UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"DISEÑO DE LA UNIDAD ARITMETICA DE LA ELECTRONICA DE UN DETECTOR TIPO MEPSICRON"

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

ADRIAN HERNANDEZ ALVA

ASESOR: M.C. JOSE LUIS PEREZ SILVA

MEXICO, D. F.

1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Con mi cariño y agradecimiento a :

MAURA ALVA DUARTE (mi madre)
porqué sin sus sacrificios y desvelos, esto no sería posible

mi FAMILIA
quienes a pesar de sus limitaciones, siempre me apoyaron

NORMA ISABEL ORTEGA MARTINEZ
a quien admiro y respeto por su valor y amor a la vida.

INDICE

1 INTRODUCCION

1.1	Antecedentes	1.2
1.2	Objetivos	1.3
1.3	Metodología	1.3

2 EL DETECTOR MEPSICRON

2.1	El detector MEPSICRON	2.1
2.2	Operación del detector MEPSICRON	2.2
2.2.1	Transductor	2.3
2.2.2	Acondicionamiento de señal	2.3
2.2.3	Interfaz	2.4
2.2.4	Procesamiento de la información	2.4
2.2.5	Módulo de control	2.5
2.3	Máxima velocidad de muestreo	2.6

3 DISPOSITIVOS PROGRAMABLES PARA LOGICA COMPLEJA

3.1	Dispositivos programables para lógica compleja	3.1
3.1.1	Familias de CPLDs	3.4
3.1.2	Arquitectura interna de los CPLDs	3.5
3.2	El Sistema de desarrollo de lógica programable	3.8
3.2.1	Métodos de captura	3.9
3.2.2	El compilador	3.16
3.2.3	Simulador y Analizador de tiempos de tránsito	3.19
3.2.4	El programador	3.19

4 DESARROLLO DEL SISTEMA DIGITAL

4.1	Sistemas Digitales	4.1
4.2	Diseño de circuitos digitales empleando un PLDS	4.3
4.3	Características del sistema digital	4.7
	4.3.1 Elementos necesarios para desarrollar el sistema digital	4.9
4.4	Investigación de algoritmos	4.11
	4.4.1 Adición y sustracción binarias	4.11
	4.4.2 División binaria	4.28
4.5	Construcción del Sistema	4.48
	4.5.1 Primer prototipo (MEPSI1)	4.48
	4.5.2 Segundo prototipo (MEPSI3)	4.53
	4.5.3 Diseño final (MEPSI5)	4.58
4.6	Características finales (resultados)	4.66

5 CONCLUSIONES 5.1

AGRADECIMIENTOS i

BIBLIOGRAFIA ii

1 INTRODUCCION

Las actividades relacionadas con la ingeniería incluyen el diseño, la invención y el perfeccionamiento de herramientas y sistemas; los ingenieros aplican los conocimientos científicos y técnicos para determinar las condiciones óptimas para la realización y funcionamiento de un proyecto. Este documento corresponde a las memorias de un proyecto de investigación en el que fui partícipe; considerando que esas actividades corresponden a las desarrolladas por un ingeniero, hoy presento este documento como la parte escrita de mi examen profesional.

1.1 ANTECEDENTES

Actualmente en el Instituto de Astronomía y en el Centro de Instrumentos de la UNAM, se realizan estudios orientados a lograr el funcionamiento óptimo de un sistema diseñado para realizar investigaciones astronómicas llamado MEPSICRON.

El detector MEPSICRON es un sistema que registra las coordenadas de los puntos de impacto de fotones sobre una superficie; la información obtenida por medio de un transductor, es acondicionada para ser posteriormente procesada y almacenada por una computadora personal.

Originalmente, el cálculo de coordenadas era realizado por un programa en la computadora del sistema; sin embargo debido a las múltiples tareas que ésta debe realizar, la operación del detector se limita a aproximadamente 280,000 eventos por segundo, en tanto que el módulo de adquisición de datos fue diseñado para procesar la información correspondiente a un millón de eventos por segundo.

Se decidió realizar el algoritmo antes mencionado en un módulo independiente (un sistema digital); las dimensiones físicas del sistema de adquisición de datos del detector MEPSICRON, imponen el uso de dispositivos de muy reducidas dimensiones, esa limitación elimina la posibilidad de uso de elementos discretos (circuitos integrados MSI). Por lo anterior, para la construcción de ese módulo se consideró usar dispositivos electrónicos de alta velocidad y alta escala de integración, tales como Dispositivos Programables para Lógica Compleja (CPLD's), y Procesadores Digitales de Señales (DSP's).

En el Laboratorio de Electrónica del Centro de Instrumentos, se han diseñado y construido varios de los elementos que constituyen el detector MEPSICRON; este departamento cuenta con un sistema de desarrollo de lógica programable, por lo que en este lugar se evaluó la posibilidad de construir el módulo antes mencionado, haciendo uso de ese sistema.

El diseño del circuito digital y el desarrollo del mismo usando el PLDS, me fueron asignados; en este documento se registra la forma en que diseñe el circuito digital, presento los algoritmos utilizados e indico algunos de los elementos y procedimientos que deben tomarse en cuenta al trabajar con un Sistema de Desarrollo de Lógica Programable.

1.2 OBJETIVOS :

Diseñar un sistema digital electrónico, que permita optimizar el proceso de adquisición de datos del detector MEPSICRON.

Este circuito digital deberá sustituir al programa que realiza el cálculo de coordenadas en el detector MEPSICRON y contar con las siguientes características :

- debe ser diseñado mediante el empleo de un sistema de desarrollo de lógica programable
- el tiempo máximo de tránsito debe ser de un microsegundo
- debe ocupar el mínimo espacio posible.

1.3 METODOLOGIA USADA

1. Conocer los elementos que forman el detector MEPSICRON y el funcionamiento del mismo; esto permitió determinar las características del circuito a diseñar (capítulo 2)

2. Estudiar las características del Sistema de Desarrollo de Lógica Programable para conocer las ventajas y desventajas del mismo (capítulo 3)

3. Diseño del sistema digital

- identificación de los elementos a diseñar y sus características
- investigación de algoritmos
- evaluación de alternativas
- construcción del sistema
- pruebas al sistema
- acciones para lograr el desempeño óptimo
- especificar las características finales

(capítulo 4)

4. CONCLUSIONES
(capítulo 5)

2 EL DETECTOR MEPSICRON

2.1 ORIGEN DEL DETECTOR MEPSICRON

La historia ha registrado detallados estudios de los cuerpos celestes realizados por antiguas civilizaciones como la asiria, la egipcia y la maya, también se registran grandes aportaciones individuales como es el caso de la invención del telescopio por Galileo Galilei, y del análisis del movimiento de los planetas realizado por Kepler; recientemente los adelantos técnicos han permitido los viajes espaciales, la construcción de radiotelescopios y colocar en órbita un telescopio; lo anterior demuestra la fascinación que el hombre siente por el espacio.

La Universidad Nacional Autónoma de México es una de las instituciones que promueve el desarrollo y la investigación espacial en nuestro país, el Programa Universitario de Investigación y Desarrollo Espacial (PUIDE), en el que se construyó el primer satélite mexicano; y el detector bidimensional de fotones MEPSICRON, son dos de los proyectos más ambiciosos con los que la UNAM participa en este campo de investigación .

El detector MEPSICRON es el resultado del trabajo conjunto entre el Instituto de Astronomía y el Centro de Instrumentos. MEPSICRON es un sistema diseñado para detectar los fotones que inciden sobre una superficie, y determinar las coordenadas del punto de impacto con respecto a un sistema de referencia bidimensional, cuyo origen se fijó arbitrariamente en el centro geométrico de dicha superficie.

El objetivo del detector MEPSICRON es registrar eventos relacionados con la evolución de los cuerpos celestes; por sus características es capaz de realizar en minutos, registros astronómicos que los procesos fotográficos realizan en horas; el sistema esta formado por un transductor (fotosensor), un acondicionador de señales, una computadora personal y un conjunto de programas que procesan la información (determinan las coordenadas, la despliegan en pantalla y la almacenan para su posterior análisis).

2.2 OPERACION DEL MEPSICRON

Originalmente el detector MEPSICRON constó de los siguientes módulos:

- Un transductor, constituido por :

  a) un fotocátodo multialcalino que se deposita sobre una placa de cuarzo
  b) una serie de cinco placas microcanal
  c) un ánodo resistivo de baja distorsión

- Un acondicionador de señales formado por :

  a)Cuatro sistemas de preamplificación con conversión carga-voltaje
  b)Cuatro convertidores analógicos-digitales

- Un grupo de registros como interfaz entre el módulo acondicionador de señales y un puerto paralelo de una computadora personal

- Una computadora personal y un conjunto de programas que realizan las siguientes actividades :

  a) lectura de 48 bits de información del puerto en paralelo
  b) cálculo de las coordenadas del punto de impacto en el sensor
  c) presentar la imagen en 1000 X 1000 pixeles y que posteriormente guardar la imagen en un dispositivo de memoria.

- Un circuito de control que permite el adecuado flujo de información entre los diferentes módulos del sistema.

2.2.1 TRANSDUCTOR

Al incidir un fotón sobre el fotocátodo, se desprende un electrón, éste es acelerado por medio de un campo eléctrico externo, llevándolo hasta la primera placa microcanal, éstas placas estan constituidas por microcanales capilares multiplicadores de electrones, de cuyas paredes, se desprenden electrones secundarios; esos electrones secundarios repiten el proceso en las siguientes placas microcanal, hasta que finalmente se obtiene un haz del orden de 100 millones de electrones que al salir de la última placa microcanal chocan contra el ánodo resistivo; siendo esa cantidad de carga suficiente para procesar las señales.

El ánodo resistivo tiene en sus cuatro vértices, conexiones para instalar el circuito electrónico que permite acondicionar la señal para su posterior procesamiento.

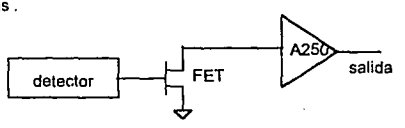
2.2.2 ACONDICIONAMIENTO DE SEÑAL

El acondicionamiento de señal implica tres procesos :

a) Preamplificación
b) Conversión carga-voltaje
c) Conversión analógica-digital

El dispositivo seleccionado para realizar la preamplificación fue el A250 fabricado por AMPTEK, éste es un dispositivo capaz de realizar también la conversión carga-voltaje. Por sus características es un circuito que satisface muchos requerimientos para aplicaciones de laboratorio, por lo que es usado en investigaciones nucleares y espaciales. Este dispositivo ofrece excelentes características de linealidad y rechazo a señales de ruido.

En el diagrama esquemático se presenta una forma de aplicación típica, en donde se utiliza un transistor de efecto de campo (FET) para acoplar impedancias .



La siguiente etapa del acondicionador realiza la conversión analógica-digital de la señal entregada por los preamplificadores; dicha conversión debe cumplir con las siguientes características :

- tiempo de conversión del orden de 1 microsegundo
- resolución en doce bits
- bajo nivel de ruido

Con base en lo anterior se eligió el circuito integrado AD9003; éste dispositivo es un convertidor analógico-digital con una resolución en doce bits, y con reloj integrado. fabricado por Analog-Devices.

Este dispositivo tiene la capacidad de realizar un millón de conversiones por segundo, incluyendo el tiempo necesario para recibir y generar las señales de control.

2.2.3 INTERFAZ

Este módulo esta formado por un grupo de registros con salida del tipo tres estados; la información que entregan los convertidores, es almacenada en los registros durante el tiempo necesario para que la computadora la almacene en alguno de sus dispositivos de memoria; las salidas tres estados permiten la transferencia de información a la computadora en forma secuencial.

2.2.4 PROCESAMIENTO DE LA INFORMACION

Posteriormente la información debe ser procesada para determinar las coordenadas del punto donde incidió el fotón.

Cuando se obtiene las coordenadas de un evento, la información se almacena de tal forma que se puede usar en el momento que se requiera, ya sea para ser presentada en pantalla (1024 x 1024 pixeles), o para ser analizada.

2.2.5 MODULO DE CONTROL

El módulo de control del sistema tiene a su cargo las siguientes tareas :

- Reconocer eventos
- Controlar el funcionamiento del convertidor A/D y de los preamplificadores
- Permitir el flujo de información en los registros de salida
- Mantener el protocolo de comunicación entre el módulo de adquisición de datos y la computadora.

Para reconocer que se ha producido el impacto de un fotón en la superficie del sensor, se utiliza un circuito sumador construido a partir de un amplificador operacional de alta velocidad; las entradas a ese circuito son las señales de salida de los preamplificadores; en el momento en el cual la salida del sumador ha superado un valor considerado como ruido se activan los convertidores analógicos-digitales.

Posteriormente, se permite que el convertidor A/D procese dichas señales. El convertidor envía una señal en el instante en el cual se ha concluido la conversión; cuando el control la recibe, activa los registros de salida para mantener la información en tanto la computadora puede recibirla, además se inicia el protocolo de comunicación para la transferencia de datos con la computadora, via el puerto paralelo.

La forma en la que la computadora procesa la información, hace necesario bloquear el funcionamiento de la etapa de preamplificación hasta que la computadora termine de procesar y almacenar un paquete de información.

En el instante que la computadora puede recibir nueva información, ésta genera una señal que le indica al circuito de control, que puede iniciarse nuevamente el proceso; habilitándose nuevamente a los preamplificadores. El tiempo que transcurre desde el instante del impacto del fotón hasta que el control le indica a la computadora que existen datos válidos es de 1.02 µs.

2.3 MAXIMA VELOCIDAD DE MUESTRO DEL DETECTOR

Originalmente, el cálculo de coordenadas era realizado por un programa en la computadora del sistema, sin embargo debido a las múltiples tareas que ésta debe realizar, la operación del detector se limita a aproximadamente 280,000 eventos por segundo. Lo anterior implica que se subutiliza el detector, ya que en las condiciones óptimas, se deben registrar un millón de eventos en ese mismo período de tiempo.

Considerando este problema se decidió desarrollar el algoritmo en un módulo independiente, utilizando dispositivos electrónicos de alta velocidad y alta escala de integración como los Dispositivos Programables para Lógica Compleja (CPLDs) o los Procesadores Digitales de Señales (DSPs).

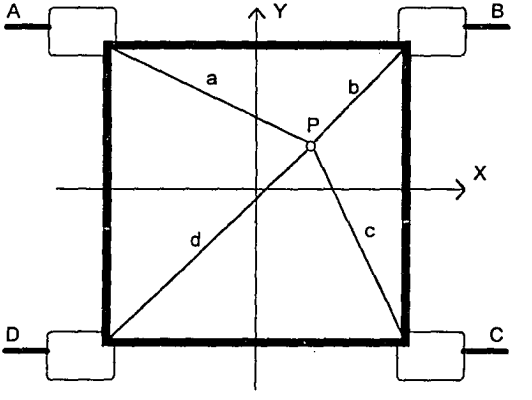


Figura 2.1 Esquema del transductor del detector MEPSICRON

En la figura 2.1 a,b,c y d representan la mínima distancia entre el punto de impacto P y los vertices del transductor, en los cuales se encuentran parte de los acondicionadores de señal. A, B, C y D representan las señales generadas por el transductor del sistema.

Las ecuaciones que permiten conocer las coordenadas del punto de impacto son :

$$X = \left(\frac{((B+C)-(A+D))}{A+B+C+D} \right)$$

$$Y = \left(\frac{((A+B)-(C+D))}{A+B+C+D} \right)$$

Otra de las acciones orientadas a mejorar el proceso de adquisición de datos, es usar una computadora con un bus de datos de treinta y dos bits, con lo que se puede realizar la transferencia de información en menor tiempo. Esto se debe considerar al realizar el diseño del módulo antes mencionado.

3 DISPOSITIVOS PROGRAMABLES PARA LOGICA COMPLEJA

3.1 INTRODUCCION A LOS DISPOSITIVOS PROGRAMABLES PARA LOGICA COMPLEJA

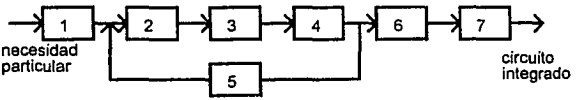
Los Dispositivos Progamables para Lógica Compleja (CPLD's), son circuitos integrados que el usuario puede configurar para desarrollar sistemas digitales complejos que resuelvan sus necesidades particulares, en estos circuitos es posible crear funciones lógicas combinacionales y circuitos secuenciales.

Estos dispositivos son una alternativa a los circuitos integrados para aplicación específica tradicionales como los dispositivos GATE ARRAY y los Arreglos Lógicos Programables (PALs); los CPLDs se distinguen por combinar el buen desempeño trabajando a altas frecuencias de los dispositivos PLA con la alta escala de integración de los GATE ARRAY, además permiten mayor flexibilidad para el diseño.

Los CPLD's permiten desarrollar en nuestro propio laboratorio, un circuito integrado digital, evitando prolongados tiempos de espera, altos costos, condiciones no rentables e incluso el riesgo de plagio de una idea original situaciones que pueden presentarse cuando se desarrolla un diseño empleando la técnica GATE ARRAY, y se solicita a determinada empresa que lo fabrique.

Un Sistema de Desarrollo de Lógica Programable (PLDS) es un conjunto de programas (software) que permiten configurar los dispositivos PLD. Estos programas permiten registrar las características del diseño (captura), ayudan a optimizar, a depurar errores, simular su funcionamiento y finalmente permiten "programar" el circuito en un PLD.

La idea original de estos sistemas es facilitar el desarrollo de circuitos integrados digitales para aplicaciones específicas; de tal forma que sea posible obtener un circuito de éste tipo en unas cuantas horas, aprovechando los adelantos técnicos en los campos de la computación y en el desarrollo de circuitos CMOS.



1) concepción del diseño
2) captura
3) compilación
4) simulación
5) correcciones y mejoras
6) "programación" del dispositivo
7) pruebas al dispositivo

Figura 3.1 Proceso de desarrollo de un circuito integrado para una aplicación específica

El sistema que fue utilizado en el presente trabajo es llamado MAX2plus, desarrollado por "ALTERA", este fabricante ofrece una gran variedad de CPLDs.

De aquí en adelante se describiran las características de los dispositivos y el sistema de desarrollo que este fabricante ofrece.

3.1.1 FAMILIAS DE CPLDs

Altera ofrece cinco familias de dispositivos programables para lógica compleja (CPLDs) :

La familia CLASSIC que incluye circuitos integrados con 20 y hasta 68 terminales (pines), requieren poca potencia de mantenimiento, incluyen hasta mil ochocientas compuertas y permiten frecuencias de trabajo de hasta 125 MHz.

La familia MAX5000 que incluye desde 20 y hasta 100 terminales por circuito integrado, combinan la alta velocidad y el fácil uso de los dispositivos PAL's con altas densidades de integración. Esta familia puede sustituir de 20 a 25 PAL's o aproximadamente 100 funciones MSI dentro de un solo encapsulado (7500 compuertas) y trabajan a frecuencias próximas a los 110 MHz.

La familia MAX7000 que representa la segunda generación de la arquitectura MAX, estos dispositivos se encuentran disponibles en encapsulados que cuentan con 44 y hasta 288 terminales, sus recursos internos llegan a equivaler a 10,000 compuertas lógicas por circuito integrado y pueden trabajar a frecuencias cercanas a 125 MHz.

Las tres familias antes mencionadas, corresponden a dispositivos borrables; es decir se pueden "programar" más de una vez. Por lo que en los manuales, sus codigos aparecen precedidos de una E (borrable por medio de luz ultravioleta) o de EE (borrable electricamente), por ejemplo EPMAX5128 o EEMAX7192

La familia MPLD (máscaras programables) que proporcionan una alternativa de bajo costo para altos volúmenes de producción.

Existen dispositivos denominanados de aplicación especifica, como el EPS448 SAM EPLD (un microsecuenciador configurable por el usuario).

3.1.2 ARQUITECTURA INTERNA DE LOS CPLDs

Un CPLD está formado por bloques llamados MACROCELDAS, (figura 3.2), estos bloques son arreglos de circuitos digitales elementales, constituídas por :

a) Un arreglo lógico, en donde se crean todas las funciones lógicas-combinacionales necesarias para el diseño

b) Un registro programable, que puede ser configurado como tipo D,T,JK o SR

c) Una terminal programable que puede configurarse como entrada, salida o como terminal bidireccional según lo requiera el diseño.

EL ARREGLO LOGICO

Las terminales de entrada del dispositivo CPLD estan conectadas a las entradas de las macroceldas, en estas se genera (mediante inversores), el valor complemento de cada una de las entradas, posteriormente cada una de esas variables se conecta mediante un eslabón programable (elementos EPROM o EEPROM) a las entradas de un arreglo de compuertas AND de tal forma que es posible generar términos producto de las variables de entrada; las salidas del arreglo AND se conectan también mediante eslabones programables a un arreglo de compuertas OR, de manera que es posible generar los términos suma necesarios. Como se puede observar, la estructura de este módulo es idéntica a la un dispositivo PAL. Y en él, el sistema de desarrollo crea las funciones lógicas combinacionales que sean necesarias.

La salida de la compuerta OR, se encuentra conectada a una de las dos entradas de una compuerta XOR, la entrada restante está conectada a un eslabón programable, el cual permite que el sistema pueda usar lógica positiva o negativa. Además permite simplificar las estructuras lógicas mediante el empleo de los teoremas de De Morgan. Mediante el uso de los teoremas de De Morgan, un término suma puede ser sustituido por términos producto. en la figura 3.2 se observa que la cantidad de recursos para crear términos producto es mucho mayor que la existente para crear términos suma, por lo que se puede apreciar la importancia de este recurso.

La cantidad de recursos lógicos combinacionales contenidos en una macrocelda depende de cada tipo de dispositivo, por ejemplo, en el dispositivo EP1810 los recursos de una macrocelda equivalen a aproximadamente cuarenta compuertas NAND, en tanto que en un dispositivo EPM5128, cada macrocelda equivale a treinta y dos compuertas NAND.

REGISTRO PROGRAMABLE

Los registros programables son utilizados para crear una gran variedad de circuitos secuenciales usando un mínimo de recursos del dispositivo CPLD, cada registro programable puede ser configurado como un flip-flop de tipo D,T,JK o SR, estos registros se activan con el flanco de subida de la señal de reloj, si en un diseño no es necesario este tipo de elementos, simplemente son ignorados mejorando así el tiempo de respuesta.

TERMINALES PROGRAMABLES

Las terminales programables estan formadas por circuitos de tres estados, que son controlados por medio de un circuito especial, estos elementos pueden ser configurados como entradas, salidas o como terminales bidireccionales, además se cuentan con al menos dos líneas de realimentación por cada Macrocelda, de tal forma que estas conexiones facilitan la construcción de circuitos como contadores, registros de corrimiento y máquinas de estados.

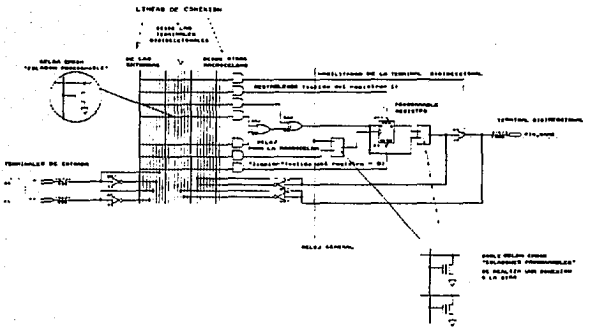


Figura 3.2. CELDA LOGICA (macrocelda)

3.2 SISTEMA DE DESARROLLO DE LOGICA PROGRAMABLE

Este sistema de desarrollo (conjunto de programas), es una poderosa herramienta para cualquier diseñador de sistemas digitales, proporciona sencillos métodos de captura, evaluación y optimación de sistemas digitales.

El sistema de desarrollo de lógica programable usado permite las siguientes actividades :

a) Capturar el diseño de circuitos digitales

b) Ensamblar y relacionar adecuadamente los archivos que constituyen un proyecto, indicando errores en declaraciones y/o construcción

c) Simular el comportamiento del sistema

d) Analizar el tiempo de tránsito de cada salida con respecto a cada una de las entradas

e) "Programar" el diseño en un dispositivo CPLD.

3.2.1 METODOS DE CAPTURA

EL EDITOR DE TEXTOS

El editor de textos permite introducir diseños lógicos que se describen mediante un lenguaje de programación especializado llamado Lenguaje de Descripción de Hardware Altera (AHDL). Las posibilidades de este lenguaje van más allá de simplemente permitir la descripción de un diagrama lógico; permite el desarrollo directo de diseños a partir de tablas de verdad, ecuaciones booleanas y máquinas de estado.

A continuación se explica brevemente como se estructuran los programas hechos mediante AHDL :

Las secciones que componen un diseño hecho mediante AHDL pueden ser una sección de diseño o una de subdiseño/lógica.

- La declaración del título (opcional) proporciona comentarios para un archivo de reporte que generará un compilador (descrito posteriormente)
- Declaración de constantes (opcional) se especifica un nombre simbólico que pueda sustituir a una constante.
- Declaración de funciones patrón (opcional) se declaran los puertos de entrada y salida de una función incluida en los archivos del sistema de dasarrollo (por ejemplo sumadores, registros de corrimiento o funciones construidas por el usuario).
- Sección de diseño (obligatoria), especifica dispositivos, uso de circuitos de las librerías (es decir, el conjunto de programas que sirven de patrón para resolver determinados problemas), asociaciones, terminales de entrada/salida, asignación de macroceldas y opciones lógicas.
- Sección de subdiseño (obligatoria), declara las entradas. salidas y puertos bidireccionales del diseño.
- La sección de variables (opcional) declara variables que representan y mantienen información interna del diseño.
- La sección lógica (obligatoria) define las operaciones lógicas del diseño.

Si son usados los enunciados y declaraciones opcionales, estos deberán encontrarse en los primeros renglones del archivo tipo texto (TDF).

Las últimas entradas en un TDF deberán ser las secciones de subdiseño, sección lógica y la sección de variables (opcional); las que en conjunto describen el comportamiento del diseño.

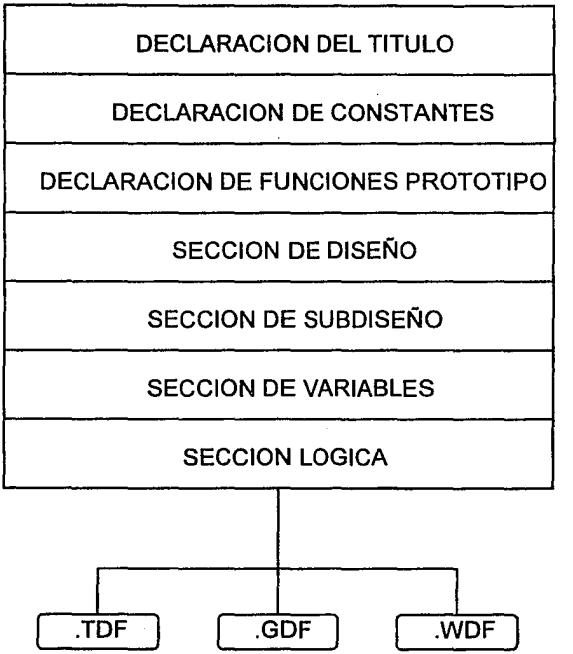


Figura 3.3 Estructura de un diseño utilizando el editor de textos

Los archivos correspondientes a subdiseños del diseño que los contiene, ya sean del tipo texto (TDF), gráfico (GDF) y/o señales digitales (WDF), son relacionados con el archivo de más alto nivel a través de la sección lógica.

La figura 3.4 es un ejemplo de diseño utilizando el editor de textos.

```
% este archivo genera un registro de corrimiento bidireccional%
% tomando como función patrón el registro de corrimiento 74194%


% declaracion de la función patrón, indicando entradas y salidas %
FUNCTION 74194 (SLSI,SRSI,A,B,C,D,S0,S1,CLRN,CLK) RETURNS (QA,QB,QC,QD);

SUBDESIGN rl9412    % nombre del circuito %

(
SLSI,SRSI,D[11..0],S0,S1,CLRN,CLK  : INPUT;  %entradas %
Q[10..0],QM                :OUTPUT;  %salidas%
)

% seccion de variables %
VARIABLE
rc[2..0]                   : 74194;      % nombre que usaran las %
                                         % las funciones patrón %


BEGIN          % declaración de inicio de la sección lógica %

% asociaciones entre los elementos del circuito y los de la función patrón %

% entradas %

rc2.(SLSI,SRSI,S0,S1,CLRN,CLK) = (SLSI,SRSI,S0,S1,CLRN,CLK);
rc1.(SLSI,SRSI,S0,S1,CLRN,CLK) = (SLSI,SRSI,S0,S1,CLRN,CLK);
rc0.(SLSI,SRSI,S0,S1,CLRN,CLK) = (SLSI,SRSI,S0,S1,CLRN,CLK);
rc2.(A,B,C,D) = D[11..8];
rc1.(A,B,C,D) = D[7..4];
rc0.(A,B,C,D) = D[3..0];
rc1.(SRSI) = rc2.(QD);
rc0.(SRSI) = rc1.(QD);
rc2.(SLSI) = rc1.(QA);
rc1.(SLSI) = rc0.(QA);

% salidas %

Q[10..8] = rc2.(QB,QC,QD);
Q[7..4] = rc1.(QA,QB,QC,QD);
Q[3..0] = rc0.(QA,QB,QC,QD);
QM = rc2.(QA);

END;  % fin de la sección lógica %
```

Figura 3.4 Ejemplo de diseño empleando el editor de textos

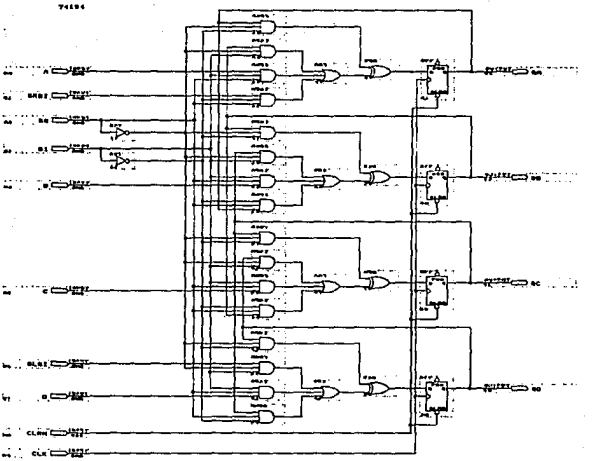
EDITOR DE GRAFICOS

En el editor de gráficos la captura de un diseño se realiza con base en una extensa librería de esquemas de compuertas lógicas, cada uno de esos esquematicos es llamado un "PRIMITIVO"; además existe una gran cantidad de "MACROFUNCIONES" (archivos que pueden ser usados como patrón) que incluyen un gran número de funciones LSI y MSI, tales como sumadores, comparadores, multiplexores y contadores, dentro de un ambiente desde el cual se invocan los elementos que sean necesarios para construir el circuito deseado, posteriormente por medio de líneas que representan el alambrado, se realizan las conexiones requeridas hasta terminar un diseño.

Este tipo de captura es sencillo y rápido debido a que resulta familiar para toda persona que ha trabajado con diagramas de circuitos lógicos, como es el caso de los estudiantes de ingeniería, y en particular para aquellos que han utilizado paquetes como ORCAD, WORKBENCH o SPICE; en la figura 3.5 se muestra el diseño de un circuito creado mediante el editor de gráficos.

Es importante señalar, que una vez que se termina la captura de un diseño, el sistema de desarrollo lo archiva como una macrofuncion más, y a partir de ese momento el diseño se puede invocar como un subdiseño o elemento de un circuito digital más complejo.

Cabe mencionar que el sistema impone ciertas reglas para la correcta utilización de esta herramienta; sin embargo, dichas reglas no se mencionarán en el presente trabajo ya que no se pretende que éste sea una traducción de los manuales del sistema.



Las terminales de entrada siempre aparecen a la izquierda de los elementos
Las terminales de salida siempre aparecen a la derecha de los elementos

Figura 3.5 Ejemplo de diseño usando el editor de gráficos (registro de corrimiento bidireccional (74194))

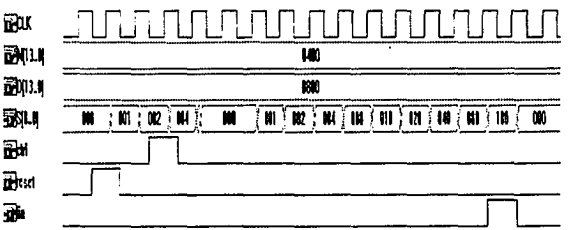
EDITOR GRAFICO DE SEÑALES DIGITALES

Este editor nos permite indicar el comportamiento de las señales de entrada y observar el funcionamiento del circuito, una vez que se ha concluido la captura y se han eliminado todos los errores cometidos.

Mediante un ambiente gráfico podemos indicar el comportamiento de las señales de entrada con respecto al tiempo, y posteriormente observar las salidas que son generadas por el diseño. Básicamente este editor nos presenta una pantalla dividida en renglones, cada uno de los cuales puede representar una entrada, una salida o bien un grupo de ellas.

Como es de esperarse los nodos de entrada pueden adquirir los valores lógicos "0", "1" y "X" (no importa); y los nodos de salida pueden adquirir los valores anteriores y además "Z" (alta impedancia) en el caso de que sea un nodo de salida bidireccional.

Como ya se menciono, este editor es capaz de asociar nodos, con lo cual se forman grupos (bus de datos), haciendo más sencillo el manejo de información. Las figuras 3.6 y 3.7 corresponden al ambiente del editor de señales digitales.



CLK : señal de reloj (entrada)
N[13..0] : representa un grupo de variables digitales (un "bus" de datos) de entrada : N13, N12, N11 ... N0;
S[8..0] : representa un grupo de variables o datos de salida : S8, S7 ... , S0.
ctrl : es una señal de entrada
fin : señal de salida

Figura 3.6 Ambiente del editor de señales digitales

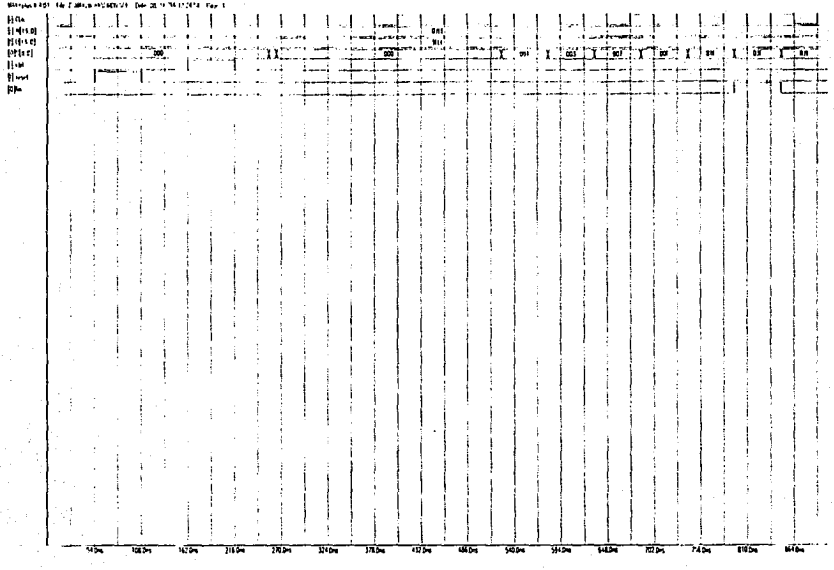


Figura 3.7 Ambiente del editor de señales digitales

3.2.2 EL COMPILADOR

El primer módulo del compilador tiene como tarea el comprobar que el diseño fue construído de acuerdo a las reglas de construcción. Una vez que se han eliminado los errores, se procede a construir una base de datos que representa el funcionamiento del sistema.

El siguiente paso es realizar la síntesis lógica del proyecto; mediante algoritmos especializados se eliminan aquellos elementos que no son utilizados; de esta forma se asegura que se utilizarán adecuadamente los recursos internos del CPLD.

Si por las características del diseño es imposible ajustarlo dentro de un solo CPLD, se invoca al módulo de partición, el cual segmenta el diseño para ser implantado en dos o más CPLDs de la misma familia.

Además se aplica un algoritmo que distribuye los elementos constitutivos del circuito en la estructura interna de los CPLDs, de esta forma se optimizan las rutas críticas (tiempo de propagación).

El compilador genera el archivo reporte del proyecto, en el cual se indica la forma en la que fueron utilizados los recursos del dispositivo CPLD.

Finalmente, el compilador genera los archivos necesarios para la simulación y el análisis de tiempos de respuesta, herramientas que permiten estudiar el comportamiento de nuestro diseño, reconocer fallas y realizar acciones orientadas a lograr la optimación del circuito.

En las siguientes 2 paginas se incluye un ejemplo de archivo-reporte.

```
% ARCHIVO REPORTE DE LA "PROGRAMACION" DE UN CIRCUITO %

Project Information                                   c:\max2work\suma83.rpt

MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 07/05/94 09:01:10 %FECHA DE REALIZACION DEL PROCESO%

***** Project compilation was successful

% RECURSOS UTILIZADOS  %

** DEVICE SUMMARY **

    % DISPOSITIVO%       % TERMINALES %       % PORCENTAJE DE RECURSOS %

Chip/                   Input   Output   Bidir              Shareable
POF       Device        Pins    Pins     Pins   LCs  Expanders % Utilized

suma83    EPM5032       9       5        0      14       32          43 %

User Pins:              9       5        0

Project Information                                   c:\max2work\suma83.rpt

% FUNCIONES PATRON UTILIZADAS %

** FILE HIERARCHY **


|7483:16|

% RESULTADO DE LA COMPILACION %

Device-Specific Information:                          c:\max2work\suma83.rpt
suma83

***** Logic for device 'suma83' compiled without errors.
```

Reporte generado por el PLDS del desarrollo de un diseño.

```
% DIAGRAMA INDICATIVO DE ASIGNACION DE TERMINALES %

Device: EPM5032       Security: OFF

                              EPM5032
                              - - - - -
                        b3 - |1       28| - a1
                        b4 - |2       27| - a2
                  RESERVED - |3       26| - c1
                  RESERVED - |4       25| - co
                  RESERVED - |5       24| - s1
                  RESERVED - |6       23| - s2
                       VCC - |7       22| - VCC
                       GND - |8       21| - GND
                  RESERVED - |9       20| - s3
                  RESERVED - |10      19| - s4
                  RESERVED - |11      18| - RESERVED
                  RESERVED - |12      17| - RESERVED
                        b2 - |13      16| - a3
                        b1 - |14      15| - a4
                              - - - - -

N.C. = Not Connected.
VCC = Dedicated power pin, which MUST be connected to VCC.
GND = Dedicated ground pin or unused dedicated input,
which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.

% REPORTE DE RECURSOS UTILIZADOS %

** RESOURCE USAGE **

                                                        Shareable
Logic Array Block      Logic Cells    I/O Pins          Expanders

A:      LC1 - LC32     14/32( 43%)    6/16( 37%)      34/64( 53%)
Total dedicated input pins used:                         8/   8 (100%)
Total I/O pins used:                                     6/  16 ( 37%)
Total logic cells used:                                 14/  32 ( 43%)
Total shareable expanders used:                         32/  64 ( 50%)
Total shareable expanders not available (n/a):           2/  64 (  3%)
Total input pins required:                               9
Total output pins required:                              5
Total bidirectional pins required:                       0
Total logic cells required:                             14
Total flipflops required:                                0
Total shareable expanders in database:                  32
Synthesized logic cells:                                 9/  32 ( 28%)
```

3.2.3 SIMULADOR Y ANALIZADOR DE RETARDOS

El primero de estos módulos determina el comportamiento de las distintas salidas del circuito dependiendo de la forma en la que cambian las señales de entrada, reflejando estos resultados en el archivo tipo generador de formas de onda del circuito; el analizador de retardos genera una tabla en la cual se presenta el tiempo que existe entre el instante en que aparece una señal en una entrada y el instante en el que una salida reacciona como consecuencia de dicho evento.

Estos módulos también indican cuando existen problemas ocacionados por no considerar los tiempos de tránsito entre elementos del sistema, y la frecuencia máxima de trabajo de un determinado diseño. Un ejemplo de los resultados que presenta el análizador se presenta en la figura 3.8.

3.3.4 PROGRAMADOR

Este bloque del sistema permite programar el diseño dentro de la estructura interna del CPLD, por medio de la unidad de programación; la programación de un dispositivo se realiza aplicando un voltaje de 12.5 V, a la terminal de acceso de los eslabones programables (transistores NMOS modificados) adecuados. Además se puede utilizar para revisar el contenido de un CPLD, así como para realizar la simulación de algún circuito contenido por el mismo.

Como consecuencia de la gran diversidad de dispositivos CPLD, existe una unidad maestra de programación que permite acoplar diferentes adaptadores, cada adaptador permite la programación de un grupo determinado de CPLD's.

Delay Matrix

Destination

	S2	S1	Z	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
S2	31.0ns/47.0ns	31.0ns	31.0ns	31.0ns	31.0ns	31.0ns	31.0ns	42.0ns	42.0ns	47.0ns	47.0ns/55.0ns	47.0ns/55.0ns	47.0ns/55.0ns
S1		31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	31.0ns/42.0ns	47.0ns/47.0ns	42.0ns/47.0ns	47.0ns/47.0ns	47.0ns/55.0ns	47.0ns/55.0ns	47.0ns/55.0ns
S2			31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	47.0ns/47.0ns	47.0ns/55.0ns	47.0ns/55.0ns	47.0ns/55.0ns
S3				31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns	47.0ns/55.0ns	47.0ns/55.0ns	47.0ns/55.0ns
S4					31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns
S5						31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns
S6							31.0ns	31.0ns/47.0ns	31.0ns/47.0ns	31.0ns/47.0ns	47.0ns/47.0ns	47.0ns/47.0ns	47.0ns/47.0ns
S7								31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/31.0ns	47.0ns/47.0ns	47.0ns/47.0ns	47.0ns/47.0ns
S8									31.0ns/31.0ns	31.0ns/31.0ns	31.0ns/47.0ns	31.0ns/47.0ns	31.0ns/47.0ns
S9											31.0ns	31.0ns/47.0ns	31.0ns/47.0ns
S10												47.0ns	31.0ns/47.0ns
S11													47.0ns
S12	31.0ns/47.0ns	31.0ns	31.0ns	31.0ns	31.0ns	31.0ns	31.0ns	42.0ns	42.0ns	47.0ns	47.0ns	47.0ns	47.0ns
S1		31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns	47.0ns/55.0ns	47.0ns/55.0ns	47.0ns/55.0ns
S2			31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns	47.0ns/55.0ns	47.0ns/55.0ns	47.0ns/55.0ns
S3				31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns	47.0ns/55.0ns	47.0ns/55.0ns	47.0ns/55.0ns
S4					31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns
S5						31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/42.0ns	31.0ns/42.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns	42.0ns/47.0ns
S6							31.0ns	31.0ns/47.0ns	31.0ns/47.0ns	31.0ns/47.0ns	47.0ns/47.0ns	47.0ns/47.0ns	47.0ns/47.0ns
S7								31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/31.0ns	47.0ns/47.0ns	47.0ns/47.0ns	47.0ns/47.0ns
S8									31.0ns/47.0ns	31.0ns/31.0ns	31.0ns/47.0ns	31.0ns/47.0ns	31.0ns/47.0ns
S9											31.0ns	31.0ns/47.0ns	31.0ns/47.0ns
S10												47.0ns	31.0ns/47.0ns
S11													47.0ns

MATRIZ DE RETARDOS

Se presentan los tiempos de propagación en el sistema o circuito; es decir se indica el tiempo que transcurre para que una salida reaccione como resultado de un cambio en una señal de entrada relacionada con esa salida.

- La primera columna corresponde a las entradas del circuito analizado
- El nombre de cada salida esta indicada en la primera fila

Figura 3.8 Resultado de un Análisis de tiempos de propagación (Matriz de retardos)

4 DESARROLLO DEL SISTEMA DIGITAL

4.1 SISTEMAS DIGITALES

Un sistema digital electrónico es un conjunto de elementos que manipulan o procesan información discreta.

La información binaria almacenada en un sistema digital puede clasificarse ya sea como datos o bien como control de información. Los datos son elementos discretos de información que se manipulan para realizar tareas de aritmética, lógica, corrimiento y otras tareas similares de procesamiento de datos. Estas operaciones se desarrollan con componentes digitales como sumadores, decodificadores, multiplexores, contadores y registros de corrimiento. La información de control proporciona señales de mando que supervisan las diversas operaciones de la sección de datos con objeto de llevar a cabo las tareas deseadas de procesamiento de información.

El diseño lógico de un sistema digital puede dividirse en dos partes distintas. Una parte se ocupa del diseño de los circuitos digitales que llevan a cabo las operaciones de procesamiento de datos. La otra parte se ocupa del diseño del circuito de control que supervisa las operaciones y las secuencias.

Las relaciones entre el control lógico y el procesador de datos en un sistema digital se muestran en la figura 4.1. El subsistema procesador de datos manipula los datos en los registros de acuerdo con los requisitos del sistema. El control lógico indica los mandos en la secuencia apropiada al procesador de datos. El control lógico usa las condiciones de estado para servir como variables de decisión para determinar la secuencia de las señales de control.

El control lógico que genera las señales para dar la secuencia de las operaciones en el procesador de datos es un circuito secuencial cuyos estados internos dictan los mandos de control para el sistema. En cualquier momento el estado del control secuencial inicia un conjunto prescrito de mandos. Dependiendo de las condiciones de estado y otras entradas externas, el circuito secuencial pasa al estado siguiente para iniciar otras operaciones. Los circuitos digitales que actúan como el control lógico proporcionan una secuencia de tiempo de señales para iniciar las operaciones en el procesador de datos y determinar el siguiente estado del subsistema de control.

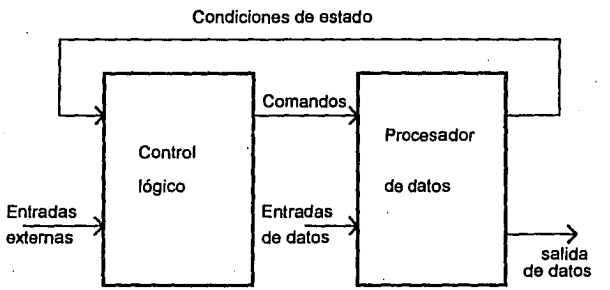


Figura 4.1 Subsistemas de control y procesamiento de datos

4.2 DISEÑO DE CIRCUITOS DIGITALES EMPLEANDO UN PLDS

El Sistema de Desarrollo de Lógica Programable permite realizar el diseño de un sistema digital en forma rápida y eficiente; es posible registrar facilmente las características de un diseño preliminar, simular su comportamiento, evaluarlo y realizar acciones orientadas a mejorar su desempeño, para finalmente "programar" un circuito integrado, obteniendo así, un circuito digital que resuelva un problema en particular.

La forma en la que se desarrolla un circuito digital mediante un PLDS, no es muy diferente a la forma de diseño tradicional; es más sencilla, ya que se cuenta con recursos que permiten evaluar el sistema diseñado sin tener que construir fisicamente el circuito digital. Además, se evita tener que realizar inversiones económicas y el riesgo de dañar los dispositivos.

Un factor que se debe considerar al trabajar con Dispositivos Programables para Lógica Compleja (CPLDs), es que debido a la forma en la que estan constituídos estos dispositivos (a base de celdas lógicas), el tiempo de propagación entre los módulos que forman un diseño no es un parámetro fácil de determinar

El tiempo de respuesta de un circuito digital desarrollado con un Sistema de Desarrollo de Lógica Programable (PLDS) depende de dos factores :

1. las características del diseño
2. el dispositivo usado

Existen diversos factores que se deben tomar en cuenta para diseñar un sistema digital, entre ellos estan los siguientes :

a) la forma de procesar la información

El procesamiento secuencial de información es más lento que el procesamiento en paralelo, sin embargo el procesamiento secuencial requiere menos recursos; existe una relación entre velocidad de procesamiento y la cantidad de recursos empleados, el diseñador debe encontrar la relación óptima entre estas variables.

b) la eficiencia de los algoritmos usados

Generalmente, se pude resolver un problema en diferentes formas; una vez que que se conocen las alternativas, el diseñador debe evaluarlas y seleccionar aquella que presente mayores ventajas

c) los elementos usados para construir la arquitectura del sistema

La elección de los elementos adecuados para desarrollar un sistema digital es uno de los aspectos críticos del diseño, en ocasiones se pueden evaluar diversas alternativas; en otros casos, las condiciones del problema imponen el uso de determinados circuitos.

Las características de un CPLD que determinan el desempeño de un sistema digital son :

a) el tiempo de propagación por compuerta

El tiempo de propagación por compuerta es el mínimo tiempo requerido para que una señal se propague a través de los elementos lógicos combinacionales de una macrocelda (un inversor, el arreglo AND, el arreglo OR y una compuerta XOR).

En una macrocelda, cada señal de entrada debe propagarse a través de los elementos lógicos combinacionales, antes de llegar a una terminal de salida. Lo anterior implica que sí se quiere usar un CPLD, sólo para obtener el valor complemento de una señal (como inversor), usando un dispositivo con tiempo de retardo por compuerta de veinte nanosegundos, el tiempo de tránsito del circuito será de veinte nanosegundos.

Si se desarrollan diversos subdiseños en forma independiente y posteriormente se integran en un diseño de mayor complejidad, no es posible afirmar que el tiempo de respuesta final será la suma de los tiempos de tránsito de los elementos que forman la trayectoria crítica (mas larga); el tiempo de retardo final dependerá de la síntesis realizada por el PLDS y de la forma en que los elementos del diseño sean distribuidos en la estructura interna del CPLD. En la figura 4.2 se observa el comportamiento de diversos elementos en un diseño realizado con un dispositivo con retardo por compuerta de veinte nanosegundos.

b) el consumo de potencia

Estos dispositivos usan pequeñas cantidades de energía para su funcionamiento (característica típica de los dispositivos CMOS). En algunos CPLDs adicionalmente existe un circuito que reduce más la disipación de potencia.

Este circuito incluye un detector de transición; cuando cambia el valor de una señal de entrada, permite que el arreglo lógico sea energizado durante el tiempo necesario para que el valor adquirido se propague hasta las salidas, en donde un elemento se asegura que dicho valor se conserve, en ese momento se desenergiza el arreglo lógico y sólo se conserva el valor que adquirió la entrada correspondiente, para poder compararla y así detectar el momento en el que la entrada cambie de valor nuevamente.

El circuito que permite ese ahorro de energía, involucra un incremento en el tiempo de propagación por compuerta del orden del 40 por ciento .

Existe en los dispositivos de la subfamilia MAX7000 y CLASSIC, la posibilidad de deshabilitar dicho circuito, eleminandose el consiguiente retardo a las entradas de las celdas lógicas (macroceldas).

Delay Matrix

Destination

	s	s0	s0	sc
c	20.0ns			
c	20.0ns			
c	20.0ns	32.0ns	44.0ns	
CLK				18.0ns

matriz de retardos

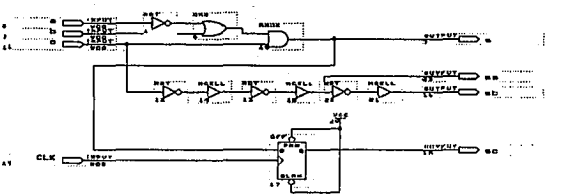


Figura 4.2 Tiempo de retardo para algunos elementos dentro de un diseño usando un CPLD con retardo por compuerta de veinte nanosegundos.

4.3 CARACTERISTICAS DEL SISTEMA

De acuerdo a las características del detector MEPSICRON, podemos enunciar nuestro problema de diseño como :

diseñar un sistema digital que resuelva las ecuaciones 4.1 y 4.2 en un tiempo menor a 1.02 microsegundo y ocupe el mínimo espacio posible.

El circuito digital a desarrollar se puede representar como un sistema con las siguientes características :

VARIABLES DE ENTRADA

a) cuatro números binarios identificados con los símbolos A, B, C y D, respectivamente, cada uno de ellos formado por doce dígitos (bits); los cuatro números son entregados en el mismo instante (en paralelo), y corresponden a valores positivos; estos datos pueden ser actualizados a intervalos de 1.02 microsegundos.

b) una señal que indica que puede considerarse válida, la información presente en las entradas (A, B, C y D),

c) otras señales (sincronización, restablecer y limpiar registros, etc.).

VARIABLES DE SALIDA

a) El sistema debe presentar cada una de coordenadas X y Y, como un número formado por diez dígitos binarios (bits), para facilitar el posterior procesamiento de información (despliegue en una pantalla de 1024 X 1024 "pixeles").

c) Se debe generar la señal que indique que los datos presentes a la salida, pueden ser considerados como información válida, esta señal es necesaria para el correcto flujo de información entre el presente sistema y los otros elementos del detector MEPSICRON.

d) otras señales.

RELACIONES ENTRE LAS VARIABLES DE ENTRADA Y SALIDA

El sistema debe calcular el resultado de las siguientes ecuaciones :

$$X = \left(\frac{((B+C)-(A+D))}{A+B+C+D} \right) \qquad ...(4.1)$$

$$Y = \left(\frac{((A+B)-(C+D))}{A+B+C+D} \right) \qquad ...(4.2)$$

En donde X y Y corresponden a las coordenadas del punto de impacto del fotón contra la superficie del transductor.

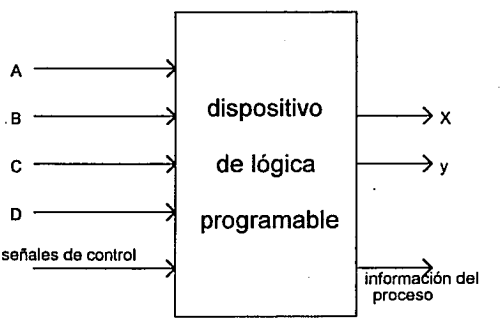


Figura 4.3 Esquemático del Sistema Digital a desarrollar

4.3.1 IDENTIFICACION DE LOS ELEMENTOS NECESARIOS PARA DESARROLLAR EL SISTEMA

De la forma de las ecuaciones (4.1) y (4.2), se observó que deben realizarse las siguientes operaciones :

a) cinco adiciones
b) dos sustracciones
c) dos divisiones

Se identificarón además cuatro características importantes :

1) los elementos del divisor son iguales para ambas ecuaciones

2) el resultado de la división siempre será un número menor que 1 (en valor absoluto)

3) en ambas ecuaciones el dividendo puede tener valores positivos o negativos.

4) existe la posibilidad de realizar varias operaciones simultáneamente

MAGNITUD DE LAS VARIABLES

El conocer el comportamiento de las variables de entrada permitiría tomar decisiones con respecto a el tamaño de los circuitos del procesador que se debía desarrollar, característica directamente relacionada con la cantidad de recursos usados y el tiempo de respuesta; debido a situaciones ajenas a las personas involucradas con el presente trabajo, no fue posible contar con el reporte que describe el desarrollo de las ecuaciones 4.1 y 4.2. Por lo que fue necesario considerar la situación crítica.

La situación crítica corresponde a recibir como entradas al sistema, cuatro valores $(FFF)_{16}$, de donde se deduce que al considerar los dígitos de acarreo, los resultados de las operaciones : $(A+B+C+D)$, $((A+B)-(C+D))$ y $((B+C)-(A+D))$ tendran una longitud de catorce dígitos binarios.

ELEMENTOS NECESARIOS

De acuerdo al resultado anterior, y considerando una arquitectura en paralelo, se requieren los siguientes elementos :

- cuatro circuitos sumadores para números de doce dígitos binarios
- un circuito sumador para números de trece dígitos binarios

(en ambos casos los circuitos deben generar un dígito de "acarreo")

- dos circuitos que ejecuten la sustracción de números de trece bits
- dos circuitos divisores que reciban números de catorce dígitos (dividendo y divisor) y entreguen el resultado (cociente) como un número de diez dígitos.

La posibilidad de realizar la transferencia de datos a la computadora, a través de un bus de datos de treinta y dos bits; y el hecho de que las entradas A, B, C y D, se entregen en paralelo, motivaron el orientar el diseño hacia una arquitectura totalmente en paralelo.

4.4 INVESTIGACION DE ALGORITMOS

4.4.1 ADICION Y SUSTRACCION

ADICION

Considerando que el tiempo y la cantidad de recursos utilizados son los factores importantes en nuestro problema, y que el PLDS nos permite diferentes alternativas para construir un circuito sumador, se procedió a determinar las diferencias entre cada uno de esos métodos, para determinar cual era la mejor opción.

Se identificó la posibilidad de construir un sumador en cuatro formas diferentes :

a) como un sumador con acarreo anticipado (carry look-ahead)

b) usando una arquitectura especial, presente en los archivos del sistema como una arquitectura patrón denominada FAAD.

c) usando la operación aritmética (adición) definida en el el editor de textos

d) usar sumadores completos (full adders) en paralelo.

En adelante se hará referencia a esos circuitos como : sumador SCLAD, sumador SFAAD, sumador STEXT y sumador SCOMP respectivamente.

Los cuatro circuitos se desarrollaron en el mismo tipo de CPLD (EPLDMAX5128), se realizó el análisis de tiempos de tránsito y de cantidad de recursos empleados por cada uno de ellos, esa información se incluye en la tabla 4.1

# de dígitos binarios	tipo	porcentaje de recursos	máximo tiempo de retardo [ns]
12	SCLAD	30	115
12	SFAAD	15	135
12	STEXT	12	155
12	SCOMP	27	235

tabla 4.1

La opción que presentó el menor tiempo de propagación, fue la arquitectura de un sumador con acarreo anticipado (SCLAD), pero requiere una gran cantidad de recursos; este circuito sería una buena alternativa para un problema que requiera gran velocidad, y no tenga limitaciones con respecto a espacio. (figura 4.3)

La adición desarrollada por el editor de textos (STEXT), fue relativamente lenta pero, también la que menos recursos emplea: esa característica lo hace ideal para sistemas que tienen limitaciones de espacio, y si se considera una arquitectura en paralelo el tiempo de retardo podría dejar de ser un problema. (figura 4.5)

El sumador denominado SFADD utilizó sólo un veinticinco por ciento más de recursos que la opción del editor de textos y presentó menor tiempo de tránsito; esta podría ser considerada la opción más completa para una aplicación que requiera un rápido procesamiento sin ocupar muchos recursos. (figura 4.4)

La opción de ocupar sumadores completos, en paralelo definitivamente se descarta; de hecho cualquier persona que tenga experiencia trabajando con circuitos sumadores digitales, sabe que este tipo de circuito es poco eficiente sin embargo era nuestra intención probar su comportamiento al usar el Sistema de Desarrollo de Lógica Programable.

La tabla 4.2 registra el comportamiento del sumador STEXT en diferentes tipos de dispositivos CPLD, y también el efecto de usar el circuito que permite reducir el consumo de energía (turbo bit).

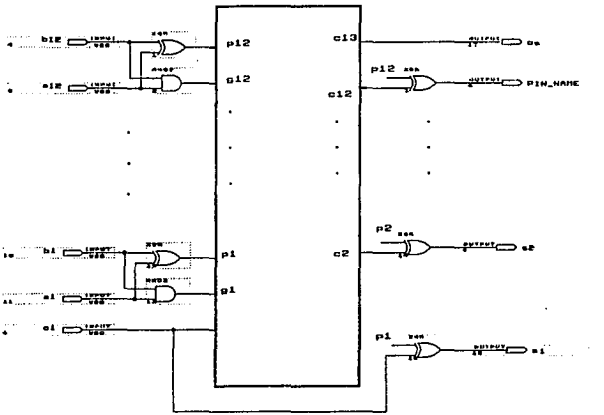
sumador (doce dígitos binarios)		
dispositivo	turbo bit habilitado	tiempo de tránsito [ns]
MAX5128-20	no existe	135
MAX5196-20	no existe	135
MAX7128-20	no	147
MAX7128-20	si	68
MAX7256-20	no	147
MAX7256-20	si	68
MAX7256-12	no	104
MAX7256-12	si	56

tabla 4.2

Si desarrollamos un circuito sumador usando la operación aritmética (adición) definida por el editor de textos, y se usa un dispositivo MAX7256-12 (que tiene un tiempo de retardo por compuerta de doce nanosegundos), se logra un sumador rápido y que requiere una cantidad mínima de recursos.

SUSTRACCION

Una de las formas más eficientes de ejecutar una sustracción es sumar al minuendo el complemento a dos del sustraendo; si utilizamos los distintos circuitos sumadores desarrollados anteriormente, el tiempo de tránsito y la cantidad de recursos usados deben resultar aproximadamente los mismos que en el caso de la adición; por lo cual sólo se realizó el análisis de la sustracción definida en el editor de textos (figura 4.6).



Ecuaciones para generar un sumador con acarreo anticipado :

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$

$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

Figura 4.3a Esquema de conexiones y ecuaciones para generar un sumador binario con acarreo antipado

Archivo 1 % se generan los valores P y G %

```
SUBDESIGN FAD12
(A[11..0],B[11..0] : INPUT ;
P[11..0],G[11..0] : OUTPUT ;)

VARIABLE
SF1[11..0] : SOFT ;
SF2[11..0] : SOFT ;

BEGIN
SF1[].IN = A[] XOR B[] ;
P[] = SF1[].OUT ;
SF2[].IN = A[] AND B[] ;
G[] = SF2[].OUT ;
END;
```

Archivo 2 % se generan los valores C_i %

```
SUBDESIGN CARRY12
(    P[11..0], G[11..0]              : INPUT;
     CI                              : INPUT;
     C[11..0],CS            : OUTPUT;)

BEGIN
C0 = CI;
C1 = (P0&CI) # G0;
C2 = (P0&P1&CI) # (P1&G0) # G1;
C3 = (P0&P1&P2&CI) # (P1&P2&G0) # (P2&G1) # G2;
C4 = (P0&P1&P2&P3&CI) # (P1&P2&P3&G0) # (P2&P3&G1) # (P3&G2) # G3;
C5 = (P0&P1&P2&P3&P4&CI) # (P1&P2&P3&P4&G0) #
        (P2&P3&P4&G1) # (P3&P4&G2) #
     (P4&G3) # G4;
C6 = (P0&P1&P2&P3&P4&P5&CI) # (P1&P2&P3&P4&P5&G0) # (P2&P3&P4&P5&G1) #
     (P3&P4&P5&G2) # (P4&P5&G3) # (P5&G4) # G5;
C7 = (P0&P1&P2&P3&P4&P5&P6&CI) # (P1&P2&P3&P4&P5&P6&G0) #
                (P2&P3&P4&P5&P6&G1) #
     (P3&P4&P5&P6&G2) # (P4&P5&P6&G3) # (P5&P6&G4) # (P6&G5) # G6;
C8 = (P0&P1&P2&P3&P4&P5&P6&P7&CI) # (P1&P2&P3&P4&P5&P6&P7&G0) #
     (P2&P3&P4&P5&P6&P7&G1) # (P3&P4&P5&P6&P7&G2) # (P4&P5&P6&P7&G3) #
     (P5&P6&P7&G4) # (P6&P7&G5) # (P7&G6) # G7;
C9 = (P0&P1&P2&P3&P4&P5&P6&P7&P8&CI) # (P1&P2&P3&P4&P5&P6&P7&P8&G0) #
     (P2&P3&P4&P5&P6&P7&P8&G1) # (P3&P4&P5&P6&P7&P8&G2) #
                (P4&P5&P6&P7&P8&G3) #
     (P5&P6&P7&P8&G4) # (P6&P7&P8&G5) # (P7&P8&G6) # (P8&G7) # G8;
C10 = (P0&P1&P2&P3&P4&P5&P6&P7&P8&P9&CI) #
                (P1&P2&P3&P4&P5&P6&P7&P8&P9&G0) #
     (P2&P3&P4&P5&P6&P7&P8&P9&G1) # (P3&P4&P5&P6&P7&P8&P9&G2) #
     (P4&P5&P6&P7&P8&P9&G3) # (P5&P6&P7&P8&P9&G4) # (P6&P7&P8&P9&G5) #
     (P7&P8&P9&G6) # (P8&P9&G7) # (P9&G8) # G9;
C11 = (P0&P1&P2&P3&P4&P5&P6&P7&P8&P9&P10&CI) #
     (P1&P2&P3&P4&P5&P6&P7&P8&P9&P10&G0) #
                (P2&P3&P4&P5&P6&P7&P8&P9&P10&G1) #
     (P3&P4&P5&P6&P7&P8&P9&P10&G2) # (P4&P5&P6&P7&P8&P9&P10&G3) #
```

```
    (P5&P6&P7&P8&P9&P10&G4) # (P6&P7&P8&P9&P10&G5) #
                (P7&P8&P9&P10&G6) #
    (P8&P9&P10&G7) # (P9&P10&G8) # (P10&G9) # G10;
CS = (P0&P1&P2&P3&P4&P5&P6&P7&P8&P9&P10&P11&Ci) #
                (P1&P2&P3&P4&P5&P6&P7&P8&P9&P10&P11&G0) #
                (P2&P3&P4&P5&P6&P7&P8&P9&P10&P11&G1) #
                (P3&P4&P5&P6&P7&P8&P9&P10&P11&G2)#
                (P4&P5&P6&P7&P8&P9&P10&P11&G3) # (P5&P6&P7&P8&P9&P10&P11&G4) #
                (P6&P7&P8&P9&P10&P11&G5) # (P7&P8&P9&P10&P11&G6) #
    (P8&P9&P10&P11&G7) # (P9&P10&P11&G8) # (P10&P11&G9) # (P11&G10) #
                G11;

END;
```

Archivo 3 % se generan los valores S, %

```
SUBDESIGN FINFAD12
(P[11..0],C[11..0] : INPUT ;
S[11..0] : OUTPUT ;)
BEGIN
S[] = P[] XOR C[] ;
END;
```

Figura 4.3 b Archivos tipo texto para construir un sumador con acarreo anticipado (SCLAD) de doce bits.

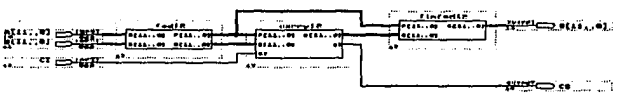


Figura 4.3 c Archivo tipo gráfico que reúne los archivos tipo texto necesarios para un sumador con acarreo anticipado (SCLAD)

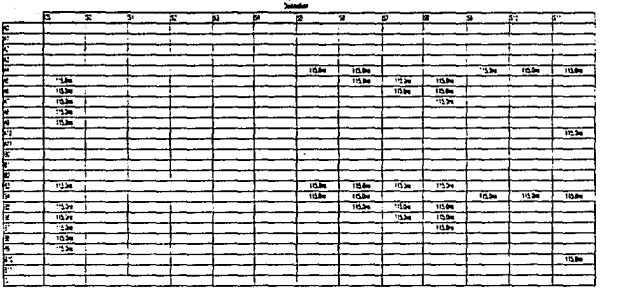


Figura 4.3d Tiempos máximos de propagación para un sumador con *acarreo* rápido (SCLAD) de doce bits.

```
MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 08/11/94 14:23:00

***** Project compilation was successful

** DEVICE SUMMARY **

Chip/                    Input  Output  Bidir      Shareable
POF     Device           Pins   Pins    Pins   LCs Expanders  % Utilized

fadd12b  EPM5128          25     13      0     39    78        30 %

User Pins:               25     13      0
```

Figura 4.3e Archivo reporte para del diseño del sumador con acarreo anticipado (SCLAD) de doce bits.

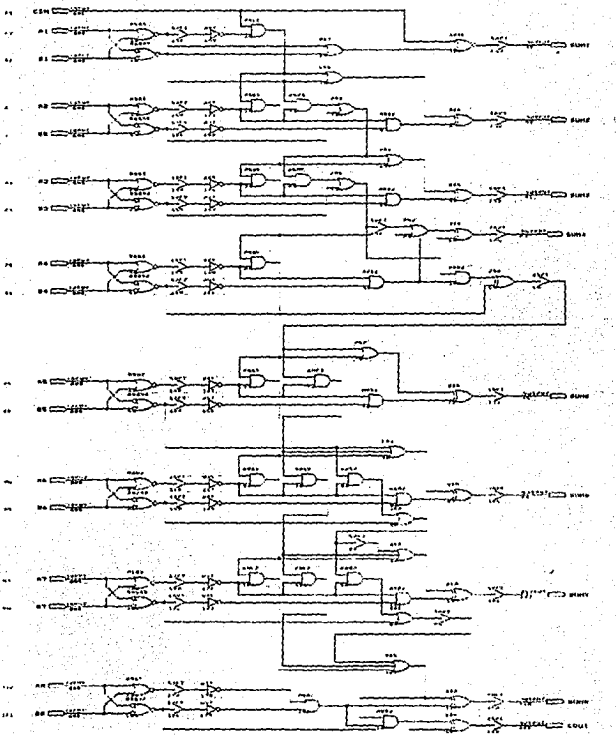


Figura 4.4a Arquitectura del sumador denominado FAAD presente en los archivos del sistema de desarrollo como una arquitectura patron (sumador de 8 bits)

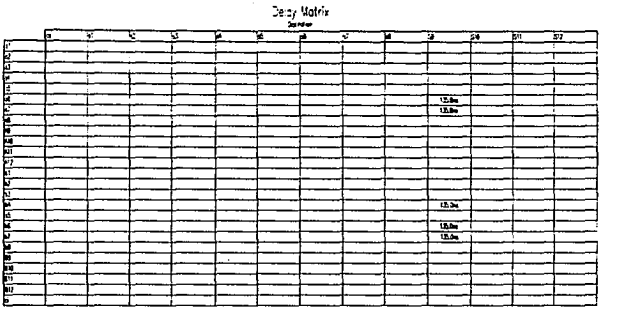


Figura 4.4 bTiempos de retardo máximos para un sumador de doce bits usando la arquitectura FAAD.

Project Information c:\max2work\sfad12b.rpt

MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 07/29/94 15:10:20
***** Project compilation was successful
** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	LCs	Shareable Expanders	% Utilized
sfad12b	EPM5128	25	13	0	20	27	15 %
User Pins:		25	13	0			

Figura 4.4 c Archivo reporte del diseño de un sumador de doce bits, usando la arquitectura FAAD.

```
SUBDESIGN Suma12b
(
A[11..0],B[11..0] : INPUT ;
S[12..0]          : OUTPUT ;
)

VARIABLE

AA[12..0],BA[12..0],SA[12..0] : NODE ;


BEGIN

AA[12] = GND;
BA[12] = GND;
AA[11..0] = A[];
BA[11..0] = B[];


SA[] = BA[] + AA[] ;

S[] = SA[12..0];

END;
```

Figura 4.5a Archivo de construcción de un sumador de doce bits empleando la operación aritmética definida en el editor de textos.

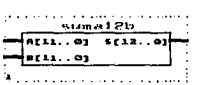


Figura 4.5 b Simbolo del sumador construido a partir de la operación aritmética definida en el editor de textos (STEXT)

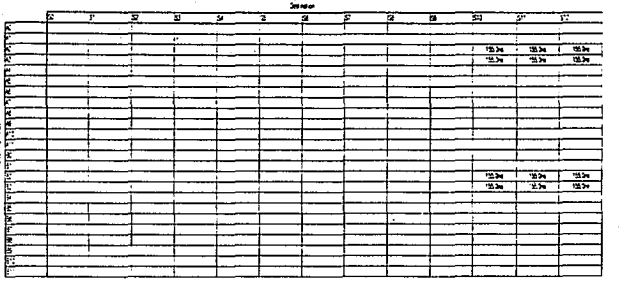


Figura 4.5c Retardos máximos en el sumador tipo STEXT de doce bits.

Project Information c:\max2work\suma12b.rpt

MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 07/20/94 13:26:26

***** Project compilation was successful
** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	Shareable LCs	Expanders	% Utilized
suma12b	EPM5128	24	13	0	16	24	12 %
User Pins:		24	13	0			

Figura 4.5d Archivo reporte del sumador tipo STEXT de doce bits.

```
SUBDESIGN resta2
( D[11..0],E[11..0] : INPUT;          %entradas%
  F[11..0],CS       : OUTPUT;)        %salidas%

VARIABLE
RD[12..0],RE[12..0],RF[12..0] : NODE;

BEGIN
RD[12]=GND;
RE[12]=GND;
RD[11..0]=D[11..0];
RE[11..0]=E[11..0];

      RF[]=RD[]-RE[];
F[11..0]=RF[11..0];
CS=RF[12];
END;
```

Project Information c:\max2work\adrian\resta2.rpt

MAX+plus II Compiler Report File
Version 2.11 04/06/92
***** Project compiled without errors

** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	Shareable MCs	EXPs	% Utilized
resta2	EPM5128	24	13	0	17	24	13 %

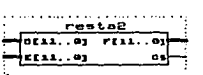


Figura 4.6 a Archivos de construcción, reporte y simbolo del archivo para un restador de doce bits .

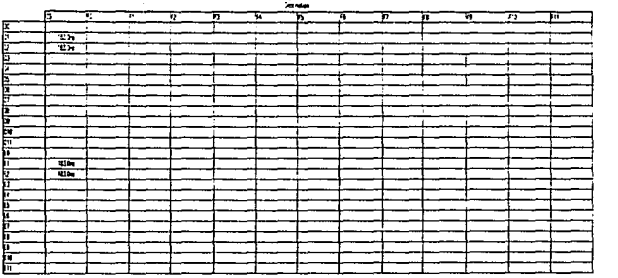


Figura 4.6 b Máximo tiempo de propagación de un circuito que realiza la sustracción entre dos números de doce bits, (empleando el MAX5128).

ARQUITECTURA PARA REALIZAR LAS ADICIONES Y SUSTRACCIONES

Las adiciones y sustracciones indicadas en las ecuaciones 4.1 y 4.2, pueden realizarse en dos niveles :

en el primer nivel se pueden realizar simultaneamente : (A+B), (B+C), (A+D) y (C+D).

en el segundo nivel se deben realizar las sustracciones y la suma ((A+B) + (C+D))

Considerando la cantidad de recursos requeridos, la mejor opción para esta etapa del sistema era usar circuitos sumadores tipo STEXT.

De acuerdo al reporte de construcción del sumador tipo STEXT (figura 4.5d), se usaron 12 macroceldas; así que para realizar las siete operaciones se requieren aproximadamente 84 macroceldas (12 * 7).

Para asegurar que se realizen en forma simultánea, el número máximo de operaciones, debía seleccionarse un dispositivo cuyos recursos internos permitieran crear una arquitectura en paralelo.

El dispositivo seleccionado fue el MAX7256, éste CPLD cuenta con 256 macroceldas (es el más grande de la familia MAX7000); el número de terminales de entrada (164), permite recibir todos los datos y generar los resultados de salida en paralelo; existen tres versiones del MAX7256, la diferencia entre ellos es el retardo por compuerta (12, 15 y 20 nanosegundos); lo que permite mayor flexibilidad para el diseño.

El análisis de tiempo de propagación (figura 4.7), indica que si se utiliza el dispositivo con tiempo de retardo por compuerta de 12 ns, para realizar todas las adiciones y sustracciones se requieren 62 ns; además se dispone aún del cuarenta y siete por ciento del dispositivo para desarrollar otros módulos del sistema (figura 4.7.

```
% ESTE ARCHIVO REALIZA TODAS LAS ADICIONES Y SUSTRACCIONES %
% REQUERIDAS POR LAS ECUACIONES  4.1 y 4.2%

SUBDESIGN Unidsyrs
(A[12..1],B[12..1],C[12..1],D[12..1]              : INPUT ;        %entradas%
NX[13..1],NY[13..1],DS[14..1],SIGX,SIGY    : OUTPUT ; )     %salidas%

VARIABLE

AA[14..1],AB[14..1],AC[14..1],AD[14..1] : NODE ;                %conexiones%
SAB[14..1],SCD[14..1],SBC[14..1],SAD[14..1] : NODE ;        %internas%
SS[14..1],RX[14..1],RY[14..1] : NODE ;

BEGIN

AA[14..13] = GND;
AB[14..13] = GND;
AC[14..13] = GND;
AD[14..13] = GND;
AA[12..1] = A[];
AB[12..1] = B[];
AC[12..1] = C[];
AD[12..1] = D[];

% MODULO DE SUMAS %

SAB[] = AA[] + AB[] ;
SCD[] = AC[] + AD[] ;
SBC[] = AB[] + AC[] ;
SAD[] = AA[] + AD[] ;

% MODULO DE SUMA TOTAL Y SUSTRACCIONES %

SS[] = SAB[] + SCD[] ;
RX[] = SBC[] - SAD[] ;
RY[] = SAB[] - SCD[] ;

% MODULO DE SALIDAS %

NX[13..1] = RX[13..1] ; %dividendo x%
NY[13..1] = RY[13..1] ; %dividendo y%
DS[] = SS[14..1] ;          %divisor%
SIGX = RX[14] ;             %signo de x%
SIGY = RY[14] ;             %signo de y%

END;
```

Figura 4.7a Archivo *UNIDSYRS*, este diseño realiza todas las adiciones y sustracciones indicadas en las ecuaciones 4.1 y 4.2

```
Project Information                    c:\max2work\unidsyrs.rpt
MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 08/11/94 15:22:32
***** Project compilation was successful
** DEVICE SUMMARY **
```

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	LCs	Shareable Expanders	% Utilized
unidsyrs	EPM7256EGC192-12	48	42	0	136	131	53 %
User Pins:		48	42	0			

Figura 4.7 Reporte de construcción del subdiseño UNIDSYRS

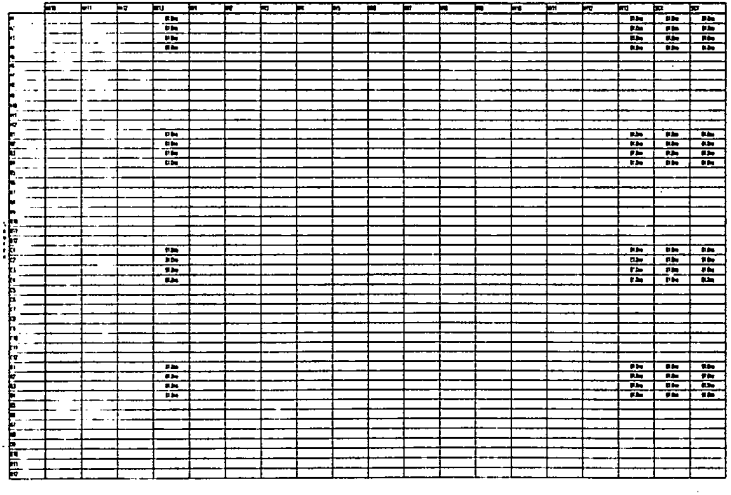


Figura 4.7b Máximos tiempos de retardo del diseño UNIDSYRS

4.4.2 DIVISOR DIGITAL

La primera actividad para desarrollar este elemento del sistema, consistió en investigar algoritmos para la división binaria. Se modificó uno de los algoritmos encontrados de tal manera que, se puede aplicar directamente a la división binaria de dos números enteros, en la que el divisor es mayor que el denominador.

Para establecer este algoritmo basta recordar el algoritmo de una división decimal, el cual se enuncia a continuación :

- Como el divisor es mayor que el dividendo, se asigna un cero como parte entera del cociente.
- Se considera el valor del dividendo como el primer residuo.
- Se multiplica por diez al residuo y se compara con el divisor

  si el residuo es menor, se asigna un cero como parte del cociente

  si es mayor o igual, se cálcula el dígito que al ser multiplicado por el divisor genere el valor más cercano posible al del residuo; el producto se sustrae del residuo y la resta pasa a ser el nuevo residuo

- Se repite el paso anterior, tantas veces como dígitos del cociente se deseen; el primer dígito calculado es el más significativo del cociente.

Ahora debemos deducir el algoritmo para implantar la división binaria, tomando en cuenta :

1. Como se trata de una base binaria, en vez de multiplicar por diez, se hace por dos, (en ambos casos, la acción de multiplicar por la base agrega un cero a la derecha del número).

2. Una cifra del cociente sólo puede tener dos valores (cero o uno).

De acuerdo con esto, el algoritmo para la división binaria de dos números enteros, en donde el divisor es mayor al dividendo, es el siguiente :

- Como el divisor es mayor que el dividendo, se asigna un cero como parte entera del cociente.

- Se considera el valor del dividendo como el primer residuo.

- Se multiplica por dos al residuo y se compara con el divisor

  si el residuo es menor se asigna un cero como parte del cociente

  en caso contrario el unico valor que puede formar parte del cociente es un uno, ahora no tiene sentido multiplicar el cociente por el divisor (multiplicar por uno), por lo que el divisor se resta del residuo y el resultado es considerado como el nuevo residuo.

- Se repite el paso anterior, tantas veces como dígitos del cociente se deseen; al igual que en el algoritmo anterior, las primeras iteraciones generan los dígitos más significativos.

  El algoritmo requiere que tanto el dividendo como el divisor sean números positivos.

DIAGRAMA DE FLUJO

El algoritmo de la división puede representarse por medio del siguiente diagrama de flujo :

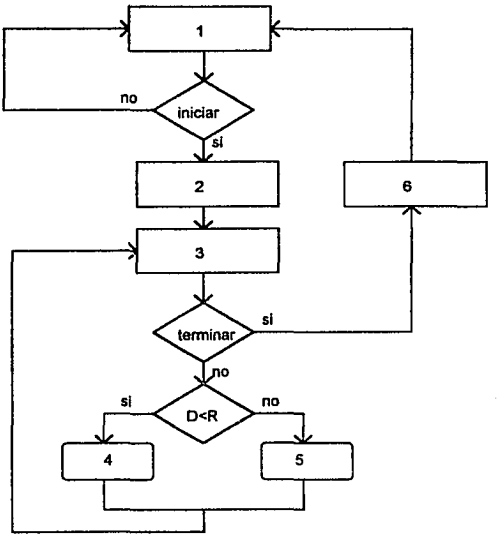


Figura 4.8 diagrama de flujo para el algoritmo de la división

1) INICIO :
indicar el número deseado de dígitos del cociente pero incrementado en uno, alamacenar el dato en un registro (I)
almacenar el valor del dividendo (N)
almacenar el valor del divisor (D)

2) considerar al dividendo como el primer residuo :
transferir el valor del dividendo al registro que guarda el residuo (R)
"limpiar" el registro que guarda el cociente (S)

3) multiplicar el residuo por dos
disminuir a I en uno

4) sí el residuo es menor que el divisor :
guardar un cero como el elemento S[I]

5) sí el residuo es mayor o igual al divisor :
guardar un uno como el elemento S[I]
asignar a R el resultado de substraer D a R

6) FIN :
indicar que se ha terminado el proceso

En este algoritmo se propuso utilizar registros para almacenar los valores correspondientes al dividendo y al divisor; sin embargo considerando que los datos presentes a las entradas del sistema en desarrollo, provienen de registros, y que todos los circuitos electrónicos correspondientes al transductor, al acondicionador de señales y a los desarrollados por el presente trabajo, serán instalados dentro de un elemento que elimine señales de ruido, se decidió suprimir dichos elementos, los cuales representan un consumo de recursos, así como un aumento en el tiempo de tránsito del sistema.

ARQUITECTURA DEL DIVISOR

El siguiente paso fue proponer la arquitectura del procesador y el control lógico que ejecuten el anterior algoritmo.

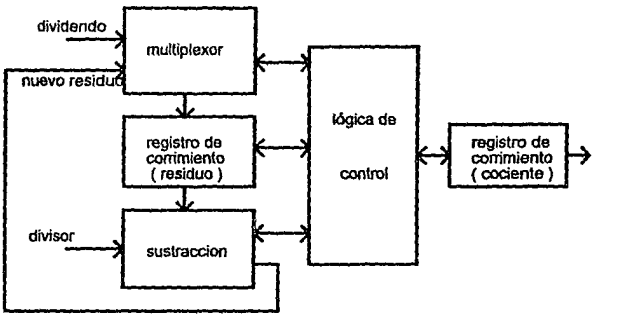


Figura 4.8 Elementos del circuito divisor

CARTA ASM

Para analizar el flujo de información y la correcta sincronización del sistema, se realizó un diagrama de máquina de estados (carta ASM). figura 4.9

En la carta ASM 1 (figura 4.9), se tienen las siguientes asignaciones :

c : variable de control que se verifica alta si el residuo es menor que el divisor
rc1 : variable de control que se verifica baja indicando que debe iniciar el proceso
rc2 : variable de salida que se verifica alta en el instante que se termina el proceso.
R : registro que almacena al residuo
S: registro que almacena al cociente
D: divisor
N: dividendo
I: registro que almacena el número de iteracion
nb: número de bits desados en el cociente

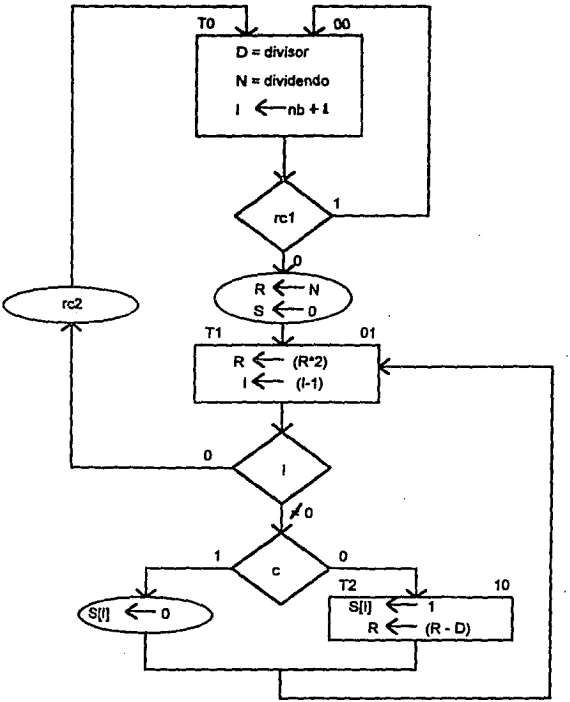


Figura 4.9 Carta ASM 1

Para obtener un dígito del cociente, se requieren dos estados si el residuo es mayor o igual que el divisor y solamente un estado si el residuo es menor.

Siempre que inicia una iteración y sin importar el resultado de la comparación anterior, se debe multiplicar por dos al residuo; para optimizar el proceso, se propuso realizar la multiplicación del residuo a la entrada del comparador, mediante el simple corrimiento de las líneas de conexión, como se indica en la figura 4.11; de esta forma sólo se requiere un estado por cada iteración. La carta ASM 2 (figura 4.10) indica el flujo de información considerando la modificación antes descrita.

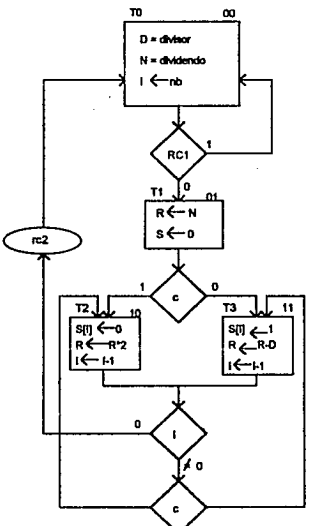
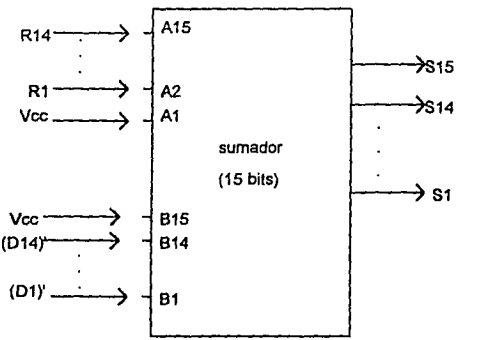


Figura 4.10 Carta ASM 2

Considerando que es más rápido realizar una suma que una resta, se utilizó un circuito sumador para realizar la comparación entre el residuo y el divisor, usando la técnica de sumar al minuendo el complemento a dos del sustraendo; por lo tanto fue necesario obtener el complemento a dos del divisor (sustraendo), esto se hizo mediante el uso de inversores, y se sumó el uno (lógico) necesario aprovechando el dígito binario que se ha dejado libre al realizar el desplazamiento en el residuo.



Al desplazar las líneas de conexión del residuo (R14..R1) se realiza la multiplicación por dos; se aplica un uno (lógico), a la terminal de entrada libre para realizar el complemento a dos del divisor.

En las otras terminales de entrada se aplica el complemento a uno (lógico) del divisor.

En la terminal correspondiente al bit más significativo del sustraendo (divisor) se aplica un uno (lógico) considerando que para el divisor ese bit no existe.

Figura 4.11

Para construir el circuito divisor se eligieron los siguientes dispositivos :

- 74157 (multiplexores 2:1)
- 74194 (registros de corrimiento universal)
- 74164 (registros de corrimiento unidireccional)
- sumador (construido con base en la operación aritmética definida en el editor de textos)
- lógica complementaria
- circuito de control (máquina de estados declarada en el editor de texto)

Los archivos correspondientes se presentan a continuación

MUXB14 : corresponde al diseño de un multiplexor 2:1 para números de catorce dígitos binarios. (figura 4.12)

NOT14 : permite obtener el complemento a uno de una palabra de catorce dígitos binarios.(figura 4.13)

CIRR14: define un registro de corrimiento universal para números de catorce dígitos binarios.(figura 4.14)

RISSP : es un elemento que permite almacenar cada uno de los dígitos binarios del cociente.(figura 4.15)

CIRSUMA permite realizar la comparación de dos números de quince dígitos binarios, fue necesaria tal longitud debido a que el residuo fue multiplicado por dos, por lo cual el residuo aumenta en un dígito su longitud, considerando que el dígito más significativo correspondiente al divisor nunca se ocupará, el resultado de la comparación se genera en el dígito quince del resultado. (figura 4.16)

```
FUNCTION 74157 (SEL,A1,A2,A3,A4,B1,B2,B3,B4,GN) RETURNS (Y1,Y2,Y3,Y4);

SUBDESIGN muxb14
(SEL,D[13..0],E[13..0],GN   : INPUT;
F[13..0]                    : OUTPUT;)

VARIABLE
MI[3..0]                    :74157;

BEGIN
MI3.(SEL,GN) = (SEL,GN);
MI2.(SEL,GN) = (SEL,GN);
MI1.(SEL,GN) = (SEL,GN);
MI0.(SEL,GN) = (SEL,GN);
MI3.(A3,A4) = D[13..12];
MI2.(A1,A2,A3,A4) = D[11..8];
MI1.(A1,A2,A3,A4) = D[7..4];
MI0.(A1,A2,A3,A4) = D[3..0];
MI3.(B3,B4) = E[13..12];
MI2.(B1,B2,B3,B4) = E[11..8];
MI1.(B1,B2,B3,B4) = E[7..4];
MI0.(B1,B2,B3,B4) = E[3..0];
F[13..12] = MI3.(Y3,Y4);
F[11..8] = MI2.(Y1,Y2,Y3,Y4);
F[7..4] = MI1.(Y1,Y2,Y3,Y4);
F[3..0] = MI0.(Y1,Y2,Y3,Y4);

END;
```

Figura 4.12 Multiplexor 2:1 para números de catorce bits

```
SUBDESIGN not14
(d[13..0] : INPUT;
q[13..0] : OUTPUT;)

BEGIN
q[] = NOT d[];
END;
```

Figura 4.13 Módulo para obtener el complemento de un número de catorce bits

```
FUNCTION 74194 (SLSI,SRSI,A,B,C,D,S0,S1,CLRN,CLK) RETURNS (QA,QB,QC,QD);
SUBDESIGN  CIRR14
(SLSI,SRSI,D[13..0],S0,S1,CLRN,CLK  : INPUT;
Q[13..0]                    :OUTPUT;)
VARIABLE
rc[3..0]                    : 74194;
BEGIN
rc3.(SLSI,SRSI,S0,S1,CLRN,CLK) = (SLSI,SRSI,S0,S1,CLRN,CLK);
rc2.(SLSI,SRSI,S0,S1,CLRN,CLK) = (SLSI,SRSI,S0,S1,CLRN,CLK);
rc1.(SLSI,SRSI,S0,S1,CLRN,CLK) = (SLSI,SRSI,S0,S1,CLRN,CLK);
rc0.(SLSI,SRSI,S0,S1,CLRN,CLK) = (SLSI,SRSI,S0,S1,CLRN,CLK);
rc3.(C,D) = D[13..12];
rc2.(A,B,C,D) = D[11..8];
rc1.(A,B,C,D) = D[7..4];
rc0.(A,B,C,D) = D[3..0];
rc2.(SRSI) = rc3.(QD);
rc1.(SRSI) = rc2.(QD);
rc0.(SRSI) = rc1.(QD);
rc3.(SLSI) = rc2.(QA);
rc2.(SLSI) = rc1.(QA);
rc1.(SLSI) = rc0.(QA);
Q[13..12] = rc3.(QC,QD);
Q[11..8] = rc2.(QA,QB,QC,QD);
Q[7..4] = rc1.(QA,QB,QC,QD);
Q[3..0] = rc0.(QA,QB,QC,QD);
END;
```

Figura 4.14 Registro de corrimiento universal para un número de catorce bits, usando el dispositivo 74194 como circuito patrón.

```
FUNCTION 74164 (A,B,CLRN,CLK) RETURNS (QA,QB,QC,QD,QE,QF,QG,QH);
SUBDESIGN rlssp
(A,B,CLRN,CLK       : INPUT;
Q[8..0]             :OUTPUT;)
VARIABLE
rp[1..0]            :74164;
BEGIN
rp1.(CLRN,CLK) = (CLRN,CLK);
rp0.(CLRN,CLK) = (CLRN,CLK);
rp1.(B) = (B);
rp0.(A,B) = (A,B);
rp1.(A) = rp0.(QH);
Q[7..0] = rp0.(QH,QG,QF,QE,QD,QC,QB,QA);
Q[8] = rp1.(QA);
END;
```

Figura 4.15 Registro de corrimiento (hacia la izquierda) para un número de catorce bits, usando el dispositivo 74164 como circuito patrón.

```
SUBDESIGN CIRSUMA
( A[13..0],B[13..0]  : INPUT;        % A[ ] corresponde al residuo ;  B[ ] corresponde al complemento %
  F[13..0],cs        : OUTPUT;)      % del divisor, cs corresponde al resultado de la comparación %
                                     % F[ ] corresponde a la diferencia entre el residuo multiplicado %
VARIABLE                             % dos y el divisor %
SA[14..0],SB[14..0],SC[14..0] : NODE;

BEGIN


SA[14..1]=A[13..0];          % se realizan las acciones indicadas en la figura 10 %
SA[0]=VCC;                   % (desplazamiento y asignación de valores a dos bits%
SB[14]=VCC;
SB[13..0]=B[13..0];

    SC[]=SA[]+SB[];

F[13..0]=SC[13..0];
cs = SC[14];

END;
```

Figura 4.16 Archivo de construcción de un sumador de quince bits (tipo STEXT) En donde se reciben como entradas dos números de catorce bits, internamente se realiza la multiplicación por dos .

UNIDAD DE CONTROL

VARIABLES DE CONTROL

Para generar los comandos de control, primero se identificaron las variables a controlar, considerando los elementos de la arquitectura propuesta para el procesador y las indicaciones de la carta ASM.

a) La entrada de selección en el multiplexor (SEL) : debe ser cero (lógico) durante el tiempo necesario para que el registro CIRR14, la almacene (al menos veinte nanosegundos); posteriormente siempre deberá valer uno (lógico), para permitir la actualización del residuo.

b) Las entradas de selección del registro de corrimiento universal (S1, S0) deben permitir almacenar el nuevo residuo, o en su caso multiplicar por dos al residuo anterior (realizar un corrimiento hacia la izquierda), dependiendo del resultado de la comparación (c); sin embargo durante los estados iniciales (T0 y T1), no se debe tomar en cuenta el valor de la variable de control c, para asegurar que el dividendo sea considerado como el primer residuo;

c) La entrada que permite limpiar el registro RISSP (reset), ésta señal debe ser un cero lógico en el mismo estado en el cual se transfiere el dividendo al residuo.

DESARROLLO DEL MODULO DE CONTROL

El dígito más significativo a la salida del sumador (cs), representa el resultado de la comparación, si su valor fue igual a cero (lógico), el valor del divisor (D), fue menor o igual al valor obtenido al multiplicar el residuo por dos; si el valor de cs fue uno (lógico), entonces el divisor fue mayor. Por lo tanto este elemento fue considerado como la variable de control " c ".

El complemento de la variable c (c'), representa el dígito del cociente correspondiente al número de iteración en proceso, el cual será capturado por el registro que almacena al cociente (RISSP) cuando se presente la proxima transición de nivel bajo a nivel alto (flanco positivo) en la señal de reloj.

De la tabla de verdad de las entradas de control del registro 74194 se observó que, sí se mantiene un uno (lógico) en la entrada S1, es posible controlar mediante S0 las dos acciones de interes.

RC1 indica el inicio del proceso de división, esta señal se aplicó a la entrada de seleccion del multiplexor (SEL) y a la entrada que "limpia" el registro que almacena el resultado (reset); finalmente mediante la función $S0 = (c \cdot RC1)'$ se logró la transferencia del dividendo al registro que guarda el residuo.

Era posible utilizar registros (flip-flops) o un contador para generar las señales de control, sin embargo se optó por generar una máquina de estados usando el editor de textos; con esta herramienta se creó el circuito a partir de la simple declaración de las señales para cada estado.

El circuito de control CONTROL -0 tiene tres entradas :

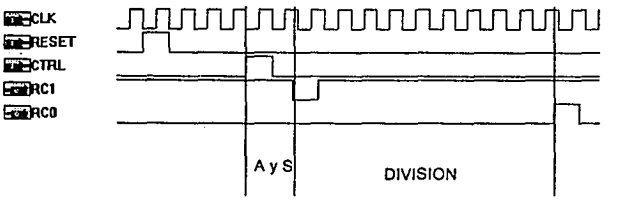
- CLK : que es la señal de reloj (sincronización).
- RESET : señal necesaria para inicializar los registros que constituyen la máquina de estados. Esta señal será aplicada solamente en el momento que el MEPSICRON inicie su operación.
- CTRL : esta señal será la que indique que debe procesarse un paquete de información (inicio).

Las salidas son la señal RC1 y la señal RC0, la primera se genera dos estados despues de que se presento el comando CTRL, esto permite que el módulo que realiza las adiciones y sustracciones cumpla su tarea (considerando su tiempo máximo de propagación); la señal RC0 indica el fin del proceso una vez que se han realizado las iteraciones necesarias en el divisor (CIRDIV).

El archivo correspondiente se muestra en la figura 4.17, en donde además se presenta una simulación .

```
SUBDESIGN CONTROL0
% ENTRADAS Y SALIDAS %
(CTRL,CLK,RESET  : INPUT ;
rc1,rc0          : OUTPUT ;)

% DECLARACION DE LOS ESTADOS (VALORES DE SALIDA) %
VARIABLE
SD : MACHINE OF BITS (RC[1..0])          % SD ES EL NOMBRE DE LA MAQUINA DE ESTADOS %
                                  % estado = salidas (B"rc1rc0") %
        WITH STATES               (      S0 = B"10",      %rc0 : INDICA EL FIN DEL PROCESO %
                                         S1 = B"10",      %rc1 : PERMITE LA TRANSFERENCIA %
                                         S2 = B"00",      %  DE DATOS DEL DIVIDENDO AL %
                                         S3 = B"10",      %     RESIDUO %
                                         S4 = B"10",
                                         S5 = B"10",      % B"10" : 10 (binario) %
                                         S6 = B"10",
                                         S7 = B"10",
                                         S8 = B"10",
                                         S9 = B"10",
                                         S10 = B"10",
                                         S11 = B"10",
                                         S12 = B"11");
BEGIN
SD.CLK = CLK;          %señal de reloj%
SD.RESET = RESET;      % Inicializar los registros (estado inicial) %

TABLE
% CONDICIONES PARA LAS TRANSICIONES %
% ESTADO          CONDICION   ESTADO %
% PRESENTE                    SIGUIENTE %

        SD,       CTRL    =>  SD;

        S0,       0       =>  S0;
        S0,       1       =>  S1;          % B"X" : no importa %
        S1,       B"X"    =>  S2;
        S2,       B"X"    =>  S3;
        S3,       B"X"    =>  S4;
        S4,       B"X"    =>  S5;
        S5,       B"X"    =>  S6;
        S6,       B"X"    =>  S7;
        S7,       B"X"    =>  S8;
        S8,       B"X"    =>  S9;
        S9,       B"X"    =>  S10;
        S10,      B"X"    =>  S11;
        S11,      B"X"    =>  S12;
        S12,      B"X"    =>  S0;
        END TABLE ;
END;
```

Figura 4.17a Archivo de construcción del subsistema de control, como una máquina de estados.



A y S : ADICIONES Y SUSTRACCIONES.

Figura 4.17 b Simulación del circuito de control para la división; en donde se considera el tiempo de propagación del módulo que realiza las adiciones y sustracciones.

Contando con todos los módulos necesarios, se procedió a construir el primer circuito divisor al que se llamó " CIRDIV", (figura 4.18).

La frecuencia de trabajo se determina por la suma de tiempos de tránsito en la trayectoria más larga del sistema, que corresponde a la actualización de un residuo. Del análisis de tiempos de tránsito del multiplexor y del registro de corrimiento 1 se observó que cada uno de ellos genera un retardo de doce nanosegundos, y podría pensarse que para propagarse a través de los dos, se requieren 24 nanosegundos (la suma de ellos, considerando que forman una trayectoria en serie), sin embargo de acuerdo a lo explicado a cerca de la arquitectura de las macroceldas, es posible "programar" ambos módulos en una macrocelda (se usa una macrocelda por cada dígito).

multiplexor,registro 1....	12 ns	
sumador	52 ns	(15 dígitos binarios, usando un dispositivo MAX7256-12)
	64 ns.	

por lo tanto la frecuencia máxima de trabajo a la cual se esperaría trabajar es de 15.6 MHz.

Empleando el Analizador de tiempos del Sistema de Desarrollo, se obtuvó una frecuencia máxima de trabajo de 18.8 MHz (53 ns en la trayectoría crítica); esto es resultado de la forma en que el sistema desarrolla (simplifica y utiliza los recursos del CPLD) el circuito.

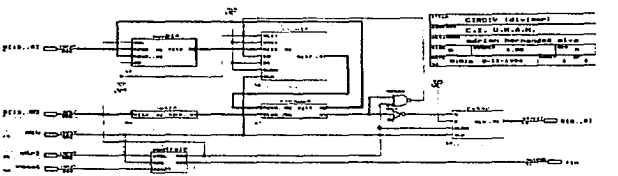


Figura 4.18a Circuito Divisor (CIRDIV), este diseño permite realizar la división entre números binarios de catorce bits (positivos); en donde el divisor (D), es mayor que el dividendo (N). El cociente (S) es entregado como un número de nueve bits.

```
Project Information                    c:\max2work\cirdiv.rpt
MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 08/11/94 06:03:30
***** Project compilation was successful
Title: circuito divisor
Company: C.I. U.N.A.M.
Designer: adrian hernandez alva
Rev: A
Date: 8:01a 8-11-1994
** DEVICE SUMMARY **
Chip/                          Input  Output  Bidir      Shareable
POF     Device                  Pins   Pins    Pins   LCs  Expanders  % Utilized
cirdiv  EPM7256EGC192-12        31     10      0      47   27         18 %
User Pins:                      31     10      0
```

Figura 4.18b Reporte de construcción del circuito divisor.

En las siguientes tres páginas se presentan algunas simulaciones del funcionamiento de este circuito; los valores del dividendo, divisor y del cociente son presentados en forma hexadecimal, debido a que el PLDS agrupa los bits en forma ascendente, se debe considerar que el elemento más significativo del cociente (S[8..0]), corresponde a un bit y no a un grupo de cuatro. Por lo tanto el resultado S[8..0] = $1A8_h$,debe interpretarse como $0.1\ 1010\ 1000_B(0.1\ A\ 8)_h$

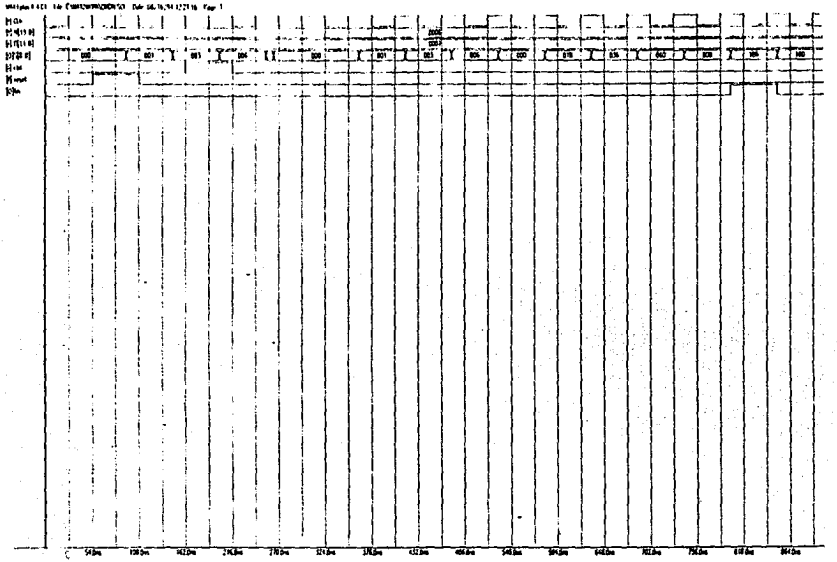


Figura 18c1 Simulación del funcionamiento del circuito divisor (CIRDIV).

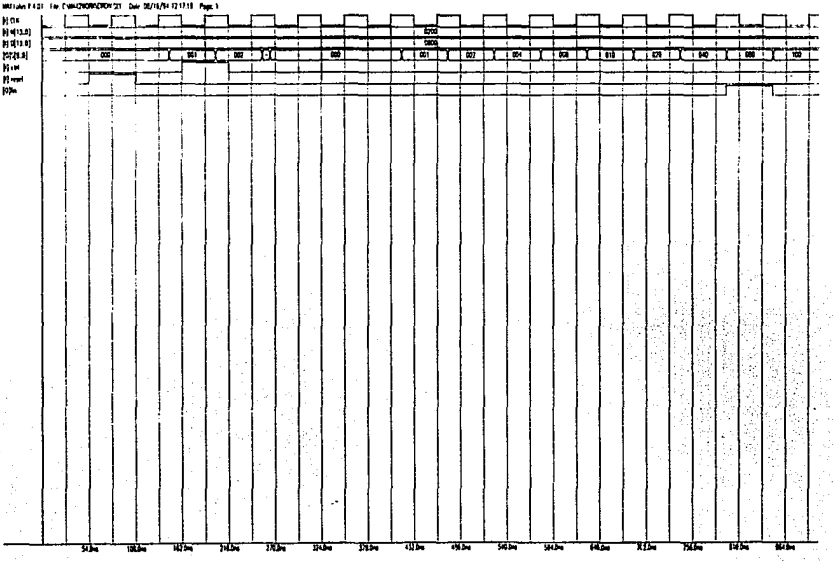


Figura 18c2 Simulación del funcionamiento del circuito divisor (CIRDIV).

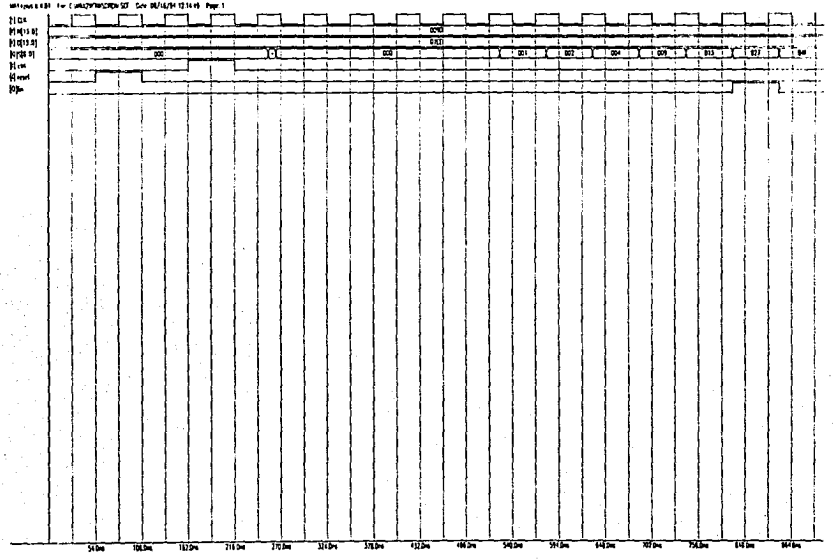


Figura 18c3 Simulación del funcionamiento del circuito divisor (CIRDIV).

4.5 CONSTRUCCION DEL SISTEMA

4.5.1 PRIMER PROTOTIPO "MEPSI1"

Para construir el sistema digital que resuelva las ecuaciones 4.1 y 4.2 fue necesario considerar que el algoritmo de la división que se eligió, sólo funciona cuando el dividendo y el divisor son positivos. Por lo cual se realizó el siguiente circuito :

UNIDSYR0 (figura 4.19), corresponde al circuito que realiza todas las adiciones y sustracciones indicadas en las ecuaciones 4.1 y 4.2, además si el resultado de alguno o ambos dividendos resulta ser negativo, el circuito obtiene su complemento a dos y sólo se conserva el signo, de tal forma que el circuito divisor puede trabajar en forma adecuada. El tiempo de propagación de este circuito fue de 75 ns y requiere el 66 % de un EPMAX7256-12.

```
% ESTE ARCHIVO REALIZA TODAS LAS ADICIONES Y SUSTRACCIONES %
% REQUERIDAS POR LAS ECUACIONES,ADEMAS SI EL RESULTADO %
% DE LAS SUSTRACCIONES ES NEGATIVO, SE CALCULA SU COMPLEMENTO A DOS %
% Y SOLO SE CONSERVA EL SIGNO %
SUBDESIGN Unidsyr0
% ENTRADAS Y SALIDAS %
(A[12..1],B[12..1],C[12..1],D[12..1]    : INPUT ;
NX[14..1],NY[14..1],DS[14..1],SIGX,SIGY : OUTPUT ;)

% CONEXIONES INTERNAS %
VARIABLE
AA[14..1],AB[14..1],AC[14..1],AD[14..1] : NODE ;
SAB[14..1],SCD[14..1],SBC[14..1],SAD[14..1] : NODE ;
SS[14..1],RX[14..1],RY[14..1] : NODE ;
CDX[13..1],CDY[13..1],NAX[13..1],NAY[13..1] : NODE ;

BEGIN
AA[14..13] = GND;
AB[14..13] = GND;
AC[14..13] = GND;
AD[14..13] = GND;
AA[12..1] = A[];
AB[12..1] = B[];
AC[12..1] = C[];
AD[12..1] = D[];
```

```
% SUMAS %
SAB[] = AA[] + AB[] ;                    % A + B %
SCD[] = AC[] + AD[] ;                    % C + D %
SBC[] = AB[] + AC[] ;                    % B + C %
SAD[] = AA[] + AD[] ;                    % A + D %

% SUMA TOTAL Y SUSTRACCIONES %
SS[] = SAB[] + SCD[] ;                   % ( A + B ) + ( C + D ) %
RX[] = SBC[] - SAD[] ;                   % ( B + C ) – ( A + D ) %
RY[] = SAB[] - SCD[] ;                   % ( A + B ) – ( C + D ) %

% COMPLEMENTO A DOS DE LAS SUSTRACCIONES (SI SUS RESULTADOS SON NEGATIVOS)%

NAX[13..1] = (RX[13..1] XOR RX[14]) ;
NAY[13..1] = (RY[13..1] XOR RY[14]) ;
CDX[1] = RX[14] ;
CDY[1] = RY[14] ;
CDX[13..2] = GND ;
CDY[13..2] = GND ;
NX14 = GND ;
NY14 = GND ;

% SALIDAS %
NX[13..1] = (NAX[13..1] + CDX[13..1]);   %DIVIDENDO DE LA ECUACION 4.1 %
NY[13..1] = (NAY[13..1] + CDY[13..1]); %DIVIDENDO DE LA ECUACION 4.2 %
DS[] = SS[14..1] ;   % DIVISOR %
SIGX = RX[14] ;      % SIGNO DE X%
SIGY = RY[14] ;      % SIGNO DE Y%

END;
```

Figura 4.19a Archivo de construcción del módulo UNIDSYR0

```
Project Information                    c:\max2work\unidsyr0.rpt
MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 07/09/94 11:02:51
***** Project compilation was successful
** DEVICE SUMMARY **
Chip/                          Input  Output  Bidir      Shareable
POF     Device                 Pins   Pins    Pins   LCs Expanders % Utilized
unidsyr0 EPM7256EGC192-12      48     44      0      171  181       66 %
User Pins:                     48     44      0
```

Figura 4.19b Reporte de construcción del módulo UNIDSYR0

Delay Matrix

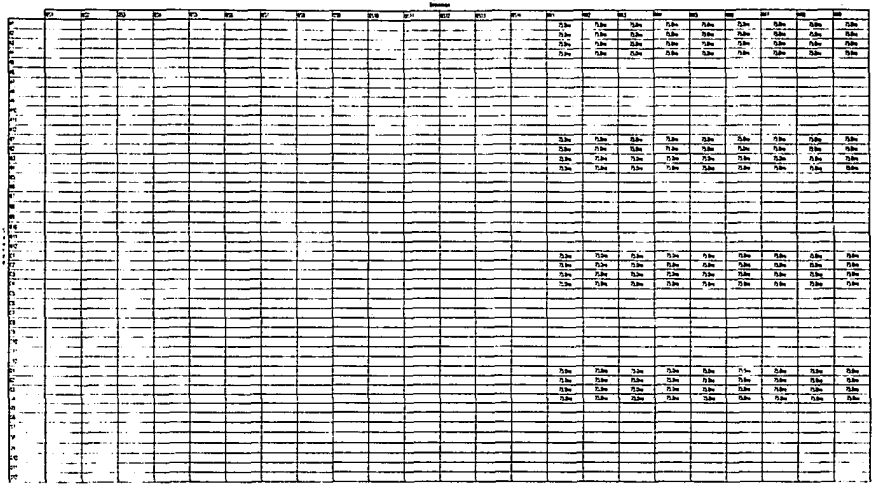


Figura 19c Máximos retardos en el circuito UNIDSYR0

La estructura del sistema MEPSI1 es la siguiente :

1) La unidad de procesamiento comprende los módulos UNIDSYR0 y CIRDIV1; cabe aclarar que el módulo CIRDIV1 es semejante al CIRDIV, suprimiendo solamente el módulo CONTROL-0; puesto que se realizarían dos divisiones en paralelo y con un sólo módulo de control, fue posible dirigir el proceso completo.

2) El subsistema de control fue el módulo CONTROL-0. Considerando que el tiempo de propagación del módulo UNIDSYR0 es de 75 ns y el periodo de trabajo del circuito divisor es de 53 ns; podemos afirmar que la estructura interna del módulo de control es adecuada, por que considera dos ciclos de reloj (106 ns) como tiempo de retardo del módulo UNIDSYR0.

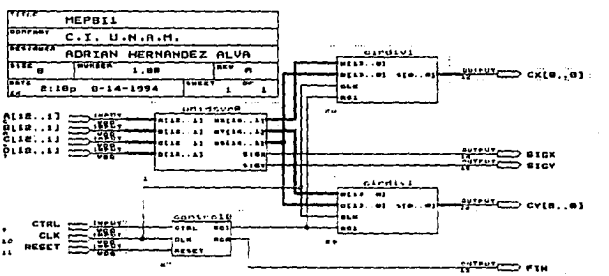


Figura 4.20a Archivo tipo gráfico para la construcción del sistema digital MEPSI1

Project Information c:\max2work\mepsi1.rpt
MAX+plus II Compiler Report File
Version 4.01 2.07.94
Compiled: 08/06/94 19:52:01
**** Project compilation was successful
Title: MEPSI1
Company: C.I. U.N.A.M.
Designer: ADRIAN HERNANDEZ ALVA
Rev: A
Date: 7 27p 8-06-1994
** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	LCs	Shareable Expanders	% Utilized
mepsi1	EPM7256EGC192	50	12	0	237	226	92 %
mepsi11	EPM7256EGC192	4	10	0	14	2	5 %
TOTAL:		54	22	0	251	228	49 %
User Pins:		51	21	0			

Figura 4.20b Reporte de construcción del sistema digital MEPSI1

El Sistema Digital MEPSI1, es capaz de resolver las ecuaciones 4.1 y 4.2 en sólo 660 ns, tiene el inconveniente de utilizar dos de los dispositivos de mayor tamaño, siendo necesario sólo un cinco por ciento de los recursos de uno de ellos; este problema se podría haber resuelto considerando lo siguiente:

en el módulo que realiza todas las adiciones y sustracciones (UNIDSYR0), existen dos sumadores que tienen como objetivo sumar uno al complemento del resultado de las sustracciones (para así obtener el complemento a dos); si consideramos que se trabaja con números de doce bits, el sumar uno es poco significativo; de acuerdo a lo anterior podríamos eliminar esos sumadores. Sin embargo esto no se realizó porque no se conoce el comportamiento exacto de las variables A, B, C y D; y no sería posible asegurar que la acción anterior no tendría consecuencias.

Por otro lado, para desarrollar el sistema digital MEPSI1 se utilizó la versión con menor tiempo de retardo por compuerta (EPMAX7256-12). La diferencia de precios entre el EPLMAX7256-12 y el EPLMAX7256-20 motivó el estudio de alternativas que permitieran usar el dispositivo menos oneroso.

Al desarrollar el circuito UNIDSYR0 en un EPMAX7256-20, se obtuvó un tiempo de propagación de 122 ns, si consideramos que este módulo fue creado a partir de la arquitectura del sumador que menos recursos emplea (STEXT), podemos afirmar que ese módulo es muy eficiente: por lo anterior, las posteriores actividades se orientaron a mejorar el diseño del circuito divisor.

4.5.2 MEPSI3

Si se usa un EPLMAX7256-20 para desarrollar el circuito divisor CIRDIV, se observa que la frecuencia máxima de trabajo se reduce a 10 MHz, esto es consecuencia de que el sumador tipo STEXT presenta un tiempo de propagación de 80 ns y los otros elementos que forman la trayectoria crítica (el multiplexor y el registro de corrimiento) generan un retardo de 20 ns. Haciendo que el sistema MEPSI1 requiera 1.2 microsegundos para procesar un grupo de datos.

Se decidió utilizar la arquitectura del sumador con acarreo anticipado para sustituir el sumador tipo STEXT, considerando que el tiempo de propagación tiene prioridad sobre la cantidad de recursos usados.

El archivo DIVISOR3 (figura 4.21), corresponde a la construcción del circuito divisor; como se puede apreciar, el circuito divisor completo fue desarrollado en el editor de textos.

```
FUNCTION FADD15B (CI A[14..0],B[14..0]) RETURNS (S[14..0],CS) ;  %funciones prototipo usadas%

SUBDESIGN DIVISOR3

(N[14..1],D[14..1],RC1 CLK      : INPUT ;                       %asignar entradas y salidas%
S[9..1]                         ; OUTPUT ;)

VARIABLE
        A1D[14..1],A2D[9..1]    : DFF ;                         %nombres de las funciones%
        MS                      : FADD15B ;                     %prototipo%
BEGIN
A1D[].CLK = CLK ;
A1D[].PRN = VCC ;                                               %habilitar registros%
A2D[].CLK = CLK ;       % AD1[14..1] CORRESPONDE AL REGISTRO QUE ALMACENA EL RESIDUO %
A2D[].PRN = VCC ;
A1D[].CLRN = VCC ;      % AD2[9..1] CORRESPONDE AL REGISTRO QUE ALMACENA EL COCIENTE %
A2D[].CLRN = RC1 ;

A2D[9..2].D = A2D[8..1].q ;                                     %habilitar sumador%

MS.A[14..1] = A1D[14..1].q ;
MS.A[0] = GND ;
MS.B[14] = VCC ;                % MS ES EL NOMBRE DEL SUMADOR DE 15 BITS QUE REALIZA %
MS.B[13..0] = NOT(D[]) ;        % LA COMPARACION Y LA SUSTRACCION %
MS.CI = VCC ;
A2D[1].d = NOT(MS.S[14]) ;

IF  RC1  THEN
```

```
        IF MS.S[14] THEN
                A1D[14..2].d = A1D[13..1].q ;          % CONDICIONES PARA EL FLUJO DE DATOS %
                A1D[1].d = GND ;
        ELSE                                           % MC.S[14] CORESPONDE AL BIT MAS SIGNIFICATIVO%
                A1D[14..1].d = MS.(S[13..0]) ;         % DE LA SUMA, ES DECIR, ES LA VARIABLE c %
        END IF ;                                       % INDICADA EN LA CARTA ASM 2 %
ELSE
        A1D[14..1].d = N[14..1] ;
END IF ;
S[] = A2D[].q ;

END;
```

Figura 4.21a Archivo tipo texto correspondiente al diseño del circuito divisor CIRDIV3.

```
Project Information                    c:\max2work\cirdiv3.rpt
MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled 08/14/94 17:02:17
***** Project compilation was successful
** DEVICE SUMMARY **
Chip/                                  Input  Output  Bidir      Shareable
POF     Device                         Pins   Pins    Pins   LCs Expanders % Utilized
cirdiv3 EPM7256EGC192-20               30     9       0      103 49        40 %
User Pins:                             30     9       0
```

Figura 4.21b Reporte de construcción del circuito divisor CIRDIV3

Si usamos un EPMAX7256-20, este circuito es capaz de trabajar a una frecuencia de 15.6 MHz (como se muestra en figura 4.21c); por lo que al emplearlo como parte del sistema digital final (MEPSI3) sería posible procesar las ecuaciones 4.1 y 4.2 en 780 ns.

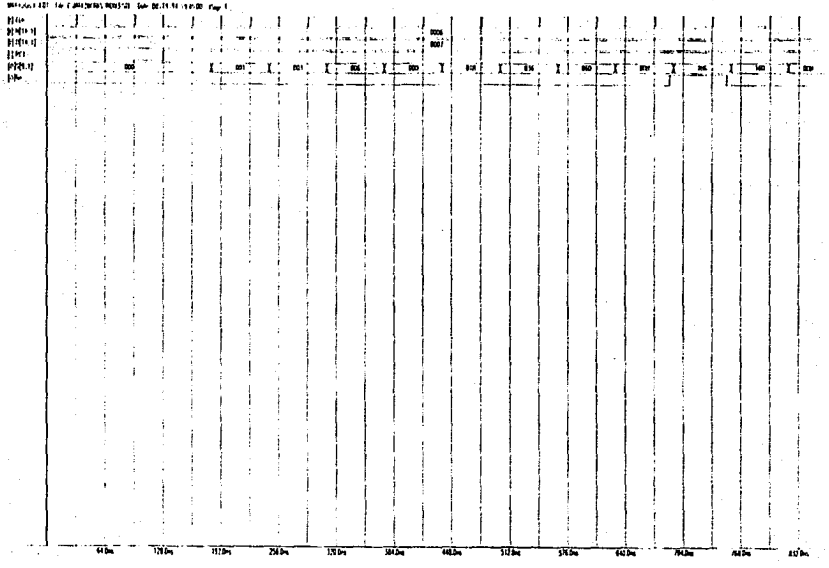


Figura 4.21c Simulación del funcionamiento del circuito CIRDIV3

La estructura del Sistema Digital MEPSI3 es la siguiente :

1. El subsistema procesador de datos utiliza el módulo UNIDSYR0 para realizar las adiciones y sustracciones; los módulos DIVISOR3 realizan las divisiones.

2. El módulo CONTROL-0 se encarga de dirigir el flujo de información.

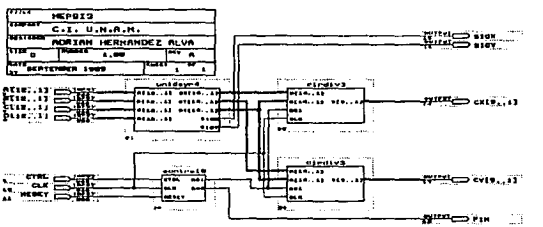


Figura 4.22a Archivo tipo gráfico que desarrolla el sistema digital MEPSI3

Project Information c:\max2work\mepsi3.rpt
MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 06/05/94 18:41:54
***** Project compilation was successful
** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	Shareable LCs	Expanders	% Utilized
mepsi3	EPM7256EQC192-20	47	24	0	183	116	73 %
mepsi31	EPM7256EQC192-20	44	36	0	197	126	78 %
TOTAL:		92	60	0	385	252	75 %
User Pins:		51	21	0			

Figura 4.22b Archivo reporte de construcción del sistema digital MEPSI3

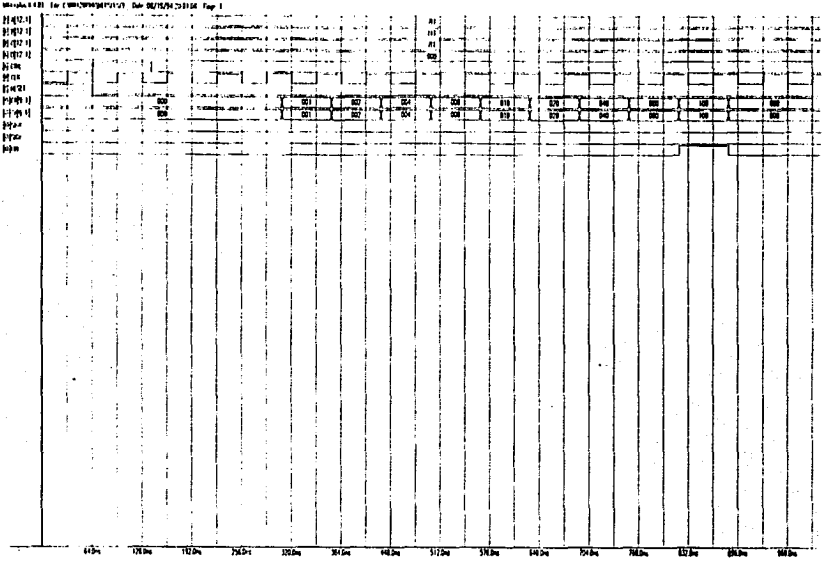


Figura 4.22c Simulación del funcionamiento del sistema MEPSI3

4.5.3 MEPSI5

La cantidad de recursos que utiliza el sumador con acarreo anticipado, fue el factor decisivo en el uso de dos dispositivos para desarrollar el sistema MEPSI3; empleándose en el siguiente diseño la arquitectura del sumador SFAAD (que requiere menos recursos), para realizar la comparación y la sustracción necesaria en la división.

```
%funciones prototipo usadas%
FUNCTION 8FADDB (CIN,A[8..1],B[8..1]) RETURNS (SUM[8..1],COUT) : %SUMADOR TIPO SFAAD%
SUBDESIGN DIVISOR4
(N[14..1],D[14..1],RC1 CLK      : INPUT : %asignar entradas y salidas%
S[9..1]                         : OUTPUT :)

VARIABLE
        A1D[14..1],A2D[9..1]    : DFF :   %SE DECLARAN DOS REGISTROS UNO DE 14 Y OTRO DE 9 BITS%
                                          % A1D CORRESPONDE AL RESIDUO Y A2D AL COCIENTE %
        MS[2..1]                : 8FADDB :  % SE USAN DOS SUMADORES TIPO SFAAD (8 BITS CADA UNO)%
BEGIN
A1D[].CLK = CLK ;
A1D[].PRN = VCC ;                                   %habilitar registros%
A2D[].CLK = CLK ;
A2D[].PRN = VCC ;
A1D[].CLRN = VCC ;
A2D[].CLRN = RC1 ;
A2D[9..2].D = A2D[8..1].q ;  % SE CREA UN REGISTRO DE CORRIMIENTO A LA IZQUIERA %
                             % PARA ALMACENAR EL COCIENTE %

% SE DECLARAN LAS ENTRADAS A EL SUMADOR%
MS1.(CIN) = VCC ;                   % SE USA EL ACARREO DE ENTRADA PARA SUMAR UNO%
MS1.(A[1]) = GND ;                  % EL BIT MENOS SIGNIFICATIVO DEL SUMANDO A  (A1), QUEDA LIBRE %
MS1.(A[8..2]) = A1D[7..1].q ;
MS2.(A[7..1]) = A1D[14..8].q ;      % LOS DATOS CORRESPONDIENTES AL RESIDUO (A1D[ ].q) %
MS2.(A[8]) = GND ;                  % SE PRESENTAN DESPLAZADOS EN LAS ENTRADAS DEL %
MS2.(CIN) = MS1.(COUT) ;            % SUMANDO A, PARA REALIZAR LA MULTIPLICACION POR DOS %
MS1.(B[8..1]) =NOT (D[8..1]) ;
MS2.(B[6..1]) =NOT (D[14..9]);
MS2.(B[7]) = VCC ;                  % LOS DATOS CORRESPONDIENTES AL DIVISOR (D[ ]) %
MS2.(B[8]) = VCC ;                  % SE ENTREGAN NEGADOS %

A2D[1].d = NOT(MS2.SUM7) ;          %EL BIT MS2.SUM7,  ES EL BIT QUINCE DE LA SUMA Y CORRESPONDE%
                                    % AL VALOR DE LA VARIABLE DE CONTROL " c " DE LAS CARTAS ASM %
IF  RC1  THEN               % SI RC1 = 1 ENTONCES %

        IF MS2.SUM7 THEN            %SI C =  1 ENTONCES %

        A1D[14..2].d = A1D[13..1].q ;       % REALIZAR UN CORRIMIENTO A LA IZQUIERDA%
        A1D[1].d = GND                      % ES DECIR,  MULTIPLICAR AL RESIDUO POR DOS %

        ELSE                        % SI C = 0 ENTONCES%

        A1D[14..9].d = MS2.(SUM[6..1])      % ACEPTAR EN EL REGISTRO 1  LA SALIDA DEL SUMADOR%
```

```
        A1D[8..1].d = MS1.(SUM[8..1]) :   % ES DECIR EL NUEVO RESIDUO ES EL RESULTADO DE LA RESTA%

        END IF ;

ELSE                        % SI RC1 = 0 ENTONCES%

        A1D[14..1].d = N[14..1] :       % SE ACEPTA EN EL REGISTRO 1 EL VALOR DE N[ ]%
                % O EN OTRAS PALABRAS SE ACEPTA EL DIVIDENDO COMO PRIMER RESIDUO %
END IF ;

S[] = A2D[].q ;     % LA SALIDA DEL MóDULO ES LA SALIDA DEL REGISTRO A2D[] (COCIENTE)%

END;
```

Figura 4.23a Archivo tipo texto para la construcción del circuito divisor DIVISOR4

Este circuito divisor es capaz de trabajar a una frecuencia de 16.4 MHz es decir con un periodo de 61 ns; este resultado muestra que este diseño es más eficiente que el desarrollado utilizando el sumador más rápido (SCLAD); así se comprueba lo explicado al principio de este capítulo: no es posible predecir como se comportara un diseño desarrollado con un PLDS, hasta que es analizado completamente.

```
Project Information                    c:\max2work\divisor4.rpt
MAX+plus II Compiler Report File
Version 4.01 2/07/94
Compiled: 08/06/94 20:13:05
***** Project compilation was successful
** DEVICE SUMMARY **
Chip/                Input  Output  Bidir    Shareable
POF    Device        Pins   Pins    Pins   LCs  Expanders  % Utilized

divisor4 EPM7256EGC192-20 30     9       0     41    25         16 %

User Pins:           30     9       0
```

Figura 4.23b Archivo reporte del circuito divisor DIVISOR4

Se presentan algunas simulaciones del funcionamiento del módulo divisor4 en las siguientes tres páginas.

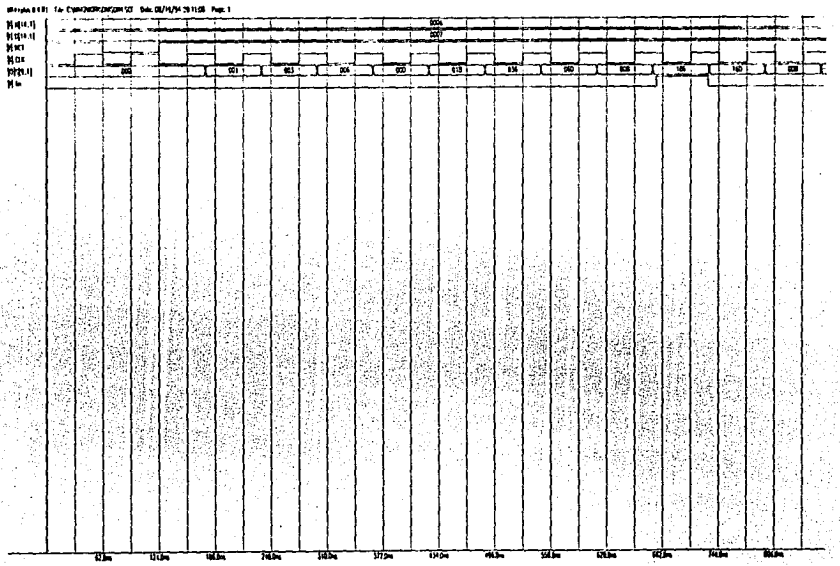


Figura 4.23c Simulación del funcionamiento del circuito DIVISOR4

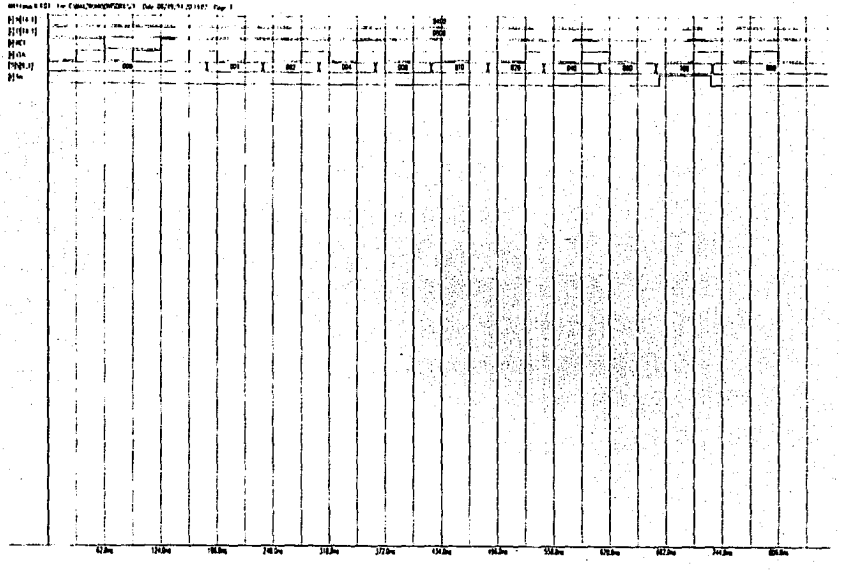


Figura 4.23c Simulación del funcionamiento del circuito DIVISOR4

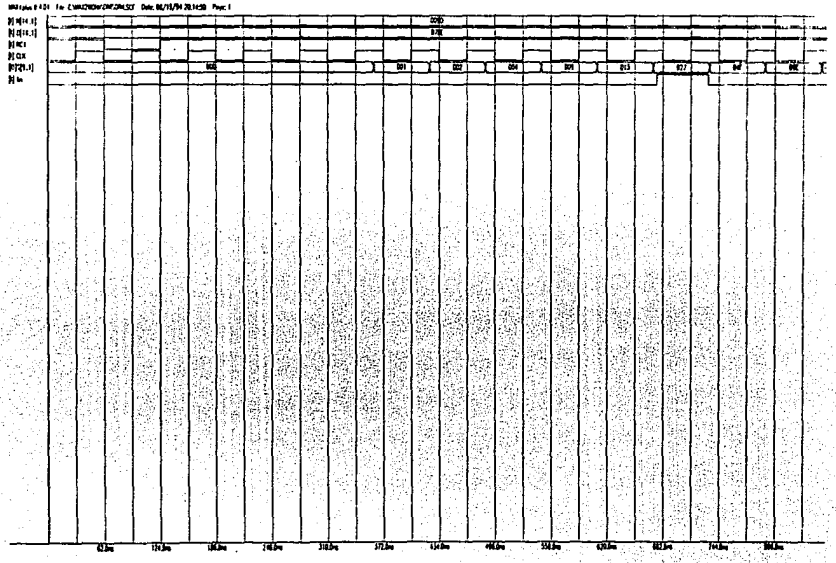


Figura 4.23c Simulación del funcionamiento del circuito DIVISOR4

El sistema digital MEPSI5 fue desarrollado utilizando los módulos:
a) UNIDSYR0 y DIVISOR4 como subsistema procesador y,
b) CONTROI-0 como subsistema de control.

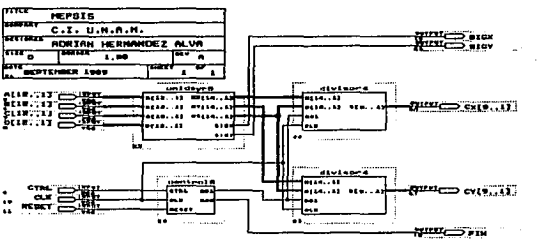


Figura 4.24 Archivo de construcción del sistema digital MEPSI5

El "MEPSI5" cumple con la condición de resolver las ecuaciones 4.1 y 4.2 en menos de 1.02 microsegundos; como se puede apreciar en las figuras procesa información en sólo 840 ns y ocupa un solo CPLD MAX7256-20.

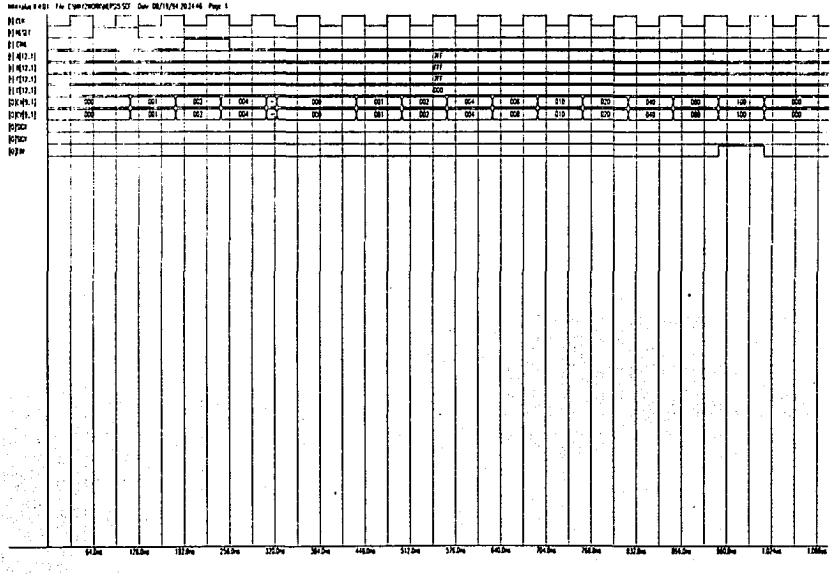


Figura 4.24b Simulación del funcionamiento del sistema digital MEPSI5

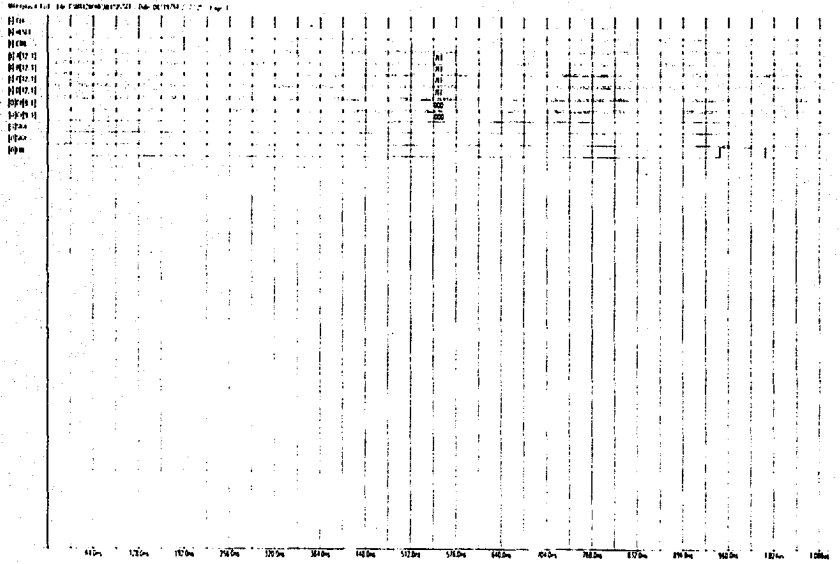


Figura 4.24b Simulación del funcionamiento del sistema digital MEPSI5

4.6 CARACTERISTICAS FINALES

Nombre del sistema digital : MEPSI5

Entradas :

- Una señal (reset), que debe verificarse alta durante el primer ciclo de reloj posterior al encendido del detector MEPSICRON; esta señal reestablece las condiciones iniciales de los registros que forman el subsistema de control.
- Cuatro números binarios, cada uno formado por doce bits; estas entradas se reciben en paralelo.
- Una señal (ctrl), que se verifica alta en el momento que debe iniciar el procesamiento de información, es decir, cuando el convertidor analógico-digital termina de procesar un grupo de datos.
- La señal de sincronización (clk), el ciclo de trabajo de esta señal es indistinto, ya que los registros se accionan con el flanco positivo de la señal.

Salidas :

- Dos números, cada uno formado por nueve bits; estos corresponden a las coordenadas X y Y de las ecuaciones 4.1 y 4.2.
- Una variable que corresponde al signo de la coordenada X (sigx)
- Una variable que corresponde al signo de la coordenada Y (sigy)

En ambas salidas, sí el valor es uno (lógico), el valor de la coordenada se considera negativo. Todos estos datos de salida se presentan en paralelo, con lo que se asegura que la transferencia de datos a la computadora a través de un bus de datos de treinta y dos bits, ocupara un mínimo de tiempo.

- Una variable (fin) que se verifica alta indicando que la información de las otras salidas es válida.

Consumo de potencia :

De acuerdo a la siguiente figura, suministrando 5 V de C.D. la disipación de potencia es de aproximadamente 2.5 W (a temperatura ambiente).

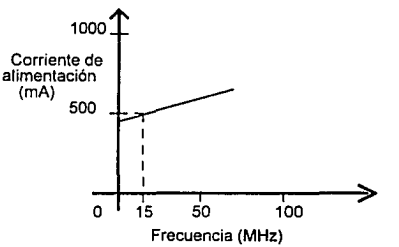


Figura 4.26 Relación Frecuencia-Corriente de Mantenimiento para el EPMAX7256-20

Cantidad de recursos :

El sistema fue desarrollado en un dispositivo EPMAX2756-20 utilizando el 99 por ciento de sus recursos; las dimensiones de este circuito integrado son 4.35 cm por lado cumpliendo con los límites de espacio.

Tiempo de respuesta :

El sistema puede procesar la información correspondiente a un evento, en doce ciclos de reloj; trabajando a una frecuencia de 15 MHz, esto representa ochocientos nanosegundos: cumpliendose el objetivo con respecto al máximo tiempo de propagación.

La frecuencia máxima de operación es de 15.4 MHz.

5 CONCLUSIONES

Los resultados obtenidos con el sistema digital MEPSI5, me permiten afirmar que cumplí los objetivos del proyecto, el tiempo máximo de propagación fue veinte por ciento menor al límite establecido y las dimensiones del circuito permiten que sea facilmente integrado al detector MEPSICRON sin realizar grandes modificaciones. Un aspecto importante es que lo anterior se logró a pesar de usar un dispositivo relativamente lento (EPMAX7256-20), con lo que se evitó tener que recurrir a un dispositivo con menor tiempo de propagación por compuerta (el EPMAX7256-12) pero tres veces más caro.

Lo anterior fue posible por :

- haber diseñado una arquitectura en paralelo
- seleccionar adecuadamente los algoritmos empleados

La estructura de las ecuaciones 4.1 y 4.2, permitió realizar varias operaciones en forma simultánea; el aprovechar esta característica hizó posible desarrollar un sistema con menor tiempo de propagación, que aquel que podría obtenerse con algunos dispositivos especialmente diseñados para realizar operaciones aritméticas y que trabajan a mayores frecuencias, pero que realizan el procesamiento de información secuencialmente.

Por otro lado, el haber diseñado un circuito divisor que permite calcular un dígito del cociente por cada ciclo de reloj y el haber seleccionado los sumadores adecuados en cada módulo del sistema, permitió obtener un diseño eficiente con respecto a los recursos empleados y al tiempo de respuesta.

A continuación, se compara el desempeño del circuito diseñado, contra dos circuitos comunmente usados para este tipo de aplicaciones.

El usar un microprocesador 80486 de INTEL, para calcular el valor de las ecuaciones 4.1 y 4.2, implica usar aproximadamente ciento veinte ciclos de reloj lo que representa un tiempo de respuesta de 1.8 microsegundos, si se trabaja a 66 MHz.

Si usaramos el DSP56116 de MOTOROLA, serían necesarios sesenta y seis ciclos de reloj, lo que significa1.65 microsegundos, trabajando a 40 MHz.

Con el circuito desarrollado en el presente trabajo se requieren sólo doce ciclos de reloj para realizar las mismas operaciones (800 ns, si se trabaja a 15 MHz).

No es posible afirmar, que el diseño realizado es mejor que un DSP56116 o mejor que un microprocesador 80486; debido a que se diseñó un sistema digital cuyas características se adaptan perfectamente a las del detector MEPSICRON, ese diseño se comporta mejor que un sistema que use dispositivos diseñados para aplicaciones generales.

La desventaja del circuito desarrollado en este trabajo, es el precio; un DSP56116 cuesta aproximadamente 240 nuevos pesos, en tanto que un EPMAX7256-20 tiene un costo de 600 nuevos pesos; sin embargo, se debe tomar en cuenta que actualmente el Centro de Instrumentos no cuenta con un sistema de desarrollo para procesadores digitales de señales y adquirirlo representaría un costo superior a los 2,100 nuevos pesos.

El sistema diseñado puede ser considerado para formar parte del detector MEPSICRON: la decisión final depende de los responsables del proyecto.

COMENTARIO ADICIONAL

Los Dispositivos Programables para Lógica Compleja (CPLDs), son una alternativa muy interesante para resolver problemas específicos, un PLDS hace sencillo diseñar sistemas digitales, pero el obtener circuitos realmente eficientes depende de la capacidad del diseñador.

Actualmente existen en México sólo tres sistemas de desarrollo como el descrito en este documento; por la experiencia que adquirí durante el tiempo que he trabajado con este sistema, puedo afirmar que es conveniente que en la asignatura de Diseño de Sistemas Digitales[1], los alumnos tengan acceso a un sistema de este tipo. La anterior afirmación la puedo fundamentar considerando que :

[1]la cual se imparte en los últimos semestres de las carreras de ingeniería en computación y electrónica en la Facultad de Ingeniería de la UNAM

- la Organización de las Naciones Unidas ha recomendado el uso de estos sistemas para la enseñanza de materias relacionadas con el diseño digital.

- los CPLDs por sus características, son dispositivos que estan desplazando rápidamente a los arreglos lógicos programables; circuitos que son estudiados en la asignatura de diseño de sistemas digitales.

AGRADECIMIENTOS

A los ingenieros Yukihiro Minami Koyama, Jorge Rodríguez Cuevas y Gabriel Jaramillo Morales, quienes me brindaron su apoyo y consejos a lo largo de mi estudios profesionales.

A los Ingenieros Miguel Angel Bañuelos Saucedo, José Castillo Hernández, Wilfredo Martínez Payan y al Físico Alejandro Padrón Godínez; por sus comentarios acerca de este documento.

Al Maestro en Ciencias José Luis Pérez Silva, por todos los incentivos, el apoyo y la confianza que recibí durante el desarrollo del presente trabajo.

BIBLIOGRAFIA

1. Pérez S.,Fuentes G., Nogueira J. "Electrónica para detector MEPSICRON" Memorias del VIII Congreso Nacional de Instrumentación, pp 50-52, 1993.

2. Morris, M. M. Digital Logic and Computer Design. Englewood Cliffs, N.J.; Prentice-Hall Inc., 1989.

3. ALTERA Data Book. San José, Calif.: ALTERA Corporation.,1991

4. MAXPLUS II User Guide. San José, Calif.: ALTERA Corporation.,1991

5. ALTERA Applications Handbook;San José, Calif.: ALTERA Corporation.,1991

6. DSP56116 User's Manual. Phoenix, Ariz.: Motorola Semiconductor Products, Inc. 1990.

7. DSP56000/DSPP56001 User's Manual. Phoenix, Ariz.: Motorola Semiconductor Products, Inc. 1990.