UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingeniería

AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADO A PAL'S

T E S I S

PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

SERGIO AMBRIZ MAGUEY
MARTIN PEREZ MONDRAGON

DIRECTOR DE TESIS: M. EN I. JESUS SAVAGE CARMONA

MEXICO, D. F. 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

PROLOGO

Uno de los campos que más impulso ha recibido en la última década es la graficación por computadora.

Algunos de los sistemas de computación más avanzados que se emplean hoy en dia, se diseñan contemplando la generación de despliegues gráficos, reconociendose asi el valor de una imagen como un medio eficaz de comunicación.

Prácticamente, no existe en la actualidad ningún área en la cual no pueda utilizarse la gráficación con alguna ventaja; así, se encuentran gráficas en la mayoría de las aplicaciones basadas en ambientes de computación. Sin embargo, las primeras aplicaciones en ciencia e ingeniería se basaron en equipos costosos y complicados, lo cual se ha superado debido a los adelantos registrados en computación permitiendo así acceder a la graficación como una herramienta útil y práctica a bajo costo.

A finales de la década de los 70's, la graficación por computadora se integró al proceso de diseño, originando el crecimiento de la tecnología CAD *-Computer-Aided Design-* o Diseño Asistido por Computadora. Esta tecnología ofrece poderosas herramientas para el diseño en ingeniería, permitiendo, entre otras cosas, agilizar el proceso de conceptualización, modelado, análisis y documentación de un producto con las etapas respectivas de retroalimentación, formando así, un ambiente de control de ciclo cerrado que contrasta con los mecanismos tradicionales de diseño.

La influencia de la tecnología CAD a alcanzado las áreas eléctrica y electrónica, con ejemplos como el diseño de circuitos electrónicos empleando sistemas interactivos basados en iconos gráficos para la representación de los diferentes componentes, con lo cual un diseñador

puede construir un circuito, agregando componentes en forma sucesiva extraídos de una Base de Datos. Además, el despliegue gráfico puede emplearse para comparar circuitos equivalentes con la finalidad de minimizar el número de componentes o bien, para llevar a cabo una mayor integración de sus elementos.

INTRODUCCION

El estudio e investigación de los dispositivos PAL (*Arreglo Lógico programable)* promete resultados en una diversidad de áreas, siendo de particular interés el área electrónica.

El propósito del sistema AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADO A PAL's consiste en reducir las horas-hombre dedicadas al diseño de circuitos electrónicos, reemplazando técnicas tradicionales mediante la aplicación de tecnologías de vanguardia.

Los dispositivos PAL, se emplean para obtener una reducción considerable de un circuito lógico. Algunos beneficios directos que proporcionan son: reducción física del número de elementos de un circuito, reprogramación del PAL con objeto de una nueva aplicación, reducción del tiempo de respuesta de los diseños al sustituir elementos convencionales TTL SSI/MSI por dispositivos PAL y permitir una ganancia en la reducción de un circuito lógico de 4:1, lo cual, físicamente, se traduce en una mayor integración.

Un punto fundamental, referido a los PAL's, lo constituye el manejo de formatos. En el esquema de diseño de este trabajo, se opto para la generación del mapa de fusibles el formato estandard JEDEC el cual es uno de los más utilizados hasta el momento a nivel mundial.

Además es posible mencionar varios aspectos relevantes que se involucran en la creación del sistema:

i.- Análisis y aplicación de una técnica avanzada de diseño en electrónica que permita acelerar el proceso clásico de creación, modificación y optimización de circuitos lógicos.

ii.- Estudio y empleo de sistemas de Diseño Asistido por Computadora.

iii.-Generación de recursos humanos en el área de los dispositivos PAL.

iv.- Generación de una alternativa económicamente atractiva a corto plazo para el área de diseño de circuitos lógicos.

En el capítulo I se describen y mencionan a grandes rasgos los diferentes dispositivos lógicos programables (PLD), llevando a cabo una comparación con el fin de identificar las ventajas y desventajas que presentan los dispositivos PAL con respecto a otros PLD para justificar su elección en la realización del sistema.

En el capítulo II se describe la técnica de máquina de estados, sus carácteristicas y aplicaciones, así como tópicos involucrados con el diseño de sistemas digitales. Para continuar con una descripción detallada del algoritmo de máquina de estados (ASM) el cual es aplicado en el área electrónica.

El capítulo III, muestra y describe el ambiente utilizado para el desarrollo del sistema; como el empleo de equipo RT-PC y los dispositivos de control involucrados, así como el estandard de graficación PHIGS empleado para el control de los atributos gráficos.

El capítulo IV, describe a grandes rasgos y en forma detallada las fases del sistema; las estructuras de información, la técnica empleada para la optimización de funciones booleanas, así como la utilización de un software para la generación del mapa de fusibles en formato JEDEC.

CAPITULO I

LOGICA PROGRAMABLE

1.1 Introducción

La aparición en el mercado de grandes bloques LSI (integración a grande escala) y MSI (integración a mediana escala) a bajo precio, ha cambiado el planteamiento para la implantación de circuitos digitales. Así, el concepto de costo mínimo, que cuando se utilizaban compuertas SSI (integración a baja escala) coincidía con la utilización del mínimo número de éstas con el mayor número de entradas, al utilizar bloques LSI, se convierte en número mínimo de patas del circuito integrado. Esto es así, dado que el costo de un circuito LSI depende más del número de patas que de la complejidad interna del mismo, porque el mayor peso en el costo total del circuito integrado recae en el número de conexiones que hay que realizar entre las pastilla de silicio y el mundo exterior y este número de conexiones es evidentemente igual al de patas.

La fiabilidad de un circuito es en gran parte función del número de soldaduras que hay que realizar, por lo que al implantar circuitos utilizando bloques LSI la fiabilidad aumenta notablemente. Esto se debe a que el número de soldaduras resulta muy disminuido dado que, como se ha indicado, utilizando bloques LSI se minimiza el número de patas así como el de interconexiones a realizar. La tarjeta de circuito impreso resulta así mismo de menor tamaño y complejidad que utilizando circuitos SSI o MSI.

De entre los diferentes bloques LSI disponibles, los más apropiados para implantar circuitos digitales son las memorias (PROM) y los arreglos lógicos programables (PAL).

En el presente capitulo se hace una recopilación de las diferentes caracteristicas de los dispositivos lógicos programables (PLD's), asi como de los dispositivos PAL (Arreglo Lógico Programable) a los cuales se encuentra enfocado el desarrollo del sistema que en capitulos posteriores se presentara de manera formal. De esta manera, se persigue proporcionar los antecedentes necesarios para el mejor manejo y utilización del sistema *Ambiente CAD para diseño de sistemas digitales orientado a PAL's.*

I.2 Anatomía de un PAL

Los dispositivos que emplean un arreglo lógico programable (PAL) poseen varias caracteristicas en común con las memorias (PROM) y los arreglos lógicos programables (PLA), ya que todos estos dispositivos comparten una estructura básica interna a base de compuertas AND-OR, pero varian en sus caracteristicas lógicas y en el modo de programación.

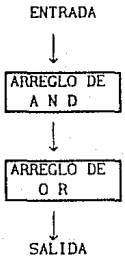


Figura 1.1 Arquitectura básica

En la figura 1.1 se muestra la estructura básica, la cual consiste de 2 niveles; en donde el primer nivel es un arreglo de compuertas AND que acepta las entradas para la creación de funciones booleanas (minterminos) y el segundo nivel lo constituye el arreglo OR que presenta las salidas que forman los mastérminos. Asi, el arreglo OR puede combinar varias funciones AND de acuerdo al tipo de dispositivo programable utilizado, es decir, se cuenta con diferentes dispositivos que presentan caracteristicas especificas tales como: el

número de entradas, números de salidas, así como diferentes tipos de salidas con las que se podrán realizar las funciónes booleanas más sofisticadas.

Esta estructura básica, hace que los dispositivos programables sean ideales para la implementación de lógica booleana (suma de productos) la cual es generada por alguna técnica de diseño lógico para la obtención de funciones booleanas como los mapas de Karnaugh [1].

I.3 Descripción de una arquitectura PAL

La arquitectura de una PAL se muestra en la figura 1.2 de donde la estructura básica de un dispositivo PAL es opuesta a la de una PROM: el arreglo de compuertas AND es programable y el arreglo de compuertas OR es fijo, la única diferencia con la PROM es que las entradas no son decodificadas; ya que no se cumple que el número de compuertas AND sea 2^n (donde n es el número de entradas) razón por la cual los dispositivos PAL eliminan la ineficiencia de las PROM donde se debe de cumplir que existan 2^n compuertas AND para n entradas, por lo tanto, de lo anterior podemos afirmar que la utilización de los dispositivos PAL permite tener un número de entradas más considerable.

En otras palabras, el incremento del número de entradas de los PAL no incrementa dramáticamente la cantidad de fusibles requeridos para el empleo del arreglo de compuertas, por ejemplo; para 6 entradas y 16 compuertas AND, tendremos 12x16 fusibles, para 10 entradas se tendrán 20x16 fusibles. Al hablarse fusibles se esta haciendo referencia a la conexión física que existe en el arreglo AND, dada su presentación original posee conexión con todas las entradas, así, la programación se lleva a cabo quemando los fusibles utilizando un dispositivo especial que elimina la conexión que presenta originalmente. Esta alteración de la configuración interna del dispositivo puede ser temporal o definitiva de acuerdo al dispositivo que se utilice.

El arreglo de compuertas OR de los dispositivos PAL es dedicado. A una compuerta OR de salida se pueden tener varias conexiones de

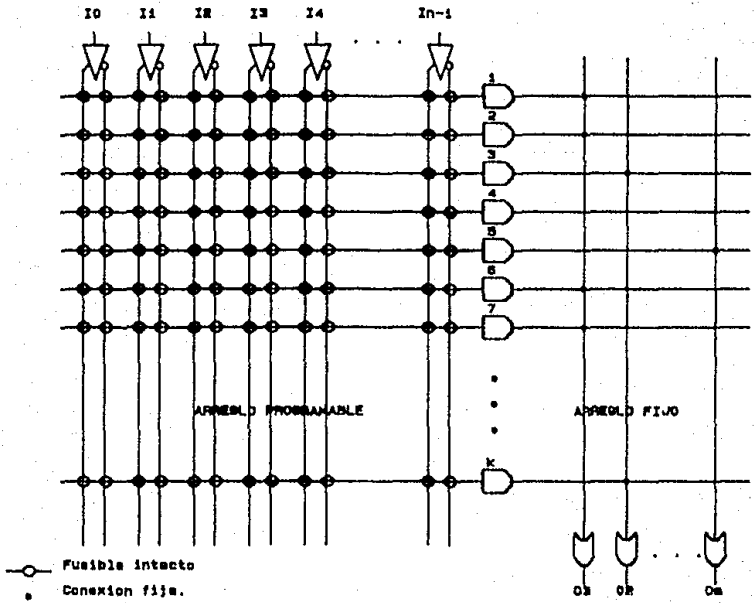


Figura 1.2 Arquitectura de un PAL.

compuertas AND en particular, por ejemplo podemos tener 4 compuertas AND dedicadas para cada una de estas compuertas OR de salida. En un dispositivo PAL, la salida que nos proporciona esta limitada por el número de compuertas AND dedicadas a un compuerta OR como se mencionó anteriormente; de lo cual se concluye que el número de compuertas AND requeridas para una función lógica no debe exceder este número de compuertas. Por lo tanto, concierne al diseñador identificar previamente el tipo de dispositivo que satisfaga los requerimientos.

Por el número de entradas y las características que presenta su arquitectura, los dispositivos PAL son ideales para la implementación de una gran variedad de funciones lógicas; dentro de estas características se incluye: la programación de entradas/salidas, salidas combinacionales o con registros, con realimentación interna hacia el arreglo de compuertas AND, o niveles de salida.

I.3.1 Arquitectura del plano AND

Este plano proporciona la interconexión de las entradas (ambas verdadera o complemento) que requieren compuertas AND de manera lógica para crear de esta forma los *términos producto* y *términos producto control*, de donde los términos producto lógico son utilizados para funciones lógicas típicas y los términos producto control son requeridos para aplicaciones en funciones de control como: habilitación de salidas, inicialización, precargado de datos y observabilidad.

El número total de entradas y el término producto es determinado por el tamaño del plano AND que ofrece la arquitectura del dispositivo PAL, por tal motivo se debe seleccionar el tipo de PAL que cumpla con los requerimientos de la función lógica deseada.

I.3.2 Arquitectura del plano OR

Este plano determina la conectividad de las compuertas AND con las salidas y en el cual además se definen 3 caraterísticas principales:

- El número de compuertas OR
- El número de términos producto (PT's) por salida
- La distribución de términos producto

En una arquitectura típica de un dispositivo PAL, las salidas de las compuertas AND son conectadas a compuertas OR mediante el arreglo del plano OR.

Empero, existe una limitación en los PAL's a ser considerada, la cual está presente en el plano AND-OR donde hay un número finito de entradas a compuertas AND y un número de compuertas OR dedicadas como salidas que serán tomadas en cuenta al momento de diseñar.

I.3.3 Arquitectura de las macroceldas de salida

Las especificaciones mínimas para las macroceldas de salida se enuncian a continuación:

- Como salida normal
- Como uso de flip-flop para almacenamiento
- Salidas organizadas
- Flexibilidad para realimentación

Por otra lado, la(s) salida(s) se pueden configurar para ser utilizadas de acuerdo a las siguientes propiedades lógicas requeridas por la aplicación del diseño. Salida secuencial y combinacional.

I.3.3.1 Salida secuencial

Los circuitos secuenciales se pueden clasificar en dos grandes grupos: asincronos y sincronos.

En los sistemas secuenciales asincronos, los cambios de estado se producen en cuanto están presentes las entradas adecuadas, con los retrasos inherentes a las velocidades finitas de conmutación de los dispositivos fisicos utilizados.

En un sistema secuencial sincrono, por el contrario, los cambios de estado se producen únicamente cuando, además de estar presentes las entradas adecuadas, se produce la transición de un cierta señal, compartida por todos los biestables (flip-flops) del sistema y que, por lo tanto, sincroniza su funcionamiento. Esta señal se denomina, reloj del sistema, y los cambios de estado se producen en sus transiciones de 0 a 1 ó de 1 a 0, dependiendo de la tecnología propia de los circuitos electrónicos utilizados (generalmente transiciones negativas en tecnología TTL y positiva en tecnologías CMOS).

Debe entenderse, esta clasificación como funcional, es decir, externa a la constitución de los circuitos. En realiadad un circuito lógico de cualquier tipo puede considerarse como formado por compuertas, elementos intrinsecamente asincronos, puesto que proporcionan salidas instantáneas en cuanto están presentes las entradas adecuadas. La combinación de estas compuertas entre si es lo que da a un circuito su característica funcional de *combinacional o secuencial* y, dentro de estos últimos, sincronos o asincronos.

De los párrafos anteriores, se aprecian las características funcionales de los circuitos secuenciales, así, el uso de flip-flop en la macrocelda de salida del dispositivo PAL se puede clasificar en alguno de los siguientes tipos:

- flip-flop D (disparado por flanco de subida)
- flip-flop J-K
- flip-flop R-S
- flip-flop T o latches

Las salidas con flip-flop tienen su mayor aplicación dentro del área de diseño de sistemas síncronos, mientras que las salidas con latches son requeridas para aplicaciones lógicas asíncronas.

Tradicionalmente, *la velocidad de respuesta, la simplicidad de arquitectura y el uso de flip-flop D* en la salida, son los criterios más importantes para la selección de los dispositivos PAL's para la realización de aplicaciones especializadas.

I.3.3.2 Salida combinacional

La estructura de las macroceldas de salida, determinan la flexibilidad para la realimentación; esta característica es uno de los requerimientos más importantes para tener salidas combinacionales, salidas con registros, configuración de pines para entrada/salida. Por otra parte, la realimentación puede utilizar líneas múltiples o simples, con el fin de incrementar la flexibilidad del dispositivo para aplicaciones más especializadas. Ver figura 1.3.

I.3.4 Arquitectura de pines entrada-salida

La programación de los pines para ser empleados como entrada-salida, es uno de los más importantes recursos que presentan los dispositivos PAL, con lo cual se pueden llevar a cabo implementaciones de las más complejas y variadas funciones lógicas. Además, se pueden determinar los pines que funcionarán como entradas dedicadas, pines para salida ó control dinámico de pines para entrada-salida de acuerdo a los requerimientos de diseño.

I.3.4.1 Flexebilidad para controlar la habilitación de salidas y entrada-salida bidireccional

El diagrama lógico de una estructura bidireccional de un PAL se muestra en la figura 1.4, de donde se puede observar que: la función

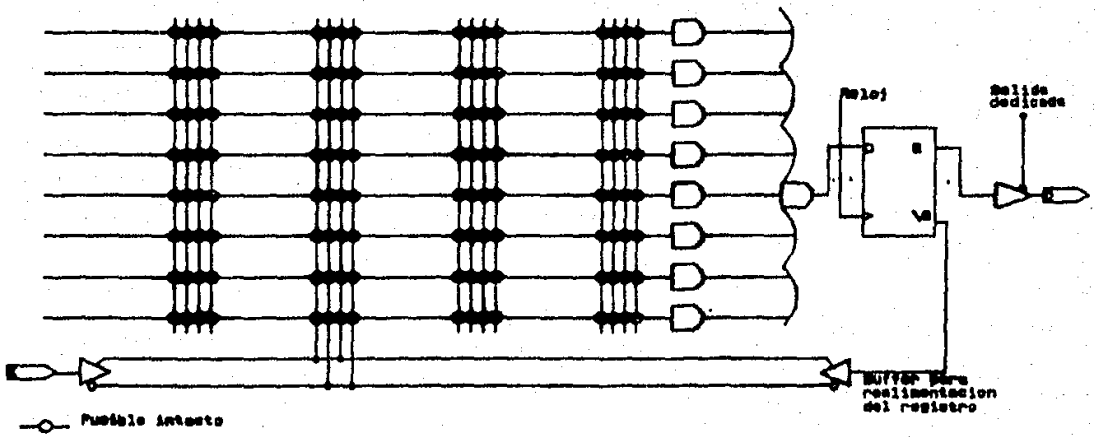


Figura 1.3 Salida combinacional.

más importante de los dispositivos PAL con estructura bidireccional, es la flexibilidad que proporcionó en el control para habilitación de salidas, así, la salida habilitada pueder ser dedicada, es decir, controlada por un pin, programada ó controlada por un término producto de un arreglo de compuertas AND.

Por otro lado, el buffer asociado con el pin de salida puede ser programado en alguno de los siguientes modos de acuerdo a los requerimientos:

- Como salida dedicada
- Como entrada dedicada
- Control dinámico para entrada-salida

Cuando se opta por la programación como salida dedicada, el buffer a la salida se encuentra habilitado, por lo tanto, la función lógica es realimentada hacia el arreglo AND. Con esta realimentación se permite la implementación de funciones lógicas más complejas haciendo uso de dos o más niveles de compuertas AND-OR.

De otra manera, cuando se programa como entrada dedicada, el término AND-OR asociado con este pin no es utilizado. Por lo tanto, el diseñador no se encuentra limitado a un número fijo de pines para ser empleados como entrada-salida, teniendo así un rango de programación de acuerdo a las necesidades de diseño.

Finalmente, cuando se programa para un control dinámico de entrada-salida i.e., habilitar o deshabilitar por lógica combinacional una o más entradas, este pin puede ser usado como entrada con el fin de retener la capacidad lógica de las compuertas AND-OR. Esta peculiaridad se usa especialmente en aplicaciones de control, transferencia de datos, etc. Una entrada-salida serial es un ejemplo del concepto antes mencionado; cuando se utiliza el pin para corrimientos hacia la izquierda es una entrada serial, y si existe corrimiento hacia la derecha funciona como salida serial.

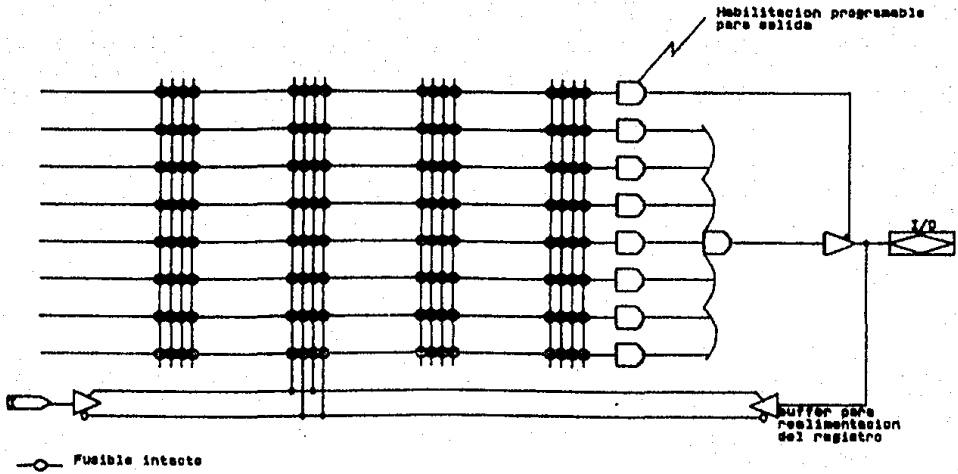


Figura 1.4 Estructura bidireccional.

Con el modo de programación control dinámico se proporciona la máxima utilización de los recursos que permite la arquitectura de un PAL. También es importante mencionar la utilización de la salida en activo bajo, con realimentación hacia el arreglo AND, la cual es útil para la implementación de niveles lógicos múltiples; las compuertas extras hacen de estas salidas excelentes para el control de generación de señales, codificación y decodificación de las mismas, así podemos decir que se pueden utilizar los dispositivos PAL en las áreas de comunicaciones, sistemas de adquisición y procesamiento de señales donde se requiera velocidad de respuesta, fiabilidad y precisión, así como en el diseño de estaciones de trabajo [6].

I.3.5 Características de la arquitectura PAL

Los dispositivos PAL poseen una gran cantidad de atributos que los hacen perfectos para la implementación de funciones lógicas, de los cuales a continuación se mencionan su características más distintivas de dichos dispositivos a manera de recapitulación, figura 1.6.

- Pines programables para entrada-salida
- Flexibilidad para el control de habilitación de salidas y entradas-salidas bidireccionales.
- Estructura de salida dedicada
- Polaridad programable
- Flexibilidad de esquema de frecuencia
- Características adicionales:
    - accesabilidad
    - controlabilidad
    - observabilidad

I.3.6 Características generales de los PAL

Finalmente podemos hacer una descripción general de los dispositivos PAL, a manera de justificar su elección al comparar con otros dispositivos lógicos programables (PLD), de tal manera que a

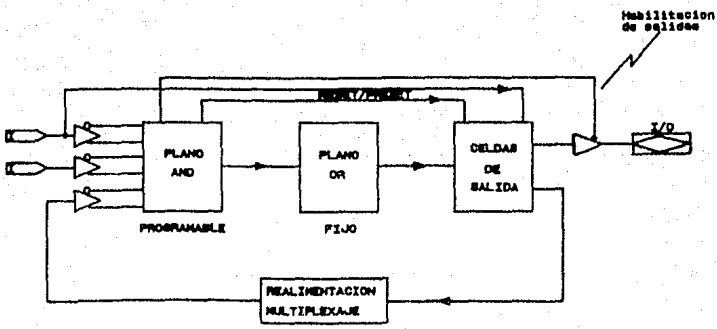


Figura 1.8 Caracteristicas de una arquitectura PAL.

continuación se mencionan en forma concisa y general las características de los dispositivos PAL.

I.3.6.1 Tiempo requerido para el proceso de diseño

Los dispositivos lógico programables son la via más óptima para la reducción del tiempo involucrado dentro de la fase de diseño. La flexebilidad de la arquitectura que presentan los dispositivos PAL, permite realizar la interface entre diferentes bloques lógicos, por lo tanto, cada bloque puede ser diseñado con un mínimo de interacciones para de esta manera simplificar la elaboración de: prototipos de algún sistema, tarjetas para computadoras o simplemente ofrecer más funciones por tarjeta, lo cual se traduce a una mayor integración física.

Otra ventaja que ofrecen los dispositivos lógicos programables, es la reducción gradual de los costos así como el tiempo asociado que inplican los cambios de errores lógicos en el diseño de sistemas que se encuentran en procesos finales, así como modificaciones a productos ya existentes; dichas modificaciones se pueden llevar a cabo reprogramando los dispositivos, creando de está manera un nuevo mapa de fusibles para el arreglo de compuertas AND alterando así la configuración anterior.

I.3.6.2 Velocidad de ejecución

La velocidad en el tiempo de ejecución que requieren los sistemas, se puede incrementar de manera notable a través del uso de lógica programable, dado que el diseñador tiene la libertad para óptimizar la arquitectura del diseño, empleando un dispositivo lógico programable (PLDs) para la implementación de una aplicación más precisa. Así, el diseñador puede elaborar su sistema de la manera más eficiente e incrementar la velocidad de ejecución de dicho desarrollo.

I.3.6.3 Reducción en los tiempos de retardo de las señales

Cuando una función lógica es implementada utilizando elementos standard TTL SSI/MSI, (lógica combinacional) el retardo total en la respuesta del sistema incluye el tiempo de respuesta requerido por algunos buffer contenidos en los circuitos integrados, así como el tiempo de respuesta requerido por las compuertas utilizadas; no ocurre lo mismo cuando se utiliza un dispositivo lógico programable, donde el tiempo de propagación de la señal se reduce considerablemente debido a su arquitectura.

I.3.6.4 Reducción de costos

La disponibilidad de los dispositivos PAL's, puede proporcionar el desarrollo de una lógica compleja equivalente a 300 compuertas, con lo cual se puede reemplazar más del 90% de los elementos standard TTL. La implementación de un diseño utilizando lógica programable puede significar, de manera considerable, la reducción de espacio en diseño de circuitos impresos ó el número de tarjetas necesarias para implementar una funcion especifica.

Como consecuencia de lo antes mencionado, el resultado es un bajo costo del sistema final o alternativamente la disponibilidad de ofrecer más funciones en la misma tarjeta.

I.3.6.5 Reducción de componentes

Comparando contra los elementos standard TTL SSI/MSI, la lógica programable nos proporciona una reducción en el número de circuitos integrados necesarios para llevar a cabo la implementación de una función lógica, logrando de esta manera una mayor integración fisica del diseño.

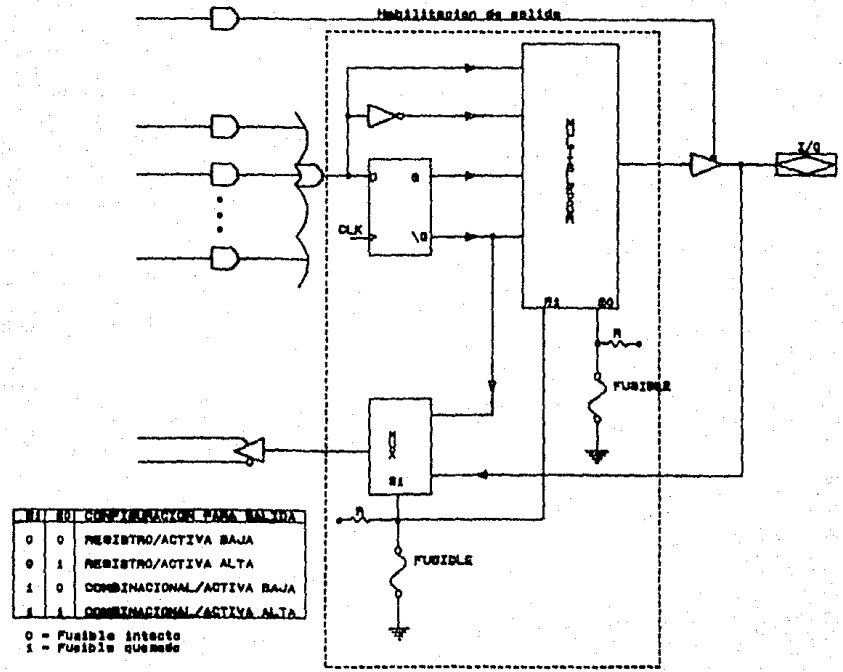


Figura 1.8 Macrocelda.

I.3.6.6 Reducción de interconexiones

La parte con menor confiabilidad dentro de un sistema digital son las conexiones que existen entre los diferentes circuitos integrados. La reducción del número de circuitos integrados requeridos para un diseño, implica un menor numero de conexiones y por consiguiente, se incrementa su confiabilidad.

I.3.6.7 Seguridad del diseño

Así, para poder utilizar los dispositivos PAL se requiere de una programación especial, de está manera se previene la duplicación de dichos dispositivos, lo cual es ideal para cualquier aplicación especializada donde la seguridad del diseño es lo esencial.

I.4 Bloques LSI

Además de los dispositivos PROM's y PLA's, existen actualmente algunos otros tipos de dispositivos lógicos programables disponibles para el diseño en el área de electrónica. Los dispositivos PAL's; los cuales son un caso especial de los PLA's, poseen la misma estructura básica de compuertas AND/OR, pero en los PAL's, solamente el arreglo AND es programable y cada compuerta OR es dedicada a un número fijo de compuertas AND. Ver figura 1.2.

Como consecuencia, las funciónes lógicas se pueden realizar en un dispositivo PAL utilizando términos que pueden ser comunes a más de una función. Además, los PAL's son menos costosos en comparación con otros dispositivos programables, y de facil programación.

Existen otras variantes de dispositivos que se construyen empleando la estructura básica AND/OR, entre los cuales se puede mencionar a los dispositivos lógicos reprogramables (EPLD) desarrollados por la Corporación Altera [4]. Dichos dispositivos consisten de una serie de macroceldas (se muestra en la figura 1.6), conteniendo típicamente un

arreglo lógico programable, así como flip-flop a la salida, etc. para mayor detalle [4].

Una de las desventajas que presentan los arreglos programables, es lo concerniente a la "*rigidez*" de la arquitectura, impuesta por las interconexiones fijas que se encuentran involucradas. Está rigidez es resultado de un pequeño porcentaje (normalmente menor al 20%) de compuertas que son utilizadas en una aplicación típica.

Empero, se elimina esta ineficiencia, ya que dada la estructura del arreglo de compuertas; las cuales típicamente nos proporcionan un vasto arreglo de 2 entradas a compuertas AND. Las funciones más complejas pueden ser implementadas en un arreglo utilizando las interconexiones apropiadas (usando la facilidad de realimentar el arreglo de compuertas AND, mediante el uso de flip-flops, etc.). Por otra parte, la utilización de compuertas dentro del arreglo puede ser mayor del 90% en algunas aplicaciones especializadas.

Hasta hace poco los PAL's no eran muy empleados dada su baja velocidad y alto costo. Empero, los recientes avances tecnológicos permiten la construcción de dispositivos PAL con tiempos de respuesta de pocos nanosegundos y un precio altamente competitivo. Existen en la actualidad PAL's que pueden ser programados por el usuario después de haber sido construido el circuito integrado.

Como se puede apreciar en la figura 1.7 un PAL tiene n entradas I_0 a I_{n-1} de las cuales se dispone internamente tanto en su forma directa como en la complementada. Las compuertas AND, 1 a k poseen cada una n entradas y son capaces, por lo tanto, de realizar cada una de ellas un producto de todas o algunas de las entradas, tanto en su forma directa como complementada. Se pueden formar, por lo tanto, K productos distintos. El producto concreto que hará cada compuerta AND depende de las conexiones indicadas por un punto (figura 1.7) localizados en el plano AND. Así, la compuerta uno realiza el producto $K_1 = I_0 \backslash I_1 \ldots$ y la compuerta tres $K_2 = I_2 \ldots$

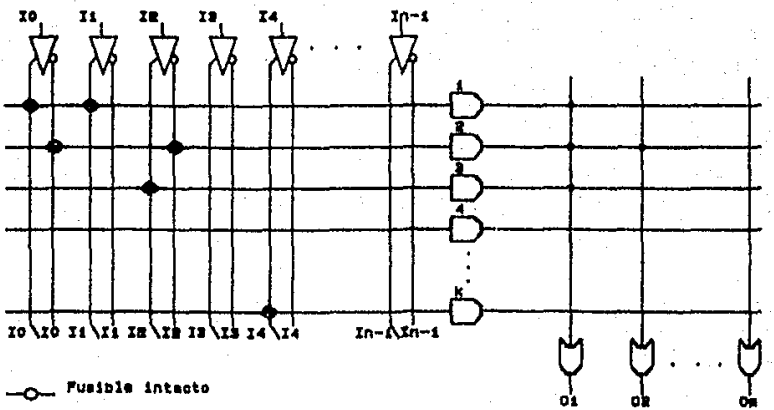

I0
I1
I2
I3
I4
In-1
1
2
3
4
K
I0 \I0 I1 \I1 I2 \I2 I3 \I3 I4 \I4
In-1 \In-1
Fusible intacto
O1
O2
Om

Figura 1.7

Las *m* salidas O_1 a O_m del dispositivo PAL proceden de las compuertas OR de k entradas, cada una puede realizar la suma de todos o algunos de los productos K_1 a K_k según las conexiones que se realicen en el plano OR. Así para el ejemplo indicado en la figura 1.7, las salidas O_1 y O_2 realizan las siguientes funciones:

$O_1 = I_0\backslash I_1 \ldots + \backslash I_0\backslash I_2 \ldots + I_2 \ldots + \ldots$
$O_2 = \backslash I_0\backslash I_2 \ldots + \ldots$

El número de productos k que se pueden realizar con un dispositivo PAL, es un número pequeño y muy inferior a las 2^n combinaciones posibles de las entradas (ver bloques LSI). Un dispositivo PAL típico tiene n = 14 entradas, m = 8 salidas y sólo puede realizar K = 96 productos en vez de los 16384 posibles con 14 variables.

Para una mejor apreciación de la utilidad de los dispositivos PAL, se muestra, a manera de ejemplo didáctico, un circuito combinacional de cuatro entradas *a, b, c y d* y tres salidas S_1, S_2 y S_3, cuyas funciones de salida, ya simplificadas son las siguientes:

$$S_1 = a\backslash c + bc\backslash d + \backslash a\backslash bcd \qquad (1.1)$$
$$S_2 = ab\backslash c + a\backslash bcd + a\backslash cd + bc\backslash d + \backslash a\backslash c\backslash d \qquad (1.2)$$
$$S_3 = a\backslash c + a\backslash bcd \qquad (1.3)$$

se utilizara por razones de claridad, un dispositivo PAL hipotético de cuatro entradas, seis estados (compuertas AND) y tres salidas para realizar estas funciones. En la figura 1.8 se muestra el esquema de la solución obtenida con el citado PAL. Como puede comprobarse la salida S_1 será la suma de los productos K_1, K_2 y K_3 estos productos son respectivamente a\c, bc\d y a\bcd, por lo que resulta:

$$S_1 = a\backslash c + bc\backslash d + \backslash a\backslash bcd \qquad (1.4)$$

que es la expresión (1.1) que se desea obtener. De forma análoga se comprueba que las salidas S_2 y S_3 cumplen con las expresiones (1.2) y (1.3) respectivamente.

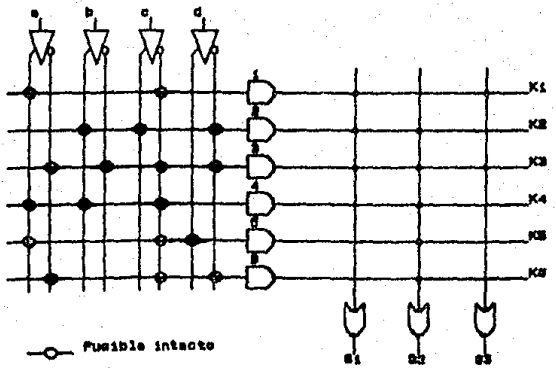


Figura 1.8

De lo anteriormente indicado se deduce que los dispositivos PAL están especificamente diseñados para realizar funciones lógicas que tengan un pequeño número de términos de un gran número de variables y que hay que utilizar la función ya simplicada. Para aplicaciones de este tipo el empleo de PAL conduce a una solución más económica y fiable que la utilización de PROM.

Así, por el ejemplo, si se desea implementar un circuito combinacional que tenga 12 entradas y 8 salidas se puede realizar o bien con 4 PROM de 1K-byte (1024 x 8), o bien con un dispositivo PAL de 14 entradas y 8 salidas, siempre que el número total de productos distintos para las 8 salidas sea menor o igual que 96. Para la primera solución se conectarian 10 de las 12 entradas del sistema a las 10 entradas (2^{10}= 1024) de todas las PROM, y las otras dos a las dos entradas de inhibición de cada PROM. Como con dos variables se pueden formar 4 combinaciones posibles, hacen falta 4 PROM, en la figura 1.9 se muestra el esquema de esta solución. Evidentemente, la solución mediante PAL resulta notablemente más ventajoso en este caso.

Para algunas aplicaciones el número de productos que se pueden realizar con las variables de entrada es demasiado reducido. Esto se puede solucionar conectando el paralelo las entradas de varios dispositivos PAL y realizar la función OR con las salidas equivalentes de los distintos PAL. En la figura 1.10(a) se puede apreciar el esquema de esta solución. El número de productos distintos que se pueden tomar ahora es de 96, r siendo el número de dispositivos PAL utilizados. El número de entradas y salidas corresponde a cualquiera de los PAL, es decir, 14 entradas y 8 salidas.

Cuando el número de salidas sea insuficiente, se pueden utilizar varios PAL con sus entradas equivalentes conectadas en paralelo. De esta forma se obtiene un PAL que tiene el mismo número de entradas y términos que cualquiera de los PAL, siendo, sin embargo, el número de salidas igual al producto de número de PAL utilizados por el número de salidas. Una solución similar puede emplearse para expandir el número de salidas de una PROM.

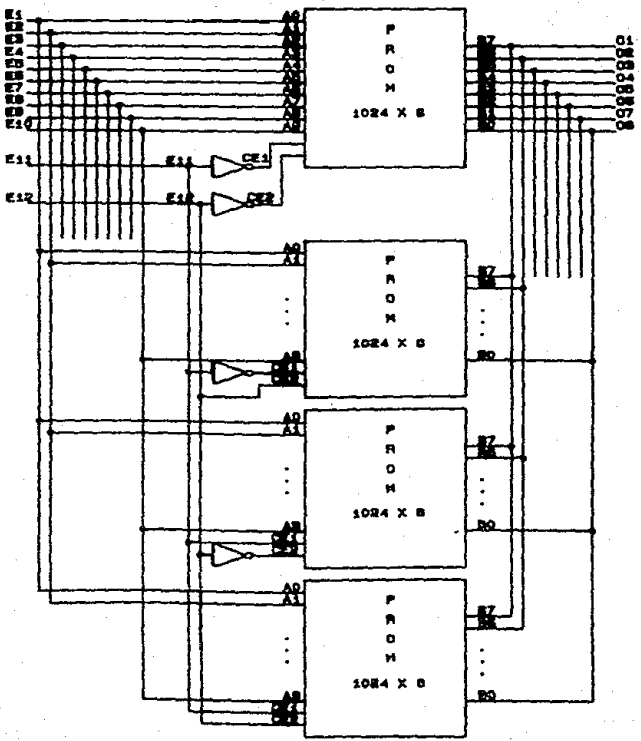
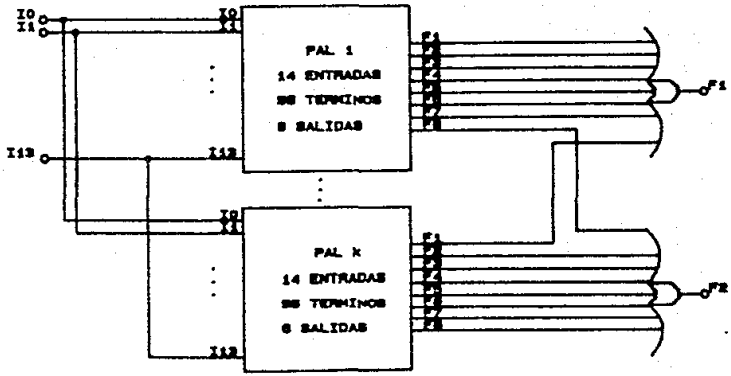
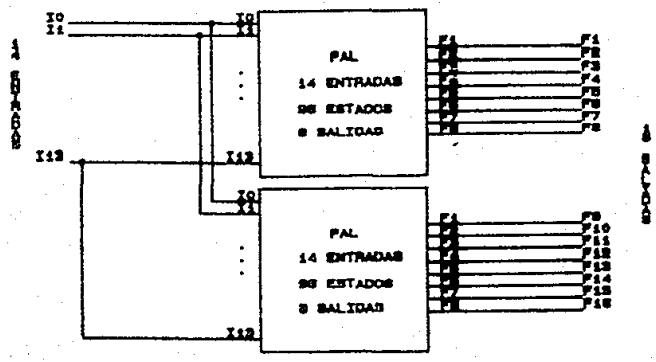


Figure 1.8

En la figura 1.10(b) se presenta un dispositivo PAL de 14 entradas, 96 estados y 16 salidas, obtenida al configurar en paralelo las entradas de dos PAL's de 14 entradas, 96 estados y 8 salidas.



(A)



(B)

Figura 1.10

CAPITULO II

TECNICAS DIGITALES

II.1 Introducción

Un sistema secuencial sincrono es aquél en el que los cambios de estado se producen únicamente cuando, además de estar presente la combinación adecuada de entradas, tiene lugar la transición de cierta señal, denominada generalmente *reloj (clock)* que, sincroniza el funcionamiento de los flip-flop's de todo el sistema.

En este capitulo se introduce el concepto de *algoritmo de máquina de estados*, el cual se conoce dentro de los procesos tradicionales de diseño lógico como; cartas ASM, así como los conceptos básicos de diseño lógico.

II.2 Conceptos básicos de diseño lógico

Una caja negra, es una parte funcional que realizá una tarea especifica, la cual puede ser sustituida dentro de un algoritmo.

Un concepto importante dentro de un sistema lógico, es la función que realiza una máquina lógica, ya que ahi se describe en forma detallada las caracteristicas que posee dicha máquina. Este concepto nos conduce a una representación de la máquina mediante cajas negras, permitiendo predecir de esta manera funciones por bloques, creando así lo que se conoce como: *diagrama de bloques*.

El comportamiento de una máquina estados se refiere a la relación que existe entre las entradas y salidas mostrandose en un diagrama de bloques como, *terminales en el tiempo* implicados en una caja negra. Así una caja negra se muestra en la figura 2.1

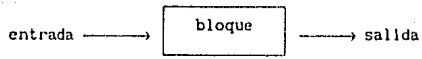


Figura 2.1 Caja negra.

Las caja negras son utilizadas dentro de un sistema lógico para hacer referencia a una función en especifico, siendo estas descritas por álgebra Booleana. Las entradas y las salidas son dos variables, esto es, que pueden presentar información verdadera o falsa. Este hecho hace posible una especificación exacta del comportamiento de la caja negra, porque no se pueden tener entradas o salidas ambiguas. Un sistema lógico es de gran intéres en el área de electrónica porque se deben de utilizar especificaciones exactas, para lograr un circuito más simple y funcional.

El proceso de diseño lógico puede ser descrito por el diagrama de la figura 2.2, el cual presenta las siguientes fases:

1. *Definición.* Seleccionar un conjunto de elementos que definen las funciones de hardware para la realización de una aplicación especifica.
2. *Descripción.* Describir un algoritmo para los requerimientos del hardware definido en la fase 1 para llevar a cabo la ejecuación de la aplicación.
3. *Evaluación.* Evaluar las operaciones contenidas en la definición y descripción. Si la ejecución del sistema no satisface los requerimientos de diseño, se procede a alterar las fases antes descritas mientras no se satisfagan.
4. *Síntesis.* Realizar la implementación del algoritmo en hardware (circuiteria).
5. *Pruebas.* Checar la ejecución del hardware para verificar las operaciones descritas en el algoritmo.

Empero, los conceptos, *diseño de sistemas y diseño lógico* son de gran importancia en el proceso de un sistema lógico. De donde el

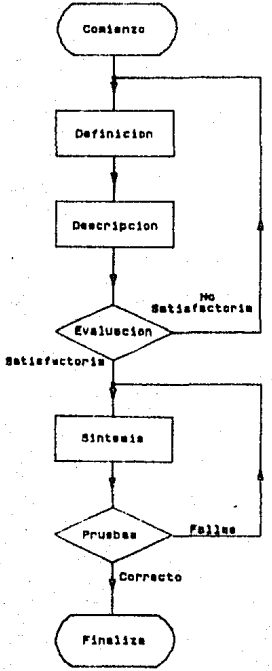


Figura 2.2 Fases para el proceso de diseño.

diseño de un sistema se define utilizando cajas negras, las cuales cumplen con una tarea en especifico, y los terminales para cada una de ellas, a lo antes descrito se le conoce como: *módulos funcionales*. El diseño de un sistema, depende del comportamiento del diseño lógico involucrado, siendo de gran importancia la detallada organización que se presente en cada módulo funcional.

Las tópicos involucrados en el proceso de sintetización de un módulo, se les conoce como: *definición, descripción y síntesis.*

Así, la definición se enfocada al diseño de sistemas y selección de terminales, por otro lado, la descripción muestra los detalles de las operaciones lógicas dentro de cada uno de los módulos empleados y finalmente la síntesis involucra el diseño y el hardware para llevar cabo la descripción.

II.3 Descripción de un proceso utilizando un algoritmo

La descripción fundamental de la operación que realiza un sistema lógico es, *el algoritmo.*

Para poder utilizar un algoritmo en una aplicación en especifico, se procede a definir los siguientes conceptos: *seudocodigo, diagrama de flujo y proceso de diseño.*

El algoritmo, describe en forma libre los pasos a seguir, conociendosele como *seudocodigo*, consistiendo esté de: un nombre del algoritmo, descripción del proposito del algoritmo y finalmente un listado de los pasos a seguir. La realización de los pasos nos lleva a cabo la ejecución de una tarea. Para ilustrar las fases antes descritas utilizaremos el siguiente ejemplo: el algoritmo E, dados dos números positivos enteros m y n buscar su común divisor:

Paso 1 (Buscar el residuo) Dividir m y n asignandole a r el residuo: donde (0 <= r <= n)

Paso 2 (Probar si es cero) terminar el algoritmo si r = 0 y

asignarle a n el valor de la respuesta.

Paso 3 (Realizar intercambio) Sustituye m con n y n con r y volver al Paso 1.

Como se aprecia, el algoritmo E, se puede sustituir por un diagrama de flujo que represente la secuencia de los pasos mostrados dentro del algoritmo, ver figura 2.3.

El diagrama de flujo y el seudocodigo, presentan información equivalente. La diferencia esencial dentro de un diagrama de flujo, es la existencia de la separación de la información que describe las operaciones, y la información que describe la secuencia. Sin embargo, ambas formas muestran como primer paso dividir m entre n y finaliza el algoritmo si el residuo es igual a cero en caso contrario, m es reemplazado por n y n es reemplazado por r. Si bien, se le puede llamar un proceso, un método, una técnica o una rutina, la palabra "algoritmo" denota más que un conjunto de reglas para dar una secuencia de operaciones para la solución de un problema. Un algoritmo posee cinco características distintivas:

1. *Finito.* Un algoritmo debe concluir después de un número finito de pasos, es decir, el número de pasos debe ser contabilizado.
2. *Definido.* Cada paso del algoritmo debe ser definido. Las acciones de las salida no deben ser ambiguas.
3. *Entradas.* Un algoritmo puede requerir ó no entradas, se pueden proporcionar variables antes de comenzar el algoritmo.
4. *Salidas.* Un algoritmo tiene una o varias salidas tipicas o también puede existir relación con las entradas.
5. *Eficiencia.* El algoritmo debe producir resultados eficientes, esto es, que los campos del algoritmo produscan resultados predecibles en la aplicación.

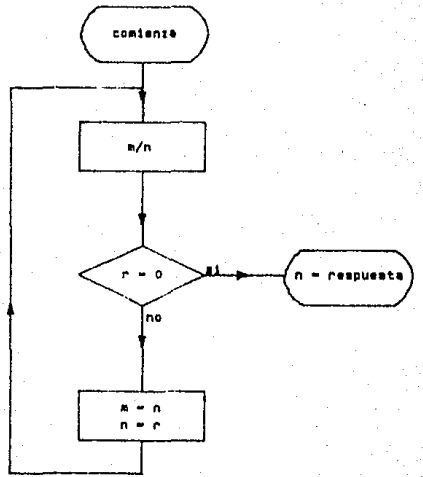


Figura 2.3 Diagrama de flujo del algoritmo E.

II.4 Máquina de estados

En el desarrollo de software para computadora, los diagramas de flujo son muy utilizados como medio de asistencia en el diseño, por otra parte, en el área de desarrollo de hardware digital, es muy empleado el diagrama de flujo dedicado al diseño de sistemas digitales conociendose en este campo como: Cartas ASM, de donde ASM es la abreviación de "*Algoritmo de Maquina de Estados*". Esta terminologia surgé del desarrollo en sistemas digitales como ya se mencionó, donde el diagrama de flujo describe la manera mediante la cual el sistema requiere moverse de estado a estado siguiendo un esquema de control ó algoritmo.

La notación dentro de la carta ASM es similar a la utilizada en el diseño de software para computadora. Las flechas son usadas para indicar la dirección de transición de estado a estado (notese que en un sistema sincrono la transición se lleva a cabo mediante los ciclos de reloj). En la figura 2.4 se muestran las cajas básicas para la elaboración de una carta.

Asi, la transición en un sistema de estados, se representa por la flecha entrando a la caja, la cual nos identifica al estado, en lo sucesivo se utilizara la notación "caja-estado". El nombre del estado se localiza en la parte superior izquierda de la caja, ver figura 2.4(b).

En diferentes sistemas, alguna transición de un estado presente a un estado siguiente, depende no solamente de la naturaleza del estado presente, sino del valor actual de una o más variables de entrada. Tipicamente, si la entrada satisface alguna condición especifica, el sistema puede pasar de un estado presente a un estado siguiente, si la condición no se satisface, el sistema puede moverse a algún otro estado de acuerdo al esquema de diseño.

Esta forma de "*camino condicional*" es representado por la caja-condición ver figura 2.4(a); la condición para moverse de un estado a otro, depende del contenido de la caja-condición, dicho

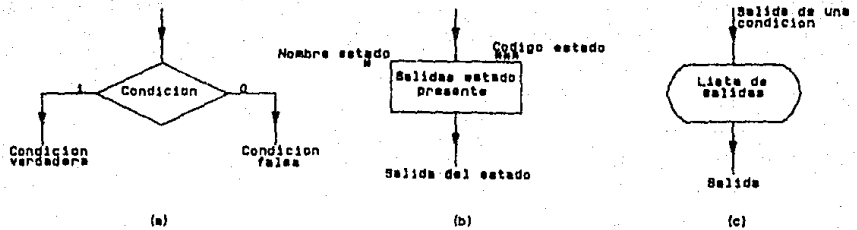


Figura 2.4 Cajas basicas para una carta ASM.
(a) Caja para condicion (b) Caja estado
(c) Caja para salida condicional

contenido debe ser una expresión booleana.

Algunas veces, cuando el sistema se encuentra en un estado en particular, esté requiere ejecutar ciertas salidas solamente si la condición especificada se satisface dentro de la ejecución del estado presente. Así, estas acciones son listadas en la "caja-salida-condicional" figura 2.4(c), que aparecen en la carta ASM como rectangulos con los bordes redondeados, de los anterior se observa que las salidas condicionales deben aparecer inmediatemente después de la caja-condición en un punto dado de la carta.

Todos los módulos de un sistema lógico se pueden representar por un modelo general llamado *máquina de estados*. Este modelo contiene los elementos necesarios para la descripción de un módulo en función de: entradas, salidas y el tiempo. En la figura 2.5, se muestran las tres diferentes fases que forman el modelo general de donde se puede identificar: *función estado siguiente*, *función estado presente* y *función de salida*.

El estado de máquina: es la memoria del estado presente para poder determinar el estado siguiente. El término *estado de máquina*, proporcionó la información suficiente para determinar las salidas y el estado siguiente en función de las entradas en el estado presente. La máquina, es semejante a una memoria, la cual generalmente es hecha de circuitos biestables llamados *Flip-Flops*.

Un grupo de Flip-Flops forman un estado, al cual se le conoce como: *registro-estado*. Se define un estado diferente para cada combinación de bits almacenados, de donde se pueden tener 2^n posibles estados para n registros-estado (Flip-Flops).

Los Flip-Flops del estado, son identificados con nombre de variables y se definen como *declaración de estado*. A cada variable se le puede dar un nombre arbitrario; A, B, ó 1H, etc. Un grupo de variables forman lo que se conoce como código de estado, por ejemplo: el estado DCBA en donde A, B, C y D son variables de estado, y además A=0, B=0, C=1, D=1 entonces el código de estado será 1100.

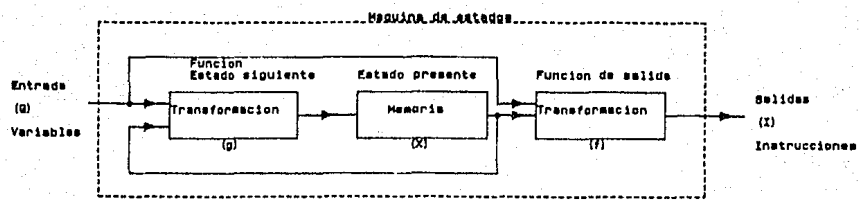


Figura 2.5 M O D E L O G E N E R A L.

Cada estado de máquina se encuentra relacionado con un estado siguiente, el cual es determinado por la función *estado-siguiente*. El estado presente es normalmente determinado por una entrada periódica en el *registro-estado*, así, al finalizar el estado presente tenemos un estado siguiente. La función *estado-siguiente* (g), ver figura 2.5, depende del estado presente (X) y las entradas ó variables (Q), si el estado presente es representado por T (variable en el tiempo discreto) y K es un número entero, entonces X(kT) representa el estado presente en tiempo discreto kT, por lo tanto, se puede definir a la función *estado-siguiente* (g) como:

$$X((k-1)) = g[X(kT), Q(kT)]$$

la notación para la operación estado-siguiente se simplifica notablemente utilizando el *operador-retardo* que es una flecha $\rightarrow$ ó $\leftarrow$ con lo cual se indica la transición hacia el estado siguiente, de esta manera la función estado-siguiente se define como:

$$X \leftarrow g[X, Q]$$

Así, el valor de X es alterado al finalizar el estado presente, donde X y Q son valores en dicho estado. Las modificaciones en el estado presente son mantenidas para poder determinar el uso del estado siguiente.

La *función-salida*, genera un conjunto de salida(s) o instruccion(es) (I) del estado y además la información de entrada requerida para cada estado, a semejanza con la función estado-siguiente, consiste de un operador de transformación llamado (f), definiendose como:

$$I(kT) = f[X(kT), Q(kT)]$$

donde k es un número entero y T es una variable en el tiempo discreto. La notación para la funcion-salida puede ser simplificada utilizando el símbolo " = " como operador *inmediato* que indica la operación en el estado presente, así, se presenta la expresión simplificada:

$$I = f[X, Q]$$

II.4.1 Algoritmo de máquina de estados

El algoritmo de máquina de estados, es una herramienta de gran utilidad dentro del área de diseño lógico, porque en ella se muestran simultaneamente un algoritmo y los estados de máquina, a esto se le conoce como Cartas ASM, dado que la carta es una descripción Fesquemática de la *función-salida* y la *función estado-siguiente* de una de máquina general (figura 2.5), es decir, se separa la fase conceptual del diseño para llevar a cabo la implementación de un circuito electrónico. Se le proporciona al diseñador la libertad para expresar los más complejos algoritmos para el desarrollo de un circuito. Cuando el algoritmo de la carta ASM satisface los requerimientos de diseño, se tiene la opción de implementar la carta utilizando algún tipo de familia lógica TTL ó dispositivos lógicos programables como memorias PROM ó arreglos lógicos programables.

En una carta ASM las operaciones se describen dentro de una caja, ver figura 2.4, así como otras fases asociadas a la caja-estado ejecutandose simultaneamente en el tiempo.

Dentro de esta técnica de diseño se utilizan las cajas básicas antes mencionadas y además el *bloque ASM*, el cual se describe en párrafos posteriores.

Un estado simple es indicado por una caja-estado, el cual contiene la lista de salidas que se deben realizar en el estado presente como se muestra en la figura 2.4(b), en donde la lista de salidas consiste de mnemónicos seleccionados por el diseñador para definir un conjunto de operaciones en dicho estado.

Dentro de la caja-desición, se evalua una expresión booleana, ahora bien, se debe proporcionar las variables de entrada (si son requeridas) que deben ser sensadas en el estado de máquina actual. En la figura 2.4(a) se muestran los atributos para éste elemento de la carta ASM como: dos salidas, las cuales son indicadas por "1" para condicion verdadera y "0" complemento. Las entradas son llamadas "variables" siendo sensadas para pasar a una salida condicional o

llevar a cabo una transición, teniendose así representada una relación lógica.

Por otra parte, dentro de las salidas condicionales como elementos de la carta ASM, describen las salidas que requieren que exista una variable o más, como condición para ejecutar la salida requerida en el estado presente de la máquina. Dicha estructura se muestra en la figura 2.4(c).

II.5 Bloque ASM

Un bloque ASM, es una estructura que consiste de una caja de estado, caja-desición y caja-salida-condicional, es importante mencionar que un bloque ASM nos define un estado presente.

Un bloque ASM puede tener varias salidas las cuales son proporcionadas por una estructura de cajas de desición como se muestra en la figura 2.6

Ahora bien, una carta ASM consiste de uno o más bloques ASM interconectados, de donde cada salida de un bloque debe ser conectado a un estado siguiente. Cada posible conexión de un estado presente a un estado siguiente es llamado "*camino de unión*" donde el camino de unión depende de la estructura que conforman las cajas de desición, siendo ahí donde se sensan las variables de entrada para así definir la conexión entre estados de la máquina.

Un bloque ASM, describe la operación de la maquina de estados durante un estado presente. Como se puede observar, cada bloque ASM representa: un estado presente (X), las salidas en el estado f[X], las salidas condicionales f[X, Q] y el estado siguiente g[X, Q] así como las entradas generales de un estado de máquina (Q).

Existen restrincciones obvias para llevar a cabo la interconexión de bloques en una carta ASM, las cuales se deben de considerar al momento de elaborar el diseño. Estas son:

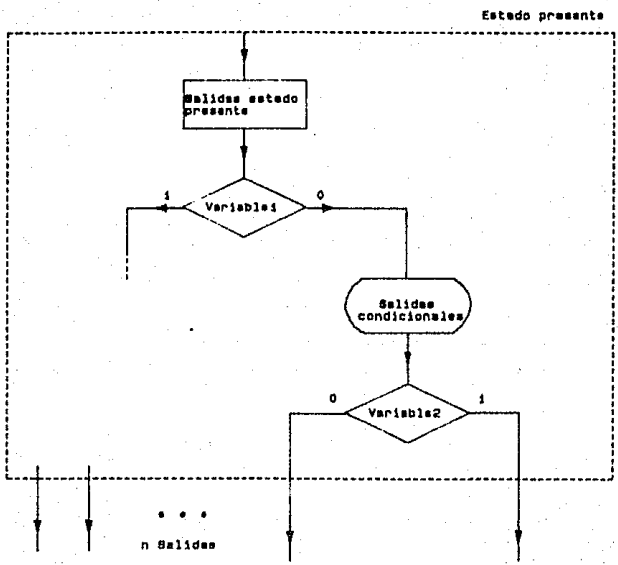


Figura 2.6 Bloque ASM.

- Se debe tener solo un estado siguiente y variables de entrada estables (sincronas) para cada estado presente.

- Si existe más de una caja de desición, de las cuales la salida al estado siguiente de cada una de ellas, pasa al mismo estado, esto no es valido, ya que el algoritmo por definición no puede sensar dos variables de entrada simultaneamente.

Finalmente, en la figura 2.7. se muestra un ejemplo de un bloque ASM no permitido, por otra parte, en la figura 2.8 se muestra un bloque equivalente.

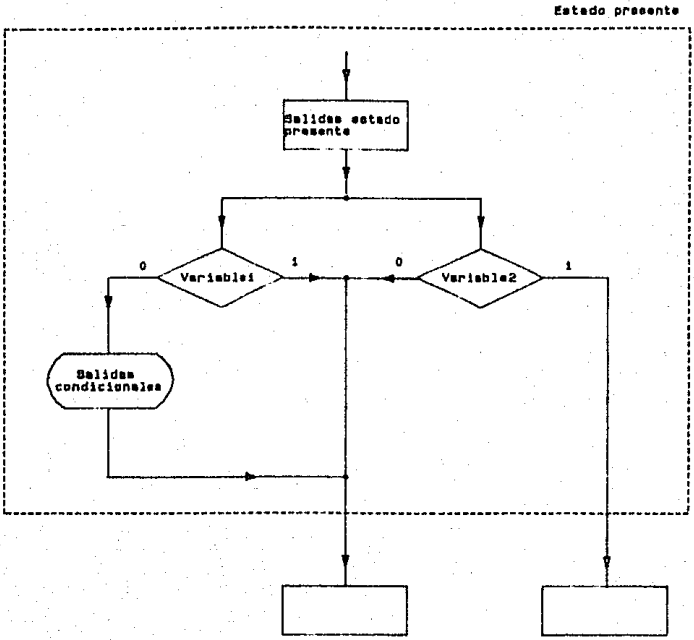


Figura 2.7 Bloque ASM no permitido.

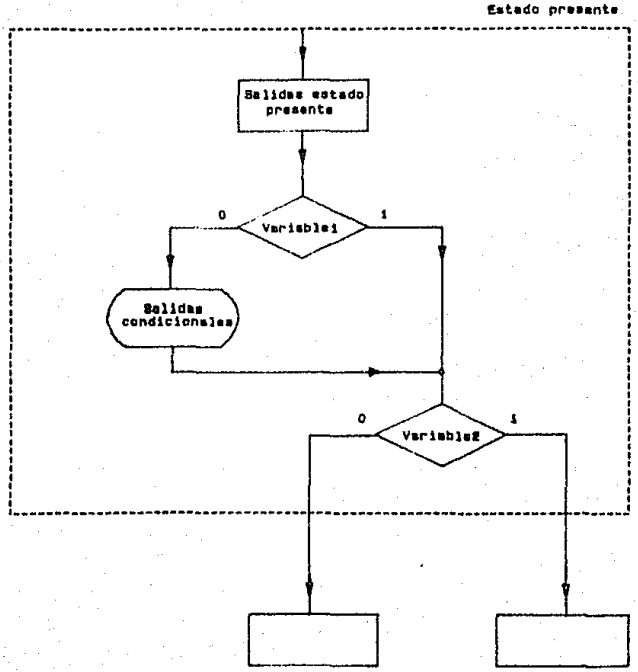


Figura 2.8 Bloque ASM equivalente.

CAPITULO III

AMBIENTE DE TRABAJO

AMBIENTE DE TRABAJO

III.1 Introducción

El presente capítulo, se encuentra orientado a la utilización del estandard gráfico PHIGS, basandose en la versión graPHIGS empleada en equipos IBM RT-PC. Esta versión proporciona el acceso a todas las facilidades que brinda el standard gráfico, además de algunas otras que son propias de graPHIGS. El desarrollo del software presentará en el capítulo IV, fue realizado en lenguaje de alto nivel C, bajo un ambiente AIX 2.1 y utilizando un equipo de trabajo RT-PC, por lo que la configuración empleada en las funciones del estandard gráfico , es particular para este tipo de equipo. Si se desea profundizar en algún tópico de graficación [15].

III.2 Estandard gráfico (PHIGS)

El estandard gráfico PHIGS (*Programmer's Hierarchical Interactive Graphics Standard*) constituye una interface de programación que soporta la definición, modificación y despliegue de datos gráficos organizados jerarquicamente, basado en las especificaciones propuestas por la American National Standards Institute (ANSI), para el desarrollo de aplicaciones gráficas interactivas en 2D y 3D.

La utilización de estandares gráficos permite una gran independencia entre la aplicación y los dispositivos físicos, brindando una gran portabilidad del software hacia diferentes tipos de hardware, conservando sus caracteristicas funcionales y evitando la necesidad de efectuar cambios drásticos.

PHIGS incorpora muchas de las facilidades de estandares como el Sistema Gráfico de Kernel (GKS) para el manejo de entidades gráficas, además de permitir un manejo independiente de las mismas por medio de unidades o segmentos de programa llamadas *estructuras*, con lo cual se tiene acceso a nuevas capacidades.

El manejo de estructuras permite el agrupamiento de elementos gráficos con sus caracteristicas y atributos particulares en bloques, que definen una entidad gráfica y que puede ser tratada como un solo elemento dentro de una aplicación. Esta característica proporciona grandes ventajas al desarrollar grandes sistemas, por la facilidad de utilizar pequeños bloques de información gráfica para formar sistemas complejos a partir de elementos simples. Así mismo, dentro de una estructura es posible invocar a otras estructuras formando árboles jerárquicos con elementos que son tratados en forma semejante a una subrutina. Debido a esto, la utilización de estructuras permite agregar o eliminar bloques de información en memoria en forma dinámica con lo cual se logra una mayor eficiencia en la utilización de recursos al permitir alterar solo la información necesaria.

Además, la edición de sus elementos, permite modificar el contenido de la estructura en el momento de estar interactuando con el modelo en tiempo real. Entre la variedad de recursos para la edición de una estructura, se cuenta con las siguientes caracteristicas: eliminación e inserción de nuevas primitivas gráficas, cambio de atributos, copia o ejecución de una estructura dentro de otra, etc., de manera similar al manejo de un editor de texto convencional.

Un estandard gráfico permite tener un interfaz entre la aplicación desarrollada por el usuario y los dispositivos físicos de graficación con que se cuenta (teclados, monitor, ratón, tableta digitalizadora, etc). Cada uno de los cuales es considerado una estación de trabajo con la característica de ser tomados en cuenta como elementos del sistema sin necesidad de conocer a fondo sus características individuales para su programación, puesto que el estandard se encarga de habilitar dichos dispositivos de control. Los dispositivos físicos asociados a una estación de trabajo pueden ser clasificados en tres

categorías, de acuerdo a las características que presentan:

- De entrada.
- De salida.
- De entrada-salida.

Las características físicas de cada dispositivo determinan su categoría. Por ejemplo, un teclado o un digitalizador solamente son capaces de proporcionar datos de entrada; mientras que una impresora o un graficador solo proporcionan salidas. Por otra parte, un monitor posee la capacidad de brindar facilidades de interacción con el usuario, para permitir tanto entradas como salidas, a través de una pluma óptica, o de un pick (utilización del cursor para seleccionar algún elemento gráfico en el monitor).

Así mismo, PHIGS permite un manejo independiente del almacenamiento y asignación de información gráfica hacia cada dispositivo, evitando la redundancia y el manejo de toda o parte de la información en cada estación de trabajo definida. Esto proporciona una gran flexibilidad al desarrollar una aplicación interactiva, dado que constantemente es necesario efectuar accesos a información gráfica, dependiendo de los requerimientos particulares.

El grupo de funciones disponibles en el estandard gráfico (en los sucesivo graPHIGS) puede ser accesado desde un lenguaje de alto nivel *p.ej.* PASCAL, C, o FORTRAN, simplemente declarandolas como funciones externas y ejecutandolas desde el programa con sus respectivos parámetros. El acceso a éste grupo de funciones se logra solamente después de haber habilitado el sistema mediante la ejecución de la rutina GPOPPH(), con lo cual, graPHIGS efectua la inicialización de las listas de estados y genera una serie de tablas con las características del hardware y software, así como información a cerca del control de la estación de trabajo y el estado que presenta la aplicación, además las características de las estructuras de datos empleadas como parámetros de las funciones de graPHIGS. La información proporcionada mediante éstas tablas y listas, permite el desarrollo de aplicaciones con capacidades de configuración dinámica adaptable a

las características específicas del hardware y software que se utiliza, permitiendo de esta manera una mayor portabilidad.

Por otra parte, existen rutinas que proporciona el estandard que permite la apertura de estaciones de trabajo tanto físicas como lógicas, de esta manera la información gráfica puede ser procesada en cada una de ellas en forma independiente.

Las listas y tablas generadas permiten realizar consultas con respecto al estado actual del sistema:

- *Lista de Estados de graPHIGS (PSL).* Contiene los parámetros dinámicos del sistema, por medio de los cuales queda determinado el modo de operación de graPHIGS, tales como: abrir el sistema, estado de las vistas, transformaciones aplicadas sobre las vistas, dimensión de las ventanas y puertos, etc. (independientemente de la estación de trabajo utilizada).

- *Lista de Estados de la Estación de Trabajo (WSL).* Contiene información del estado que guarda cada estación de trabajo. Esta información indica las características y el modo de operación del dispositivo, así como las vistas que estan asociadas a el (cada estación de trabajo tiene su propia lista y puede ser checada por medio de las funciones proporcionadas por el estandard).

- *Lista de Estados de la Estructura (SSL).* Se guarda la información de cada estructura (entidad gráfica independiente) definida durante la ejecución de la aplicación, indicando la forma en que será accesada y las características de los elementos que la forman, así como las transformaciones y atributos asociados a ella.

- *Lista de Estados de Error (ERSL).* Proporciona información de los errores ocurridos durante la ejecución de la aplicación y permite hacer un seguimiento de los casos en que se presentan, a través de una rutina dedicada para tál efecto.

- *Lista de Estados de Utileria (USL).* Contiene información sobre los parámetros utilizados en las funciones de utileria, las cuales permiten la creación de matrices de vista.

Para poder utilizar las capacidades que ofrece graPHIGS, el sistema proporciona dos grupos de tablas, en las cuales describen las caracteristicas generales de cada tipo de estación de trabajo soportada y las estructuras de datos, que deben ser empleadas en los parámetros de las funciones asociadas a cada uno de ellos. Estas tablas son las siguientes:

- *Tablas de Descripción de graPHIGS (PDT).* Contiene la información que describe las capacidades y caracteristicas generales.

- *Tablas de Descripción de la Estación de Trabajo (WDT's).* Estas tablas contienen información sobre la inicialización y configuración de cada estación de trabajo.

Una véz abierto (cuando se tiene acceso a graPHIGS) el sistema, es posible crear y modificar los datos relacionados con la aplicación sin necesidad de abrir alguna estación de trabajo especifica. La información estará en memoria principal o en algún medio de almacenamiento, pero no será visible hasta que sea asociada a una estación de trabajo para su despliegue en video o su impresión en graficador. Esto se logra mediante la llamada a la funcion GPOPWS (abre estación de trabajo), con lo cual se cargan las funciones que efectuan lo siguiente:

- Se construye e inicializa la WSL con las caracteristicas y capacidades de la estación de trabajo indicada en los parámetros de la función:
  a) Capacidades de color de la estación.
  b) Dispositivos de entrada disponibles.
  c) Tamaño de la paleta de color.

- Se establece una conexión fisica con el dispositivo.
- Limpia la superficie de despliegue si es necesario.

Para finalizar con la aplicación, es necesario cerrar la estación de trabajo abierta mediante la función GPCLWS (cierra estación de trabajo), con lo cual se libera la estación de trabajo, así como la lista de estados asociada a ella y se vacía la cola de eventos del dispositivo. Además de esto, se requiere cerrar el sistema mediante el empleo de la función GPCLPH (cierra graPHIGS). El empleo de esta función, cierra cada una de las estaciones de trabajo abiertas durante la ejecución de la aplicación; sin necesidad de declararlo explícitamente, además de liberar las listas de estados y tablas de descripción asociadas a dichas estaciones de trabajo, así como los recursos asignados a las mismas, por lo que se puede prescindir de hacerlo directamente y solo ejecutar GPCLPH() para terminar con la ejecución del programa.

III.2.1 Funciones de graPHGIS

Estas funciones se clasifican dentro de las siguientes categorías:

Funciones de control. Mediante este tipo de funciones es posible efectuar el acceso y control de las estaciónes de trabajo y del sistema. Para esto, se proporcionan funciones que permiten abrir alguna estación de trabajo, abrir el sistema, actualizar los datos gráficos asociados a la estación, enviar mensajes o simplemente controlar el seguimiento de la ejecución del programa, así como también se incluyen funciones para cerrar las estaciones de trabajo y el sistema.

Funciones primitivas de salida. Las funciones incluidas dentro de éste grupo permiten la creación de entidades gráficas elementales como: líneas, polígonos, elipses, círculos, arcos (circulares y elípticos), pixels y los diferentes tipos de texto. Algunas de estas primitivas pueden ser utilizadas en ambientes 2D y 3D, mientras que otras solo pueden ser empleadas en 2D. Todas las funciones de este grupo deben ser empleadas como elementos dentro de una estructura para poder ser manipuladas y desplegadas, puesto que para su ejecución y despliegue se debe hacer referencia a el nombre de la estructura a la

que están asociadas dichas primitivas.

Las primitivas de graPHIGS pueden ser agrupados en *clases* para proporcionarles atributos de:

- Detectabilidad
- Luminisencia
- Visibilidad

Así también, existen funciones que permiten agregar filtros para determinar que primitivas gráficas son identificables:

- Filtro para detectabilidad
- Filtro para luminusencia
- Filtro para visibilidad

Por otro lado, una primitiva de salida es habilitada si cumple con lo siguiente:

- Si es miembro de una clase especifica en un correspondiente filtro de inclusión.
- Si sus clases no estan contenidas en un filtro de exclusión correspondiente.

Finalmente, al momento de inicializar graPHIGS los filtros estan vacios y:

- las primitivas no son detectables
- las primitivas si se iluminan
- las primitivas no son invisibles

Funciones de atributos. Este tipo de funciones permite especificar la peculiaridad de las primitivas de salida como: tamaño, forma, estilo y color. Para su utilización se requiere declararlas dentro de una estructura abierta, puesto que se crean como elementos dentro de la misma. Al ejecutar una estructura, cada uno de estas funciones altera las primitivas de salida declaradas.

Los valores de los atributos pueden ser declarados de manera individual o ser referenciados a través de un índice hacia una tabla en la WSL. Para determinar la forma en que serán accesados estos valores, se utiliza una bandera de fuente de atributos (ASF). Si durante la ejecución del programa, se hace referencia a un valor fuera del rango soportado por la estación de trabajo, el sistema utiliza por "*default*" el valor 1 [9].

Funciones para inicialización de tablas de estaciones de trabajo. Cuando el sistema es abierto, las tablas adquieren valores iniciales proporcionados por el sistema, los cuales pueden ser modificados de acuerdo a las necesidades del usuario.

Mediante este grupo de funciones es posible asignar valores a las tablas de las estaciónes de trabajo para determinar las características de las primitivas de salida.

Entre las peculiaridades que pueden ser alteradas mediante el empleo de estas funciones se encuentran: tipo de texto fuente, modelo de color (RGB, HSV, CMY), tipo de representación de aristas, filtro de visibilidad de primitivas detectables, tipo de rellenado de polígonos, tipo de marcas a utilizar, presentación de texto en video, características y prioridad de las vistas, etc.

Funciones para edición de estructuras. Permiten manipular estructuras; así como su contenido mediante las siguientes operaciónes:

- Crear y/o borrar una estructura o grupo de estructuras.
- Crear estructuras jerárquicas.
- Proporcionar los elementos para crear estructuras dinámicas.
- Editar el contenido de una estructura.
- Ejecutar estructuras

Existe un apuntador a los elementos de la estructura que normalmente se encuentra apuntando hacia el último elemento agregado a la estructura, por lo que si se desea modificar alguno de los

elementos de la estructura, es necesario apuntar a él. Si a una estructura se le agrega un nuevo elemento, éste será colocado en la posición actual del apuntador.

Así, las estructuras permiten definir, organizar y controlar las partes de una aplicación gráfica, esto es:

- Primitivas de salida
- Asignación de atributos
- Transformaciones elementales
- Invocar subestructuras
- Poner etiquetas a elementos gráficos
- Información del dispositivo pick.

Es importante mencionar que, una estructura es una agrupación de elementos que permiten en conjunto, definir una entidad gráfica como un elemento de la aplicación.

Funciones de entrada. Este grupo de funciones, permite obtener datos de cualquier dispositivo de entrada asociado a la estación de trabajo, permitiendo la interacción entre una aplicación y el usuario. Existen seis tipos de dispositivos lógicos de entrada: *String, Valuator, Locator, Stroke, Pick y Choice*. Cada uno de los cuales puede ser programado para funcionar en alguno de los siguientes tres modos de interacción: *Sample, Request o Event*.

- Modo request: Para la utilización del dispositivo en este modo, la aplicación requiere de la función GPRQCH(). Al momento de detectar dicha instrucción la aplicación, la ejecución se detiene hasta que se obtenga información:
  - estado (ninguno = 1, verdadera = 2)
  - número de tecla oprimida
- Modo sample: Para realizar la obtención de información en dicho modo, se emplea la siguientes funciones de graPHICS; GPSMCH(), GPSMPK(), GPSMLC(), GPSMST(). En éste modo, el dispositivo envia información de manera

constante independientemente de si se oprime o no una tecla.

- Modo event: Para el empleo de este modo, es requerida la utilización previa de la función de graPHIGS; GPAWEV(), con la cual se obtiene la clase de dispositivo que envía información.

Mediante la utilización de las funciones de entrada, se pueden efectuar las siguientes acciones:

- Inicialización de los dispositivos de entrada.
- Determinación del modo de operación del dispositivo.
- Recuperación de datos desde el dispositivo.
- Verificar si existe un evento desde algún dispositivo.
- Manejo de la cola de eventos.
- Utilización de los valores almacenados en la cola de eventos.
- Ajuste de las características del eco asociado al dispositivo.

Las atributos de cada dispositivo pueden ser obtenidas mediante las *funciones de requerimiento* proporcionadas por graPHIGS.

PHIGS soporta seis clases de dispositivos de entrada, los cuales pueden ser programadas de acuerdo a sus características y las necesidades de la aplicación; de los seis que se enlistan solo los tres primeros son utilados por el sistema *AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADO A PAL's.*

Clases de dispositivos de entrada

- *Locator* (ratón y tablero)
- *Choice* (lpfk, pfk)
- *Pick* (teclas del ratón)
- *String* (teclado)
- *Stroke* (ratón, pluma óptica)
- *Valuator* (diales)

Cada dispositivo tiene un eco asociado, mediante el cual es representado en la pantalla. Así mismo tiene asociado un área de eco de la cual obtiene y envía su información. Debido a esto, cuando un dispositivo es utilizado, puede requerir ser inicializado para que exista un mapeo entre la superficie de acción y la superficie de despliegue del dispositivo.

Dispositivo Choice
- LPFK (Lighted Program Function Keyboard)
- PFK (Program Function Keyboard)
- String (Teclado alfanumerico)

La utilización de funciónes para el dispositivo, proporcionan un número entero positivo que corresponde al número de tecla oprimida, mientras que si no se oprime una tecla retorna un cero. Además, el modo de operación de un dispositivo puede ser establecido mediante la función GPCHMO().

Para la inicialización del dispositivo *choice* se requiere que el dispositivo se encuentre en modo *request* (no activo). Al momento de inicializar el dispositivo se le puede asignar los siguientes atributos:

- Tipo de dispositivo utilizado.
- Si utiliza área de eco o no.
- Area de detección o superficie de despliege.
- Longitud del registro de datos.
- Registro de datos (teclas empleadas).

además, para obtener información del dispositivo choice se debe tener en consideración el modo de operación para utilizar ya sea la función GPRQCH(), GPSMCH() o GPQTCH().

Dispositivo locator (ratón y tableta digitalizadora)

Este dispositivo regresa una posición en coordenadas del mundo en 2D. Mientras que en la pantalla se manejan coordenadas de video; por

tal motivo es necesario realizar un mapeo -Transformación Ventana-Puerto-.

Para establecer el modo de operación del dispositivo locator, ya sea: request, sample o event; se debe utilizar la instrucción GPLCMO()

Previamente se debio de haber inicializado el dispositivo mediante la función GPINLC(), para definir los siguientes parámetros:

- Indice asociado a la vista
- Posición inicial (coordenas de mundo)
- Tipo de eco-prompt, pudiendo ser:
  - Pequeñas lineas cruzadas
  - Lineas cruzadas del tamaño del video
  - Linea que une un punto inicial con el final
  - Un rectangulo
  - Estructura predefinida
- Area de eco (coordenas de dispositivo)
- longitud del registro de datos utilizados
- Registro de datos

Es importante mencionar, que para obtener datos del dispositivo locator se debe tomar en cuenta su modo de operación y dependiendo de este se utilizará ya sea la función: GPRQLC(), GPSMLC() o GPGTLC().

Dispositivo Pick

El dispositivo pick obtiene el identificador y la posición de una primitiva, que forma parte de una estructura, así como el identificador de la estructura, esta información solo se puede obtener si se activa el filtro de detección del pick (identificadores de primitivas), además de que las primitivas deben estar presentes en video para ser detectables mediante el uso de un ratón y éste dispositivo, anteriormente debe ser inicializado con la función GPINPK() y definir el modo de operación de éste para poder definir cual de las siguientes funciones se utilizara para obtener dicha información: GPGTPK(), GPSMPK(), GPRQPK().

Funciones de despliegue. El grupo de funciones de despliegue permiten efectuar:

- Asociar una estructura a una vista o una estación de trabajo.
- Desasociar una estructura de una vista o estación de trabajo.
- Inicialización de una o todas las vistas.

La información gráfica generada durante la ejecución de una aplicación, debe ser asociada a una vista para poder ser desplegada en video; a su vez, una estación de trabajo puede tener asociada una o más vistas de manera totalmente independiente, permitiendo así desplegar en video parte de la información gráfica y al mismo tiempo enviar a graficador otra parte de la información; requiriendo solamente asignar a cada estación de trabajo las vistas que contengan la información correspondiente.

Funciones de transformación. graPHIGS permite el manejo de tres tipos de categorías de funciones de transformación: *de modelado, de vista y de estación de trabajo.*

Las transformaciones de modelado solo pueden ser aplicadas a las primitivas que constituyen una estructura, modificando los valores de transformación utilizados por el sistema durante la ejecución de las funciones gráficas.

Las transformaciónes de vista permiten modificar la tabla asociada a cada vista en una estación de trabajo.

Finalmente, las transformaciónes de estación de trabajo permiten la modificación del mapeo de las coordenadas normalizadas de proyección (NPC) a coordenadas de dispositivo (DC) para una estación de trabajo en específico.

Mapeo de NPC a dispositivo de salida

En graPHIGS, las estructuras creadas pueden ser asociadas a un vista definida por medio de la función GPARV(). Esto es útil cuando se tienen elementos gráficos de diversas categorías y se desea manejarlos como elementos.

Por otra parte, La superficie de despliegue de video utilizá coordenadas normalizadas de proyección, esto es, sus coordenadas son de -1 a 1 en x, y, z; mientras que los dispositivos de entrada utilizan un sistema de coordenas en VCS (Sistema de Coordenas de Vista) cuyo rango para los tres ejes es de -0.28448 a 0.28448 metros, por lo que es necesario efectuar un mapeo (ventana-puerto) entre la superficie de despliegue y el área de trabajo de los dispositivos.

El mapeo se logra utilizando las funciones GPVMT2() o GPVMT3(), según se requiera. Estas funciones son utilizadas además para efectuar traslaciones, rotaciones y otras transformaciones en general.

Una vez realizadas las transformaciones adecuadas, deben ser mapeadas al puerto de vista, definiendo lo siguiente:

- Borde del puerto.
- Parte visible al observador (clipping).
- Color del fondo del video.
- Prioridad de una vista.

lograndose lo anterior mediante las funciones GPVMP2() o GPVMP3() y GPVCH().

Funciones de utileria. Las funciones de este grupo, se utilizan para efectuar cálculos y modificar datos para ser empleados en las transformaciones aplicadas a modelos. Algunas de las funciones ejecutan transformaciónes requiriendo matrices, mientras que otras permiten alterar las caracteristicas de una vista en el sistema coordenado de vista (VCS).

Funciones para manejo de errores. Permiten definir un medio para el manejo de errores; graPHIGS utiliza un caracter de default para el control del manejo de errores, que permanece en estado ON. Si se desea cambiar su estado a OFF, el sistema proporciona una función mediante la cual se logra que el proceso continue hasta que se encuentre una condición (Para mayor información sobre el manejo de errores [14].

Funciones de consulta. Proporciona información sobre los siguientes aspectos:

- Características básicas del sistema.
- Estado actual del sistema.
- Características básicas de la estación de trabajo.
- Estado actual de la estación de trabajo.
- Configuración de la estación de trabajo.
- Estructuras y su interrelación.
- Contenido de cada estructura.
- Estado de errores y contenido de los mensajes de error.

de las cuales exiten tres tipos de funciones de consulta; las orientadas a accesar información contenida en la lista de estados de la estación de trabajo (WSL), las que proporcionan información contenida en la tabla de datos de la estación de trabajo (WDT) y las de propósito general que proporcionan información independiente de una estación de trabajo.

III.2.2 Control de acceso a graPHIGS

PHIGS permite la utilización de sus funciones y todas sus capacidades solamente después de haber habilitado el sistema; esto se logra al ejecutar la función GPOPPH (abre graPHIGS), creandose las listas PSL, ERSL, y USL, así como las tablas PDT y WDT, quedando está información en disponibilidad para su utilización.

La función GPOPPH() acepta dos parámetros los cuales deben ser proporcionados por el usuario. El primer parámetro es de tipo string

con una longitud de 8 bytes y debe ser el nombre del archivo al que serán enviados los mensajes de error que se hayan generados por un manejo equivocado de las funciones de graPHIGS durante la ejecución de la aplicación -El nombre del archivo debe ser declarado llenando los 8 espacios del campo para que pueda ser aceptado por el sistema, por ejemplo: "sysprint"-. El segundo parámetro es de tipo entero y permite modificar las características de la estación de trabajo y del sistema. Si se desea utilizar las características iniciales del sistema y la estación de trabajo; su valor debe ser 0, si se desea hacer alguna modificación, consultese [8], apendice A "Defaults and Nicknames" para ver su contenido y formatos.

Una véz abierto el sistema, la aplicación puede manipular todas las primitivas o funciones de control del estandard sin necesidad de asociar la información a algun dispositivo de entrada o salida, simplemente puede ser alterada o preparada y posteriormente, utilizada para el despliegue de información gráfica. PHIGS considera a cada dispositivo como una estación de trabajo y para su manipulación, se le asocia un identificador, el cuál debe ser soportado por el sistema de acuerdo a la información contenida en la tabla WDT. Para obtener una lista de las peculiaridades de WDT y de la tabla WSL propia de la estación utilizada, se debe emplear la función GPQWCV().

Para llevar a cabo el acceso a alguna estación de trabajo, se hace uso de la rutina GPOPWS (abre estación de trabajo), la cuál habilita la estación referenciada, dejandola en disponibilidad para su utilización, después de cargar previamente la lista de estados de la estación de trabajo (WSL). La información contenida en esta lista es obtenida del sistema operativo (AIX 2.1) al establecer la conexión especificada incluyendo: capacidades de color, tamaño de la tabla de colores y dispositivos de entrada disponibles para la estación de trabajo definida, así como también se establece una conexión física entre el sistema operativo y la estación, limpiando la superficie de despliegue cuando sea necesario.

Para poder realizar lo antes descrito, se requiere de la función GPOPWS(), a la cual se le debe proporcionar los parámetros:

identificador de la estación de trabajo, identificador de conexión con el dispositivo físico y tipo de estación de trabajo respectivamente. El parámetro identificador de estación de trabajo, debe ser de tipo entero, con el cual se identifica la estación de trabajo hacia la cual se desea enviar información. Los otros parámetros deben ser de tipo string con longitud de 8 bytes; el identificador de conexión con el dispositivo físico, indica el dispositivo que será accesado. Por ejemplo un display o un graficador.

Para una estación de trabajo RT-PC, el identificador de conexión con dispositivo físico se le debe asignar "console ". Si la información se desea enviar a un graficador, la conexión con el dispositivo físico se puede declarar con cualquier identificador (menor de ocho caracteres). En cualquier caso, se deben proporcionar los 8 campos de la variable, llenando con espacios vacios cuando sea necesario.

El parámetro tipo de estación de trabajo, debe tener asignado un identificador que sea soportado por el sistema. Para el caso del equipo RT-PC, el identificador de la estación de trabajo es "NATIVE ", mientras que para un graficador se utiliza "GDF ".

Para cerrar alguna estación de trabajo abierta durante la ejecución del programa y liberar los recursos asignados a ella, se utiliza la función GPCLWS(). Esta función emplea un parámetro de tipo entero, que debe coincidir con el valor de la estación de trabajo que se desea cerrar. Al cerrar una estación de trabajo, se liberan los recursos utilizados por ella y de esta manera se recuperan las áreas de memoria utilizadas por las listas y tablas de información gráfica asociada a la estación, evitando que el sistema se sature y disminuya su tiempo de respuesta. Esto es muy útil al manejar grandes bloques de información que solo es empleada durante periodos muy cortos; como en el caso de un graficador, en el cual dicha información no requiere estar todo el tiempo presente en la estación, sino solo al momento de generar el archivo para impresión.

Al ejecutar la función GPCLWS(), el sistema desecha la lista de estados de la estación de trabajo y su identificador es borrado de la PSL (lista de estados del sistema), así como la conexión establecida al abrir la estación y la cola de eventos de los dispositivos de entrada asignados a esta estación son liberados.

Para cerrar el acceso al sistema, se utiliza la función GPCLPH(), la cual no utiliza parámetros, permitiendo terminar con todos los procesos gráficos, además de cerrar todas las estaciónes de trabajo abiertas durante la ejecución de la aplicación. Con esta función, todos los buffers son liberados y los archivos del sistema son cerrados, además quedan en disponibilidad la lista PSL y la tabla WDT.

III.3 Consideraciones AIX - C - graPHIGS

Debido a que C es un lenguaje de alto nivel que utiliza sus parámetros por valor y además graPHIGS asume siempre que sus parámetros son pasados por referencia, es necesario incluir el archivo graPHIGS.h dentro de la aplicación que se este realizando.

Los parámetros que devuelven datos deben estar precedidos por el caracter &, como se muestra en la siguiente función GPGTST(longitud,&número,cadena); de la cual las variables número y cadena contienen valores que regresa la función.

compilación C - graPHIGS

AIX 2.1 soporta simultaneamente el proceso de compilación y ligado de aplicaciones, para lo cual se emplea el comando:

```
cc -f programa.c/usr/lib/graPHIGS.a -lfm
```

si la aplicación no tiene errores, se genera un archivo ejecutable identificado con el nombre a.out, y los errores que se generan al momento de ejecutar una aplicación que emplea rutinas de graPHIGS, son enviados a un archivo identificado con el nombre de sysprint por

ejemplo, en donde se pueden detectar las funciones de graPHIGS que provocan la anomalía [14].

III.4 Características de Hardware y software empleado

Equipo	: IBM RT-PC.
Monitor	: 1024 x 1024 pixeles de resolución.
Procesador 6150	: Procesador Motorola 68020 y tres procesadores en paralelo de punto flotante.
Dispositos de control	: Teclado de funciones programadas (LPKF), Tableta digitalizadora y ratón.
Disco duro	: 2 discos duros de 70 Mb cada uno.
Tarjeta gráfica	: Megapel.
Memoria RAM	: 12 Mb.
Estación de trabajo	: 5081
Sistema operativo	: AIX 2.1.
Estandar gráfico	: graPHIGS, versión 3.4.
Lenguaje de alto nivel	: C estandard.

En la figura 3.1 se ilustra la configuración básica, empleada para el desarrollo del sistema *AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADO A PAL's*.

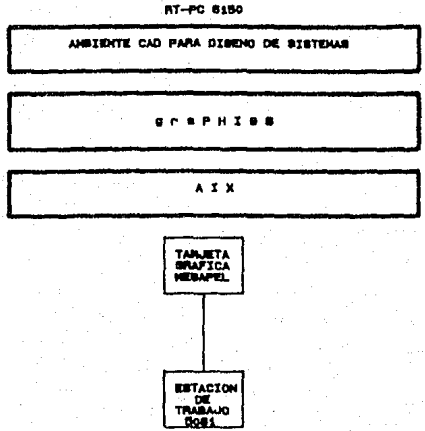


Figure 3.1 Configuracion basica.

CAPITULO IV

DESCRIPCION DEL SISTEMA

DESCRIPCION DEL SISTEMA

IV.1 Introducción

La característica principal del sistema es la utilización de estaciones de trabajo para Diseño Asistido por Computadora (CAD) las cuales se emplearon para desarrollar la aplicación enfocada al área electrónica.

El sistema esta desarrollado de tal manera que el usuario no requiera de conocimientos previos de computación, ya que se cuenta con una serie de menús para la elaboración de las cartas ASM (Algoritmo de máquina de estados), la cual es una técnica enfocada al diseño electrónico, donde se describen las secuencias o pasos, para elaborar una aplicación de diseño, y mediante una serie de manipulaciones de los atributos no gráficos obtener las funciones booleanas.

Dentro del menú se ofrecen los iconos básicos para el diseño de la carta ASM; además se cuenta con acercamientos, traslaciones, borrado de elementos, limpiar el área de despliegue, etc. tomando en cuenta que podemos requerir de tales herramientas al momento del diseño. Además si se requiere se puede almacenar el archivo de trabajo de la carta ASM en diskette o realizar una impresión.

Se decidió integrar algunos elementos de un ambiente CAD como auxilio en las fases iniciales de agregación de información gráfica, de la cual, el sistema, extraera los atributos no gráficos de la misma y los procesara. Las acciones interactivas se llevan a cabo empleando una estación de diseño RTPC-NATIVE y procesador 6150, bajo un ambiente AIX 2.1 con base en el lenguaje de alto nivel C y empleando el estandard graPHIGS (para mayor detalle capitulo III) como elemento de control de la estación de diseño.

De la carta ASM se mapeará a la generación de las funciones booleanas; dentro de dicho proceso se analizará y si es posible, se lleva a cabo un proceso de optimización que incida sobre las funciones booleanas; mediante las cuales se genera un archivo, conteniendo el mapa de fusibles en formato JEDEC (dicho formato posee opciones que son utilizadas para realizar la transferencia de datos entre el sistema de desarrollo para dispositivos logicos programables y el dispositivo programador) para su posterior utilización en la quema de fusibles del dispositivo PAL (ver figura 4.1).

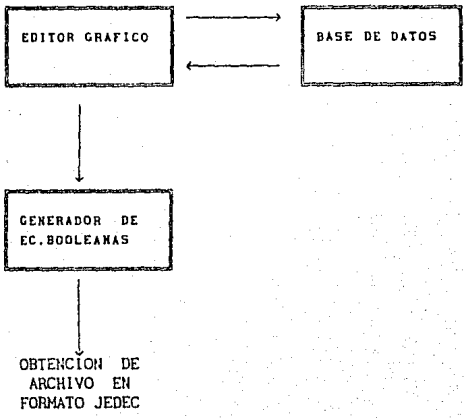


figura 4.1 Diagrama de bloques

IV.2 Panorama general del sistema

Como se mencionó en capitulos anteriores, el sistema utiliza tecnologia de vanguardia, empero, la operación de equipo es muy similar a las computadoras personales.

El sistema presenta un ambiente de trabajo de fácil manipulación, dado que se cuenta con una serie de menús. La manera de seleccionar una opción, se puede realizar de la siguiente forma: utilizando un "*Ratón*" ó "*Teclado de funciones*" (para mayor detalle capitulo III). En el módulo del editor gráfico las opciones de entrada, utilizan un conjunto de *iconos*, -simbolos gráficos que se parecen a la opción de procesamiento que deben representar-. La ventaja de la utilización de iconos, es que se puede ocupar menos espacio en video que la descripción textual correspondiente de las funciones y además son fáciles de entender.

A continuación se describen la diferentes funciones que realizan las teclas del "*Ratón*" cuya representación aparece en la parte inferior izquierda de la pantalla al momento de activar el editor:

- TECLA 1: Selecciona una opción; posteriormente se utiliza para indicar y repetir donde queremos colocar los diferentes iconos.

- TECLA 2: Borra un icono, inmediatamente después de haberlo creado.

- TECLA 3: Realizá acercamientos ó alejamientos totales de lo contenido en pantalla, como facilidad para la elaboración de la carta ASM.

- TECLA 4: Rechaza o concluye la ejecución de una opción elegida.

Se debe tener en consideración la función que realiza cada una de las teclas del ratón, ya en lo posterior solo se mencionará solamente el número de tecla que debe emplearse.

El menú principal del sistema nos presenta las siguientes opciones: DIRECTORIO TABLA DE ESTADOS HERRAMIENTAS SALIR DE SESION ilustrandose su esquema en la figura 4.2

IV.2.1 Menú principal

IV.2.1.1 Directorio: Nos muestra los archivos contenidos en el directorio de trabajo del *AMBIENTE CAD PARA DISENO DE SISTEMAS DIGITALES ORIENTADO A PAL's* de donde podemos elegir algún archivo existente para realizar modificaciones.

IV.2.1.2 Generación de funciones booleanas: Esta opción genera las funciones booleanas de la carta ASM previamente editada, ver figura 4.2.1. Las funciones obtenidas y que presenta el sistema son: *estado siguiente, salida en estado presente y salida condicional.*

Se puede mencionar que, para la obtención de las funciones booleanas, se parte de la tabla de verdad que se genera a partir de la información no gráfica contenida en la carta ASM. Posteriormente se realizan una serie de manipulación de las estructuras de información, obteniendo las funciones boolenas optimizadas y que son presentadas por el sistema con un formato de fácil acceso por el programa *PLAN.*

En la actualidad existen diferentes métodos para la reducción de funciones booleanas que son empleados para el diseño electrónico. Mapas de Karnough y método de Quine-McCluskey. Es importante mencionar que no son las únicas técnicas que existen, empero, para fines del desarrollo del sistema, se mencionan y se lleva a cabo una comparación funcional entre las más populares.

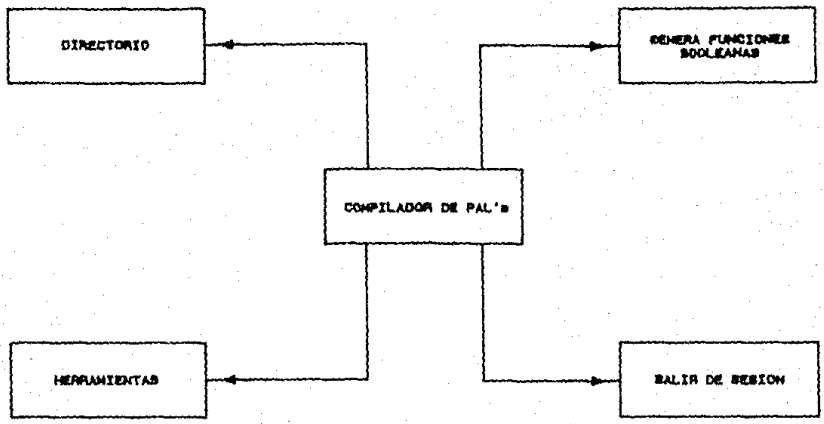


Figura 4.2 Menu principal

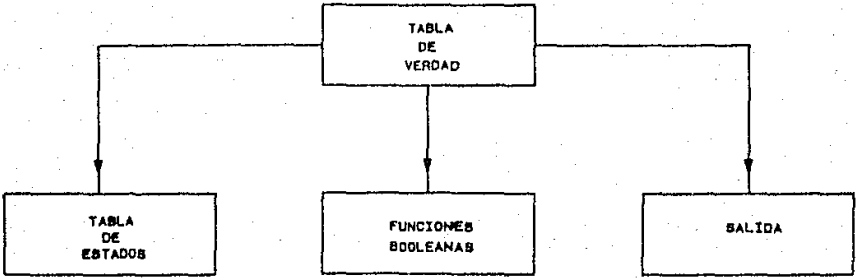


TABLA
DE
VERDAD
TABLA
DE
ESTADOS
FUNCIONES
BOOLEANAS
SALIDA

Figura 4.2.1

En los siguientes párrafos se describe los tópicos involucrados en la obtención de la funciones booleanas del sistema, así como el método utilizado para realizar la simplificación de dichas funciones.

IV.2.1.2.1 Formas canónicas o normales

Supongase el conjunto B = $\{X_1, X_2, X_3\}$ y las dos operaciones (+), (.) forman un álgebra de Boole de tres elementos. Una función booleana puede ser entonces:

$$F(X_1, X_2, X_3) = X_1 + \backslash X_2 . X_3$$

en lo sucesivo se utilizara **F** en lugar de $F(X_1, X_2, X_3)$.

Es conveniente poder expresar un función en forma *canónica o normal*, en la cual se supone que la función viene expresada como la suma de un número de productos, donde cada uno de los cuales contiene cada una de las variables del conjunto B; donde también pueden aparecer las variables complementadas *p. ej.*, la función :

$$g(X_1, X_2, X_3) = X_1 . X_2 . \backslash X_3 + X_1 . X_2 . X_3$$

está en forma canónica.

Para comvertir una función en forma canónica debemos usar la ley de complementación

$$(X + \backslash X) = I$$

Consideremos de nuevo la función

$$F = X_1 + \backslash X_2 . X_3$$

Como la multiplicación por I deja todo elemento invariable

$$F = X_1 . I + \backslash X_2 . X_3$$

haciendo I = $(X_2 + \backslash X_2)$ entonces:

$$F = X_1.(X_2 + \backslash X_2) + \backslash X_2.X_3$$

Así

$$F = X_1.X_2 + X1.\backslash X_2 + \backslash X_2.X_3$$

Esta no es aún la forma canónica, ya que B tiene tres elementos, y los tres no están incluidos en los productos de todos los términos.

Pero multiplicando de nuevo cada término por I

$$F = (X_1.X_2).I + (X_1.\backslash X_2).I + (\backslash X_2.X_3).I$$

y sustituyendo, *i.e.*,

$$F = X_1.X_2.(X_3+\backslash X_3) + X_1.\backslash X_2.(X_3+\backslash X_3) + \backslash X_2.X_3.(X_1+\backslash X_1)$$

desarrollando se obtiene:

$$F = X_1.X_2.X_3 + X_1.X_2.\backslash X_3 + X_1.\backslash X_2.X_3 + X_1.\backslash X_2.\backslash X_3 + \backslash X_1.X_2.X_3 + \backslash X_1.\backslash X_2.X_3$$

la cual es la forma canónica, dado que todo término es un producto que contiene a todos los elementos de B, (incluyendo el complemento).

Evidentemente, esta técnica se puede emplear para expresar en forma conónica una función cualquiera de cualquier número de variables. Los productos como éstos, que contienen a todos los elementos (incluyendo el complemento) de un álgebra de Boole se llaman *términos canónicos*.

Definición: Si se tiene un álgebra de Boole sobre el conjunto $\{B = X_1, X_2, \ldots, X_n\}$ con los operadores (+), (.), entonces los productos de la forma:

$$x_1^z, x_2^z, \ldots, x_n^z$$

donde z puede ser el complemento, se llaman *términos canónicos*.

De los antes mencionado el número de términos canónicos depende del número de variables *n*; en otras palabras , existen 2^n términos canónicos diferentes.

IV.2.1.2.2 Reducción de funciones booleanas

El método de mapas de Karnaugh utilizado para la reducción de funciones Booleanas, progresivamente se empieza a complicar conforme aumenta el número de variables de estado. La técnica de los mapas de Karnaugh para funciones com más de cuatro variables de estado, es un reto a la habilidad humana para reconocer los "encerramientos" a los cuales se les conoce como *cubos* y en consecuencia, la implementación de un algoritmo para computadora, no es realizable. El método de Quine-McCluskey es un procedimiento sistemático tabular, el cual se puede implementar fácilmente en una computadora. Por tal motivo el método de Karnough es ineficiente para los fines funcionales del sistema desarrollado, siendo el método de Quine-McCluskey el que más se adapta a las necesidades, para llevar a cabo la manipulación de la información no gráfica con la cual se obtinen las funciones booleanas.

Método de Quine-McCluskey

El método comienza con un lista de los minterminos de la función los cuales se van a reducir. Los minterminos son almacenados en una tabla formando los *0-cubos* (en lo sucesivo ,un cubo se refiere a los agrupamientos de elementos como se realizan en la técnica de mapas de Karnaugh) los cuales son ordenados para así facilitar el proceso de reducción i.e., el proceso de combinación de *0-cubos* para formar cubos de mayor dimensión.

Para facilitar la presentación del procedimiento, se utilizara un ejemplo: reducir la siguiente función:

$$F = \sum m(1, 5, 7, 8, 9, 10, 11, 14, 15) \qquad (4.1)$$

así, el arreglo-ON para está función es:

$$ON = \begin{bmatrix} 0001 \\ 0101 \\ 0111 \\ 1000 \\ 1001 \\ 1010 \\ 1011 \\ 1110 \\ 1111 \end{bmatrix}$$

A manera de asistencia didactica del procedimiento, se muestra a continuación el mapa de Karnaugh para la función F.

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	0	0
01	0	1	1	0
11	0	0	1	1
10	1	1	1	1

Figura 4.3 Mapa de Karnaugh para la ecuación (4.1)

El primer paso del método, consiste en buscar todos los posibles *1-cubos* agrupados en pares formando así dichos cubos en el mapa de Karnaugh.

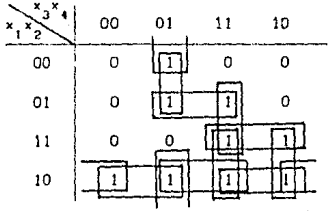


Figura 4.4 Todos los posibles pares de 1's en el mapa de Karnaugh.

Cada par de encerramientos en el mapa de Karnaugh corresponde a un par de *o-cubos* en el arreglo ON, diferiendo tan solo en una variable, por ejemplo el par (0001, 0101) y (1110, 1111). Esto nos indica que se pueden encontrar todos los posibles *1-cubos*, buscando en el arreglo ON todos los pares de entradas, los cuales difieran en el valor de una de sus variables. El procedimiento de busqueda es muy simple pero no todos los *0-cubos* se pueden combinar para formar un *1-cubo* que se distingue por la propiedad particular que a continuación se menciona:

- El número de 1's de un *o-cubo* a otro debe diferir en una unidad (por ejemplo: considerese los pares (0001, 0101) y (0111, 1111)).

El método de Quine-McCluskey, utiliza la ventaja que nos representa el factor antes mencionado; por lo tanto, se debe realizar un ordenamiento al arreglo-ON de los *o-cubos* de acuerdo al número de 1's que estos contienen. Así el ordenamiento para el ejemplo se muestra en la siguiente tabla:

núm. de 1's	0-Cubos	Equivalencia decimal
1	0001 1000	1 8
2	0101 1001 1010	5 9 10
3	0111 1011 1110	7 11 14
4	1111	15

Figura 4.5 tabla de 0-cubos

El correcto uso de la tabla de la figura 4.5 permite determinar fácilmente los pares de *0-cubos* que se pueden combinar para formar los *1-cubos*.

Los *0-cubos* del grupo superior de la figura 4.5 (*i.e.*, los cubos que contienen un solo 1') no se pueden combinar con los cubos que tengan 3 o más 1's; así, solamente se requieren los *0-cubos* del segundo grupo los cuales son considerados para realizar las combinaciones con el grupo superior. Por consiguiente, cada *0-cubo* del grupo superior es comparado con cada uno de los *0-cubos* del siguiente grupo, formando así los *1-cubos*, en otras palabras, se lleva a cabo una relación entre el elemento seleccionado contra todos los elementos del siguientes grupo. Si en el par de *0-cubos* seleccionados, se identifica que difieren en una sola variable, el correspondiente *1-cubo* se obtiene poniendo una "X" en la posición del elemento en la cual los dos *0-cubos* difieren obteniendose de está manera una tabla de *1-cubos* con su respectivo valor decimal equivalente de los dos *0-cubos* combinados. Se puede apreciar en la figura 4.6 de *1-cubos* que el número de grupos disminuye en uno en comparación con la tabla de los *0-cubos* (ver figura 4.4). A continuación se muestra la tabla de los *1-cubos:*

1 -cubos	Combinación de 0-cubos
0x01	1, 5
x001	1, 9
100x	8, 9
10x0	8, 10
11x1	5, 7
10x1	9, 11
101x	10, 11
1x10	10, 14
x111	7, 15
1x11	11, 15
111x	14, 15

Figura 4.6 Tabla de 1-cubos

De lo antes descrito, el procedimiento comienza a comparar el *0-cubo* 0001 con los cubos del siguiente grupo inferior; como se ve *este 0-cubo* puede ser combinado con los cubos 0101 y 1001, ya que solamente con estos existe una diferencia de una variable, por consiguiente se forman los *1-cubo* 0x01 y x001 respectivamente y son almacenados en la tabla de los *1-cubos*, ver figura 4.6. Posteriormente, el segundo *0-cubo* en el grupo superior 1000 se compara con los tres *0-cubos* del grupo inferior, pudiendose combinar solamente con los cubos 1001, 1010 formandose así 100x y 10x0 respectivamente, agregando los *1-cubos* a la tabla de la figura 4.6.

De esta manera se obtienen todos los *1-cubos* que se pueden formar con los *0-cubos* del grupo superior. Se dibuja una linea abajo de estos *1-cubos* (listados en la figura 4.6) para así poder hacer una distinción entre este grupo del siguiente que se genere.

El siguiente conjunto de *1-cubos* es formado por la comparación de los *0-cubos* del segundo grupo (es decir, el grupo donde el número de 1's es igual a dos, figura 4.5) con el tercer grupo. El proceso es exactamente igual a la manera como se obtuvo el primer *1-cubo*, y se concluyé la obtención de la tabla completa de los *1-cubos* (figura 4.6) en forma similar a como se mencionó anteriormente.

En aplicaciones generales, el método de Quine-McCluskey dentro del proceso de pares sucesivos en grupos, continua mientras no se encuentre el último par posible.

Como parte del proceso de obtención de la tabla de *1-cubos*, los *0-cubos* que fueron combinados más de una vez, se identifican con una marca (los 1-cubos obtenidos corresponden exactamente a los encerramientos en el mapa de Karnough de la figura 4.4); la razón se explica y se comprendera más adelante.

El siguiente paso del procedimiento consiste de una busqueda a través de la tabla de la figura 4.6 para identificar los pares de *1-cubos* que se puedan combinar para formar cubos de mayor dimensión (*2-cubo*) Como se puede apreciar en la figura 4.6 se tienen tres grupos, de donde se principia a realizar la comparación de estos cubos del grupo superior con el segundo grupo.

En este caso, los cubos requieren solamente ser comparados con respecto a la posición de la "X". Así, la primer entrada en la figura 4.6, ver 0x01, necesita ser comparada con el cubo 1x10 y como se observa, difiere en sus tres elementos; por lo tanto, no existe combinación concluyendo la comparación del cubo 0x01 ya que no puede ser combinado con otro *1-cubo*. El siguiente cubo de la lista es x001 el cual no puede se combinado con ningun otro *1-cubo* porque no existen cubos que tengan X en la misma posición dentro del segundo grupo de los *1-cubos*, se concluyé así que no existe combinación posible. El tercer cubo en la lista, ver 100x, el cual difiere del 101x en tan solo un posición, así estos dos cubos se pueden combinar formando el *2-cubo* 10xx y se almacena en una nueva tabla y se marcan los elementos que fueron combiandos; finalmente el *1-cubo* del primer grupo 10x0 es comparado con 01x1 y 10x1 (cada uno tiene una "X" en la tercera posición). 10x0 no puede se combinada con el primero de estos; pero se puede combinar con el segundo, generando el *2-cubo* 10xx, sin embargo, como ya existe éste no se debe agregar a la tabla y se marcan los *1-cubos* que los generaron. Completandose así la comparación para el primer grupo de *1-cubos*.

El primer *1-cubo* en el segundo grupo es comparado contra los *1-cubos* del tercer grupo de la tabla de la figura 4.6 para determinar los siguientes *2-cubos*. Estas comparaciones proporcionan uno o más *2-cubos*, ver 1x1x, que se obtiene de la combinación del par (101x, 111x) y por la combinación del par (1x10, 1x11).

Para este ejemplo, el proceso de combinación de cubos de mayor orden finaliza porque los *2-cubos* de la siguiente tabla figura 4.7, no es posible combinarlos.

2-cubos	Combinación de 1 cubos
10xx	8, 9, 10, 11
1x1x	10, 11, 14, 15

Figura 4.7 Tabla de 2-cubos

En el caso general (para *n* variables), el proceso continua, produciendo cubos de mayor dimensión, mientras la combinación de cubos sea posible.

Antes de explicar la información que nos proporcionan los datos de las tablas obtenidas; la cual es utilizada para formar la mínima expresión de la función F (la cual se utilizó para manejar el ejemplo); es importante mencionar la razón del porque en los *1-cubos*; *i.e.*, en comparaciones para obtener los *2-cubos*, solamente es necesario comparar tan solo los pares de los *1-cubos* que tengan la "X" en la misma posición. Considerese la formación de los *2-cubos* 10xx de la figura 4.7, donde se puede apreciar que estan formados por los *0-cubos* 8, 9, 10 y 11, en otras palabras, el *2-cubo* 10xx es formado por la combinación de los minterminos m_8, m_9, m_{10}, m_{11}. Escribiendo la combinación mediante procesos algebraicos se tiene:

$$m_8 + m_9 + m_{10} + m_{11} = X_4\backslash X_3\backslash X_2\backslash X_1 + X_4\backslash X_3\backslash X_2 X_1 + X_4\backslash X_3 X_2\backslash X_1 + X_4 X_3\backslash X_2 X_1 \qquad (4.2)$$

$$= X_4\backslash X_3\backslash X_2(\backslash X_1 + X_1) + X_4\backslash X_3 X_2(\backslash X_1 + X_1) \quad (4.3)$$

$$= X_4\backslash X_3\backslash X_2 + X_4\backslash X_3 X_2 \quad (4.4)$$

$$= X_4\backslash X_3(\backslash X_2 + X_2) \quad (4.5)$$

$$= X_4\backslash X_3 \quad (4.6)$$

Los términos en (4.4) representan dos *1-cubos* 100x y 101x. En cada uno de estos *1-cubos*, la X nos indica que la variable X_1 puede ser eliminada. Por definición, para un problema dado, *1-cubo* representa un producto de variables de las cuales una puede ser omitida, porque es la combinación de dos *0-cubos* con la misma variable, por lo tanto, esto implica que se pueden combinar solamente *1-cubos* que tengan una variable X en la misma posición. Por esta razón los *2-cubos* (4.6) se pueden combinar solamente si sus dos "X's" se encuentran en la misma posición. En general, la combinación de los *k*-cubos se puede realizar solamente si se tienen todas sus *k* "X's" en la misma posición.

Retornando al problema de minimización de la función F; se puede apreciar que los *2-cubos*; figura 4.7, corresponden exactamente a dos variables de los encerramientos de cuatro 1's en la representación de mapas de Karnaugh como se muestra a continuación:

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	0	0
01	0	1	1	0
11	0	0	1	1
10	1	1	1	1

Donde los *2-cubos* solamente son un subconjunto de los minterminos dados en la función F. Sin embargo, comparando con la figura 4.4 se aprecia que los *2-cubos* son representados por siete *1-cubos* (ver encerramientos de 1's) lo cual indica que debieron checarse siete

encerramientos de 1's) lo cual indica que debieron checarse siete *1-cubos* en la tabla, figura 4.6. Estos *2-cubos* se consideran como la mínima expresión de la función F (pero no suficiente) en (4.1).

Primos implicados

Definición. *Un primo implicado es un cubo de una función que no es completamente cubierto por un cubo de mayor dimensión de la función dada.*

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	0	0
01	0	1	1	0
11	0	0	1	1
10	1	1	1	1

Figura 4.8 2-cubos representados en el mapa de Karnough

Para los cuatro *1-cubos* que no fueron checados figura 4.6, estos son considerados como; *primos implicados* de la función F (4.1) Así, en la figura 4.8 se indican solamente 2 de estos *1-cubo*, los cuales son requeridos para involucrar todos los minterminos de (4.1). El método de Quine-Mccluskey, es un método que proporciona una función simplificada directamente de las tablas finales de los cubos (figura 4.7) requeridos.

La tabla para el ejemplo que se ha estado manejando se muestra en la figura 4.9(a), donde cada renglon corresponde a uno de los primos implicados y cada columna los relaciona con uno de los minterminos (los *0-cubos*)(4.1).

Los *primos implicados* son ordenados en la tabla de acuerdo a la dimensión del cubo al cual pertenecen. Los cubos de mayor dimensión, que involucran el mayor número de minterminos, se almacenan primero.

para ser almacenados, pero en un caso general, pueden existir varios tipos de cubos almacenandose en la tabla (figura 4.9(a)) los cubos de mayor a menor dimensión. Los primos implicados son almacenados en las columnas de la izquierda y representados por los minterminos que involucran; así en este ejemplo, cada primo implicado es asignado a un renglon en la tabla, ver figura 4.9(b), donde cada renglon corresponde a un primo implicado y es marcado en las columnas de la derecha las cuales corresponden a los minterminos relacionados con los primos implicados.

La busqueda para la mínima cobertura -la mínima cobertura para una función F, es el mínimo conjunto de cubos necesarios para involucrar los minterminos de una función dada- comienza con la busqueda a través de las columnas de la tabla para identificar que minterminos fue checado solamente una vez; para este ejemplo, se aprecia que corresponde a la columna 8 y 14. Cuando un mintermino es checado una sola vez, implica que el *2-cubo* 10XX (que involucra los minterminos 8, 9, 10 y 11) es el primo implicado identificandose con el mintérmino 8, encontrandose así la mínima covertura para la ecuación 4.1 dada, la cual incluye el *2-cubo* 10XX, en otras palabras, el 10XX es un *primo implicado esencial*. Similarmente, el *2-cubo* 1X1X es también un primo implicado dado que solamente identifica al mintermino 14. Así, se pueden incluir estos dos primos implicados que son marcados con (*) dentro de la mínima cobertura.

Los minterminos identificados con (*) a la izquierda de la columna, son *primos implicados esenciales*.

En este punto del procedimiento, la tabla de los primos implicados muestran la forma que se observa en la figura 4.9(b). Los *primos implicados esenciales* nos proporcionan la información necesaria pero no suficiente para completar la función F, por lo tanto, ahora se obtiene la mínima expresión de la función utilizando los primos implicados que han quedado sin marcar en la tabla de los *1-cubos*. Como se puede deducir de la figura 4.8 estos son los tres posibles pares de *1-cubos* que se pueden seleccionar para completar la cobertura mínima de los minterminos: (1, 5 y 5, 7) (1, 5 y 7, 15) y (1, 9 y 5, 7); de

donde se debe elegir el par de cubos que cubran los minterminos restantes sin involucrar un mintermino que exista dentro de otro primo implicado, en otras palabras, si seleccionamos el par de cubos (1, 5 y 7, 15) esta abarcando un mintermino que se encuentra definido en un *primo implicado esencial*.

Las posibles selecciones de los primos implicados para completar la cobertura, puede tener como consecuencia algún incremento en el costo de hardware (porque cada posible selección consiste de un par de *1-cubos*); considerando lo antes descrito, se eligé el par (1, 5 y 5, 7). Estos dos *1-cubos* y los *2-cubos* que fueron identificados como *primos implicados esenciales*, (PIE's) proporcionan la minima expresión de (4.1).

Finalmente, se escribe la minima expresión de suma de productos, en donde los productos asociados a cada uno de estos primos implicados se realizan de la siguiente manera:

$$
\begin{array}{lllll}
8,\ 9,\ 10,\ 11 & \longrightarrow & 10XX & \longrightarrow & X_4\backslash X_3 \\
10,\ 11,\ 14,\ 15 & \longrightarrow & 1X1X & \longrightarrow & X_4X_2 \\
1,\ 5 & \longrightarrow & 0X01 & \longrightarrow & \backslash X_4\backslash X_2X_1 \\
5,\ 7 & \longrightarrow & 01X1 & \longrightarrow & \backslash X_4X_3X_1
\end{array}
$$

obteniendose asi, de esta manera, la minima expresión para (4.1):

$$F = X_1\backslash X_2 + X_1X_3 + \backslash X_1\backslash X_3X_4 + \backslash X_1X_2X_4$$

Notese que esta es la minima suma de productos pero no es única. Dado que se puede elegir otro par de *1-cubos* para proporcionar otra minima expresión equivalente a la función antes presentada.

	Minterminos cubiertos por PI.	1	5	7	8	9	10	11	14	15
	8, 9, 10, 11				+	+	+	+		
	10, 11, 14, 15						+	+	+	+
	1, 5	+	+							
	1, 9	+				+				
	5, 7		+	+						
	7, 15		+							+

(a)

	Minterminos cubiertos por PI.	1	5	7	8	9	10	11	14	15
*	8, 9, 10, 11				+	+	+	+		
*	10, 11, 14, 15					+	+	+	+	
	1, 5	+	+							
	1, 9	+				+				
	5, 7		+	+						
	7, 15		+							+
					+	+	+	+	+	+

(b)

Figura 4.9 Primos implicados: (a)tabla inicial (b)tabla después de identificar los PIE'S

Los métodos de Quine-McCluskey y Karnough, aunque aparentemente distintos, se basan en la misma ley de complementación $X + \backslash X = 1$; ambos consisten en una agrupación -La de Karnough es geometrica y la de Quine-McCluskey es numérica- y hacen uso de la ley de tautologia [1] y factorizan la solución final.

De aquí su analogía. El método de la representación gráfica es útil hasta las funciones de cuatro variables, mientras que el método numérico se considera para mayor número de variables. Es interesante observar que Karnough desarrolló su método a comienzos de la década de

los 50, cuando la computadora aún no habia tomado fuerza, mientras que Quine-McCluskey desarrolló el suyo a finales de dicha década.

Debemos hacer una observación final acerca del método de simplificación de Quine-McCluskey. El artificio necesario en este método, especialmente cuando se usa la tabla de términos primos, es mucho mayor que el método de la represetación gráfica. Sin embargo, el primero se puede extender para simplificar funciones de tantas variables como se desee. Además, el papel de la tabla de términos primos se puede sustituir mediante programación en computadora. Por otra parte, el método de la representación gráfica, es muy efectivo para funciones de hasta cuatro variables, se hace complicado para cinco y más o menos imposible de seis en adelante.

Por lo tanto, el sistema utiliza el método Quine-McCluskey para llevar a cabo la reducción de funciones booleanas; las razones son obvias, para mayor detalle [2].

IV.2.1.3 Herramientas: Son una serie de utilerias que permiten al usuario: hacer un respaldo de su información es disco flexible, realizar una impresión de carta ASM para lo cual se utiliza un graficador, eliminación de algun archivo de trabajo y finalmente permite regresar al menú principal del sistema. Ver figura 4.10.

IV.2.1.3.1 Eliminación de archivos (Borrar archivo): Proporciona una forma de eliminación de archivos de trabajo; para tal fin debemos primeramente de listar el directorio de archivos actuales para asi poder seleccionar con ayuda del "*Ratón*" el archivo que se desea eliminar, en seguida se presiona la <TECLA 2>.

IV.2.1.3.2 Salida a graficador (Graficador): La manera para obtener una impresión en papel se lleva a cabo realizando una asociación del archivo de trabajo con una estación de trabajo diferente a la utilizada para despliegue en video (ver capitulo III). De tal manera, toda o parte de la información gráfica es enviada al graficador que se encuentra asociado a otra estación de trabajo. Esté proceso es transparente al usuario del sistema, dado que solo se requiere

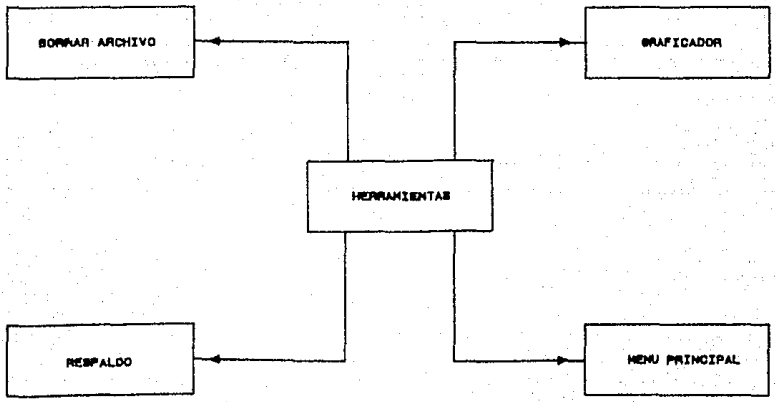


Figura 4.10 UTILERIAS GENERALES

seleccionar dicha opción y tener en video el archivo que contiene la carta ASM que se desea gráficar.

IV.2.1.3.3 Respaldo de información (Respaldo): Permite al usuario realizar un respaldo en disco flexible de sus archivos de trabajo. El respaldo de información se lleva a cabo utilizando comandos del sistema operativo AIX 2.1 [10].

IV.2.1.3.4 Menú principal (Menu_princ): Es utilizada para abandonar la opción de *herramientas*, regresando al menú principal del sistema.

IV.2.1.4 Salir de sesión: Es utilizada para *abandonar la sesión*.

Empero, si no se eligé ninguna de las opciones antes mencionadas, se debe crear un archivo de trabajo proporcionando un nombre para el nuevo archivo y prosteriormente presionar la tecla <ACTION>, y de está manera se da principio al uso del editor gráfico.

IV.2.2 Menú editor gráfico

Después de entrar al editor gráfico el sistema presenta el siguiente menú: LINEA ESTADO CONDICION SALIDA CONECT TEXTO OPCION MENU P ver figura 4.11. Al momento de elegir alguna opción aparece el ícono (capitulo II) que lo caracteriza para indicar que la opción ha sido activada.

A continuación se describe en detalle cada una de las diferentes opciones, las cuales requieren para su uso el manejo del "*Ratón*" ó "*Teclado de funciones*", además es importante mencionar; para cada ícono presentado por el editor gráfico en la elaboración de las cartas ASM, aparecerán líneas pequeñas en los bordes para indicar donde debe comenzar a trazarse la línea que se emplea como unión con otro elemento gráfico.

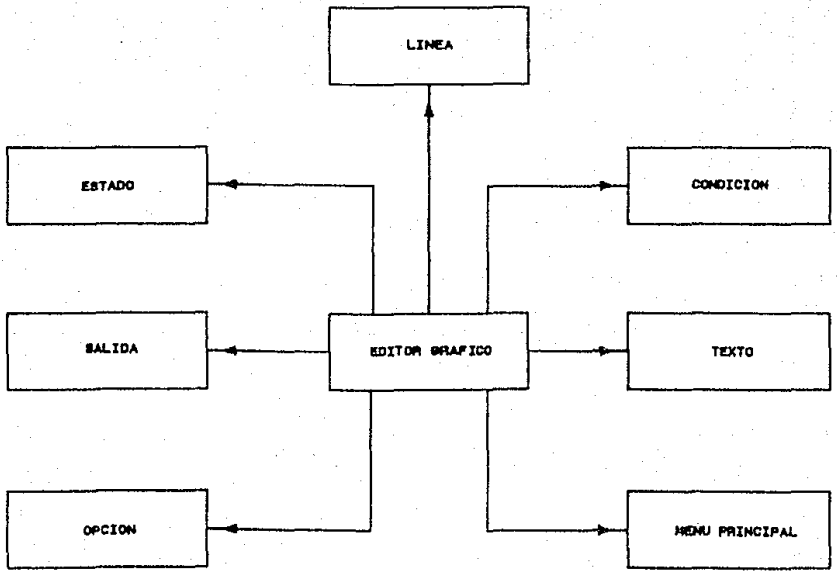


Figura 4.11 Menu del editor grafico.

IV.2.2.1 Linea: Proporciona la facilidad para dibujar lineas; pero su activación solo se lleva a cabo cuando se unen dos elementos de la carta ASM. La manera de utilizarse es la siguiente:

Con la <TECLA 1> indicamos el primer punto de la linea, si presionamos nuevamente la <TECLA 1> indicaremos otro punto; uniendo ambos puntos el sistema. Así sucesivamente podemos dibujar una secuencia de lineas, las cuales aparecerán de color rojo para indicar que ha finalizado el trazado de la linea y de color blanco hasta que no se indique un punto que pertenesca a algún elemento de la carta; para finalizar la edición de lineas se presiona la <TECLA 4>.

IV.2.2.2 Estado: Sirve para activar el icono utilizado para definir un caja-estado, como se muestra en la figura 2.4(b). Presionando la <TECLA 1> en la posición deseada en la pantalla se crea dicha caja-estado y así, de esta manera se puede ir elaborando las cajas-estados con solo presionar la <TECLA 1>. Finalmente para concluir la edición se presiona la <TECLA 4>.

IV.2.2.3 Condición: Proporciona el icono para la condición (ver, figura 2.4(a)). Se crea la caja-condición en la posición requerida presionando la <TECLA 1> hasta completar el número de cajas-condición deseadas. Para concluir la edición se presiona la <TECLA 4>.

IV.2.2.4 Salida: Esta opción nos proporciona, el icono para la caja-salida-condicional (ver figura 3.4(c)). Así, la manera de utilizar el dispositivo para crear las cajas-salida-condicional necesarias, se realiza de manera semejante a los procesos antes descritos, es decir, se debe presionar la <TECLA 1> en la posición deseada para la creación de la caja-salida-condicional. Y se concluyé la edición presionando la <TECLA 4>.

IV.2.2.5 Texto: Esta opción es muy importante dado que proporciona la manera para llevar a cabo la documentación de la carta ASM; es decir, se puede incluir el texto que requiere la carta como: nombre de variables, código de estado, nombre del estado y salidas, etc. Es

importante porque representa la información no gráfica que se procesara para la obtención de funciones booleanas. Para poder hacer uso de la opción texto, se deber seguir la siguiente secuencia de pasos que a continuación se mencionan:

seleccionar la opción "*Texto*" del menú presentado por el editor gráfico, a continuación con la <TECLA 1> se indica la posición donde se va a incluir el texto, posteriormente se teclea el texto (el cual no debe de contener más de 25 caracteres), finalmente se presiona la tecla <ACTION>, con lo cual el sistema acepta el texto en la posición antes indicada.

De esta manera, se pueden incluir los textos necesarios para la documentación de la carta ASM. Para finalizar la utilización de la opción se presiona la <TECLA 4>.

IV.2.2.6 Almacenamiento en disco duro (Menu-P): Con esta opción, se puede llevar a cabo el almancenamiento en disco duro del archivo de trabajo ó comenzar a editar un nuevo archivo.

IV.2.3 Utilerías del editor gráfico

Se presenta en la figura 4.12, las diferentes opciones que ofrece como utilerías el editor gráfico.

Opción: Presenta un menú con el cual se podrá manipular las cartas ASM; ya que proporciona las herramientas necesarias para facilitar al usuario la elaboración de su carta; ofreciendo lo siguiente:

CAMB CUR CURS N RESETEAR AUMENTO1 AUMENTO2 TRASLADA BORRA LIMPIA RETORNA, con lo cual se pueden realizar las siguientes transformaciones básicas; alejamiento total, acercamiento parcial traslación total de la carta a otra posición en video, eliminación de elementos, limpiar área de despliegue para comenzar a editar una nueva carta, así como, después de una serie de alteraciones a la carta la

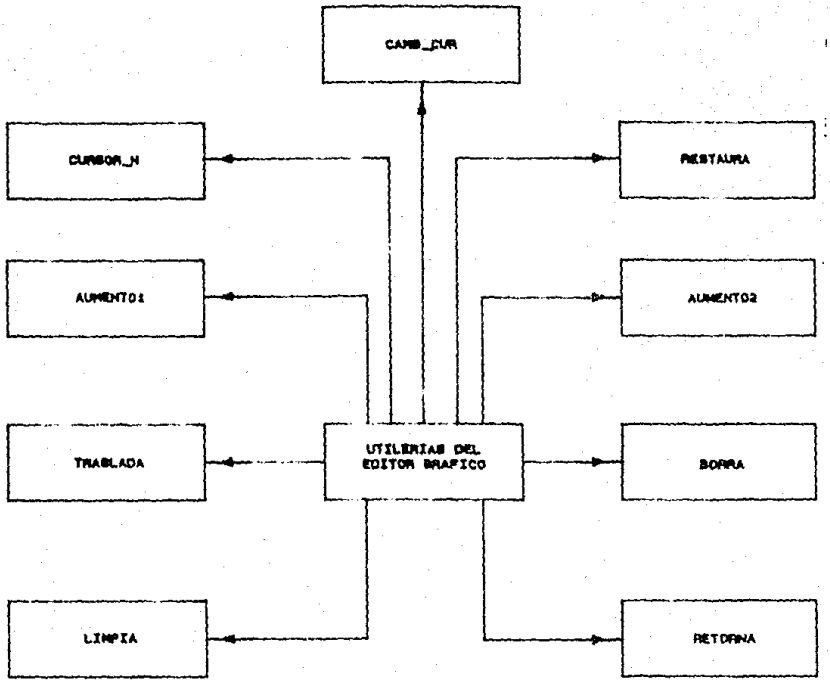


Figura 4.12 Utilerias del editor grafico.

podemos volver a su estado original. El menú antes presentado aparece en la parte inferior del video al momento de seleccionar OPCION.

A continuación se describen en forma detalla cada una de las opciones presentadas en los párrafos anteriores.

IV.2.3.1 Modificación del cursor (Camb_cur): Proporciona la facilidad para modificar la apariencia que presenta el cursor; ya que al inicializarse el sistema es una cruz pequeña. Por lo tanto, está opción nos ofrece la alternativa para poder emplear un cursor de cruz grande (abarca toda el área de despliegue), el cual es útil para elaborar lineas ortogonales para mayor exactitud en el trazado.

IV.2.3.2 Inicialización del cursor (Curs_N): Con está opción (la cual es el complemento de la antes descrita) se lleva a cabo la inicialización del cursor, es decir, el cursor presenta la apariencia original.

IV.2.3.3 Restauración (Restaurar): La restauración es muy útil, ya que nos permite hacer una presentación de la carta en su apariencia original. Dado que la carta se puede manipular aplicandole una serie de transformaciones elementales como: acercamientos totales o parciales y traslaciones. Con esta opción se restablece las condiciones iniciales después de haber realizado alteraciones temporales.

IV.2.3.4 Acercamiento parcial (Aumento1): Nos permite poder realizar acercamientos parciales de determinada zona de la carta; se debe tomar en cuenta que el sistema solamente permite tres niveles internos de acercamiento, es decir, si se realiza un acercamiento de una determinada zona, entonces a está zona solamente se le podrán aplicar dos acercamientos más. La utilidad de está opción se hace presente cuando se elabora una carta de dimensiones mayores al área de trabajo que presenta el video, entonces se procede a aplicar un alejamiento (más adelante se explica la manera de llevarlo a cabo) para tener una vista total de la carta y así poder agregarle más información; de tal manera que después se podrá requerir realizar un acercamiento parcial

de la carta, ya sea con fines informativos o para realizar correcciones.

Después de haber explicado la utilización de está opción, se describen los pasos para poder emplearla: debemos de indicar con la <TECLA 1> un punto superior izquierdo y un punto inferior derecho (con la misma tecla) para crear un borde del área a la cual se le aplicará el acercamiento.

IV.2.3.5 Alejamiento total (Aumento2): Nos permite poder realizar acercamientos o alejamientos totales de la carta. Su utilidad se encuentra ligada con el *acercamiento parcial*, ya que ambas opciones facilitan la elaboración de la carta ASM. Así, se procede a continuación a describir la forma en que se lleva a cabo el alejamiento y acercamiento:

- *Acercamiento*; la realización de este proceso se lleva a cabo a partir del área de despliegue que presenta el video, es decir, el dispositivo emplea coordenadas normalizadas (de 0 a 1), por lo tanto, el punto (0,0) corresponde al centro del video. De esta forma se define el área de aplicación para la función de acercamiento, el cual corresponde al área definida por los valores: $X = \pm 1$ y $0 <= Y < 1$. Se identifica que corresponde al área del punto (0,0) hacia arriba del video, donde dicha zona está activa para realizar los acercamientos moviendo el cursor y presionando la <TECLA 1>; definiendo así un nivel de acercamiento.
- *Alejamiento*: La forma en que se definió el área para la realización del alejamiento es de manera similar al proceso que se lleva a cabo en el acercamiento. Por tal motivo solo se mencionó su parte funcional. Se debe realizar a partir del punto (0,0) del video hacia abajo y presionando la <TECLA 1>

IV.2.3.6 Traslación (Traslada): Con la traslación se puede realizar los movimientos totales (la traslación se aplica en forma global) que se requieran para llevar a cabo la creación de la carta. Se pueden

realizar movimientos hacia un lugar específico de video, dado que se puede requerir mover la carta totalmente con el fin de agregar más información. Es importante mencionar, que la utilización de las opciones antes descritas en combinación con la traslación, fácilitan el proceso de elaboración de la carta ASM. Para poder llevar a cabo la traslación se debe considerar lo siguiente:

El punto que se indique con la <TECLA 1> dentro del área de despliegue, se mueve al centro del video.

IV.2.3.7 Eliminación de elementos (Borra): Se pueden eliminar elementos gráficos (previamente editados), lo cual es de gran utilidad al momento de realizar la edición de la carta ASM. Para tal fin se debe indicar el elemento a borrar presionando la <TECLA 1> a lo cual el elemento se "ilumina" y posteriormente se presiona la <TECLA 2> para confirmar la eliminación de dicho elemento.

IV.2.3.8 Inicialización de archivos editados (Limpia): Está opción nos proporciona la facilidad para borrar totalmente la carta ASM (la información se pierde definitivamente), con el fin de preparar el sistema para elaborar una nueva aplicación con el mismo nombre de archivo.

IV.2.3.9 Retorno al editor gráfico (Retorna): Finalmente se cuenta con una opción que permite regresar al editor gráfico y poder continuar con la edición de la carta ASM. Es importante mencionar que si dentro de las utilerías del editor gráfico, ha sido aplicada algún acercamiento a alejamiento y regresamos al menú principal del editor gráfico, la alteración permanece hasta que no se aplique la opción de utilleria "Restaura".

IV.3 Estructuras de información

Los programas en computadora operan normalmente con tablas de información. En la mayoría de los casos dichas tablas no son simplemente masas amorfas de valores numéricos; involucran importantes

relaciones estructurales entre los datos. En su forma más simple, una tabla debe ser una lista lineal de elementos.

En situaciones más complicadas, la tabla puede ser un conjunto bidimensional (*p.ej.*, una matriz) o puede ser un conjunto *n*-dimensional para valores de *n* superiores a dos o una estructura en árbol poniendo de manifiesto relaciones jerárquicas o colaterales, o, finalmente, una compleja estructura multienlazada con varias interconexiones.

A continuación, se resumen las particularidades más importantes sobre las estructuras de la información: propiedades estáticas y dinámicas de las distintas clases de estructuras; formas de ubicación en la memoria y representación de datos estructurados; así como algoritmos eficientes para la creación, modificación, acceso y eliminación de información estructural. A lo largo de este subcapítulo, se enunciaran las partes del sistema involucradas con alguna o varias de estas técnicas para los procesos de entrada/salida que realiza el sistema *AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADO A PAL's*.

Es importante definir, en este punto diversos términos y notaciones que se usaran frecuentemente de ahora en adelante. La información en una tabla consiste de una serie de nodos (llamados registros o entidades); se denominaran ocacionalmente *ítem* en vez de *nodo*. Cada nodo consiste en una o más palabras consecutivas en la memoria de la computadora, divididas en partes denominadas *campos*. En el caso más simple, un nodo es una única palabra de memoria y tiene sólo un campo que es la palabra entera. La dirección de un nodo, llamada también *apuntador o referencia* de este nodo, es la dirección en memoria de su primer palabra, dicha dirección se toma, con relación a la estructura de información.

Normalmente existe gran cantidad de información estructural en los datos, de lo que realmente se desea representar en computadora.

Esta, por lo tanto, claro, que se debe decidir en cada caso qué estructura deben tener las tablas de información. Para tomar esta decisión, se necesita conocer qué operaciones se realizan sobre los datos. Para cada problema considerado en el desarrollo del sistema y presentado en este capítulo, *se considera, por consiguiente, no sólo la estructura de los datos sino también la clase de operaciones a realizar con estos* , así como sus propiedades intrínsecas.

lista lineal. Una *lista lineal* es una serie de n >= 0 nodos x[1], x[2], ... , x[n] cuyas propiedades estructurales incluyen, esencialmente, sólo las posiciones relativas lineales (una dimensión) de los nodos: de hecho si n > 0, x[1] es el primer nodo; cuando 1 < k < n, el nodo *k*-ésimo x[k] está precedido por x[k-1] y seguido por x[k+1]; y x[n] es el último nodo.

Las operaciones que se realizan con listas lineales incluyen, por ejemplo, las siguientes:

1) Acceso al nodo *k*-ésimo de la lista, con el fin de examinar y/o cambiar los contenidos de los campos.
2) Insertar un nuevo nodo, inmediatamente delante del nodo *k*-ésimo.
3) Suprimir el *k*-ésimo nodo.
4) Cambiar dos o más listas lineales en una sola lista.
5) Separar una lista en dos o más listas.
6) Realizar una copia de una lista lineal.
7) Determinar el número de nodos de una lista
8) Clasificar los nodos de una lista en orden ascendente, basándose en ciertos campos de los nodos.
9) Buscar las aparición de nodos en una lista con el valor determinado en algún campo.

Las listas lineales en las que tiene lugar inserciones, supresiones y consultas en el primer y último nodo; que son utilizadas con mucha frecuencia son:

Una *pila* es una lista lineal en la que se realizan todas las inserciones y supresiones (y normalmente todos los accesos), en un extremo de la lista.

Una *cola* es una lista lineal en la que se realizan todas las inserciones en un extremo de la lista; todas las supresiones (y normalmente todos los accesos) se realizan en el otro extremo.

Para una mayor flexebilidad en el manejo de listas lineales, se puede incluir dos apuntadores en cada nodo, direccionando los elementos a cada lado de dicho nodo.

Una lista doblemente ligada ocupa más espacio en memoria que una liga simple. Las operaciones adicionales que pueden realizarse con mayor eficacia son a menudo una buena compensación de este espacio extra. Además de poder recorrer la lista doblemente ligada en ambos sentidos.

Las estructuras de información comparten dos operaciones básicas; *almacenar un item y recuperar un item.*

IV.3.2 Asignación ligada

En lugar de almacenar una lista lineal en direcciones de memoria consecutivas (la manera más simple y natural) se utiliza un esquema mucho más flexible en el que cada item contenga un apuntador con el siguiente item de la lista.

Asignación secuencial

Direccion	Contenido
L_0 + C	item 1
L_0 + 2C	item 2
L_0 + 3C	item 3
⋮	⋮
L_0 + nC	item n

Asignación ligada

Direccion	Contenido	Liga
A	item 1	B
B	item 2	C
C	item 3	D
⋮	⋮	⋮
F	item n	∧

donde:

∧ : apuntador nulo

C : número de palabras por item

L_0: dirección de la estructura de información

Después de haber descrito las diferentes estructuras de información, se procede a presentar varias características obvias:

1) La asignación ligada requiere un espacio de memoria adicional para los apuntadores.
2) Es fácil eliminar un item en cualquier parte de una lista ligada.
3) Es fácil insertar un item en cualquier parte de una lista cuando se está empleando un esquema de apuntadores.
4) Las referencias aleatorias en un determinado, item de la lista son mucho más rápidos en el caso secuencial.
5) El esquema ligado facilita la unión de dos listas o su separación.
6) El esquema de apuntadores se presta, por si solo, para la obtención de estructuras más complejas que las simples listas lineales.
7) Las operaciones simples, como el recorrido secuencial de una lista, son ligeramente más rápidos para las listas secuenciales, en varias computadoras.

Por lo tanto, vemos que la técnica de apuntadores nos libera de las limitacines impuestas por la naturaleza consecutiva de las posiciones de memoria de la computadora, nos proporciona un mayor rendimiento en

algunas operaciones, mientras que, por otro lado, se pierden algunas posibilidades. Normalmente queda claro qué técnica de asignación será la más apropiada en una situación dada y, a menudo se utilizan ambos métodos durante el desarrollo del sistema.

IV.3.3 Arboles

De un modo general, la estructura en árbol significa un relación de ramificación entre nodos.

Definición: Un árbol es un conjunto finito T *de uno o más nodos, tales que:*

a) *Existe un nodo especial llamado la raiz del árbol, raiz (T); y*

b) *Los nodos restantes (excluyendo la raiz) están agrupados en* m >= 0 *conjuntos disjuntos* T_1, ... ,T_m *se llaman subárboles de la raiz.*

Se desprende de la definición que cada nodo de un árbol es la raiz de algún subárbol contenido en la totalidad del mismo. El número de subárboles de un nodo se llama el grado de este nodo. Un nodo de grado cero se llama *nodo terminal* (hoja).

Un nodo terminal se llama ocacionalmente *nodo rama*. El nivel de un nodo respecto a T se define diciendo que la raiz tiene el nivel 0 y los otros nodos tienen un nivel superior en un grado, al que tiene la raiz de subárbol T_j que los contiene. Estos conceptos se ilustran en la figura 4.13. La raiz es A y tiene dos subárboles {B} y {C, D, E, F, G} . El árbol {C, D, E, F, G} tiene el nodo C como raiz. El nodo C es de nivel 1 con respecto a la totalidad del árbol.

Arboles binarios. Los árboles generales se representan normalmente en términos de algún árbol binario equivalente en la computadora.

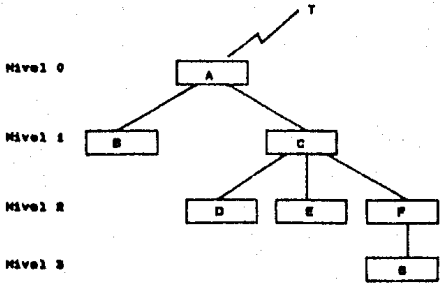


Figura 4.13 Arbol.

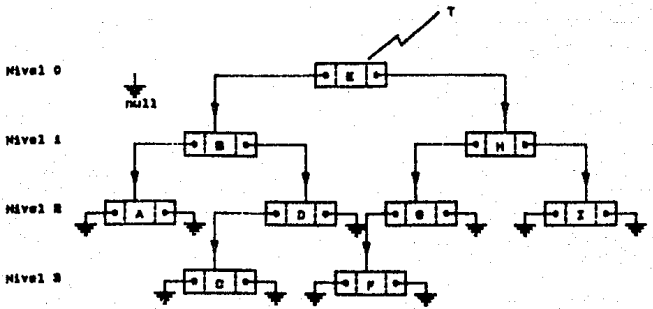


Figura 4.14 Arbol binario.

Definición. Un árbol binario, es un conjunto finito de nodos que, o está vacío o consta de una raíz junto con dos árboles binarios.

Esta definición nos sugiere una forma natural de representar árboles binarios en una computadora: deben existir dos apuntadores o referencia, liga-izquierda y liga-derecha en cada nodo, y una variable de enlace T que nos "direcciona el árbol". Si el árbol está vacío, T = nulo, si no, T es la dirección del nodo raíz del árbol, mientras que liga-izquierda y liga-derecha direccionan los subárboles de la izquierda y derecha de la raíz, respectivamente, ver figura 4.14.

Esta representación en memoria simple y natural, justifica la especial importancia de las estructuras de árboles binarios; muchos árboles que surgen en las aplicaciones son binarios, por lo que los árboles binarios son de interes por derecho propio.

Existen diferencias básicas entre los árboles y los árboles binarios como:

- Un árbol nunca está vacío, es decir, siempre tiene por lo menos, un nodo y cada nodo de un árbol puede tener 0, 1, 2, ... hijos.
- Un árbol binario puede estar vacío y cada uno de sus nodos puede tener 0, 1, ó 2 hijos; distinguiendo entre el hijo de la "izquierda" y el de la "derecha".

Arboles_B. Un criterio muy sensible fue postulado R.Bayer en 1970, todas las páginas (excepto una) contienen entre n y 2n nodos para una constante *n* dada. Por lo tanto, en un árbol con N de *item* y con un tamaño máximo de 2n nodos por página, en el peor de los casos se requiere hacer $\log_n N$ accesos de página y en donde el acceso de página domina claramente el esfuerzo de busqueda. Por otra parte el factor de almacenamiento utilizado es al menos del 50% puesto que las páginas son siempre llenadas al menos a la mitad, con todas estas ventajas el esquema involucra algoritmos simples de comparación, inserción y borrado.

A esta estructura de datos se le llamara *Arbol_B* y poseen las siguientes características; *n* es el orden del árbol_B [21].

1.- Cada página contiene a lo más 2n ítem (llaves).
2.- Cada página excepto la pagina raiz contiene al menos *n* ítem.
3.- Cada página de alguna forma es una página hoja, *i.e.*, sin descendientes o con m + 1 descendientes, donde m es el número de llaves.
4.- Todas las páginas hoja aparecen al mismo nivel.

En la figura 4.14.1 se muestra un árbol_B de orden 2 y dos niveles, todas las páginas contienen 2, 3 o 4 *ítems*.

IV.3.4 Asignación dinámica de memoria

En párrafos anteriores se ha visto como el uso de apuntadores implica que las tablas (estructuras de información) no esten colocados secuencialmente en memoria; un número de tablas puede independientemente crecer y disminuir en un área común de memoria. Como medida de administración de la memoria en lugar de utilizar simplemente el máximo tamaño necesario y perder espacio en los ítem menores, se utiliza un ítem algo menor y se emplea lo que se podría llamar la clásica *filosofía de memoria ligada* (si no existe espacio para la información en una determinda dirección, se coloca en cualquier otra parte que se encuentre disponible y se crea un apuntador que la direccione).

Empero, para otras muchas aplicaciones un tamaño único de ítem es razonable; a menudo se desea tener ítem de diferente longitud compartiendo un área en común de memoria; dicho de otro modo, se requieren de algoritmos para reservar y liberar bloques de memoria de tamaño variable de un amplia área de memoria, donde dichos bloques se componen de direcciones consecutivas de memoria. Tales técnicas se conocen generalmente por algoritmos de *asignación dinámica de memoria*.

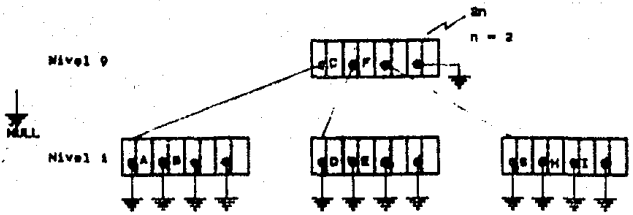


Figure 4.14.1 Arbol_B.

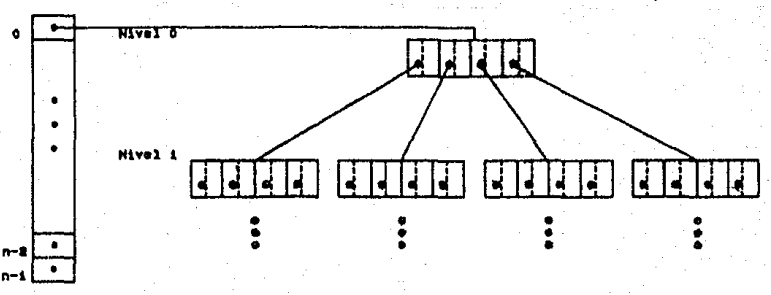


Figure 4.15

IV.4 Descripción interna del funcionamiento del sistema

Cuando el sistema *AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADA A PAL's* es accesado por primera vez se crea una lista lineal ligada simple, la cual contiene los nombres de los archivos en orden alfabetico, los cuales han sido creados previamente dentro del sistema. Cada nodo de la lista contiene el nombre del archivo, un campo de liga a nodo así como un identificador del tipo entero; que será utilizado con la finalidad de realizar accesos o borrado de archivos, además se tiene la ventaja de tener un directorio actualizado en forma inmediata.

Si se requiere crear un archivo, basta con teclear el nombre o seleccionar desde el directorio que se muestra en la zona de despliegue para tal fin, posteriormente, el sistema busca la existencia del archivo, en caso de encontrarlo, se inicia el proceso de recuperación de las estructuras de información contenidas en el archivo que serán utilizandas por el sistema internamente, una estructura híbrida, es decir, una combinación de un arreglo de apuntadores y arboles_B, será utilizada para clasificar la información de cada icono, así como accesar la información en forma rápida, proporcionando la facilidad de insertar y borrar información, ver figura 4.15.

Dentro de la etapa de edición; cuando el usuario se encuentra editando, tiene a su disposición cinco iconos que utilizará para poder elaborar una carta ASM, mediante el uso de un teclado de funciones (LPFK) así como la opción de utilizar un ratón y una tableta digitalizadora. Cada icono que se dibuje tendrá una información asociada que se agregará a la estructura de datos, en caso de eliminar un icono su información asociada inmediatamente será borrada de la estructura de datos. Cada nodo que forma parte del árbol_B contiene la siguiente estructura de información:

- Cuatro campos de liga.
- Cuatro estructuras de datos.

de donde cada estructura se encuentra formada por los siguientes campos:

i - Tipo de estructura
ii - Identificador de estructura
iii- Posición
iv - Código
v - Información asociada

Dicha estructura de datos híbrida es empleada cuando se selecciona las opciones del menú del editor: *estado, condición, salida condicional, texto*; de las cuales por cada una de estas opciónes se asocia a un árbol_B; por otro lado, en caso de elegir la opción de línea, se utiliza un tipo de estructura diferente, se emplea una lista doblemente ligada para las líneas que se dibujen. La información para cada nodo agregado a la lista es:

1.- Estructura línea
2.- Apuntador a coordenadas de puntos (lista ligada simple)
3.- Apuntador a nodo anterior (campo de liga)
4.- Apuntador a nodo posterior (Campo liga)

Por otra lado, la estructura de la línea se forma por:

1.1 Identicador
1.2 Tipo
1.3 Número de puntos
1.4 Conexión 1
1.5 Tipo 1
1.6 Conexión 2
1.7 Tipo 2

Se ilustra la estructura de datos empleada figura 4.16 presentandose gráficamente el *nodo línea:*

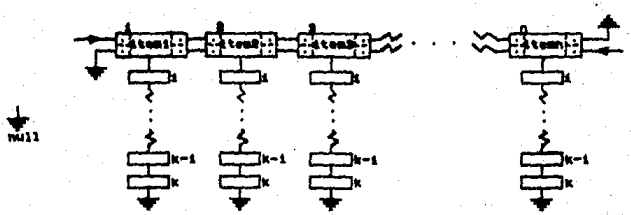


Figure 4.16

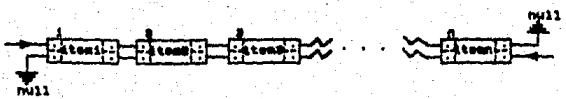
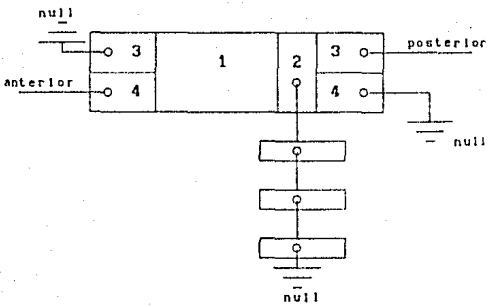


Figure 4.17



Después de haber concluido de editar una carta ASM; la información contenida en la estructura datos híbrida y en la lista doblemente ligada será almacenada en disco duro o en un diskette de 5 $^{1}/_{4}$ de alta densidad.

El siguiente paso es la asignación del código de estado, es decir, a cada estado le es asignado una secuencia binaria, para tal fin se utiliza el código Gray [5] por las ventajas inherentes a dicho código. Para la generación del código Gray se utilizá una lista doblemente ligada, ver figura 4.17 cuyos nodos son semejantes a los que se manejan en la generación de líneas, se utiliza una asignación dinámica porque se requería de accesar la información contenida en los nodos anteriores y en forma simultanea agregar información al final de la lista. El número de bits por cada secuencia esta directamente relacionado con el número de estados, es decir:

$$\#bit = \frac{\log(\#\text{ de estados})}{\log(2.0)} + 1$$

después de haber generado la lista doblemente ligada que contiene la secuencia de cada uno de los estados, se procede a insertar esta información en cada uno de los nodos del árbol_B que contiene la información de los estados, pero para asignar la secuencia adecuada a cada estado se genera un arreglo que contiene la información de las líneas, además se agrega un campo de marca a cada elemento del arreglo; el tamaño del arreglo es igual al número de nodos de la lista

doblemente ligada que se encuentra asociada a la generación de líneas, así la creación de este arreglo permite que el sistema pueda accesar toda la información que se encuentra asociada a la carta ASM. Este arreglo es ordenado en forma ascendente utilizando como llave el campo llamado *conex* 1. Para realizar el ordenamiento se utilizá el método de Quick-Sort.

La asignación del código Gray se procede por niveles, es decir, cada icono *If* (variable de entrada) que se encuentre,representa otro nivel ver figura 4.18, la secuencia es asociada a cada estado de derecha a izquierda según su aparición (la numeración indica la secuencia de asignación como se muestra en la figura 4.19). Para la asignación de código Gray por niveles se utiliza una lista ligada simple conteniendo la información de cada condición, está lista es utilizada como una cola donde las primeras entradas son las primeras salidas (FIFO).

Por otra parte, al concluir la asignación de código Gray, el siguiente paso es la generación de tablas de estados. En esta fase del proceso del diseño, el sistema utilizá las siguientes estructuras de datos:

- Estructura híbrida (arreglo de apuntadores a árboles_B).
- Arreglo dinámico, el cual contiene la información de cada línea.

la utilización del arreglo dinámico permite hacer el recorrido de la carta ASM, como se ilustra en la figura 4.20, así mediante el empleo del arreglo dinámico en combinación con la estructura híbrida (figura 4.18) se genera cada renglón de la tabla de estados; donde cada renglon se forma utilizando la estructura que a continuación se muestra:

25 bits	50 bits	25 bits	50 bits	50 bits
Estado inicial	Entradas	Estado siguiente	Salidas	Salidas Condicion

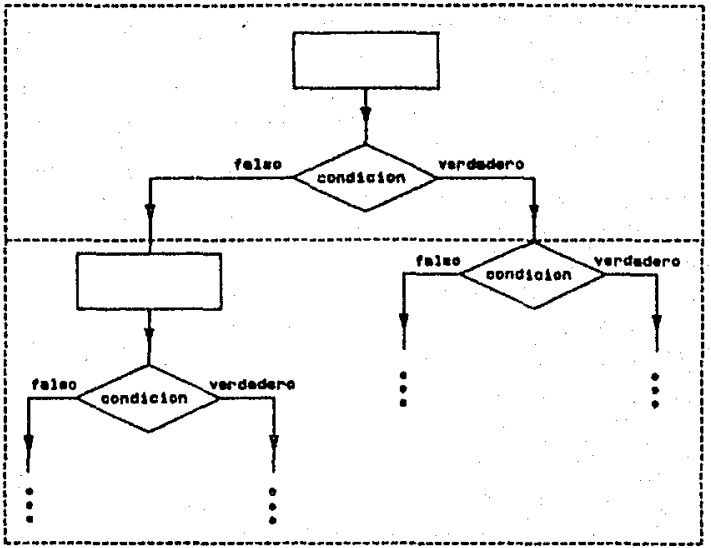


Figura 4.18 Asignacion de codigo estado

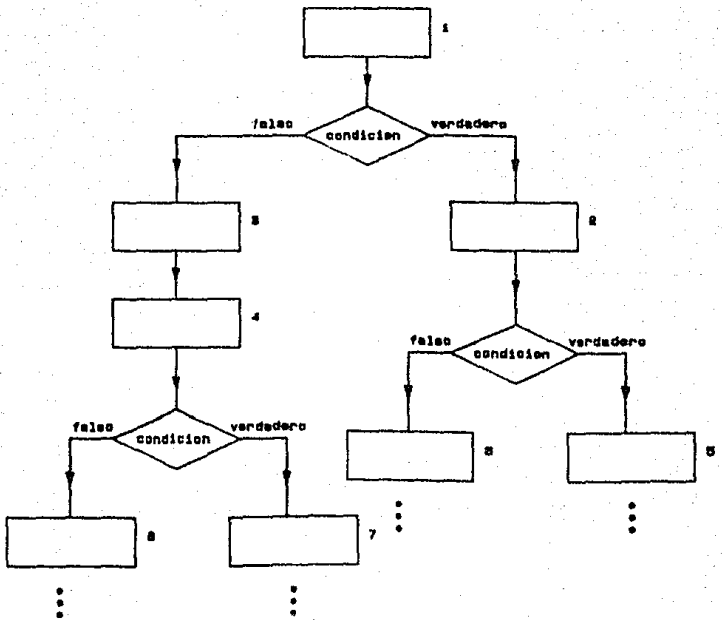


Figura 4.10 Secuencia de asignacion de codigo estado.

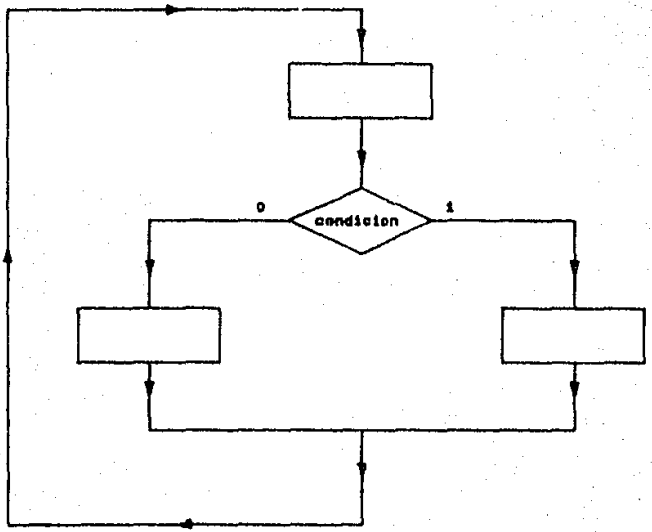


Figura 4.20

La información de cada renglón de esta tabla se encuentra contenida en los nodos de una lista ligada simple. Para la generación de cada renglon de la tabla de estados ocurren los siguientes eventos: Se crean tres árboles binarios de donde cada árbol se asocia a entradas, salidas y salidas condicionales accesandose la información que se encuentra contenida en cada árbol_B de la estructura hibrida de donde; el árbol B asociado a la información de los *estados* genera el árbol binario de salidas, el árbol_B que se encuentra asociado a *condición* genera el árbol binario con la información de las entradas y finalmente el árbol_B que se encuentra asociado al campo de la estructura *salidas condionales* genera el árbol binario con la información para salidas condicionales. Los tres árboles binarios contienen todas las entradas, salidas y salidas condicionales contenidas en la carta ASM; además estos árboles son utilizados para marcar los campos: *entradas, salidas y salidas condicionales* de la tabla de estados,al momento que se realiza el recorrido; cada nodo de los árboles binarios contienen una entrada, salida o salida condicional según sea el caso y un campo de posición y liga que será utilizado para marcar el *bit* del campo correspondiente en la tabla de estados de manera fácil y sencilla.

Al concluir de generar los renglones de la tabla de estados, el siguiente paso es, obtener las funciones booleanas; cabe mencionar, que cada renglon es un conjunto de elementos lógicos; 1, 0 y * -donde 1 = activo, 0 = no activo y * no importa-. Para la generación de las funciones booleanas, se toma en cuenta los 1 lógicos que aparecen en los campos de los renglones de la tabla de estados, donde dichos campos proporcionan la información necesaria y suficiente para la generación de la funciones boolenas: *estado siguiente, salida en estado presente y salida condicional*. Para la obtención de las funciones booleanas se empleo una estructura dinámica en combinación con un arreglo como se ilustra en la figura 4.21. dicha estructura se utilizá para la generación de la *función estado siguiente y salida condicional*.

Después de generadas las estructuras dinámicas asociadas a las funciones de estado siguiente, salida en estado presente y salida

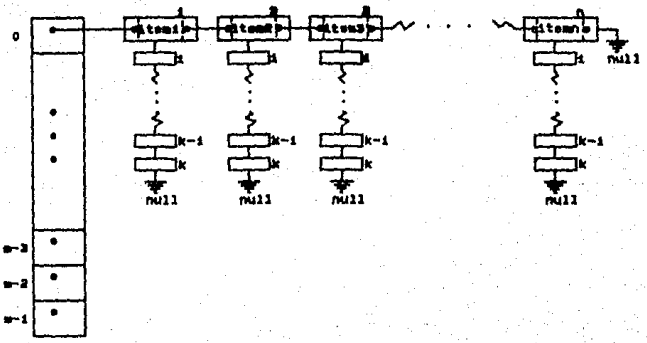


Figure 4.21

condicional, el siguiente paso es la reducción de dichas funciones, para lo cual se empleo el método de Quine-McCluskey (explicado a gran detalle en IV.2.1.2.2). Para el empleo de este método se utilizan estructuras dinámicas en combinación con listas de liga simple, después de aplicado el método se obtienen las estructuras dinámicas asociadas a: *estado siguiente, salida en estado presente y salida condicional* finalmente se procede a realizar una relación *uno a uno* entre cada *bit* y la información que se encuentra asociada a éste(nombre de la entrada, salida y salida condicional) obteniendose en pantalla las funciones boolenas que caracterizan el comportamiento funcional de la carta ASM editada.

Finalmente para la generación del archivo con el mapa de fusibles en formato JEDEC, se utiliza el programa *PLAN,* el cual además presenta el tipo de dispositivo PAL requerido (el más óptimo) para sus programación. El funcionamiento del programa *PLAN* es el siguiente; emplea las funciones booleanas generadas por el sistema *AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADO A PAL's* . De está manera, se lleva a cabo la generación del archivo en formato JEDEC, el cual es utilizado por un dispositivo programador, para proceder a la quema de fusibles del dispositivo PAL. Es importante mencionar, que el programa *PLAN* utilizado, no forma parte del desarrollo del sistema presentado.

Cartas ASM

automata
carta
carta1.pal
carta2.pal
carta3.pal
list.fil
PAL00001.rtgdf
terea.pal

Directorio Tablas de estado Herramientas Salirse de sesion

Archivo :

Cartas ASM

automata
carta
carta1.pal
carta2.pal
carta3.pal
PRL00001.rtgdf
terea.pal

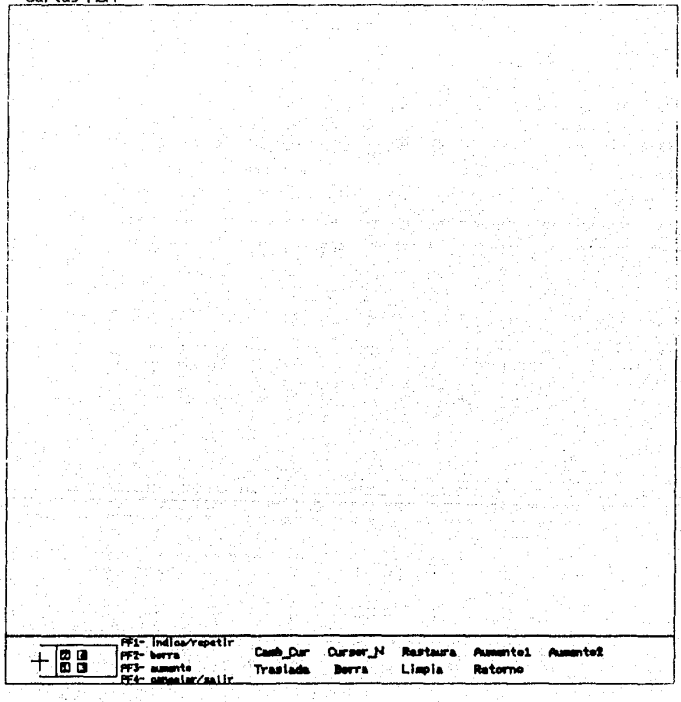
Directorio	Tablas de estado	Herramientas	Salirse de sesion
Borrar archivo	Graficador	Respaldo	Menu Principal

Cartas ASM

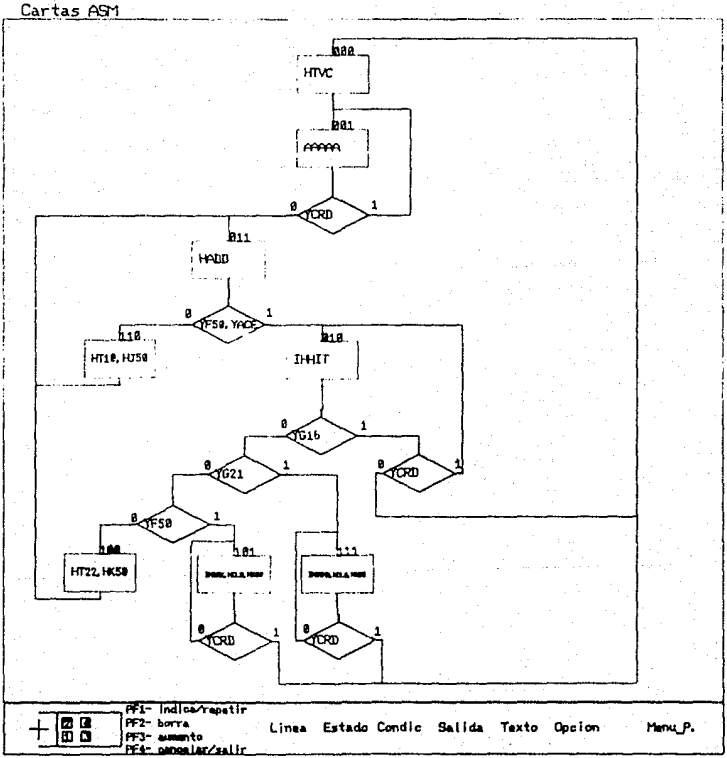
PF1- Indice/repetir
PF2- borra
PF3- aumento
PF4- cancelar/salir

Linea Estado Condic Salida Texto Opcion Menu_P.

Cartas ASM


PF1- Indice/repetir
PF2- borra
PF3- aumento
PF4- ampliar/salir
Cash_Cur
Traslada
Curser_N
Borra
Restaura
Limpia
Aumento1
Retorno
Aumento2

Cartas ASM


000
HTVC
001
AAAAA
0
YCRD
1
011
HADD
0
YF50, YAGE
1
110
HT10, HJ50
010
IHHIT
0
YG16
1
0
YG21
1
0
YCRD
1
0
YF50
1
100
HT22, HK50
101
111
0
YCRD
1
0
YCRD
1
PF1- Indica/repetir
PF2- borra
PF3- aumento
PF4- cancelar/salir
Linea Estado Condic Salida Texto Opcion Menu_P.

Cartas ASM

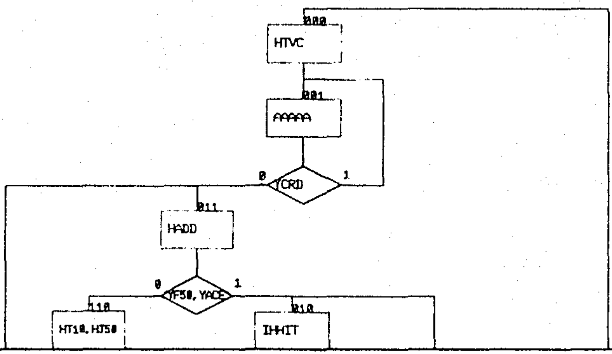


Tabla de Estados

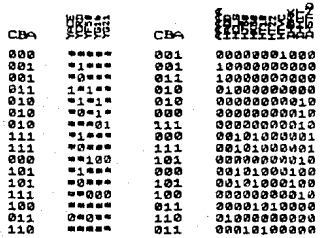
CBA		CBA	
000	*****	001	00000001000
001	*1***	001	10000000000
001	*0***	011	10000000000
011	1*1**	010	01000000000
010	*1*1*	010	00000000010
010	*0*1*	000	00000000010
010	***01	111	00000000010
111	*1***	000	00101000001
111	*0***	111	00101000001
000	**100	101	00000000010
101	*1***	000	00101000100
101	*0***	101	00101000100

Directorio Tablas de estado Herramientas Salirse de sesion

Tablas de Estados Funciones Bool. Salida

Cartas ASM

Tabla de Estados



CBA		CBA	
000	*=****	001	0000000010000
001	*1****	001	1000000000000
001	*0****	011	1000000070000
011	1*1***	010	0100000000000
010	*1*1**	010	0000000000010
010	*0*1**	000	0000000000100
010	***01	111	0000000000010
111	*1****	000	0010100000001
111	*0****	111	0010100040001
000	***100	101	0000000000010
101	*1***0	000	0010100030100
101	*0***0	101	0010100010100
111	**000	100	0000000000010
100	******	011	0000101000000
011	0*0**0	110	0100000000000
110	******	011	0000101000000

Directorio	Tablas de estado	Herramientas	Salirse de sesion
Tablas de Estados	Funciones Bool.		Salida

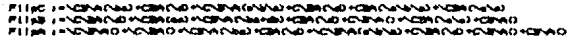
Cartas ASM

ECUACIONES BOOLEANAS

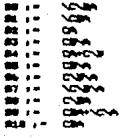
Relacion de variables

Entradas | Salidas | Salidas condicionales

Funciones de Estado Siguiente



Salidas de Estado Presente



Salidas Condicionales

Directorio	Tablas de estado	Herramientas	Salirse de sesion
Tablas de Estados	Funciones Bool.	Salida	

CONCLUSIONES

El sistema *AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADO A PAL's*, se diseño de tal manera que, cuando se requiera emigrar el software a un hardware diferente al que se utilizó para su creación, sea necesarios realizar pequeñas correciones como: asociación a diferentes dispositivos de control o una estación de trabajo diferente a la utilizada, debido a que se emplearon técnicas de programación que contemplan su funcionamiento bajo cualquier otro tipo de lenguaje de alto nivel C. De tal manera, es importante recalcar, que el sistema funciona y se desarrollo bajo técnicas de vanguardia como: uso de estaciones de trabajo 5081, sistema operativo AIX 2.1 (vease Unix) así como la totalidad del software fue desarrollado en lenguaje de alto nivel C estandard, además esta orientado a la utilización de dispositovos PAL.

Los problemas que involucro el desarrollo del sistema y que se presentaron más frecuentemente, requerian el uso de algoritmos para programación dinámica y de listas lineales ligadas. Dichos algoritmos se utilizaron para el almacenamiento de información al momento de generación de la carta ASM, así como posteriormente la recuperación de la información gráfica y no gráfica para su procesamiento, la cual se almacena en disco duro. Con lo que respecta a la generación de la tabla de estados que se obtiene a partir de la información no gráfica que contiene la carta ASM, se utilizó listas lineales para tener acceso, modificación y manipulación de los item de importancia para la creación de dicha tabla. De manera similar, dentro del proceso de generación de funciones booleanas se requirió de algoritmos de manejo dinámico de memoria para la generación de los minterminos y reducción de éstos, para finalmente llegar a la presentación de funciones booleanas en forma óptima.

La desventaja que presenta el sistema, concierne al uso funcional, es decir, para su funcionamiento se requiere una configuración básica como la que se muestra en la figura 4.2.1, por tal motivo se convierte en un sistema elitista.

Empero, debido a la tendencia de las aplicaciones enfocadas a técnicas CAD, el sistema *AMBIENTE CAD PARA DISEÑO DE SISTEMAS DIGITALES ORIENTADO A PAL's* se creo contemplando el uso de un sistema *propio* para la automatización de procesos de diseño lógico.

De lo antes descrito se puede concluir que el sistema presenta las siguientes ventajas con respecto a las restricciones de uso: no existe restricciones para la creación de estados en una carta ASM, se pueden definir 50 variables de entrada, 50 variables para salida, para la generación de las función estado-siguiente, se baso en la utilización de la tabla de excitación de flip-flops D por el uso de las macroceldas de salida que presentan los dispositivos PAL.

Finalmente, el sistema es de fácil manipulación, fiable en todos sus procesos, por lo tanto, se sometera a una serie de pruebas con problemas de diferentes usuarios con la finalidad de detectar posibles fallas.

Referencias

[1] Kaye, D., *Sistemas booleanos*, Ed. alhambra, madrid, 1982, pp 131-164.

[2] Clare Christopher R., *Designing logic systems using state machines*, Ed. Mc Graw-Hill book company, 1973.

[3] Altera Corp., *Applications (user-configurable logic)*, Handbook, julio, 1988.

[4] Altera Corp., *User-configurable logic*, Databook, septiembre, 1988.

[5] Downs Thomas, Shulz Mark F., *Logic design with pascal*, Computer-Aided Design Techniques, 1988.

[6] Advanced Micro Devices, *Programmable logic*, Handbook/Databook, 1986-1987.

[7] Monolithic Memories Corp., *Pal/Ple Device programmable logic array*, 1985.

[8] IBM Corp., *The graPHIGS programming interface*, writing applications, september 1986.

[9] IBM Corp., *The graPHIGS programming interface*, subroutine reference, release 3.4, march 1988.

[10] IBM Corp., *Advanced interactive executive (AIX 2.1)*, technical reference, Volume 1, january 1987.

[11] IBM Corp., *Advanced interactive executive (AIX 2.1)*, technical reference, Volume 2, January 1987.

[12] Aldana Mayor F., Esparza Olcina R., *Electronica industrial: Técnicas digitales*, Marcombo Boixareu Ed., España, 1980.

[13] IBM Corp., *the graPHIGS programming interface*, Understanding concepts, release 3.4, march 1988.

[14] IBM Corp., *graPHIGS*, Messages and error codes, release 3.4, march 1988.

[15] Hearn Donald. Baker Pauline M., *Gráficas por computadora*, ed. Prentice Hall, México, 1989.

[16] Knurt Donald E., *Algoritmos Fundamentales*, Volumen 1, ed. Reverte, barcelona, 1980, pp 253-341.

[17] R.G.Bennetts. J.L.Washington. D.W.Lewin., *A Computer Algorithm for State Table Reduction*, based on a paper presented at the IEEE 'Eurocom 71', Switzerland, october 1971.

[18] L.Edward A. Wai-Hung Ho. C.Edwin E. B.Jeffrey C. and S.Bhattacharyya, *Design Environment for DSP*, IEEE transactions on Acoustic, vol.37, núm.11, november 1989.

[19] L.G.Paul. B.Albert. B.Patricia. G.Thierry, *A Data Flow-oriented Language for Signal processing*, IEEE transactions on acoustics, vol.assp.34, núm.2, april 1986.

[20] H.T. Mouftah. K.S.Shanmugan., *Computer-Aided Techniques for Communications Systems Engineering*, july 1987, vol.25, núm.7, IEEE Communications Magazine.

[21] Wirth. Niclaus., *Algorithms + Data structures = Programs*, Ed. Prentice-Hall, new jersey, USA, 1976.