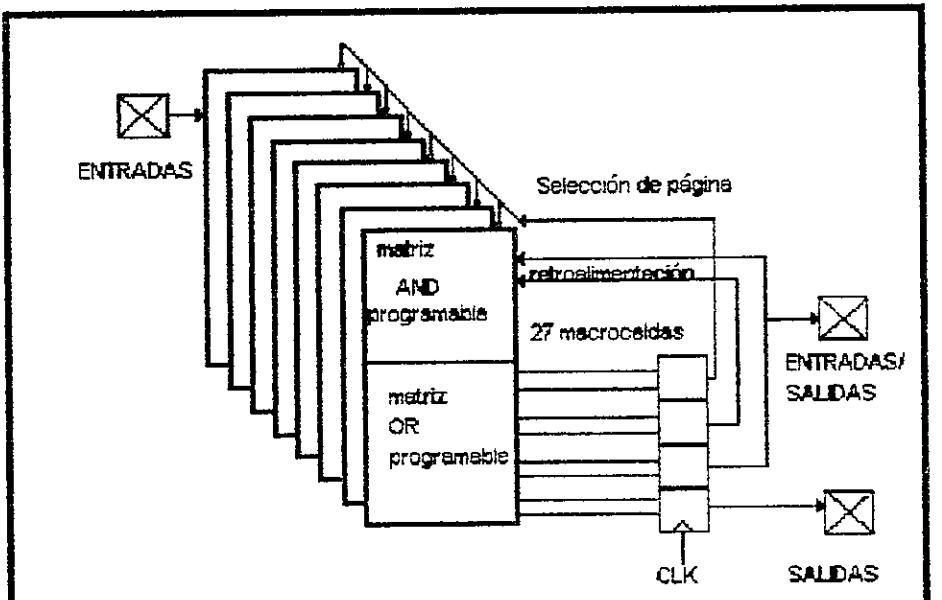


figura 1.11 Diagrama a bloques de un MAPL

La familia MAPL es compatible para ser acoplada con otros PLDs, al igual que los GALs cuenta con Macroceldas Lógicas de Salida (OLMC) que pueden ser programadas en distintos modos según el tipo de salida requerida.

FAMILIAS MACH

Las familias MACH (Macro arreglo CMOS de Alta Densidad) son también PLDs de alta densidad, internamente contienen varios bloques PAL. Están hechos con tecnología CMOS eléctricamente borrable (EECMOS). Este tipo de arquitectura puede sustituir a varios dispositivos PAL ahorrando espacio, energía y tiempo en el diseño e implementación del proyecto a desarrollar.

Un circuito MACH consta de bloques PAL interconectados por medio de una matriz de switchs programables. A través de la matriz se da la comunicación entre los distintos bloques, así como la ruta de una entrada a uno de ellos. Cada bloque PAL puede consta de un arreglo de minitérminos, un localizador lógico, macroceldas y celdas de entrada/salida. El arreglo de minitérminos constituye la lógica básica. El numero de minitérminos que pueden ser conectado a una macrocelda es variable. La función de el localizador lógico es distribuir los minitérminos entre las macroceldas requeridas para el diseño. En el capítulo seis se utilizan circuitos MACH 215 para realizar un controlador y en el apéndice C se explican con mayor detalle las características de este tipo de circuitos.

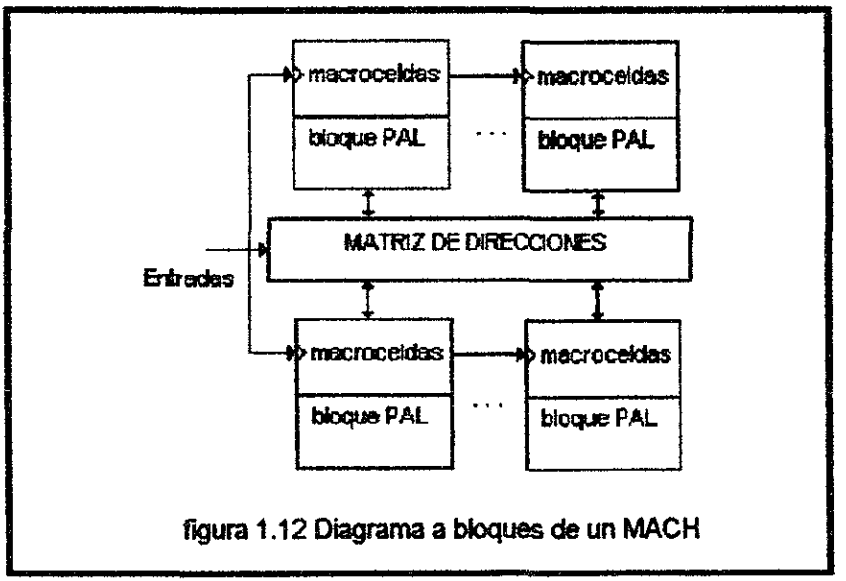


figura 1.12 Diagrama a bloques de un MACH

I.6 HARDWARE DE PROGRAMACIÓN

El diseño e implementación de PLDs requiere de dos herramientas básicas: el software y el hardware. El hardware esta constituido por el propio circuito que será utilizado y además de un dispositivo por medio del cual se pueda grabar la configuración interna de los fusibles. A este dispositivo se le llama programador, contiene un manejador electrónico de pines y un circuito de control; se conecta a la computadora en el puerto serie o paralelo. Las instrucciones de control , grabación o borrado son suministradas por la computadora de acuerdo al software que se este utilizando. La mayoría de los PLDs pueden ser programados en un socket montado directamente en el programador y otros requieren de algún tipo de adaptador.

Dependiendo de los distintos tipos de circuitos que es capaz de grabar o borrar se puede decir que un programador es universal o específico. Los programadores específicos solo pueden grabar algún tipo de circuito programable, mientras que los universales graban prácticamente todos los tipos de PLDs existentes en el mercado, siempre y cuando se cuente con el software adecuado.

El nombre de los circuitos lógicos programables define sus características principales, estas varían un poco dependiendo de la familia de que se trate y de la empresa que los fabrica. En la figura 1.13 se presenta una nomenclatura para las familias MACH y en la figura 1.14 otra más general que abarca a las familias de circuitos PAL, GAL, y MAPL.

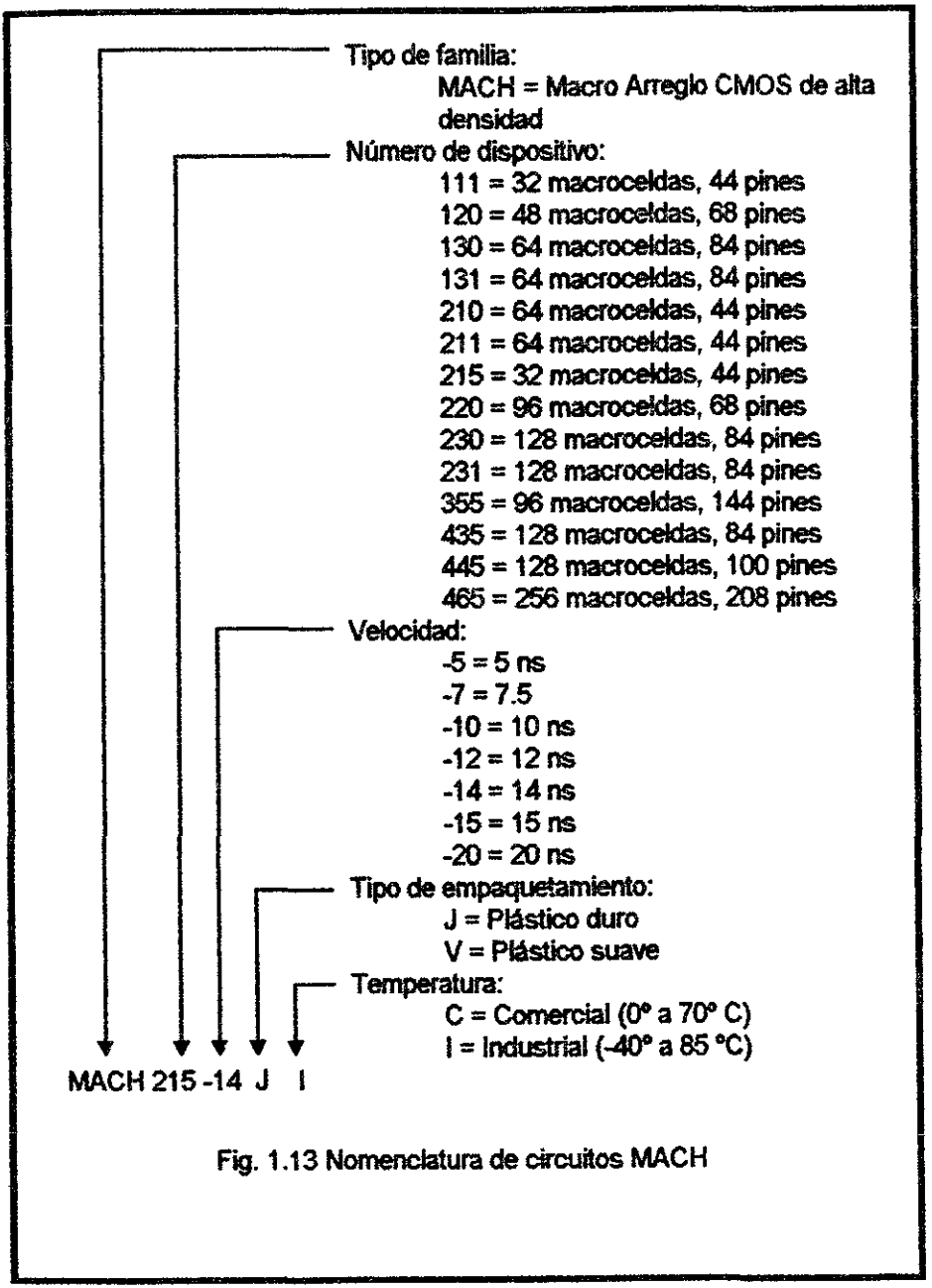


Fig. 1.13 Nomenclatura de circuitos MACH

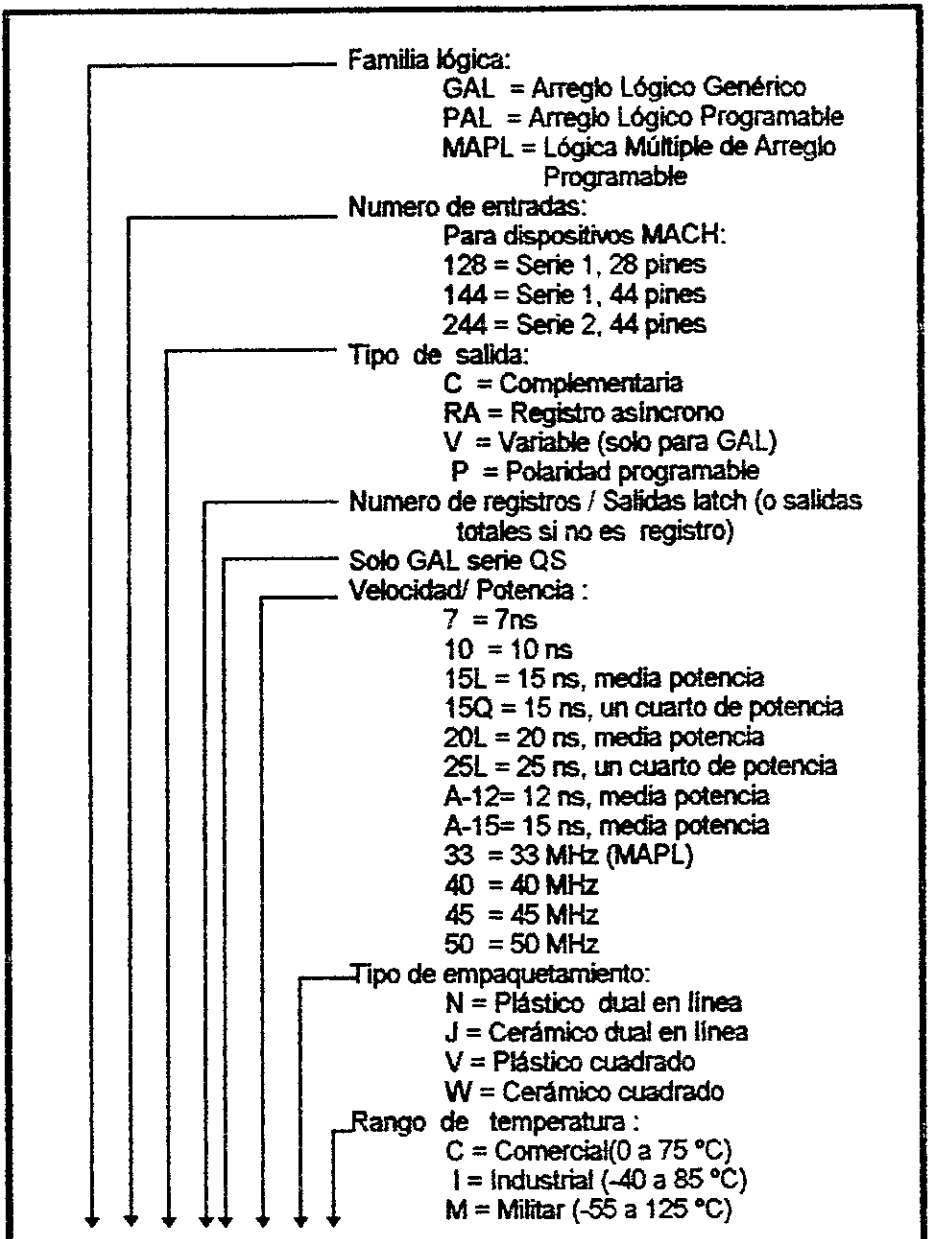


fig. 1.14 Nomenclatura de PAL, GAL Y MAPL

I.7 SOFTWARE DE PROGRAMACIÓN

Existen varios paquetes de programación que facilitan el diseño de aplicaciones con PLDs. Generalmente contienen un editor en el que se escribe el programa que permitirá grabar la configuración interna del circuito, el cual se compila y posteriormente se realiza la simulación. El software utilizado entrega como resultado final un archivo que contiene el mapa de fusibles internos del PLD. Este archivo se llama archivo JEDEC, comúnmente se utiliza la extensión .JED. En algunos de los casos el programa puede generar las ecuaciones lógicas a partir de la tabla de verdad o del diagrama de estados, lo que constituye una gran ayuda para el diseñador.

Los paquetes para programación de PLDs pueden contener una gran variedad de opciones como son el manejo de las ecuaciones booleanas y reducción automática de las funciones. La mayoría de ellos pueden programar a los PLDs mas comunes y algunos se ocupan de circuitos específicos. A continuación se exponen brevemente las características principales de algunos de ellos.

OPAL

Fue desarrollado por **National Semiconductors**, puede usar máquinas de estado[8], tablas de verdad, y ecuaciones booleanas. Puede realizar la optimización, verificación, y simulación de una gran variedad de PLDs como PALs, GALs y MAPLs. Utiliza un menú amigable al usuario, realiza conversiones de PALs a GALs, lo que ahorra tiempo en el diseño.

Consta de un ambiente gráfico, módulos ejecutables, un simulador de gráficos, una librería de dispositivos , ejemplos y una demostración del paquete.

OPAL jr

Es otro paquete desarrollado por **National Semiconductors**. Es totalmente compatible con OPAL y puede programar dispositivos PAL y GAL. **OPALjr** realiza el archivo JEDEC a partir de las ecuaciones booleanas del diseño. También incluye un modulo de conversión de PAL a GAL a partir del archivo JEDEC.

El hardware necesario para ejecutar **OPAL** y **OPALjr** es una computadora PC del tipo AT o XT con al menos 350 Kb de memoria RAM disponibles, un monitor del tipo VGA o EGA (en caso de usar monitor CGA no se podrán ver en pantalla los diagramas de tiempo[9]). Sistema operativo **MS-DOS** versión 2.1 o mayor y mouse con su correspondiente software.

---

[8] Las máquinas de estado se utilizan en la descripción de circuitos secuenciales, para una información detallada ver el capítulo seis.

[9] Diagramas de trenes de pulsos que muestran el comportamiento de un circuito secuencial.

Design Center/AMD

Fue creado por **MicroSim**, el creador de **PSpice**[10]. Este paquete ofrece una captura esquemática interactiva, así como la simulación del diseño. Esta disponible en dos versiones: una de ellas sin simulación síncrona ni editor gráfico para entrada de vectores de prueba; y una versión avanzada que si cuenta con estas opciones,. Permite programar dispositivos PAL y MACH.

Para ejecutar este paquete se recomienda una computadora PC 486 con al menos 8 Mb de memoria RAM, 20 Mb de espacio disponible en disco duro, sistema operativo **MS - DOS** 3.0 o mayor, **Windows** versión 3.1 o superior.

AMD/Synario

Es un software que permite el diseño digital utilizando circuitos PAL y MACH en un ambiente totalmente integrado a **Windows**, realiza diseños jerárquicos y la simulación del diseño. Ofrece dos modos de entrada: descripciones esquemáticas y de comportamiento. El hardware que se recomienda para utilizarlo es una computadora PC 386 o 486 con un mínimo de 8 Mb de memoria RAM (es recomendable 16 Mb), 20 Mb disponibles en disco duro, **MS-DOS** versión 5.0 o mayor, **Windows** versión 3.1 o mayor.

ABEL

El paquete **ABEL** (Lenguaje Avanzado de Expresiones Booleanas) es un ejemplo de software de desarrollo de PLDs de segunda generación. Fue desarrollado por la empresa **Data I/O Corporation** desde su primera versión, la cual programaba muchos de los PLDs existentes realizando la reducción lógica, simulación y generación de la documentación del diseño. **ABEL** se convirtió en una de las herramientas estándar de software para programación de PLDs. Las características principales de la versión mas reciente de **ABEL** son: Simulación para dispositivos asíncronos y macroceldas, soporte en sintaxis de múltiples retroalimentaciones, librería de dispositivos específicos, macros , funciones y conversión de archivos JEDEC a **ABEL** para recuperar diseños no documentados.

El software **ABEL** está disponible para computadoras del tipo **IBM** PC. En esta tesis se utilizó la versión 5 de **ABEL** - EDU cuyas características se describen en detalle en los capítulos 2 y 3, así como en el apéndice A

PALASM2

Es un paquete que permite únicamente la programación de circuitos PAL , fue desarrollado por **AMD** (Advanced Micro Device). Permite la simulación y verificación del diseño a partir del programa fuente, el cual puede ser escrito en

[10] Uno de los simuladores más utilizados en el desarrollo de microelectrónica.

cualquier editor que genere código ASCII. Genera el archivo JEDEC con el mapa de fusibles internos del circuito a programar.

ATGEN

Desarrollado por **ACUGEN Software INC.**, genera vectores de prueba para PLDs CMOS y ECL. Este software realiza simulaciones efectivas para prácticamente todos los diseños de PLDs incluyendo aquellos que contienen retroalimentación, máquina de estados, memoria interna y entrada/salida bidireccional.

CUPL

Creado por **Logical Devices Inc.**, es un compilador de alto nivel similar a **ABEL**. Provee un numero de funciones que no se encuentran normalmente en los ensambladores estándar de PLDs. La sintaxis del programa fuente esta basada en el lenguaje de programación C. Sus características principales son: formato flexible, utilización de nombres simbólicos, macros , opción de entrada a la maquina de estados, documentación sobre la sintaxis, selección de la polaridad de salida. El uso de macros permite una considerable reducción de los datos de entrada, sobre todo en funciones complejas.

LOG/iC

Es una herramienta de software que permite la programación de dispositivos PAL y GAL . Utiliza el concepto HYPERPLD en el cual el usuario puede introducir, simular y procesar un diseño sin necesidad de especificar el dispositivo a utilizar.

Después de realizar el proceso de minimización la base de datos de circuitos puede ser utilizada para elegir el mejor de acuerdo con la aplicación. Este proceso de selección toma como base los resultados de la optimización HYPERPLD así como otros parámetros como velocidad, consumo de energía, empaquetamiento, etc.

PGADesigner

Es un paquete que permite el diseño utilizando PLDs y FPGAs. Utiliza un lenguaje de alto nivel para los archivos de entrada, realiza la simulación del diseño y muestra las formas de onda generadas. Puede seleccionar automáticamente el dispositivo a utilizar y entrega como salida un archivo JEDEC.

ViewPLD

Es un paquete que ofrece múltiples formatos de entrada , partición automática, adaptación de dispositivos, simulación lógica, análisis de las formas de onda. Los diseños de entrada son realizados por el formato de entrada de **ABEL** o por

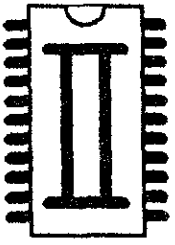
importación de archivos JEDEC. Se puede realizar una selección automática del dispositivo.

PLDesigner

Es un diseñador lógico universal desarrollado por la empresa **MINK INCORPORATED**. La principal diferencia entre PLDesigner y otros paquetes es que la fase de diseño esta separada de la selección del dispositivo. En este paquete es posible realizar un diseño antes de que el o los dispositivos con los que se va a trabajar sean seleccionados. Lo cual permite que el diseñador se concentre en el diseño y la simulación, con lo que se superan las limitaciones de utilizar un dispositivo especifico y elegirlo antes del diseño. Requiere de una computadora **IBM** PC con sistema operativo **MS-DOS** o **PC-DOS** versión 2.0 o mayor, un mínimo de 640 Kb de memoria RAM y disco duro. Se puede utilizar un monitor CGA, EGA o Hercules. El mouse y la impresora son opcionales.

ProLogic

Es un paquete de diseño lógico desarrollado por **Texas Instruments** para diseñar y programar PLDs. Es flexible y permite el diseño a partir de ecuaciones booleanas, tabla de verdad o diagramas de estado, puede realizar la simulación y genera un archivo JEDEC con el mapa de fusibles del dispositivo a programar.



CAPÍTULO II. INTRODUCCIÓN AL LENGUAJE AVANZADO DE EXPRESIONES BOOLEANAS

II.1 EL LENGUAJE ABEL

ABEL (Lenguaje Avanzado de Expresiones Booleanas) es un lenguaje para optimizar la descripción de circuitos que son implementados en PLDs o FPGAs. La versión 1.0 de este lenguaje fue introducida por **Data I/O Corporation** en 1984. Esta primera versión fue un intento sólido para la descripción de circuitos lógicos basados en PLDs, y la simplicidad del lenguaje original reflejaba la relativa sencillez de los dispositivos programables disponibles en ese tiempo. Al desarrollarse dispositivos programables más complejos **ABEL** fue actualizándose de acuerdo a ellos. En la versión 4 el lenguaje original fue sobrepasado significativamente, y actualmente puede ser usado para diseños de lógica de propósito general independientes de un dispositivo o tecnología determinados. Las características principales de la versión mas reciente (versión 5) de **ABEL** son: Simulación para dispositivos asíncronos y macroceldas, soporte en sintaxis de múltiples retroalimentaciones, librería de dispositivos específicos, macros y funciones, conversión de archivos JEDEC a **ABEL** para recuperar diseños no documentados.

El diseño de circuitos utilizando **ABEL** es similar en muchos aspectos al diseño de software utilizando lenguajes como **Pascal** o **C**. Los diseños de **ABEL** se introducen en un editor de texto y son compilados, optimizados y ejecutados (por medio de la simulación) antes de la implementación en hardware.

ABEL difiere de los lenguajes de programación en que es usado para describir funciones que trabajan en paralelo. Todos los enunciados en un diseño **ABEL** pueden ser ejecutados al mismo tiempo. Esta es una particularidad importante cuando se describen circuitos secuenciales.

ABEL, al igual que otros lenguajes de descripción de hardware (HDLs), provee diferentes formatos de entrada que pueden ser combinados como sea necesario según las especificaciones del sistema. Los métodos de descripción lógica disponibles en **ABEL** son: ecuaciones de alto nivel, tablas de verdad y diagramas de estado. En cada módulo de **ABEL** es usado uno o más de ellos para especificar completamente el diseño del circuito.

II.2 SINTAXIS BÁSICA

ABEL es un lenguaje de formato libre, de tal manera que la posición exacta de las palabras reservadas y otros elementos en el archivo fuente no es significativo para el programa compilador. Espacios, tabulaciones, saltos de línea y otros caracteres en blanco son ignorados, excepto cuando sirven para delimitar los distintos elementos de descripción del diseño. Las líneas en el archivo fuente no deben de exceder de 150 caracteres.

Las palabras reservadas, identificadores y números pueden ser separados por otros elementos de sintaxis como espacios en blanco (espacio, tabulador o salto de línea), separadores no alfanuméricos (coma, comillas, apóstrofe) o por un operador de expresión (paréntesis, corchetes).

PALABRAS RESERVADAS

ABEL cuenta con 27 palabras reservadas (ver tabla 1), las cuales se consideran parte del lenguaje y cumplen una función específica dentro de un programa, no deben ser usadas como identificadores en un archivo fuente y pueden ser escritas con letras mayúsculas, minúsculas o ambas. A lo largo de este trabajo las reservadas están escritas con letras negritas con el fin de diferenciarlas de otros elementos en los programas.

TABLA 1 PALABRAS RESERVADAS DE ABEL

DECLARATIONS	CASE	DEVICE	ELSE	ENABLE
ENCASE	END	EQUATIONS	ENDWITH	FUSES
FLAGS	IF	GOTO	LIBRARY	MACRO
MODULE	NODE	OPTIONS	PIN	STATE
STATE_DIAGRAM	THEN	TITLE	TRACE	WHEN
TRUTH_TABLE	WITH			

IDENTIFICADORES

Los identificadores son palabras creadas por el programador para dar un nombre único a los módulos, señales, constantes y argumentos que se utilizan en un programa. Al igual que las palabras reservadas, pueden ser escritos con letras mayúsculas, minúsculas o ambas; pero para el procesador será distinto un identificador escrito con mayúsculas a otro escrito con minúsculas. Por ejemplo, un identificador llamado *FILI* será distinto de otro llamado *fili*.

Un identificador está formado por una cadena de caracteres alfanuméricos que no deben exceder de 32 . Se pueden utilizar las letras mayúsculas de la "A" a la "Z", minúsculas de la "a" a la "z", los números del "0" al "9" y el símbolo "_" (subrayado). Los nombres de identificadores deben empezar siempre con una letra.

La unidad básica de diseño en **ABEL** es el módulo. Un diseño típico basado en PLDs consiste de un módulo simple, mientras que uno complejo basado en FPGAs puede contener varios módulos que se combinan durante el proceso de programación del dispositivo.

Un módulo **ABEL** consta de tres partes principales: declaraciones, descripciones lógicas y vectores de prueba que pueden aparecer en cualquier orden en el archivo fuente. Cuando un archivo fuente contiene varios módulos estos serán procesados por el compilador en el orden encontrado.

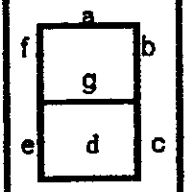
En la figura 2.1 se muestra la tabla de verdad de un codificador binario a siete segmentos, las salidas son activas en bajo, es decir, que para encender uno de los segmentos es necesario que en dicha salida exista un 0 lógico también se muestra la disposición de los siete segmentos en el display para cada combinación de entrada. En la figura 2.1a se puede observar el archivo fuente de **ABEL** correspondiente a este ejemplo y se marcan las tres secciones mencionadas.

II.3.1 DECLARACIONES

Las declaraciones son enunciados que se escriben generalmente al principio de un módulo (antes de cualquier sección de descripción lógica). Contienen información general del diseño que se está realizando: nombre del módulo, título del diseño, autor, etc.; además, como en todo lenguaje estructurado se deben declarar o definir las variables, constantes, señales y vectores[11] que serán utilizados en el resto del programa, en caso contrario el compilador no los reconocerá y marcará un error de sintaxis.

La palabra reservada **declarations** especifica el inicio de la sección de declaraciones, pero no existe error si se omite. La palabra reservada **module** se coloca al principio del programa seguida del nombre del módulo, se recomienda que este nombre sea mismo que el del archivo fuente. Sin embargo se les puede dar nombres diferentes sin que ello cause problemas en el diseño. El enunciado **title** es utilizado para escribir el título del circuito. Se puede incluir una breve descripción del dispositivo y algunos comentarios o el nombre del autor . Toda esta información se debe encerrar entre apóstrofes.

[11] En esta tesis un vector se refiere a un conjunto de señales lógicas que tienen alguna característica en común.



Display de siete segmentos

Entradas f1 f2 f3 f4	SALIDAS a b c d e f g	DISPLAY	Entradas f1 f2 f3 f4	SALIDAS a b c d e f g	DISPLAY
0 0 0 0	0 0 0 0 0 0 1		0 1 0 1	0 1 0 0 1 0 0	
0 0 0 1	1 0 0 1 1 1 1		0 1 1 0	0 1 0 0 0 0 0	
0 0 1 0	0 0 1 0 0 1 0		0 1 1 1	0 0 0 1 1 1 1	
0 0 1 1	0 0 0 0 1 1 0		1 0 0 0	0 0 0 0 0 0 0	
0 1 0 0	1 0 0 1 1 0 0		1 0 0 1	0 0 0 1 1 0 0	

figura 2.1 Tabla de verdad de un codificador binario a siete segmentos

DECLARACIONES

```
module bseven
title 'codificador de binario a siete segmentos salida activa en
bajo usando ecuaciones Celia Rosa Fierro Santillán y José
Filiberto Lule Flores';
f1,f2,f3,f4 pin;
a,b,c,d,e,f,g pin istype 'com';
ENTRADAS = [f1,f2,f3,f4];
SALIDAS = [a,b,c,d,e,f,g];
```

ECUACIONES

```
EQUATIONS
a = !f1&f2&!f3&!f4 # !f2&!f3&f4;
b = !f1&!f2&f3&!f4;
c = !f1&f2&!f3&f4 # !f1&f2&f3&!f4;
d = !f1&!f2&!f3&f4 # !f1&f2&!f4;
e = !f1&!f2&f4 # !f1&!f2&f3 # !f1&f3&f4;
f = !f1&f4 # !f1&f2&!f3;
g = !f1&!f2&!f3 # !f1&f2&f3&f4;
```

VECTORES DE PRUEBA

```
TEST_VECTORS (ENTRADAS -> SALIDAS)
[0, 0, 0, 0]->[0, 0, 0, 0, 0, 0, 1];
[0, 0, 0, 1]->[1, 0, 0, 1, 1, 1, 1];
[0, 0, 1, 0]->[0, 0,1, 0, 0, 1, 0];
[0, 0, 1, 1]->[0, 0, 0, 0, 1, 1, 0];
[0, 1, 0, 0]->[1, 0, 0, 1, 1, 0, 0];
[0, 1, 0, 1]->[0, 1, 0, 1, 0, 0, 0];
[0, 1, 1, 0]->[0, 1, 0, 0, 0, 0, 0];
[0, 1, 1, 1]->[0, 0, 0, 1, 1, 1, 1];
[1, 0, 0, 0]->[0, 0, 0, 0, 0, 0, 0];
[1, 0, 0, 1]->[0, 0, 0, 0, 1, 0, 0];
END
```

figura 2.1a Archivo fuente de un codificador binario a siete segmentos con salida activa en bajo.

DECLARACIONES DE DISPOSITIVOS

La declaración del dispositivo en el cual se grabará el programa es opcional. No es necesaria para la compilación y simulación de un diseño pero sí para generar archivo de mapa de fusibles. Si se utiliza se debe escribir antes que cualquier declaración de pines o nodos. Por ejemplo el siguiente enunciado es una declaración de un circuito GAL 16V8 (GAL con 8 pines de entrada y 8 pines de entrada/salida).

```
midisp DEVICE 'GAL16V8';
```

La palabra *midisp* es el nombre que tendrá el archivo de mapa de fusibles, este es asignado arbitrariamente por el diseñador. El declarar un dispositivo en un archivo fuente de **ABEL** no restringe el diseño a ese dispositivo particular, pero puede implicar ciertas características del diseño (como el tipo de registros o inversión de la señal de salida).

DECLARACIONES DE PINES Y NODOS

ABEL maneja dos tipos básicos de señales: pines y nodos, los cuales deben ser declarados antes de utilizarlos. Un pin representa una señal que es accesible fuera del modulo actual (al igual que el pin de un circuito), y un nodo es una señal interna del circuito que puede ser creada o removida durante el proceso de asignación de pines y nodos.

Los siguientes enunciados son ejemplos de declaraciones de pines y nodos:

```
i1,i2,i3 PIN;

Buf2,Buf1 NODE ISTYPE 'REG';
```

Se utilizan las palabras reservadas **pin** o **node** para diferenciar entre un pin o un nodo. En los ejemplos anteriores i1,i2 e i3 son nombres de pines y Buf1, Buf2 son nombres de nodos. La declaración puede incluir el número de pin o nodo. Cuando se especifican estos números son almacenados en el archivo intermedio generado por el compilador para utilizarlos en la grabación del dispositivo. Los números de pines no son importantes a menos que se haya declarado el dispositivo. Ejemplo de una declaración de número de pines:

```
D3, D2, D1, D0  PIN 2,3,4,5;
```

o también `D3 PIN 1; D2 PIN 3; D1 PIN 4; D0 PIN 5;`

En estos enunciados se especifica que la señal D3 será asignada al pin 1, D2 al 3, D1 al 4 y D0 al 5.

ENTRADAS Y SALIDAS

Las declaraciones de señales en **ABEL** no especifican el estado de cada una de ellas (entrada, salida o bidireccionales). Esta información está determinada por la forma en como se utilizan las señales dentro del programa. Se puede utilizar el enunciado **ISTYPE** para designar los atributos de cada señal, lo que ayuda a leer mejor el programa. Los distintos tipos de atributos de señales se muestran en la tabla 2. La descripción detallada de cada uno de ellos se incluye en el apéndice A.

TABLA 2 ATRIBUTOS DE SEÑALES

REG	REG_D	REG_T	REG_SR
REG_JK	REG_G	COM	BUFFER
INVERT	POS	NEG	DC
XOR			

II.3.2 DESCRIPCIONES LÓGICAS

ECUACIONES EN ABEL

La palabra reservada **equations** se utiliza para señalar el inicio de la sección de ecuaciones. En **ABEL** se pueden usar ecuaciones booleanas para describir el funcionamiento de un circuito, Los operadores utilizados por **ABEL** se muestran en la tabla 3.

TABLA 3 OPERADORES BOOLEANOS EN ABEL	
OPERADOR	DESCRIPCIÓN
=	ASIGNACIÓN
==	COMPARACIÓN
!	NOT
&	AND
#	OR
$	XOR
!$	XNOR

Consideraremos el ejemplo de la figura 2.1 (página 27). En este codificador tenemos cuatro señales de entrada: f1,f2,f3,f4, que nos dan un total de 16 combinaciones posibles (necesitamos 10 combinaciones para los dígitos del 0 al 9). También tenemos 7 salidas: a,b,c,d,e,f,g que corresponden con los siete segmentos de un display. Agrupamos las entradas y salidas en los vectores ENTRADAS y SALIDAS respectivamente. Esto facilita el trabajo al escribir los vectores de prueba. Las ecuaciones de este ejemplo están escritas de la siguiente forma:

```
a = !f1&f2&!f3&!f4 # !f2&!f3&f4;
b = !f1&!f2&f3&!f4;
c = !f1&f2&!f3&f4 # !f1&f2&f3&!f4;
d = !f1&!f2&!f3&f4 # !f1&f2&!f4;
e = !f1&!f2&f4 # !f1&!f2&f3 # !f1&f3&f4;
f = !f1&f4 # !f1&f2&!f3;
g = !f1&!f2&!f3 # !f1&f2&f3&f4;
```

Se escribe en primer lugar el nombre de la señal de salida seguida de un signo de igual y en la parte izquierda de la ecuación se describe la interacción de las señales de entrada por medio de los operadores booleanos, cada ecuación debe terminar con el símbolo ";".

A cada salida le corresponde una ecuación individual escrita en forma de suma de productos. Las cuales pueden ser mapeadas directamente en arreglo lógico de un

GAL. En caso necesario **ABEL** procesa las ecuaciones booleanas para minimizarlas. En este ejemplo las ecuaciones mínimas son:

```
a  = !( f1 & f2 # f3 # f2 & f4 # !f2 & !f4 );
b  = !( f1 # f2 # !f3 # f4 );
c  = !( f1 # !f2 # f3 & f4 # !f3 & !f4 );
d  = !( f1 # f2 & f4 # f3 & f4 # !f2 & !f4 );
e  = !( f1 # f2 & !f3 # f2 & !f4 # !f3 & !f4 );
f  = !( f1 # !f2 & !f4 # f3 & !f4 );
g  = !( f1 # !f2 & f3 # f2 & !f3 # f3 & !f4 );
```

Si comparamos las ecuaciones del archivo fuente con estas últimas observamos ligeros cambios debido a la reducción de la lógica que hace **ABEL**, pero el funcionamiento del circuito es el mismo.

TABLAS DE VERDAD

Las tablas de verdad son muy útiles para describir circuitos del tipo decodificadores, en los cuales la relación de las entradas a las salidas no sigue un patrón regular. Está compuesta por un encabezado que especifica el orden de las entradas y las salidas, seguido de vectores que muestran el estado de dichas señales . Un ejemplo es el que se muestra en la figura 2.2, que describe el mismo decodificador que la figura 1.1 pero a través de una tabla de verdad .

La tabla de verdad se señala con el encabezado **Truth_table** seguido de los nombres de las señales de entradas y salida, o en su caso, de los vectores que las representan encerrados entre paréntesis. Dichas señales se relacionan a través de los operadores de asignación -> y :> utilizados para diseños combinacionales y secuenciales respectivamente. Después se escriben todas las combinaciones de entradas al circuito y sus correspondientes salidas, representándolas con ceros y unos, cada renglón de la tabla de verdad representa un minitérmino o combinación de entrada y se debe terminar con ";".

Las tablas de verdad nos permiten describir de forma mas concisa el funcionamiento de un circuito. En el ejemplo las entradas fueron agrupadas en el vector ENTRADAS y las salidas en el vector SALIDAS, estos son los encabezados de la tabla de verdad. A continuación simplemente llenamos los vectores de la tabla de verdad con ceros y unos de acuerdo al comportamiento que se espera del circuito, también pueden ser utilizadas las constantes especiales .x. (condición de no importa) y .z. (alta impedancia en salida tri-estado).

DECLARACIONES

TABLA DE VERDAD

VECTORES DE PRUEBA

```
module bseven1
title 'codificador de binario a siete segmentos salida activa en
bajo usando tabla de verdad Celia Rosa Fierro Santillán y José
Filiberto Lule Flores';
f1,f2,f3,f4 pin;
a,b,c,d,e,f,g pin istype 'com';
ENTRADAS = [f1,f2,f3,f4];
SALIDAS = [a,b,c,d,e,f,g];
TRUTH_TABLE (ENTRADAS ->SALIDAS)
[ 0, 0, 0, 0] -> [0, 0, 0, 0, 0, 0, 1];
[ 0, 0, 0, 1] -> [1, 0, 0, 1, 1, 1, 1];
[ 0, 0, 1, 0] -> [0, 0, 1, 0, 0, 1, 0];
[ 0, 0, 1, 1] -> [0, 0, 0, 0, 1, 1, 0];
[ 0, 1, 0, 0] -> [1, 0, 0, 1, 1, 0, 0];
[ 0, 1, 0, 1] -> [0, 1, 0, 0, 1, 0, 0];
[ 0, 1, 1, 0] -> [0, 1, 0, 0, 0, 0, 0];
[ 0, 1, 1, 1] -> [0, 0, 0, 1, 1, 1, 1];
[ 1, 0, 0, 0] -> [0, 0, 0, 0, 0, 0, 0];
[ 1, 0, 0, 1] -> [0, 0, 0, 1, 1, 0, 0];
[ 1, 0, 1, 0] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.];
[ 1, 0, 1, 1] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.];
[ 1, 1, 0, 0] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.];
[ 1, 1, 0, 1] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.];
[ 1, 1, 1, 0] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.];
[ 1, 1, 1, 1] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.];
TEST_VECTORS (ENTRADAS -> SALIDAS)
[0, 0, 0, 0]->[0, 0, 0, 0, 0, 0, 1];
[0, 0, 0, 1]->[1, 0, 0, 1, 1, 1, 1];
[0, 0, 1, 0]->[0, 0, 1, 0, 0, 1, 0];
[0, 0, 1, 1]->[0, 0, 0, 0, 1, 1, 0];
[0, 1, 0, 0]->[1, 0, 0, 1, 1, 0, 0];
[0, 1, 0, 1]->[0, 1, 0, 0, 1, 0, 0];
[0, 1, 1, 0]->[0, 1, 0, 0, 0, 0, 0];
[0, 1, 1, 1]->[0, 0, 0, 1, 1, 1, 1];
[1, 0, 0, 0]->[0, 0, 0, 0, 0, 0, 0];
[1, 0, 0, 1]->[0, 0, 0, 1, 1, 0, 0];
END
```

figura 2.2 Programa fuente de un codificador binario a siete segmentos , utilizando tabla de verdad

DIAGRAMAS DE ESTADO

En electrónica digital un diagrama de estado es una representación gráfica que describe el funcionamiento de un circuito secuencial, pero en el caso particular de **ABEL** se considera como la descripción de los estados que puede tomar el diseño, cuales variables determinan la transición de un estado a otro y las salidas que se

generan en cada uno de ellos. Los diagramas de estado son esenciales para la descripción de circuitos secuenciales.

Los diagramas de estado se inician con la instrucción **State_diagram** seguida de los nombres de las variables de estado encerradas entre corchetes. Las tablas de verdad pueden ser utilizadas para describir máquinas de estado[12], particularmente aquellas que contienen un gran número de transiciones similares. Si al circuito codificador de los ejemplos anteriores se le agrega un contador se obtiene el diseño de la figura 1.3. Cada estado debe tener un nombre, en este ejemplo los 10 estados se designaron con letras del abecedario. También se asigna el valor esperado para cada una de las variables de salida. La transición de un estado a otro se hace por medio de la instrucción **if then else** correspondiente a un salto condicional. Por ejemplo, en el estado A tenemos:

```
If Clk then  B else A;
```

Si se cumple la condición de que la señal Clk esté en 1 lógico se pasará al estado B en caso contrario el circuito permanecerá en el estado A, manteniendo las señales de salida con el valor que se especifica en este. Así se pueden diseñar circuitos secuenciales fácilmente sin necesidad de resolver mapas de karnaught o especificar ecuaciones de transición de flip-flops.

	module bseven2 title 'codificador de binario a siete segmentos salida activa en bajo usando diagrama de estado Celia Rosa Fierro Santillán y José Filiberto Luie Flores'; Clk, OE pin "señal de reloj y señal de habilitación"; Q1, Q2, Q3, Q4 *node istype* 'reg'; a, b, c, d, e, f, g pin istype 'com'; A = 0; B = 1; C = 2; D = 3; E = 4 "valor de los estados"; F = 5; G = 6; H = 7; I = 8; J = 9 "valor de los estados"; ENTRADAS = [Clk]; SALIDAS = [[Q1,Q2,Q3,Q4],a,b,c,d,e,f,g]
ECUACIONES	EQUATIONS [Q1,Q2,Q3,Q4].clk = Clk; [Q1,Q2,Q3,Q4].oe = !OE;

DECLARACIONES

[12] Los términos "circuito secuencial" y "máquina de estado" son equivalentes.

DIAGRAMA DE ESTADO

```
STATE_DIAGRAM [Q1, Q2, Q3, Q4]
    state A:
        a=0; b=0; c=0; d=0; e=0; f=0; g=1;
        If Clk then B else A;
    state B:
        a=1; b=0; c=0; d=1; e=1; f=1; g=1;
        If Clk then C else B;
    state C:
        a=0; b=1; c=0; d=0; e=0; f=1; g=0;
        If Clk then D else C;
    state D:
        a=0; b=0; c=0; d=0; e=1 ; f=1; g=0;
        If Clk then E else D;
    state E:
        a=1; b=0; c=0; d=1; e=1; f=0; g=0;
        If Clk then F else E;
    state F:
        a=0; b=1; c=0; d=0; e=1; f=0; g=0;
        If Clk then G else F;
    state G:
        a=0; b=1; c=0; d=0; e=0; f=0; g=0;
        If Clk then H else G;
    state H:
        a=0; b=0; c=0; d=1; e=1; f=1; g=1;
        If Clk then I else H;
    state I:
        a=0; b=0; c=0; d=0; e=0; f=0; g=0;
        If Clk then J else I;
    state J:
        a=0; b=0; c=0; d=1; e=1; f=0; g=0;
        If Clk then A else J;
```

VECTORES DE PRUEBA

```
TEST_VECTORS ([Clk,OE]->[[Q1,Q2,Q3,Q4],a,b,c,d,e,f,g])
[.c.,0]->[B , 1, 0, 0, 1, 1, 1, 1];
[.c.,0]->[C , 0, 0, 1, 0, 0, 1, 0];
[.c.,0]->[D , 0, 0, 0, 0, 1, 1, 0];
[.c.,0]->[E , 1, 0, 0, 1, 1, 0, 0];
[.c.,0]->[F , 0, 1, 0, 0, 1, 0, 0];
[.c.,0]->[G , 0, 1, 0, 0, 0, 0, 0];
[.c.,0]->[H , 0, 0, 0, 1, 1, 1, 1];
[.c.,0]->[I  , 0, 0, 0, 0, 0, 0, 0];
[.c.,0]->[J  , 0, 0, 0, 0, 1, 1, 0];
[.c.,0]->[A  , 0, 0, 0, 0, 0, 0, 1];
END
```

figura 2.3 Programa en **ABEL** para un circuito contador y codificador a siete segmentos con salida activa en bajo, utilizando diagrama de estado.

II.4 METODOLOGÍA DE DISEÑO EN ABEL

En **ABEL** podemos programar circuitos lógicos que resuelvan problemas específicos, además, el lenguaje permite elegir entre los tres tipos de descripciones lógicas explicadas en el tema anterior y la programación es independiente del dispositivo o tecnología a utilizar. Sin embargo, se debe tener una metodología general de diseño, los pasos a seguir son los siguientes:

1. Plantear el problema a resolver.

2. Resolver el diseño con una tabla de verdad, ecuaciones booleanas o diagrama de estado.

3. Escribir el módulo correspondiente al diseño en el editor de **ABEL** o en cualquier otro editor.

4. Compilar el programa de acuerdo a estos pasos:

   - Compilar
   - Si existen errores de sintaxis en el programa el compilador enviará un listado de los errores.
   - Corregir errores.

Estos pasos se repiten hasta que el programa esté libre de errores.

5. Simulación de los vectores de prueba.

   - Si los resultados de la simulación son los esperados continuar.
   - Si los resultados de la simulación no son los esperados se debe corregir el programa y volver a compilar.

6. Especificar dispositivo en el cual se va a grabar el diseño por medio de la instrucción **device**.

7. Crear el archivo .JED el cual contiene el mapa de fusibles a grabar.

8. Programar el dispositivo.

Estos pasos se pueden visualizar mejor en el diagrama de flujo de la figura 2.4.

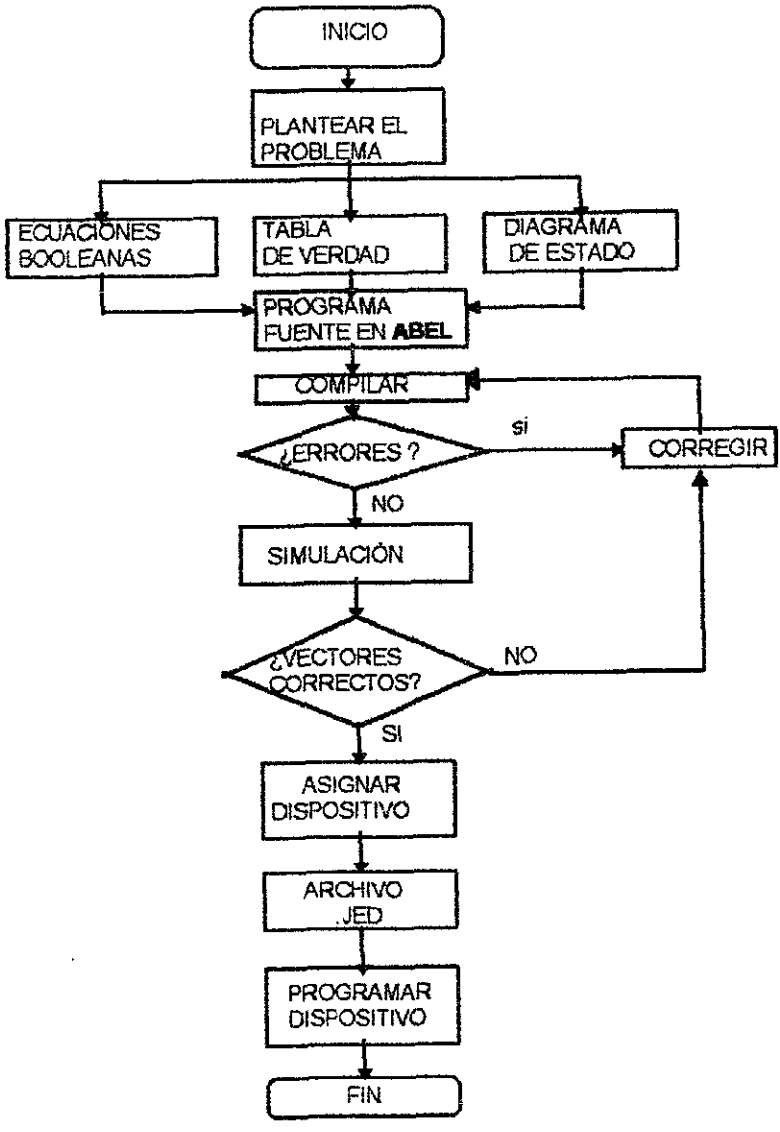


Figura 2.4 Diagrama de flujo de diseño de un dispositivo en ABEL

II.5 COMPILACIÓN Y SIMULACIÓN

Una vez que se ha escrito el programa fuente de un diseño en **ABEL** con ecuaciones booleanas, tablas de verdad o diagrama de estado; el siguiente paso es compilarlo por medio del comando *COMPILE*, si el programa fuente tiene errores, el compilador lo indicará. Este proceso genera varios archivos con el mismo nombre que el programa fuente, pero con distinta extensión. El contenido de estos archivos se puede visualizar por medio del submenú del comando *VIEW*, como se muestra en la tabla 4.

TABLA 4 ARCHIVOS GENERADOS EN LA COMPILACIÓN

TIPO DE ARCHIVO	EXTENSIÓN	COMANDO
Lista de compilación	LST	Compiler listing
Ecuaciones compiladas	EQ1	Compiled equations
Ecuaciones optimizadas	EQ2	Optimized ecuations
Asignación de pines	.FIT	Fitter assignments
Reporte de el PLD	DOC	PLDmap report
Mapa de fusibles	JED	Jedec/ prom fuse file
Resultados de simulación	SM1	Simulation results

El programa fuente debe ser depurado hasta que este libre de errores de sintaxis, lo cual no significa que el diseño no tenga errores de lógica en su funcionamiento. Estos últimos se identifican por medio de la simulación del diseño. **ABEL** ofrece 5 formatos distintos para mostrar la simulación en pantalla, los cuales se seleccionan por medio de las opciones del subcomando *Trace options* del comando *Compile*, estas opciones son:

1. NO TRACE: Simulación sin trazado. El resultado es un mensaje de cuantos vectores de prueba se simularon y el total de los que cumplieron con los resultados esperados. A continuación se presenta el ejemplo de simulación del codificador de 7 segmentos con salida activa en bajo en este formato.

```
Simulate ABEL-EDU 5.02  Date: Wed Sep 10 17:16:42 1997
Fuse file: 'bseven2.tt1'  Vector file: 'bseven2.tmv'  Part: 'PLA'
codificador de binario a siete segmentos salida activa en bajo usando diagrama de
estado Celia Rosa Fierro Santillán y José  Filiberto Lule Flores
10 out of 10 vectors passed.
```

Se puede observar la fecha y la hora , el nombre de los archivos de ecuaciones y de vectores, así como el título que se escribió en el programa fuente. El resultado en el último renglón es que de los 10 vectores simulados todos obtuvieron un resultado satisfactorio.

2. PINS FORMAT: Formato de pines. En este formato **ABEL** representa el valor de a las entradas y salidas del circuito con una L (Low) para el estado lógico 0 o nivel bajo y una H (High) para el estado lógico 1 o nivel alto de voltaje. A continuación se muestra la simulación para el mismo codificador bajo este formato.

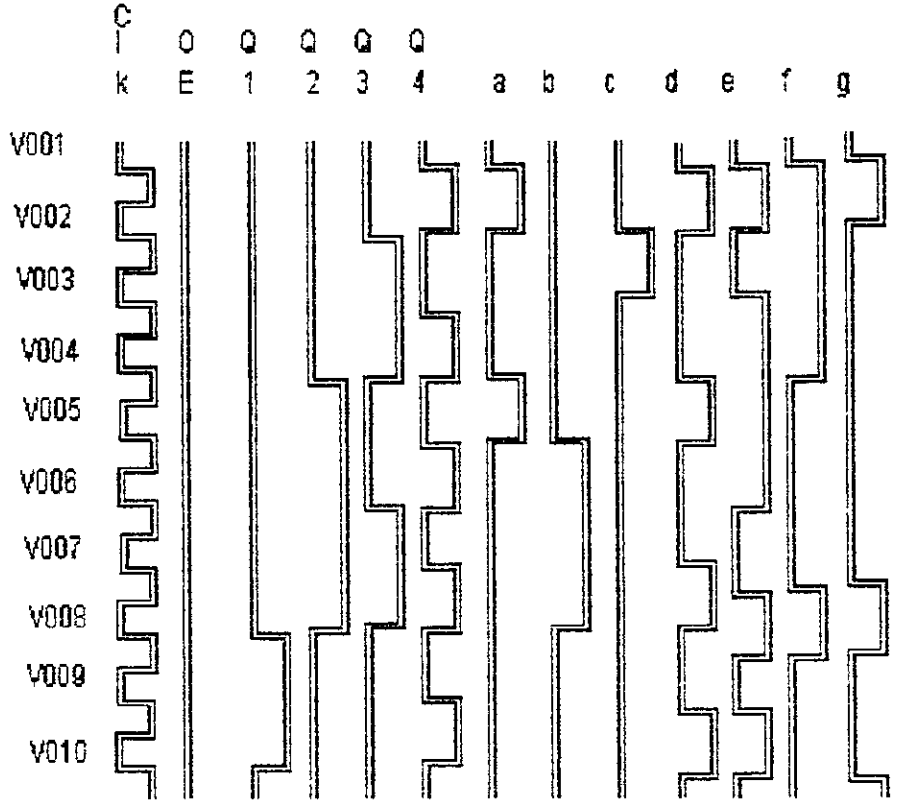
```
Simulate ABEL-EDU 5.02  Date: Wed Sep 10 17:23:35 1997
Fuse file: 'bseven2.tt1'  Vector file: 'bseven2.tmv'  Part: 'PLA'
codificador de binario a siete segmentos salida activa en bajo usando diagrama de
estado Celia Rosa Fierro Santillán y José  Filiberto Lule Flores

Vector 1
Vector In  [C0.....]
Device In  [00000110011110000011111000]
Device Out [.....HLLHHHHLLLLHHHHHLLL]
Vector 2
Vector In  [C0.....]
Device In  [00001000100100000011110100]
Device Out [.....LLHLLHLLLLLHHHHLHLL]
Vector 3
Vector In  [C0.....]
Device In  [00001100001100000011111100]
Device Out [.....LLLLHHLLLLLHHHHHHLL]
Vector 4
Vector In  [C0.....]
Device In  [00010010011000000011110010]
Device Out [.....HLLHHLLLLLLHHHHLLHL]
Vector 5
Vector In  [C0.....]
Device In  [00010101001000000011111010]
Device Out [.....LHLLHLLLLLLHHHHHLHL]
Vector 6
Vector In  [C0.....]
Device In  [00011001000000000011110110]
Device Out [.....LHLLLLLLLLLHHHHLHHL]
Vector 7
Vector In  [C0.....]
Device In  [00011100011110000011111110]
Device Out [.....LLLHHHHLLLLHHHHHHHL]
Vector 8
Vector In  [C0.....]
Device In  [00100000000000000011110001]
Device Out [.....LLLLLLLLLLLHHHHLLLH]
Vector 9
Vector In  [C0.....]
Device In  [00100100011000000011111001]
Device Out [.....LLLHHLLLLLLHHHHHLLH]
```

Vector 10
Vector In [C0.....]
Device In [00000000000010000011110000]
Device Out [.....LLLLLLHLLLLHHHHLLLL]

10 out of 10 vectors passed.

3. WAVE FORMAT: Formato de diagrama de tiempos. En este caso se muestra la gráfica de valores de las entradas y salidas de los vectores de prueba en forma de tren de pulsos . A continuación se muestra este formato para el ejemplo anterior.

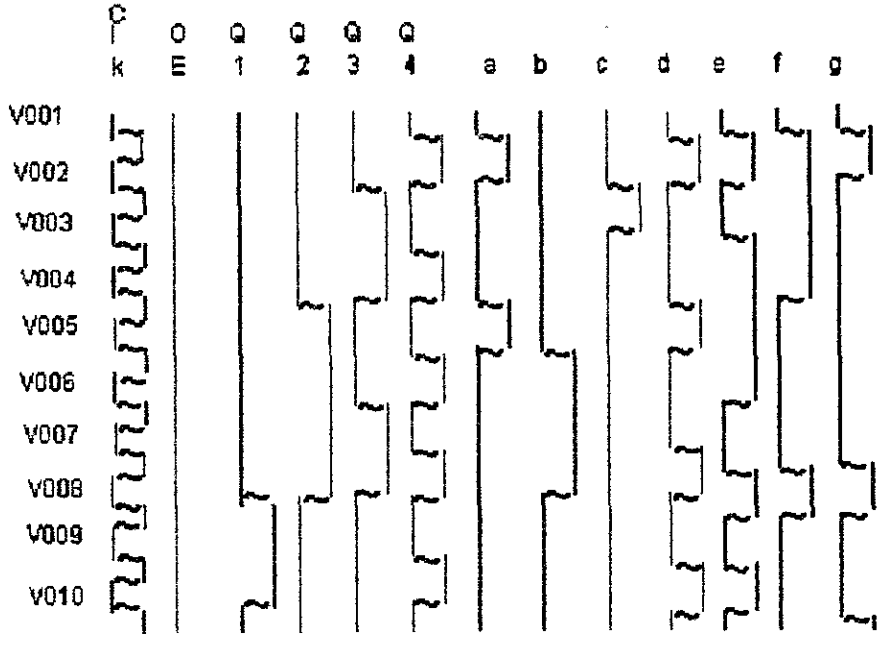
Simulate ABEL-EDU 5.02 Date: : Wed Sep 10 17:23:35 1997
Fuse file: 'bseven2.tt1' Vector file: 'bseven2.tmv' Part: 'PLA' codificador de binario a siete segmentos salida activa en bajo usando diagrama de estado Celia Rosa Fierro Santillán y José Filiberto Lule Flores



10 out of 10 vectors passed.

4. WAVE FORMAT ASCII: Este formato es similar al anterior, con la única diferencia de que los trenes de pulsos se construyen por medio de gráficos en código ASCII. A continuación se muestra el mismo ejemplo que en los casos anteriores pero bajo este formato .

Simulate ABEL-EDU 5.02 Date: Thu Dec 4 18:14:09 1997
Fuse file: 'bseven2.tt1' Vector file: 'bseven2.tmv' Part: 'PLA'
codificador de binario a siete segmentos salida activa en bajo usando diagrama de estado
Celia Rosa Fierro Santillán y José Filiberto Lule Flores



10 out of 10 vectors passed.

5. TABLE FORMAT: Formato de tabla de verdad. En este caso se muestra la tabla de verdad correspondiente a las entradas y salidas del circuito, señalando con L el 0 lógico y con H el 1 lógico. A continuación se muestra este formato de simulación para el ejemplo empleado.

Simulate ABEL-EDU 5.02 Date: Wed Sep 10 17:39:49 1997
Fuse file: 'bseven2.tt1' Vector file: 'bseven2.tmv' Part: 'PLA'
codificador de binario a siete segmentos salida activa en bajo usando diagrama de estado Celia Rosa Fierro Santillán y José Filiberto Luie Flores

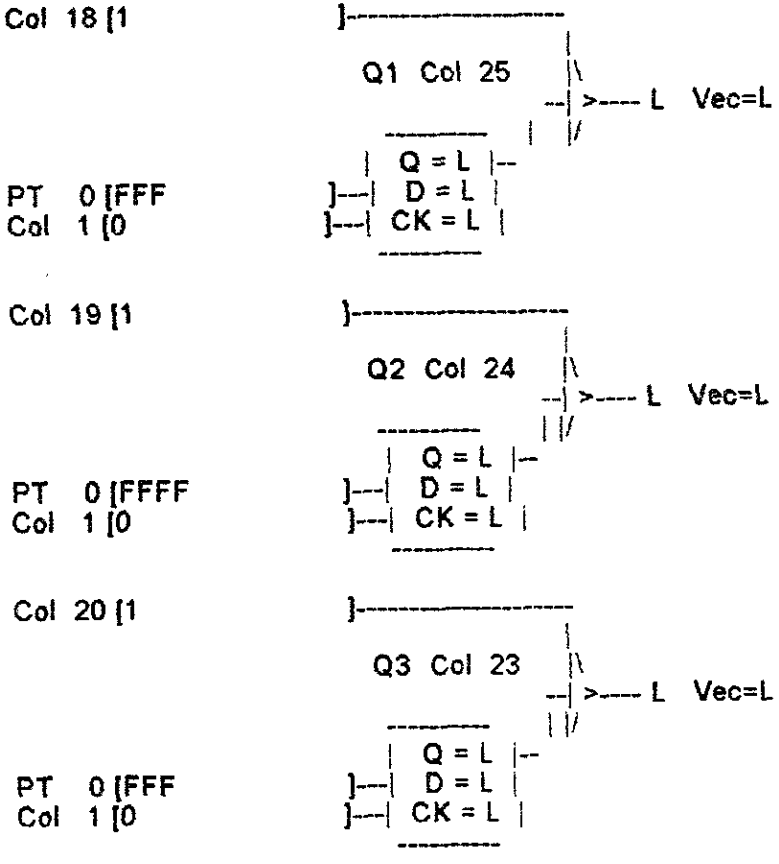
	C												
	l	O	Q	Q	Q	Q							
	k	E	1	2	3	4	a	b	c	d	e	f	g
V0001	0	0	L	L	L	L	L	L	L	L	L	L	H
	1	0	L	L	L	H	H	L	L	H	H	H	H
	0	0	L	L	L	H	H	L	L	H	H	H	H
V0002	0	0	L	L	L	H	H	L	L	H	H	H	H
	1	0	L	L	H	L	L	L	H	L	L	H	L
	0	0	L	L	H	L	L	L	H	L	L	H	L
V0003	0	0	L	L	H	L	L	L	H	L	L	H	L
	1	0	L	L	H	H	L	L	L	L	H	H	L
	0	0	L	L	H	H	L	L	L	L	H	H	L
V0004	0	0	L	L	H	H	L	L	L	L	H	H	L
	1	0	L	H	L	L	H	L	L	H	H	L	L
	0	0	L	H	L	L	H	L	L	H	H	L	L
V0005	0	0	L	H	L	L	H	L	L	H	H	L	L
	1	0	L	H	L	H	L	H	L	L	H	L	L
	0	0	L	H	L	H	L	H	L	L	H	L	L
V0006	0	0	L	H	L	H	L	H	L	L	H	L	L
	1	0	L	H	H	L	L	H	L	L	L	L	L
	0	0	L	H	H	L	L	H	L	L	L	L	L
V0007	0	0	L	H	H	L	L	H	L	L	L	L	L
	1	0	L	H	H	H	L	L	L	H	H	H	H
	0	0	L	H	H	H	L	L	L	H	H	H	H
V0008	0	0	L	H	H	H	L	L	L	H	H	H	H
	1	0	H	L	L	L	L	L	L	L	L	L	L
	0	0	H	L	L	L	L	L	L	L	L	L	L
V0009	0	0	H	L	L	L	L	L	L	L	L	L	L
	1	0	H	L	L	H	L	L	L	H	H	L	L
	0	0	H	L	L	H	L	L	L	H	H	L	L
V0010	0	0	H	L	L	H	L	L	L	H	H	L	L
	1	0	L	L	L	L	L	L	L	L	L	L	H
	0	0	L	L	L	L	L	L	L	L	L	L	H

10 out of 10 vectors passed.

6. MACRO - CELL FORMAT: Formato de macrocelda, el cual muestra el estado de cada macrocelda utilizada para generar las salidas de los vectores de prueba. A continuación se muestra este formato, en el que se observa el valor que toman los flip- flops, el reloj y las compuertas, todos son elementos internos a la macrocelda. En la simulación del primer vector, por ejemplo, se tiene un estado presente Q = 0 (L), un flip-flop D =0, una señal de reloj CK = 0.

Simulate ABEL-EDU 5.02 Date: Wed Sep 10 17:43:25 1997
Fuse file: 'bseven2.tt1' Vector file: 'bseven2.tmv' Part: 'PLA'
codificador de binario a siete segmentos salida activa en bajo usando diagrama de estado
Celia Rosa Fierro Santillán y José Filiberto Lule Flores
Simulate ABEL-EDU 5.02 Date: Thu Apr 16 16:39:26 1998
Fuse file: 'bseven2.tt1' Vector file: 'bseven2.tmv' Part: 'PLA'
codificador de binario a siete segmentos salida activa en bajo usando diagrama de estado Celia Rosa Fierro Santillan y Jose Filiberto Lule Flores

Vector 1
Vector In [C0.....]



10 out of 10 vectors passed.

II.6 ARCHIVO .JED

Una vez que se comprueba que los vectores de prueba son correctos y que el circuito realiza la función para la que fue diseñado se genera el archivo .JED, el cual contiene el mapa de fusibles que será grabado en el circuito GAL. Esto se hace por medio del comando *Compile* que genera automáticamente el archivo .JED con el nombre del dispositivo asignado en la sección de declaraciones del programa fuente. A continuación se incluye una parte del archivo bseven2.jed correspondiente al ejemplo manejado con anterioridad.

```
□ABEL-EDU 5.03.005 Data I/O Corp. JEDEC file for: MACH215A V
Created on: Wed Sep 10 17:54:44 1997
codificador de binario a siete segmentos salida activa en bajo usando diagrama de estado
Celia Rosa Fierro Santillán y José Filiberto Lule Flores
QP44* QF11936* QV10* F0*
X0*
NOTE Table of pin names and numbers*
NOTE PINS Clk:13 OE:33 Q1:2 Q4:3 Q2:4 Q3:5 a:6 b:7 c:8 d:9 e:21 f:20*
NOTE PINS g:19*
L0000 00000000000001000100000010110101010101011100110*
L0088 11111111111111111111111110111111111111111111111111*
L0264 11111111111111111111111111101111111110111011011110*
L0308 11111111111111111111111111110110111011101110111101*
L0352 11111111111111111111111111101110111011101110111111*
L0440 11111111111111111111111110111111111111111111111111*
L0616 11111111111111111111111111110110111111111111111110*
L0660 11111111111111111111111111111110111011101101101110*
L0704 11111111111111111111111111110110111011111111111101*
L0748 11111111111111111111111111111111110111011101101101*
L0792 11111111111111111111111111101111111111111111111111*
L0968 11111111111111111111111111110111111101111111111110*
L1012 11111111111111111111111111110111111101111101111111*
L1056 11111111111111111111111111110110111011101110111101*
L1100 11111111111111111111111111111110111011011111111111*
L1144 11111111111111111111111110111111111111111111111111*
L1320 11111111111111111111111111111110111111111101111110*
L1364 11111111111111111111111111111110110111111111101101*
```

El último paso del diseño es grabar el circuito GAL, por medio del subcomando *Program Device* del comando *Part Map*, El resultado es un circuito integrado que responde a necesidades específicas. El proceso de diseño, simulación y grabación de una GAL puede consumir desde unos minutos hasta algunas horas, lo cual reduce significativamente el tiempo de diseño y los costos.

CAPÍTULO III. EL SOFTWARE ABEL, SU DESCRIPCIÓN

III.1 CARACTERÍSTICAS DE ABEL

ABEL es un lenguaje de diseño en el que se pueden realizar programas con lógica compleja. El diseño completo puede ser compilado, simulado , mapeado y grabado dentro de un PLD utilizando únicamente **ABEL**. Para crear un diseño se puede usar el editor de texto provisto por el ambiente de diseño de este lenguaje o cualquier otro editor de texto que grabe los archivos en código ASCIII. Cuando el diseño está listo para procesarlo se procede a la simulación , lo que genera dos archivos, uno con los resultados de la simulación y otro con los datos en formato JEDEC para la programación del dispositivo. Este software es fácil de usar, y cuenta con ayuda en línea[13] con información detallada de las opciones de programación.

Características de la versión 5.0 de **ABEL**-EDU:

1. Soporte de memoria extendida.
2. Interfase compatible con Windows.
3. Ayuda en línea.
4. Reducción lógica.
5. Asignación automática de pines y nodos.
6. Simulación .

---

[13] En la parte inferior de la pantalla se muestran mensajes de ayuda acerca de los comandos.

III.2 AMBIENTE DE DISEÑO DE ABEL

El ambiente de diseño es el conjunto de programas que realizan las funciones de edición, compilación, simulación y programación de un archivo fuente en el que se describe un diseño de lógica digital. Para entrar al ambiente de diseño de **ABEL** desde MS-DOS basta con escribir **ABEL5** después del prompt de MSDOS. En la pantalla se presenta un mensaje referente a la licencia de uso de **ABEL** y posteriormente una pantalla como la siguiente:

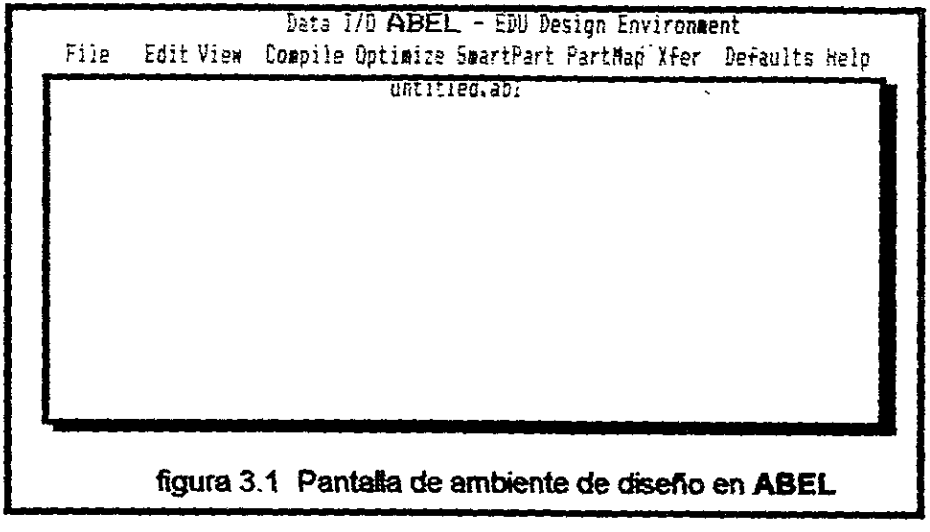


figura 3.1 Pantalla de ambiente de diseño en **ABEL**

En la parte superior se observa un mensaje referente a la empresa que produce el software **ABEL**, Data I/0, el segundo renglón lo ocupa la línea de comandos que contiene los 10 comandos principales. La mayor parte de la pantalla la ocupa la ventana de edición, donde se escriben los programas; en la parte superior de esta ventana se muestra el nombre del archivo que se está usando, cuando aún no se ha determinado un nombre para el archivo **ABEL** escribe untitled.abl (los archivos fuente tienen la extensión .ABL). La última línea es utilizada para desplegar mensajes de ayuda referentes al procedimiento que se esté llevando a cabo.

CUADRO DE DIALOGO

Un cuadro de diálogo es una ventana que contiene las distintas opciones del programa que se está ejecutando. La selección puede hacerse por medio de los botones de comandos. Las teclas de flechas se usan para moverse en las opciones de un cuadro de diálogo. Al presionar la barra espaciadora se marca una selección . También se puede seleccionar un comando presionando Enter.

Elementos en un cuadro de diálogo:

(*) Son usados para elegir una opción de varias posibilidades, la barra espaciadora permite seleccionar la opción.

[X] Selecciona una opción y la activa o desactiva utilizando la barra espaciadora cuando el cursor marca esa opción en particular.

<OK> Guarda las entradas hechas desde el cuadro de diálogo y regresa al menú principal.

TECLAS ESPECIALES

En el ambiente de diseño de **ABEL** algunas teclas tienen funciones especiales, las cuales describimos a continuación:

TECLAS DE PROPÓSITO GENERAL

<<F1>>	Ayuda en línea.
<<ESC>>	Alterna entre la ventana de edición y la barra de menús.
<<^L>>	Redibuja la pantalla.

TECLAS QUE SE UTILIZAN EN UNA LISTA

Espacio o Enter	Ejecuta la selección actual.
<<ESC>> o Enter	Ejecuta la selección desde la lista.
Home (dos veces)	Mueve el cursor hacia el primer punto desplegado.
End (dos veces)	Mueve el cursor hacia el ultimo punto desplegado.
Teclas de flechas	Mueven el cursor hacia arriba y abajo en una lista.
Pg Up	Avanza hacia atrás la lista.
Pg Dn	Avanza hacia adelante la lista.
Tab	Mueve el cursor al siguiente campo en un cuadro de diálogo.

TECLAS DE EDICIÓN

Pg Up	Desplaza el cursor una página hacia arriba.
Pg Dn	Desplaza el cursor una página hacia abajo.
Home	Mueve el cursor al primer caracter de la línea actual.
End	Mueve el cursor al último caracter de la línea actual.
^Home	Mueve el cursor al principio del archivo.
^End	Mueve el cursor al final del archivo.
^D	Borra una línea.
^R	Duplica una línea.
Ins	Alterna el modo de inserción en el editor
Del	Borra el caracter actual.
Enter	Inserta una línea.
Backspace	Borra el caracter que está antes del cursor.

TECLAS USADAS EN LOS MENÚS

<-	Selecciona el comando que se encuentra a la izquierda del actual.
->	Selecciona el comando que se encuentra a la derecha del actual.
flechas	Mueven la barra iluminada del menú en el sentido de la flecha.
Enter	Ejecuta la opción seleccionada en el menú actual.

TECLAS USADAS EN LOS CUADROS DE DIALOGO

Shift-Tab	Mueve el cursor al campo anterior en un cuadro de diálogo.
<<F2>>	Pasa hacia arriba una lista de opciones.
<<F5>>	Guarda las opciones seleccionadas y sale del cuadro de dialogo (lo mismo que <OK>).
ESC	Sale del cuadro de diálogo sin grabar los cambios.

III. 3 MENÚ FILE (ARCHIVO)

El menú archivo contiene varias opciones para crear nuevos diseños, abrir diseños existentes, grabar un diseño, etc. Al oprimir ALT - F se despliegan sus opciones en la pantalla (figura 3.2). A continuación se describen cada una de estas opciones:

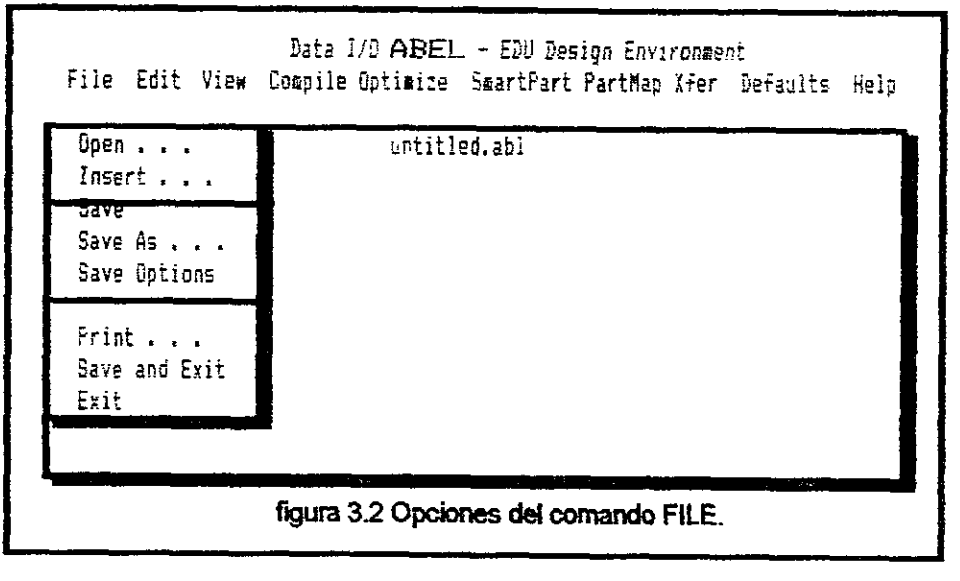


figura 3.2 Opciones del comando FILE.

New	Borra de la pantalla el archivo actual y comienza uno nuevo.
Open...	Abre un archivo de diseño ya existente (creando otro archivo con el mismo nombre y extensión .sav el cual contiene la misma información).
Insert...	Inserta un archivo en el actual.

Save	Graba en disco el archivo actual , lo mismo que sus opciones.
Save As...	Graba el archivo actual y sus opciones con un nuevo nombre.
Save Options	Graba únicamente las opciones de diseño.
Print...	Imprime un archivo.
<<OSSHELL>>	Temporalmente sale al sistema operativo, para regresar a **ABEL** se escribe EXIT en el prompt de MSDOS.
Save and Exit	Graba el archivo actual y sale del editor de **ABEL**.
Exit	Sale de el editor de **ABEL** sin grabar el archivo.

III.4 MENÚ EDIT (EDICIÓN)

Este menú contiene opciones para editar el archivo con el que se está trabajando. Las funciones disponibles son limitadas, por lo que se puede optar por utilizar otro editor más amigable al usuario, siempre y cuando grabe los archivos en formato ASCIII. Al oprimir la combinación de teclas ALT-E se despliegan las opciones de este menú en la pantalla (figura 3.3). A continuación se describen cada una de ellas:

Delete Line	Borra la línea actual en la ventana de edición.
Replicate Line	Copia la línea actual en la ventana de edición.
Search...	Busca el texto especificado en la ventana de edición.
Next	Encuentra la próxima ocurrencia del texto buscado.
Edit	Especifica el editor de texto (cuando el programa no se escribió en el editor de **ABEL**).
My Text Editor Is...	Invoca a el editor de texto en el archivo actual (cuando no se utiliza el editor de **ABEL**).
Repaint	Redibuja la pantalla.

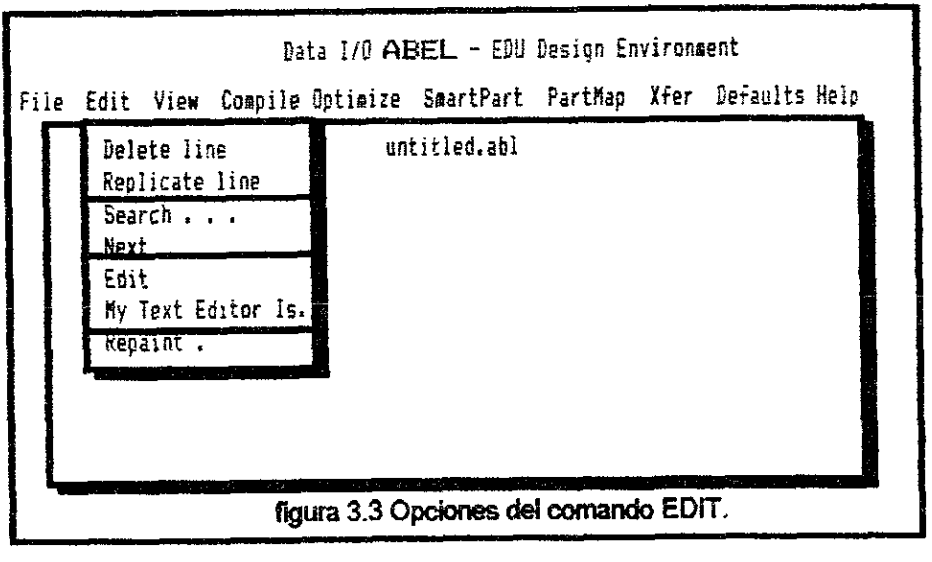


figura 3.3 Opciones del comando EDIT.

III.5 MENÚ VIEW (VER)

El *menú view* permite visualizar y examinar los reportes generados por un programa del ambiente de diseño **ABEL**. Si el reporte correspondiente es requerido y no existe, los programas apropiados serán ejecutados (cuando sea posible) para crear el archivo. Al oprimir ALT - V se despliegan estas opciones en pantalla (ver figura 3.4).

Compiler Listing	Ver la lista de compilación (*.lst).
Compiled Equations	Ver las ecuaciones generadas por el compilador(*.tt1).
Optimized Equations	Ver las ecuaciones optimizadas (*.tt2).
Fitted Equations	Ver las ecuaciones de asignación de pines (*.tt3).
Device Candidates	Ver la lista de posibles dispositivos(*.sel).
Fitter Assignments	Ver las asignaciones de pines (*.fit).
PLDmap Report	Ver el archivo de documentación(*.doc).
JEDEC/PROM Fuse File	Ver el archivo de fusibles (*.jed).
Simulation Results	Ver los resultados de la ultima simulación (*.sm? o *.sim).
PLDgrade Report	Ver los resultados del archivo JEDEC .
Errors	Ver los errores de la ultima ejecución del programa.
View File...	Ver cualquier archivo que se elija.

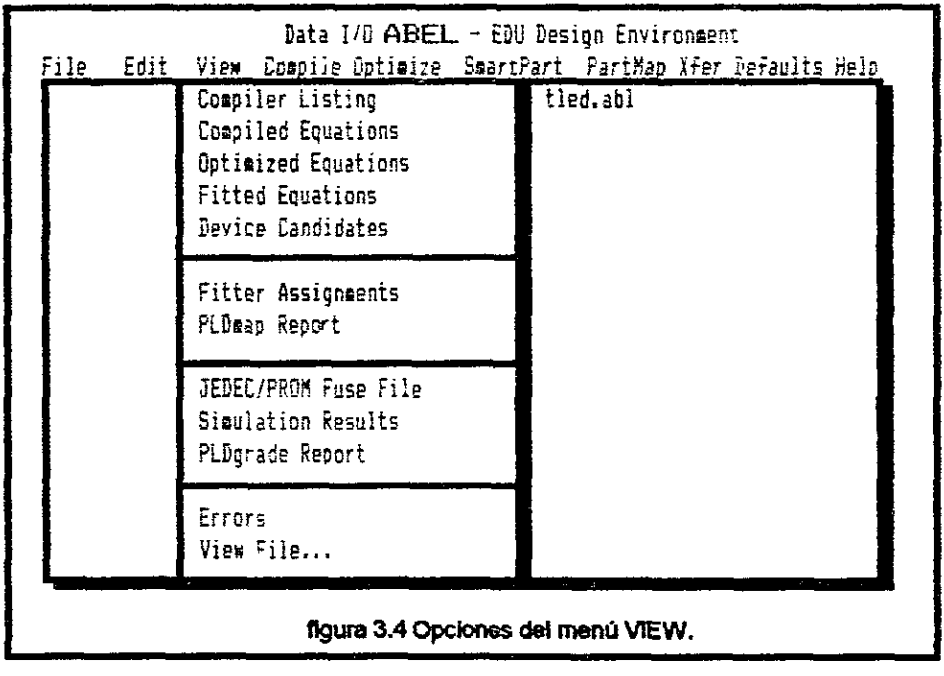


figura 3.4 Opciones del menú VIEW.

III.6 MENÚ COMPILE (COMPILAR)

Las opciones del menú *compile* permiten controlar la operación al compilar programas de HDL a PLA y de PLA a HDL[14], para esto se ejecuta el simulador **ABEL** -PLA que verifica el funcionamiento correcto del compilador. Si se oprime ALT - C se observan las opciones de este comando (figura 3.5), las cuales se explican a continuación:

Compile	Compila el programa y crea los archivos **ABEL** - PLA.
Error Check	Checa únicamente los errores de sintaxis del diseño.
Vectors Only	Compila únicamente los vectores de prueba.
Options...	Establece las opciones del proceso de compilación.
Simulate Equations	Simula las ecuaciones generadas por el compilador .
Re-Simulate	Simula el diseño con vectores de prueba actualizados.
Trace Options...	Establece las opciones de trazado de los vectores de prueba.

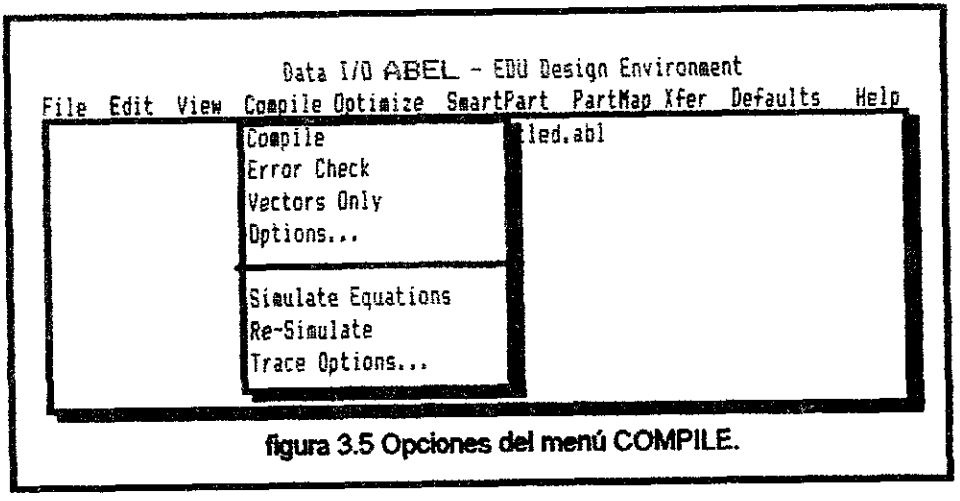


figura 3.5 Opciones del menú COMPILE.

OPCIONES DE COMPILACIÓN

Las opciones de compilación (figura 3.6) con que cuenta **ABEL** se pueden seleccionar oprimiendo la barra espaciadora una vez que se ha seleccionado *OPTIONS* en el menú *COMPILE*. Permiten seleccionar el formato de compilación: HDL a PLA (**ABEL**) o PLA a HDL, este último no genera un listado de archivo c Si se selecciona la lista de compilación en el menú *VIEW* mientras se edita un archivo

[14] HDL(Lenguaje de Programación de Hardware) y PLA (Arreglo de Lógica Programable) generan distintos tipos de código en lenguaje máquina cuando se compilan y a veces es necesario convertir de un tipo a otro.

fuente HDL se observará un listado generado como por la compilación del archivo asociado .ABV (vectores de prueba **ABEL**).

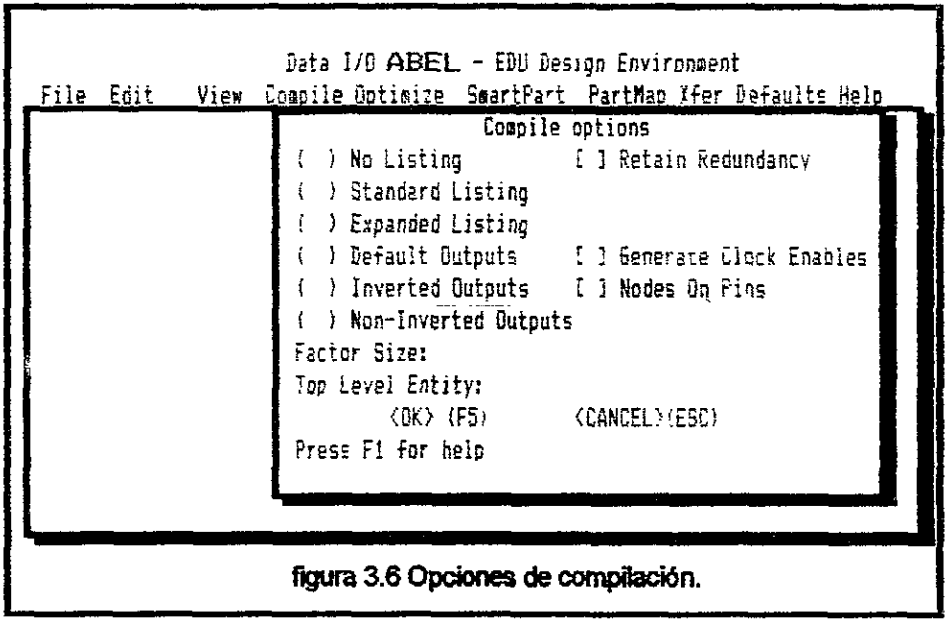


figura 3.6 Opciones de compilación.

Las opciones de la lista de compilación de **ABEL** son:

( ) No Listing

No se generará un listado.

( ) Standard Listing

Se generará un listado conteniendo las líneas del archivo fuente numeradas y los mensajes de error (en caso de errores de compilación).

( ) Expanded Listing

Se generará un listado conteniendo las líneas del archivo fuente numeradas, macros expandidas, directivas y mensajes de error (en caso de ocurrir errores de compilación).

[ ] Retain Redundancy

Conservar la redundancia de compilación. Si se marca este cuadro deshabilita el la conversión de HDL a PLA en reducciones triviales como las siguientes:

A # !A = 1

(A & !B) # (A & B) = A

Se puede usar en caso de que el diseño requiera mantener los minitérminos redundantes (para protección). Otra alternativa para conservar la redundancia es especificar el atributo 'retain' en el archivo fuente en la descripción de las salidas del diseño.

ARGUMENTOS DE COMPILACIÓN.

El área de argumentos de compilación se usa para especificar que argumentos en el texto serán sustituidos por argumentos falsos especificados en la palabra reservada **MODULE** de un archivo fuente **ABEL - HDL**. Si no se especifican argumentos falsos en el diseño se puede dejar esta área en blanco.

[ ] Nodes On Pins

Esta opción instruye al compilador HDL para asignar todas las señales y variables internas a pines en lugar de nodos. Se usa si el dispositivo destino esta densamente ocupado en sus rutas para nodos internos pero tiene pines disponibles.

Factor Size:

Especifica el número máximo de minitérminos por ecuación. Cuando se utiliza, el compilador HDL desarrolla las ecuaciones de tal manera que no se exceda el número máximo de minitérminos de un dispositivo específico.
Se recomiendan los siguientes valores: 8 para circuitos tipo PAL, 16 para PLDs más complejos como los MACH, y valores pequeños como 4 para arquitecturas FPGA .

Top Level Entity:

Especifica el máximo nivel de diseño que será elaborado. La condición inicial es la última arquitectura analizada.

[ ] Generate Clock Enables

Genera una señal de habilitación para funciones complejas de reloj (cuando una señal de reloj depende de múltiples entradas). Esta opción es usada si se tienen dispositivos con habilitación de reloj por flip-flops

( ) Default Outputs

Invierte las salidas basado en el uso que se les da en el diseño. Esta opción se usa si el dispositivo tiene polaridad de salida programable (como en una GAL) y si se controla la inversión de pines desde el diseño.

( ) Inverted Outputs

Sintetiza todas las salidas con polaridad invertida. Esta opción funciona mejor si se tiene un dispositivo que invierte sus salidas de registro (como un PAL 16R8).

( ) Non-Inverted Outputs

Establece que todas las salidas serán del tipo no invertidas. Esta opción funciona mejor si se tiene un dispositivo que no invierte las salidas de registro (como los de ALTERA[15]).

[15] ALTERA es una empresa de fabricación de PLDs.

OPCIONES DE TRAZADO DE LA SIMULACIÓN

Como ya se explicó en el capítulo dos estas opciones permiten seleccionar el formato en que se mostrarán los resultados de la simulación. Podemos seleccionarlas por medio de la barra espaciadora cuando el cursor se encuentra señalando la opción deseada (figura 3.7) Se tienen los siguientes formatos:

( ) No trace

No se generará una salida de simulación.

( ) Pins format

Los valores aparecerán como pines de entrada y salida para cada vector de prueba y tomarán los valores L (bajo) y H (alto).

( ) Wave format

Los valores aparecerán en forma vertical como pulsos usando los caracteres gráficos estándar de IBM.

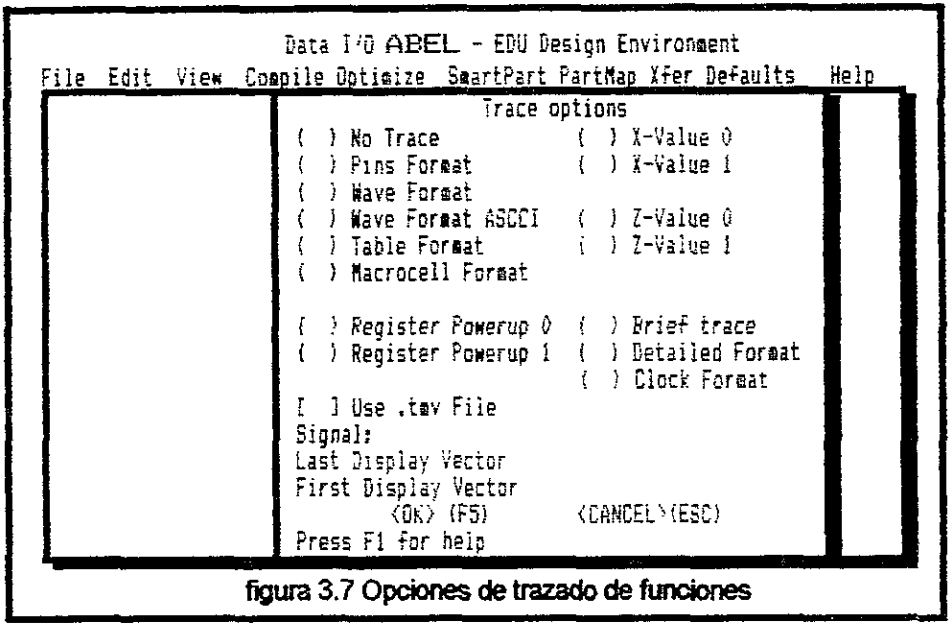


figura 3.7 Opciones de trazado de funciones

( ) Wave format ASCII

Los valores aparecerán en forma de pulsos verticales usando los caracteres estándar ASCII que todas las impresoras y terminales manejan.

( ) Table format

Los valores aparecerán como pines de entrada y salida con los valores L (bajo) y H (alto) en formato de tabla de vectores.

( ) Macrocell format

Los resultados de simulación se mostrados para todas las extensiones punto[16] asociadas con macroceldas. Esta opción debe ser usada junto con la opción "Signal" (señal) que reduce el tamaño del reporte de salida.

( ) Brief trace

Esta opción genera un reporte de resultados de simulación para cada ciclo de reloj, se usa en diseños con registros, o para la estabilización de valores de salida en diseños combinacionales

( ) Detailed format

Esta opción genera un reporte de simulación de resultados para cada nivel de miniterminos del circuito simulado. Es usada para depurar circuitos lógicos complejos.

( ) Clock format

Genera un reporte de simulación que muestra los valores de registro cuando el valor de la señal de reloj pasa de 0 a 1 y otra vez a 0 (un ciclo completo de reloj) para cada vector. El formato de reloj se usa en conjunto con el trazo de macroceldas para depurar circuitos secuenciales asíncronos.

Signal:

Se introduce un espacio en blanco para separar cada señal de la lista de los resultados de simulación. La lista puede contener nombres de señales o números de pines o nodos, por ejemplo:

Signal: Clk Reset LoadA LoadB Q3 Q2 Q1

donde se tienen 5 señales de salida: Clk, Reset, LoadA, LoadB, Q3,Q2, Q1.Si esta área se deja en blanco los resultados de la simulación mostrarán todas las señales que se utilizan en el circuito en cuestión.

Last Display Vector

Establece el número del último vector que se quiere ver en el archivo de resultados de simulación, ejemplo:

Last Display Vector: 5

El último vector simulado será el vector 5. Si se deja esta área en blanco se hace la simulación hasta el último vector .

First Display Vector

Establece el número del primer vector en el archivo de resultados de simulación , por ejemplo:

First Display Vector: 3

El primer vector simulado será el 3. Si se deja esta área en blanco se simulan los vectores a partir del primero.

( ) Register Powerup 0

Todos los registros son inicializados a 0 antes de empezar la simulación.

---

[16] En el apéndice A se explican en detalle las extensiones punto usadas en ABEL.

( ) Register Powerup 1

Todos los registros son inicializados a 1 antes de empezar la simulación.

( ) X-value 0

Usa 0 como valor para las condiciones de no importa durante la simulación.

( ) X-value 1

Usa 1 como valor para las condiciones de no importa durante la simulación.

( ) Z-value 0

Usa el valor de 0 para las condiciones de alta impedancia (. Z .) durante la simulación.

( ) Z-value 1

Usa el valor de 1 para las condiciones de alta impedancia (.Z.) durante la simulación.

[ ] Use .tmv File

Esta opción obliga al archivo JEDsim (simulador JEDEC) a usar los vectores del archivo .tmv en vez de los vectores del archivo JEDEC. El simulador de **ABEL**-PLA (PLAsim) también usa el archivo .tmv.

III.7 MENÚ OPTIMIZE (OPTIMIZAR)

Las opciones del menú *OPTIMIZE* permiten controlar la operación del programa de optimización PLAopt de **ABEL**. Este programa está basado en el programa Express[17] desarrollado por la Universidad de California y tiene muchas opciones que permiten que los diseños sean optimizados para diferentes arquitecturas de dispositivos. Al oprimir ALT - O se observan las opciones de este comando (figura 3.8), las cuales se explican a continuación:

Reduce	Ejecutar el programa PLAopt.
Options...	Establece las opciones de optimización.
Simulate Optimized	Simula las opciones optimizadas.
Trace Options...	Establece las opciones de trazado de la simulación

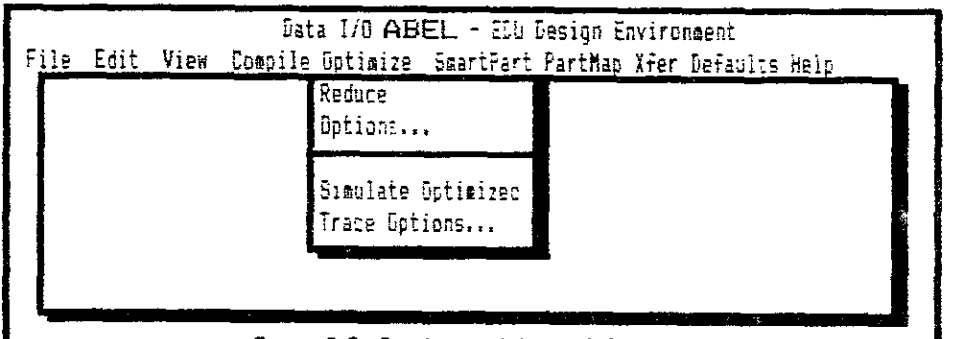


figura 3.8 Opciones del menú OPTIMIZE.

[17] Programa de compilación de PLDs que tiene varias opciones para aprovechar mejor los recursos de un circuito.

Las opciones de trazado de la simulación son las mismas que se describieron en el menú *COMPILE*.

Las opciones de optimización (figura 3.9) son las siguentes:

(.) Use default

Agrupa las partes fijas para FPLA y establece la polaridad automáticamente basándose en el dispositivo especificado en el archivo fuente.

( ) Reduce by pin, auto polarity

Reduce la lógica para que cada señal tenga el menor número de miniterminos posible. La optimización producirá ecuaciones de encendido y apagado que serán usadas en las partes de polaridad programable. Esta es la reducción lógica que se hace si las partes no son especificadas, y se recomienda para muchos tipos de PLDs, incluyendo aquellos que tienen salidas de polaridad fija.

( ) Reduce by pin, fixed polarity

Reduce la lógica para que cada señal tenga el menor número de miniterminos posible, manteniendo la polaridad de las señales que fueron especificadas en el archivo fuente. Esta opción debe usarse solo cuando se quiere que las señales de salida tengan una polaridad específica.

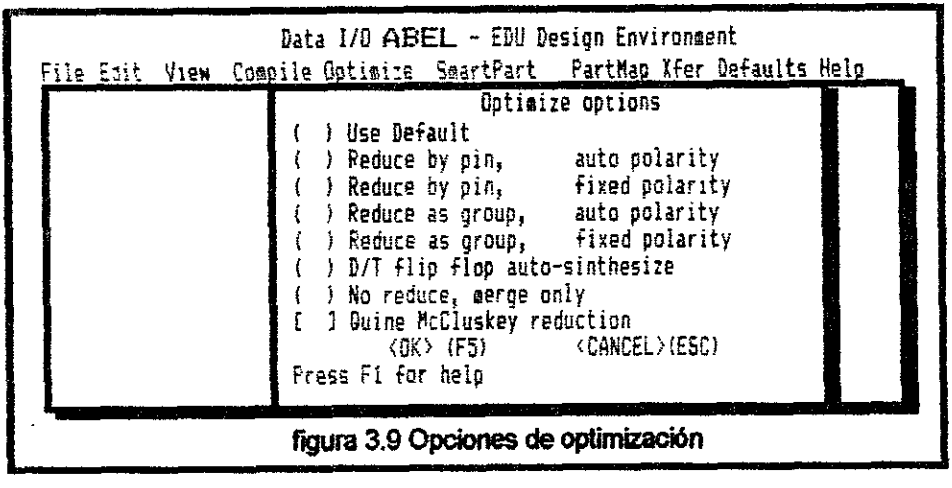


figura 3.9 Opciones de optimización

( ) Reduce as group, auto polarity

Reduce todas las señales como un grupo para partes de FPLA. La combinación de polaridades de salida produce un número menor de minitérminos . Esta opción de reducción puede consumir mucho tiempo. Se recomienda no usarla a menos que sea necesario.

( ) Reduce as group, fixed polarity

Reduce todas las señales como un grupo de partes de FPLA. Se mantiene la polaridad especificada en el archivo fuente .

( ) D/T flip-flop auto-synthesize

Reduce cada grupo de señales de salida tipo flip - flop D y T. Dando como resultado que sea seleccionado el menor número de minitérminos para cada salida. Es muy utilizada cuando se optimizan dispositivos con flip-flops tipo D o T. Para usar esta opción la descripción lógica en el archivo fuente debe contener ecuaciones con extensión .D o .T (ver apéndice B).

( ) No Reduce, merge only

Combina todas las ecuaciones compiladas en un archivo de **ABEL**. No realiza reducción. Puede ser usada si se quiere preservar la lógica redundante de cualquiera de las salidas del diseño.

[X] *Quine McCluskey Reduction*

Al seleccionar esta opción se ejecuta el programa de minimización exacta con el método de implicantes primos de Quine Mc. Cluskey, el cual produce una lógica de minimización más completa . Esta opción puede consumir mucho tiempo en el caso de diseños medianos y grandes, por lo que debe ser seleccionada solo si es absolutamente necesario.

III.8 MENÚ SMARTPART (ASIGNACIÓN INTELIGENTE)

Este menú es una opción de alto nivel de **ABEL**. No está disponible en la versión de **ABEL**-EDU (versión para estudiantes), pero sí en la versión de **ABEL** profesional. Cuando alguna de las opciones de un menú están desactivadas (aparecen en letras más tenues y no se pueden ejecutar) es porque no se tiene esta opción instalada.

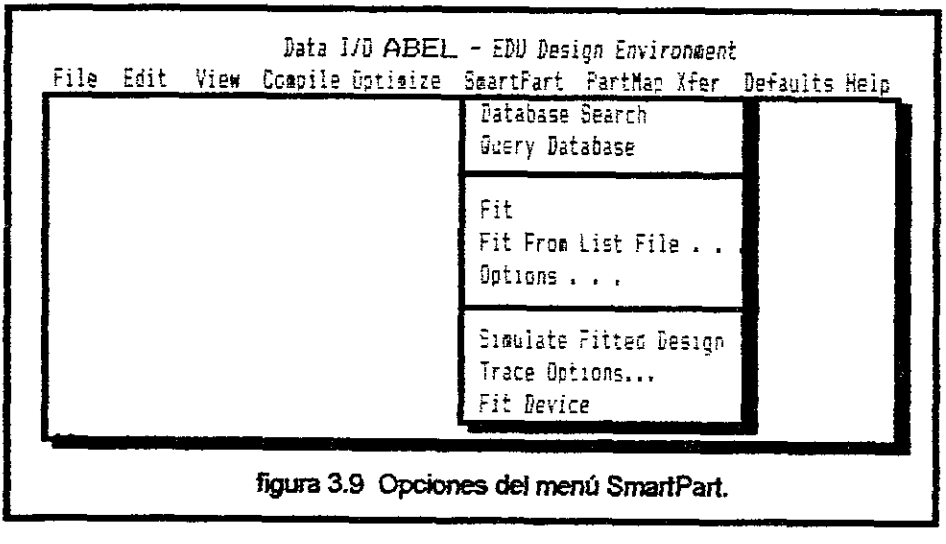


figura 3.9 Opciones del menú SmartPart.

El menú *SmartPart* incluye opciones para selección y asignación de pines en dispositivos. Selecciona dispositivos de acuerdo a un criterio y ejecuta la función de "búsqueda en base de datos". El resultado es una lista de posibles dispositivos que pueden ser utilizados como entrada para la asignación de pines (seleccionando *Fit from List File*, asignación desde el archivo de lista) o puede seleccionarse una arquitectura determinada a partir de la lista de dispositivos y especificarla en el programa de asignación de pines. Al oprimir ALT - S observamos las opciones del menú *SmartPart* (figura 3.9), que explicamos a continuación:

Database Search	Busca en la base de datos y selecciona los dispositivos candidatos.
Query Database	Pregunta que base de datos se utiliza.
Modify Criteria...	Establece el criterio de selección de dispositivos.
Fit	Asigna los pines de acuerdo a la arquitectura especificada.
Fit from List File...	Asigna los pines desde una lista de dispositivos.
Options...	Establece las opciones de asignación de pines.
Simulate Fitted Design	Simula el diseño después de la asignación de pines.
Trace Options...	Establece las opciones de trazado de la simulación.
Fit Device	Realiza la asignación de las señales en los pines del dispositivo especificado.

Las opciones de trazado de la simulación son las mismas que se describieron para el menú COMPILE y al igual que las de asignación de pines se pueden seleccionar con las teclas de flechas y oprimiendo la barra espaciadora en la opción correspondiente, estas últimas (figura 3.10) se explican a continuación:

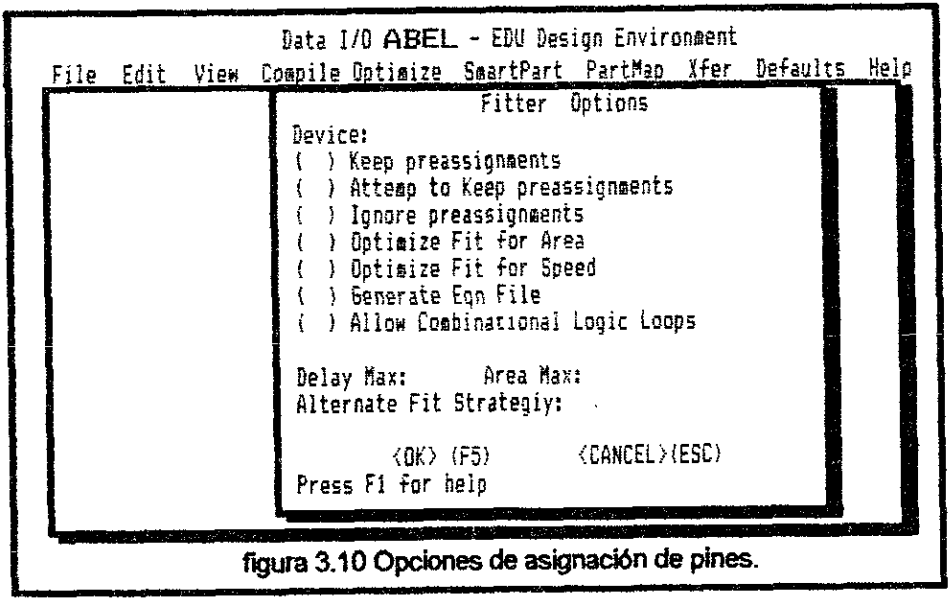


figura 3.10 Opciones de asignación de pines.

( ) Keep preassignments

No cambia los números de pines y nodos especificados en el programa fuente en **ABEL**. Solo las ecuaciones que no están asignadas (o asignadas al pin o nodo 0) son asignadas a los pines o nodos libres.

(.) Attempt to keep preassignments

Intenta preservar los números de pines y nodos especificados en el archivo fuente en **ABEL**. Esta es la condición normal de operación.

( ) Ignore preassignments

Ignora todos los números de pines y nodos especificados en el archivo fuente en **ABEL**. Esta opción se ejecuta más rápido si no existen especificaciones de pines y nodos en el archivo fuente en **ABEL**.

( ) Optimize Fit for Área

La optimización por área genera el circuito más pequeño posible.

( ) Optimize Fit for Speed

La optimización por velocidad genera el circuito en el que las señales se propaguen internamente con la mayor velocidad posible.

( ) No Optimization

No optimiza el circuito.

[ ] Generate Eqn File

Esta opción genera un archivo de ecuaciones que representan la optimización lógica como será implementada en el dispositivo. La extensión de archivos para las ecuaciones FPGA es .eqn. No todas las asignaciones de dispositivos pueden usar esta opción.

[ ] Allow Combinational Logic Loops

Esta opción permite que la lógica combinacional se retroalimente en si misma. Normalmente esto es considerado como un error. La retroalimentación en loops lógicos se rompe antes de la implementación en el dispositivo. No todas las asignaciones de dispositivos pueden usar esta opción.

Delay Max

Esta opción especifica el número máximo de niveles lógicos / nanosegundo que puede tener la ruta de acceso más larga en un circuito, por ejemplo:

Delay Max: 5

Esto especifica que no se permiten más de 5 nanosegundos para que la señal recorra su ruta de acceso.

No todas las asignaciones de dispositivos pueden usar esta opción.

Área Max

Esta opción especifica el número máximo de bloques lógicos que los circuitos pueden usar, ejemplo:

Área Max: 15

Este dispositivo no puede tener más de 15 bloques lógicos.

No todas las asignaciones de dispositivos pueden usar esta opción.

Alternate Fit Strategy :

Esta opción se usa para especificar las estrategias opcionales de asignación de pines para los programas que tienen dicha característica.

III. 9 MENÚ PARTMAP (MAPEAR DISPOSITIVOS)

Este menú permite controlar la operación del programa mapeador de dispositivos. En **ABEL** mapear se refiere a crear un archivo que será cargado en el programador de PLDs para grabar el archivo JEDEC (Mapa de fusibles). Al oprimir ALT -M se observan sus opciones (figura 3.11).

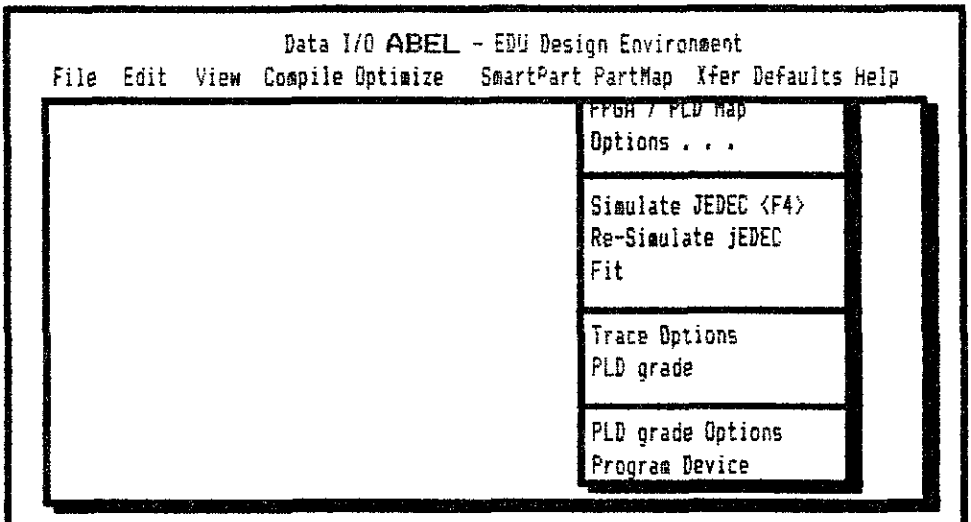


figura 3.11 Opciones del menú PartMap.

El mapeador de dispositivos puede ser invocado para usar una arquitectura específica o para todos los dispositivos y sucesivamente ajustarse por la selección hecha en *SmartPart*. El programa de mapeo de dispositivos produce un archivo (típicamente un archivo JEDEC) que puede ser transferido al programador de dispositivos usando la opción *Program Device* (Programar dispositivo). La opción *Simulate JEDEC* invoca al dispositivo simulador para verificar que el diseño mapeado funcione antes de la programación del dispositivo.

Las opciones de este menú se explican a continuación:

FPGA / PLD Map — Mapea el dispositivo (Crea un archivo de carga del programador)

Options...	Establece las opciones de mapeo de PLDs.
Simulate JEDEC <<F4>>	Simula el archivo JEDEC.
Re-Simulate JEDEC	Hace la simulación con nuevos vectores de prueba.
Trace options...	Establece las opciones de trazado de la simulación del archivo JEDEC.
PLDgrade	Condición inicial del archivo JEDEC.
PLDgrade options...	Establece las opciones del archivo JEDEC.
Program Device	Invoca al emulador de terminal.

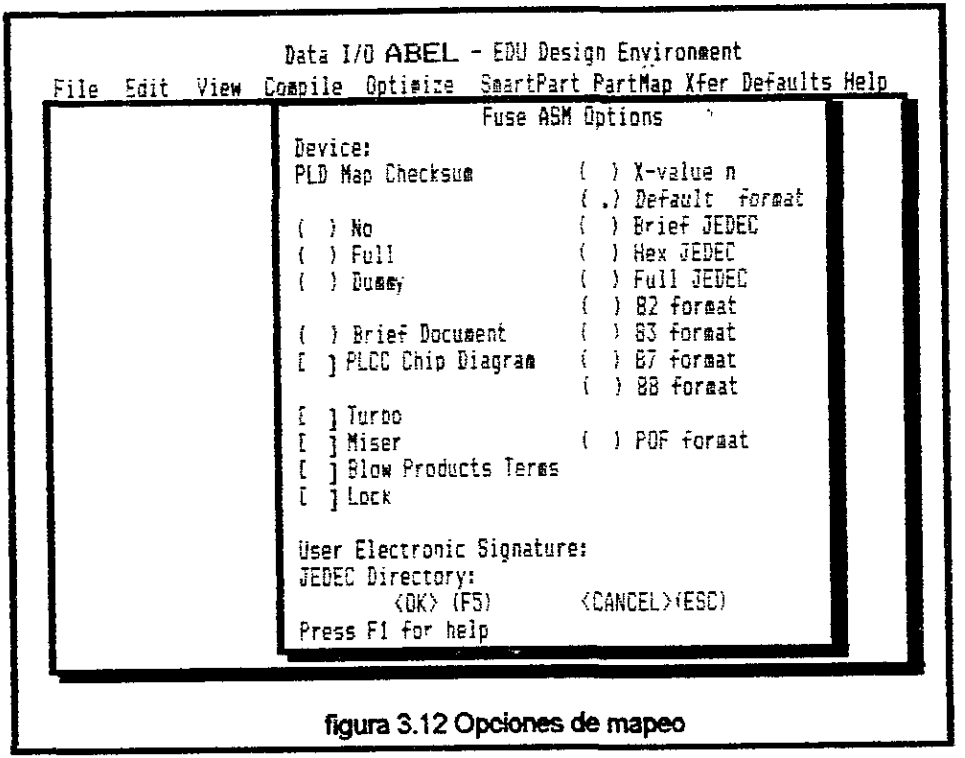


figura 3.12 Opciones de mapeo

Las opciones de trazado de la simulación son las mismas que se describieron para el menú COMPILE., y las de mapeo (figura 3.12) se describen a continuación:

PLDmap Checksum

Selecciona una opción de verificación. Las opciones válidas son:

( ) No

No produce una verificación de suma de productos.

( ) Full

Produce una verificación de suma de productos al final del archivo JEDEC.

( ) Dummy

Produce una verificación ficticia (todos ceros) al final del archivo JEDEC.

( ) Brief document

Se genera un reporte que incluye el mapeo de ecuaciones, un diagrama del chip y un sumario de utilización.

( ) Long document

Se genera un reporte que incluye la información listada en el párrafo anterior, así como el mapa de fusibles y los vectores de prueba.

[ ] PLCC Chip Diagram

Genera el diagrama del chip en el archivo de reporte (.doc) en el formato PLCC.

[ ] Turbo

Esta opción hace que el programador de fusibles habilite la opción de alta velocidad en el dispositivo que será grabado.

[ ] Miser

Hace que el programador de fusibles habilite el modo de baja potencia en el dispositivo que será grabado.

[ ] Blow product terms

Se desconectan todos los fusibles no utilizados en el dispositivo que será grabado. Esto puede producir una mayor velocidad de operación del dispositivo.

[ ] Lock

Se programa el fusible de seguridad (si es que lo hay) en el dispositivo que será grabado.

( ) X-value n

Asigna un valor a las condiciones de no importa en el mapa de fusibles. Si n vale 0, se asignará un 0 a todas las constantes .X.; si n vale 1, se asignará un 1 a todas las constantes .X.

(.) Default format

Selecciona el formato de archivo apropiado para grabar el diseño. Generalmente se utiliza el formato JEDEC.

( ) Brief JEDEC

Produce un archivo JEDEC estándar conteniendo los fusibles y los datos de los vectores de prueba (.jed). Los fusibles no usados no se incluyen.

( ) Hex JEDEC

Produce un archivo JEDEC estándar conteniendo los fusibles y los datos de los vectores de prueba (.jed). Estos están escritos en hexadecimal.

( ) Full JEDEC

Produce un archivo JEDEC estándar conteniendo los fusibles y los datos de los vectores de prueba (.jed). Todos los fusibles son incluidos.

( ) 82 format

Produce un archivo con formato de PROM **Motorola** de 8 bits (.p82).

( ) 83 format

Produce un archivo con formato de PROM **Intel** de 8 bits (.p83).

( ) 87 format
Produce un archivo con formato de PROM **Motorola** de 16 bits (.p87).

( ) 88 format
Produce un archivo con formato de PROM **Intel** de 16 bits (.p88)

( ) POF format
Produce un archivo con formato para programador de **ALTERA** (.pof). Los archivos con formato POF están limitados a dispositivos de **ALTERA**.

User Electronic Signature :
Introduce una secuencia de caracteres para programar dentro de los fusibles de firma o clave (si el dispositivo los tiene). Se utilizan ocho fusibles por caracter, por ejemplo:
User Electronic Signature: JP0690

JEDEC Directory:

PLDmap Directory:
Introduce un nombre de directorio válido para FUSEASM que escribirá en los archivos JEDEC o PROM, por ejemplo:
Directory for JEDEC file: design5/jedec

III.10 MENÚ XFER (TRANSFERIR)

Este menú provee un mecanismo para trasladar un diseño hecho en **ABEL** formatos usado por otros lenguajes de diseño de PLDs. Esta opción no está disponible para **ABEL- EDU**. Al oprimir ALT - X se observan las opciones de este menú (figura 3.13), las cuales se explican a continuación:

Translate	Traslada un diseño hecho en **ABEL** a otro formato.
Translate Options...	Especifica que alternativas de formato serán usadas.

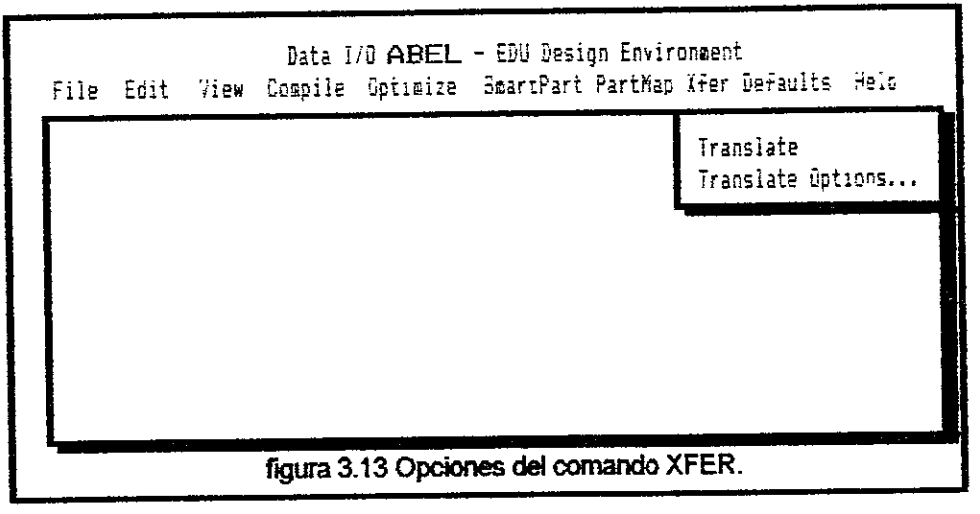


figura 3.13 Opciones del comando XFER.

Las opciones de formato que podemos usar con este comando (figura 3.14) son las siguientes:

( ) Xilinx PDS

Produce un archivo con formato PDS (usado por **PALASM II**) con extensiones especiales que el software **XilinxXACT** puede importar. El archivo generado tiene el nombre modulo.pds.

( ) Actel PDS

Produce un archivo con formato PDS (usado por **PALASM II**) con extensiones especiales que el software **Actel** puede importar. El archivo generado es modulo.pds.

( ) Plusasm

Produce un archivo con formato Plus Logic Plusasm que puede ser usado para importar a el software **Plus Logic**. El archivo generado es modulo.pld.

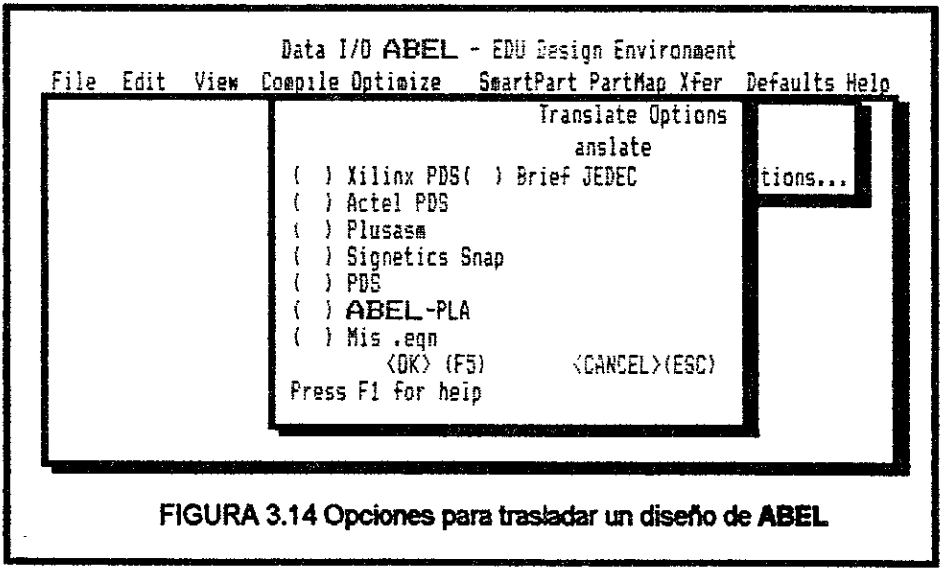


FIGURA 3.14 Opciones para trasladar un diseño de **ABEL**

( ) Signetics Snap

Produce un archivo del tipo **Signetics Snap** que puede ser leído por el software del mismo nombre. Los archivos generados son modulo.bee y modulo.pin.

( ) PDS

Produce un archivo con formato PDS (usado por **PALASM II**) que el software Actel puede importar. El archivo generado es modulo.pds.

( ) ABEL-PLA
Produce un archivo **Berkeley** PLA con extensiones de **ABEL**. El archivo generado es modulo.pla.

( ) MIS .eqn
Produce un archivo **Berkeley** MIS .eqn el cual puede ser leído por el programa **Chortle** de la Universidad de Toronto.

III. 11 MENÚ DEFAULTS (CONDICIONES INICIALES)

Contiene opciones para modificar el ambiente de diseño de **ABEL**. Al oprimir la combinación de teclas ALT - D se muestran las opciones de este menú (figura 3.15), las cuales se explican a continuación:

Auto Update	Habilita la actualización automática del diseño.
Force Fit Update	Obliga a que la asignación sea llamada durante la actualización automática del diseño.
Spaces to tabs	Convierte los espacios en tabuladores cuando graba el archivo.
Read Only	El archivo es solo de lectura y no se puede editar.
Program Pause	Hace una pausa después de que se ejecuta un programa.

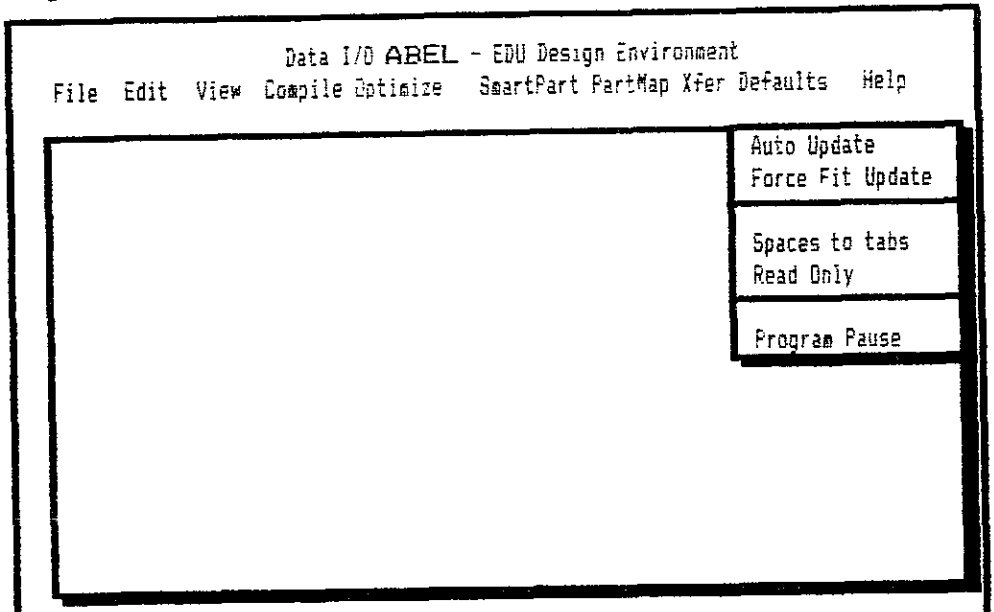


figura 3.15 Opciones del menú DEFAULTS.

III.12 MENÚ HELP (AYUDA)

Consta de varios métodos para accesar la ayuda en línea de **ABEL**. Al oprimir ALT-H se observan las opciones de este menú (figura 3.16), las cuales se explican a continuación:

Help For Help...	Ayuda sobre como usar la ayuda.
Index...	Índice alfabético de los temas de ayuda.
Keyboard...	Ayuda acerca de las teclas especiales.
Design Process...	Ayuda sobre el procesador de diseños de **ABEL**.
Menús...	Ayuda acerca de los menús de comandos.
Program Options...	Ayuda sobre las opciones de programación en **ABEL**.
Language...	Ayuda sobre el lenguaje **ABEL** - HDL.
Devices...	Ayuda acerca de los dispositivos que se pueden utilizar en **ABEL**.
Errors...	Ayuda sobre los números de error que genera el compilador de **ABEL**.
About...	Acerca del ambiente de diseño de **ABEL**-HDL.

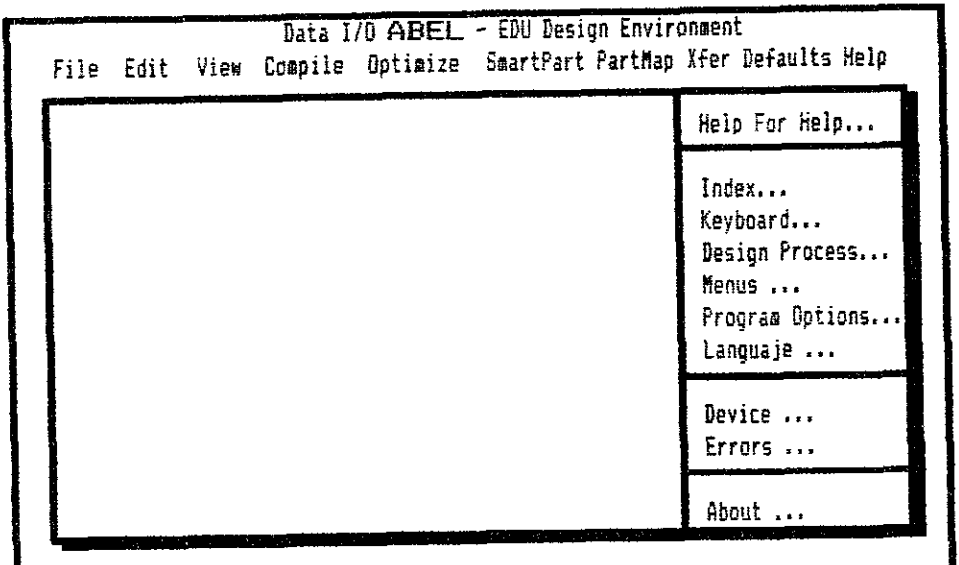


figura 3.16 Opciones de ayuda de **ABEL**.

FALTA PAGINA

No. 66

CAPÍTULO IV. SÍNTESIS DE CIRCUITOS COMBINACIONALES MEDIANTE ABEL

IV. 1 METODOLOGÍA DE DISEÑO DE CIRCUITOS COMBINACIONALES EN ABEL

Un circuito combinacional consta de compuertas lógicas cuyas salidas en cualquier momento se determinan directamente a partir de los valores de entrada, sin requerir de elementos de memoria.

El diseño de circuitos combinacionales usando PLDs empieza con la especificación del problema y termina con la grabación del dispositivo. En este proceso se aplican los siguientes pasos:

1. A partir de las especificaciones del circuito se determina el número de entradas y salidas, asignándoles un nombre de variable a cada una.
2. Se establece la tabla de verdad que define la relación que existe entre las entradas y las salidas.
3. Se establecen las ecuaciones lógicas del circuito.
4. Se escribe el programa correspondiente en el software **ABEL** utilizando tabla de verdad o ecuaciones.
5. Se compila el programa y si hay errores se corrigen hasta que el programa esté libre de errores.
6. Se hace la simulación de los vectores de prueba y si existen errores se corrigen.
7. Se graba el dispositivo.

En este capítulo analizaremos como el software **ABEL** puede ser utilizado para describir una gran variedad de circuitos combinacionales pequeños o medianos, tales como multiplexores, codificadores, sumadores, etc. Estos circuitos básicos comúnmente son usados en muchos diseños más grandes. En este trabajo se utilizarán para ilustrar importantes consideraciones de diseño e implementación.

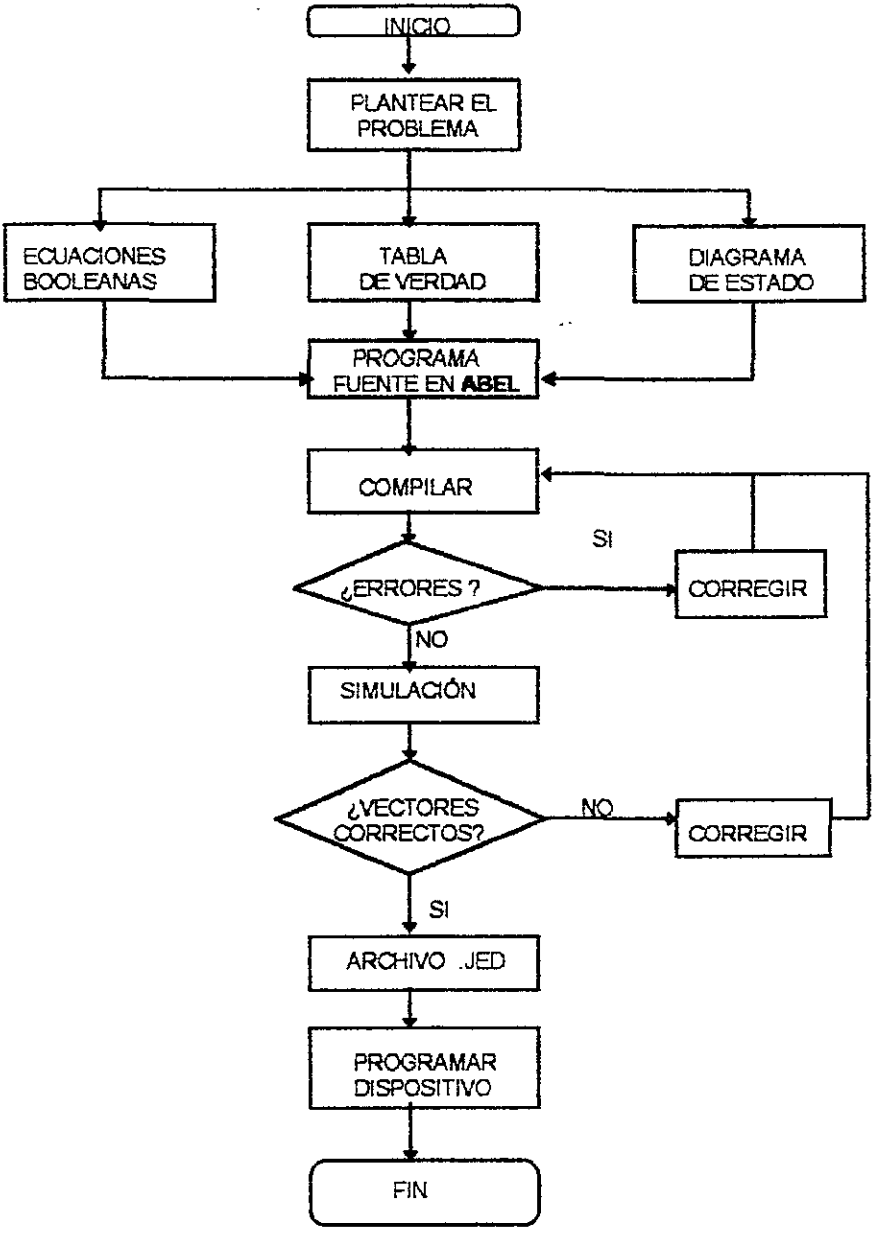


figura 4.1 Metodología de diseño de circuitos combinacionales usando **ABEL**.

IV.2 COMPUERTAS BÁSICAS.

El circuito GAL 16V8, el primero de la familia GAL, tiene 8 entradas y 8 macroceldas lógicas de salida que se pueden configurar como entradas en caso de ser necesario. Este primer ejemplo de aplicación describe un circuito que contiene las siguientes compuertas: una AND, una NAND de tres entradas, una OR, una NOR, una OR- EXCLUSIVA y una NOT en un GAL 16V8.

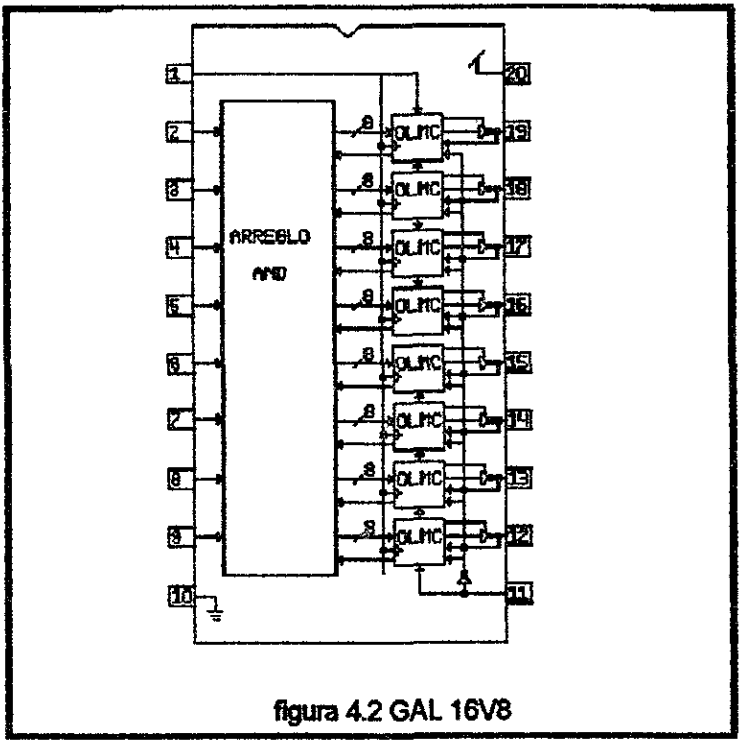


figura 4.2 GAL 16V8

La figura 4.2 muestra el diagrama de un GAL 16V8, en donde se observa que cuenta con 20 pines, de los cuales 2 son utilizados para polarización, 10 son entradas y 8 están conectados a macroceldas lógicas de salida (ver apéndice B), las OLMC pueden funcionar como entradas o salidas. El programa fuente en **ABEL** para este ejemplo es el siguiente:

```
module compt
title 'Compuerta AND,NAND,OR,NOR , X-OR y NOT
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores'
disp1 DEVICE 'g16v8';
A,B,C,D,E,F,G,H,I,J,K pin; //Entradas
AND, NAND, OR, NOR, XOR, NOT pin istype 'com'; //Salidas
ENTRADAS = [A,B,C,D,E,F,G,H,I,J,K];
SALIDAS = [AND,NAND,OR, NOR,XOR,NOT];
```

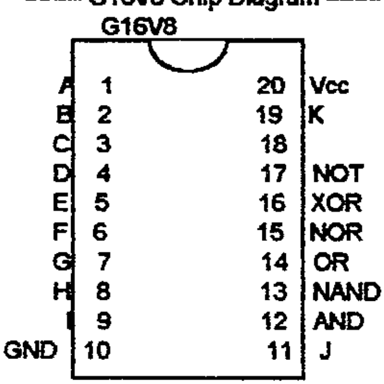
```
EQUATIONS
     AND = A & B;
     NAND = !(C & D);
     OR = E # F;
     NOR = !(G # H);
     XOR = I $ J;
     NOT = !K;
TEST_VECTORS (ENTRADAS -> SALIDAS)
        [0,0,0,0,0,0,0,0,0,0,0] -> [0,1,0,1,0,1];
        [0,1,0,1,0,1,0,1,0,1,0] -> [0,1,1,0,1,1];
        [1,0,1,0,1,0,1,0,1,0,1] -> [0,1,1,0,1,0];
        [1,1,1,1,1,1,1,1,1,1,1] -> [1,0,1,0,0,0];
END
```

Una vez que se compila el programa se genera un reporte el siguiente reporte del dispositivo, que en este caso es el siguiente:

```
                                   Page 1
ABEL-EDU 5.04.005 - Device Utilization Chart     Thu Oct 16 20:02:59 1997
Compuertas AND,NAND,OR,NOR , X-OR  y NOT
   Celia Rosa Fierro Santillán y José Filiberto Luie Flores
  ==== E0320 Programmed Logic ====
AND     = ( A & B );
NAND    = !( C & D );
OR      = !( !E & !F );
NOR     = ( !G & !H );
XOR     = ( !I & J #  I & !J );
NOT     = ( !K );
```

Esta parte contiene las ecuaciones con las que fue diseñada la lógica del circuito.

```
                                 Page 2
ABEL-EDU 5.04.005 - Device Utilization Chart     Thu Oct 16 20:02:59 1997
Compuertas AND,NAND,OR,NOR ,  X-OR  y NOT
   Celia Rosa Fierro Santillán y José Filiberto Luie Flores
              ==== G16V8 Chip Diagram ====
```



Esta parte corresponde a la asignación de pines de entrada y salida en el circuito GAL 16V8 para un circuito que contiene una compuerta AND, una NAND, una OR, una NOR, una OR-EXCLUSIVA y una NOT

SIGNATURE: N/A

Page 3

ABEL-EDU 5.04.005 - Device Utilization Chart Thu Oct 16 20:02:59 1997
Compuertas AND,NAND,OR,NOR , X-OR y NOT
Celia Rosa Fierro Santillán y José Filiberto Lule Flores

==== E0320 Resource Allocations ====

Device Resources	Resource Available	Design Requirement	Part Utilization	Unused
Dedicated input pins	10	11	10	0 (0 %)
Combinatorial inputs	10	10	10	0 (0 %)
Registered inputs	-	0	-	-
Dedicated output pins	-	6	-	-
Bidirectional pins	8	0	7	1 (12 %)
Combinatorial outputs	-	6	-	-
Registered outputs	-	0	-	-
Reg/Com outputs	8	-	6	2 (25 %)
Two-input XOR	-	0	-	-
Buried nodes	-	0	-	-
Buried registers	-	0	-	-
Buried combinatorials	-	0	-	-

En esta parte (reporte de utilización del dispositivo) se observan los requerimientos del diseño y las características del dispositivo en el que se graba dicho diseño. En este diseño en particular se requieren 11 entradas pero solo hay 10 entradas disponibles en el GAL16V8, por lo que una OLMC se utilizó en el modo de entrada. Se utilizan 6 salidas combinacionales, por lo que 6 OLMCs se configuraron como salidas combinacionales. Las entradas del circuito se utilizaron al 100%, las OLMC al 87.5%, no se utilizaron nodos ni entradas o salidas de tipo registro. Se puede decir que el diseño se optimizó utilizando un dispositivo GAL 1816V8.

Page 4

ABEL-EDU 5.04.005 - Device Utilization Chart Thu Oct 16 20:02:59 1997
Compuerta AND,NAND,OR,NOR , X-OR y NOT
Celia Rosa Fierro Santillán y José Filiberto Lule Flores

==== G16V8 Product Terms Distribution ====

Signal Name	Pin Assigned	Terms Used	Terms Max	Terms Unused
AND	12	1	8	7
NAND	13	1	8	7
OR	14	1	8	7
NOR	15	1	8	7
XOR	16	2	8	6
NOT	17	1	8	7

En esta parte (distribución de minitérminos) se observan las señales de salida, el número de pin al que fue asignada cada una, el número de minitérminos utilizados (compuertas de la matriz AND), el número máximo de minitérminos que se pueden utilizar para cada señal de salida (generalmente 8), y los minitérminos no utilizados.

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

Signal Name	Pin	Pin Type
A	1	CLK/IN
B	2	INPUT
C	3	INPUT
D	4	INPUT
E	5	INPUT
F	6	INPUT
G	7	INPUT
H	8	INPUT
I	9	INPUT
J	11	INPUT
K	19	BIDIR

En la lista de entradas y retroalimentaciones se observan las señales de entrada, el pin a que fue asignada cada entrada y el tipo de entrada.

ABEL-EDU 5.04.005 - Device Utilization Chart Thu Oct 16 20:02:59 1997

Compuerta AND,NAND,OR,NOR, X-OR y NOT
Celia Rosa Fierro Santillán y José Filiberto Lule Flores

==== E0320 Unused Resources ====

Pin Number	Pin Type	Product Terms	Flip-flop Type
18	BIDIR	NORMAL 8	D

En esta parte se observan los pines que no fueron utilizados en el circuito, para el caso particular de este diseño no se utilizó el pin 18 que corresponde a una OLMC.

ABEL-EDU 5.04.005 - Device Utilization Chart Thu Oct 16 20:02:59 1997
Compuerta AND,NAND,OR,NOR , X-OR y NOT
Celia Rosa Fierro Santillán y José Filiberto Lule Flores

===== I/O Files =====
Module: 'compt'
Input files
==========

ABEL PLA file: compt.tt3
Vector file: compt.tmv
Device library: G16V8.dev

Output files
==========
Report file: compt.doc
Programmer load file: disp1.jed

En esta parte se observan los archivos generados por el módulo 'compt', uno de los programas del ambiente de diseño **ABEL**. Estos son de dos tipos: archivos de entrada (archivo de PLA, vectores de prueba, librería del circuito GAL16V8) y archivos de salida (archivo de reporte y archivo de mapa de fusibles).

El archivo JEDEC que resultó después de compilar y simular este diseño es el siguiente:

□ABEL-EDU 5.04.005 Data I/O Corp. JEDEC file for: G16V8
Created on: Mon Oct 27 18:37:41 1997
Compuerta AND,NAND,OR,NOR , X-OR y NOT
Celia Rosa Fierro Santillán y José Filiberto Lule Flores
*
QP20* QF2916* QV4* F0*
X0*
NOTE Table of pin names and numbers*
NOTE PINS A:1 B:2 C:3 D:4 E:5 F:6 G:7 H:8 I:9 J:11 K:19 AND:12 NAND:13*
NOTE PINS OR:14 NOR:15 XOR:16 NOT:17*
L0648 11111111111111111111111111111101*
L0936 11111111111111111111111111111111*
L0972 11111111111111101101111111111111*
L1008 11111111111111110011111111111111*
L1260 11111111111111111111111111111111*
L1296 11111111111101010111111111111111*
L1584 11111111111111111111111111111111*
L1620 11111111010111111111111111111111*
L1908 11111111111111111111111111111111*
L1944 11111010111111111111111111111111*
L2232 11111111111111111111111111111111*

```
L2268 10101111111111111111111111111111*
L2556 11111111111111111111111111111111*
L2592 11111111111111111111111111111111*
L2628 11111111111111111111111111111111*
L2664 11111111111111111111111111111111*
L2700 11111111111111111111111111111111*
L2736 11111111111111111111111111111111*
L2772 11111111111111111111111111111111*
L2808 11111111111111111111111111111111*
L2844 11111111111111111111111111111111*
L2880 00000000110011001100000000001100*
L2912 11*
L2914 11*
V0001 000000000N0LHLHLHX0N*
V0002 010101010N1LHHLHHX0N*
V0003 101010101N0LHHLHLX1N*
V0004 111111111N1HLHLLLX1N*
C5BAC*
□3939
```

IV.3 MULTIPLEXORES

Un multiplexor es un circuito combinacional que selecciona una de varias entradas y la dirige a una sola línea de salida. En este ejemplo se diseñó un circuito que contiene dos multiplexores, uno de 4:1 y otro de 8:1 con una GAL 20V8. Se tienen entonces, 12 entradas, 2 salidas y 3 líneas de selección (S0,S1,S2), los dos bits menos significativos sirven como selección para el multiplexor de 4:1 y los tres bits de selección se usan en el multiplexor de 8:1. En este ejemplo y los siguientes de este capítulo se incluyen únicamente el programa fuente del diseño en **ABEL** (..ABL), el archivo de mapa de fusibles (.JED) y el diagrama de chip; se omite el reporte de dispositivo por ser muy extenso y contener información redundante.

PROGRAMA EN **ABEL**

```
module mux81
title 'Circuito que contiene dos multiplexores uno de 4:1 y otro de 8:1
    usando el GAL16V8
   Celia Rosa Fierro Santillán y José Filiberto Lule Flores'
dispositivo DEVICE 'G16V8';
A0,A1,A2,A3,A4,A5,A6,A7  PIN; "Entradas al multiplexor 1
B0,B1,B2,B3 PIN; "Entradas al multiplexor 2
S0,S1,S2 PIN; "Bits de selección
SALA, SALB PIN ISTYPE 'com'; " Salidas de los multiplexores
OE PIN; "Habilitación
SELECCION = [S0,S1,S2];
EQUATIONS
SALA.OE = SALA;
SALB.OE = SALB;
```

```
TRUTH_TABLE ([ A0, A1, A2, A3, A4, A5, A6, A7, S0, S1, S2, OE] -> [SALA])
[X,,X,,X,,X,,X,,X,,X,,X,,X,,X,,X, 1 ] -> [0];
[ 0 ,X,,X,,X,,X,,X,,X,,X,,X, 0 , 0 , 0 , 0 ] -> [0];
[ 1 ,X,,X,,X,,X,,X,,X,,X,,X, 0 , 0 , 0 , 0 ] -> [1];
[X,, 0 ,X,,X,,X,,X,,X,,X,,X, 0 , 0 , 1 , 0 ] -> [0];
[X,, 1 ,X,,X,,X,,X,,X,,X,,X, 0 , 0 , 1 , 0 ] -> [1];
[X,,X,, 0 ,X,,X,,X,,X,,X,,X, 0 , 1 , 0 , 0 ] -> [0];
[X,,X,, 1 ,X,,X,,X,,X,,X,,X, 0 , 1 , 0 , 0 ] -> [1];
[X,,X,,X,, 0 ,X,,X,,X,,X,,X, 0 , 1 , 1 , 0 ] -> [0];
[X,,X,,X,, 1 ,X,,X,,X,,X,,X, 0 , 1 , 1 , 0 ] -> [1];
[X,,X,,X,,X,, 0 ,X,,X,,X,,X, 1 , 0 , 0 , 0 ] -> [0];
[X,,X,,X,,X,, 1 ,X,,X,,X,,X, 1 , 0 , 0 , 0 ] -> [1];
[X,,X,,X,,X,,X,, 0 ,X,,X,, 1 , 0 , 1 , 0 ] -> [0];
[X,,X,,X,,X,,X,, 1 ,X,,X,, 1 , 0 , 1 , 0 ] -> [1];
[X,,X,,X,,X,,X,,X,, 0 ,X,, 1 , 1 , 0 , 0 ] -> [0];
[X,,X,,X,,X,,X,,X,, 1 ,X,, 1 , 1 , 0 , 0 ] -> [1];
[X,,X,,X,,X,,X,,X,,X,, 0 , 1 , 1 , 1 , 0 ] -> [0];
[X,,X,,X,,X,,X,,X,,X,, 1 , 1 , 1 , 1 , 0 ] -> [1];
TRUTH_TABLE ([ B0, B1, B2, B3, S1, S2, OE] -> [SALB])
[X,,X,,X,,X,,X,,X,, 1 ] -> [0];
[ 0 ,X,,X,,X,, 0 , 0 , 0 ] -> [0];
[ 1 ,X,,X,,X,, 0 , 0 , 0 ] -> [1];
[X,, 0 ,X,,X,, 0 , 1 , 0 ] -> [0];
[X,, 1 ,X,,X,, 0 , 1 , 0 ] -> [1];
[X,,X,, 0 ,X,, 1 , 0 , 0 ] -> [0];
[X,,X,, 1 ,X,, 1 , 0 , 0 ] -> [1];
[X,,X,,X,, 0 , 1 , 1 , 0 ] -> [0];
[X,,X,,X,, 1 , 1 , 1 , 0 ] -> [1];
TEST_VECTORS([B0,B1,B2,B3,S1,S2,OE] -> [SALB])
        [ 1, 0, 0, 0, 0, 0, 1] -> [0];
        [ 0, 0, 0, 1, 0, 0, 0] -> [0];
        [ 1, 0, 0, 0, 0, 0, 0] -> [1];
        [ 0, 0, 0, 0, 0, 1, 0] -> [0];
        [ 0, 1, 0, 0, 0, 1, 0] -> [1];
        [ 1, 0, 0, 0, 1, 0, 0] -> [0];
        [ 0, 0, 1, 0, 1, 0, 0] -> [1];
        [ 0, 1, 0, 0, 1, 1, 0] -> [0];
        [ 0, 0, 0, 1, 1, 1, 0] -> [1];
TEST_VECTORS ([A0,A1,A2,A3,A4,A5,A6,A7,S0,S1,S2,OE] -> [SALA])
        [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1] -> [0];
        [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0] -> [0];
        [ 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0] -> [1];
        [ 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0] -> [0];
        [ 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0] -> [1];
        [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0] -> [0];
        [ 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0] -> [1];
        [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0] -> [0];
```

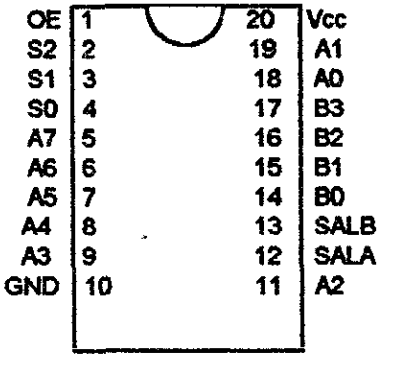
```
        [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0] -> [1];
        [0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0] -> [0];
        [0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0] -> [1];
        [0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0] -> [0];
        [0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0] -> [1];
        [0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0] -> [0];
        [0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0] -> [1];
        [0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0] -> [0];
        [0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0] -> [1];
END
```

ARCHIVO JEDEC

```
□ABEL-EDU 5.04.005 Data I/O Corp. JEDEC file for: G16V8
Created on: Sat Oct 18 22:58:46 1997
Circuito que contiene dos multiplexores uno de 4:1 y otro de 8:1
    usando el GAL20V8
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores
*
QP20* QF2916* QV26* F0*
 X0*
NOTE Table of pin names and numbers*
NOTE PINS A0:18 A1:19 A2:11 A3:9 A4:8 A5:7 A6:6 A7:5 B0:14 B1:15 B2:16*
NOTE PINS B3:17 S0:4 S1:3 S2:2 SALA:12 SALB:13 OE:1*
L1944 01101011111111111111111111111111101111*
L1980 01011011111111111111111111110111111*
L2016 01100111111111111111111111111011111111*
L2052 01010111111111111111111111101111111111*
L2232 11111111111111111111111101111111111111*
L2268 01101010101111111111111111111111111111*
L2304 01011010111011111111111111111111111111*
L2340 01100110111110111111111111111111111111*
L2376 01010110111111101111111111111111111111*
L2412 01101001111111111011111111111111111111*
L2448 01011001111111111101111111111111111111*
L2484 01100101111111111111111111111111111110*
L2520 01010101111111111111111111111111111011*
L2556 11111111111111111111101111111111111111*
L2592 11111111111111111111111111111111111111*
L2628 11111111111111111111111111111111111111*
L2664 11111111111111111111111111111111111111*
L2700 11111111111111111111111111111111111111*
L2736 11111111111111111111111111111111111111*
L2772 11111111111111111111111111111111111111*
L2808 11111111111111111111111111111111111111*
L2844 11111111111111111111111111111111111111*
L2880 00000000000000000000000011001100*
L2912 11*
L2914 11*
V0001 100XXXXXXNXXL1000XXN*
V0002 000XXXXXXNXXL0001XXN*
```

V0003 000XXXXXXNXXH1000XXN*
V0004 010XXXXXXNXXL0000XXN*
V0005 010XXXXXXNXXH0100XXN*
V0006 001XXXXXXNXXL1000XXN*
V0007 001XXXXXXNXXH0010XXN*
V0008 011XXXXXXNXXL0100XXN*
V0009 011XXXXXXNXXH0001XXN*
V0010 100000000N0LXXXXX00N*
V0011 000000100N0LXXXXX00N*
V0012 000000100N0HXXXXX10N*
V0013 010000010N0LXXXXX00N*
V0014 010000001N0HXXXXX01N*
V0015 001000100N0LXXXXX00N*
V0016 001000010N1HXXXXX00N*
V0017 011000100N0LXXXXX00N*
V0018 011000001N0HXXXXX00N*
V0019 000101000N0LXXXXX00N*
V0020 000100010N0HXXXXX00N*
V0021 010110000N1LXXXXX00N*
V0022 010100100N0HXXXXX00N*
V0023 001100000N1LXXXXX00N*
V0024 001101000N1HXXXXX00N*
V0025 011100001N0LXXXXX00N*
V0026 011110010N0HXXXXX00N*
C5B28*
□EA79

DIAGRAMA DE CHIP



V. 4 CODIFICADOR DE DECIMAL A BINARIO

Un codificador es un circuito combinacional que convierte información escrita en algún código a otro distinto. En este caso se tiene un codificador cuyas entradas pueden tomar los valores de 0 a 9 decimal y entrega una salida en binario. En este caso se utilizó una tabla de verdad para describir el circuito, por ser la forma más sencilla de describir la lógica de este diseño, el cual se grabó en una GAL 20V8.

PROGRAMA EN **ABEL**

```
module decbin
title 'codificador de decimal a binario 0 a 9 con prioridad del numero menor
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores'
disp6 DEVICE 'G16V8';
A0,A1,A2,A3,A4,A5,A6,A7,A8,A9 PIN; "Entradas en decimal
OE PIN;"Entrada de habilitación activa en bajo
S0,S1,S2,S3 PIN ISTYPE 'com'; "Salidas en binario
EQUATIONS
OE.OE = OE;
TRUTH_TABLE ([A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, OE] -> [S0,S1,S2,S3])
        [.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X., 1] -> [ 0, 0, 0, 0];
        [ 1 ,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X., 0] -> [ 0, 0, 0, 0];
        [ 0 , 1 ,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X., 0] -> [ 0, 0, 0, 1];
        [ 0 , 0 , 1 ,.X.,.X.,.X.,.X.,.X.,.X.,.X., 0] -> [ 0, 0, 1, 0];
        [ 0 , 0 , 0 , 1 ,.X.,.X.,.X.,.X.,.X.,.X., 0] -> [ 0, 0, 1, 1];
        [ 0 , 0 , 0 , 0 , 1 ,.X.,.X.,.X.,.X.,.X., 0] -> [ 0, 1, 0, 0];
        [ 0 , 0 , 0 , 0 , 0 , 1 ,.X.,.X.,.X.,.X., 0] -> [ 0, 1, 0, 1];
        [ 0 , 0 , 0 , 0 , 0 , 0 , 1 ,.X.,.X.,.X., 0] -> [ 0, 1, 1, 0];
        [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 ,.X.,.X., 0] -> [ 0, 1, 1, 1];
        [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 ,.X., 0] -> [ 1, 0, 0, 0];
        [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0] -> [ 1, 0, 0, 1];
TEST_VECTORS ([A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,OE] -> [S0,S1,S2,S3])
        [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1] -> [ 0, 0, 0, 0];
        [ 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0] -> [ 0, 0, 0, 0];
        [ 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0] -> [ 0, 0, 0, 1];
        [ 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0] -> [ 0, 0, 1, 0];
        [ 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0] -> [ 0, 0, 1, 1];
        [ 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0] -> [ 0, 1, 0, 0];
        [ 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0] -> [ 0, 1, 0, 1];
        [ 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0] -> [ 0, 1, 1, 0];
        [ 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0] -> [ 0, 1, 1, 1];
        [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0] -> [ 1, 0, 0, 0];
        [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0] -> [ 1, 0, 0, 1];
END
```

ARCHIVO JEDEC

□ABEL-EDU 5.04.005 Data I/O Corp. JEDEC file for: G16V8

Created on: Sun Oct 19 22:19:03 1997

codificador de decimal a binario 0 a 9 con prioridad del numero menor

Celia Rosa Fierro Santillán y José Filiberto Lule Flores

*

QP20* QF2916* QV11* F0*

X0*

NOTE Table of pin names and numbers*

NOTE PINS A0:11 A1:9 A2:8 A3:7 A4:6 A5:5 A6:4 A7:3 A8:2 A9:1 OE:12*

NOTE PINS S0:13 S1:14 S2:15 S3:16*

L0972 10011101110111011101011111111111111111*

L1008 11111001110111011101011111111111111111*

L1044 11111111100111011101011111111111111111*

L1080 11111111111110011101011111111111111111*

L1116 11111111111111111001011111111111111111*

L1260 11111111111111111111111111111111111111*

L1296 11111011010111110101011111111111111111*

L1332 11111110010111110101011111111111111111*

L1368 11111111111110110101011111111111111111*

L1404 11111111111111100101011111111111111111*

L1584 11111111111111111111111111111111111111*

L1620 11111011111010101010111111111111111111*

L1656 11111101111010101010111111111111111111*

L1692 11111111101101010101011111111111111111*

L1728 11111111111001010101011111111111111111*

L1908 11111111111111111111111111111111111111*

L1944 10110101010101010101011111111111111111*

L1980 11100101010101010101011111111111111111*

L2232 11111111111111111111111111111111111111*

L2556 11111111111111111111011111111111111111*

L2592 11111111111111111111111111111111111111*

L2628 11111111111111111111111111111111111111*

L2664 11111111111111111111111111111111111111*

L2700 11111111111111111111111111111111111111*

L2736 11111111111111111111111111111111111111*

L2772 11111111111111111111111111111111111111*

L2808 11111111111111111111111111111111111111*

L2844 11111111111111111111111111111111111111*

L2880 00000000000011001100110011000000*

L2912 11*

L2914 11*

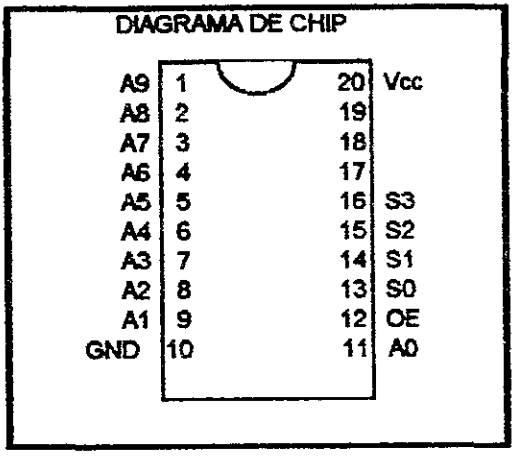
V0001 100000000N01LLLLXXXN*

V0002 000000001N10LLLLXXXN*

V0003 000000011N00LLLHXXXN*

V0004 000000110N00LLHLXXXN*

V0005 000001100N00LLHHXXXN*
V0006 000011000N00LHLLXXXN*
V0007 000110000N00LHLHXXXN*
V0008 001100000N00LHHLXXXN*
V0009 011000000N00LHHHXXXN*
V0010 110000000N00HLLLXXXN*
V0011 100000000N00HLLHXXXN*
C7643*
□ACE1



IV. 5 COMPARADOR

Un comparador es un circuito combinacional que determina si un número es mayor , menor o igual a otro número. En este caso se diseñó un comparador binario de cuatro bits en una GAL 20V8. La descripción lógica está hecha en forma de ecuaciones utilizando compuertas OR-EXCLUSIVAS, ya que una tabla de verdad sería demasiado grande y ocuparía más minitérminos de los que se tienen disponibles para cada salida. La lógica que se siguió fue la de comparar ambos números (A y B) partiendo del bit más significativo hasta el bit menos significativo, se utilizaron los nodos C0,C1,C2 y C3 para comparar cada pareja de bits de entrada.

PROGRAMA EN **ABEL**

```
module comp
title 'comparador de cuatro bits
    Celia Rosa Fierro Santillán y José Filiberto Luis Flores'
disp7 DEVICE 'G16V8';
A0,A1,A2,A3 PIN; "Entrada de un número A de cuatro bits
B0,B1,B2,B3 PIN; "Entrada de un número B de cuatro bits
OE PIN; "Entrada de habilitación
AMAYOR, IGUALES, BMAYOR PIN ISTYPE 'com';"Salidas del comparador
C0,C1,C2,C3 NODE ISTYPE 'COM'; "Nodos de comparación
```

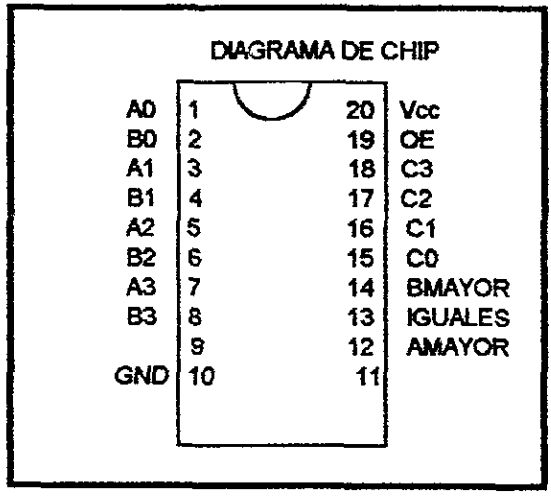
```
EQUATIONS
OE.OE = OE;
C0 = !(A0 $ B0);
C1 = !(A1 $ B1);
C2 = !(A2 $ B2);
C3 = !(A3 $ B3);
IGUALES = C0 & C1 & C2 & C3 & !OE;
AMAYOR =(A0&!B0 # A1&!B1&C0 # A2&!B2&C0&C1 # A3&!B3&C0&C1&C2)&!OE;
BMAYOR =(!A0&B0 # !A1&B1&C0 # !A2&B2&C0&C1 # !A3&B3&C0&C1&C2)&!OE;
TEST_VECTORS ([A0,A1,A2,A3,B0,B1,B2,B3,OE] -> [AMAYOR,IGUALES,BMAYOR])
                [ 0, 0, 0, 0, 0, 0, 0, 0, 1] -> [0,0,0];
                [ 0, 0, 0, 0, 0, 0, 0, 1, 0] -> [0,0,1];
                [ 0, 1, 0, 0, 0, 1, 0, 0, 0] -> [0,1,0];
                [ 1, 1, 0, 0, 1, 0, 0, 0, 0] -> [1,0,0];
                [ 1, 1, 1, 0, 1, 0, 1, 0, 0] -> [1,0,0];
                [ 1, 1, 0, 1, 1, 1, 1, 0, 0] -> [0,0,1];
                [ 1, 0, 1, 0, 1, 0, 1, 1, 0] -> [0,0,1];
                [ 1, 0, 0, 1, 1, 0, 0, 1, 0] -> [0,1,0];
                [ 1, 1, 1, 0, 1, 1, 1, 1, 0] -> [0,0,1];
                [ 0, 0, 0, 1, 0, 0, 0, 0, 0] -> [1,0,0];
END
```

ARCHIVO JEDEC

```
□ABEL-EDU 5.04.005 Data I/O Corp. JEDEC file for: G16V8
Created on: Mon Oct 27 20:48:17 1997
comparador de cuatro bits
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores
*
QP20* QF2916* QV10* F0*
 X0*
NOTE Table of pin names and numbers*
NOTE PINS A0:1 A1:3 A2:5 A3:7 B0:2 B1:4 B2:6 B3:8 OE:19 AMAYOR:12*
NOTE PINS IGUALES:13 BMAYOR:14*
NOTE Table of node names and numbers*
NOTE NODES C0:15 C1:16 C2:17 C3:18*
L0288 11111111111111111111111111111111110*
L0324 11111111111110101111111111111111111*
L0360 11111111111101011111111111111111111*
L0612 11111111111111111111111111111111111*
L0648 11111111101011111111111111111111111*
L0684 11111111010111111111111111111111111*
L0936 11111111111111111111111111111111111*
L0972 11111010111111111111111111111111111*
L1008 11110101111111111111111111111111111*
L1260 11111111111111111111111111111111111*
L1296 10101111111111111111111111111111111*
L1332 01011111111111111111111111111111111*
L1584 11111111111111111111111111111111111*
L1620 01101111111111111111111111111111101*
```

L1656 11110110111111111111111111011111101*
L1692 11111111011011111111111111010111101*
L1728 11111111111101101111111111010101101*
L1908 11111111111111111111111111111111111*
L1944 11111111111111111111111111010101001*
L2232 11111111111111111111111111111111111*
L2268 10011111111111111111111111111111101*
L2304 11111001111111111111111111011111101*
L2340 11111111100111111111111111010111101*
L2376 11111111111110011111111111010101101*
L2556 11111111111111111111111111111111111*
L2592 11111111111111111111111111111111111*
L2628 11111111111111111111111111111111111*
L2664 11111111111111111111111111111111111*
L2700 11111111111111111111111111111111111*
L2736 11111111111111111111111111111111111*
L2772 11111111111111111111111111111111111*
L2808 11111111111111111111111111111111111*
L2844 11111111111111111111111111111111111*
L2880 00001100110011001100110011001100*
L2912 11*
L2914 11*
V0001 00000000XNXLLLXXXX1N*
V0002 00000001XNXLLHXXXX0N*
V0003 00110000XNXLHLXXXX0N*
V0004 11100000XNXHLLXXXX0N*
V0005 11101100XNXHLLXXXX0N*
V0006 11110110XNXLLHXXXX0N*
V0007 11001101XNXLLHXXXX0N*
V0008 11000011XNXLHLXXXX0N*
V0009 11111101XNXLLHXXXX0N*
V0010 00000010XNXHLLXXXX0N*
C8B01*
□DA8E

DIAGRAMA DE CHIP



IV.6 SUMADOR

Un sumador es un circuito combinacional que forma la suma aritmética de los bits de entrada. En este caso se diseñó un sumador binario de cuatro bits con acarreo de entrada y acarreo de salida. La descripción lógica se hizo por medio de ecuaciones lógicas, utilizando compuertas OR - EXCLUSIVAS, ya que para este ejemplo una tabla de verdad sería muy extensa, se utilizaron los nodos C1, C2 y C3 como acarreos internos del sumador.

PROGRAMA EN **ABEL**

```
module suma
title 'Sumador de 4 bits con acarreo
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores'
disp7 DEVICE 'G16V8';
A0,A1,A2,A3 PIN; "Entradas del numero A de 4 bits
B0,B1,B2,B3 PIN; "Entradas del numero B de 4 bits
C PIN; "Entrada de acarreo
S0,S1,S2,S3 PIN ISTYPE 'COM'; "Salidas del sumador
C0 PIN ISTYPE 'COM';"Salida de acarreo
C1,C2,C3 NODE ISTYPE 'COM'; "Acarreos internos del sumador
EQUATIONS
C3 = A3 & B3 + C  & (A3 $ B3);
C2 = A2 & B2 + C3 & (A2 $ B2);
C1 = A1 & B1 + C2 & (A1 $ B1);
C0 = A0 & B0 + C1 & (A0 $ B0);
S3 = A3 $ B3 $ C ;
S2 = A2 $ B2 $ C3;
S1 = A1 $ B1 $ C2;
S0 = A0 $ B0 $ C1;
TEST_VECTORS ([A0,A1,A2,A3,B0,B1,B2,B3,C ] -> [C0,S0,S1,S2,S3])
        [ 1, 1, 1, 1, 1, 1, 1, 1, 0] -> [ 1, 1, 1, 1, 0];
        [ 1, 1, 1, 1, 0, 1, 1, 1, 0] -> [ 1, 0, 1, 1, 0];
        [ 1, 1, 1, 0, 0, 1, 1, 1, 1] -> [ 1, 0, 1, 1, 0];
        [ 1, 1, 1, 0, 0, 1, 1, 0, 0] -> [ 1, 0, 1, 0, 0];
        [ 1, 1, 1, 0, 1, 0, 1, 1, 1] -> [ 1, 1, 0, 1, 0];
        [ 1, 1, 1, 0, 1, 1, 1, 0, 0] -> [ 1, 1, 1, 0, 0];
        [ 1, 1, 0, 1, 1, 0, 1, 1, 1] -> [ 1, 1, 0, 0, 1];
END
```

ARCHIVO JEDEC

□ABEL-EDU 5.04.005 Data I/O Corp. JEDEC file for: G16V8

Created on: Mon Oct 27 21:06:19 1997

Sumador de 4 bits con acarreo

Celia Rosa Fierro Santillán y José Filiberto Luis Flores

*

QP20* QF2916* QV7* F0*

X0*

NOTE Table of pin names and numbers*

NOTE PINS A0:8 A1:6 A2:4 A3:1 B0:9 B1:7 B2:5 B3:2 C:3 S0:12 S1:13*

NOTE PINS S2:14 S3:15 C0:16*

NOTE Table of node names and numbers*

NOTE NODES C1:17 C2:18 C3:19*

L0000 10101111111111111111111111111111111111*

L0036 10111011111111111111111111111111111111*

L0072 11101011111111111111111111111111111111*

L0288 11111111111111111111111111111111111111*

L0324 11111110101111111111111111111111111111*

L0360 11111110111111111111111111111111111110*

L0396 11111111101111111111111111111111111110*

L0612 11111111111111111111111111111111111111*

L0648 11111111111010111111111111111111111111*

L0684 11111111111011111111111111111111111011*

L0720 11111111111110111111111111111111111011*

L0936 11111111111111111111111111111111111111*

L0972 11111111111111101011111111111111111111*

L1008 11111111111111101111111111111111101111*

L1044 11111111111111111011111111111111101111*

L1260 11111111111111111111111111111111111111*

L1296 10101011111111111111111111111111111111*

L1332 01011011111111111111111111111111111111*

L1368 01100111111111111111111111111111111111*

L1404 10010111111111111111111111111111111111*

L1584 11111111111111111111111111111111111111*

L1620 11111110101111111111111111111111111110*

L1656 11111101011111111111111111111111111110*

L1692 11111101101111111111111111111111111101*

L1728 11111110011111111111111111111111111101*

L1908 11111111111111111111111111111111111111*

L1944 11111111111010111111111111111111111011*

L1980 11111111110101111111111111111111111011*

L2016 11111111110110111111111111111111110111*

L2052 11111111111001111111111111111111110111*

L2232 11111111111111111111111111111111111111*

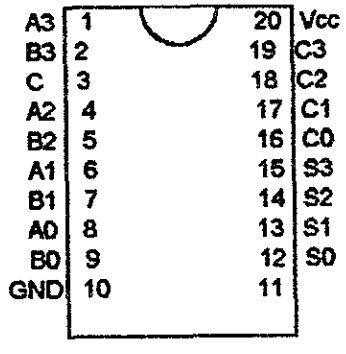
L2268 11111111111111101011111111111111101111*

L2304 11111111111111101011111111111111101111*

L2340 11111111111111101101111111111111011111*

L2376 11111111111111100111111111111011111*
L2556 11111111111111111111111111111111111*
L2592 11111111111111111111111111111111111*
L2628 11111111111111111111111111111111111*
L2664 11111111111111111111111111111111111*
L2700 11111111111111111111111111111111111*
L2736 11111111111111111111111111111111111*
L2772 11111111111111111111111111111111111*
L2808 11111111111111111111111111111111111*
L2844 11111111111111111111111111111111111*
L2880 11001100110011001100110011001100*
L2912 11*
L2914 11*
V0001 110111111NXHHHLHXXXN*
V0002 110111110NXLHHLHXXXN*
V0003 011111110NXLHHLHXXXN*
V0004 000111110NXLHLLHXXXN*
V0005 011111011NXHLHLHXXXN*
V0006 000111111NXHHLLHXXXN*
V0007 111011011NXHLLHHXXXN*
CBC4C*
□1DE0

DIAGRAMA DE CHIP



IV.7 RESTADOR

La resta o sustracción de números binarios se puede hacer por el método del complemento a dos. Este método consiste en calcular el complemento a dos de un número B y sumarlo a un número A. El complemento a dos de B se puede obtener determinando el complemento a uno (los unos se cambian por ceros y los ceros por unos) y sumando uno al bit menos significativo. En circuitos digitales el complemento a uno se realiza utilizando inversores y se agrega uno a la suma haciendo que el acarreo de entrada sea 1. En este diseño se utilizaron ecuaciones lógicas muy

similares a las que se usaron en el circuito sumador, con la única diferencia de que los 4 bits del número A se invierten y el acarreo de entrada es 1. El diseño se grabó en una GAL 16V8.

PROGRAMA EN **ABEL**

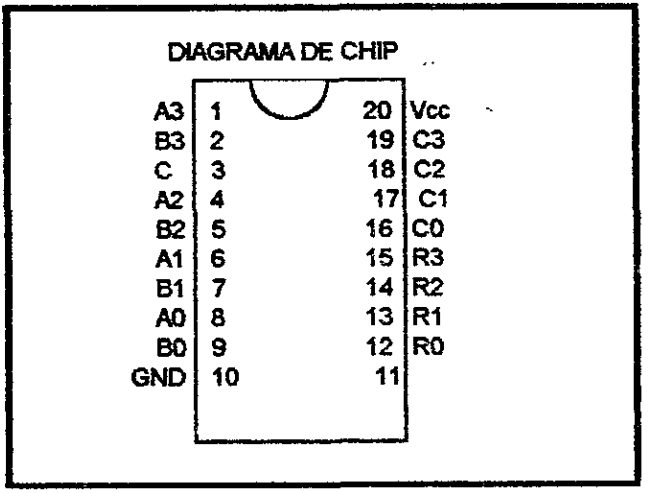
```
module resta
title 'Restador binario de cuatro bits
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores'
disp8 DEVICE 'g16v8';
A0,A1,A2,A3 PIN; "Numero binario A de cuatro bits (minuendo)
B0,B1,B2,B3 PIN; "Numero binario B de cuatro bits (sustraendo)
C PIN; "Acarreo de entrada
R0,R1,R2,R3 PIN ISTYPE 'COM'; "Número R de cuatro bits (resta)
C0 PIN ISTYPE 'COM'; "Acarreo de salida
C3,C2,C1 NODE ISTYPE 'COM'; "Acarreos internos del restador
EQUATIONS
C3 = !A3 & B3 # C  & (!A3 $ B3);
C2 = !A2 & B2 # C3 & (!A2 $ B2);
C1 = !A1 & B1 # C2 & (!A1 $ B1);
C0 = !A0 & B0 # C1 & (!A0 $ B0);
R3 = A3 $ B3 $ C ;
R2 = A2 $ B2 $ C3;
R1 = A1 $ B1 $ C2;
R0 = A0 $ B0 $ C1;
TEST_VECTORS ([A0,A1,A2,A3,B0,B1,B2,B3,C] -> [C0,R0,R1,R2,R3])
        [ 1, 0, 1, 0, 0, 0, 1, 1,0] -> [ 0, 0, 1, 1, 1];
        [ 0, 1, 1, 0, 0, 1, 0, 1,1] -> [ 0, 0, 0, 0, 0];
        [ 1, 0, 0, 1, 0, 1, 0, 1,0] -> [ 0, 0, 1, 0, 0];
        [ 0, 1, 0, 1, 0, 0, 1, 1,1] -> [ 0, 0, 0, 0, 1];
        [ 1, 1, 1, 0, 1, 1, 1, 1,0] -> [ 1, 1, 1, 1, 1];
        [ 0, 1, 1, 1, 0, 0, 1, 1,0] -> [ 0, 0, 1, 0, 0];
        [ 1, 0, 0, 0, 0, 0, 0, 0,0] -> [ 0, 1, 0, 0, 0];
END
```

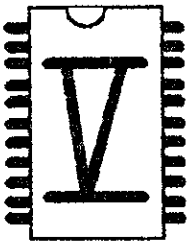
ARCHIVO JEDEC

```
□ABEL-EDU 5.04.005 Data I/O Corp. JEDEC file for: G16V8
Created on: Tue Oct 21 17:42:50 1997
Restador binario de cuatro bits
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores
*
QP20* QF2916* QV7* F0*
X0*
NOTE Table of pin names and numbers*
NOTE PINS A0:8 A1:6 A2:4 A3:1 B0:9 B1:7 B2:5 B3:2 C:3 R0:12 R1:13*
NOTE PINS R2:14 R3:15 C0:16*
NOTE Table of node names and numbers*
NOTE NODES C3:17 C2:18 C1:19*
L0000 11111111110110111111111111111111111*
```

```
L0036 11111111110111111111111111111111011*
L0072 111111111111101111111111111111111011*
L0288 111111111111111111111111111111111111*
L0324 111111011011111111111111111111111111*
L0360 111111011111111111111111111111101111*
L0396 111111111011111111111111111111101111*
L0612 111111111111111111111111111111111111*
L0648 011011111111111111111111111111111111*
L0684 011110111111111111111111111111111111*
L0720 111010111111111111111111111111111111*
L0936 111111111111111111111111111111111111*
L0972 111111111111110110111111111111111111*
L1008 1111111111111101111111111111111111110*
L1044 11111111111111111011111111111111111110*
L1260 111111111111111111111111111111111111*
L1296 101010111111111111111111111111111111*
L1332 010110111111111111111111111111111111*
L1368 011001111111111111111111111111111111*
L1404 100101111111111111111111111111111111*
L1584 111111111111111111111111111111111111*
L1620 111111101011111111111111111111101111*
L1656 111111010111111111111111111111101111*
L1692 111111011011111111111111111111011111*
L1728 111111100111111111111111111111011111*
L1908 111111111111111111111111111111111111*
L1944 111111111110101111111111111111111011*
L1980 111111111010111111111111111111111011*
L2016 111111111011011111111111111111110111*
L2052 111111111100111111111111111111110111*
L2232 111111111111111111111111111111111111*
L2268 1111111111111110101111111111111111110*
L2304 1111111111111101011111111111111111110*
L2340 111111111111110110111111111111111101*
L2376 111111111111111001111111111111111101*
L2556 111111111111111111111111111111111111*
L2592 111111111111111111111111111111111111*
L2628 111111111111111111111111111111111111*
L2664 111111111111111111111111111111111111*
L2700 111111111111111111111111111111111111*
L2736 111111111111111111111111111111111111*
L2772 111111111111111111111111111111111111*
L2808 111111111111111111111111111111111111*
L2844 111111111111111111111111111111111111*
L2880 110011001100110011001100110011001100*
L2912 11*
L2914 11*
V0001 010110010NXLHHHLXXXN*
```

V0002 011101100NXLLLLLXXXN*
V0003 110000110NXLHLLLXXXN*
V0004 111011000NXLLLHLXXXN*
V0005 010111111NXHHHHHXXXN*
V0006 110111000NXLHLLLXXXN*
V0007 000000010NXHLLLLXXXN*
CBD29*
□205D





CAPÍTULO V. DESCRIPCIÓN DE MÁQUINAS DE ESTADO MEDIANTE ABEL

V.1 CIRCUITOS SECUENCIALES Y MÁQUINAS DE ESTADO.

En los circuitos combinacionales, como los analizados en el capítulo anterior, las salidas dependen únicamente de las entradas al circuito; en cambio en un circuito secuencial las salidas dependen además de las retroalimentaciones de cada estado del circuito, para lo cual requieren de elementos de memoria.

Existen dos tipos de circuitos secuenciales: síncronos y asíncronos. Un circuito síncrono tiene un comportamiento definido por sus señales en instantes discretos de tiempo; en cambio, el comportamiento del asíncrono depende del orden en el cual cambian las señales de entrada en cualquier instante de tiempo. En la práctica se utilizan más los circuitos asíncronos.

En un circuito secuencial síncrono se utiliza un dispositivo llamado reloj maestro el cual genera un tren de pulsos periódicos que permiten sincronizar el circuito. Los pulsos de reloj se distribuyen de tal forma que los elementos de memoria pueden cambiar de estado solo cuando el pulso de reloj cambia de 0 a 1 lógico y mantienen este estado hasta que haya un nuevo pulso . Los elementos básicos de memoria en estos circuitos son flip-flops controlados por reloj

Un circuito asíncrono, a diferencia del anterior, no usa pulsos de reloj. El cambio del estado interno ocurre cuando hay un cambio en las variables de entrada. Los elementos básicos de memoria son flip-flops controlados por alguna de las señales de entrada. El cambio de estado del sistema ocurre inmediatamente después de los cambios de entrada.

Los circuitos secuenciales son llamados también máquinas de estado, por lo que se pueden utilizar los dos términos indistintamente y referirnos a ellos como máquinas de estado síncronas y asíncronas.

La secuencia en tiempo de las entradas y salidas en una máquina de estado se puede representar en forma gráfica en un diagrama de estado, un estado se representa con un círculo y la transición entre estados se indica con líneas dirigidas que los conectan (figura 5.1). El número binario dentro de cada círculo identifica que estado se representa en el . Las líneas dirigidas están etiquetadas por las variables de entrada que provocan la transición de un estado a otro y las salidas se

representan con el nombre de la variable de salida precedido por el símbolo ↓↑ . Una línea dirigida que conecta al círculo consigo mismo indica que no ocurre un cambio de estado. En **ABEL**, como se vio en el capítulo 1, se puede realizar un diseño a partir del diagrama de estado. En cada uno de los ejemplos que se desarrollarán en este capítulo se realizará el programa en **ABEL** a partir del diagrama de estados correspondiente.

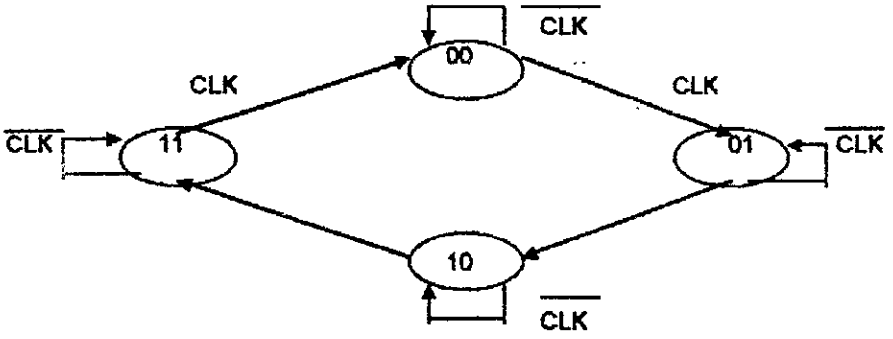


figura 5.1 Diagrama de estados de un contador binario de 0 a 3. La única entrada es el reloj CLK y no se tienen salidas, únicamente transiciones entre los estados.

V. 2 DESCRIPCIÓN DE MÁQUINAS DE ESTADO EN ABEL

En **ABEL** existen dos métodos para describir el funcionamiento de una máquina de estado: las tablas de transición de estados y los diagramas de estado . El método a utilizar depende de la complejidad de la máquina de estado que se está diseñando.

TABLAS DE TRANSICIÓN DE ESTADOS

Las tablas de verdad del lenguaje **ABEL** pueden ser utilizadas para describir máquinas de estado, particularmente aquellas que contienen un gran número de transiciones similares. Puede ser mas eficiente utilizar la tabla de transición de estados que el diagrama de estados cuando muchas de las salidas descritas en la tabla de verdad contienen condiciones de no importa .

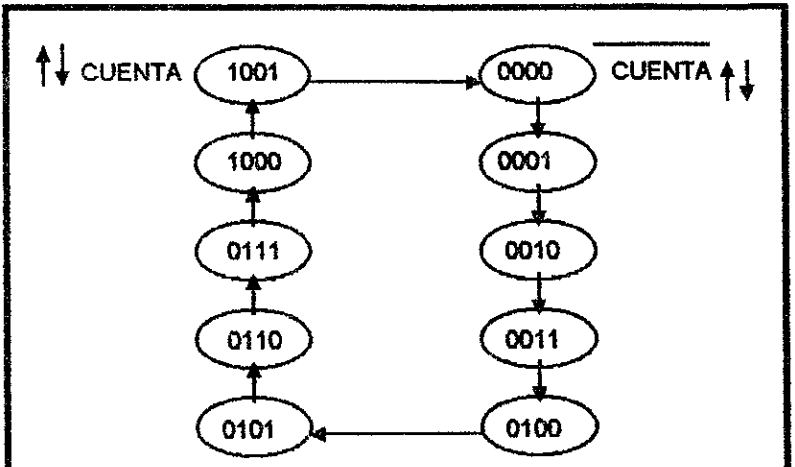


figura 5.2 Diagrama de estados de un contador binario de 0 a 9.

Para ejemplificar como una máquina de estados es descrita con una tabla de transición de estados consideremos el diagrama de estados de la figura 5.2 correspondiente a un contador binario de 0 a 9, que produce una salida CUENTA cuando completa un ciclo y $\overline{\text{CUENTA}}$ cuando inicia el ciclo. Se describe el funcionamiento de este contador en una tabla de verdad con todas las condiciones de transición posibles como se muestra en el programa fuente en **ABEL**. Cada línea de la tabla representa una combinación de entradas única. Las primeras 10 de ellas describen la transición lógica para la máquina de estados, y las últimas cuatro líneas describen lo que ocurre con la variable de salida CUENTA mientras la máquina se encuentra en los estados 0000 y 1001. El símbolo := es utilizado para asignar un valor a un registro o flip-flop. El símbolo :> en la tabla de transición de estados significa que para las entradas y el estado determinado se produce un cambio de estado, con sus respectivas salidas, las cuales se describen después del símbolo ->.

PROGRAMA EN **ABEL**

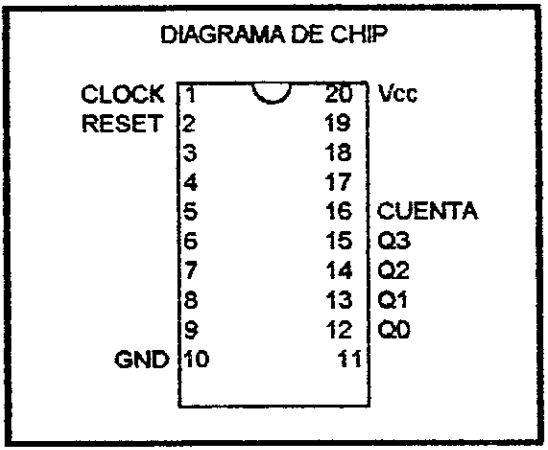
```
module conta1
title 'Contador binario de 0 a 9
      Celia Rosa Fierro Santillán y José Filiberto Lule Flores';
disp41 DEVICE 'g16v8';
CLOCK, RESET PIN; "Entradas del contador
Q0,Q1,Q2,Q3 PIN ISTYPE 'REG'; "Flip-flops
CUENTA PIN ISTYPE 'COM'; "Salida del contador cuando se completa un ciclo
```

```
EQUATIONS
        [Q0,Q1,Q2,Q3].clk = CLOCK;
TRUTH_TABLE ([CLOCK,RESET,Q0,Q1,Q2,Q3]:>[Q0,Q1,Q2,Q3]->[CUENTA])
            [     0 ,0,.X.,.X.,.X.,.X.]:>[ 0, 0, 0, 0]->[ 0];
            [     0 ,  1 , 0, 0, 0, 0]:>[ 0, 0, 0, 0]->[ 0];
            [     1 ,  1 , 0, 0, 0, 0]:>[ 0, 0, 0, 1]->[ 0];
            [     0 ,  1 , 0, 0, 0, 1]:>[ 0, 0, 0, 1]->[ 0];
            [     1 ,  1 , 0, 0, 0, 1]:>[ 0, 0, 1, 0]->[ 0];
            [     0 ,  1 , 0, 0, 1, 0]:>[ 0, 0, 1, 0]->[ 0];
            [     1 ,  1 , 0, 0, 1, 0]:>[ 0, 0, 1, 1]->[ 0];
            [     0 ,  1 , 0, 0, 1, 1]:>[ 0, 0, 1, 1]->[ 0];
            [     1 ,  1 , 0, 0, 1, 1]:>[ 0, 1, 0, 0]->[ 0];
            [     0 ,  1 , 0, 1, 0, 0]:>[ 0, 1, 0, 0]->[ 0];
            [     1 ,  1 , 0, 1, 0, 0]:>[ 0, 1, 0, 1]->[ 0];
            [     0 ,  1 , 0, 1, 0, 1]:>[ 0, 1, 0, 1]->[ 0];
            [     1 ,  1 , 0, 1, 0, 1]:>[ 0, 1, 1, 0]->[ 0];
            [     0 ,  1 , 0, 1, 1, 0]:>[ 0, 1, 1, 0]->[ 0];
            [     1 ,  1 , 0, 1, 1, 0]:>[ 0, 1, 1, 1]->[ 0];
            [     0 ,  1 , 0, 1, 1, 1]:>[ 0, 1, 1, 1]->[ 0];
            [     1 ,  1 , 0, 1, 1, 1]:>[ 1, 0, 0, 0]->[ 0];
            [     0 ,  1 , 1, 0, 0, 0]:>[ 1, 0, 0, 0]->[ 0];
            [     1 ,  1 , 1, 0, 0, 0]:>[ 1, 0, 0, 1]->[ 0];
            [     0 ,  1 , 1, 0, 0, 1]:>[ 1, 0, 0, 1]->[ 1];
            [     1 ,  1 , 1, 0, 0, 1]:>[ 0, 0, 0, 0]->[ 1];
TEST_VECTORS ([CLOCK,RESET]->[Q0,Q1,Q2,Q3,CUENTA])
            [     0   ,  0  ]->[ 0, 0, 0, 0, 0 ];
            [     1   ,  0  ]->[ 0, 0, 0, 0, 0 ];
            [     0   ,  1  ]->[ 0, 0, 0, 0, 0 ];
            [     1   ,  1  ]->[ 0, 0, 0, 1, 0 ];
            [     0   ,  1  ]->[ 0, 0, 0, 1, 0 ];
            [     1   ,  1  ]->[ 0, 0, 1, 0, 0 ];
            [     0   ,  1  ]->[ 0, 0, 1, 0, 0 ];
            [     1   ,  1  ]->[ 0, 0, 1, 1, 0 ];
            [     0   ,  1  ]->[ 0, 0, 1, 1, 0 ];
            [     1   ,  1  ]->[ 0, 1, 0, 0, 0 ];
            [     0   ,  1  ]->[ 0, 1, 0, 0, 0 ];
            [     1   ,  1  ]->[ 0, 1, 0, 1, 0 ];
            [     0   ,  1  ]->[ 0, 1, 0, 1, 0 ];
            [     1   ,  1  ]->[ 0, 1, 1, 0, 0 ];
            [     0   ,  1  ]->[ 0, 1, 1, 0, 0 ];
            [     1   ,  1  ]->[ 0, 1, 1, 1, 0 ];
            [     0   ,  1  ]->[ 0, 1, 1, 1, 0 ];
            [     1   ,  1  ]->[ 1, 0, 0, 0, 0 ];
            [     0   ,  1  ]->[ 1, 0, 0, 0, 0 ];
            [     1   ,  1  ]->[ 1, 0, 0, 1, 1 ];
            [     0   ,  1  ]->[ 1, 0, 0, 1, 1 ];
            [     1   ,  1  ]->[ 0, 0, 0, 0, 0 ];
            [     0   ,  1  ]->[ 0, 0, 0, 0, 0 ];
            END
```

ARCHIVO JEDEC

```
□ABEL-EDU 5.03.005 Data I/O Corp. JEDEC file for: G16V8 V9.0
Created on: Fri Oct 31 12:54:47 1997
Contador binario de 0 a 9
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores
*
QP20* QF2916* QV23* F0*
 X0*
NOTE Table of pin names and numbers*
NOTE PINS CLOCK:1 RESET:2 Q0:12 Q1:13 Q2:14 Q3:15 CUENTA:16*
L0972 11101111111111111111110010110111111111*
L1260 11111111111111111111111111111111111111*
L1296 01101111111111111111111010110111111111*
L1332 10101111111111111111111010101111111111*
L1368 01101111111111111111110111111011111111*
L1404 10101111111111111111110111111011111111*
L1584 11111111111111111111111111111111111111*
L1620 01101111111111111111101111011111111111*
L1656 11101111111111111111101111001111111111*
L1692 10101111111111111111101110110111111111*
L1908 11111111111111111111111111111111111111*
L1944 01101111111111111111101101111111111111*
L1980 11101111111111111111101101101111111111*
L2016 11101111111111111111101100111111111111*
L2052 10101111111111111111101011010111111111*
L2232 11111111111111111111111111111111111111*
L2268 01101111111111111111100101111111111111*
L2304 11101111111111111111100101011111111111*
L2340 10101111111111111111101101010111111111*
L2556 11111111111111111111111111111111111111*
L2592 11111111111111111111111111111111111111*
L2628 11111111111111111111111111111111111111*
L2664 11111111111111111111111111111111111111*
L2700 11111111111111111111111111111111111111*
L2736 11111111111111111111111111111111111111*
L2772 11111111111111111111111111111111111111*
L2808 11111111111111111111111111111111111111*
L2844 11111111111111111111111111111111111111*
L2880 00000000000011001111111111111111*
L2912 11*
L2914 11*
V0001 00XXXXXXXNXLLLLLXXXN*
V0002 10XXXXXXXNXLLLLLXXXN*
V0003 01XXXXXXXNXLLLLLXXXN*
V0004 11XXXXXXXNXLLLHLXXXN*
V0005 01XXXXXXXNXLLLHLXXXN*
V0006 11XXXXXXXNXLLHLLXXXN*
V0007 01XXXXXXXNXLLHLLXXXN*
V0008 11XXXXXXXNXLLHHLXXXN*
V0009 01XXXXXXXNXLLHHLXXXN*
```

```
V0010 11XXXXXXXNXLHLLLXXXN*
V0011 01XXXXXXXNXLHLLLXXXN*
V0012 11XXXXXXXNXLHLHLXXXN*
V0013 01XXXXXXXNXLHLHLXXXN*
V0014 11XXXXXXXNXLHHLLXXXN*
V0015 01XXXXXXXNXLHHLLXXXN*
V0016 11XXXXXXXNXLHHHLXXXN*
V0017 01XXXXXXXNXLHHHLXXXN*
V0018 11XXXXXXXNXHLLLLXXXN*
V0019 01XXXXXXXNXHLLLLXXXN*
V0020 11XXXXXXXNXHLLHHXXXN*
V0021 01XXXXXXXNXHLLHHXXXN*
V0022 11XXXXXXXNXLLLLLXXXN*
V0023 01XXXXXXXNXLLLLLXXXN*
C7687*
□0062
```



DIAGRAMAS DE ESTADO

Un diagrama de estados es un método relativamente sencillo para describir una máquina de estados . La diferencia entre un diagrama de estados y una tabla de transición es que en el primero se describe la máquina de estado en función de sus posibles estados, y en la segunda en términos de sus transiciones. Cuando se tienen máquinas de estado complejas pude resultar más sencillo describir su funcionamiento a través de un diagrama de estados.

Para ejemplificar como se describe un circuito secuencial a través de un diagrama de estados se realizó un contador binario de 0 a 9, similar al descrito en el apartado anterior, solo que la entrada de reloj es la variable CUENTA producida por el otro contador, y al completar un ciclo se genera una variable de salida

CUENTA1, teniendo dos contadores en cascada que pueden contar de 0 a 99. Esta aplicación se utilizará posteriormente en el capítulo seis.

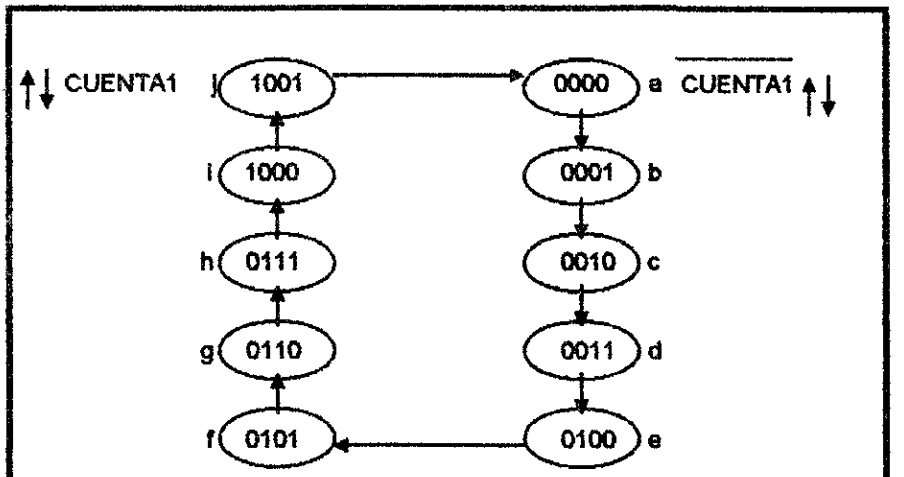


figura 5.3 Diagrama de estados de un contador binario de 0 a 9.

Cada uno de los estados se designa con una letra para mayor sencillez. En el programa se describe lo que sucede en cada estado, el cual, dependiendo de las *entradas puede pasar a otro estado o permanecer en el mismo*, además de establecer el valor de la salida CUENTA1 cuando es necesario. La sección del programa correspondiente al diagrama de estados se inicia con el encabezado **STATE_DIAGRAM**, seguido de la variable QESTADO que designa al vector [Q0,Q1,Q2,Q3] (variables de estado). Cada estado se inicia con su nombre después del enunciado **STATE** y puede tener uno o más enunciados de transición para describir las los saltos en la máquina de estados. Se utilizan palabras reservadas que establecen ciclos o condiciones tales como **While, if ... then...else** y **case**. En este ejemplo se utilizan condiciones del tipo **if...then ...else** para realizar las transiciones de un estado a otro.

PROGRAMA FUENTE EN **ABEL**

```
module conta2
title 'Contador binario de 0 a 9
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores';
disp42 DEVICE 'g16v8';
CUENTA, RESET PIN; "Entradas del contador
Q0,Q1,Q2,Q3 PIN ISTYPE 'REG'; "Flip-flops
CUENTA1 PIN ISTYPE 'COM'; "Salida del contador cuando se completa un ciclo
```

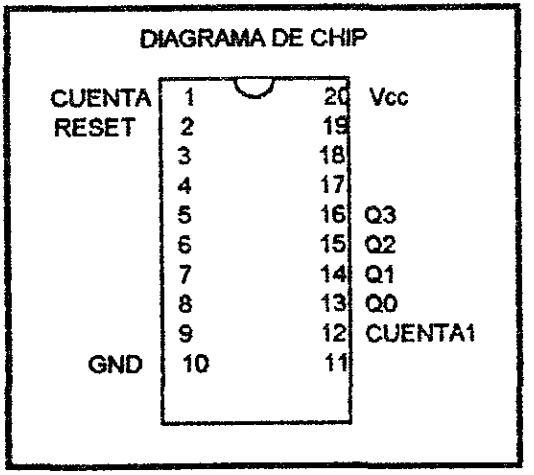
```
QESTADO = [Q0,Q1,Q2,Q3];
    A = [ 0, 0, 0, 0];
    B = [ 0, 0, 0, 1];
    C = [ 0, 0, 1, 0];
    D = [ 0, 0, 1, 1];
    E = [ 0, 1, 0, 0];
    F = [ 0, 1, 0, 1];
    G = [ 0, 1, 1, 0];
    H = [ 0, 1, 1, 1];
    I  = [ 1, 0, 0, 0];
    J  = [ 1, 0, 0, 1];
EQUATIONS
        [Q0,Q1,Q2,Q3].clk = CUENTA;
STATE_DIAGRAM QESTADO
STATE A:
        CUENTA1 = 0;
        IF !RESET THEN A
        ELSE
        IF CUENTA THEN B
        ELSE A;
STATE B:
        IF !RESET THEN A
        ELSE
        IF CUENTA THEN C
        ELSE B;
STATE C:
        IF !RESET THEN A
        ELSE
        IF CUENTA THEN D
        ELSE C;
STATE D:
        IF !RESET THEN A
        ELSE
        IF CUENTA THEN E
        ELSE D;
STATE E:
        IF !RESET  THEN A
        ELSE
        IF CUENTA THEN F
        ELSE E;
STATE F:
        IF !RESET  THEN A
        ELSE
        IF CUENTA THEN G
        ELSE F;
STATE G:
        IF !RESET  THEN A
        ELSE
        IF CUENTA THEN H
        ELSE G;
```

```
STATE H:
        IF !RESET  THEN A
        ELSE
        IF CUENTA THEN I
        ELSE H;
STATE I:
        IF !RESET THEN A
        ELSE
        IF CUENTA THEN J
        ELSE I;
STATE J:
        CUENTA1 = 1;
        IF !RESET THEN A
        ELSE
        IF CUENTA THEN A
        ELSE J;
TEST_VECTORS ([CUENTA,RESET,Q0,Q1,Q2,Q3]->[Q0,Q1,Q2,Q3,CUENTA1])
                [       0 ,  0 , 0, 1, 1, 0]->[ 0, 0, 0, 0, 0];
                [       0 ,  1 , 0, 0, 0, 0]->[ 0, 0, 0, 0, 0];
                [       1 ,  1 , 0, 0, 0, 0]->[ 0, 0, 0, 1, 0];
                [       0 ,  1 , 0, 0, 0, 1]->[ 0, 0, 0, 1, 0];
                [       1 ,  1 , 0, 0, 0, 1]->[ 0, 0, 1, 0, 0];
                [       0 ,  1 , 0, 0, 1, 0]->[ 0, 0, 1, 0, 0];
                [       1 ,  1 , 0, 0, 1, 0]->[ 0, 0, 1, 1, 0];
                [       0 ,  1 , 0, 0, 1, 1]->[ 0, 0, 1, 1, 0];
                [       1 ,  1 , 0, 0, 1, 1]->[ 0, 1, 0, 0, 0];
                [       0 ,  1 , 0, 1, 0, 0]->[ 0, 1, 0, 0, 0];
                [       1 ,  1 , 0, 1, 0, 0]->[ 0, 1, 0, 1, 0];
                [       0 ,  1 , 0, 1, 0, 1]->[ 0, 1, 0, 1, 0];
                [       1 ,  1 , 0, 1, 0, 1]->[ 0, 1, 1, 0, 0];
                [       0 ,  1 , 0, 1, 1, 0]->[ 0, 1, 1, 0, 0];
                [       1 ,  1 , 0, 1, 1, 0]->[ 0, 1, 1, 1, 0];
                [       0 ,  1 , 0, 1, 1, 1]->[ 0, 1, 1, 1, 0];
                [       1 ,  1 , 0, 1, 1, 1]->[ 1, 0, 0, 0, 0];
                [       0 ,  1 , 1, 0, 0, 0]->[ 1, 0, 0, 0, 0];
                [       1 ,  1 , 1, 0, 0, 0]->[ 1, 0, 0, 1, 1];
                [       0 ,  1 , 1, 0, 0, 1]->[ 1, 0, 0, 1, 1];
                [       1 ,  1 , 1, 0, 0, 1]->[ 0, 0, 0, 0, 0];
                [       0 ,  1 , 1, 0, 0, 0]->[ 0, 0, 0, 0, 0];
END
```

ARCHIVO JEDEC

```
□ABEL-EDU 5.03.005 Data I/O Corp. JEDEC file for: G16V8 V9.0
Created on: Fri Oct 31 13:57:51 1997
Contador binario de 0 a 9
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores
*
QP20* QF2916* QV22* F0*
 X0*
NOTE Table of pin names and numbers*
NOTE PINS CUENTA:1 RESET:2 Q0:13 Q1:14 Q2:15 Q3:16 CUENTA1:12*
L0972 01101111111111111111101111110111111*
L1008 01101111111111111111111111010110111111*
L1044 10101111111111111111111101111101111111*
L1080 10101111111111111111111110101011111111*
L1260 11111111111111111111111111111111111111*
L1296 01101111111111111111111011110111111111*
L1332 10101111111111111111111011101101111111*
L1368 11101111111111111111111011110011111111*
L1584 11111111111111111111111111111111111111*
L1620 01101111111111111111111011011111111111*
L1656 11101111111111111111111011001111111111*
L1692 10101111111111111111111010110101111111*
L1728 11101111111111111111111011011011111111*
L1908 11111111111111111111111111111111111111*
L1944 01101111111111111111111100101111111111*
L1980 10101111111111111111111011010101111111*
L2016 11101111111111111111111100101011111111*
L2232 11111111111111111111111111111111111111*
L2268 11111111111111111111111100101101111111*
L2556 11111111111111111111111111111111111111*
L2592 11111111111111111111111111111111111111*
L2628 11111111111111111111111111111111111111*
L2664 11111111111111111111111111111111111111*
L2700 11111111111111111111111111111111111111*
L2736 11111111111111111111111111111111111111*
L2772 11111111111111111111111111111111111111*
L2808 11111111111111111111111111111111111111*
L2844 11111111111111111111111111111111111111*
L2880 00000000000011111111111111111111100*
L2912 11*
L2914 11*
V0001 00XXXXXXXNXL0110XXXN*
V0002 01XXXXXXXNXL0000XXXN*
V0003 11XXXXXXXNXL0000XXXN*
V0004 01XXXXXXXNXL0001XXXN*
V0005 11XXXXXXXNXL0001XXXN*
V0006 01XXXXXXXNXL0010XXXN*
V0007 11XXXXXXXNXL0010XXXN*
```

V0008 01XXXXXXXNXL0011XXXN*
V0009 11XXXXXXXNXL0011XXXN*
V0010 01XXXXXXXNXL0100XXXN*
V0011 11XXXXXXXNXL0100XXXN*
V0012 01XXXXXXXNXL0101XXXN*
V0013 11XXXXXXXNXL0101XXXN*
V0014 01XXXXXXXNXL0110XXXN*
V0015 11XXXXXXXNXL0110XXXN*
V0016 01XXXXXXXNXL0111XXXN*
V0017 11XXXXXXXNXL0111XXXN*
V0018 01XXXXXXXNXL1000XXXN*
V0019 11XXXXXXXNXH1000XXXN*
V0020 01XXXXXXXNXH1001XXXN*
V0021 11XXXXXXXNXL1001XXXN*
V0022 01XXXXXXXNXL1000XXXN*
C7743*
□F024



V.3 OPTIMIZACIÓN DE LAS MÁQUINAS DE ESTADO

En algunos diseños resulta más fácil utilizar alguno de los dos métodos de descripción de máquinas de estado explicados en el apartado anterior. pero en algunos casos puede ser que se produzcan tablas de transición o diagramas de estado demasiado largos y complejos. en estos casos es mejor utilizar una combinación de ambos.

Cuando el número de estados que se requieren es muy grande la lógica no puede ser grabada en un solo circuito integrado (GAL o PAL), en tal caso se puede utilizar un PLD de alta densidad (PAL o MACH) o hacer una partición de la

máquina de estados para grabar todo el diseño en dos o más circuitos integrados. A continuación se explican algunos métodos para optimizar la descripción de máquinas de estado en **ABEL**. El uso de circuitos MACH se explicará en el siguiente capítulo.

USO DE TABLA DE VERDAD Y DIAGRAMA DE ESTADOS

En algunas ocasiones una máquina de estados puede tener varias salidas que se utilizan en todos o la mayoría de los estados, en este caso es conveniente utilizar una descripción lógica basada en tabla de verdad y diagrama de estados. En el diagrama de estados se establecen las condiciones necesarias para pasar de un estado a otro o para permanecer en el mismo estado. En la tabla de verdad se establecen las salidas que tiene cada estado.

Para ejemplificar lo anterior se realizó otra versión del ejemplo mostrado en el capítulo 1, en el cual tenemos un contador binario de 0 a 9 con un codificador de binario a siete segmentos. El diagrama a bloques de este diseño se muestra en la figura 5.5.

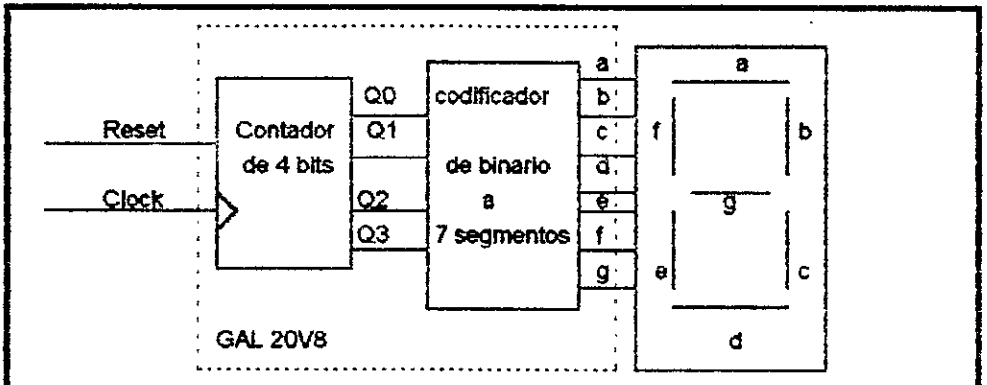


figura 5.4 contador binario de 0 a 9 con decodificador de binario a siete segmentos

El diagrama de estados del programa en **ABEL** contiene únicamente las transiciones de los diez estados y en la tabla de verdad correspondiente se establecen los valores para las salidas a,b,c,d,e,f y g en cada estado.

PROGRAMA EN **ABEL**

```
module conta3
title 'contador y codificador de binario a siete segmentos salida activa en bajo usando
tabla de verdad y diagrama de estados
     Celia Rosa Fierro Santillán y José Filiberto Lule Flores';
RESET, CLOCK  pin; "Entradas
Q0,Q1,Q2,Q3 node istype 'reg';
a,b,c,d,e,f,g pin istype 'com';
ENTRADAS = [RESET, CLOCK];
SALIDAS = [a,b,c,d,e,f,g];
ESTADO= [Q0,Q1,Q2,Q3];
    A = [ 0, 0, 0, 0];
    B = [ 0, 0, 0, 1];
    C = [ 0, 0, 1, 0];
    D = [ 0, 0, 1, 1];
    E = [ 0, 1, 0, 0];
    F = [ 0, 1, 0, 1];
    G = [ 0, 1, 1, 0];
    H = [ 0, 1, 1, 1];
    I = [ 1, 0, 0, 0];
    J = [ 1, 0, 0, 1];
EQUATIONS
ESTADO.CLK = CLOCK;
TRUTH_TABLE ([ESTADO, RESET] ->SALIDAS)
[.X..1] -> [0, 0, 0, 0, 0, 0, 1];
[ A ,0] -> [0, 0, 0, 0, 0, 0, 1];
[ B ,0] -> [1, 0, 0, 1, 1, 1, 1];
[ C ,0] -> [0, 1, 0, 0, 1, 0, 0];
[ D ,0] -> [0, 0, 0, 0, 1, 1, 0];
[ E ,0] -> [1, 0, 0, 1, 0, 1, 0];
[ F ,0] -> [0, 0, 1, 0, 0, 1, 0];
[ G ,0] -> [0, 0, 1, 1, 0, 0, 0];
[ H ,0] -> [0, 0, 0, 0, 1, 1, 1];
[ I ,0] -> [0, 0, 0, 0, 0, 0, 0];
[ J ,0] -> [0, 0, 0, 0, 1, 0, 0];
STATE_DIAGRAM (ESTADO)
STATE A:
        IF RESET THEN A ELSE
          IF CLOCK THEN B ELSE A;
STATE B:
        IF RESET THEN A ELSE
          IF CLOCK THEN C ELSE B;
STATE C:
        IF RESET THEN A ELSE
          IF CLOCK THEN D ELSE C;
STATE D:
        IF RESET THEN A ELSE
          IF CLOCK THEN E ELSE D;
```

```
STATE E:
        IF RESET THEN A ELSE
          IF CLOCK THEN F ELSE E;
STATE F:
        IF RESET THEN A ELSE
          IF CLOCK THEN G ELSE F;
STATE G:
        IF RESET THEN A ELSE
          IF CLOCK THEN H ELSE G;
STATE H:
        IF RESET THEN A ELSE
          IF CLOCK THEN I ELSE H;
STATE I:
        IF RESET THEN A ELSE
          IF CLOCK THEN J ELSE I;
STATE J:
        IF RESET THEN A ELSE
          IF CLOCK THEN A ELSE J;
TEST_VECTORS ([CLOCK, RESET,ESTADO] ->SALIDAS)
[.C.,1,D]->[0, 0, 0, 0, 0, 0, 1];
[.C.,0,A]->[1, 0, 0, 1, 1, 1, 1];
[.C.,0,B]->[0, 1, 0, 0, 1, 0, 0];
[.C.,0,C]->[0, 0, 0, 0, 1, 1, 0];
[.C.,0,D]->[1, 0, 0, 1, 0, 1, 0];
[.C.,0,E]->[0, 0, 1, 0, 0, 1, 0];
[.C.,0,F]->[0, 0, 1, 1, 0, 0, 0];
[.C.,0,G]->[0, 0, 0, 0, 1, 1, 1];
[.C.,0,H]->[0, 0, 0, 0, 0, 0, 0];
[.C.,0, I ]->[0, 0, 0, 0, 1, 0, 0];
[.C.,0,J ]->[0, 0, 0, 0, 0, 0, 1];
END
```

Como las salidas del contador Q0,Q1,Q2, y Q3 son entradas al codificador, en el programa se describieron como nodos internos del circuito, las salidas de los siete segmentos son activas en bajo, por lo que funcionan para un display de ánodo común. La entrada RESET, activa en alto, hace que el contador vuelva al estado 0000. En este ejemplo se utilizó la constante especial **.C.** en los vectores de prueba, esta constante es equivalente a un disparo de reloj, es decir, una transición de 0 a 1 en la señal correspondiente.

ECUACIONES DE ESTADO

El diagrama de bloques de la figura 5.4. corresponde a un contador binario y un codificador binario a siete segmentos, ambos en un mismo dispositivo. Otro método para describir el funcionamiento de este circuito es combinar la función del

codificador con la lógica de la máquina de estado. En este diseño la máquina de estado utiliza siete bits de estado, el valor de cada uno corresponde a las condiciones de salida del codificador. en la tabla de verdad se establecen las condiciones que tiene cada uno de ellos para en un estado determinado. En este caso no interesa el estado presente, solo se está tomando en cuenta el estado siguiente. Para lograr la transición de un estado a otro se utiliza la ecuación de registro:

ESTADO := (ESTADO.FB + 1) & (ESTADO.FB < 9) & !RESET;

El símbolo := indica que se trata de una ecuación de tipo registro. La extensión .FB significa que se utiliza la señal retroalimentada, a esta señal se le suma 1 para que la máquina de estado pase al estado siguiente , además se establece la condición de que el valor de el estado presente sea menor que nueve, ya que el máximo estado que se puede tener es nueve. Esta ecuación es válida solo cuando la señal RESET es cero, en caso contrario el circuito vuelve al estado 0000.

Los bits que determinan el valor de los estados (Q0,Q1,Q2 y Q3) fueron declarados como tipo 'dc,reg,invert'[15]. En un programa de **ABEL** se pueden utilizar varios tipos para un pin o nodo, en este caso que utilizaron tres de ellos.

El tipo dc significa que las ecuaciones para un pin o nodo pueden ser optimizadas utilizando condiciones de no importa. El tipo reg, corresponde a un pin o nodo de tipo registro y el tipo invert significa que la señal de salida del pin o nodo será invertida o negada.

PROGRAMA EN **ABEL**

```
module conta4
title 'contador binario de 0 a 9  y codificador de binario a siete segmentos
      salida activa en bajo usando tabla de verdad y ecuaciones
      Celia Rosa Fierro Santillán y José Filiberto Lule Flores';
RESET, CLOCK  pin; "Entradas
Q0,Q1,Q2,Q3 node istype 'dc,reg,invert';"Salidas del contador
a,b,c,d,e,f,g pin istype 'dc,com';
ESTADO= [Q0,Q1,Q2,Q3];
EQUATIONS
ESTADO.CLK = CLOCK;
ESTADO := (ESTADO.FB + 1) & (ESTADO.FB < 9) & !RESET;
TRUTH_TABLE ([ESTADO.FB] ->[a,b,c,d,e,f,g])
[ 0 ] -> [0, 0, 0, 0, 0, 0, 1];
[ 1 ] -> [1, 0, 0, 1, 1, 1, 1];
[ 2 ] -> [0, 1, 0, 0, 1, 0, 0];
```

[15] Ver los atributos de señal en el apéndice A.

```
[3] -> [0, 0, 0, 0, 1, 1, 0];
[4] -> [1, 0, 0, 1, 0, 1, 0];
[5] -> [0, 0, 1, 0, 0, 1, 0];
[6] -> [0, 0, 1  1, 0, 0, 0];
[7] -> [0, 0, 0, 0, 1, 1, 1];
[8] -> [0, 0, 0, 0, 0, 0, 0];
[9] -> [0, 0, 0, 0, 1, 0, 0];
TEST_VECTORS ([CLOCK, RESET] ->[ESTADO,a,b,c,d,e,f,g])
[.C..1]->[0,0, 0, 0, 0, 0, 0, 1];
[.C.,0]->[1,1, 0, 0, 1, 1, 1, 1];
[.C.,0]->[2,0, 1, 0, 0, 1, 0, 0];
[.C.,0]->[3,0, 0, 0, 0, 1, 1, 0];
[.C.,0]->[4,1, 0, 0, 1, 0, 1, 0];
[.C.,0]->[5,0, 0, 1, 0, 0, 1, 0];
[.C.,0]->[6,0, 0, 1, 1, 0, 0, 0];
[.C.,0]->[7,0, 0, 0, 0, 1, 1, 1];
[.C..0]->[8,0, 0, 0, 0, 0, 0, 0];
[.C.,0]->[9,0, 0, 0, 0, 1, 0, 0];
END
```

En esta versión del contador y codificador se observa que el tamaño del programa se redujo significativamente en comparación con las versiones anteriores, logrando la optimización del tiempo de escritura del programa, pero el funcionamiento del circuito es el mismo.

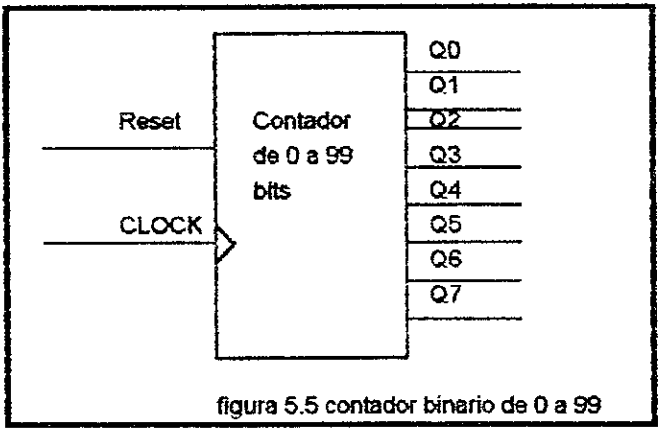


figura 5.5 contador binario de 0 a 99

PARTICIÓN DE UNA MÁQUINA DE ESTADO

En algunos casos el tamaño de la máquina de estado impide que se pueda grabar en un solo dispositivo. Cuando se tienen un gran número de transiciones entre estados se puede particionar la máquina de estado en varios bloques lógicos. Por ejemplo, para un contador binario de 0 a 99 , el diagrama de estados se forma con 100 posibles estados y se requieren 8 bits de salida. Describir este circuito en **ABEL** mediante un diagrama de estados o una tabla de transición de estados es un trabajo excesivo y no se puede grabar en un circuito GAL o cualquier otro PLD de mediana o baja densidad. El diagrama a bloques de este contador se muestra en la figura 5.5. Si se dividen los 8 bits de salida en dos grupos, uno que represente las decenas y otro que represente las unidades. cuando se completa un ciclo de 0 a 9 en las unidades la variable de salida CUENTA toma el valor de 1, al iniciar el ciclo nuevamente pasa a cero. esta variable es el reloj de el segundo contador. De tal manera que se tienen dos contadores de 0 a 9 en cascada y para describirlos se usan 20 estados en vez de 100, optimizando el circuito (figura 5.6).

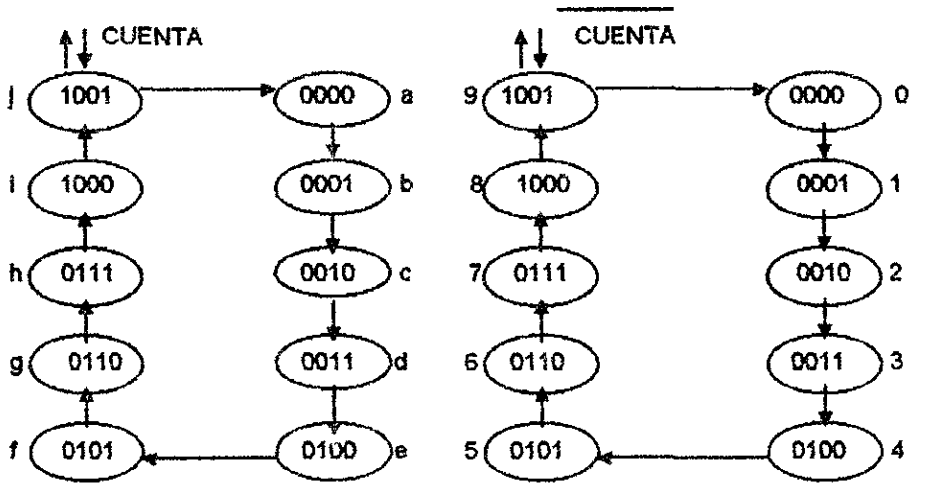


Diagrama de estados de las decenas activado por la señal de reloj CUENTA

Diagrama de estado de las unidades activado por la señal de reloj CLOCK

figura 5.5 Partición de la máquina de estado para un contador binario de 0 a 99.

PROGRAMA EN **ABEL**

```
module conta5
title 'contador binario de 0 a 99 particionando la máquina de estado
    Celia Rosa Fierro Santillán y José Filiberto Luie Flores'
CLOCK, RESET PIN; "Entradas al contador
Q0,Q1,Q2,Q3 PIN ISTYPE 'REG';"Bits de las decenas
Q4,Q5,Q6,Q7 PIN ISTYPE 'DC,REG'; "Bits de las unidades
CUENTA NODE ISTYPE 'COM';
A=0;B=1;C=2;D=3;E=4;F=5;G=6;H=7;I=8;J=9;"Valores de los estados
UNIDADES = [Q4,Q5,Q6,Q7];
DECENAS = [Q0,Q1,Q2,Q3];
EQUATIONS
UNIDADES.CLK = CLOCK;
DECENAS.CLK = CLOCK;
STATE_DIAGRAM UNIDADES
STATE 0: CUENTA = 0;
      IF CLOCK THEN 1 ELSE 0;
STATE 1: IF RESET THEN 0 ELSE
        IF CLOCK THEN 2 ELSE 1;
STATE 2: IF RESET THEN 0 ELSE
        IF CLOCK THEN 3 ELSE 2;
STATE 3: IF RESET THEN 0 ELSE
        IF CLOCK THEN 4 ELSE 3;
STATE 4: IF RESET THEN 0 ELSE
        IF CLOCK THEN 5 ELSE 4;
STATE 5: IF RESET THEN 0 ELSE
        IF CLOCK THEN 6 ELSE 5;
STATE 6: IF RESET THEN 0 ELSE
        IF CLOCK THEN 7 ELSE 8;
STATE 7: IF RESET THEN 0 ELSE
        IF CLOCK THEN 8 ELSE 7;
STATE 8: IF RESET THEN 0 ELSE
        IF CLOCK THEN 9 ELSE 8;
STATE 9: CUENTA = 1;
      IF RESET # CLOCK THEN 0 ELSE 9;
STATE_DIAGRAM DECENAS
STATE A: IF CUENTA THEN B ELSE A;
STATE B: IF RESET THEN A ELSE
        IF CUENTA THEN C ELSE B;
STATE C: IF RESET THEN A ELSE
        IF CUENTA THEN D ELSE C;
STATE D: IF RESET THEN A ELSE
        IF CUENTA THEN E ELSE D;
STATE E: IF RESET THEN A ELSE
        IF CUENTA THEN F ELSE E;
STATE F: IF RESET THEN A ELSE
        IF CUENTA THEN G ELSE F;
STATE G: IF RESET THEN A ELSE
```

```
        IF CUENTA THEN H ELSE G;
STATE H: IF RESET THEN A ELSE
        IF CUENTA THEN I ELSE H;
STATE I: IF RESET THEN A ELSE
        IF CUENTA THEN J ELSE I;
STATE J: IF CUENTA # RESET THEN A ELSE J;
TEST_VECTORS ([CLOCK,RESET]->[Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7,CUENTA])
        [ .C. ,  1 ]->[ 0, 0, 0, 0, 0, 0, 0, 0, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 0, 0, 1, 0, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 0, 0, 1, 1, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 0, 1, 0, 1, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 0, 1, 1, 0, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 0, 1, 1, 1, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 0, 1, 0, 0, 1, 1];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 1, 0, 0, 0, 0, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 1, 0, 0, 0, 1, 0];
        [ .C. ,  0 ]->[ 0, 0, 0, 0, 1, 0, 0, 1, 0, 0];
END
```

V. 4 MAQUINAS DE ESTADO ASÍNCRONAS.

Todas las máquinas de estado descritas hasta ahora son de tipo síncrono y usan flip-flops tipo D o tipo T. En cambio, las máquinas de estado asíncronas no requieren de flip-flops dedicados, en la práctica existen más aplicaciones de máquinas de estado asíncronas que de máquinas síncronas. Los diagramas de estado del lenguaje **ABEL** pueden ser utilizados también para la descripción lógica de máquinas de estado asíncronas. En el diagrama de estados correspondiente las señales que hacen que el circuito cambie de estado o permanezca en el mismo son entradas del sistema y no hay una señal de reloj. Para explicar el funcionamiento de un circuito de estas características se realizó un diseño que cuenta las personas que compran en una tienda y activa una alarma cuando paga el cliente número 100 con la finalidad de hacerle un descuento, se tiene un sensor que detecta cuando pasa una persona por la caja. El diagrama a bloques se muestra en la figura 5.7 y el diagrama de estado correspondiente se muestra en la figura 5.8.

En este circuito las entrada son SENSOR y CUENTA<99, las salidas generadas son: CLOCK que funciona como reloj del circuito contador de 0 a 99 que descrito en el apartado anterior ; y la señal CLIENTE100 que activa la alarma . Aunque SENSOR determina la transición de un estado a otro no es realmente una señal de reloj, ya que el tiempo que transcurre entre un cliente y otro es variable, puede ser muy corto o muy largo, por lo tanto se trata de una señal asíncrona.

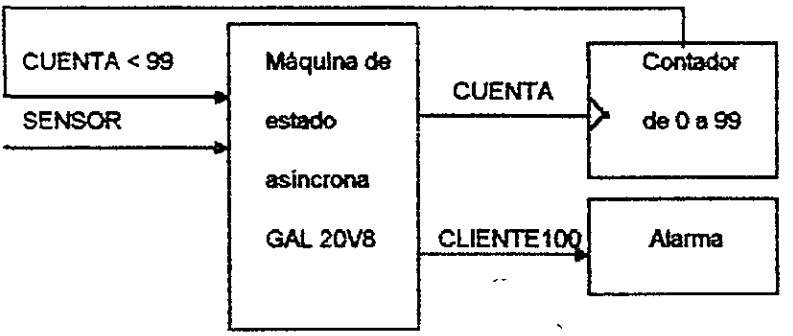


figura 5.6 Diagrama a bloques de una máquina de estado asíncrona que cuenta a las personas y activa una alarma al llegar a la persona 100.

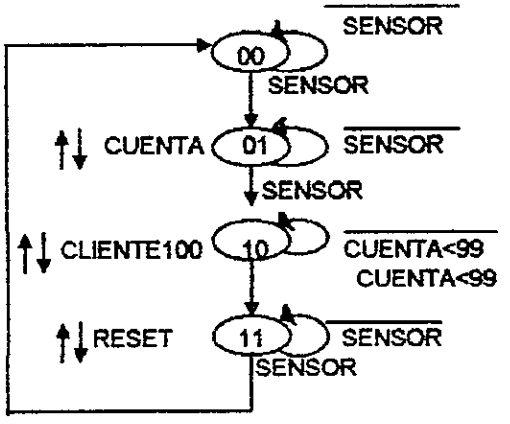


FIGURA 5.7 Diagrama de estados de un circuito que cuenta a las personas y activa una alarma al llegar a la persona 100.

PROGRAMA EN **ABEL**

```
module conta6
title 'Máquina de estado asincrona que cuenta personas y activa una alarma
    al llegar a la persona numero 100
    Celia Rosa Fierro Santillán y José Filiberto Lule'
disp46 DEVICE 'g16v8';
SENSOR, CUENTA99 PIN;"Entradas al circuito
RESET,CUENTA, CLIENTE100 PIN ISTYPE 'COM';"Salidas del circuito
Q0,Q1 NODE ISTYPE 'REG';"Bits de numero de estado
ESTADO = [Q0,Q1];
EQUATIONS
ESTADO.CLK = SENSOR;
ESTADO.CLK = CUENTA99;
STATE_DIAGRAM (ESTADO)
STATE 0:
    RESET = 1;
    CUENTA = 0;
    CLIENTE100 = 0;
    IF SENSOR THEN 1 ELSE 0;
STATE 1:
    RESET = 0;
    CUENTA = 1;
    IF SENSOR THEN 2 ELSE 1;

STATE 2:
    CUENTA = 0;
    IF CUENTA99 THEN 3 ELSE 2;
STATE 3:
    RESET = 1;
    CLIENTE100 = 1;
    IF SENSOR THEN 0 ELSE 3;
TEST_VECTORS ([SENSOR, CUENTA99]->[Q0,Q1,RESET,CUENTA,CLIENTE100])
        [ 0 ,0]->[0,0,1,0,0];
        [.C.,0]->[0,1,0,1,0];
        [.C.,0]->[1,0,0,0,0];
        [ 0 ,1]->[1,1,1,0,1];
        [.C.,0]->[0,0,1,0,0];
END
```

El circuito diseñado es un controlador sencillo, los controladores son máquinas de estado asincronas, Los cuales se explican más extensamente en el siguiente capitulo.

FALTA PAGINA

No. 110

CAPÍTULO VI. DISEÑO DE CONTROLADORES DE APLICACIÓN ESPECÍFICA MEDIANTE ABEL

VI. 1 CONTROLADORES

Un controlador digital es un circuito secuencial o máquina de estado, comúnmente asíncrona, que recibe señales o pulsos de entrada y proporciona señales o pulsos de salida de acuerdo a sus estados internos. Dependiendo de las condiciones del estado presente y de las entradas el controlador pasará a otro estado o permanecerá en el mismo, determinando así el funcionamiento de un sistema complejo. El diagrama a bloques de un controlador se muestra en la figura 6.1

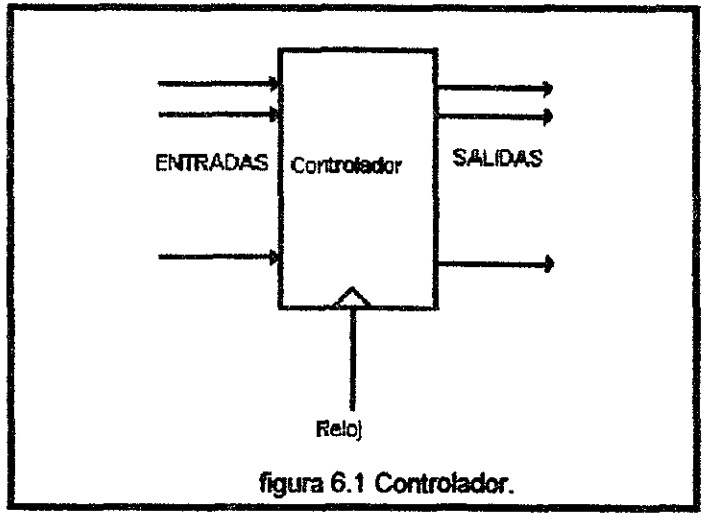


figura 6.1 Controlador.

La secuencia que sigue el controlador se especifica a través de un algoritmo de hardware, que consta de un número finito de pasos o procedimientos que describen sin ambigüedad el funcionamiento completo del sistema a controlar.

Para establecer adecuadamente la lógica de control de un sistema utilizando **ABEL** se deben realizar los siguientes pasos:

1. Analizar detalladamente como funciona el sistema que se desea controlar y plantear las posibles soluciones a la parte física del sistema (velocidad, fricción, temperatura, etc.).

2. Establecer un algoritmo que describa en detalle el funcionamiento del sistema (la parte física y la parte electrónica).
3. Realizar un diagrama de flujo que muestre la secuencia de funcionamiento del sistema.
4. Separar el sistema total en varios subsistemas como parte mecánica, electrónica de potencia, controlador, sensores, etc.
5. Identificar cuales serán las entradas y salidas del controlador.
6. Realizar la carta ASM del controlador.
7. Establecer el diagrama de estados del controlador.
8. Diseñar el controlador con la arquitectura más adecuada al problema. En este trabajo se utilizaron los circuitos MACH215 para diseñar un controlador debido a lo complejo del sistema
9. Escribir el programa en **ABEL**.
10. Compilar el programa y eliminar errores de sintaxis.
11. Simular el funcionamiento de el controlador.
12. Si la simulación es la esperada se graba el circuito controlador.
13. Si la simulación no es correcta se revisa el programa y regresar al punto 10.

A lo largo de este capítulo se desarrolla como ejemplo un controlador para una máquina cortadora de cartón, utilizando la metodología antes descrita.

VI.2 CARTA ASM

La secuencia que realiza un controlador se puede especificar a través de un algoritmo de hardware, a partir del cual se realiza un diagrama de flujo. El tipo de diagrama especial que ha sido desarrollado específicamente para definir algoritmos de hardware digitales se llama carta de la máquina de estado algorítmico (carta ASM).

La carta ASM se compone de tres elementos básicos: la casilla de estado, la casilla de decisión y la casilla condicional (figura 6.2). La casilla de estado es un rectángulo dentro del cual se escriben operaciones de registro o nombres de la señal de salida que se generan mientras el circuito se encuentra en ese estado.

El estado recibe un nombre simbólico, el cual se coloca en la parte superior izquierda de la casilla. El código binario asignado al estado se coloca en la parte superior derecha de la casilla. Dentro de la casilla se escribe la operación del registro, por ejemplo SENSOR $\leftarrow$ 0, lo que significa que el registro SENSOR toma el valor de cero, cuando el controlador está en el estado 010 (figura 6.3).

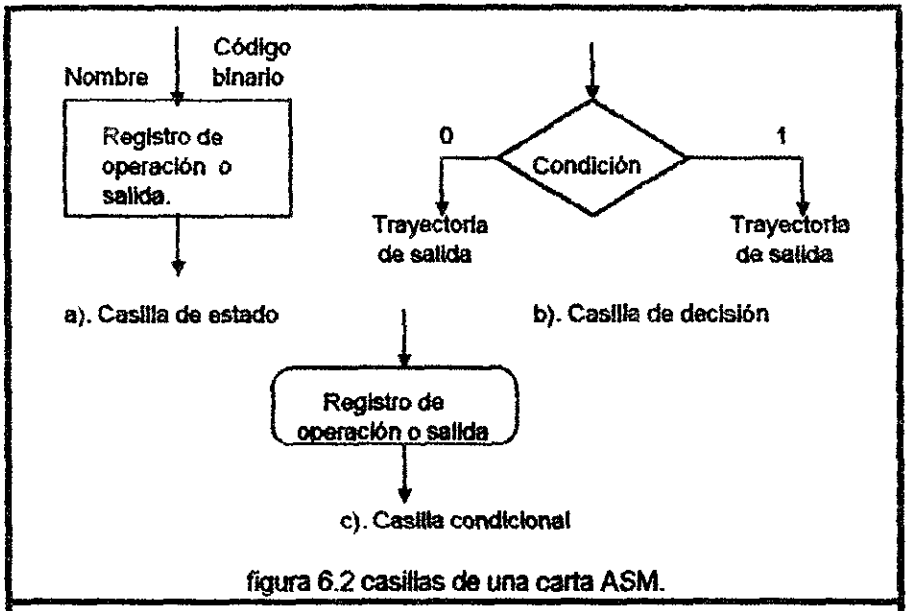


figura 6.2 casillas de una carta ASM.

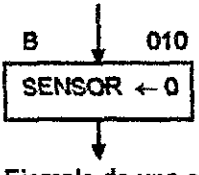


figura 6.3 Ejemplo de una casilla de estado.

La casilla de decisión establece el efecto de una entrada sobre el controlador, tiene forma de rombo con dos trayectorias de salida (figura 6.2 b). La condición de entrada se escribe dentro de la casilla, una de las trayectorias de salida se toma si la condición es cierta, y la otra si es falsa. Cuando a una condición de entrada se le asigna un valor binario las trayectorias se indican con 1 para un valor verdadero y con 0 para un valor falso, por ejemplo, si se tiene una variable de entrada S1, la casilla de decisión puede quedar como se muestra en la figura 6.4

Estos dos tipos de casillas son muy conocidas porque su función es similar a la que se tiene en diagramas de flujo convencionales. En cambio, la casilla condicional se utiliza exclusivamente en las cartas ASM, su forma es ovalada y la trayectoria de entrada debe llegar desde una de las trayectorias de salida de una casilla de decisión. Las operaciones de registro o salidas escritas dentro de una casilla condicional se generan durante un estado siempre que se satisfaga la condición de entrada, por ejemplo cuando S1 = 0 se asigna el valor de 1 a la variable de salida MOTOR y cuando S1 = 1 se le asigna el valor de 0 a la misma variable (figura 6.4).

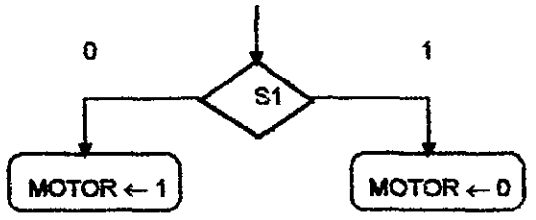


Figura 6.4 Ejemplo de una casilla de decisión con sus correspondientes casillas condicionales.

Un bloque ASM es una estructura que consta de una casilla de estado y todas las casillas de decisión y condicionales conectadas a sus trayectorias de salida, por lo que tiene solo una entrada pero puede tener varias salidas, en la figura 6.5 se muestra un ejemplo de bloque ASM de acuerdo con los ejemplos de casillas dados anteriormente. Cada bloque en la carta ASM describe el estado del controlador durante un pulso de reloj, por lo que fácilmente se puede pasar a un diagrama de estados para describir el funcionamiento de un controlador.

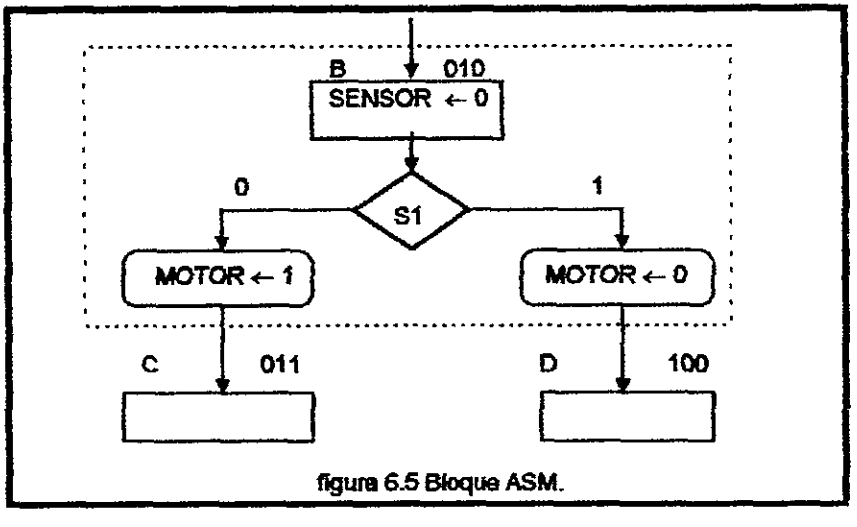


figura 6.5 Bloque ASM.

Frecuentemente un controlador se define por los registros que contiene y las operaciones que se realizan en los datos almacenados en ellos. En el sentido más amplio, el término registro incluye registros de almacenamiento, registros de corrimiento, contadores y flip-flops solos. Las operaciones que se pueden realizar con registro son: corrimiento, incremento, adición, despeje y transferencia de datos. En la

tabla 6.1 se describe la notación simbólica que se utiliza para las operaciones con registros en una carta ASM.

Tabla 6.1 Operaciones con registros.

Notación simbólica	Descripción
A ← B	Transferencia del contenido del registro B al registro A
R ← 0	Despejar el registro R
F ← 1	Establecer el flip-flop en 1
A ← A + 1	Incrementar el registro A en 1
A ← A - 1	Disminuir el registro A en 1
A ← A + B	Agregar al contenido del registro A el del registro B

VI.3 DISEÑO DE UN CONTROLADOR PARA UNA MÁQUINA CORTADORA DE CARTÓN

VI.3.1 ANÁLISIS DETALLADO DEL SISTEMA

Se tiene una máquina que corta cartón de tres capas unido con papel impreso que se utiliza para hacer cajas. El problema esencial es que la cuchilla debe cortar el cartón en algún lugar preestablecido de acuerdo con la impresión del papel. El diagrama simplificado se muestra en la figura 6.3.

El motor principal mueve a los rodillos transportadores, el alimentador y la cuchilla giratoria que están sincronizados a una velocidad establecida por el control de velocidad del motor. Sin embargo, debido al engomado pueden existir atrasos o adelantos del cartón provocados por la fricción. El cartón está en rollo pero el papel impreso que se pega sobre el cartón se encuentra en forma de pliegos que son colocados sobre el rollo de cartón a través del alimentador.

Se deja cierto espacio entre un pliego de papel y otro pero este espacio debe ser solo de algunos milímetros con el fin de que no se empalmen dos pliegos de papel, pero al mismo tiempo que los rodillos transportadores no se manchen demasiado con el engomado y manchen a su vez el papel impreso. Además de lo anterior la velocidad normal de operación de la máquina es de aproximadamente 1 corte por segundo. Por lo que se requiere un sistema que pueda corregir los errores en un tiempo de 0.5 s (la mitad de un ciclo) o menos, con el fin de que el error no se acumule y provoque un defasamiento grande entre la marca de papel y el corte.

Para poder determinar la posición de la marca de papel donde se debe cortar y la cuchilla giratoria se deben emplear sensores ópticos que detecten un cambio en el color en el caso del papel, y una marca preestablecida en el caso de la cuchilla giratoria. Además existen las limitantes mecánicas del sistema. Para que el

servomotor funcione, la señal de arranque debe durar al menos 4 milisegundos y como máximo 0.4 seg. para que alcance a detenerse y no haya problemas por la inercia entre un ciclo y otro. También se debe considerar que la longitud del papel impreso, y por lo tanto, del corte es variable. La diferencia mínima entre un formato y otro es de 0.5 cm. Los formatos que se manejan pueden ser desde 20 cm a 1 5 m. por lo que la velocidad de movimiento de los rodillos transportadores de cartón está en el rango de 40 a los 90 metros por minuto.

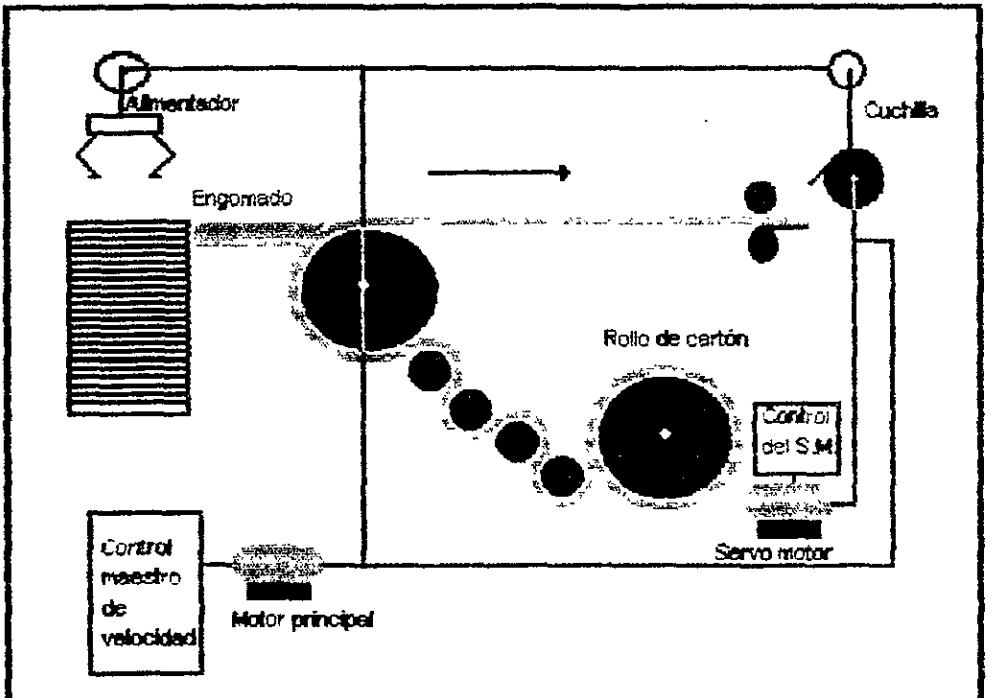


figura 6.3 Máquina cortadora de cartón.

Para corregir los errores de corte se tiene un servomotor que puede adelantar o atrasar la cuchilla giratoria dentro de cierto rango y tiene una velocidad constante. El servomotor realiza una vuelta completa de la cuchilla en 8 minutos, por lo que su velocidad angular es de 0.75 grados/segundo. La corrección que se pude hacer depende del tamaño del papel a cortar. Se tiene una longitud mínima de corrección de 0.29 mm (cuando se trabaja el papel de 20 cm) y una longitud máxima de corrección de 2.2 mm (cuando se trabaja con papel de 150 cm) hacia atrás o hacia adelante. Esta corrección es muy pequeña pero es suficiente para que el error no se acumule.

Considerando la relación de velocidades que existe cuando la cuchilla de corte funciona con el motor principal y cuando lo hace con el servomotor se tiene que el motor principal es 480 veces más rápido que el servomotor, por lo que para corregir un error el servomotor debe funcionar un tiempo 480 veces mayor que el tiempo de defasamiento entre las señales de los sensores. Considerando el tiempo mínimo que se debe mantener la señal del servomotor para que arranque tenemos que si el tiempo entre la entrada de la señal de un sensor y otro *(te)* es menor a 8.33μs no podríamos encender el servomotor y si es mayor de 8.33μs y menor de 833μs el error se puede corregir sin exceder el ciclo de corte.

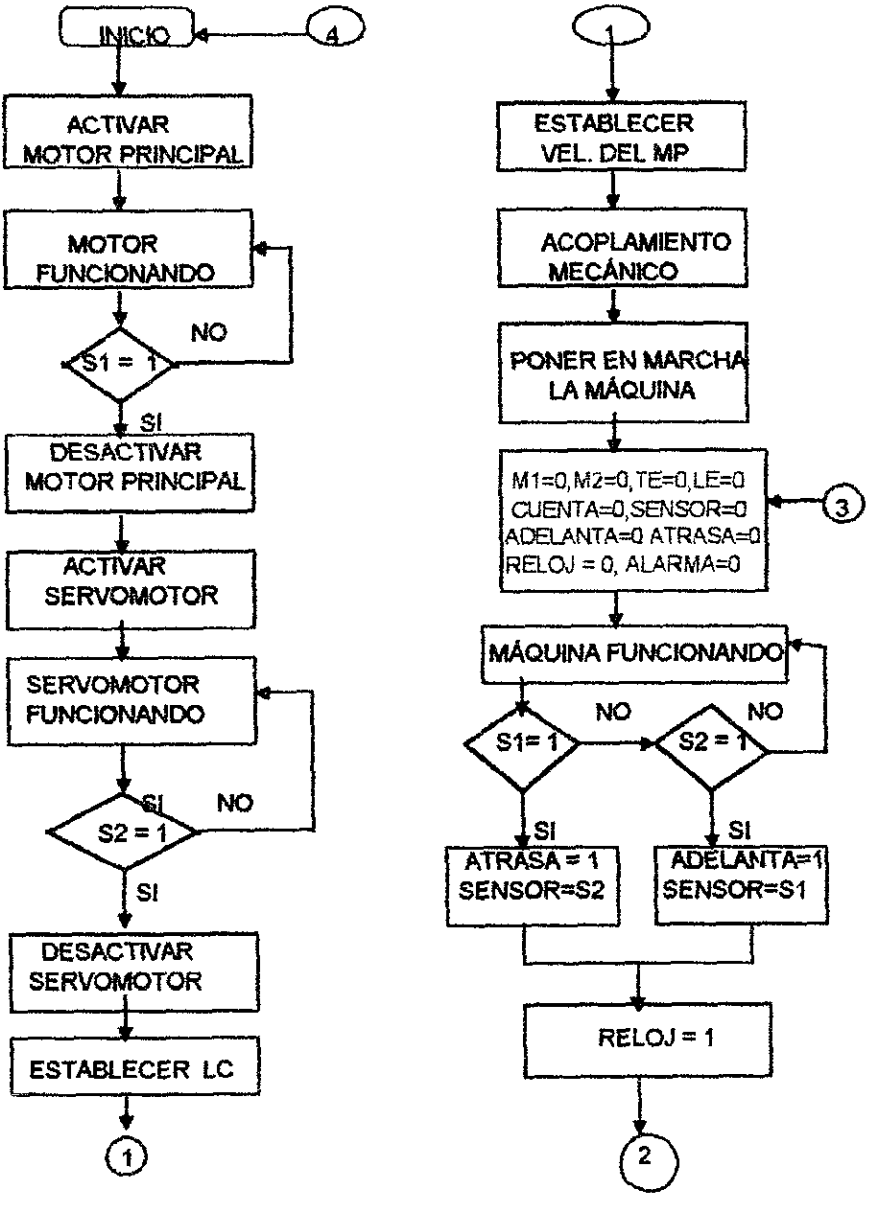
VI.3.2 ALGORITMO DE FUNCIONAMIENTO DEL SISTEMA

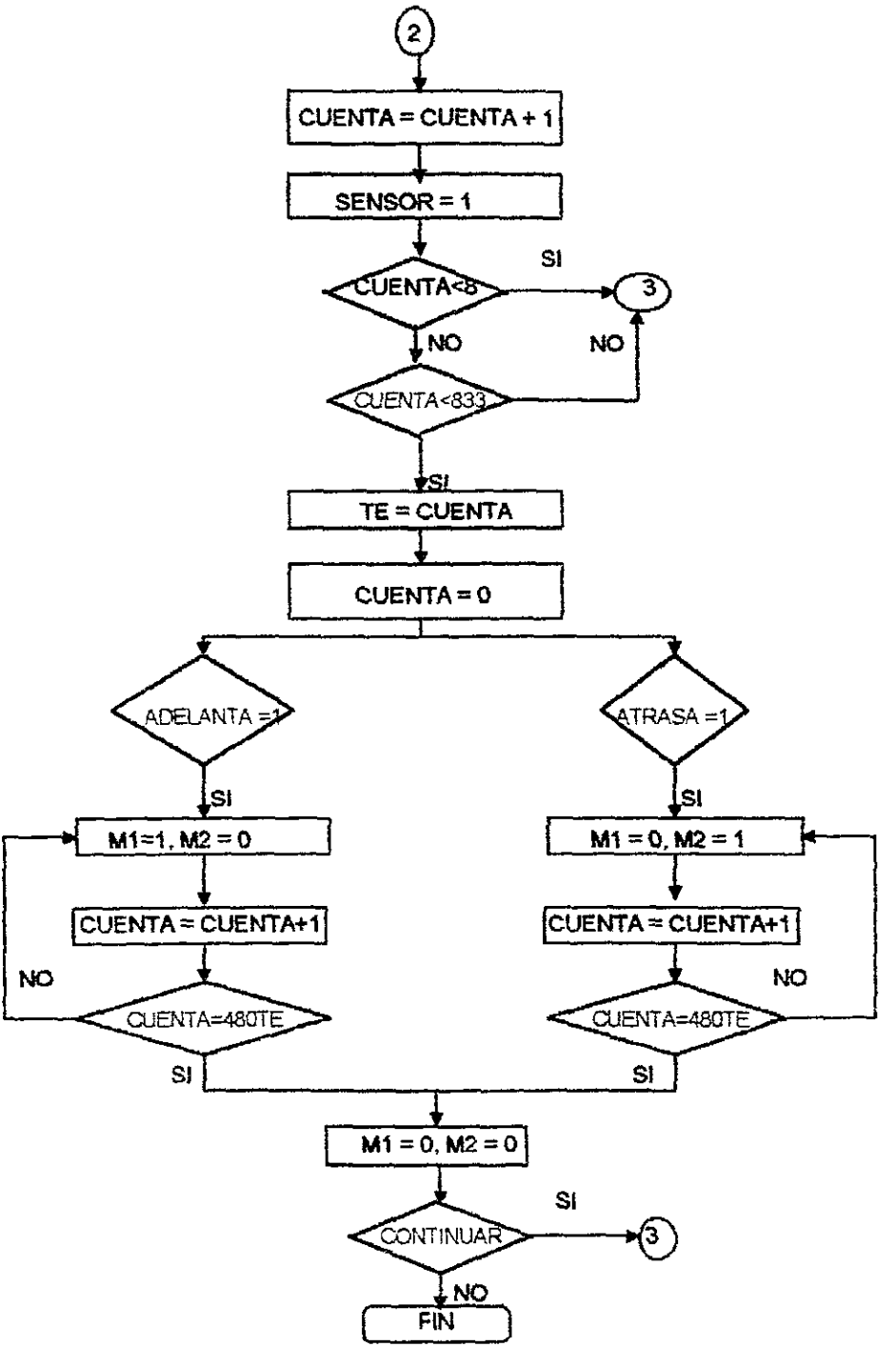
En este algoritmo se utilizan las siguientes variables: S1 representa al sensor del papel, S2 al sensor de la cuchilla, SENSOR toma el valor de S1 o S2 dependiendo de si es necesario adelantar o atrasar la cuchilla, ADELANTA se utiliza para adelantar la cuchilla, ATRASA para atrasarla, LC representa la longitud de cartón que se va a cortar, M1 y M2 establecen la dirección de giro del motor (a la derecha o al izquierda), RELOJ activa la señal de reloj de un contador, CUENTA sirve para el conteo del tiempo, TE es el tiempo de error (la diferencia en tiempo entre el registro de un sensor y otro) , LE es la longitud de error (la diferencia en longitud de la marca donde debe hacerse el corte y el corte real), ALARMA activa una señal que indica que el error en el corte es demasiado grande y que es necesario reiniciar todo el proceso.

1. Inicio.
2. Colocar el control de velocidad del Motor principal (M.P). a 0.60 m / min.
3. Activar el motor principal.
4. Motor funcionando.
5. Mientras S1 = 0 mantenerse en el paso 4.
6. Si S1 = 1 continuar
7. Desactivar el M.P.
8. Activar el servomotor para mover la cuchilla.
9. Servomotor funcionando.
10. Mientras S2 = 0 mantenerse en el paso 9.
11. Si S2 = 1 continuar.
12. Desactivar el servomotor.
13. Establecer la longitud entre un corte y otro (LC).
14. Establecer la velocidad del motor principal para que se realice un corte por segundo.
15. Establecer el acoplamiento mecánico entre la cuchilla y el motor principal (por medio del control diferencial y la caja de engranes con que cuenta la máquina) para que se realice un corte por segundo.
16. Poner en marcha la máquina.

17. Poner a cero todas las variables del sistema M1 = 0, M2 = 0, CUENTA = 0. ADELANTA = 0, ATRASA = 0, SENSOR = 0, RELOJ = 0, TE = 0, LE = 0, ALARMA = 0.
18. Máquina funcionando.
19. Mientras S1 = 0 y S2 = 0 mantenerse en el paso 18.
20. Si S1 = 1 y S2 = 0 entonces ADELANTA = 1 Y SENSOR = S2, continuar.
21. Si S1 = 0 y S2 = 1 entonces ATRASA = 1 y SENSOR = S1, continuar.
22. RELOJ = 1
23. CUENTA = CUENTA + 1
24. Si SENSOR = 0 entonces regresa al paso 22.
25. Si SENSOR = 1 continuar.
26. Si CUENTA < 8 regresa al paso 17.
27. Si CUENTA < 833 pasa al número 35.
28. TE = CUENTA
29. Si ADELANTA = 1 entonces M1 = 1 y M2 = 0, continuar (el servomotor adelanta la cuchilla).
30. Si ATRASA = 1 entonces M1 = 0 y M2 = 1, continuar (el servomotor atrasa la cuchilla).
31. CUENTA = CUENTA +1.
32. Si CUENTA < 480 * TE continuar en el paso 29 o 30.
33. Si CUENTA = 480 * TE continua.
34. M1 = 0 y M2 = 0 (el servomotor se detiene).
35. Regresa al paso 18.
36. FIN

VI.3.3 DIAGRAMA DE FLUJO DEL FUNCIONAMIENTO DEL SISTEMA.




2
CUENTA = CUENTA + 1
SENSOR = 1
SI
CUENTA<8
3
NO
NO
CUENTA<833
SI
TE = CUENTA
CUENTA = 0
ADELANTA =1
ATRASA =1
SI
SI
M1=1, M2 = 0
M1 = 0, M2 = 1
CUENTA = CUENTA+1
CUENTA = CUENTA+1
NO
NO
CUENTA=480TE
CUENTA=480TE
SI
SI
M1 = 0, M2 = 0
SI
CONTINUAR
3
NO
FIN

VI.3.4 SUBSISTEMAS DEL SISTEMA TOTAL

Para este sistema en particular se tienen los siguientes subsistemas:

1. Control de velocidad del motor principal.
2. Control del servomotor.
3. Contador
4. Comparador
5. Sensores.
6. Dos Etapas de potencia (una para cada motor).
7. Parte mecánica (cuchilla, rodillos, alimentador, etc.).

Se pueden representar las interacciones entre estos subsistemas por medio de un diagrama a bloques.

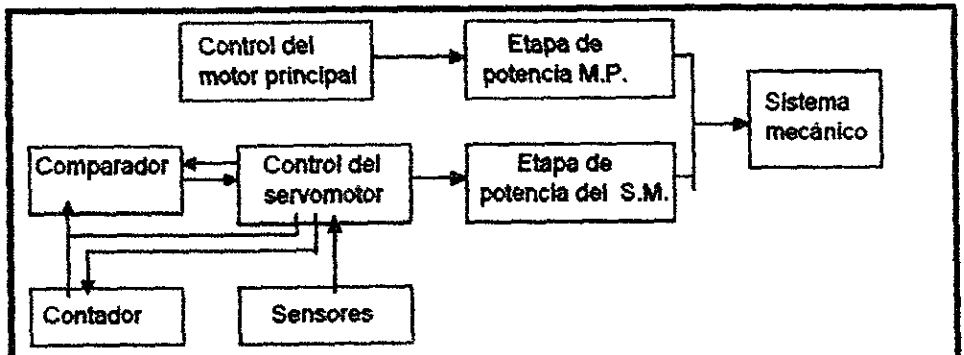


figura 6.6 Diagrama a bloques de los subsistemas que integran la máquina cortadora de cartón.

VI.3.5 DISEÑO DE LOS CIRCUITOS DIGITALES AUXILIARES

Además del controlador, se requiere un contador que pueda medir el tiempo de error y el tiempo de corrección y un circuito comparador para verificar que *TE = TC.*

DISEÑO DEL CONTADOR

El tiempo de error *(TE)* debe ser mayor 8.33 μs (8 μs) y menor de 833μs. Para contar hasta 833 se requieren 10 bits, pero se utilizan 12 bits, cuatro por cada dígito, como ya se hizo en ejemplos anteriores, para que se realice la cuenta se requieren dos señales de reloj, uno con frecuencia de 1MHz y otro de 2.083 KHz, el primero incrementará el contador cada microsegundo y permite medir el tiempo de error y el segundo lo incrementará cada 480 microsegundos para medir el tiempo de corrección. Para que el circuito funcione solo cuando sea necesario se tiene una señal de habilitación y para seleccionar el reloj que se usa en cada caso tenemos una

señal de selección. Además, la comparación de *TE* con *TC* se realiza en un comparador, fuera del controlador, así que los 12 bits del conteo se trasladan al comparador, por lo que son pines de salida. Las comparaciones TE>8 y TE<833 son nodos internos . Las entradas y salidas que del circuito son:

RELOJ1	Entrada, señal del reloj con una frecuencia de 1Mhz.
RELOJ2	Entrada, señal de reloj con una frecuencia de 2.083 KHz.
HABILITA	Cuando esta entrada vale 0 el circuito está inactivo y cuando vale 1 el circuito se encuentra funcionando.
SELECCIÓN	Cuando esta entrada vale 0 la señal de reloj es RELOJ1 y cuando vale 1 la señal de reloj es RELOJ2.
RANGO	Esta salida toma el valor de 0 si el tiempo de error es mayor o menor de lo requerido y toma el valor de 1 cuando el tiempo de error está dentro del rango y se cumple que TE>8μ s y TE<833μ s.
Q0,Q1,Q2,Q3	Bits de las centenas.
Q4,Q5,Q6,Q7	Bits de las decenas.
Q8,Q8,Q10,Q11	Bits de las unidades

El diagrama a bloques del contador queda de la siguiente manera:

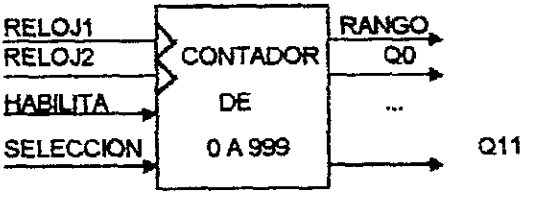


figura 6.5 Diagrama a bloques del contador para el control de la máquina contadora de cartón.

Para este contador, como ya se mencionó anteriormente, se requieren 12 bits de cuenta, además de 4 entradas, 2 salidas, 2 señales de comparación de tiempo y 2 más que funcionan como reloj para conteo de decenas y centenas, lo que nos da un total de 22 señales que deben ser colocadas en pines o nodos. Esta cantidad de señales excede con mucho la capacidad de un circuito GAL.

Para resolver este problema se tienen dos opciones: grabar el contador en un PLD de mayor densidad o particionar el diagrama de estados para que el contador completo se grabe en dos o más circuitos GAL. Se opto por la primera solución, ya que **ABEL** permite grabar también en dispositivos MACH.

Un circuito MACH 215 cuenta con 44 pines, de los cuales 32 son OLMCs que pueden ser configuradas como entradas o salidas, 2 pines de reloj, 4 pines de entrada y es equivalente a 4 circuitos PAL (ver apéndice C). Si utilizamos 22 pines para las señales descritas anteriormente, aún tenemos 16 pines libres.

PROGRAMA EN **ABEL**

```
module contado1
title 'Contador de 0 a 999 para el control de la máquina cortadora de cartón
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores'
DCONT1 DEVICE 'mach215a';
RELOJ1 PIN 13;RELOJ2 PIN 35;"Entradas de reloj
SELECCION PIN 10;HABILITA PIN 11;RESET PIN 32;"Entradas de selección , habilitación
y reset
Q0 PIN 2 ISTYPE 'REG';Q1 PIN 3 ISTYPE 'REG';Q2 PIN 4 ISTYPE 'REG';Q3 PIN 5
ISTYPE 'REG';
Q4 PIN 6 ISTYPE 'REG';Q5 PIN 7 ISTYPE 'REG';Q6 PIN 8 ISTYPE 'REG';Q7 PIN 9
ISTYPE 'REG';
Q8 PIN 14 ISTYPE 'REG';Q9 PIN 15 ISTYPE 'REG';Q10 PIN 16 ISTYPE 'REG_D';Q11
PIN 17 ISTYPE 'REG';"Salidas del contador
UNIDADES = [Q8,Q9,Q10,Q11];
DECENAS  = [Q4,Q5,Q6 ,Q7 ];
CENTENAS = [Q0,Q1,Q2 ,Q3 ];
R0 ,R1 NODE ISTYPE 'REG';"Señales internas de reloj
R2 NODE  ISTYPE 'COM';"Señal interna de reloj
CUENTA8 PIN  ISTYPE 'BUFFER';CUENTA833 PIN  ISTYPE 'COM';
RANGO PIN  ISTYPE 'COM';"Salida que establece si el tiempo esta dentro del rango
EQUATIONS
R1.CLK = R2;
R0.CLK = R1;
R2 = RELOJ1 & !SELECCION # RELOJ2 & SELECCION;
R1 := Q8 & !Q9 & !Q10 & Q11 #RESET;
R0 := Q4 & !Q5 & !Q6  & Q7 #RESET;
UNIDADES:=(UNIDADES.FB +1) & (UNIDADES.FB <9)&!RESET;
DECENAS :=(DECENAS.FB  +1) & (DECENAS.FB < 9)&!RESET;
CENTENAS:=(CENTENAS.FB +1) & (CENTENAS.FB <9)&!RESET;
UNIDADES.CLK = R2;
DECENAS.CLK  = R1;
CENTENAS.CLK = R0;
CUENTA8 = (UNIDADES >=8)#(DECENAS>=1)#(CENTENAS>=1);
CUENTA833 =
(CENTENAS<8)#((CENTENAS==8)&(DECENAS<3))#((CENTENAS==8)&(DECENAS==3)
&(UNIDADES<=3));
RANGO= CUENTA8.FB & CUENTA833.FB;
RANGO.OE = HABILITA;
TEST_VECTORS ([RELOJ1,RELOJ2,SELECCION,HABILITA,RESET]->
[CENTENAS,DECENAS,UNIDADES,RANGO])
        [.X.,.X.,0,0,.X.]->[ 0 , 0 , 0 ,.Z.];
@REPEAT 7{
        [.C.,.X.,0,1,0]->[ 0 , 0 ,.X., 0 ];
      }
        [.C.,.X.,0,1,0]->[ 0 , 0 , 8 , 1 ];
@REPEAT 80 {
        [.C.,.X.,0,1,0]->[.X.,.X.,.X., 1 ];
       }
        [.C.,.X.,0,1,0]->[ 0 , 8 , 9 , 1 ];
        [.X.,.C.,1,1,1]->[ 0 , 0 , 0 , 0 ];
```

```
        [.X.,.C.,1,1,0]->[ 0 , 0 , 1 , 0 ];
@REPEAT 87 {
        [.X.,.C.,1,1,0]->[ 0 ,.X.,.X.,.X.];
      }
        [.X.,.C.,1,1,0]->[ 0 , 8 , 9 , 1 ];
        [.X.,.C.,1,1,0]->[ 0 , 9 , 0 , 1 ];
END
```

En este programa se utilizó la directiva de compilación **@REPEAT** cuya función es la de ejecutar un grupo de instrucciones el número de veces que se indique, en este caso utilizamos esta directiva para repetir los vectores de prueba, ya que los valores posibles para el contador es de 0 a 999 y el rango válido para *TE* es de 8 a 833, de esta manera nos ahorramos el escribir por lo menos 834 vectores de prueba.

Se utilizan ecuaciones, que aunque son fáciles de escribir, involucran una lógica compleja y una gran cantidad de minitérminos. **ABEL** desarrolla estas ecuaciones y , en caso de que los minitérminos disponibles para un **PIN** o nodo (12 minitérminos) sean insuficientes para desarrollar esta lógica hace una partición de las señales creando nodos auxiliares a la señal correspondiente.

A continuación se presenta el archivo de ecuaciones contado1.eq2 desarrolladas por **ABEL** a partir del programa fuente para este contador.

```
MODULE contado1
TITLE 'Contador de 0 a 999 para el control de la máquina cortadora de cartón
      Celia Rosa Fierro Santillan y José Filiberto Lule Flores
EQUATIONS
IGUALES = (Q0 & Q1 & CARRY@4 & C1 & C0
   # Q0 & IQ1 & CARRY@4 & IC1 & C0
   # IQ0 & Q1 & CARRY@4 & C1 & IC0
   # IQ0 & IQ1 & CARRY@4 & IC1 & IC0);
IGUALES.OE = (SELECCION);
CUENTA8 = (Q8
   # Q4
   # Q5
   # Q6
   # Q0
   # Q1
   # Q2
   # Q3
   # IQ4 & IQ5 & IQ6 & Q7 & IQ1 & IQ2 & IQ3);
CUENTA833 = (IQ0
   # IQ8 & IQ9 & IQ4 & IQ5 & IQ1 & IQ2 & IQ3
   # IQ4 & IQ5 & IQ6 & Q7 & IQ1 & IQ2 & IQ3
   # IQ4 & IQ5 & IQ7 & IQ1 & IQ2 & IQ3);
RANGO = (CUENTA8.FB & CUENTA833.FB);
RANGO.OE = (HABILITA);
R1 := (Q8 & IQ9 & IQ10 & Q11
   # RESET);
```

```
R1.C = (RELOJ);
R0 := (RESET
   # Q4 & !Q5 & !Q6 & Q7);
R0.C = (R1);
Q8 := (!RESET & !Q8.FB & Q9.FB & Q10.FB & Q11.FB
   # !RESET & Q8.FB & !Q9.FB & !Q10.FB & !Q11.FB);
Q8.C = (RELOJ);
Q9 := (!RESET & !Q8.FB & Q9.FB & !Q10.FB
   # !RESET & !Q8.FB & !Q9.FB & Q10.FB & Q11.FB
   # !RESET & !Q8.FB & Q9.FB & !Q11.FB);
Q9.C = (RELOJ);
Q10 := (!RESET & !Q8.FB & !Q10.FB & Q11.FB
   # !RESET & !Q8.FB & Q10.FB & !Q11.FB);
Q10.C = (RELOJ);
Q11 := (!RESET & !Q8.FB & !Q11.FB
   # !RESET & Q8.FB & !Q9.FB & !Q10.FB & !Q11.FB);
Q11.C = (RELOJ);
Q4 := (!RESET & !Q4.FB & Q6.FB & Q6.FB & Q7.FB
   # !RESET & Q4.FB & !Q6.FB & !Q6.FB & !Q7.FB);
Q4.C = (R1);
Q5 := (!RESET & !Q4.FB & Q6.FB & !Q6.FB
   # !RESET & !Q4.FB & !Q6.FB & Q6.FB & Q7.FB
   # !RESET & !Q4.FB & Q6.FB & !Q7.FB);
Q6.C = (R1);
Q6 := (!RESET & !Q4.FB & !Q6.FB & Q7.FB
   # !RESET & !Q4.FB & Q6.FB & !Q7.FB);
Q6.C = (R1);
Q7 := (!RESET & !Q4.FB & !Q7.FB
   # !RESET & Q4.FB & !Q6.FB & !Q6.FB & !Q7.FB);
Q7.C = (R1);
Q0 := (!RESET & !Q0.FB & Q1.FB & Q2.FB & Q3.FB
   # !RESET & Q0.FB & !Q1.FB & !Q2.FB & !Q3.FB);
Q0.C = (R0);
Q1 := (!RESET & !Q0.FB & Q1.FB & !Q2.FB
   # !RESET & !Q0.FB & !Q1.FB & Q2.FB & Q3.FB
   # !RESET & !Q0.FB & Q1.FB & !Q3.FB);
Q1.C = (R0);
Q2 := (!RESET & !Q0.FB & !Q2.FB & Q3.FB
   # !RESET & !Q0.FB & Q2.FB & !Q3.FB);
Q2.C = (R0);
Q3 := (!RESET & !Q0.FB & !Q3.FB
   # !RESET & Q0.FB & !Q1.FB & !Q2.FB & !Q3.FB);
Q3.C = (R0);
C8 := (Q8.FB);
C8.C = (SELECCION);
C9 := (Q9.FB);
C9.C = (SELECCION);
C10 := (Q10.FB);
C10.C = (SELECCION);
C11 := (Q11.FB);
C11.C = (SELECCION);
C4 := (Q4.FB);
```

```
C4.C = (SELECCION);
C5 := (Q5.FB);
C5.C = (SELECCION);
C6 := (Q6.FB);
C6.C = (SELECCION);
C7 := (Q7.FB);
C7.C = (SELECCION);
C0 := (Q0.FB);
C0.C = (SELECCION);
C1 := (Q1.FB);
C1.C = (SELECCION);
C2 := (Q2.FB);
C2.C = (SELECCION);
C3 := (Q3.FB);
C3.C = (SELECCION);
CARRY@0 = (Q10 & Q11 & C11 & C10
   # Q10 & !Q11 & !C11 & C10
   # !Q10 & Q11 & C11 & !C10
   # !Q10 & !Q11 & !C11 & !C10);
CARRY@1 = (Q8 & Q9 & CARRY@0 & C9 & C8
   # Q8 & !Q9 & CARRY@0 & !C9 & C8
   # !Q8 & Q9 & CARRY@0 & C9 & !C8
   # !Q8 & !Q9 & CARRY@0 & !C9 & !C8);
CARRY@2 = (Q6 & Q7 & CARRY@1 & C7 & C6
   # Q6 & !Q7 & CARRY@1 & !C7 & C6
   # !Q6 & Q7 & CARRY@1 & C7 & !C6
   # !Q6 & !Q7 & CARRY@1 & !C7 & !C6);
CARRY@3 = (Q4 & Q5 & CARRY@2 & C5 & C4
   # Q4 & !Q5 & CARRY@2 & !C5 & C4
   # !Q4 & Q5 & CARRY@2 & C5 & !C4
   # !Q4 & !Q5 & CARRY@2 & !C5 & !C4);
CARRY@4 = (Q2 & Q3 & CARRY@3 & C3 & C2
   # Q2 & !Q3 & CARRY@3 & !C3 & C2
   # !Q2 & Q3 & CARRY@3 & C3 & !C2
   # !Q2 & !Q3 & CARRY@3 & !C3 & !C2);
END
```

Las señales CARRY@0, CARRY@1, CARRY@2, CARRY@3 y CARRY@4 son los nodos creados por **ABEL** para las señales que tienen más de 12 minitérminos de entrada. Como podemos observar, el archivo de ecuaciones es mucho más extenso que el archivo fuente porque desarrolla toda la lógica de las ecuaciones.

Cada señal, incluyendo las creadas por **ABEL** que no fueron declaradas en el archivo fuente, es asignada a un pin o nodo que internamente está conectado a alguno de los 4 bloques PAL del MACH215. El archivo de asignación de pines, contado1.fit, se muestra a continuación.

En esta primera parte se especifican los pines que fueron preasignados desde el programa fuente en **ABEL**.

User pre-assignments:

RELOJ1	**PIN** 13
SELECCION	**PIN** 10
RELOJ2	**PIN** 35
RESET	**PIN** 32
HABILITA	**PIN** 11
Q0	**PIN** 2
Q1	**PIN** 3
Q2	**PIN** 4
Q3	**PIN** 5
Q4	**PIN** 6
Q5	**PIN** 7
Q6	**PIN** 8
Q7	**PIN** 9
Q8	**PIN** 14
Q9	**PIN** 15
Q10	**PIN** 16
Q11	**PIN** 17

En esta parte se muestra el total de recursos del MACH215 tales como pines, minitérminos, entradas a los bloques PAL, macroceldas, etc. Cuántos de ellos se utilizaron y en que porcentaje, y los que quedaron libres.

Results of fitting contado1.tt2 into MACH216.

Chip-Level Resource Summary

RESOURCE	USED	FREE	UTILIZATION
product terms	47	81	36.7%
macro cells	20	12	62.5%
PIN registers	0	32	0.0%
PINs	20	18	52.6%
PAL inputs	41	47	46.6%

En esta parte se muestran los minitérminos, macroceldas, entradas y pines por cada uno de los 4 bloques PAL.

Block A Resource Summary

RESOURCE	USED	FREE	UTILIZATION
product terms	18	14	56.3%
macro cells	8	0	100.0%
PIN registers	0	8	0.0%
PINs	8	0	100.0%
PAL inputs	11	11	50.0%

Block B Resource Summary

RESOURCE	USED	FREE	UTILIZATION
product terms	10	22	31.3%
macro cells	5	3	62.5%
PIN registers	0	8	0.0%
PINs	5	3	62.5%
PAL inputs	9	13	40.9%

Block C Resource Summary

RESOURCE	USED	FREE	UTILIZATION
product terms	15	17	46.9%
macro cells	5	3	62.5%
PIN registers	0	8	0.0%
PINs	2	6	26.0%
PAL inputs	12	10	54.5%

Block D Resource Summary

RESOURCE	USED	FREE	UTILIZATION
product terms	4	28	12.5%
macro cells	2	6	26.0%
PIN registers	0	8	0.0%
PINs	0	8	0.0%
PAL inputs	9	13	40.9%

En esta sección se especifican los pines que son solo de entrada. (entradas dedicadas).

```
------------------------------------------------------------
                Detailed Fitting Results
------------------------------------------------------------

--------- Dedicated Inputs -------
PIN:10  --> SELECCION
PIN:11  --> HABILITA
PIN:32  --> RESET
PIN:33  -->
```

En esta sección se especifican las entradas de reloj

```
---------- Clock Inputs ----------
PIN:13  --> RELOJ1
PIN:35  --> RELOJ2
```

En esta sección se especifica a que pines o nodos internos fué asignada cada una de las señales utilizadas en el circuito, separandolas por bloques PAL

```
------------------------------------------------------------
    Resources of Block A
------------------------------------------------------------

Resource allocation to macro cells:
 A0  (PIN :2 ) -->
        PIN INPUT       : Q0
        PIN OUTPUT              :
Q0.REG (pts=2)
        CLOCK PT        : R0
        STEERING        : LOCAL
 A1  (NODE:45 ) -->
 A2  (PIN :3 ) -->
        PIN INPUT       : Q1
        PIN OUTPUT              :
Q1.REG (pts=3)
        CLOCK PT        : R0
        STEERING        : LOCAL
 A3  (NODE:46 ) -->
 A4  (PIN :4 ) -->
        PIN INPUT       : Q2
        PIN OUTPUT              :
Q2.REG (pts=2)
        CLOCK PT        : R0
        STEERING        : LOCAL
 A5  (NODE:47 ) -->
 A6  (PIN :5 ) -->
        PIN INPUT       : Q3
        PIN OUTPUT              :
Q3.REG (pts=2)
        CLOCK PT        : R0
        STEERING        : LOCAL
 A7  (NODE:48 ) -->
 A8  (PIN :6 ) -->
        PIN INPUT       : Q4
        PIN OUTPUT              :
Q4.REG (pts=2)
        CLOCK PT        : R1
        STEERING        : LOCAL
 A9  (NODE:49 ) -->
 A10 (PIN :7 ) -->
        PIN INPUT       : Q5
        PIN OUTPUT              :
Q6.REG (pts=3)
        CLOCK PT        : R1
        STEERING        : LOCAL
 A11 (NODE:50 ) -->
 A12 (PIN :8 ) -->
        PIN INPUT       : Q6
        PIN OUTPUT              :
Q6.REG (pts=2)
        CLOCK PT        : R1
        STEERING        : LOCAL
 A13 (NODE:51 ) -->
 A14 (PIN :9 ) -->
        PIN INPUT       : Q7
        PIN OUTPUT              :
Q7.REG (pts=2)
        CLOCK PT        : R1
        STEERING        : LOCAL
 A15 (NODE:52 ) -->
```

Resources of Block B

Resource allocation to macro cells:

```
 B0  (PIN :21 ) -->
        PIN OUTPUT                :
RANGO (pts=1)
        OE PT             : HABILITA
        STEERING          : LOCAL
 B1  (NODE:53 ) -->
 B2  (PIN :20 ) -->
 B3  (NODE:54 ) -->
 B4  (PIN :19 ) -->
 B5  (NODE:55 ) -->
 B6  (PIN :18 ) -->
 B7  (NODE:56 ) -->
 B8  (PIN :17 ) -->
        PIN INPUT         : Q11
        PIN OUTPUT                :
Q11.REG (pts=2)
        CLOCK PT          : R2
        STEERING          : LOCAL
 B9  (NODE:57 ) -->
 B10 (PIN :16 ) -->
        PIN INPUT         : Q10
        PIN OUTPUT                :
Q10.REG (pts=2)
        CLOCK PT          : R2
        STEERING          : LOCAL
 B11 (NODE:58 ) -->
 B12 (PIN :15 ) -->
        PIN INPUT         : Q9
        PIN OUTPUT                :
Q9.REG (pts=3)
        CLOCK PT          : R2
        STEERING          : LOCAL
 B13 (NODE:59 ) -->
 B14 (PIN :14 ) -->
        PIN INPUT         : Q8
        PIN OUTPUT                :
Q8.REG (pts=2)
        CLOCK PT          : R2
        STEERING          : LOCAL
 B15 (NODE:60 ) -->
```

Resources of Block C

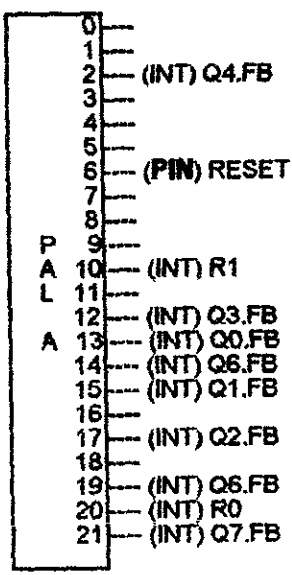
Resource allocation to macro cells:

```
 C0  (PIN :24 ) -->
        PIN OUTPUT                :
CUENTA833 (pts=4)
        STEERING          : LOCAL
 C1  (NODE:61 ) -->
 C2  (PIN :25 ) -->
        STEERING          : DOWN2
 C3  (NODE:62 ) -->
 C4  (PIN :26 ) -->
        PIN OUTPUT                :
CUENTA8 (pts=9)
        STEERING          : LOCAL
 C5  (NODE:63 ) -->
 C6  (PIN :27 ) -->
        INTERNAL FEEDBACK :
R0.REG (pts=2)
        CLOCK PT          : R1
        STEERING          : UP2
 C7  (NODE:64 ) -->
 C8  (PIN :28 ) -->
        STEERING          : UP2
 C9  (NODE:65 ) -->
 C10 (PIN :29 ) -->
 C11 (NODE:66 ) -->
 C12 (PIN :30 ) -->
 C13 (NODE:67 ) -->
 C14 (PIN :31 ) -->
 C15 (NODE:68 ) -->
```

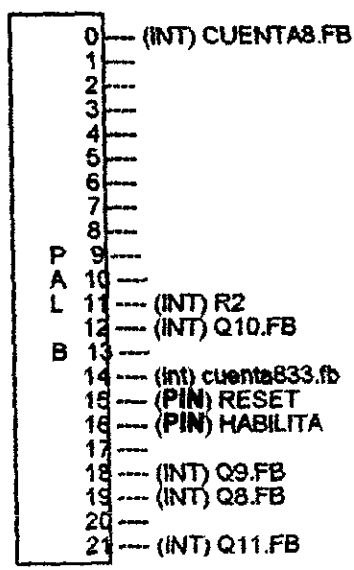
```
------------------------------
    Resources of Block D
------------------------------
Resource allocation to macro cells:
 D0  (PIN :43 ) -->
        INTERNAL FEEDBACK :
R1.REG (pts=2)
        CLOCK PT        : R2
        STEERING        : LOCAL
 D1  (NODE:69 ) -->
 D2  (PIN :42 ) -->
        INTERNAL FEEDBACK : R2
(pts=2)
        STEERING        : LOCAL
 D3  (NODE:70 ) -->
 D4  (PIN :41 ) -->
 D5  (NODE:71 ) -->
 D6  (PIN :40 ) -->
 D7  (NODE:72 ) -->
 D8  (PIN :39 ) -->
 D9  (NODE:73 ) -->
 D10 (PIN :38 ) -->
 D11 (NODE:74 ) -->
 D12 (PIN :37 ) -->
 D13 (NODE:75 ) -->
 D14 (PIN :36 ) -->
 D15 (NODE:76 ) -->
```

En esta sección se muestra la conexión de cada señal a los pines y nodos de cada uno de los bloques PAL.

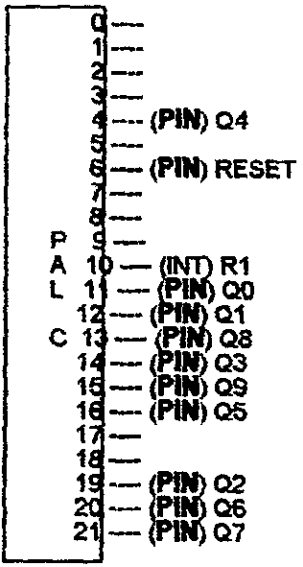
```
------------------------------
    Routing Results
------------------------------
```



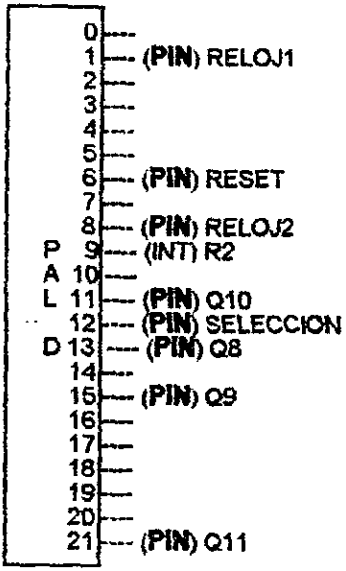
11 of 22 PAL inputs used
(utilization = 50.0%)



9 of 22 PAL inputs used
(utilization = 40.9%)



12 of 22 PAL Inputs used
(utilization = 54.5%)



9 of 22 PAL Inputs used
(utilization = 40.9%)

En la siguiente tabla se muestran cada una de las señales utilizadas. a que bloque PAL fueron asignadas y las señales que dependen de ella

Routing Table

PAL	SIGNAL	FANOUT
.B.D	R2	(5) R1.C Q8.C Q9.C Q10.C Q11.C
A.C.	R1	(5) R0.C Q4.C Q6.C Q6.C Q7.C
...D	RELOJ1	(1) R2
...D	SELECCION	(1) R2
...D	RELOJ2	(1) R2
..CD	Q8	(3) CUENTA8 CUENTA833 R1.REG
..CD	Q9	(2) CUENTA833 R1.REG
...D	Q10	(1) R1.REG
...D	Q11	(1) R1.REG
ABCD	RESET	(14) R1.REG R0.REG Q8.REG Q9.REG Q10.REG Q11.REG Q4.REG Q6.REG Q6.REG Q7.REG Q0.REG Q1.REG Q2.REG Q3.REG
..C.	Q4	(3) CUENTA8 CUENTA833 R0.REG
..C.	Q5	(3) CUENTA8 CUENTA833 R0.REG
..C.	Q6	(3) CUENTA8 CUENTA833 R0.REG
..C.	Q7	(3) CUENTA8 CUENTA833 R0.REG
.B..	Q8.FB	(4) Q8.REG Q9.REG Q10.REG Q11.REG

Routing Table

PAL	SIGNAL	FANOUT
.B..	Q9.FB	(3) Q8.REG Q9.REG Q11.REG
.B..	Q10.FB	(4) Q8.REG Q9.REG Q10.REG Q11.REG
.B..	Q11.FB	(4) Q8.REG Q9.REG Q10.REG Q11.REG
A...	Q4.FB	(4) Q4.REG Q6.REG Q6.REG Q7.REG
A...	Q6.FB	(3) Q4.REG Q6.REG Q7.REG
A...	Q6.FB	(4) Q4.REG Q6.REG Q6.REG Q7.REG
A...	Q7.FB	(4) Q4.REG Q6.REG Q6.REG Q7.REG
A...	Q0.FB	(4) Q0.REG Q1.REG Q2.REG Q3.REG
A...	Q1.FB	(3) Q0.REG Q1.REG Q3.REG
A...	Q2.FB	(4) Q0.REG Q1.REG Q2.REG Q3.REG
A...	Q3.FB	(4) Q0.REG Q1.REG Q2.REG Q3.REG
A...	R0	(4) Q0.C Q1.C Q2.C Q3.C
..C.	Q0	(2) CUENTA8 CUENTA833
..C.	Q1	(2) CUENTA8 CUENTA833
..C.	Q2	(2) CUENTA8 CUENTA833
..C.	Q3	(2) CUENTA8 CUENTA833
.B..	CUENTA8.FB	(1) RANGO
.B..	CUENTA833.FB	(1) RANGO
.B..	HABILITA	(1) RANGO.OE

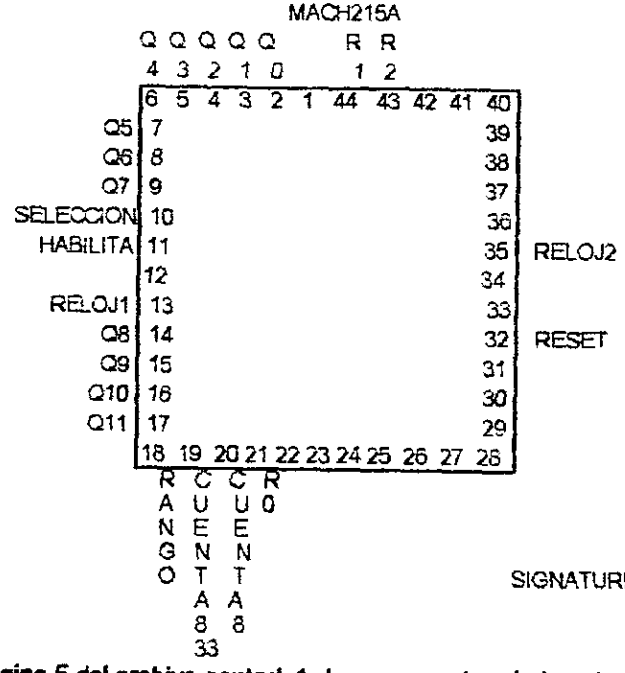
En esta sección se muestran las rutas internas necesarias para la lógica que se estableció en el programa fuente.

Current partitioning requires 41 routes.

$DEVICE MACH215A fit contado1.tt3
$PINS 20 RELOJ1:13 SELECCION:10 RELOJ2:35 RESET:32 HABILITA:11
CUENTA833:24
CUENTA8+:26 Q7:9 Q4:6 Q8:14 Q3:5 Q0:2 Q5:7 Q1:3 Q9:15 Q11:17
Q6:8 Q2:4 Q10:16 RANGO:21
$NODES 3 R1:43 R0:27 R2:42

En el archivo de documento , contado1.doc se tiene información muy similar a la expuesta anteriormente. A continuación se muestra la página 3 del archivo contado1.doc con la disposición de las señales de entrada y salida en los pines correspondientes. En la página 1 y 2 se tienen las ecuaciones de cada señal, que ya se mostraron en el archivo contado1.fit.

ABEL-EDU 6.03.005 - Device Utilization Chart Mon Nov 24 10:41:46 1997
Contador de 0 a 999
Celia Rosa Fierro Santillan y Jose Filiberto Lule Flores
==== MACH215A Chip Diagram ====



En la página 5 del archivo contado1.doc se muestra el pin asignado a cada señal así como los minitérminos utilizados, el número máximo de minitérminos permitidos y el número de minitérminos no utilizado.

ABEL-EDU 6.03.005 - Device Utilization Chart Mon Nov 24 10:41:46 1997
Contador de 0 a 999
Celia Rosa Fierro Santillan y Jose Filiberto Lule Flores
==== MACH215A Product Terms Distribution ====

Signal Name	PIN Assigned	Terms Used	Terms Max	Terms Unused
R2	42	2	4	2
CUENTA8	26	9	12	3
CUENTA833	24	4	4	0
RANGO	21	1	4	3
R1.REG	43	2	4	2

Signal Name	PIN Assigned	Terms Used	Terms Max	Terms Unused	
R0.REG		27	2	4	2
Q8.REG		14	2	4	2
Q9.REG		15	3	4	1
Q10.REG		16	2	4	2
Q11.REG		17	2	4	2
Q4.REG		6	2	4	2
Q6.REG		7	3	4	1
Q6.REG		8	2	4	2
Q7.REG		9	2	4	2
Q0.REG		2	2	4	2
Q1.REG		3	3	4	1
Q2.REG		4	2	4	2
Q3.REG		5	2	4	2

Después se tiene una lista de las señales de entrada y de las señales que fueron colocados en pines bidireccionales. Estos últimos son los pines donde hay macroceldas.

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

Signal Name	PIN	PIN Type
R2	42	BIDIR
R1	43	BIDIR
RELOJ1	13	CLK/IN
SELECCION	10	INPUT
RELOJ2	35	CLK/IN
Q8	14	BIDIR
Q9	15	BIDIR
Q10	16	BIDIR
Q11	17	BIDIR
RESET	32	INPUT
Q4	6	BIDIR
Q5	7	BIDIR
Q6	8	BIDIR
Q7	9	BIDIR
Q0	2	BIDIR
Q1	3	BIDIR
Q2	4	BIDIR
Q3	5	BIDIR
R0	27	BIDIR
HABILITA	11	INPUT

Por último, en la página 6 del archivo contado1.doc se muestra una lista de los recursos no utilizados en el circuito MACH216.

ABEL-EDU 6.03.005 - Device Utilization Chart Mon Nov 24 10:41:46 1997

Contador de 0 a 999

Celia Rosa Fierro Santillan y Jose Filiberto Lule Flores

==== MACH215A Unused Resources ====

PIN Number	PIN Type	Product Terms	Flip-flop Type
33	INPUT	-	-
45	IN_REG	-	D,LAT
46	IN_REG	-	D,LAT
47	IN_REG	-	D,LAT
48	IN_REG	-	D,LAT
49	IN_REG	-	D,LAT
50	IN_REG	-	D,LAT
51	IN_REG	-	D,LAT
52	IN_REG	-	D,LAT

==== MACH215A Unused Resources ====

PIN Number	PIN Type	Product Terms	Flip-flop Type
53	IN_REG	-	D,LAT
54	IN_REG	-	D,LAT
55	IN_REG	-	D,LAT
56	IN_REG	-	D,LAT
57	IN_REG	-	D,LAT
58	IN_REG	-	D,LAT
59	IN_REG	-	D,LAT
60	IN_REG	-	D,LAT
61	IN_REG	-	D,LAT
62	IN_REG	-	D,LAT
63	IN_REG	-	D,LAT
64	IN_REG	-	D,LAT
65	IN_REG	-	D,LAT
66	IN_REG	-	D,LAT
67	IN_REG	-	D,LAT
68	IN_REG	-	D,LAT
69	IN_REG	-	D,LAT
70	IN_REG	-	D,LAT
71	IN_REG	-	D,LAT
72	IN_REG	-	D,LAT
73	IN_REG	-	D,LAT
74	IN_REG	-	D,LAT
75	IN_REG	-	D,LAT
76	IN_REG	-	D,LAT

ARCHIVO JEDEC

ABEL-EDU 6.03.005 Data I/O Corp. JEDEC file for: MACH215A V
Created on: Mon Nov 24 10:41:46 1997
Contador de 0 a 999
Celia Rosa Fierro Santillan y Jose Filiberto Luis Flores
*
QP44* QF11936* QV181* F0*
X0*
NOTE Table of **PIN** names and numbers*
NOTE **PINS** RELOJ1:13 SELECCION:10 RELOJ2:35 RESET:32 HABILITA:11
CUENTA833:24*
NOTE **PINS** CUENTA8:26 Q7:9 Q4:6 Q8:14 Q3:5 Q0:2 Q5:7 Q1:3 Q9:15 Q11:17*
NOTE **PINS** Q6:8 Q2:4 Q10:16 RANGO:21*
NOTE Table of **NODE** names and numbers*
NOTE **NODES** R1:43 R0:27 R2:42*
L0000 0101000010001000100001011010000100110100000100*
L0044 00000000000011000000010000000000000000000000*
L0088 11111111111111111111111111111111111111111111*
L0132 11111111111111111111111111111111111111110111*
L0264 11111111111101111111111101101101111111111111*
L0308 11111111111101111111111100111101110111111111*
L0440 11111111111111111111111111111111111111111111*
L0484 11111111111111111111111111111111111111110111*
L0616 11111111111101111111111111011011110111111111*
L0660 11111111111101111111111101101110110111111111*
L0704 11111111111101111111111101011011111111111111*
L0792 11111111111111111111111111111111111111111111*
L0836 11111111111111111111111111111111111111110111*
L0968 11111111111101111111111101101111111011111111*
L1012 11111111111101111111111101011111101111111111*
L1144 11111111111111111111111111111111111111111111*
L1188 11111111111111111111111111111111111111110111*
L1320 11111111111101111111111101011111111111111111*
L1364 11111111111101111111111001110110111011111111*
L1496 11111111111111111111111111111111111111111111*
L1540 11111111111111111101111111111111111111111111*
L1672 11111011111101111111111111101111111111011101*
L1716 11110111111101111111111111101111111101110110*
L1848 11111111111111111111111111111111111111111111*
L1892 11111111111111111101111111111111111111111111*
L2024 11111011111101111111111111101111111111101111*
L2068 11110111111101111111111101111111111011101101*
L2112 11110111111101111111111111101111111111111110*
L2200 11111111111111111111111111111111111111111111*
L2244 11111111111111111101111111111111111111111111*
L2376 11110111111101111111111111111111111111101101*
L2420 11110111111101111111111111111111111111011110*
L2552 11111111111111111111111111111111111111111111*
L2596 11111111111111111101111111111111111111111111*
L2728 11110111111101111111111111111111111111111110*
L2772 11110111111101111111111111101111111101110110*
L2904 0001011100*
L2914 0001011101*
L2924 0001011110*
L2934 0001011111*
L2944 0001011100*
L2954 0001011101*

L2964 0001011110*
L2974 0001011111*
L2984 0100000000000000010000001001001100010000000100*
L3028 0000000000000000000000001010000111100010100001*
L3072 1111111111111111111111111111111101111111111111*
L3248 0111111111111111111111111111101111111111111111*
L3600 1111111111111111111111111111111111111111111111*
L3644 1111111111111111111111111111111111111111111111*
L3688 1111111111111111111111111111111111111111111111*
L3732 1111111111111111111111111111111111111111111111*
L3952 1111111111111111111111111111111111111111111111*
L3996 1111111111111111111111111111111111111111111111*
L4040 1111111111111111111111111111111111111111111111*
L4084 1111111111111111111111111111111111111111111111*
L4304 1111111111111111111111111111111111111111111111*
L4348 1111111111111111111111111111111111111111111111*
L4392 1111111111111111111111111111111111111111111111*
L4436 1111111111111111111111111111111111111111111111*
L4480 1111111111111111111111111111111111111111111111*
L4524 1111111111111111111101111111111111111111111111*
L4656 1111111111111111111111111111101111111101110*
L4700 1111111111111111111111101111101111100111110*
L4832 1111111111111111111111111111111111111111111111*
L4876 1111111111111111111101111111111111111111111111*
L5008 1111111111111111111111101111101111111101101*
L5052 1111111111111111111101111101111111101110*
L5184 1111111111111111111111111111111111111111111111*
L5228 1111111111111111111101111111111111111111111111*
L5360 1111111111111111111111101111101111101101111*
L5404 1111111111111111111111101111101111101101101*
L5448 1111111111111111111111111111101111101101110*
L5536 1111111111111111111111111111111111111111111111*
L5580 1111111111111111111101111111111111111111111111*
L5712 1111111111111111111101111101111101101101*
L5756 1111111111111111111111101111101111100111110*
L5888 1101011100*
L5898 1101011101*
L5908 1101011110*
L5918 1101011111*
L5928 0001011100*
L5938 0001011101*
L5948 0001011110*
L5958 0001011111*
L5968 0000000000001000100001000000000000000010000000*
L6012 0000000010001100000001101011101110000010101*
L6056 1111111111111111111111111111111111111111111111*
L6232 1111111111111111111101111111111111111111111111*
L6276 1111111101111111111111101010101011111101111*
L6320 1111111101111111111111101110111010111101001*
L6364 1111111101111111111111101110111010111101110*
L6584 1111111111111111111101111111111111111111111*
L6628 1111111111111111111111101111111111111111111*
L6672 1111111111111111111111111111111111111111011111*
L6716 1111111111111111111111111101111111111111111111*
L6760 1111111111111111111111111111111111111111111111*
L6936 1111111111111111111111101111111111111111111*
L6980 1111111101111111111111111111111111111111111111*
L7024 1111111111111111111111111111111101111111111*

L7068 11111111111111111111111111111111111111111110111"
L7156 11111111111111111111011111111111111111111111"
L7288 11111111101111111111111111011101110111111101001"
L7640 11111111111101111111111111111111111111111111"
L7684 11111111011111111111111111111111101111111001"
L7992 11111111111111111111111111111111111111111111"
L8036 11111111111111111111111111111111111111111111"
L8080 11111111111111111111111111111111111111111111"
L8124 11111111111111111111111111111111111111111111"
L8344 11111111111111111111111111111111111111111111"
L8388 11111111111111111111111111111111111111111111"
L8432 11111111111111111111111111111111111111111111"
L8476 11111111111111111111111111111111111111111111"
L8696 11111111111111111111111111111111111111111111"
L8740 11111111111111111111111111111111111111111111"
L8784 11111111111111111111111111111111111111111111"
L8828 11111111111111111111111111111111111111111111"
L8872 1101011100"
L8882 1101101101"
L8892 1101011110"
L8902 0001111111"
L8912 1101111100"
L8922 1101011101"
L8932 1101011110"
L8942 1101011111"
L8952 00100000000001000110110000000100010101010100"
L8996 00010000000011000101001100110011000000000011"
L9084 11111111111111111101111111111111111111111111"
L9216 11111111111111111111101101111011111111111101"
L9260 11111111111101111111111111111111111111111111"
L9568 11011111111111111111111101111111111111111111"
L9612 11111111111111101111111011111111111111111111"
L9920 11111111111111111111111111111111111111111111"
L9964 11111111111111111111111111111111111111111111"
L10008 11111111111111111111111111111111111111111111"
L10052 11111111111111111111111111111111111111111111"
L10272 11111111111111111111111111111111111111111111"
L10316 11111111111111111111111111111111111111111111"
L10360 11111111111111111111111111111111111111111111"
L10404 11111111111111111111111111111111111111111111"
L10624 11111111111111111111111111111111111111111111"
L10668 11111111111111111111111111111111111111111111"
L10712 11111111111111111111111111111111111111111111"
L10756 11111111111111111111111111111111111111111111"
L10976 11111111111111111111111111111111111111111111"
L11020 11111111111111111111111111111111111111111111"
L11064 11111111111111111111111111111111111111111111"
L11108 11111111111111111111111111111111111111111111"
L11328 11111111111111111111111111111111111111111111"
L11372 11111111111111111111111111111111111111111111"
L11416 11111111111111111111111111111111111111111111"
L11460 11111111111111111111111111111111111111111111"
L11680 11111111111111111111111111111111111111111111"
L11724 11111111111111111111111111111111111111111111"
L11768 11111111111111111111111111111111111111111111"
L11812 11111111111111111111111111111111111111111111"
L11856 0001011100"
L11866 1101011101"

```
L11876 1101011110'
L11886 1101011111'
L11896 1101011100'
L11906 1101011101'
L11916 1101011110'
L11926 1101011111'
V0001 NLLLLLLLLOONXLLLLXXXZNNXXXXXXXXXXNXXXXXXXXN'
...
V0181 NLLLLHLLH11NXLLLLXXXHNNXXXXXXXX0XNCXXXXXXXN'
CB18A'
```

DISEÑO DEL COMPARADOR

Además del contador, se requiere de un circuito que compare *TE* con *TC*, el contador maneja dos relojes distintos, uno para *TE* y otro para *TC*, así que cuando se ha terminado de contar *TE*. los bits correspondientes entran al comparador y se almacenan en nodos internos. A continuación se reciben los bits de cuenta para *TC* y se comparan con *TE*. Además. se deben tener dos señales de control del circuito. una que determine en que momento se debe guardar el valor de *TE* y cuando se debe comparar, ya que hay un momento, cuando se guarda el valor de *TE* que los bits de Q0 a Q11 son iguales a los de C0 a C11, de tal manera que la salida de comparación nos indicaría que *TC* y *TE* son iguales, lo cual es incorrecto porque aún no se ha empezado a realizar el conteo de *TC*. Las señales de entrada y salida de este circuito son las siguientes:

Q0,Q1,Q2,Q3	Bits de las centenas de entrada desde el contador.
Q4,Q5,Q6,Q7	Bits de las decenas de entrada desde el contador.
Q8,Q9,Q10,Q11	Bits de las unidades de entrada desde el contador.
ALMACENA	Señal de entrada, cuando vale 1 se almacenan los bits de Q0 a Q11 en C0 hasta C11.
COMPARA	Señal de entrada que determina que TE ya fue almacenado y que se está realizando la cuenta de TC en el contador, para que pueda ser realizada la comparación.
C0,C1,C2,C3	Bits donde se almacenan las centenas de TE.
C4.C5,C6,C7	Bits donde se almacenan las decenas de TE.
C8,C9,C10,C11	Bits donde se almacenan las unidades de TE.
IGUALES	Esta salida toma el valor de 0 si TE y TC son diferentes y el valor de 1 cuando TE y TC son iguales.

El diagrama a bloques del circuito queda como se muestra en la figura 6.6.

figura 6.6 diagrama a bloques del comparador para la máquina cortadora de cartón.

Programa en **ABEL**

```
module compara1
title 'Comparador de 12 bits para el control de la máquina cortadora de carton
      Celia Rosa Fierro Santillan y Jose Filiberto Lule Flores'
DCOMP1 DEVICE 'mach215a';
ALMACENA PIN 10; COMPARA PIN 11;"Entradas de control
Q0 PIN 2;Q1 PIN 3;Q2 PIN 4;Q3 PIN 5;
Q4 PIN 6;Q5 PIN 7;Q6 PIN 8;Q7 PIN 9;
Q8 PIN 14;Q9 PIN 15;Q10 PIN 16;Q11 PIN 17;"Entradas desde el contador
C0,C1,C2,C3,C4,C5,C6,C7,C8 ,C9,C10,C11 NODE ISTYPE 'REG';"Bits donde se
almacena TE
I1,I2,I3 NODE ISTYPE 'COM';
IGUALES PIN 18 ISTYPE 'COM';"Señal que indica que TE = TC
UNIDADES = [Q8,Q9,Q10,Q11];
DECENAS  = [Q4,Q5,Q6 ,Q7 ];
CENTENAS = [Q0,Q1,Q2 ,Q3 ];
UNI = [C8,C9,C10,C11];
DEC = [C4,C5,C6 ,C7 ];
CEN = [C0,C1,C2 ,C3 ];
EQUATIONS
UNI.CLK = ALMACENA;
DEC.CLK  = ALMACENA;
CEN.CLK = ALMACENA;
UNI := UNIDADES;
DEC := DECENAS;
CEN := CENTENAS;
I1 = (UNIDADES == UNI);
I2 = (DECENAS == DEC);
I3 = (CENTENAS == CEN);
IGUALES = I1&I2&I3;
IGUALES.OE = COMPARA;
TEST_VECTORS ([ALMACENA,COMPARA,CENTENAS,DECENAS,UNIDADES]->
[CEN,DEC,UNI,IGUALES])
        [0,0,.X.,.X.,.X.]->[ 0 , 0 , 0 ,.Z.];
        [1,0, 3 , 3 , 2 ]->[ 3 , 3 , 2 ,.Z.];
        [1,1, 2 , 2 , 2 ]->[ 3 , 3 , 2 , 0 ];
        [1,1, 3 , 3 , 2 ]->[ 3 , 3 , 2 , 1 ];
END
```

Archivo JEDEC

ABEL-EDU 6.03.005 Data I/O Corp. JEDEC file for: MACH215A V
Created on: Mon Nov 24 10:52:09 1997
Comparador de 12 bits para el control de la máquina cortadora de carton
Celia Rosa Fierro Santillan y Jose Filiberto Luie Flores
*

QP44* QF11936* QV4* F0*
X0*
NOTE Table of **PIN** names and numbers*
NOTE **PIN**S ALMACENA:13 Q8:35 Q9:25 Q10:10 Q11:11 Q4:36 Q5:32 Q6:37 Q7:38
Q0:39*
NOTE **PIN**S Q1:40 Q2:41 Q3:42 COMPARA:33 IGUALES:43*
NOTE Table of **NODE** names and numbers*
NOTE **NODE**S C3:70 C2:71 C1:72 C0:73 C7:74 C6:75 C5:17 C4:76 C11:6*
NOTE **NODE**S C10:7 C9:62 C8:8 I1:2 I2:21 I3:24 I3__15:27 I2__16:18*
NOTE **NODE**S I1__17:5*
L0000 10001000100000000110001100010010000001010101 0*
L0044 00000000000000000010000000001000001100000000 0*
L0264 11111111111110111011101101101101010111111111 1*
L0308 11111111111110111011101101101110100111111111 1*
L0352 11111111111110111011101101101101011011111111 1*
L0396 11111111111110111011101101101110101011111111 1*
L0616 11111111111111111111111111111111111111011111 1*
L0968 11111111111101110111011010110110101011111111 1*
L1012 11111111111101110111011010110110100111111111 1*
L1056 11111111111101110111011010110101101011111111 1*
L1100 11111111111101110111011010111010101011111111 1*
L1320 11111111111101110111011010111010101011111111 1*
L1364 11111111111101110111011010111010011111111111 1*
L1408 11111111111101110111011011100111010110111111 1*
L1452 11111111111101110111011011100111101010111111 1*
L1672 11111111111101110111011010111010101011111111 1*
L1716 11111111111101110111011010111010011111111111 1*
L1760 11111111111101110111011011100111010110111111 1*
L1804 11111111111101110111011011100111101010111111 1*
L2024 11111111111111111111111111111111101111111111 1*
L2376 11111111111111111111111101111111111111111111 1*
L2728 11111111111111101111111111111111111111111111 1*
L2904 1101011101*
L2914 1101111111*
L2924 1101101101*
L2934 1101011111*
L2944 0000111101*
L2954 0000111111*
L2964 0000111101*
L2974 1101111111*
L2984 00000011001110111000001010101010101000100000*
L3028 00000001000100010000000011000111111100000001*
L3248 11111101110111101111111101111110011011111110*
L3292 11111110110111101111111111011111001101111110*
L3336 11111101111011101111111110111111010101111110*
L3380 11111110111011101111111110111111010101111110*
L3600 11111111111111111111111111111101111111111111*
L3952 11111101110111011111111110111111001011111110*
L3996 11111110110111011111111110111111001011111110*
L4040 11111101111011011111111101111110100111111110*
L4084 11111110111011011111111101111110100111111110*

L4304 11111101110111011111111110111111010101111111101*
L4348 11111110110111011111111111011111010101111111101*
L4392 11111101111011011111111110111111011001111111101*
L4436 11111110111011011111111111011111011001111111101*
L4656 11111101110111011111111110111111010110111111101*
L4700 11111110110111011111111110111111010110111111101*
L4744 11111101111011011111111101111110110101111111101*
L4788 11111110111011011111111110111110110101111111101*
L5008 11111111111111111111111111111111011111111111111*
L5360 11111111111111111111111111111111111111111111111*
L5404 11111111111111111111111111111111111111111111111*
L5448 11111111111111111111111111111111111111111111111*
L5492 11111111111111111111111111111111111111111111111*
L5712 11111111111111111111111111111111111111111111111*
L5756 11111111111111111111111111111111111111111111111*
L5800 11111111111111111111111111111111111111111111111*
L5844 11111111111111111111111111111111111111111111111*
L5888 1101011101*
L5898 1101111111*
L5908 1101101101*
L5918 1101011111*
L5928 0000111100*
L5938 1101111101*
L5948 1101011110*
L5958 1101011111*
L5968 10000000000001001000001001110110101010100110*
L6012 11000000000001000000001101010100000000110011*
L6232 01111111111110111111111001100111111111011110*
L6276 01111111111110111111111010100111111111101110*
L6320 10111111111110111111111001101011111111011110*
L6364 10111111111110111111111010101011111111101110*
L6584 11111111111111111111111111111111111111110111*
L6936 01111111111110111111110101010111111111011110*
L6980 01111111111110111111110110010111111111101110*
L7024 10111111111110111111110101011011111111011110*
L7068 10111111111110111111101100110111111111101110*
L7288 01111111111110111111111010101011111111011101*
L7332 01111111111110111111111011001011111111101101*
L7376 10111111111110111111111010101101111111011101*
L7420 10111111111110111111111011001101111111101101*
L7640 01111111111110111111111100110011111111011101*
L7684 01111111111110111111111010100111111111101101*
L7728 10111111111110111111111001101011111111011101*
L7772 10111111111110111111111010101011111111101101*
L7992 11111111111111111111111111111111111111111111*
L8036 11111111111111111111111111111111111111111111*
L8080 11111111111111111111111111111111111111111111*
L8124 11111111111111111111111111111111111111111111*
L8344 11111111111111111111111111111111111111111111*
L8388 11111111111111111111111111111111111111111111*
L8432 11111111111111111111111111111111111111111111*
L8476 11111111111111111111111111111111111111111111*
L8696 11111111111111111111111111111111111111111111*
L8740 11111111111111111111111111111111111111111111*
L8784 11111111111111111111111111111111111111111111*
L8828 11111111111111111111111111111111111111111111*
L8872 1101011100*
L8882 1101111000*

L8892 1101101101*
L8902 1101011110*
L8912 1101111111*
L8922 1101011100*
L8932 1101011101*
L8942 1101011110*
L8952 00000000000010001000000110101010101001100110*
L8996 00000000000000000000000000000000000000010000*
L9040 11111111111111111111101111111111111111111111*
L9216 11111111111111111111111101111111111011111*
L9568 11111111111111111111111111111111111111111111*
L9612 11111111111111111111111111111111111111111111*
L9656 11111111111111111111111111111111111111111111*
L9700 11111111111111111111111111111111111111111111*
L9920 11111111111111111111111111111111111111111111*
L9964 11111111111111111111111111111111111111111111*
L10008 111111111111111111111111111111111111111111111*
L10052 111111111111111111111111111111111111111111111*
L10272 111111111111111111111111111111111111111111111*
L10316 111111111111111111111111111111111111111111111*
L10360 111111111111111111111111111111111111111111111*
L10404 111111111111111111111111111111111111111111111*
L10624 111111111111111111111111111111111111111111111*
L10668 111111111111111111111111111111111111111111111*
L10712 111111111111111111111111111111111111111111111*
L10756 111111111111111111111111111111111111111111111*
L10976 111111111111111111111111111111111111111111111*
L11020 111111111111111111111111111111111111111111111*
L11064 111111111111111111111111111111111111111111111*
L11108 111111111111111111111111111111111111111111111*
L11328 111111111111111111111111111111111111111111111*
L11372 111111111111111111111111111111111111111111111*
L11416 111111111111111111111111111111111111111111111*
L11460 111111111111111111111111111111111111111111111*
L11680 111111111111111111111111111111111111111111111*
L11724 111111111111111111111111111111111111111111111*
L11768 111111111111111111111111111111111111111111111*
L11812 111111111111111111111111111111111111111111111*
L11856 1101011111*
L11866 1101011000*
L11876 1101011000*
L11886 1101011000*
L11896 1101011000*
L11906 1101011000*
L11916 1101011000*
L11926 1101011000*
V0001 NXXXXXXXXXXN0XXXXXXXXNNXXXXXXXXX0NXXXXXXXXZN*
V0002 NXXXXXXXX10N1XXXXXXXXNNX0XXXXXX00N00110011ZN*
V0003 NXXXXXXXX10N1XXXXXXXXNNX0XXXXXX01N00100010LN*
V0004 NXXXXXXXX10N1XXXXXXXXNNX0XXXXXX01N00110011HN*
C331F*

VI.3.6 ENTRADAS Y SALIDAS DEL CONTROLADOR.

Lo primero es identificar las entradas al controlador que serán las lecturas provenientes de los sensores de posición del papel impreso y la cuchilla giratoria de corte, las señales RANGO e IGUALES provenientes del comparador y del contador, estos se representaron siguiente manera:

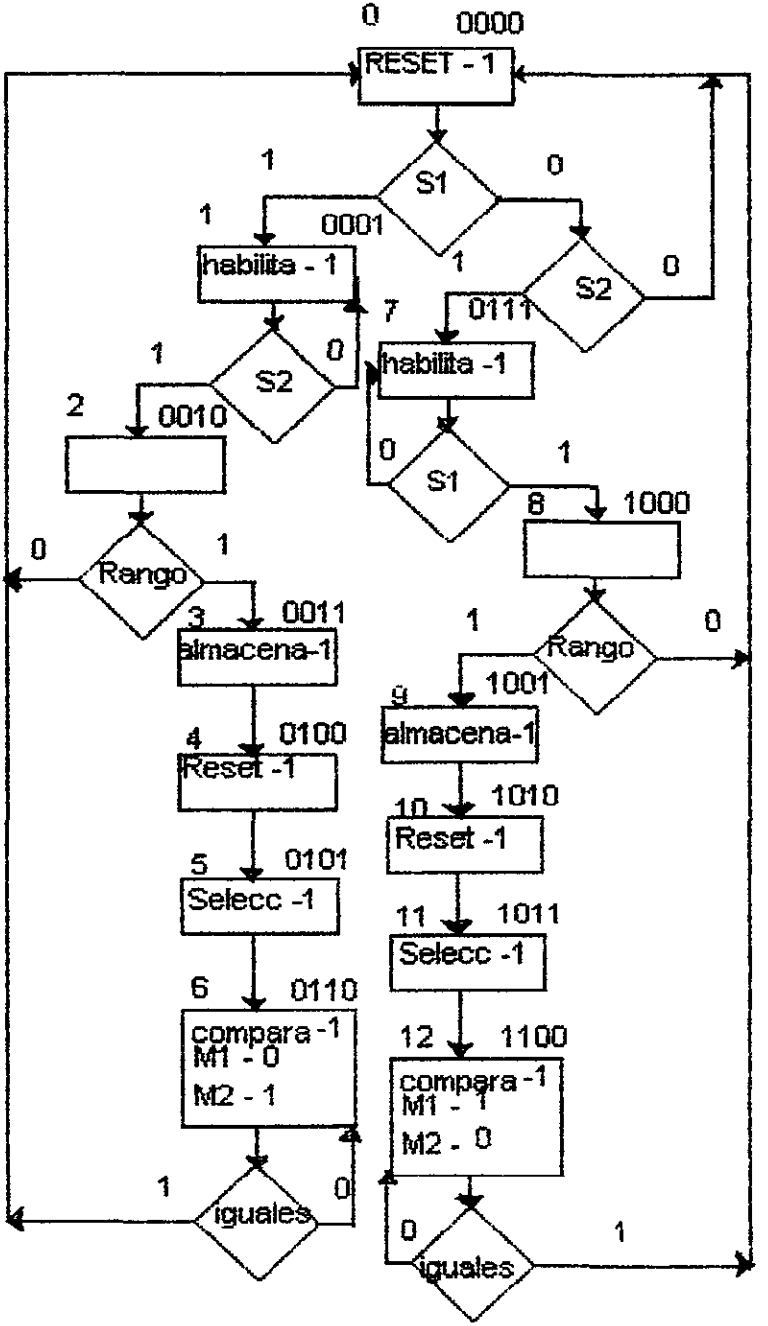
S1 = 0	El papel no está en la posición de lectura del sensor.
S1 = 1	El papel está en la posición de lectura del sensor.
S2 = 0	La cuchilla de corte no está en la posición de lectura del sensor.
S2 = 1	La cuchilla de corte está en la posición de lectura del sensor.
RANGO = 0	TE no está dentro del rango permitido.
RANGO = 1	TE está dentro del rango permitido.

Se requieren dos bits de salida del controlador que permitan al servomotor tener una polarización inversa o directa y que gire en sentido horario o antihorario según se requiera adelantar o atrasar la cuchilla, también se requieren las señales de control HABILITA, SELECCION, ALMACENA y COMPARA para el contador y el comparador. Las salidas quedan representadas de la siguiente manera:

SI M1 = 0 y M2 = 1	El servomotor gira en sentido horario (adelanta la cuchilla).
SI M1 = 1 y M2 = 0	El servomotor gira en sentido antihorario (atrasa a la cuchilla).
SI M1 = 0 y M2 = 0	El servomotor estará desconectado.
HABILITA = 1	El contador puede funcionar
SELECCION = 0	El contador utiliza el reloj de 1 MHz.
SELECCION = 1	El contador utiliza el reloj de 2.083 KHz.
ALMACENA = 1	El comparador almacena *TE*.
COMPARA = 1	El comparador compara *TE* y *TC*.

La combinación M1 = 1 y M2 = 1 no es una salida válida, ya que ocasionaría un cortocircuito en el servomotor.

VI.3.7 CARTA ASM DEL CONTROLADOR



VI.3.8 DIAGRAMA DE ESTADOS DEL CONTROLADOR.

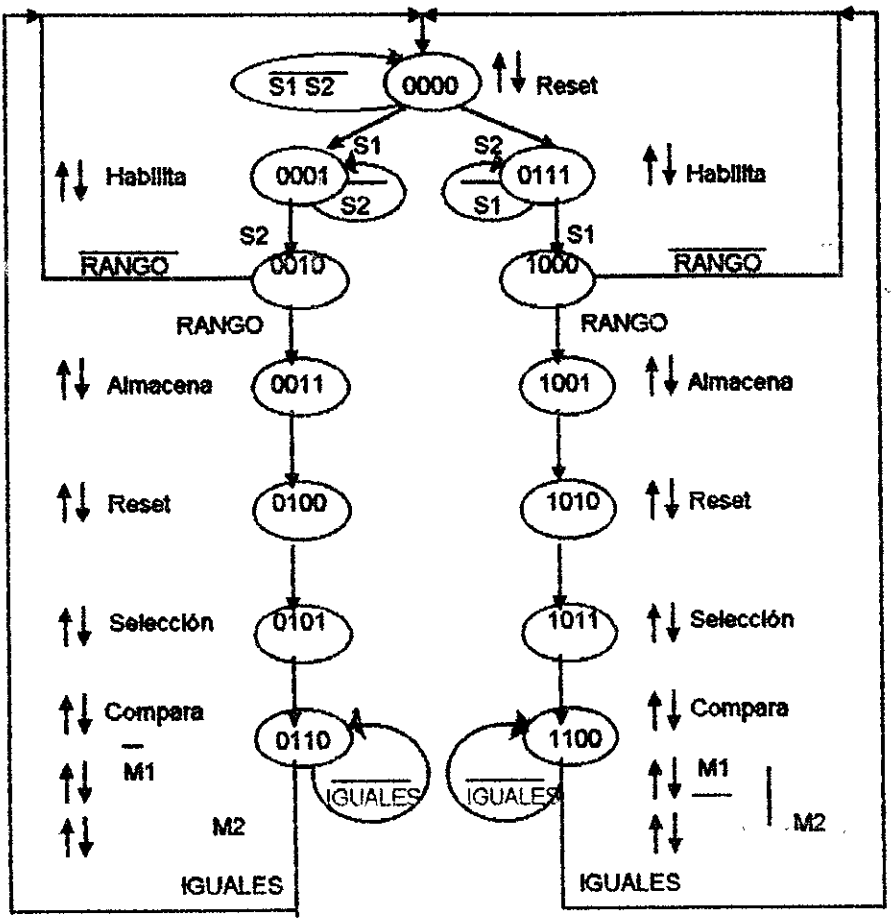


figura 6.7 Diagrama de estados del controlador del servomotor de la máquina cortadora de cartón.

VI.3.9 PROGRAMA DEL CONTROLADOR EN ABEL

```
module control1
title 'CONTROLADOR DEL SERVOMOTOR PARA UNA MAQUINA CORTADORA DE
CARTON
    Celia Rosa Fierro Santillán y José Filiberto Lule Flores';
dispositivo DEVICE 'MACH215A';
S1, S2, PAPEL, CUCHILLA PIN ;"Señales de los sensores
RELOJ PIN; "Entrada de reloj
RANGO, IGUALES PIN;"Entradas desde el contador y el comparador
M1, M2 PIN ISTYPE 'com';"Señales del motor
Q0,Q1, Q2, Q3  PIN ISTYPE 'reg';"Bits de estado
HABILITA,SELECCION,RESET, ALMACENA,COMPARA PIN ISTYPE 'com';"Señales de
control
ESTADO = [Q0,Q1,Q2,Q3];
EQUATIONS
ESTADO.clk = RELOJ;
STATE_DIAGRAM ESTADO
STATE 0: HABILITA = 0;
      SELECCION = 0;
      ALMACENA = 0;
      COMPARA = 0;
      RESET = 1;
      M1 = 0;
      M2 = 0;
      IF  S1 & IS2 THEN  1
        ELSE IF  IS1 &  S2 THEN  7
           ELSE 0;

STATE 1: HABILITA =1:
      RESET = 0;
      IF S2 THEN 2 ELSE 1;
STATE 2: HABILITA = 1;
      IF RANGO THEN 3 ELSE 0;
STATE 3: ALMACENA = 1;
      GOTO 4;
STATE 4: RESET =1;
      GOTO 5;
STATE 5: RESET = 0;
      SELECCION = 1;
      GOTO 6;
STATE 6: SELECCION =1;
      COMPARA =1;
      M1 = 0;
      M2 = 1;
      IF IGUALES THEN 0 ELSE 6;
STATE 7: HABILITA = 1;
      RESET = 0;
      IF S1 THEN 8 ELSE 7;
STATE 8: HABILITA = 1;
      IF RANGO THEN 9 ELSE 0;
```

```
STATE 9: ALMACENA = 1;
     GOTO 10;
STATE 10: RESET = 1;
     GOTO 11;
STATE 11: RESET = 0;
     SELECCION = 1;
     GOTO 12;
STATE 12: SELECCION = 1;
     COMPARA =1;
     M1 = 1;
     M2 = 0;
     IF IGUALES THEN 0 ELSE 11;

TEST_VECTORS ([RELOJ,S1,S2,RANGO,IGUALES] ->
[ESTADO,HABILITA,SELECCION,RESET,ALMACENA,COMPARA,M1,M2])
          [.C., 0, 0 , 0,.X.] -> [ 0,0 , 0 , 1 , 0 , 0, 0, 0 ];
          [.C., 1, 0 , 0,.X.] -> [ 1,1 , 0 , 0 , 0 , 0, 0, 0 ];
          [.C..X..1 , 0..X.] -> [ 2,1 , 0 , 0 , 0 , 0, 0, 0 ];
          [.C..x..x..0,.X.] -> [ 0,0 , 0 , 1 , 0 , 0, 0, 0 ];
          [.C., 1 , 0 ,0,.X.] -> [ 1,1 , 0 , 0 , 0 , 0, 0, 0 ];
          [.C..x., 1 ,1,.X.] -> [ 2,1 , 0 , 0 , 0 , 0, 0, 0 ];
          [.C..x..x., 1 ,.X.] -> [ 3,0 , 0 , 0 , 1 , 0, 0, 0 ];
          [.C..x..x..X..X.] -> [ 4,0 , 0 , 1 , 0 , 0, 0, 0 ];
          [.C..x..x..X..X.] -> [ 5,0 , 1 , 0 , 0 , 0, 0, 0 ];
          [.C..x..x..X., 0 ] -> [ 6,0 , 1 , 0 , 0 , 1, 0, 1 ];
          [.C..x..x..X., 1 ] -> [ 0,0 , 0 , 1 , 0 , 0, 0, 0 ];
          [.C., 0 , 1 ,.X.,.X.] -> [ 7,1 , 0 , 0 , 0 , 0, 0, 0 ];
          [.C., 1 ,.x., 1 ,.X.] -> [ 8,1 , 0 , 0 , 0 , 0, 0, 0 ];
          [.C..x..x., 1 ,.X.] -> [ 9,0 , 0 , 0 , 1 , 0, 0, 0 ];
          [.C..x..x..X..X.] -> [ 10,0 , 0 , 1 , 0 , 0, 0, 0 ];
          [.C..x..x..X..X.] -> [ 11,0 , 1 , 0 , 0 , 0, 0, 0 ];
          [.C..x..x..X., 0 ] -> [ 12,0 , 1 , 0 , 0 , 1, 1, 0 ];
          [.C..x..x..X., 1 ] -> [ 0,0 , 0 , 1 , 0 , 0, 0, 0 ];
END
```

VI.3.10 ARCHIVO JEDEC DEL CONTROLADOR

```
□ABEL-EDU 6.03.005 Data I/O Corp. JEDEC file for: MACH215A V
Created on: Tue Nov 25 00:39:34 1997
CONTROLADOR DEL SERVOMOTOR PARA UNA MAQUINA CORTADORA DE CARTON
     Celia Rosa Fierro Santillán y José  Filiberto Luis Flores
*
QP44* QF11936* QV18* F0*
 X0*
NOTE Table of PIN names and numbers*
NOTE PINS RELOJ:13 S1:11 S2:32 RANGO:10 IGUALES:33 Q3:3 Q2:6 Q0:8 Q1:21*
NOTE PINS M2:20 M1:19 HABILITA:18 SELECCION:17 RESET:16 ALMACENA:15
COMPARA:24*
L0000 10001000100010001000010100000010001010000110*
L0044 00000000010011000000000000000000011000000000*
L0264 11111111101110111111111111110110111111111101111*
```

L0308 11111111101110111111111111110111101111111101111*
L0352 11111111101111111111111101111110111111011111*
L0396 11111111101111111111111101011110111111111111*
L0440 11111111111111111111111111111111111111111111*
L0616 11111111101111111111111111101111011111111011111*
L0660 11111111011111111111111111101110111111111101111*
L0704 11111111011111111111111111101110110111111101111*
L0748 11111111111101111111111111101110101111111101111*
L0968 11111111011111111111111101110111011111111111111*
L1320 11111111111101111111111111101101111111111101111*
L1364 11111111101101111111111111101111101111111101111*
L1408 11111111101111111111111101011110111111111111111*
L1452 11111111011111111111111101101111011111111101111*
L1496 11111111111111111111111111111111111111111111111*
L1672 11111111011111111111111111101101111111111101111*
L1716 11111111101111111111111111101101111111111011111*
L1760 11111111101111111111111111101111011111111011111*
L1804 11111111011111111111111111111101101111111101111*
L2024 11111111011111111111111101110111011111111011111*
L2200 11111111111111111111111111111111111111111111111*
L2376 11111111101111111111111111101111111111111011111*
L2420 11111111101111111111111111111101111111111011111*
L2464 11111111011111111111111111101110101111111101111*
L2508 11111111101111111111111101111111111111111011111*
L2728 11111111011111111111111101110111011111111011111*
L2904 1101101101*
L2914 0000011111*
L2924 1101111101*
L2934 1101101111*
L2944 0000011101*
L2954 1101111100*
L2964 0000011111*
L2974 1101111110*
L2984 10001000100010001000100100010000001010001110*
L3028 00000000010011000000000000000000011000000000*
L3072 11111111111111111111111111111111111111111111111*
L3248 11111111011111111111111111111111011111111011111*
L3292 11111111101111111111011111111110111111111111111*
L3336 11111111111111111111011111111110110111111011111*
L3380 11111111111101111111101111111110101111101111*
L3424 11111111111111111111111111111111111111111111111*
L3600 11111111011111111111101011111111111111111101111*
L3776 11111111111111111111111111111111111111111111111*
L3952 11111111011111111111101111111101111111101111*
L4128 11111111111111111111111111111111111111111111111*
L4304 11111111011111111111101111111110111111011111*
L4480 11111111111111111111111111111111111111111111111*
L4656 11111111011111111111011111111101111111101111*
L4700 11111111101111111111011111111110111111101111*
L4744 11111111011111111111011111111101111111101111*
L4788 11111111101111111111011111111110111111011111*
L4832 11111111111111111111111111111111111111111111111*

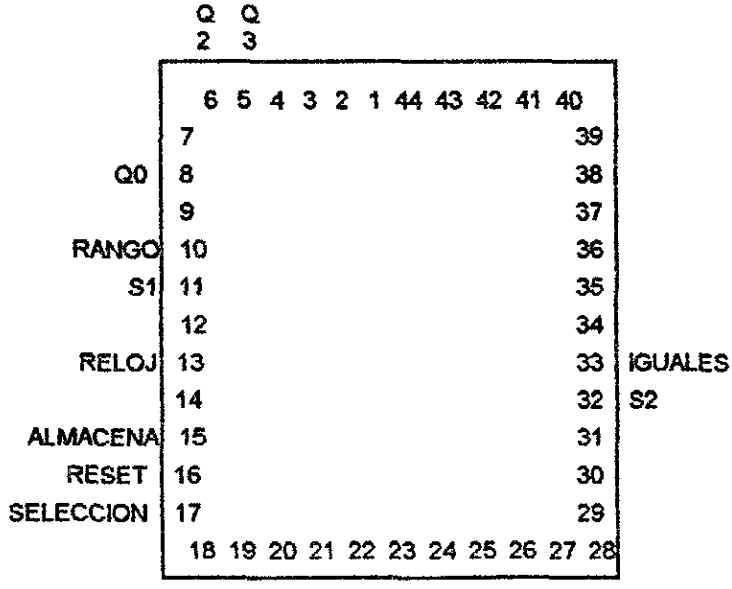
L5008 1111111110111111111110111111111101111111011111*
L5052 1111111101111111111110111111111101111111101111*
L5096 1111111101111111111110111111111101111111101111*
L5140 1111111101111111111110111111111101111111011111*
L5184 1111111111111111111111111111111111111111111111*
L5360 1111111110111111111110111111111101111111011111*
L5404 1111111111111111111110111111111101111111101111*
L5712 1111111110111111111110111111111101111111101111*
L5756 1111111110111111111110111111111101111111011111*
L5888 0000011101*
L5898 1101111100*
L5908 1101111111*
L5918 1101111110*
L5928 1101111101*
L5938 1101111100*
L5948 1101111111*
L5958 1101111110*
L5968 1000100010001000100010100001000010010100111 0*
L6012 0000000001000000000000000000000000000000000000*
L6056 1111111111111111111111111111111111111111111111*
L6232 1111111101111111111110111111111101111111101111*
L6276 1111111101111111111110111111111101111111011111*
L6584 1111111111111111111111111111111111111111111111*
L6628 1111111111111111111111111111111111111111111111*
L6672 1111111111111111111111111111111111111111111111*
L6716 1111111111111111111111111111111111111111111111*
L6936 1111111111111111111111111111111111111111111111*
L6980 1111111111111111111111111111111111111111111111*
L7024 1111111111111111111111111111111111111111111111*
L7068 1111111111111111111111111111111111111111111111*
L7288 1111111111111111111111111111111111111111111111*
L7332 1111111111111111111111111111111111111111111111*
L7376 1111111111111111111111111111111111111111111111*
L7420 1111111111111111111111111111111111111111111111*
L7640 1111111111111111111111111111111111111111111111*
L7684 1111111111111111111111111111111111111111111111*
L7728 1111111111111111111111111111111111111111111111*
L7772 1111111111111111111111111111111111111111111111*
L7992 1111111111111111111111111111111111111111111111*
L8036 1111111111111111111111111111111111111111111111*
L8080 1111111111111111111111111111111111111111111111*
L8124 1111111111111111111111111111111111111111111111*
L8344 1111111111111111111111111111111111111111111111*
L8388 1111111111111111111111111111111111111111111111*
L8432 1111111111111111111111111111111111111111111111*
L8476 1111111111111111111111111111111111111111111111*
L8696 1111111111111111111111111111111111111111111111*
L8740 1111111111111111111111111111111111111111111111*
L8784 1111111111111111111111111111111111111111111111*
L8828 1111111111111111111111111111111111111111111111*
L8872 1101011101*
L8882 1101011100*

L8892 1101011111*
L8902 1101011110*
L8912 1101011101*
L8922 1101011100*
L8932 1101011111*
L8942 1101011110*
L8952 00000000000010001000001010101010100101010110*
L9216 1111111111111111111111111111111111111111111111*
L9260 1111111111111111111111111111111111111111111111*
L9304 1111111111111111111111111111111111111111111111*
L9348 1111111111111111111111111111111111111111111111*
L9568 1111111111111111111111111111111111111111111111*
L9612 1111111111111111111111111111111111111111111111*
L9656 1111111111111111111111111111111111111111111111*
L9700 1111111111111111111111111111111111111111111111*
L9920 1111111111111111111111111111111111111111111111*
L9964 1111111111111111111111111111111111111111111111*
L10008 1111111111111111111111111111111111111111111111*
L10052 1111111111111111111111111111111111111111111111*
L10272 1111111111111111111111111111111111111111111111*
L10316 1111111111111111111111111111111111111111111111*
L10360 1111111111111111111111111111111111111111111111*
L10404 1111111111111111111111111111111111111111111111*
L10624 1111111111111111111111111111111111111111111111*
L10668 1111111111111111111111111111111111111111111111*
L10712 1111111111111111111111111111111111111111111111*
L10756 1111111111111111111111111111111111111111111111*
L10976 1111111111111111111111111111111111111111111111*
L11020 1111111111111111111111111111111111111111111111*
L11064 1111111111111111111111111111111111111111111111*
L11108 1111111111111111111111111111111111111111111111*
L11328 1111111111111111111111111111111111111111111111*
L11372 1111111111111111111111111111111111111111111111*
L11416 1111111111111111111111111111111111111111111111*
L11460 1111111111111111111111111111111111111111111111*
L11680 1111111111111111111111111111111111111111111111*
L11724 1111111111111111111111111111111111111111111111*
L11768 1111111111111111111111111111111111111111111111*
L11812 1111111111111111111111111111111111111111111111*
L11856 1101011101*
L11866 1101011100*
L11876 1101011111*
L11885 1101011110*
L11896 1101011101*
L11906 1101011100*
L11916 1101011111*
L11926 1101011110*
V0001 NXLXXLXLX00NCXLHLLLLLNNLXXXXXXX0XNXXXXXXXXXN*
V0002 NXHXXLXLX01NCXLLLHLLLNNLXXXXXXX0XNXXXXXXXXN*
V0003 NXLXXHXLX0XNCXLLLHLLLNNLXXXXXXX1XNXXXXXXXXN*
V0004 NXLXXLXLX0XNCXLHLLLLLNNLXXXXXXXXXNXXXXXXXXN*
V0005 NXHXXLXLX01NCXLLLHLLLNNLXXXXXXX0XNXXXXXXXXN*

V0006 NXLXXHXLX1XNCXLLLHLLLNNLXXXXXX1XNXXXXXXXXXN*
V0007 NXHXXHXLX1XNCXHLLLLLLNNLXXXXXXXXXNXXXXXXXXN*
V0008 NXLXXLXLXXXNCXLHLLLLHNNLXXXXXXXXXNXXXXXXXXN*
V0009 NXHXXLXLXXXNCXLLHLLLHNNLXXXXXXXXXNXXXXXXXXN*
V0010 NXLXXHXLXXXNCXLLHLLHHNNHXXXXXXXX0NXXXXXXXXN*
V0011 NXLXXLXLXXXNCXLHLLLLLNNLXXXXXXXX1NXXXXXXXXN*
V0012 NXHXXHXLXX0NCXLLLHLLHNNLXXXXXXX1XNXXXXXXXXN*
V0013 NXLXXLXHX11NCXLLLHLLLNNLXXXXXXXXXNXXXXXXXXN*
V0014 NXHXXLXHX1XNCXHLLLLLLNNLXXXXXXXXXNXXXXXXXXN*
V0015 NXLXXHXHXXXNCXLHLLLLLNNLXXXXXXXXXNXXXXXXXXN*
V0016 NXHXXHXHXXXNCXLLHLLLLNNLXXXXXXXXXNXXXXXXXXN*
V0017 NXLXXLXHXXXNCXLLHLHLHNNHXXXXXXXX0NXXXXXXXXN*
V0018 NXLXXLXLXXXNCXLHLLLLLNNLXXXXXXXX1NXXXXXXXXN*
C7676*
□BAA5

VI.3.11 DIAGRAMA DE CHIP DEL CONTROLADOR

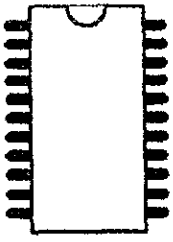
MACH215A



SIGNATURE: N/A

FALTA PAGINA

No. 154



CONCLUSIONES

En electrónica digital existen diversas formas de realizar un diseño como son : el uso de circuitos TTL, microprocesadores, microcontroladores, etc. A lo largo de esta tesis se desarrolló una metodología basada en el uso de circuitos GAL y MACH utilizando el lenguaje de diseño de hardware ABEL.

Esta metodología es muy eficiente con respecto a otras (reducción por álgebra booleana o mapas de Karnaught), ya que un programa en ABEL se puede realizar en unos minutos o en algunas horas, dependiendo de la complejidad del diseño y de la habilidad del programador. Por ejemplo, el diseño de un decodificador sencillo puede tomar unos 20 minutos; diseñar un controlador como el descrito en el capítulo seis nos llevó alrededor de una semana debido a la complejidad de los circuitos.

Con ABEL se pueden crear circuitos "a la medida", se suprimen los cableados y se utilizan menos circuitos integrados. Otra ventaja más es el hecho de que los circuitos GAL son eléctricamente borrables, lo que permite hacer correcciones aún cuando ya se haya grabado el circuito.

Las posibles desventajas de esta metodología radican únicamente en utilizarla en aplicaciones tan sencillas que podrían implementarse en uno o dos circuitos LSI o MSI (comparadores, sumadores o compuertas comerciales), o en aplicaciones tan complejas que sería mejor implementarlas en un microprocesador.

Para el desarrollo de este trabajo se enfrentaron varios problemas, el principal de ellos fue la casi nula información que existe en nuestro país

acerca de PLDs. por lo tanto todos los manuales de PLDs que se usaron como bibliografía básica fueron importados de Estados Unidos.

De los circuitos empleados en esta tesis los GAL son los más comercializados en nuestro país por lo que fueron fáciles de obtener. además tienen un costo promedio de $25.00 lo que los hace muy accesibles. En cambio los circuitos MACH no se comercializan en México.

Todos los diseños de este trabajo que se hicieron en base a circuitos GAL se implementaron en la práctica y tuvieron el funcionamiento esperado. Es importante mencionar que con este tipo de circuitos se deben tomar ciertas precauciones en su manejo para evitar la electricidad estática porque esta podría dañar la información contenida en el circuito (Los circuitos CMOS son sensibles ala electricidad estática). Pensando en dedicar el mayor tiempo posible al desarrollo de este trabajo y contar con todas las facilidades posibles los autores equiparon un pequeño laboratorio en su casa. en el que se contaron con ciertas condiciones mínimas para trabajar con circuitos CMOS como son: un humidificador de ambiente, mesa y bancos conectados a tierra, superficie de trabajo antiestática, piso conectado a una línea de tierra física; se usó ropa de algodón, zapatos y pulseras antiestáticas. Además se contó con las herramientas necesarias como computadora con el software instalado, tablero de pruebas con fuente de voltaje variable, protoboard, cables. caimanes. pinzas. etc. La única herramienta que no se tenía fue el programador de PLDs debido a su alto costo (oscilan entre $5000.00 y $10 000.00 en el mercado nacional). por lo tanto, una vez obtenidos los archivos JEDEC se mandaron a programar los circuitos GAL.

En cuanto a los circuitos MACH aunque podían ser importados a un costo accesible no podían ser grabados en los programadores a los que tuvo acceso debido a su presentación de chip cuadrado de 44 pines; por lo tanto el controlador del capítulo 6 no fue implementado. Uno de los

proyectos a corto plazo para los autores es el diseño de un programador que pueda grabar este tipo de circuitos ya que cuentan con la mayor parte del material y el costo sería menor que si se compra un programador que reúna las características adecuadas.

El hecho de que la mayor parte de los circuitos diseñados y simulados al ser grabados y físicamente probados tuvieron el comportamiento adecuado nos lleva a concluir que los tres últimos circuitos (contador, comparador y controlador para la máquina cortadora de cartón) de haber sido implementados cumplirían con lo esperado en base a que la simulación fue la correcta.

Celia Rosa Fierro Santillán y José Filiberto Lule Flores

FALTA PAGINA

No. 158

BIBLIOGRAFÍA

Morris Mano, M., Diseño digital.
1ª edición, Prentice Hall, México, 1991.

Morris Mano, M., Ingeniería computacional, diseño del hardware.
1ª edición, Prentice Hall, México, 1991.

Pellerin, David y Holley, Michael, Digital design using ABEL.
1ª edición, Prentice Hall, USA, 1994.

Programmable logic devices, databook and design guide, GAL, ECL PAL, high density MAPL, National Semiconductor, Edición 1994, USA

MACH 1,2,3, and 4 family Data Book, high density EECMOS programmable logic, Advanced Micro Devices, Edición 1995, USA.

MACH Databook 1997, Vanitis Corporation,
Edición 1997, USA.

PAL and MACH Devices data book and design guide 1996,
Advanced Micro Devices, Edición 1996, USA.

Programmable Logic data book 1996, Texas Instruments,
Edición 1996, USA.

FALTA PAGINA

No. 160

APÉNDICE A. EL LENGUAJE ABEL

A.1 DECLARACIONES

Las declaraciones son escritas al principio del modulo **ABEL** (después de los enunciados **MODULE** y **TITLE** y antes de cualquier sección de descripción lógica) o en la sección **DECLARATIONS**. Los elementos de diseño, tales como señales y constantes deben ser declaradas en la sección de declaraciones antes de cualquier descripción lógica.

DECLARACIONES DE DISPOSITIVO

ABEL puede programar la lógica en un dispositivo específico que debe ser declarado en la sección de declaraciones. La declaración de dispositivo es opcional, si se usa, debe aparecer antes de las declaraciones de pines o nodos. El siguiente enunciado declara un dispositivo (GAL 16V8) que será usado en un diseño:

```
midisp DEVICE 'G16V8';
```

Declarar un dispositivo en el archivo fuente de **ABEL** no restringe el diseño a un dispositivo particular, pero puede implicar ciertos atributos (como tipo de registros y salidas inversoras) que pueden restringir la asignación de pines para mapear el diseño dentro de un tipo de dispositivo.

DECLARACIONES DE PINES Y NODOS

ABEL maneja dos tipos de señales: pines y nodos. Un pin es una señal que está accesible fuera del módulo actual (en un pin del dispositivo), mientras que un nodo es una señal que puede ser removida (por colapsamiento) durante el proceso de asignación de pines y nodos. Todos los pines y nodos deben ser declarados antes de ser usados. Los siguientes enunciados son ejemplos de declaraciones de pines y nodos:

```
I1,I2,I3 PIN; "Entrada
Pulso PIN ISTYPE 'COM'; "Salida combinacional
Buffer1 NODE ISTYPE 'REG'; "Registro
Q1..Q6  PIN ISTYPE 'REG'; "Uso de un rango
D3,D2,D1 PIN 2,3,4,5; "Número de pines
```

NUMERO DE PINES Y NODOS

Las declaraciones de señales pueden incluir número de pin. Cuando se especifica, el número de pin es escrito para que el compilador genere un archivo intermedio que será usado en el proceso de asignación de pines. El número de pin no tiene significado para el compilador a menos que se especifique el tipo de dispositivo a grabar.

ENTRADAS Y SALIDAS

Las declaraciones en **ABEL** no proveen ningún camino para establecer cuales señales son de entrada, cuales de salida y cuales son bidireccionales. Esta información es determinada por la manera en que se utiliza cada señal en la descripción del diseño. En los ejemplos presentados en esta tesis, seguimos la convención de asignar atributos de señales (usando el enunciado **ISTYPE**) para todas las señales utilizadas como salidas en cada diseño. Esto ayuda a que el programa sea más compresible, sin embargo, alguno de los atributos de señal pueden ser innecesarios.

ATRIBUTOS DE SEÑAL

Las declaraciones de señales de salida pueden incluir el enunciado **ISTYPE** o un modificador **ISTYPE**, los cuales asignan ciertos atributos a una señal. Un modificador **ISTYPE** establece los atributos en una cadena de caracteres que sigue a las palabras reservadas **PIN** o **NODE**;

```
Q3      PIN ISTYPE 'REG';
```

Cuando una lista de señales es especificada en la declaración, el modificador **ISTYPE** se aplica a todas las señales de la lista.

Un enunciado **ISTYPE** es simplemente otra forma del modificador **ISTYPE** que puede ser usada para señales que ya fueron definidas en una declaración de pin o nodo, por ejemplo:

```
Q3      PIN;
Q3      ISTYPE 'REG';
```

En un enunciado **ISTYPE** se pueden combinar varios atributos de señal, separando cada atributo por una coma, por ejemplo:

```
Misalida PIN ISTYPE 'REG, BUFFER';
```

Los siguientes atributos de señal están disponibles en **ABEL**

ISTYPE 'REG'

La salida es tipo latch o flip-flop D activado por reloj y puede ser descrita usando descripciones lógicas de pin a pin. (Los operadores de asignación := y :> son usados en las descripciones lógicas para describir salidas con registro de pin a pin).

Que el diseño sea implementado usando flip-flops activados por reloj o latchs depende del tipo de reloj especificado en el diseño. (La extensión .CLK indica que se trata de un flip-flop activado por reloj; la extensión :LE indica un latch). Cuando una señal de salida tipo 'REG' es procesada, la correspondiente ecuación producida por el compilador es pin a pin.

ISTYPE 'REG_D'

La salida es de registro con flip-flop tipo D activada por un reloj y será descrita usando estímulos detallados al flip-flop en las descripciones lógicas. Para describir el estímulo al flip-flop para esta salida, se puede usar la extensión .D. Cuando una señal de salida de tipo 'REG_D' es procesada en un diagrama de estado, las ecuaciones correspondientes producidas por el compilador usan una extensión .D.

ISTYPE 'REG_T'

La salida es de registro usando flip-flop tipo T activado por reloj como elemento de memoria. Las salidas tipo T pueden ser descritas usando estímulos detallados al flip-flop en las descripciones lógicas. Para describir el estímulo al flip-flop para esta salida, se puede usar la extensión .T. Cuando una señal de salida de tipo 'REG_T' es procesada en un diagrama de estado, las ecuaciones correspondientes producidas por el compilador usan una extensión .T.

ISTYPE 'REG_SR'

La salida es de tipo registro usando flip-flops tipo S-R (set-reset) como elemento de memoria. Las salidas tipo SR pueden ser descritas usando estímulos detallados al flip-flop en las descripciones lógicas. Para describir el estímulo al flip-flop para esta salida, se pueden usar las extensiones .S y .R. Cuando una señal de salida de tipo 'REG_SR' es procesada en un diagrama de estado, las ecuaciones correspondientes producidas por el compilador pueden usar las extensiones .S y .R.

ISTYPE 'REG_JK'

La salida es de tipo registro usando flip-flops tipo JK como elemento de memoria. Las salidas tipo JK pueden ser descritas usando estímulos detallados al flip-flop en las descripciones lógicas. Para describir el estímulo al flip-flop para esta salida, se pueden usar las extensiones .J y .K. Cuando una señal de salida de tipo 'REG_JK' es procesada en un diagrama de estado, las ecuaciones correspondientes producidas por el compilador pueden usar las extensiones .J y .K.

ISTYPE 'REG_G'

La salida es de tipo registro usando un flip-flop tipo D como compuerta de reloj. Las salidas tipo G pueden ser descritas usando estímulos detallados al flip-flop en las descripciones lógicas. Para describir el estímulo al flip-flop para esta salida, se pueden usar las extensiones .D y .CE (habilitación de reloj). Cuando una señal de salida de tipo 'REG_G' es procesada en un diagrama de estado, las ecuaciones correspondientes producidas por el compilador pueden usar las extensiones .D y .CE.

ISTYPE 'COM'

La salida es combinacional. Todas las salidas combinacionales son inherentemente pin a pin. Cuando se describe la lógica para una salida combinacional se pueden utilizar los operadores de asignación = o ->.

ISTYPE BUFFER

El registro de la señal de salida no será invertido para el pin de salida . Si se especifica 'buffer', esto implica que la señal no sufre ninguna inversión entre la salida del flip-flop y el pin de salida. Un atributo 'buffer' o 'invert' es usualmente requerido si se describe una salida usando estímulos detallados al flip-flop (extensiones tipo .D, .T, .J / .K o .S / .R). Si no se especifica 'buffer' o 'invert' para este tipo de salidas el compilador producirá mensajes de advertencia y los resultados del circuito dependerán del diseño que se está implementando en hardware.

ISTYPE 'INVERT'

La señal de salida es de tipo registro invertido en el pin de salida. Si se especifica 'invert' se indica que la señal sufre una inversión entre la salida del flip-flop y el pin de salida.

ISTYPE 'POS'

Las ecuaciones para esta salida serán optimizadas por el compilador para polaridad positiva. Nótese que este atributo no indica una señal activa en alto. Los atributos 'pos' y 'neg' son utilizados para controlar la forma en que el compilador produce las ecuaciones y son utilizadas durante las condiciones de no importa. Si estos atributos no son especificados el compilador asume el atributo 'pos'.

ISTYPE 'NEG'

Las ecuaciones para esta salida serán optimizadas usando polaridad negativa. Este atributo no indica una señal activa en bajo. Cuando se especifica 'neg' las condiciones de no importa toman el valor de cero .

ISTYPE 'DC'

Las ecuaciones para esta salida serán optimizadas utilizando condiciones de no importa. Las condiciones de no importa implican que las tablas de verdad o los diagramas de estado están incompletamente especificados.

ISTYPE 'XOR'

Este atributo instruye al compilador para preservar los operadores de alto nivel OR-EXCLUSIVA (si estos están disponibles) para la salida indicada. Si no hay operadores OR-EXCLUSIVA en la descripción lógica este atributo no causa ningún efecto.

CONSIDERACIONES DE INVERSIÓN DE SALIDAS

Muchos de los atributos descritos definen dependencias de implementación en el diseño. Los atributos 'pos' y 'neg' implican que una señal es bidireccional. Las señales bidireccionales son determinadas por la forma en que son usadas en las descripciones lógicas, no por la forma en que son declaradas.

DECLARACIÓN DE SEÑALES ACTIVAS EN BAJO

Las declaraciones de señales activas en bajo pueden ser usadas para señales que son expresadas usando lógica activa en bajo. Para declarar que una señal es activa en bajo, un operados **!** es colocado entes del nombre de señal en la declaración **PIN** o **NODE**, por ejemplo:

```
!Habilita PIN ISTYPE 'COM';
```

CONSTANTES NUMÉRICAS

Las constantes numéricas son usadas en **ABEL** para especificar valores enteros. Todas las operaciones que involucran números en **ABEL** son procesadas con una aritmética de 128 bits sin signo. Los 128 bits disponibles permiten utilizar números en el rango de 0 a 2^{128} -1. Los números son especificados en base 10 (decimal). Si se utiliza la directiva @RADIX el número será precedido por un prefijo modificador . Los prefijos válidos son ^b, ^o, ^d, ^h.

El prefijo ^b indica que se trata de un número binario (base 2) y solo se pueden usar los caracteres 0 y 1. El prefijo ^o indica que se trata de un número octal (base 8) y solo se pueden utilizar los caracteres 0 a 7. El prefijo ^d indica que se trata de un número decimal (base 10). Este es el prefijo que se establece como condición inicial cuando no se utiliza la directiva @RADIX. El prefijo ^h indica que se trata de un número hexadecimal (base 16). Los números especificados con este prefijo pueden utilizar los caracteres 0 a 9 y 'a, 'b, 'c, 'd, 'e y 'f.

CADENAS COMO NÚMEROS

En ecuaciones, tablas de verdad y otras expresiones lógicas, una cadena de caracteres encierra caracteres que pueden ser representados como números. En este caso, el código ASCII es usado para crear un valor numérico. Cuando más de un caracter es concatenado para formar una cadena, cada caracter representa ocho bits de datos numéricos, el bit más a la derecha representa el bit menos significativo. Ejemplo:

```
Str1 = 'a';      "Evalúa el 97 decimal (61 hexadecimal)
Str2 = 'fred';   "Evalúa el 1718773092 (66726564 hexadecimal)
```

Cada número tiene un máximo de 128 bits de tamaño, el máximo número de caracteres que pueden ser concatenados para un número es 16.

A.2 EXPRESIONES

Una expresión de **ABEL** consta de una o más elementos de expresión que pueden ser operados por uno o más operadores de expresión. Un elemento de una expresión puede ser una señal, conjunto, número o constante especial. Las propias expresiones pueden ser elementos de expresiones más largas. Las expresiones construidas con los tipos básicos son usadas por **ABEL** para representar la lógica combinacional, funciones u operaciones aritméticas resultando en valores numéricos. Las expresiones de complejidad arbitraria pueden ser usadas en diseños donde se requiere la descripción de identificadores de señal, números o constantes especiales.

SEÑALES

Las señales en **ABEL** pueden ser pines o nodos y son declaradas en la sección **DECLARATIONS** antes de ser usadas en la descripción lógica y los vectores de prueba. Las señales pueden ser agrupadas en conjuntos u operadas directamente por medio de operadores de expresión. Las reglas y el orden de operación es el mismo para los conjuntos que para los valores de un solo bit. El resultado es una expresión que incluye señales en una red combinacional. Ejemplos:

```
Y1 = A & B & ( C # D);
Y2 = (A != D );
```

EXTENSIONES DE SEÑAL

Las extensiones de señal son usadas para especificar señales secundarias asociadas con la salida. Las señales secundarias incluyen reloj, reset, habilitación de salida y entradas de datos de flip-flops. Las extensiones de señal son usadas para eliminar la ambigüedad cuando se especifican retroalimentaciones desde las salidas de registro. Las siguientes extensiones de señal son soportadas por **ABEL**:

.ACLR

Borrado asíncrono (pin a pin). esta extensión es usada para especificar borrado asíncrono (reset) para la salida asociada. Esta extensión se aplica a las salidas asociadas con un flip-flop, y cuando está activa causa que el pin de salida tenga el valor de cero. Si la salida es invertida, un valor de uno será cargado en el flip-flop para limpiar la salida a cero.

.AP

Preestablecimiento asíncrono. Esta extensión es usada para especificar preestablecimiento asíncrono (preset) para la salida asociada. Esta extensión se aplica directamente a las salidas asociadas con un flip-flop, y cuando está activa causa que el pin de salida tenga el valor de uno. Si la salida es invertida, un valor de cero será cargado en el flip-flop.

.AR

Borrado asíncrono. Esta extensión es usada para especificara borrado asíncrono (reset) para la salida asociada. Esta extensión se aplica directamente alas salidas asociadas con un flip-flop, y cuando está activa causa que el pin de salida tenga el valor de cero. Si la salida es invertida, un valor de uno será cargado en el flip-flop.

.ASET

Preestablecimiento asíncrono (pin a pin). Esta extensión es usada para especificar preestablecimiento asíncrono (preset de registro) para la salida asociada. Esta extensión se aplica a los pines de salida asociados con un flip-flop, y cuando está activa causa que el pin de salida tenga el valor de cero. Si la salida es invertida, un valor de uno será cargado en el flip-flop.

.CE

Habilitación de reloj. Esta extensión especifica una función de habilitación de reloj para un flip-flop tipo D. Una expresión escrita para una extensión .CE implica una operación AND entre los flip-flops .CE y .CLK.

.CLK

Reloj, esta extensión especifica el origen de las señales de reloj para un flip-flop asociado con una señal de salida específica. La presencia de la extensión .CLK indica que el flip-flop está asociado con la salida.

.CLR

Borrado síncrono (pin a pin). Esta extensión es usada para especificar el borrado síncrono (reset) para una salida asociada. Esta extensión se aplica a una señal de salida asociada con un flip-flop y, cuando se activa durante un flanco de subida del reloj causa que el pin de salida tenga un estado de cero. Un valor de uno es colocado en el flip-flop asociado para borrar la salida acero.

.D

Entrada D a un flip-flop tipo D o latch. Esta extensión indica el dato a ser cargado en el flip-flop durante el siguiente pulso de reloj, o dentro del nivel sensitivo del latch durante el modo transparente.

.FB

Retroalimentación de registro pin a pin. Esta extensión es usada en el lado derecho de la ecuación (el valor R) para especificar que la señal retroalimentada viene directamente del flip-flop asociado. .FB es típicamente usada en conjunto con el operador := para describir funciones de registro en términos de pin a pin.

.FC

Modo de control de flip-flop. Esta extensión es especificada para dispositivos que son usados para el control dinámico de flip-flops en el modo de control de flip-flops. Algunos dispositivos lógicos programables contienen flip-flops que pueden ser dinámicamente cambiados por flip-flops tipo D o JK. y esta extensión provee un control directo sobre esta opción. Esta extensión no es recomendable para diseños lógicos de propósito general.

.J

Entrada de datos J a un flip-flop JK. Esta extensión es uno de los dos datos requeridos para las entradas de un flip-flop JK. En **ABEL** todos los flip-flops JK son controlados por reloj, así que las extensiones .J y .K deben ser usadas conjuntamente con una extensión .CLK.

.K

Entrada de datos K para un flip-flop JK (ver .J).

.LD

Carga de registro. Esta extensión, cuando está activa, causa que el flip-flop asociado al pin de salida sea cargado.

.LE

Habilitación de latch (bajo). Esta extensión es usada en lugar de la extensión .CLK para indicar que el registro es un latch de nivel sensitivo. La extensión .LE es activa en bajo. El valor aplicado a la entrada del latch es leído en el latch cuando la señal .LE está en bajo.

.LH

Habilitación de latch (alto). Esta extensión es usada en lugar de la extensión .CLK para indicar que el registro es un latch de nivel sensitivo. La extensión .LH es activa en alto. El valor aplicado al latch es leído cuando la señal .LH está en alto.

.OE

Habilitación de salida. Esta extensión es usada para especificar una función de habilitación de salida para una señal específica. La habilitación de salida puede ser especificada para salidas combinacionales o de registro.

.PIN

Retroalimentación de pin. Esta extensión es usada para especificar una retroalimentación que puede originarse en el pin especificado. Esta extensión es utilizada para describir diseños que requieren entradas / salidas bidireccionales.

.PR

Preestablecimiento genérico. Esta extensión es usada para especificar el preestablecimiento lógico asociado con una salida. Esta extensión se aplica directamente al flip-flop asociado con la salida.

.Q

Retroalimentación directa del registro. Esta extensión es usada en el lado derecho de la ecuación (el valor R) para especificar que la señal retroalimentada viene directamente de la salida Q del flip-flop o latch asociado.

.R

Entrada del dato R a un flip-flop tipo SR (ver .S).

.RE

Borrado genérico. Esta extensión es usada para especificar el borrado de la lógica para la salida asociada. Esta extensión se aplica directamente a el flip-flop asociado con la salida y cuando está activa provoca que el flip-flop tenga un valor de cero.

.S

Entrada del dato S a un flip-flop tipo SR. Esta extensión es una de las dos entradas requeridas por un flip-flop tipo SR. En **ABEL** todos los flip-flops SR son controlados por reloj, así que las extensiones .S y .R deben ser usadas en conjunto con la extensión .CLK para describir el flip-flop.

.SET

Preestablecimiento asíncrono. Esta extensión es usada para especificar preestablecimiento asíncrono (preset) para la salida asociada. Esta extensión se aplica directamente a las salidas asociadas con un flip-flop, y cuando está activa causa que el pin de salida tenga el valor de uno. Si la salida es invertida, un valor de cero será cargado en el flip-flop.

.SP

Preestablecimiento asíncrono. Esta extensión es usada para especificar preestablecimiento asíncrono (preset) para la salida asociada. Esta extensión se aplica directamente al flip-flop, y cuando está activa causa que el pin de salida tenga el valor de uno. Si la salida es invertida, un valor de cero será cargado en el flip-flop.

.SR

Borrado síncrono. Esta extensión es usada para especificar el borrado síncrono para la salida asociada. Esta extensión se aplica directamente al flip-flop asociado con la salida y cuando está activa provoca que el flip-flop tenga un valor de cero.

.T

Entrada T a un flip-flop tipo T. Esta extensión indica el evento (permanecer o cambiar de valor) que ocurre en el flip-flop durante el siguiente pulso de reloj.

CONJUNTOS

los conjuntos son colecciones de elementos de expresión encerrados entre corchetes y separados por comas. Un conjunto puede contener a otro conjunto. Cualquier elemento de expresión puede aparecer como miembro en un conjunto, y distintos tipos de elementos pueden ser mezclados en forma de un conjunto compuesto. Ejemplos:

[q3,q2,q1,q0]	"Conjunto de cuatro bits
[q3..q0]	"Notación corta para un conjunto
[0,0,q1,q0]	"Conjunto compuesto

INDEXADO DE CONJUNTOS

Cualquier conjunto puede ser indexado para obtener un subconjunto. El indexamiento del conjunto puede ser usado para seleccionar uno o más elementos desde un conjunto para usarlos en una expresión. Cuando el valor del índice es especificado, los elementos del conjunto son nombrados de izquierda a derecha, empezando con el número cero. Ejemplos:

```
Qset   = [q7..q0 ]        "Bus de ocho bits
Qhigh  = Qset[7..4]       "Bits más significativos de Qset
Qlow   = Qset[3..0]       "Bits menos significativos de Qset
```

NÚMEROS

Los números pueden ser usados como elementos de expresiones y son utilizados con señales y conjuntos para componer expresiones largas. Cuando un número es operado en una expresión de conjuntos o señales, el número es truncado al tamaño de la señal o conjunto. Ejemplo:

```
(Sel1 & Sel2 & Sel3) == 5  "Truncado a 1 bit
```

Los números son convertidos cuando se involucran en una expresión con conjuntos. Ejemplo:

```
(Sel1, Sel2, Sel3) == 5   "3 bits 1,0,1
```

CONSTANTES ESPECIALES

Las constantes especiales pueden ser usadas en una expresión. En **ABEL** se tiene la constante especial **.X.** , condición de no importa. Ejemplo:

```
([.x.,A2,A1,A0] + [.x.,B2,B1,B0]) == [C3,C2,C1,C0]
```

OPERADORES DE EXPRESIÓN

ABEL incluye una gran variedad de operadores que pueden ser aplicados en las expresiones. Los operadores se pueden agrupar en tres categorías: Booleanos, relacionales y aritméticos.

OPERADORES BOOLEANOS

Los operadores booleanos proveen todas las operaciones básicas necesarias para describir las funciones lógicas combinacionales y son los siguientes:

!	NOT (Inversión booleana)
&	AND
#	OR
$	XOR
!$	XNOR

OPERADORES RELACIONALES

Los operadores relacionales de **ABEL** establecen funciones de comparación lógica que son fácilmente descritas, los elementos de expresión a ser comparados pueden ser señales, números, conjuntos o condiciones de no importa.

Los operadores relacionales son los siguientes:

==	igual
!=	no igual
<	menor que
>	mayor que
<=	menor o igual que
>=	mayor o igual que

OPERADORES ARITMÉTICOS

Estos operadores establecen operaciones aritméticas en elementos de expresión y son los siguientes:

-	(para un elemento) negación (complemento a dos)
-	(para dos elementos) resta
+	suma
*	multiplicación
/	división
%	módulo (resto de la división entera)
<<	correr a la izquierda
>>	correr a la derecha

PRIORIDAD DE OPERADORES

Cuando se escribe una descripción en **ABEL** es importante tener en cuenta el orden de prioridad de los operadores sobre una expresión cuando se incluye más de un operador, estos niveles de prioridad son los siguientes:

PRIORIDAD	OPERADOR	DESCRIPCIÓN
1	-	(para un elemento) negación
1	!	NOT
2	&	AND
2	>>	Corrimiento a la derecha
2	<<	Corrimiento a la izquierda
2	*	Multiplicación
2	/	División
2	%	Módulo
3	+	Suma
3	-	Resta
3	#	OR
3	$	XOR
3	!$	XOR
4	==	Igualdad
4	!=	No igual
4	>	Mayor que
4	<	Menor que
4	>=	Mayor o igual que
4	<=	Menor o igual que

A.3 ECUACIONES

En la descripción de un diseño las ecuaciones deben aparecer en la sección **EQUATIONS** o en la sección **STATE_DIAGRAM** cuando definen a un estado. Puede haber cualquier número de secciones **EQUATIONS** para una descripción de diseño.

En **ABEL**, las ecuaciones consisten de tres partes básicas: del lado izquierdo la salida, un operador de relación y del lado derecho las señales, conjuntos o constantes y los respectivos operadores. La ecuación se termina con un punto y coma. Ejemplo:

```
Salida = Entrada1 & Entrada2;
```

ECUACIONES WHEN - THEN - ELSE

Además de lo anteriormente descrito **ABEL** provee una forma alterna para expresiones de lógica condicional. Las palabras reservadas **WHEN - THEN - ELSE** pueden ser usadas para escribir ecuaciones. Ejemplo:

```
WHEN Seleccion == 0  THEN  Salida = 1;
ELSE  WHEN  Seleccion == 1 THEN  Salida = 0;
```

MÚLTIPLES ECUACIONES PARA LA MISMA SEÑAL

Cuando más de una ecuación es escrita para la misma salida , las ecuaciones pueden ser combinadas por el compilador para formas una sola ecuación. Ejemplo:

```
Salida = A1;
Salida = A2;
Salida = !A3;
```

El compilador convertirá las tres ecuaciones anteriores en la ecuación:

```
Salida = A1 # A2 # !A3;
```

ECUACIONES DE REGISTRO PIN A PIN

Los operadores de asignación de registros pin a pin de **ABEL** pueden ser usados cuando un diseño requiere salidas de registro. Cuando ocurre un nuevo pulso de reloj las salidas de registro tendrán un nuevo valor cargado. Ejemplo:

```
Q1 := A1 & A2 & A3
```

El operador := describe valores pin a pin. Cuando se describen funciones de flip-flops activados por reloj, la ecuación de salida debe ser acompañada por una ecuación de reloj como la siguiente:

```
Q1.CLK = Reloj;
```

Esta ecuación indica que la señal Q1 asociada a un flip-flop es activada por la señal de reloj Reloj.

ECUACIONES ORIENTADAS A FLIP-FLOPS

La extensión .CLK es la más utilizada para flip-flops. Se requiere una ecuación .CLK para todas las señales que están asociadas con un flip-flop. Sin embargo, una ecuación que involucra a un flip-flop también puede tener otras extensiones asociadas al tipo de flip-flop como son .D, .T, .J, .K, . S y .R.

CONDICIONES DE NO IMPORTA

ABEL soporta las especificaciones lógicas que incluyen entradas con condiciones de no importa. Para especificar estas condiciones para una lógica descrita por medio de ecuaciones se pueden utilizar los operadores de asignación alternativos para completar las condiciones para una salida. Estos operadores son los siguientes:

=	Asignación combinacional
?=	Asignación combinacional de condiciones de no importa
:=	Asignación de registro
?:=	Asignación de registro de condiciones de no importa

A.4 DIAGRAMAS DE ESTADO

Cada sección **STATE_DIAGRAM** consiste de un encabezado de diagrama de estado seguido de un cierto número de estados . Cada estado puede incluir valores para las variables de salida y una transición gracia otro u otros estados.

ENCABEZADO DE UN DIAGRAMA DE ESTADO

El encabezado de un diagrama de estado es una lista de las señales (pines o nodos) que serán usados para describir los estados de la máquina de estados. Esta lista de señales es llamada el registro de estado y cada señal en el estado representa un solo bit. Ejemplo:

```
STATE_DIAGRAM        [S0,S1,S2]
```

Si las señales fueron declaradas como un conjunto puede escribirse el nombre del conjunto en el encabezado del diagrama de estado. Ejemplo:

```
ESTADO = [Q=,Q1,Q2]:
STATE_DIAGRAM ESTADO
```

VALORES DE ESTADO

Los valores de estado son introducidos para describir el estado presente en la máquina de estados y para especificar el valor del estado siguiente para cada enunciado de transición. Los valores de estado pueden ser escritos como números binarios o decimales, también pueden ser asignados con nombres de constantes.

ENUNCIADOS DE TRANSICIÓN

Cada descripción de estado incluye dos o más transiciones posibles, cuando tenemos dos transiciones podemos utilizar el enunciado de transición **IF THEN ELSE**. Otra forma, es la transición incondicional por medio del enunciado **GOTO**, otra es la del enunciado **WHEN THEN ELSE** y otra más e s la del enunciado **CASE ENDCASE**. Ejemplos:

```
STATE 1: GOTO 2;  "Salto incondicional al estado 2
STATE 2: IF A THEN 3
            ELSE 2; "Si la variable A vale 1 pasará al estado 3, en caso contrario
                    permanecerá en el estado 2
STATE 3: CASE A: 4; "Si la variable A vale 1 pasa al estado 4
              !A: 5; "Si la variable A vale cero pasa al estado 5
         ENDCASE;
```

ECUACIONES DE SALIDA DE ESTADOS

Muchas máquinas de estado son diseñadas para manejar salidas. **ABEL** puede incluir ecuaciones para esas salidas en cada estado. Ejemplo:

```
STATE 1:    Salida = 1;
            Salida1 = A & B;
            IF ... THEN...;
```

Para escribir ecuaciones asociadas con transiciones específicas o para escribir ecuaciones para salidas de tipo registro se puede usar el enunciado **WITH**. Ejemplo:

```
STATE 1: IF (Data == 12) THEN
          Write WITH {
                              registro := REGISTRO1;
                     }
         ELSE ... ;
```

A.5 TABLAS DE VERDAD

Las tablas de verdad son usadas para describir las relaciones entre los valores observados en un conjunto de señales de entrada y sus correspondientes valores en las señales de salida. Las tablas de verdad consisten en un encabezado seguido de una o más entradas a la tabla de verdad.

ENCABEZADOS DE TABLAS DE VERDAD

Al igual que en los diagramas de estado, las tablas de verdad tienen un encabezado que define las señales de entrada y salida que serán usadas en la descripción subsecuente. Ejemplos:

```
TRUTH_TABLE ([A,B,C,D] - > [Salida1, Salida2])
TRUTH_TABLE ([A,B,C,D] := [Registro1,Registro2])
```

El primer ejemplo es para una tabla de verdad con salidas combinacionales, el segundo es para una tabla de verdad con salidas del tipo registro, la diferencia entre uno y otro tipo es el operador -> o :=.

ENTRADAS A LA TABLA DE VERDAD

Las entradas a la tabla de verdad consisten en conjuntos de valores numéricos o condiciones de no importa (constantes.X.). Ejemplos:

```
TRUTH_TABLE ([A,B,C,D] - > [Salida1, Salida2])
                [.x.. 0, 1,0] - > [0,0],
                [.x...x..,0,0] - > [1,0];
```

TRUTH_TABLE ([A,B,C,D] := [Registro1,Registro2])

[0,1,1,1,] := [1,1];

[0,1,1,0] - > [0,1];

A.6 VECTORES DE PRUEBA

Los vectores de prueba son usados para describir entrada y las salidas esperadas par dichas entradas, por medio de ellos se lleva a cabo la simulación del diseño. El formato de los vectores de prueba es similar al de la tabla de verdad: se tiene un encabezado que describe las señales de entrada y salida y las entradas a los vectores de prueba describen el valor de las entradas y las salidas.

ENCABEZADO DE VECTORES DE PRUEBA

Al igual que los encabezados de tabla de verdad, los encabezados de vectores de prueba consisten en un conjunto de señales que define las entradas y las salidas que serán usadas en las subsecuentes entradas a los vectores de prueba. Ejemplo:

TEST_VECTORS([A,B,C,D] - > [Salida1, Salida2])

ENTRADAS A LOS VECTORES DE PRUEBA

Las entradas a la sección de vectores de prueba, al igual que en las tablas de verdad, consiste de un conjunto de valores numéricos o constantes especiales. Ejemplo:

TEST_VECTORS ([A,B,C,D] - > [Salida1, Salida2])

[.x.,0,1,0] - > [0,0];

[.x.,.x.,0,0] - > [1,0];

Las constantes especiales que pueden ser utilizadas en los vectores de prueba son las siguientes:

.C. Reloj (transición bajo - alto -bajo).

.D. Señal activa en bajo.

.F. Entrada o salida flotante

.K. Reloj negativo (alto - bajo -alto)

.P. Salida de registro precargada

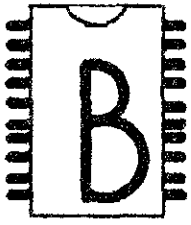
.U. Señal activa en alto

.X. Condición de no importa

.Z. Alta impedancia (Salida tri-estado)

FALTA PAGINA

No. 176



APÉNDICE B. ARREGLO LÓGICO GENÉRICO

B.1 TECNOLOGÍA DE LOS CIRCUITOS GAL

Los dispositivos GAL son fabricados usando la tecnología de compuerta flotante eléctricamente borrable, la cual fue usada inicialmente para fabricar memorias EEPROM. Esta tecnología permite grabar el dispositivo lógico varias veces. Los dispositivos GAL difieren de otros PLDs en que cuentan con Macroceldas Lógicas de Salida (OLMC), razón por la cual los GALs pueden tener salidas de distinto tipo como son: combinacional, secuencial, tri-estado y también funcionar como entradas en el mismo pin, dependiendo de la configuración que se le de a la OLMC.

Además de las OLMC los GALs más avanzados cuentan con Macroceldas de Estado Lógico (SLMC), Macroceldas Lógicas de Entrada (ILMC), y Macroceldas Lógicas de Entrada/Salida (IOLMC), con todo esto los GALs son dispositivos muy flexibles que pueden sustituir a prácticamente cualquier dispositivo PAL, e incluso ofrecer posibilidades que los PAL no ofrecen.

Los GALs tienen presentaciones de 20 y 24 pines, pueden presentarse en empaquetamiento dual o cuadrado. El voltaje nominal de alimentación es de 5 V con una tolerancia del + - 5%. La salida en nivel alto tiene un valor de 2.0 V como mínimo y hasta Vcc + 1 V como máximo y un nivel bajo de -0.5 V a 0.8 V.

B.2 ARQUITECTURA DE UN DISPOSITIVO GAL

Para analizar la arquitectura y funcionamiento interno de los GALs se analiza el ejemplo el GAL 16V8 que es el primer circuito de esta familia, los otros modelos de GAL difieren solo un poco con respecto a la descripción general que aquí se hace. Esencialmente un GAL consta de un arreglo de compuertas AND programable conectado a compuertas OR fijas, como ocurre con la arquitectura PAL. En el GAL 16V8 se tienen 8 pines de entradas fijas y 8 pines programables que pueden ser entradas o salidas. El arreglo lógico esta organizado en 16 líneas complementadas de entrada cruzando las líneas de los 64 minitérminos disponibles, con una celda programable EEPROM en cada intersección. Lo que hace un total de 2048 celdas.

Cada celda programable puede establecer una conexión entre una línea de entrada (o su complemento) y un minitérmino. Un minitérmino tiene un valor de uno lógico cuando todas las entradas conectadas a el tienen el valor de uno lógico.

En la figura B.1 los 64 minitérminos están organizados en grupos de salida, con ocho minitérminos cada una. Siete u ocho de los minitérminos en cada grupo están conectados a una compuerta OR para producir cada función lógica de salida; uno de los minitérminos puede ser utilizado para controlar la salida tri-estado.

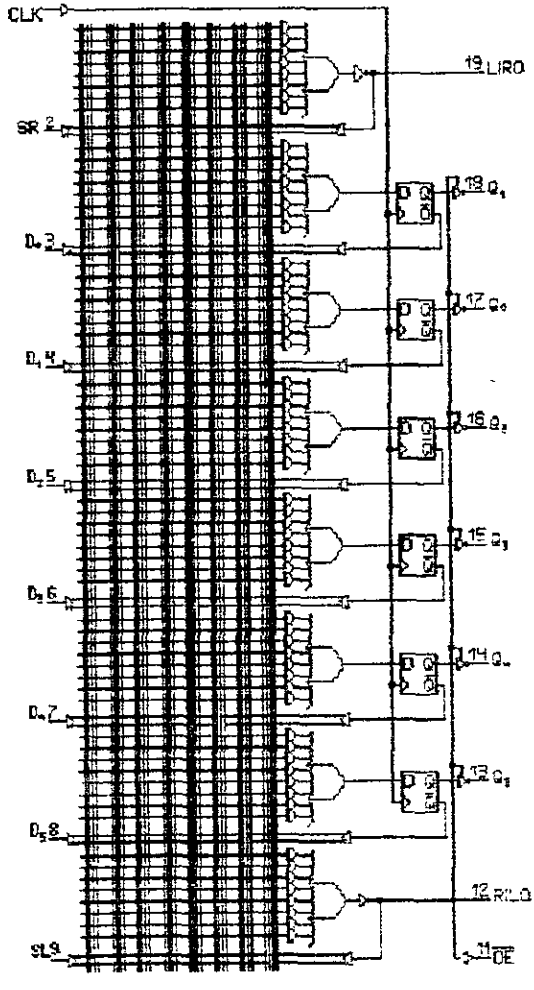


figura B.1 Diagrama lógico del GAL 16V8

La función de transferencia fundamental de cada salida en los dispositivos GAL es la suma de productos. Las herramientas de software están diseñadas para aceptar las ecuaciones booleanas y convertirlas en el patrón de programación .

En el diagrama de bloques del GAL 16V8 que se observa en la figura B.2 existen ocho funciones lógicas de salida, cada una de las funciones lógicas del arreglo AND/OR alimentan a una macrocelda lógica de salida (OLMC). Las ocho OLMCs controlan el flujo de señales de salida y entrada entre el arreglo lógico y los pines de entrada/salida del dispositivo.

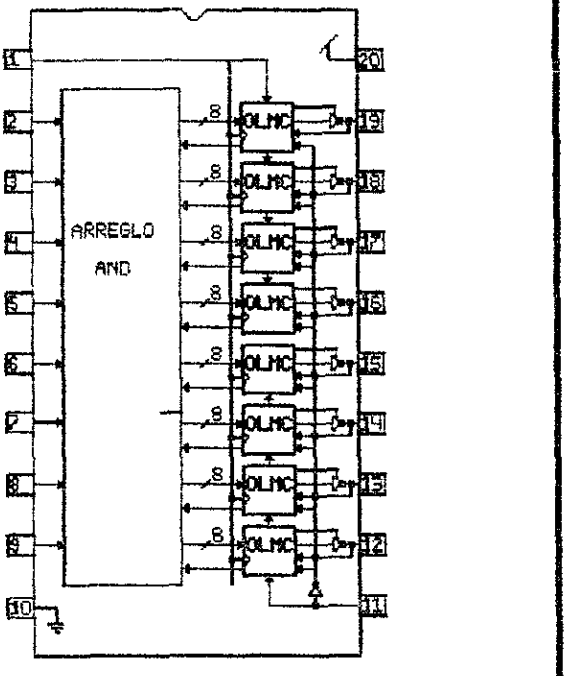


figura B.2 Diagrama a bloques del GAL 16V8

B.3 MACROCELDA LÓGICA DE SALIDA (OLMC)

En la figura B.3 se muestra el diagrama lógico correspondiente a una Macrocelda Lógica de Salida (OLMC), en el que se aprecia que recibe las salidas de ocho compuertas OR. siete de ellas representan las funciones lógicas resultantes del arreglo AND / OR y una puede servir para control o como una función mas. con este fin esta conectada al MUX1 de 2:1.

Se tiene una compuerta OR que recibe las ocho entradas. Cuando AC0 AC1 = 0 la salida del MUX1 conectada a la entrada de la OR transmitirá el valor del minitérmino correspondiente y funcionará como una entrada mas; cuando

AC0 AC1 = 1 la salida del MUX1 será 0 y el minitérmino correspondiente no afectara la salida de la compuerta OR. por lo tanto funcionará como una señal de control. La salida de la compuerta OR se conecta a una OR exclusiva con una de sus entradas conectada a una señal de control XORn (n es número de la OLMC en el GAL). Cuando XORn = 0 la salida es activa en bajo y si XORn = 1 la salida es activa en alto, según los requerimientos del diseño con lógica negativa o positiva respectivamente.

La salida de la OR exclusiva esta conectada al MUX2 de 2:1 y a un flip-flop tipo D conectado al mismo MUX2. Cuando AC0 AC1 = 0 la salida es combinacional y cuando AC0 AC1 = 1 la salida es secuencial. Al final la señal resultante se complementa por medio de una compuerta NOT y de ahí se envía directamente al pin correspondiente.

Mediante el control de la OLMC cada salida puede ser designada como combinacional o de registro. En la configuración como salida de registro, la función lógica de salida pasa por el flip-flop tipo D que se dispara en el flanco de subida de la entrada de reloj. Además, la polaridad de la función lógica de salida puede ser designada como activa en alto o activa en bajo según convenga al diseño que se esta haciendo. Las opciones de la OLMC son seleccionadas usando las señales de control. Todas las configuraciones posibles de entrada / salida del GAL 16V8 están clasificadas en tres modos básicos: modo de PAL pequeño, modo de PAL de registro y modo de PAL mediana. Estos modos corresponden a la arquitectura de las familias PAL que el dispositivo GAL puede emular. Los modos determinan las configuraciones de la OLMC que pueden ser seleccionadas para el dispositivo.

CONFIGURACIONES DE LA MACROCELDA LÓGICA DE SALIDA.

Básicamente se tienen cinco configuraciones distintas de la OLMC, estas dependen de cuatro señales de control que son: XOR, que determina si la salida será activa en bajo o activa en alto: AC0 y AC1. que determinan si el pin funcionara como una salida combinacional. salida de registro o una entrada; SYN, que habilita o deshabilita el reloj. Uno de los ocho minitérminos de entrada a la OLMC puede funcionar como señal de control para que el pin funcione como salida o entrada, también se tiene una señal de habilitación de salida G activa en bajo que funciona como la anterior. A continuación analizaremos el funcionamiento de cada uno de estos modos.

MODO 1

El estado de las señales de control es el siguiente: SYN = 1, AC0 = 0, AC1 = 1, XOR=*. La OLMC funciona como entrada dedicada; la salida de la macrocelda adyacente es retroalimentada al arreglo AND (figura B.4).

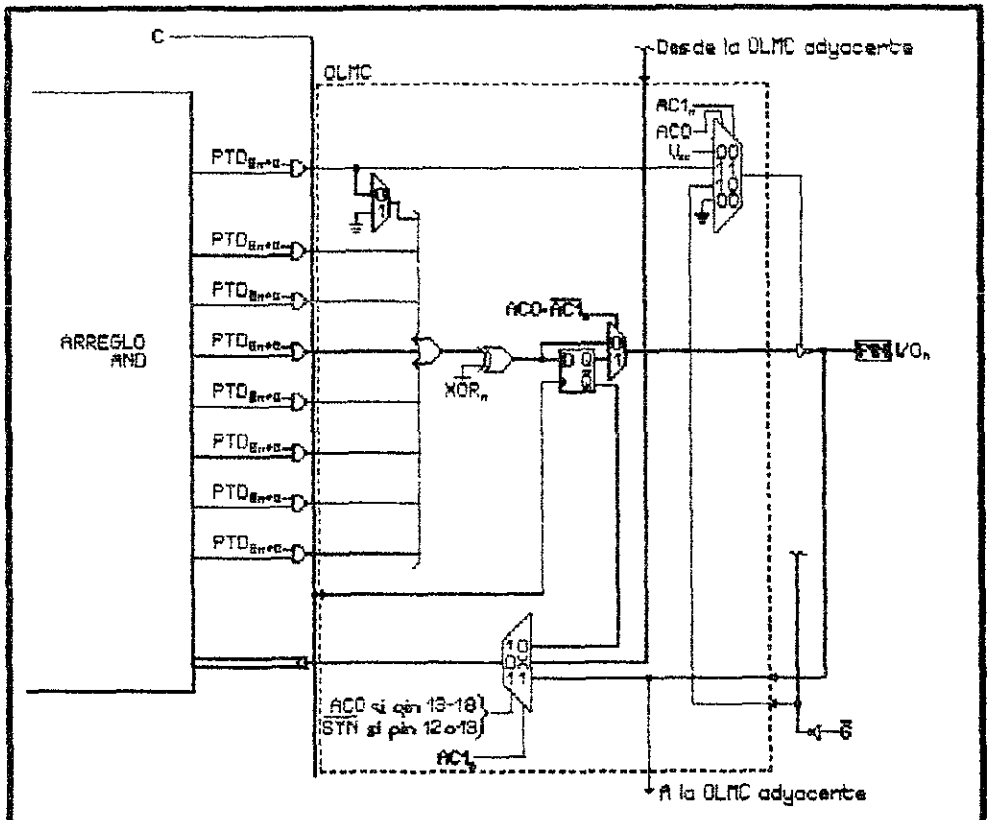


figura B.3 Macrocelda Lógica de Salida del GAL 16V8

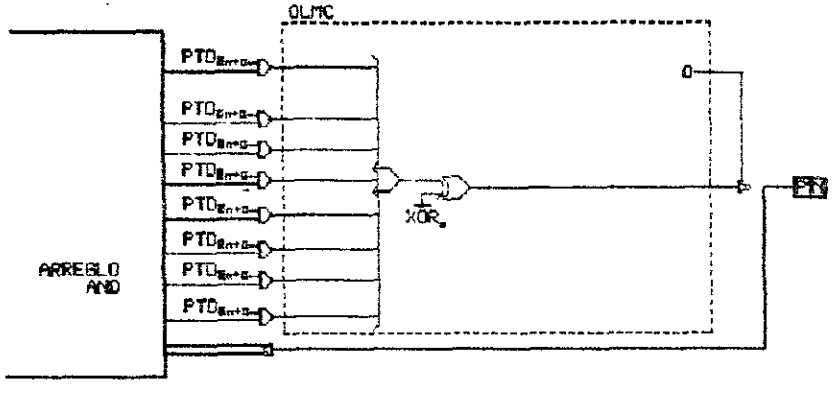


figura B.4 OLMC modo 1

MODO 2A

El estado de las señales de control es el siguiente: SYN =1, AC0= 0, AC1 = 0, XOR=0. La OLMC funciona como salida combinacional activa en bajo. La entrada habilitada como un inversor tri-estado esta conectada a Vcc. (figura B.5).

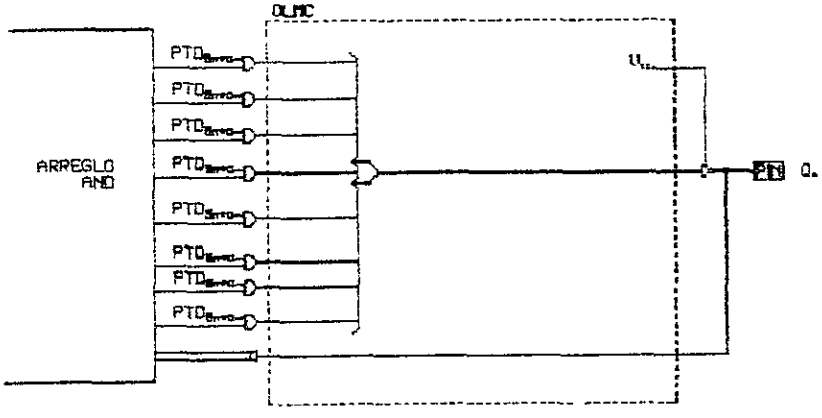


figura B.5 OLMC modo 2A

MODO 2B

El estado de las señales de control es el siguiente: SYN =1, AC0 = 0, AC1 = 0, XOR=1. La OLMC funciona como salida combinacional activa en alto. La entrada habilitada como un inversor tri-estado esta conectada a Vcc. (figura B.6).

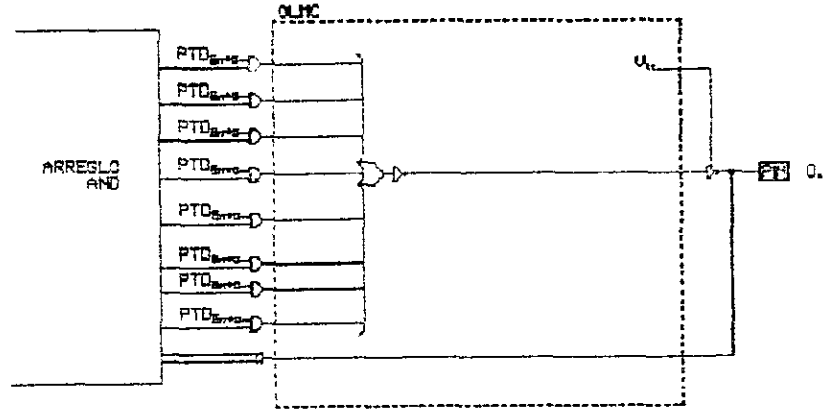


figura B.6 OLMC modo 2B

MODO 3A

El estado de las señales de control es el siguiente: SYN =1, AC0 = 1, AC1 = 1, XOR=0. La OLMC funciona como salida combinacional activa en bajo. La entrada de habilitación del inversor tri-estado es controlada por un minitérmino y la salida es retroalimentada al arreglo AND. La entrada de reloj es desconectada desde la macrocelda (figura B.7).

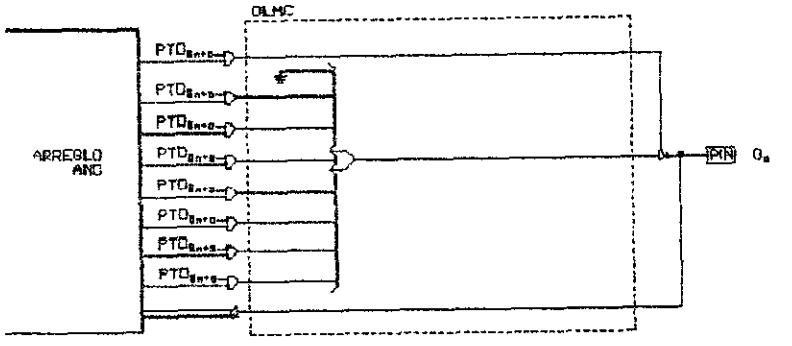


figura B.7 OLMC modo 3A

MODO 3B

El estado de las señales de control es el siguiente: SYN =1, AC0 = 1, AC1 = 1, XOR=1. La OLMC funciona como salida combinacional activa en alto. La entrada de habilitación del inversor tri-estado es controlada por un minitérmino y la salida es retroalimentada al arreglo AND. La entrada de reloj es desconectada desde la macrocelda (figura B.8).

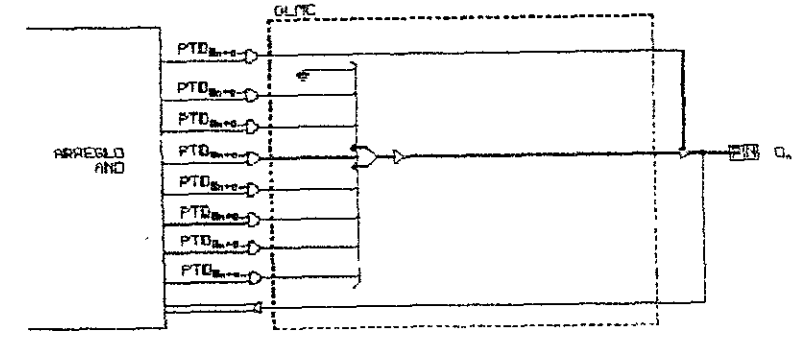


figura B.8 OLMC modo 3B

MODO 4A

El estado de las señales de control es el siguiente: SYN =0. AC0 = 1, AC1 = 1, XOR=0. La OLMC funciona como salida combinacional activa en bajo. La entrada de habilitación del inversor tri-estado es controlada por un minitérmino y la salida es retroalimentada al arreglo AND. La entrada de reloj esta conectada a la macrocelda (figura B.9).

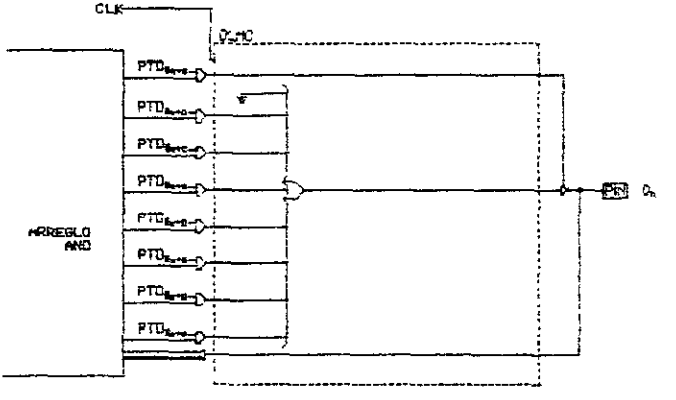


figura B.9 OLMC modo 4A

MODO 4B

El estado de las señales de control es el siguiente: SYN =0. AC0 = 1. AC1 = 1, XOR=1. La OLMC funciona como salida combinacional activa en alto. La entrada de habilitación del inversor tri-estado es controlada por un minitérmino y la salida es retroalimentada al arreglo AND. La entrada de reloj esta conectada a la macrocelda (figura B.10).

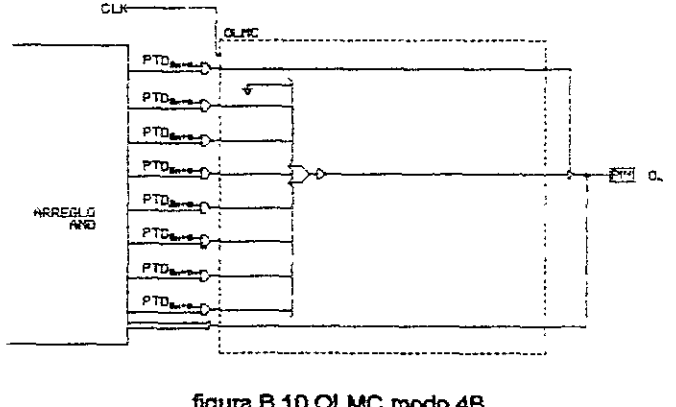


figura B.10 OLMC modo 4B

MODO 5A

El estado de las señales de control es el siguiente: SYN =0, AC0 = 1, AC1 = 0, XOR=0. La OLMC funciona como salida de registro (activa en bajo). El inversor tri-estado es activado por la línea de habilitación G (figura B.11).

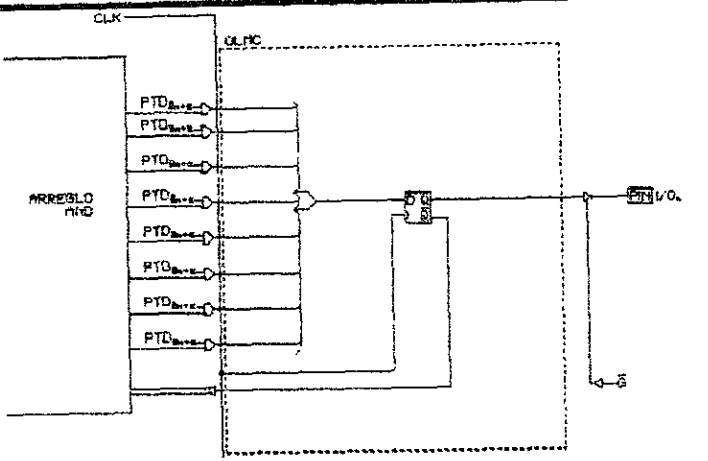


figura B.11 OLMC modo 5A

MODO 5B

El estado de las señales de control es el siguiente: SYN = 0, AC0 = 1, AC1 = 0, XOR = 1. La OLMC funciona como salida de registro (activa en alto). El inversor tri-estado es activado por la línea de habilitación G (figura B.12).

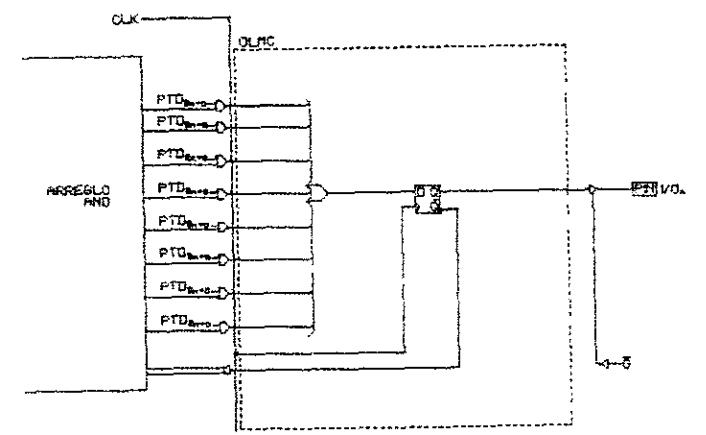


figura B.12 OLMC modo 5B

Las configuraciones posibles de la OLMC son: de salida combinacional para el modo de PAL pequeño, de salida de registro con retroalimentación para el modo de PAL de registro, de salida combinacional tri-estado para el modo de PAL mediano y de entrada para los modos de PAL pequeño y mediano.

Todos los registros en un GAL se ponen a cero lógico durante el encendido. Las salidas activas en bajo, en turno, asumen los niveles lógicos altos (si están habilitados) independientemente de la polaridad de salida seleccionada. Esto puede simplificar el diseño del circuito secuencial y su verificación. Para asegurar que al encendido los registros se reseteen el voltaje de alimentación Vcc debe subir hasta que las especificaciones de voltaje son cubiertas. Durante el encendido la entrada de reloj asume un estado lógico estable tan rápidamente como es posible (Dentro del tiempo especificado, t2PR1) para evitar interferencias con la operación de reset.

La entrada de reloj permanece estable hasta que la operación de encendido y reseteo se ha completado. En el primer pulso de reloj el flip-flop pasa al estado siguiente que propiamente le corresponde.

Como se puede notar el switcheo de cualquier entrada no conectada lógicamente a los minitérminos o las funciones lógicas no tiene efecto en el estado lógico asociado con la salida. Para reducir el consumo de energía las entradas no utilizadas pueden conectarse a tierra o a Vcc (Los circuitos GAL CMOS pueden ser conectados al voltaje de alimentación sin causar condiciones de carga excesiva).

B.4 MACROCELDA LÓGICA DE ENTRADA (ILMC)

El dispositivo GAL6001 que es el mas avanzado de la familia GAL de National Semiconductors, cuenta con otro tipo de macroceldas, además de las OLMCs. Estas macroceldas son: Macrocelda Lógica de entrada (ILMC), Macrocelda Lógica de Entrada / Salida (IOLMC) y Macrocelda Lógica de Estado (SLMC), las dos ultimas las estudiaremos posteriormente.

El GAL6001 cuenta con dos secciones de entrada configurables, del pin 2 al 11 corresponden a ILMCs. Esta sección puede ser configurada como un bloque entradas asíncronas, de latch o de registro. El pin 1, de reloj (CLK), es usado como una entrada de habilitación para los latches de las macroceldas (transparente cuando están en alto) y como reloj para las macroceldas de registro (en el flanco positivo).

Los bloques de entradas configurables representan una ventaja para el diseñador del sistema. Las entradas de registro son usadas para sincronización e intercalación de datos. Los latchs transparentes son usados cuando la entrada de datos es invalida fuera de un determinado tiempo. Las entradas directas son usadas en sistemas donde las entradas de datos están bien ordenadas en el tiempo.

Con este tipo de GAL los registros externos y latches no son necesarios. Con el GAL 6001 los registros externos y los latches no son necesarios.

Las distintas configuraciones de la ILMC son controladas programando cuatro bits de control de arquitectura, estos son: NLATCH, INSYN, IOLATCH, IOSYN. El bit SIN determina cuales de las ILMC tendrán registros, capacidad de latch o serán estrictamente asíncronas. El bit LATCH selecciona entre entradas de registro o de latch.

Las configuraciones validas de la ILMC se muestran a continuación. La tabla de verdad asociada a cada diagrama muestra los valores de los bits LATCH y SYN requeridos para cada configuración.

CONFIGURACIONES DE LA ILMC

La figura B.13 muestra el diagrama interno de una ILMC que esta constituida por dos flip-flop tipo D y un multiplexor de 4:1 que es controlado por los bits LATCH y SYN, de las cuatro combinaciones posibles solo se utilizan tres y la combinación LATCH = 0, SYN = 1 es ilegal. A continuación se analizan los tres modos de entrada que se pueden lograr con una ILMC.

MODO 1

Cuando se tiene la combinación LATCH = 1, SYN = 1 el multiplexor esta conectado directamente al pin de entrada dela ILMC por lo que se tiene una entrada directa que no depende de ningún pulso de reloj. El diagrama de esta configuración se muestra en la figura B. 14

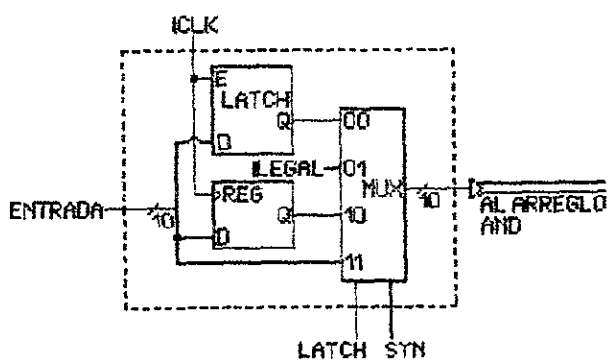


figura B.13 Diagrama interno de una ILMC.

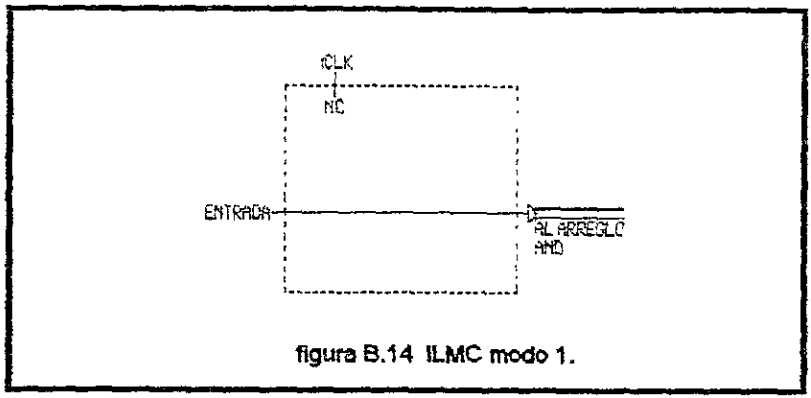


figura B.14 ILMC modo 1.

MODO 2

Cuando se tiene la combinación LATCH = 1, SYN = 0 el multiplexor esta conectado a un flip-flop , lo que hace que la entrada sea de tipo registro y que dependa de un pulso de reloj CLK. El diagrama de esta configuración se muestra en la figura B. 15

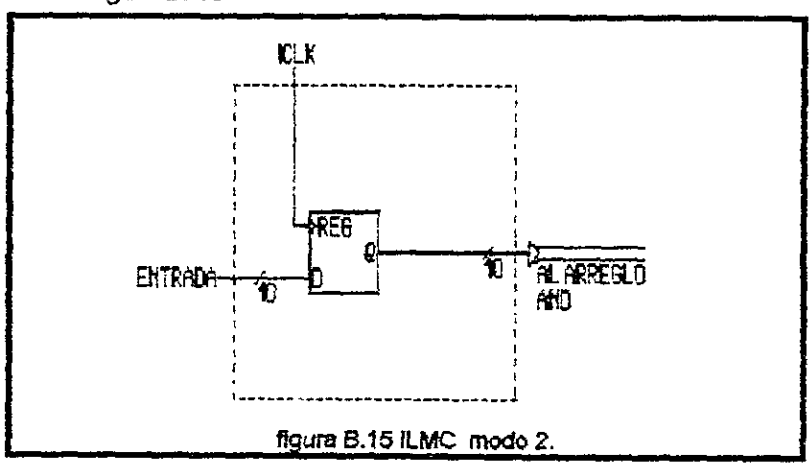


figura B.15 ILMC modo 2.

MODO 3

Cuando se tiene la combinación LATCH = 0, SYN = 0 el multiplexor esta conectado a un latch, esta entrada dependerá de un pulso de reloj. El diagrama de esta configuración se muestra en la figura B.16

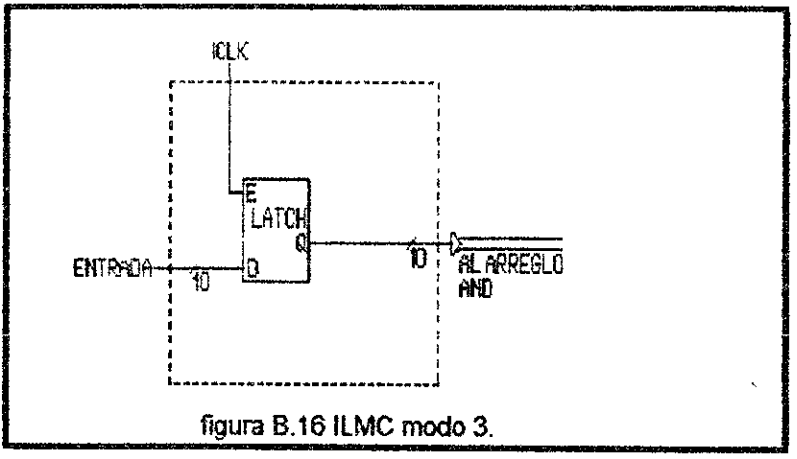


figura B.16 ILMC modo 3.

B. 5 MACROCELDA LÓGICA DE ENTRADA / SALIDA (IOLMC)

El GAL6001 cuenta con dos secciones macroceldas que se pueden configurar como entradas o salida. Las IOLMC se encuentran en los pines 14-23. Esta sección puede ser configurada como un bloque entradas asíncronas, de latch o de registro. El pin 1, de reloj (CLK), es usado como una entrada de habilitación para los latches de las macroceldas (transparente cuando están en alto) y como reloj para las macroceldas de registro (en el flanco positivo).

Las distintas configuraciones de la IOLMC son controladas programando cuatro bits de control de arquitectura, estos son: INLATCH, INSYN, IOLATCH, IOSYN. El bit SIN determina cuales de las ILMC tendrán registros, capacidad de latch o serán estrictamente asíncronas. El bit LATCH selecciona entre entradas de registro o de latch.

Las configuraciones validas de la ILMC se muestran a continuación. La tabla de verdad asociada a cada diagrama muestra los valores de los bits LATCH y SYN requeridos para cada configuración.

CONFIGURACIONES DE LA ILMC

La figura B.17 muestra el diagrama interno de una IOLMC que esta constituida por dos flip-flop tipo D y un multiplexor de 4:1 que es controlado por los bits LATCH y SYN, de las cuatro combinaciones posibles solo se utilizan tres y la combinación 01 es ilegal.

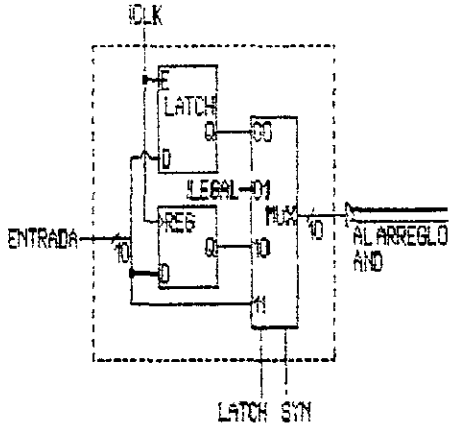


figura B.17 Diagrama interno de una IOLMC.

MODO 1

Cuando se tiene la combinación LATCH = 1, SYN = 1 el multiplexor esta conectado directamente al pin de entrada dela ILMC por lo que se tiene una entrada directa que no depende de ningún pulso de reloj. El diagrama de esta configuración se muestra en la figura B.18

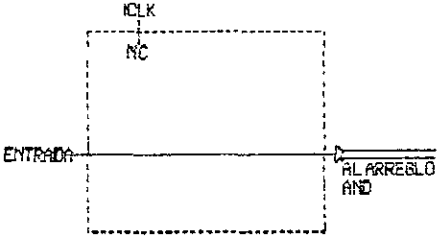


figura B.18 ILMC modo 1.

MODO 2

Cuando se tiene la combinación LATCH = 1, SYN = 0 el multiplexor esta conectado a un flip-flop , lo que hace que la entrada sea de tipo registro y que dependa de un pulso de reloj CLK. El diagrama de esta configuración se muestra en la figura B.19

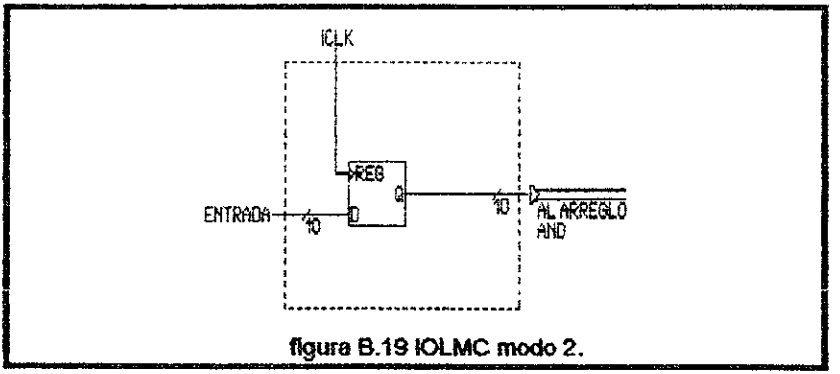


figura B.19 IOLMC modo 2.

MODO 3

Cuando se tiene la combinación LATCH = 0, SYN = 0 el multiplexor esta conectado a un latch, esta entrada dependerá de un pulso de reloj. El diagrama de esta configuración se muestra en la figura B.20.

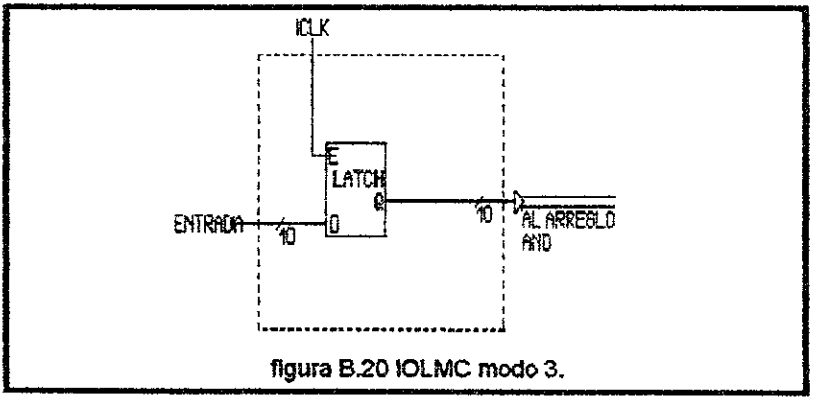


figura B.20 IOLMC modo 3.

B.6 MACROCELDA LÓGICA DE ESTADO (SLMC)

Las salidas del arreglo OR alimentan a dos tipos de macroceldas, las OLMCs que ya estudiamos y macroceldas cuyas salidas retroalimentan directamente al arreglo AND . Estas celdas son llamadas Macroceldas Lógica de Estado (SLMC), porque son usadas para construir las maquinas de estado. Las SLMC son programadas por medio de una palabra de control formada por los bits CKS, OUTSYN activa en bajo y XORE. Pero, a diferencia de las macroceldas de entrada que pueden ser configuradas cada una por separado. Las SLMC son configuradas por una macrocelda base.

El diagrama interno de una SLMC se muestra en la figura B.21, donde podemos apreciar que esta formada por una compuerta or-exclusiva controlada por el bit XORE, dos multiplexores de 2:1 controlados por el bit CKS, lo que permite determinar si el flip-flop D que se tiene será disparado por una señal asincrona E o por un pulso de reloj OCLK, se tiene además un buffer inversor controlado por la señal OUTSYN activa en bajo que determina si la salida de la SLMC será habilitada o deshabilitada. Por ultimo se tienen buffers de retroalimentación de la salida de la SLMC afirmada o negada a el arreglo AND, así como a las OLMC o a las IOLMC.

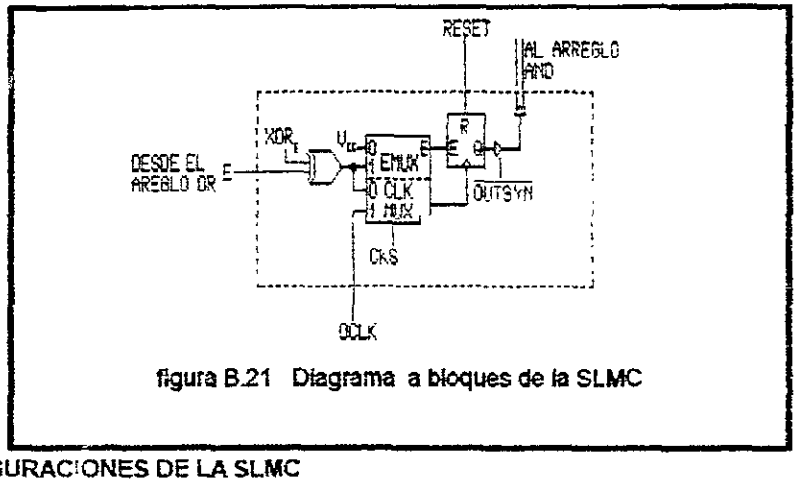


figura B.21 Diagrama a bloques de la SLMC

CONFIGURACIONES DE LA SLMC

MODO 1

Cuando las señales de control CKS = 0, OUTSYN=0 y XORE = 0 la salida será de registro asincrono, la entrada del flip-flop toma el valor de 1 y se dispara cuando la señal de entrada E tome el valor de 1, el flip-flop permanecerá inactivo mientras E tenga el valor de 0. Esta configuración es usada en diseños donde se requiera de un reloj asincrono programable, el diagrama se muestra en la figura B.22

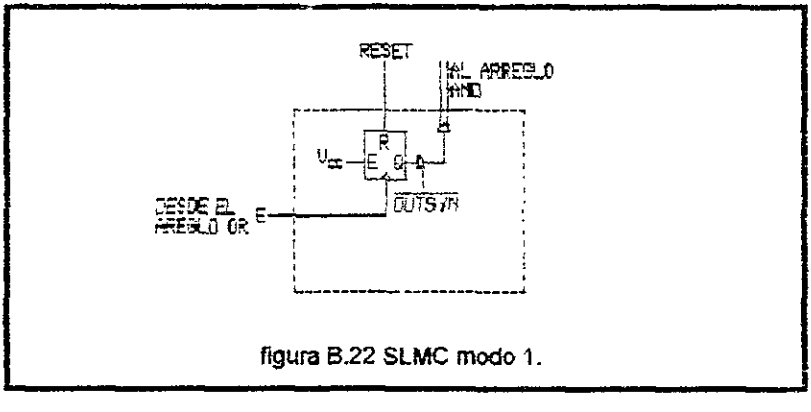


figura B.22 SLMC modo 1.

MODO 2

Cuando las señales de control CKS = 0, OUTSYN=0 y XORE = 1 la salida será de registro asíncrono, la entrada del flip-flop toma el valor de 1 y se dispara cuando la señal de entrada tome el valor de 0, el flip-flop permanecerá inactivo mientras E tenga el valor de 1. Esta configuración se utiliza en diseños donde se requiere de un reloj asíncrono programable. el diagrama se muestra en la figura B.23

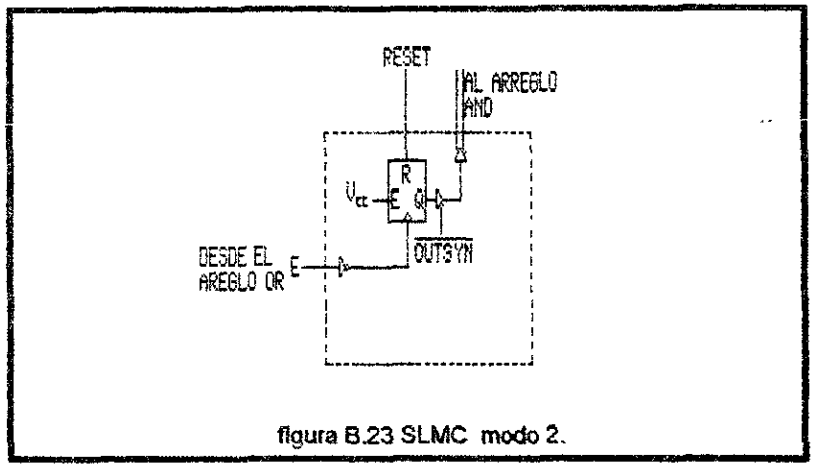


figura B.23 SLMC modo 2.

MODO 3

Cuando las señales de control CKS = 1, OUTSYN = 0, XORE = 0 y la salida de la SLMC será de registro síncrono. La entrada al flip-flop será la entrada E afirmada y el reloj será la señal OCLK. Esta configuración es utilizada para construir contadores y maquinas de estado, el diagrama se muestra en la figura B.24

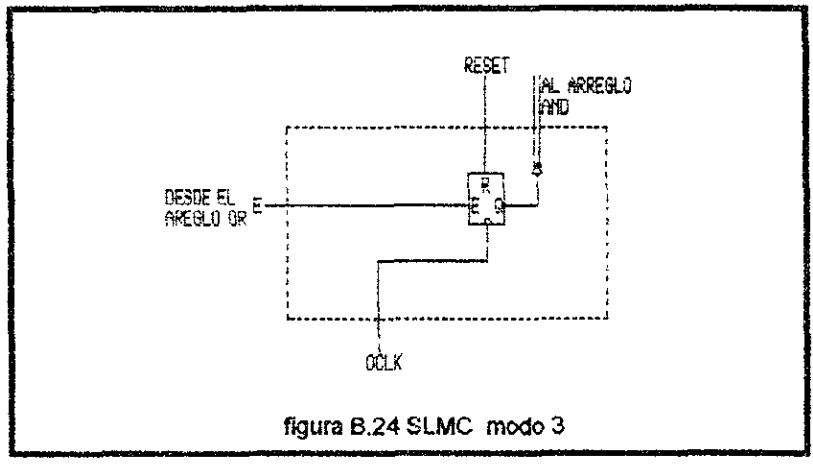


figura B.24 SLMC modo 3

MODO 4

Cuando las señales de control CKS = 1, OUTSYN=0, XORE = 1 la salida de la SLMC será de registro síncrono . La entrada al flip-flop será la entrada E negada, el reloj será la señal OCLK. Esta configuración es utilizada para construir contadores y maquinas de estado, el diagrama se muestra en la figura B.25

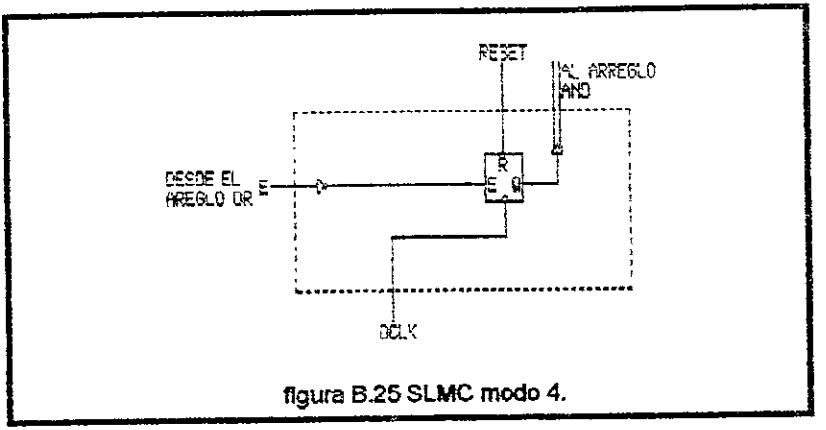
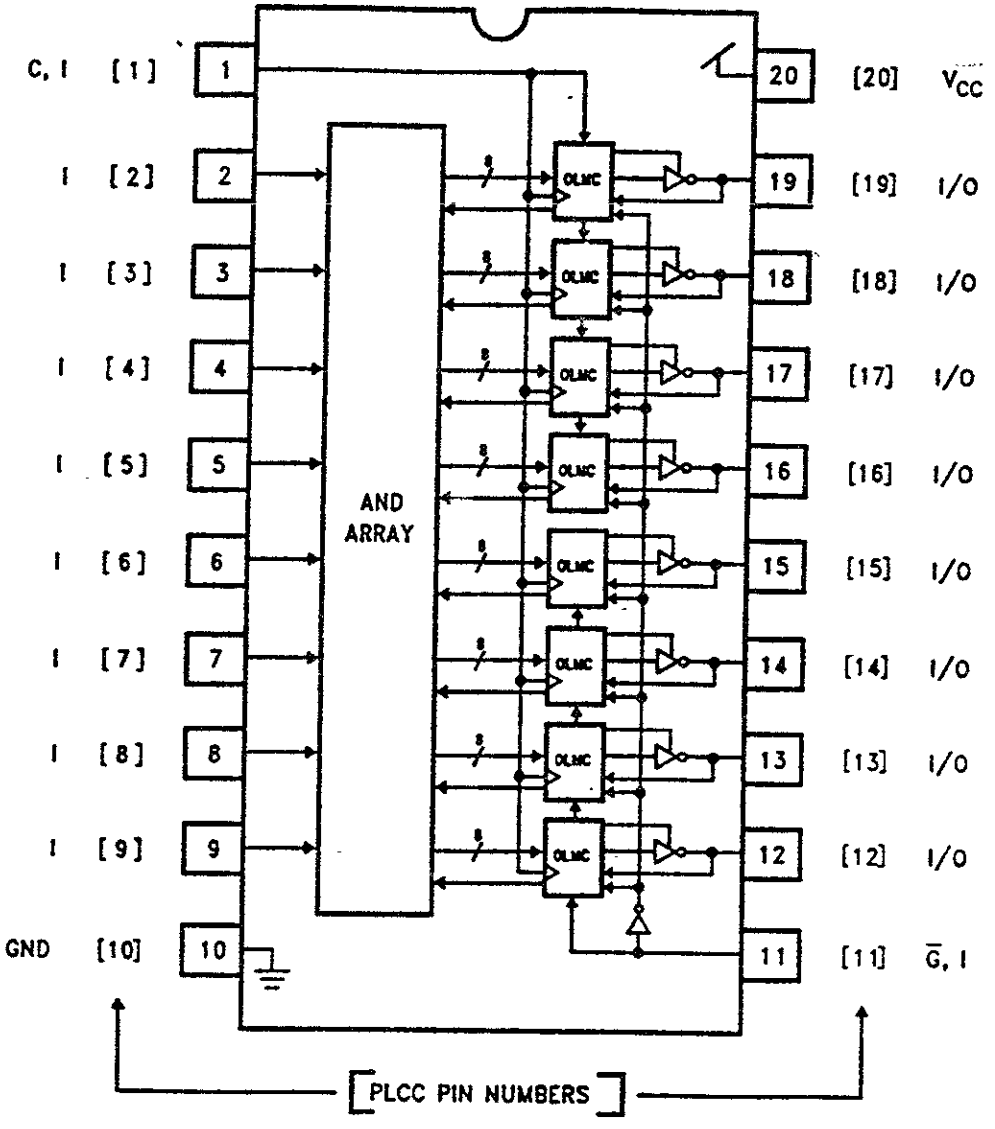
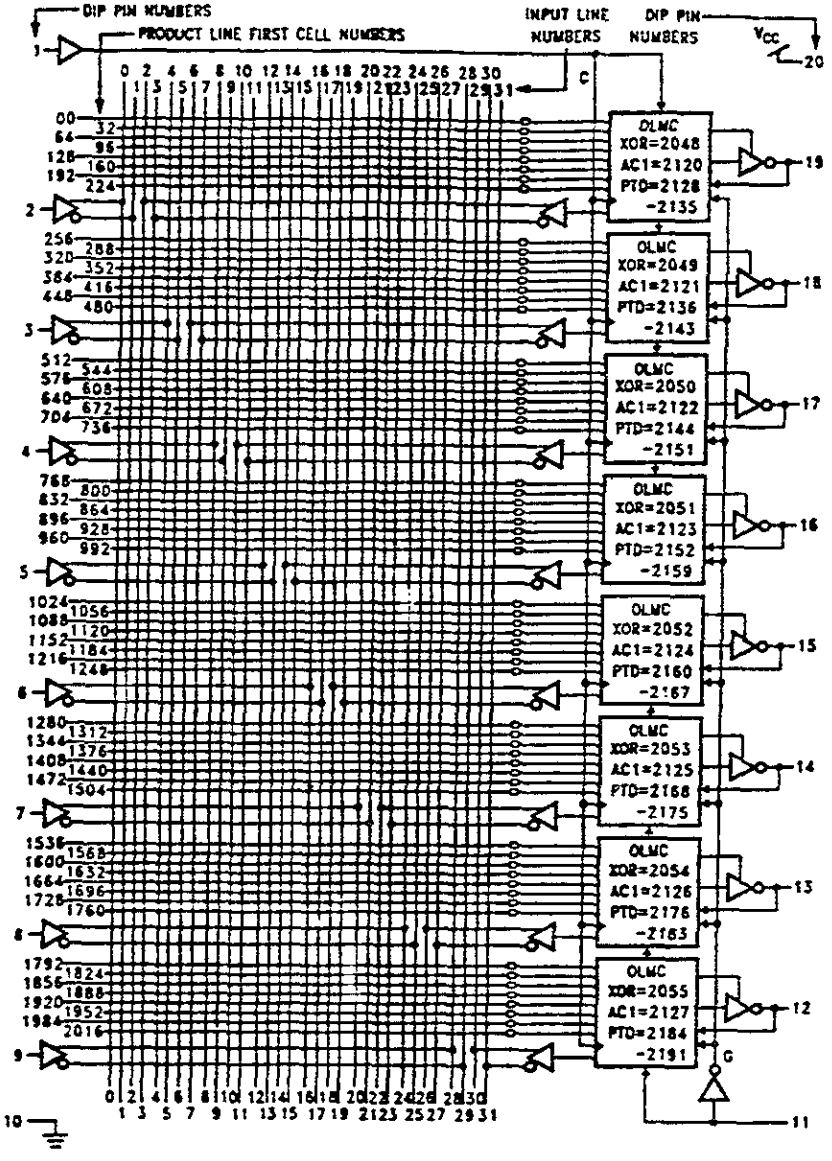


figura B.25 SLMC modo 4.

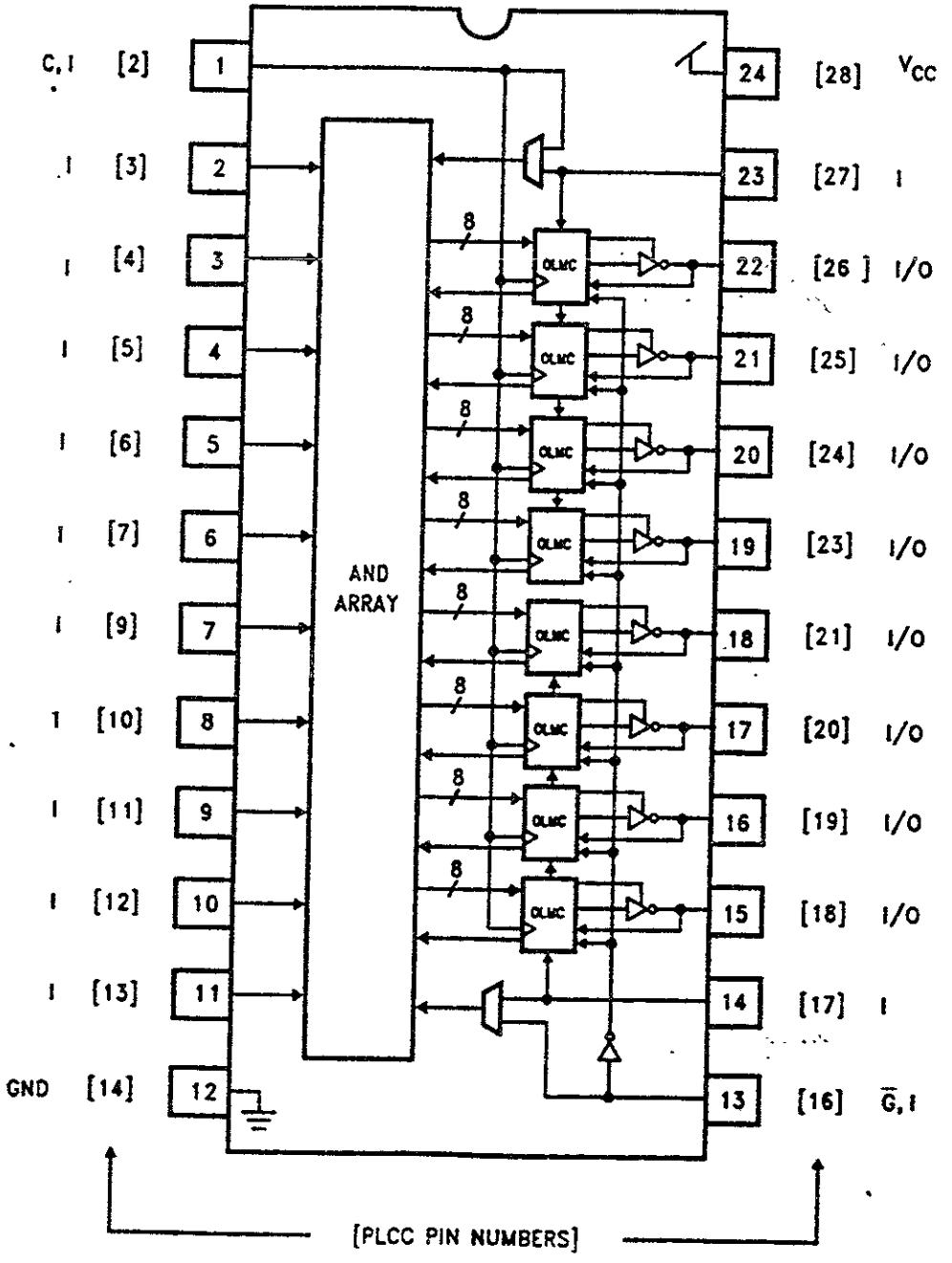
GAL16V8 Block Diagram—DIP Connections



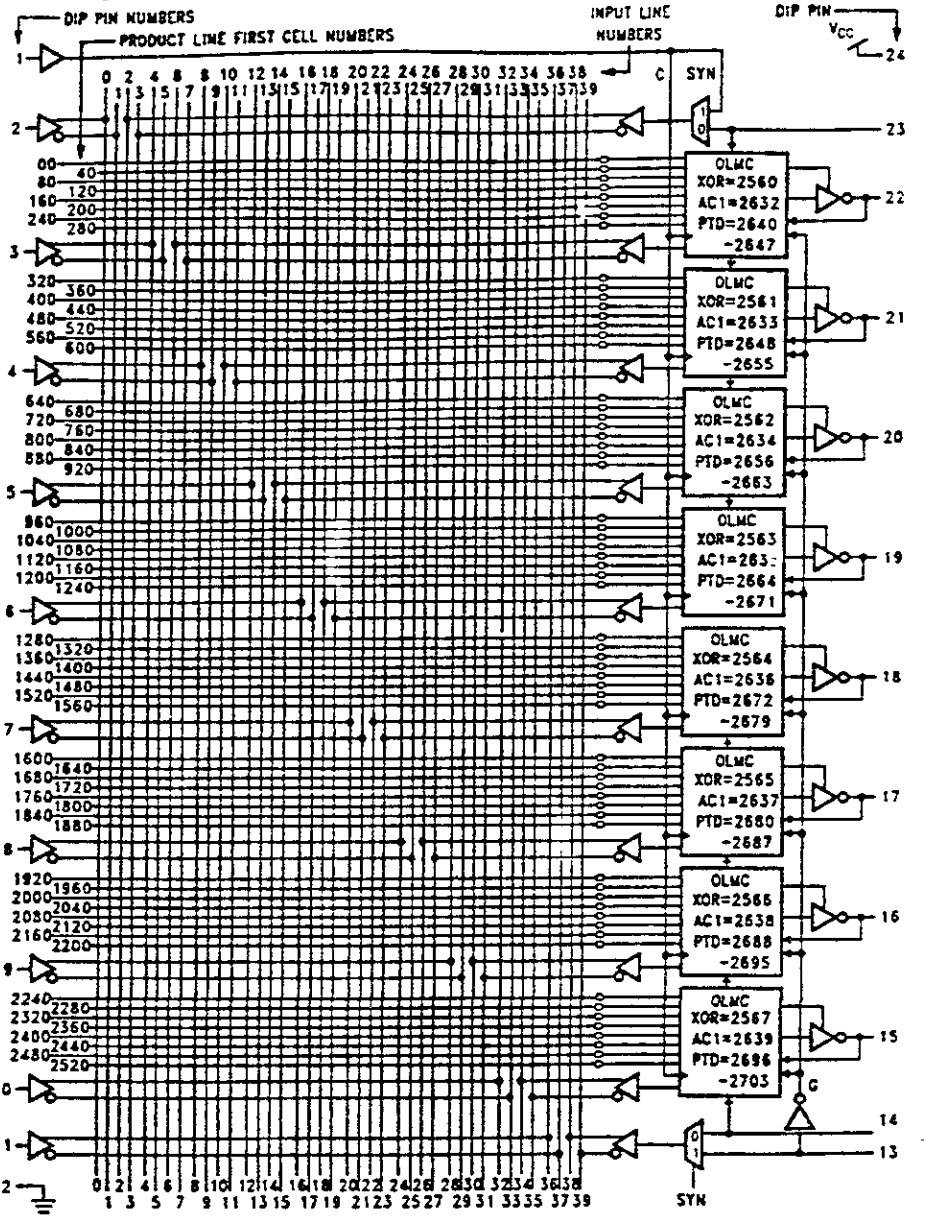
L16V8 Logic Diagram



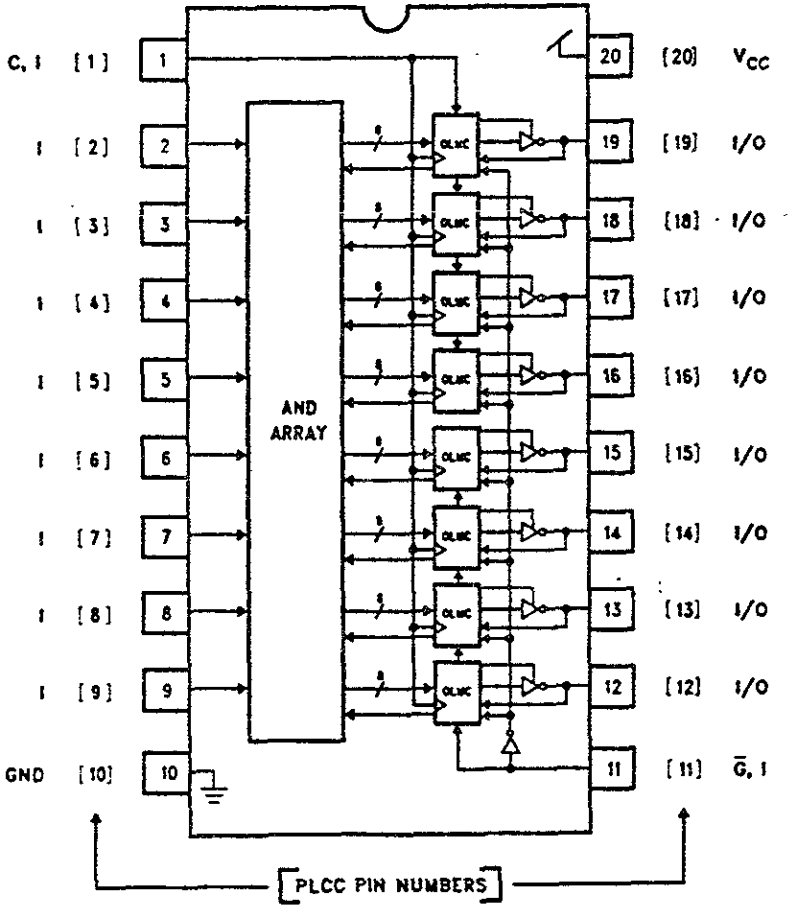
GAL20V8 Block Diagram—DIP Connections



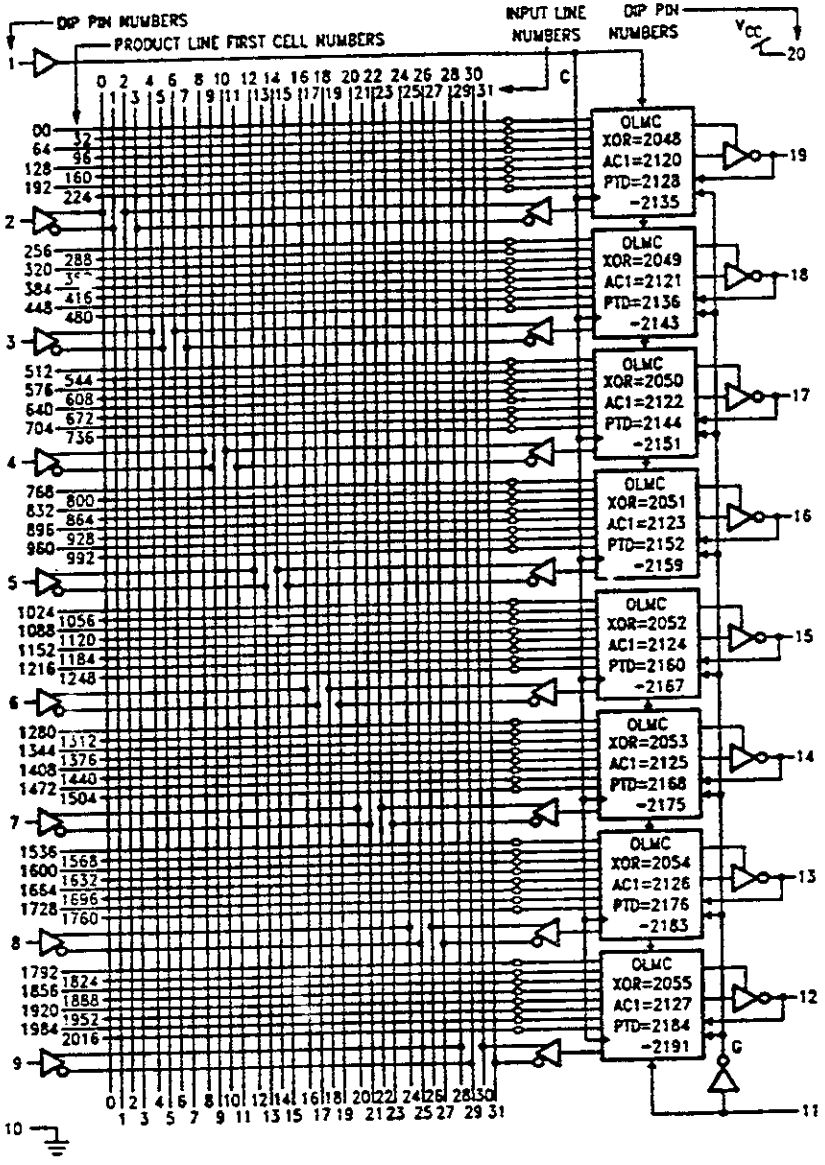
GAL20V8 Logic Diagram



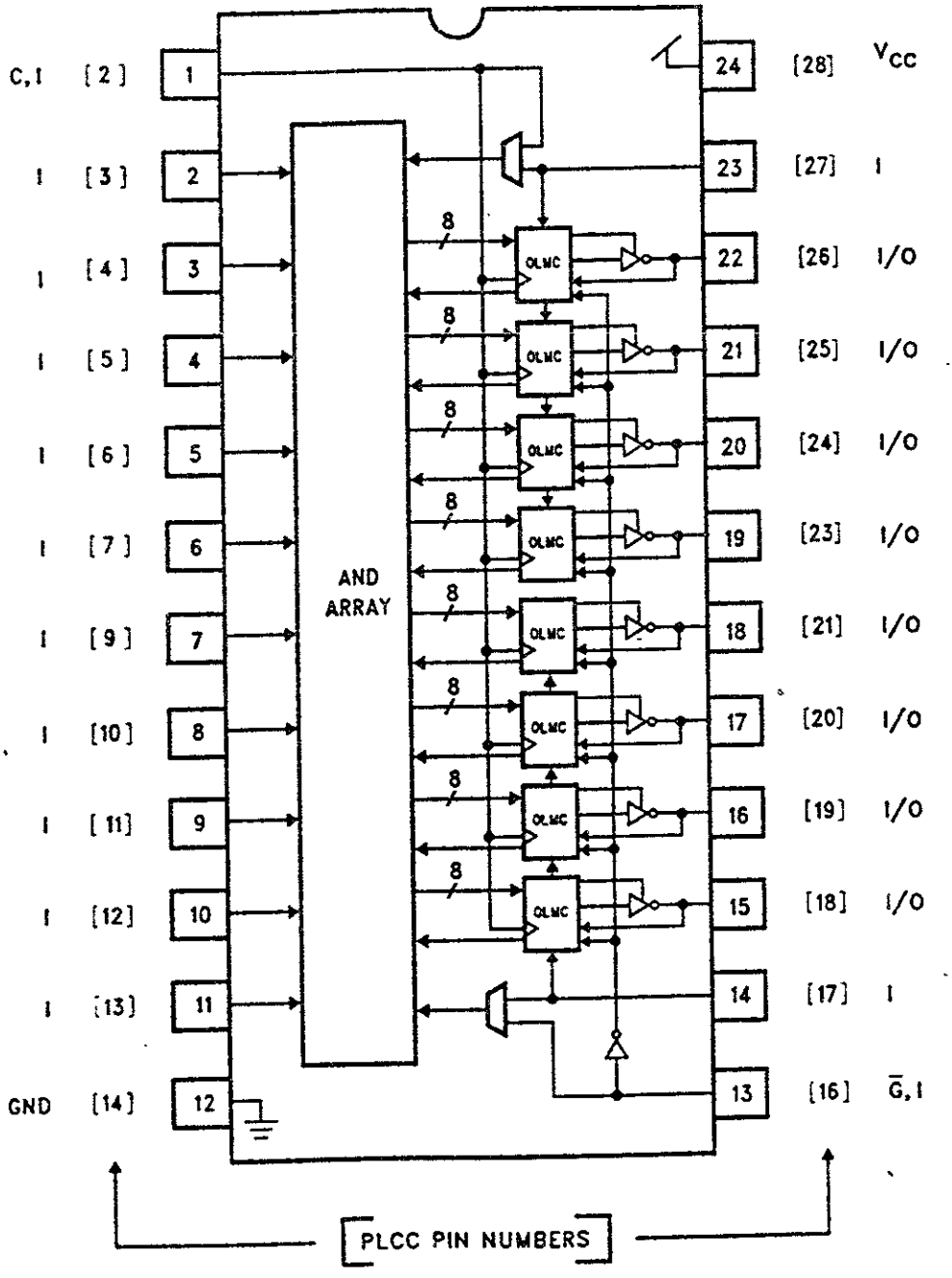
GAL16V8QS Block Diagram—DIP Connections



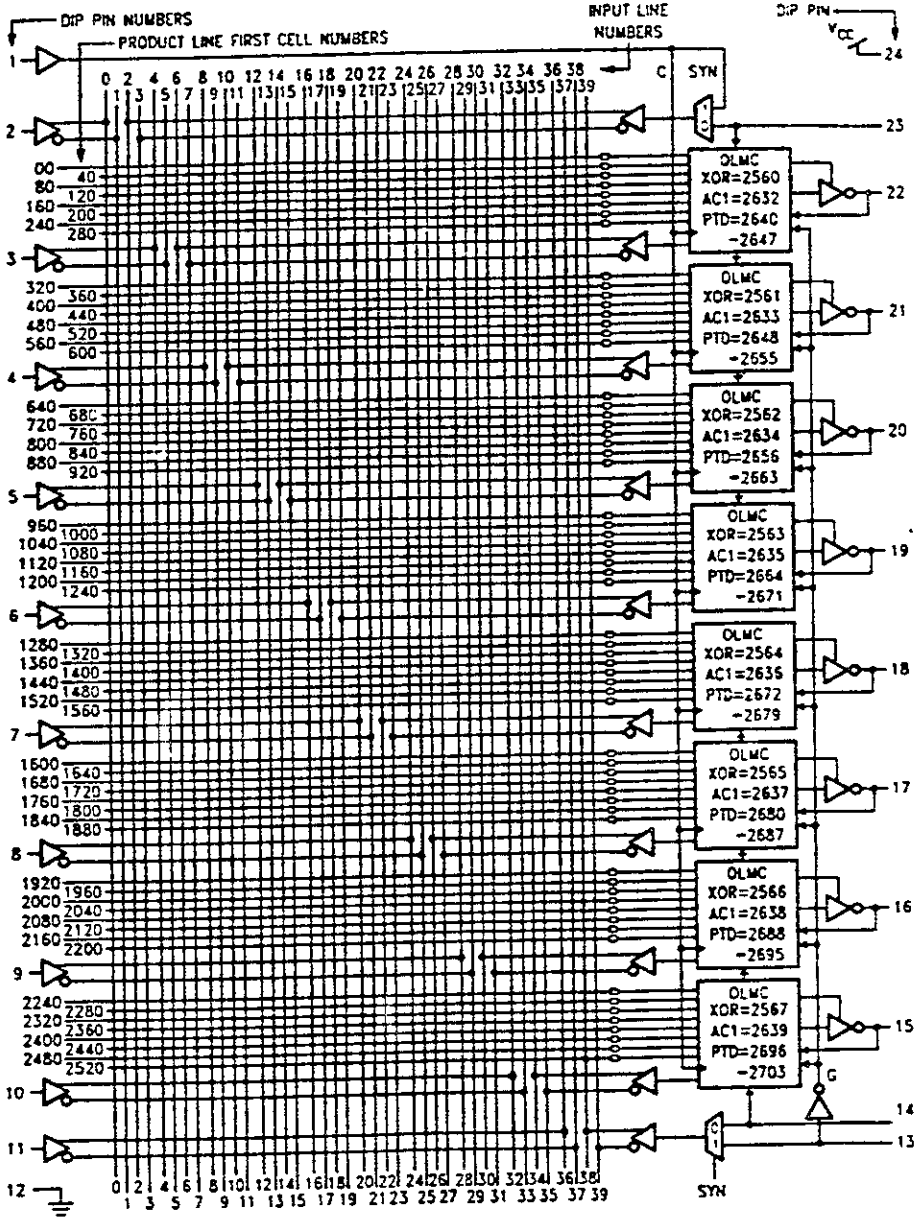
GAL16V8QS Logic Diagram



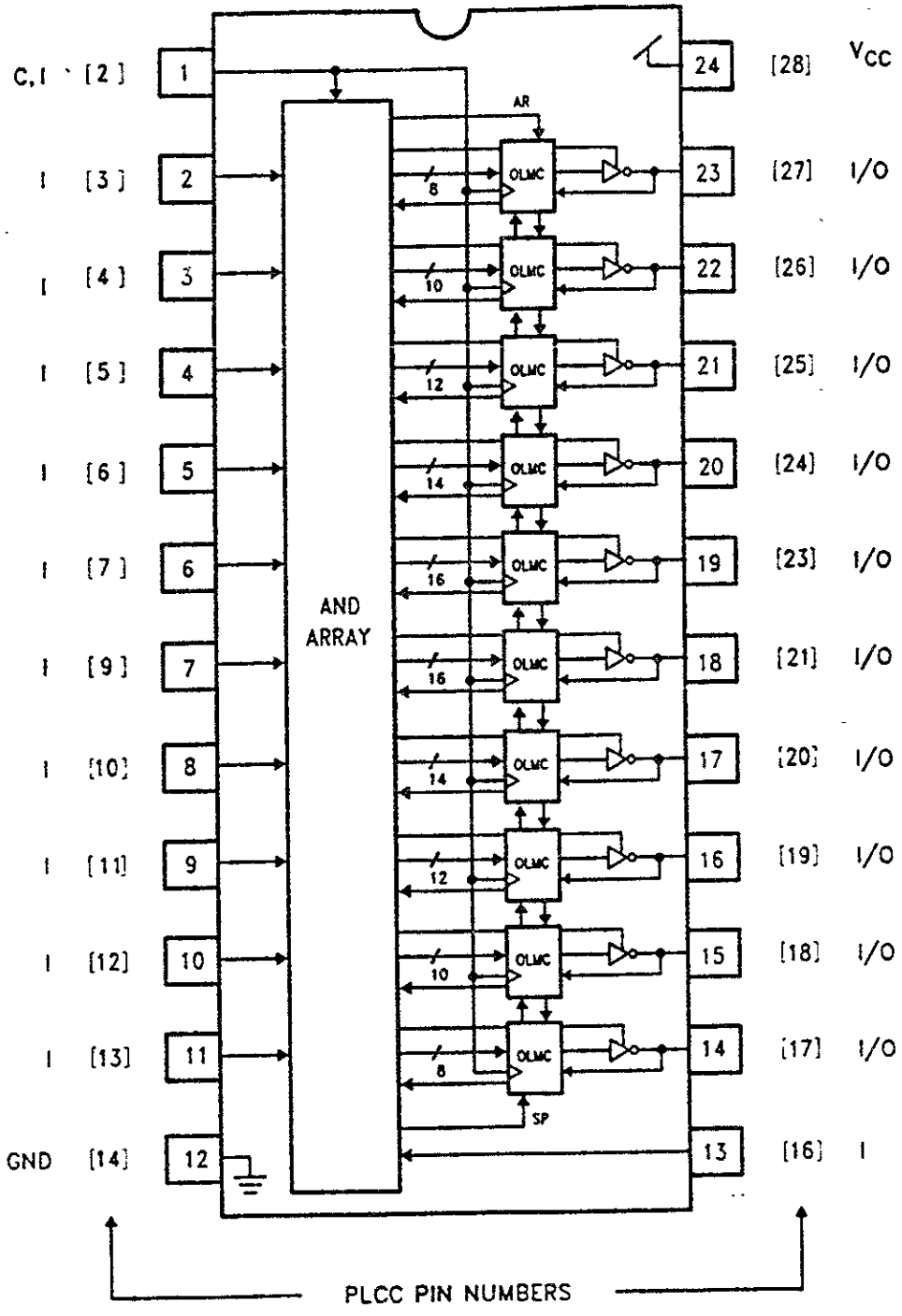
GAL20V8QS Block Diagram—DIP Connections



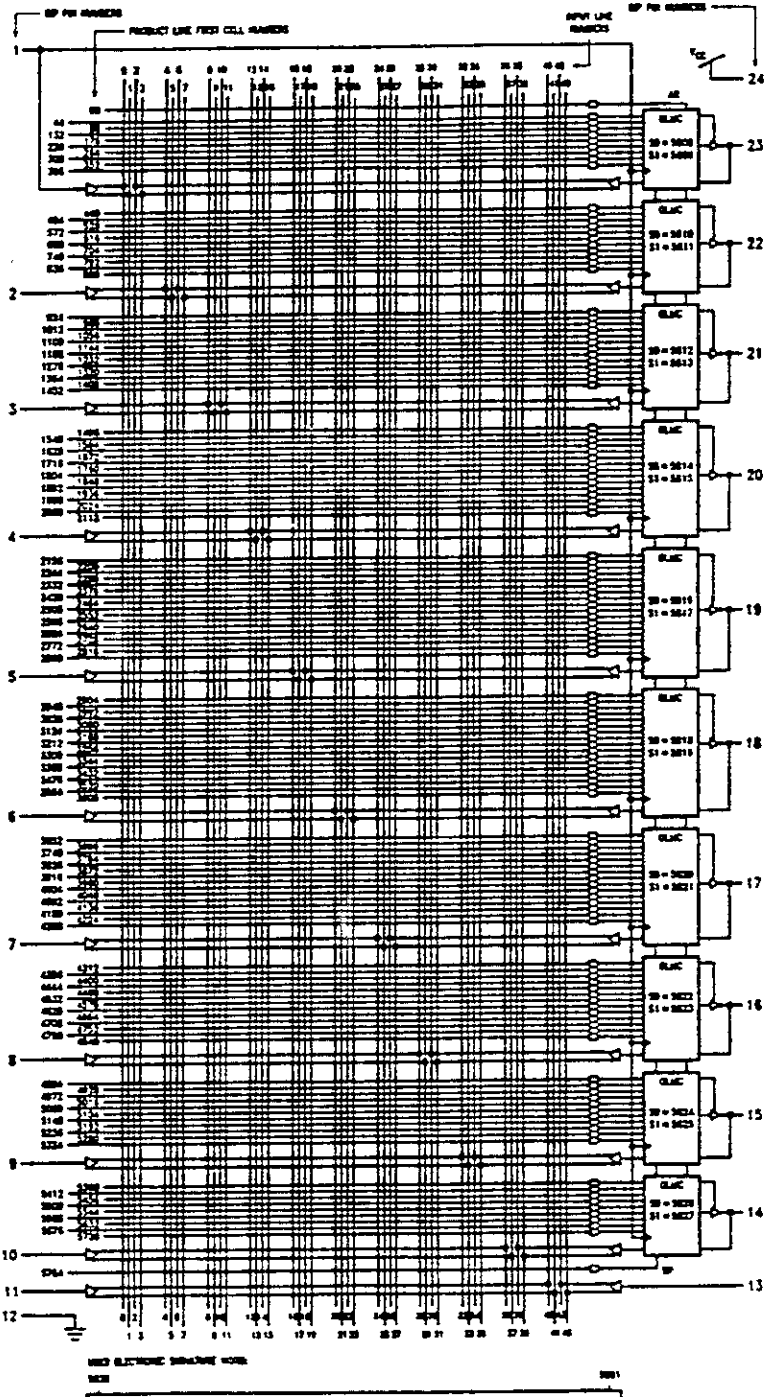
GAL20V8QS Logic Diagram



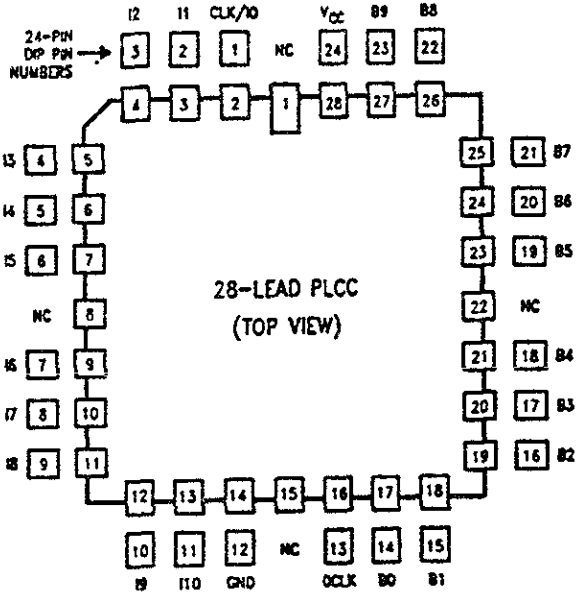
GAL22V10 Block Diagram—DIP Connections



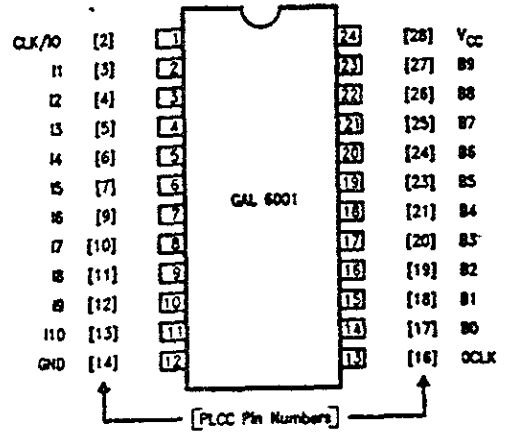
GAL22V10 Logic Diagram



28-Lead PLCC Connection Diagram

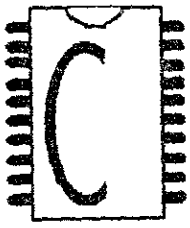


GAL Block Diagram—DIP Connections



FALTA PAGINA

No. 206



APÉNDICE C. MACH 215

C.1 ARQUITECTURA DEL MACH 215

El dispositivo MACH 215 contiene cuatro bloques PAL que están interconectados por una matriz de switch (ver figura C.1). En cada bloque PAL se tienen 8 macroceldas. Este dispositivo también cuenta con cuatro entradas dedicadas y dos entradas de reloj que van directamente a la matriz de switchs . Las entradas de reloj pueden ser utilizadas como entradas sencillas. La función de la matriz de switchs es la de tomar las salidas de las macroceldas y las conexiones de las entradas dedicadas y los relojes para proveer una ruta de acceso de las entradas a los bloques PAL.

RECURSOS DEL DISPOSITIVO

La tabla C-1 muestra los recursos del MACH 215

TABLA C-1	
RECURSO	CANTIDAD
Pines	44
Pines I/O	32
Pines de reloj	2
Pines de estrada	4
Macroceldas	32
Bloques PAL	4
Entradas a los bloques PAL	22

MACROCELDA DEL MACH 215

La macrocelda del dispositivo MACH 215 esta formada por tres componentes: el generador lógico, la celda lógica y la celda de entrada / salida. La figura C.2 muestra como están conectados estos componentes entre ellos y a la matriz de switchs en un bloque PAL.

GENERADOR LÓGICO

La función del generador lógico es la de implementar las ecuaciones combinacionales lógicas para una macrocelda. El generador lógico tiene cuatro miniterminos que pueden ser conectados a una compuerta OR local. Un generador lógico puede implementar una función de doce minitérminos (figura C.3). Las

macroceldas que se encuentran en los extremos de cada bloque PAL solamente pueden generar ocho miniterminos porque les falta una macrocelda adyacente.

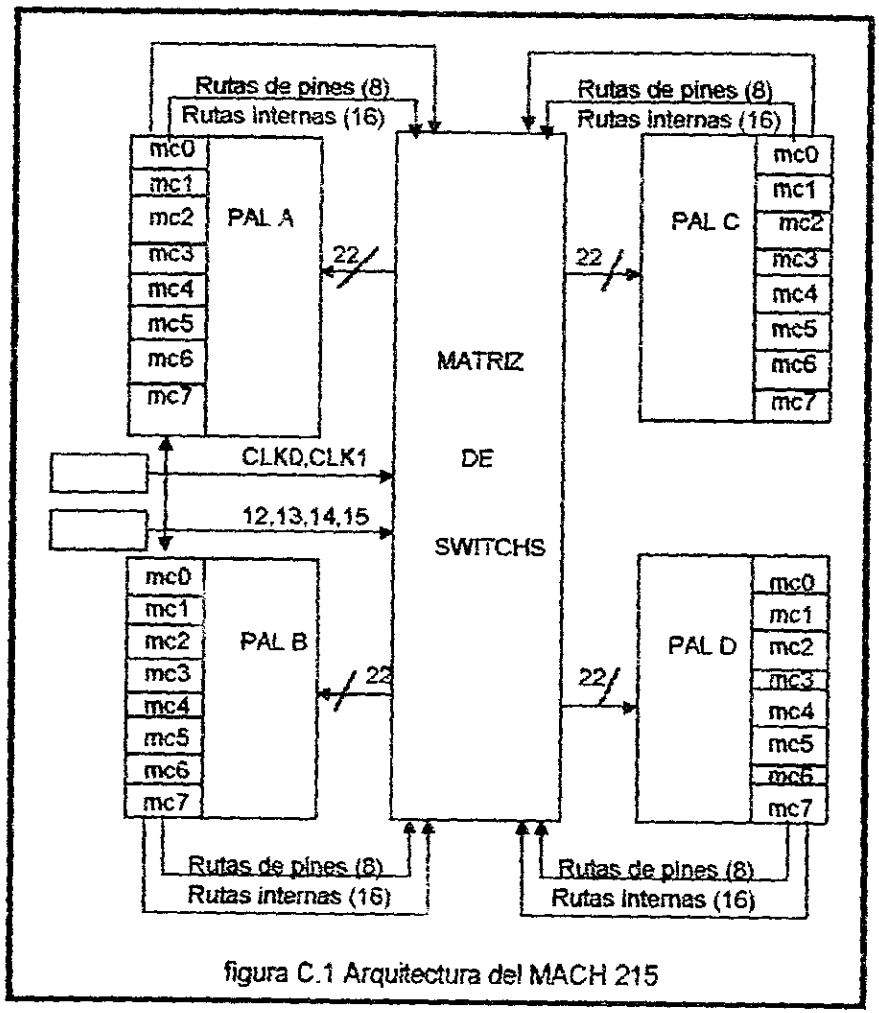


figura C.1 Arquitectura del MACH 215

CELDA LÓGICA

La celda lógica puede ser usada para registro de datos desde las funciones del generador lógico y pasa la información la celda de entrada/salida (figura C.4). La celda lógica puede ser configurada como un flip-flop tipo D, flip-flop tipo T, latch activo en bajo, o buffer combinacional.

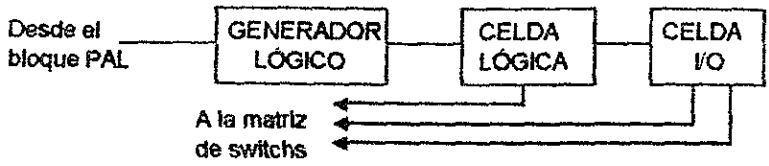


figura C.2 Diagrama a bloques de una macrocelda de salida de un MACH 215

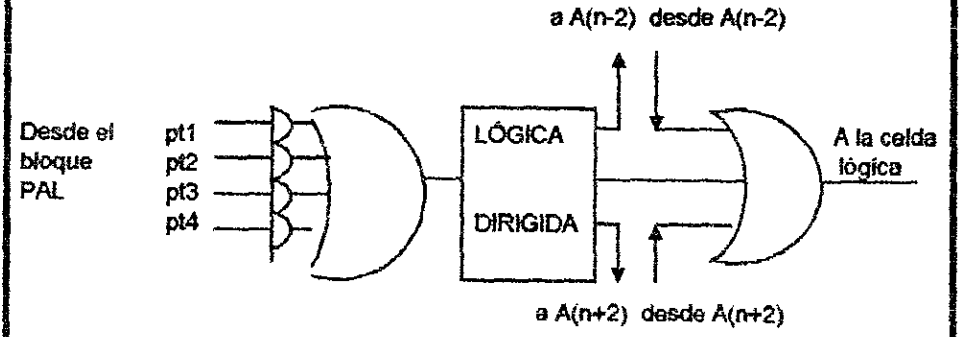


figura C.3 Generador lógico

CELDA DE ENTRADA / SALIDA

La figura C.5 muestra el tercer componente de la macrocelda, la celda de entrada/salida. La celda de entrada/salida consiste de un pin de entrada/salida, el cual está conectado a la matriz de switchs y al pin de registro. El pin de registro puede ser configurado como un flip-flop tipo D o un latch activo en bajo. La señal de reloj para el pin de registro puede venir de alguno de los dos relojes globales (CLK0 o CLK1) y puede ser usado con polaridad positiva o negativa

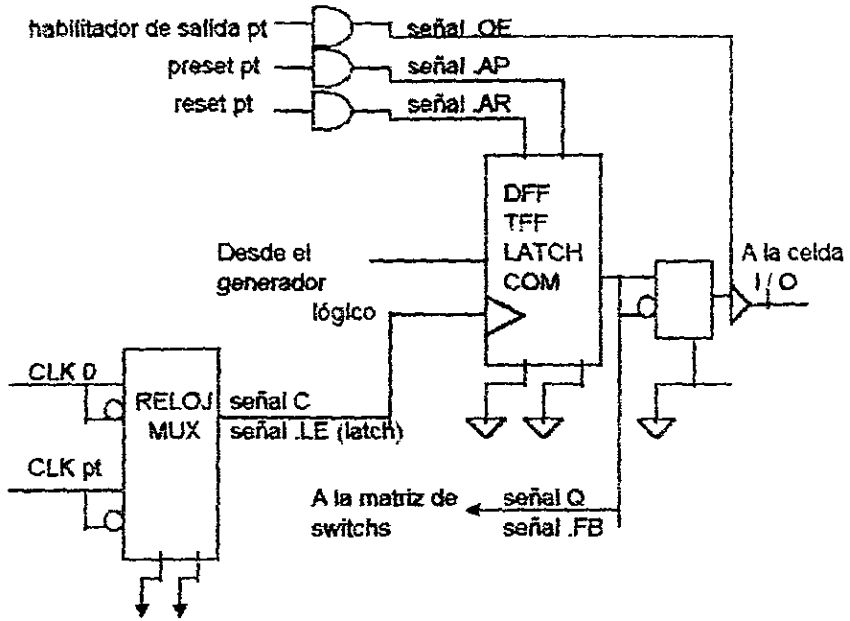


figura C.4 Celda lógica.

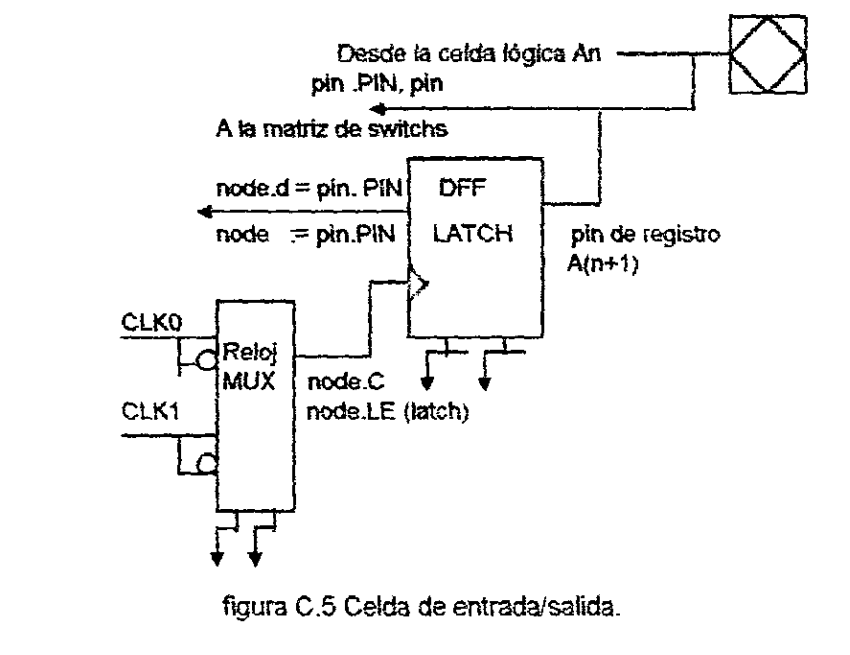


figura C.5 Celda de entrada/salida.

CONVENCIÓN DE NOMBRES

En el proceso de asignación de pines se sigue una convención para nombrar a los pines y nodos asociados a una macrocelda en una MACH 215. Los nombres de todos los pines y nodos del MACH 215 se muestran en la tabla C.2.

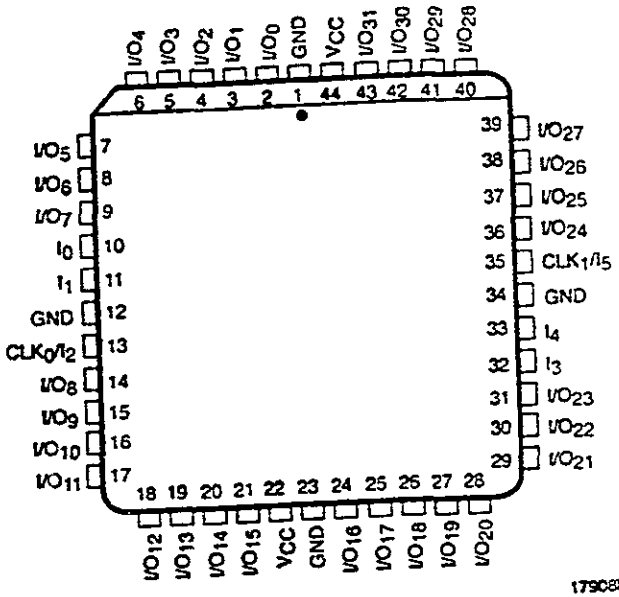
TABLA C.2 PINES Y NODOS DEL MACH 215

PIN	NOMBRE	NODO	NOMBRE	PIN	NOMBRE	NODO	NOMBRE
1	GND	-	-	23	GND	-	-
2	A0	45	A1	24	CO	61	C1
3	A2	46	A3	25	C2	62	C3
4	A4	47	A5	26	C4	63	C5
5	A6	48	A7	27	C6	64	C7
6	A8	49	A9	28	C8	65	C9
7	A10	50	A11	29	C10	66	C11
8	A12	51	A13	30	C12	67	C13
9	A14	52	A15	31	C14	68	C15
10	I0	-	-	32	I3	-	-
11	I1	-	-	33	I4	-	-
12	GND	-	-	34	GND	-	-
13	CLK	-	-	35	CLK1	-	-
14	B14	60	B15	36	D14	76	D15
15	B12	59	B13	37	D12	75	D13
16	B10	58	B11	38	D10	74	D11
17	B8	57	B9	39	D8	73	D9
18	B6	56	B7	40	D6	72	D7
19	B4	55	B5	41	D4	71	D5
20	B2	54	B3	42	D2	70	D3
21	B0	53	B1	43	D0	69	D1
22	VCC	-	-	44	VCC	-	-

C.2 DIAGRAMAS DEL MACH 215

CONNECTION DIAGRAM
Top View

PLCC



Note:
Pin-compatible with MACH110, MACH111, MACH210, MACH211, and MACH215.

PIN DESIGNATIONS

CLK/I = Clock or Input
GND = Ground
I = Input
I/O = Input/Output
Vcc = Supply Voltage

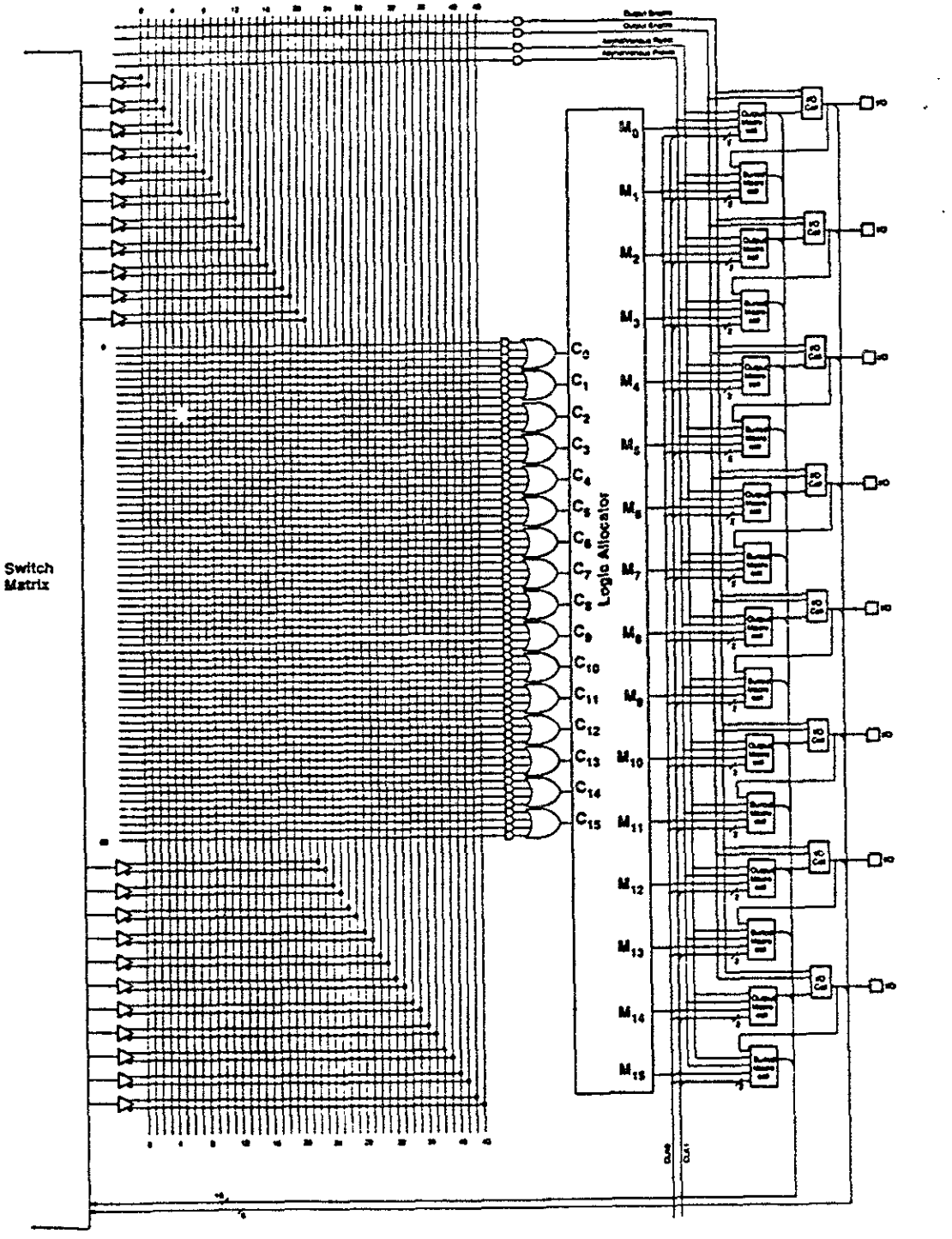


Figure 1. MACHLV210 PAL Block