58

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y DESARROLLO DE LAS TARJETAS OBJETIVO DE UNA COMPUTADORA TOLERANTE A FALLAS

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A:

JUAN ANTONIO MEJIA GALEANA

DIRECTOR DE TESIS: M. I. ESAU VICENTE VIVAS

CIUDAD UNIVERSITARIA ENERO DE 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

A mis padres,
por su tiempo, dedicación y gran amor.

A mis hermanos,
por las alegrías de todos los días.

A *My pictures of you,*
por el deleite de la imaginación cotidiana.

A mi gente,
por la razón de su existencia.

Agradecimientos

En primer lugar mi gratitud a mi asesor Esaú Vicente Vivas por su gran apoyo, dedicación, paciencia y todo el cúmulo de experiencias y enseñanzas recibidas en todos estos meses.

A los académicos y becarios de la Coordinación de Automatización del Instituto de Ingeniería, por su apoyo, ideas, asesorías y los gratos momentos, especialmente a la Doctora Cristina Verde, José Mondragón, Martín López y Roberto Moreno.

A mis compañeros de equipo Rogelio Rivera y Antonio Echeverría.

A los personajes reales e imaginarios con cuya invaluable asistencia he superado el vértigo de estos días, he sobrevivido al ozono de la atmósfera defeña y me permitió la culminación de este capítulo en mi vida.

A todos ellos, gracias.

Introducción 1

1 Conceptos de Tolerancia a Fallas 5
1.1 Introducción 5
1.2 Concepto y Objetivos de la Tolerancia a Fallas 6
1.3 Fallas, Errores y Averías 7
1.4 Aplicaciones de la Computación Tolerante a Fallas 8
1.4.1 Aplicaciones de Larga Vida 9
1.4.2 Aplicaciones de Cómputo Crítico 9
1.4.3 Aplicaciones de Mantenimiento Aplazable 10
1.4.4 Aplicaciones de Alta Disponibilidad 10
1.5 La Computadora Tolerante a Fallas del Instituto de Ingeniería 11

2 Arquitectura de la Computadora Tolerante a Fallas 13
2.1 Introducción 13
2.2 Unidades de Procesamiento (UPs) 14
2.3 Unidades de Detección de Fallas y Control (UDFCs) 15
2.4 Unidades de Aislamiento 15
2.5 Unidad de Interfaz Múltiple 16
2.6 Ductos utilizados 17

3 Diseño de Unidades de Procesamiento 24
3.1 Introducción 24
3.2 Arquitectura de UPs 25
3.3 Técnica de Sincronización para trabajo concurrente 29

3.4 Autómata de Detección de Operaciones 32
3.5 Teclado Compartido en la CTF 36
3.6 Elaboración del Circuito Impreso 37

4 Diseño de la Unidad de Detección de Fallas y Control 45
4.1 Introducción 45
4.2 Arquitectura de las UDFCs 46
4.3 Autómata de Multiplexaje de Datos 51
4.4 Autómata de Comparación de Datos y de Reconfiguración de la Arquitectura TF 54
4.5 Inicialización y Mantenimiento en Línea de la Arquitectura TF 61
4.6 Mecanismos de Autodiagnóstico de la Tarjeta 62
4.7 Protocolo de Comunicación entre UDFCs y UPs 66
4.8 Diseño del Circuito Impreso 68
4.9 Unidad de Interfaz Múltiple 74

5 Programación de Unidades de Procesamiento y de UDFCs 77
5.1 Introducción 77
5.2 Rutinas de Control para el 68HC11 contenido en las UDFCs 78
5.3 Programación de Interrupciones en UPs 91

6 Conclusiones y Recomendaciones 95
6.2 Conclusiones 96
6.3 Recomendaciones 98

Apéndice A Filosofías de Diseño y Técnicas de Tolerancia a Fallas 101
A.1 Filosofías de Diseño para Combatir Fallas 101
A.2 Técnicas de Diseño en Tolerancia a Fallas 102
A.2.1 Redundancia por *Hardware* 103
A.2.1.1 Redundancia Pasiva por *Hardware* 104
A.2.1.2 Redundancia Activa por *Hardware* 106
A.2.1.3 Redundancia Híbrida por *Hardware* 110
A.2.2 Redundancia por *Software* 114
A.2.2.1 Examinador de Consistencia 115
A.2.2.2 Examinador de Capacidad 115
A.2.2.3 Programación de N versiones 115

Apéndice B El Microprocesador 80386 y su juego de Cis TACT83000 **117**
B.1 El Microprocesador 80386SX 117
B.2 Circuitos Integrados VLSI TACT 83000 120

Apéndice C Estadísticas de los circuitos impresos diseñados **124**

Apéndice D Hojas de especificaciones técnicas **130**

Referencias **137**

Bibliografía **139**

INTRODUCCIÓN

El avance de la civilización ha ido acompañado por una necesidad siempre creciente de cálculos numéricos y el manejo de volúmenes de datos cada vez más grandes. Por consiguiente, a lo largo de los años, el hombre ha desarrollado sistemas de cálculo más veloces y capaces de manipular cantidades mayores de información, que a su vez le auxilian en la toma de decisiones en los diversos ámbitos donde se desenvuelve. Con la inserción de los sistemas digitales de cómputo (computadoras) en la vida de los seres humanos, parecería ser que se ha llegado al umbral del perfeccionamiento de los sistemas de cálculo que alguna vez planteara Charles Babbage (la máquina analítica) [Aréchiga et al. 1978]. Sin embargo, también es cierto que tal *perfeccionamiento* es relativo, pues aunque en la actualidad contamos con poderosas computadoras que en unos cuantos segundos realizan el cálculo de las órbitas celestes que a Gauss le tomó varios años, estos sistemas electrónicos se pueden *equivocar*, o dicho de otra forma, existe la probabilidad de que incurran en fallas.

El concepto de tolerancia a fallas ha tomado gran relevancia en las últimas décadas debido al incremento en el uso de las computadoras en aspectos vitales de la actividad humana. Algunas computadoras ya no son esas poderosas calculadoras cuyo mal funcionamiento originaba pérdida de tiempo y cierta frustración. Por el contrario, ahora las computadoras se utilizan en transacciones bancarias, sistemas de control aéreo comercial y militar, controladores industriales, aplicaciones espaciales, entre otras. Por ello, el comportamiento erróneo de las computadoras es catastrófico en aplicaciones críticas relacionadas entre otras con registros financieros, medio ambiente y la misma vida humana. En suma, la tolerancia a fallas es de gran importancia debido a que el trabajo de las computadoras y los sistemas digitales en aplicaciones riesgosas se ha convertido en un asunto crucial.

La computación tolerante a fallas comienza por suponer que los sistemas digitales son susceptibles de incurrir en diferentes tipos de fallas y pretende alcanzar altos índices de confiabilidad en una aplicación en particular, incorporando en el equipo varios tipos de redundancias. Estas redundancias pueden, en algunos casos, permitir que el sistema continúe sus operaciones aun ante la presencia de una o más fallas. En otros casos, la redundancia puede ayudar a minimizar el daño originado por una falla sin que necesariamente el sistema siga operando. Y en otras situaciones, la redundancia facilita el diagnóstico y la reparación de un sistema para reducir el tiempo que permanece sin operar. Las cantidades y tipos de redundancias requeridas dependen exclusivamente de la aplicación y sus posibles repercusiones en caso de que el sistema falle.

El uso de técnicas de redundancia y detección de errores se acentuó en los 50s debido a que las computadoras de los 40s y principios de los 50s se caracterizaban por su gran volumen, peso, elevado consumo de energía y la ocurrencia de fallas durante la ejecución de alguna tarea. Fue entonces cuando se idearon modelos que integraban componentes de repuesto, en su mayoría tubos al vacío, relevadores y dispositivos de almacenamiento [Pierce 1965].

Por esos años hubo contribuciones importantes [Von Neumann 1956], [Moore & Shannon 1956] de esquemas para tener equipos con ciertos índices de confiabilidad, así mismo se sentaron las bases teóricas para el uso de técnicas de redundancias [Von Neumann 1956], [Moore & Shannon 1956], [Kuehn 1969].

Con el advenimiento del transistor, los núcleos de ferrita y posteriormente los circuitos integrados, se logró elevar los índices de confiabilidad por lo cual la investigación en el campo de la tolerancia a fallas se redujo considerablemente. Sin embargo, siguieron los desarrollos en aquellas aplicaciones muy especializadas en donde se tenía la necesidad de contar con equipos de larga vida, funcionamiento ininterrumpido o donde la vida humana estuviera en peligro debido a fallas del equipo.

Dentro de tales aplicaciones altamente especializadas se tienen los sistemas de conmutación telefónica [Toy 1978], así como sistemas espaciales [Sklaroff 1976], [Lerner 1982]. Estos últimos sistemas cobraron gran relevancia debido al auge por la supremacía en la carrera espacial de las grandes potencias económicas y militares.

En la década de los 60s se reinicia el interés en los sistemas tolerantes a fallas (STF) y en los 70s tuvieron una amplia gama de aplicaciones en diversos campos por la fuerte injerencia de los sistemas de cómputo en actividades de alto riesgo como el control de vías férreas, el control de tráfico aéreo, el monitoreo de reactores nucleares, etcétera. Fueron los sistemas de Tandem, entre otros, los que dieron respuesta a la demanda de sistemas confiables de alta disponibilidad en actividades donde se requería el manejo de grandes volúmenes de información, procesamiento y transacciones.

De hecho, se ha llegado a un punto tal que se están incorporando de forma regular atributos de tolerancia a fallas a los diseños de sistemas medianos y grandes, y existe una tendencia en nuestros días de introducir los principios de tolerancia a fallas en vehículos automotrices [Shladover 1993], [Zanoni 1993] debido a la flexibilidad de los sistemas electrónicos. Para los lectores interesados en tener un panorama más amplio acerca del desarrollo de la computación tolerante a fallas ver [Rennels 1984], [Vicente 1993].

El presente escrito describe el diseño y desarrollo de los módulos objetivo que integran una Computadora Tolerante a Fallas (CTF) que se construye en el Instituto de Ingeniería de la UNAM. En el primer capítulo se introducen los conceptos básicos para el estudio de los STF. A continuación, se explica la operación general de los subsistemas que integran la CTF (Capítulo II). En el Capítulo III se detalla el diseño y construcción de las unidades de procesamiento (UPs) de la CTF, proporcionando las características más relevantes de los circuitos integrados VLSI (microprocesador y *chip sets*); además de especificar la técnica de sincronización empleada para la operación concurrente de la computadora. También se incluye la electrónica dedicada para la detección de operaciones (lectura y escritura a puertos/memoria) empleada en las tareas de enmascaramiento de fallas; por último se explica el diseño del circuito impreso. La descripción acerca del diseño de las unidades de detección de fallas y control (UDFC) se efectúa en el Capítulo IV, resaltando la función de las dos partes esenciales de este subsistema: los autómatas de alta velocidad (autómata de multiplexaje de datos y el de detección de fallas y reconfiguración de la arquitectura). y el uso de un microcontrolador (para auxiliar las tareas de inicialización del sistema y de mantenimiento en línea). También se delinean los protocolos de comunicación entre las UPs y la UDFC, que son parte esencial de la detección de fallas, de la reconfiguración del sistema y del mantenimiento en línea. Se culmina esta sección con los detalles del circuito impreso de este módulo. El planteamiento de los algoritmos de

control del microcontrolador integrado en la UDFC y las rutinas generales de atención a interrupciones en las UPs se describen en el Capítulo V. Finalmente, en el Capítulo VI se presentan las conclusiones derivadas del trabajo desarrollado, además de las recomendaciones, que a juicio del autor, redundarían en un mejor desempeño del sistema TF propuesto.

UNO

Conceptos de Tolerancia a Fallas

1.1 Introducción

En este primer capítulo se detallan algunos conceptos que sirven como preámbulo para una mejor comprensión de las ideas que se presentan a lo largo de este escrito. Se describe la terminología básica usada para diferenciar las causas y efectos de las fallas, y enseguida se definen las técnicas del diseño tolerante a fallas (TF) basadas en el uso de redundancias por *hardware* y por *software*, además de aquellas basadas en redundancia de información y redundancia en el tiempo. También se muestran las diferentes arquitecturas (prácticas y teóricas) propuestas por otros autores para lograr el atributo de tolerancia a fallas, además de explicar los principios comúnmente usados en la programación TF.

Se presenta también un panorama de los campos de aplicación donde han tenido injerencia los sistemas tolerantes a fallas, mencionando ejemplos de sistemas TF de tipo comercial y de desarrollo científico.

En la última parte de esta sección se describen los rasgos generales de la CTF que se desarrolla en el Instituto de Ingeniería.

1.2 Concepto y Objetivos de la Tolerancia a Fallas

Los STFs son aquellos que realizan correctamente su tarea encomendada aun en presencia de anomalías en la electrónica (*hardware*) o errores en la programación (*software*). Por ejemplo, el efecto de un error en algún módulo electrónico de un STF se sobrelleva de tal forma que no perjudica la operación correcta y continua del sistema.

La Tolerancia a Fallas es una filosofía de diseño que integra atributos en un sistema para que éste alcance altos índices de desempeño. El diseño de STFs debe cumplir con ciertos objetivos de funcionalidad y observancia, sin embargo, dependiendo de la aplicación también pueden reunir otros indicadores, entre ellos los más significativos son los siguientes:

- **Confiabilidad.** La confiabilidad R(t) de un sistema es una función del tiempo, que se define como la probabilidad condicional de que el sistema trabaje correctamente en el intervalo $[t_0, t]$, dado que el sistema estaba operando adecuadamente en el instante t_0. En otras palabras, la confiabilidad es la probabilidad de que el sistema funcionará correctamente en un intervalo de tiempo. Por convención, cuando se reportan índices de confiabilidad se expresan como 0.9_i, donde i representa el número de nueves a la derecha del punto decimal.

- **Disponibilidad.** Es otro objetivo de diseño cuyo valor se puede incrementar a través de la tolerancia a fallas. La disponibilidad A(t) es función del tiempo, definida como la probabilidad de que el sistema esté operando correctamente y esté disponible para realizar sus funciones en un instante de tiempo t.

- **Seguridad.** Es un atributo que algunas veces se omite en los sistemas. La seguridad S(t) es la probabilidad de que el sistema ejecute sus tareas ya sea de forma acertada o, que cese su operación de manera tal que no perjudique las operaciones de otros sistemas, ni que comprometa la integridad de personas.

- **Operatividad.** La operatividad P(L,t) de un sistema es una función del tiempo, definida como la probabilidad de que el desempeño del sistema esté en, o sobre, algún nivel L en el instante de tiempo t.

- **Reparabilidad.** La reparabilidad es una medida de la facilidad con la cual un sistema se puede reparar, una vez que ha fallado. En otros términos, la reparabilidad M(t) es la probabilidad de que un sistema con falla sea llevado a un estado operativo en un período específico de tiempo.

- **Comprobabilidad.** La comprobabilidad es la habilidad de probar algunos atributos en un sistema. Las medidas de comprobabilidad permiten saber que tan fácil se realizan algunas pruebas. Una prueba es el medio por el cual se determina la existencia y la calidad de ciertos atributos en un sistema.

- **Formalidad.** El término formalidad agrupa los conceptos de confiabilidad, disponibilidad, seguridad, reparabilidad, operatividad y comprobabilidad. La formalidad es la calidad del servicio que proporciona un sistema en particular.

Los conceptos anteriores sirven para medir la formalidad de un sistema. Existen técnicas de modelado de sistemas TF que permiten obtener indicadores de la calidad de la arquitectura de que se trate, ver [Ng & Avizienis 1980]. El proyecto de la CTF de la UNAM contempla el modelado matemático del sistema, que es parte de la tesis de un integrante del grupo.

1.3 Fallas, Errores y Averías

En el campo de la tolerancia a fallas se hace una clara distinción en la conceptualización de fallas, errores y averías. Aunque en el lenguaje cotidiano, se emplean de manera indistinta para referirse a la misma circunstancia. La principal diferencia consiste en la relación que guardan estos tres conceptos, se trata de una conexión de causa y efecto. Esto es, una falla origina un error y la avería es la manifestación del error [Rennels 1984], [Avizienis et al. 1980].

Una falla es un defecto físico, una imperfección o una ruptura dentro de un módulo de *hardware* o *software*. Algunos ejemplos de fallas pueden ser: cortos circuitos entre conductores eléctricos, rupturas en conductores, impurezas en los dispositivos semiconductores. Mientras que una falla en programación podría ser la ejecución de un ciclo iterativo del cual no se puede salir [Vicente et al. 1986].

El error es la manifestación de una falla. Específicamente, un error es una desviación de lo exacto o correcto. Por ejemplo, la falla en la salida de una compuerta que se ha quedado amarrada a un nivel bajo; al cambiar las condiciones en las entradas de la compuerta se debería generar un cambio en la salida, pero por su condición de falla en la salida (amarrado a cero) el resultado será erróneo.

Finalmente, si el error se manifiesta en las actividades del sistema, se dice que el sistema está averiado. Esto quiere decir que, una avería es la no ejecución de una tarea que se debió haber realizado.

Para el lector interesado en conocer más acerca de las técnicas disponibles para erradicar fallas y los tipos de arquitecturas para STFs, en el Apéndice A se ha recopilado material suficiente para profundizar el estudio de tales tópicos.

1.4 Aplicaciones de la Computación Tolerante a Fallas

El uso de la computación tolerante a fallas se ha incrementado en un gran número de campos por varias razones. Primero, la investigación en STFs ha crecido, desde hace veinticinco años, de unas decenas de investigadores hasta compañías dedicadas exclusivamente a los sistemas tolerantes a fallas. Ejemplo de ellas son *Tandem Computers*, *Stratus Computers* y *August Systems*. Tandem y Stratus compiten en la industria del procesamiento de transacciones, mientras que August Systems se concentra en el desarrollo de sistemas tolerantes a fallas confiables para el control de procesos industriales [Serlin 1984]. Segundo, con la aparición de los sistemas de Alta Escala de Integración (*Large Scale of Integration,* LSI) y los de muy Alta Escala de Integración (*Very Large Scale of Integration,* VLSI) , las técnicas de tolerancia a fallas se han vuelto más prácticas. Finalmente, muchos sistemas que anteriormente eran de tipo mecánico ahora son electrónicos para aprovechar la gran capacidad y flexibilidad que ofrece la electrónica.

Las aplicaciones de la tolerancia a fallas pueden dividirse en cuatro áreas principalmente: aplicaciones de larga vida, cómputo crítico, mantenimiento aplazable y alta disponibilidad. Cada aplicación presenta diferentes requisitos de diseño y diversos retos.

1.4.1 Aplicaciones de Larga Vida

Los ejemplos comunes de las aplicaciones de larga vida son las sondas espaciales y los satélites. Estas aplicaciones deben tener una probabilidad de 0.95 de ser operativos al final de un período de diez años debido a la dificultad (en ocasiones imposible) y al gran costo implicado para realizar mantenimiento en el espacio [Rennels 1980]. A diferencia de otras aplicaciones, los sistemas de larga vida permiten la extensión de su vida útil al ser puestos en operación una vez más. Además, este tipo de aplicaciones permiten a veces ser reconfigurados remotamente por un operador. Ejemplos de sistemas diseñados para este tipo de aplicaciones son: la computadora STAR (*Self-Testing And Repairing*) [Avizienis et al. 1971] y el sistema *Fault-Tolerant Building Block Computer* (FTBBC) [Rennels 1980].

1.4.2 Aplicaciones de Cómputo Crítico

Tal vez las aplicaciones de mayor auge actual son aquellas donde los cálculos y la operación continua (segura) son críticos en diversas ramas productivas, para la seguridad humana, el cuidado del medio ambiente o para la seguridad del mismo equipo. Algunos ejemplos incluyen a los servidores de redes en sistemas bancarios, controladores industriales, control aéreo y sistemas militares. En estas aplicaciones, el funcionamiento incorrecto de los sistemas acarrearía indudablemente pérdidas económicas y/o resultados devastadores. Un requisito típico para estas aplicaciones es tener una disponibilidad del 0.9, al final de un periodo de tres horas.

Entre las aplicaciones críticas más fáciles de entender está la de los lanzamientos espaciales. Una falla en el sistema de control de vuelo ya sea en el ascenso o el descenso lleva a la pérdida de la nave. Por ello, se da extremo cuidado para asegurar que este tipo de sistemas desempeñe sus tareas con amplia seguridad. De hecho, el lanzamiento debe poder continuar sus funciones de control de vuelo después de hasta tres averías [Sklaroff 1976].

Además, en los próximos años crecerá notablemente el uso de STF en la industria automotriz, en la cual ya se tienen ejemplos y proyectos muy ambiciosos para controlar digitalmente algunos subsistemas como: dirección, suspensión, frenado, etcétera [Shladover 1993], [Zanoni 1993].

1.4.3 Aplicaciones de Mantenimiento Aplazable

Este tipo de aplicaciones se encuentran frecuentemente en aquellos lugares donde el mantenimiento es extremadamente costoso, inconveniente o difícil de llevar a cabo. Ejemplos de dichas aplicaciones son las estaciones de procesamiento remoto y algunas aplicaciones espaciales. En el espacio, el mantenimiento directo es sumamente costoso de realizar en el espacio cercano [Koczela 1968] y por el momento imposible en el espacio lejano. El principal objetivo de la tolerancia a fallas en este tipo de aplicaciones es la sustitución de partes defectuosas para que el equipo continúe operando. Los sistemas de conmutación telefónica [Toy 1978] constituyen otro ejemplo que requiere el aplazamiento del mantenimiento. Entre una sesión de mantenimiento y otra los sistemas deben manipular las averías y las discontinuidades de servicio de manera autónoma.

1.4.4 Aplicaciones de Alta Disponibilidad

La disponibilidad es un parámetro que se está convirtiendo en un factor clave en muchas aplicaciones. Los sistemas bancarios y otros sistemas de tiempo compartido son ejemplo de esta clase de aplicaciones. Los usuarios de estos sistemas deben tener una alta probabilidad de obtener el servicio cuando así lo pidan. El sistema de procesamiento de transacciones NonStop de Tandem [Katzman 1977] es un ejemplo de diseño de alta disponibilidad.

Un ejemplo de diseño desarrollado para soportar aplicaciones de propósito general de alta disponibilidad es el procesador 432 de Intel [Johnson 1984] el cual integró en su diseño técnicas de tolerancia a fallas.

1.5 La Computadora Tolerante a Fallas del Instituto de Ingeniería

En el Instituto de Ingeniería (II) de la UNAM se desarrolla una computadora tolerante a fallas, la cual cuenta con redundancias en la electrónica y posee amplias posibilidades de reconfiguración, para ofrecer la operación continua del equipo aun ante la presencia de fallas en alguno de sus subsistemas. Sus características principales son

la detección de fallas (de forma transparente para el usuario), la ubicación de las mismas, el confinamiento del módulo con falla y la reconfiguración del sistema [Vicente et al 1993].

El diseño electrónico modular de la CTF facilitó su desarrollo, además de permitir el atributo de mantenimiento en línea, mediante el cual se pueden insertar nuevas tarjetas o remover unidades con fallas, todo ello, sin tener que desenergizar el equipo ni interrumpir la ejecución de los programas de aplicación. La CTF integra once tarjetas electrónicas, de las cuales se tienen cinco circuitos impresos diferentes entre sí.

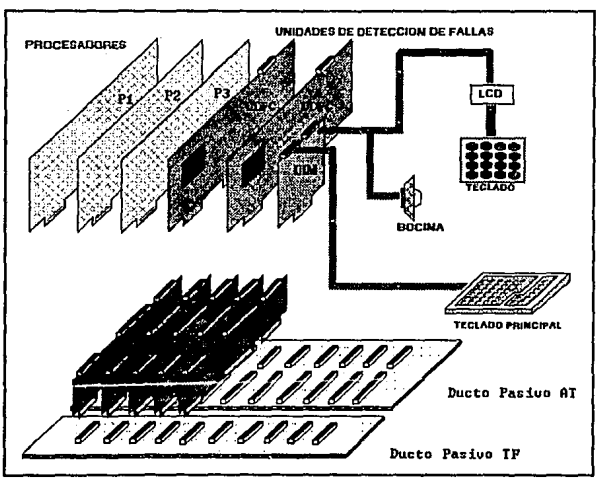


Figura 1.1 Tarjetas que integran la CTF del II UNAM

La arquitectura de la computadora está formada por seis tarjetas objetivo, cada una vinculada a una tarjeta de aislamiento, como se observa en la Figura 1.1. Tres de las tarjetas objetivo constituyen las unidades de procesamiento (UPs), basadas en microprocesadores 80386SX y los *chip sets* TACT83000; otras dos forman las

unidades de detección de fallas y control (UDFCs) y la última es la unidad de interfaz múltiple (UIM). En esta tarjeta auxiliar se conecta una pantalla de cristal líquido y un teclado pequeño para labores de mantenimiento preventivo/correctivo, además cuenta con conectores para el teclado principal y la bocina comunes a las tres UPs. En los siguientes capítulos se describirá el funcionamiento de la CTF y los módulos desarrollados como parte del trabajo de esta tesis.

Arquitectura de la Computadora Tolerante a Fallas

2.1 Introducción

En esta sección se describen los diferentes módulos que integran a una arquitectura de procesamiento concurrente basada en tres procesadores y dos unidades de detección de fallas y control, éstas últimas con capacidad de autodiagnóstico y de conmutación automática ante casos de falla. Se da un panorama general de las tarjetas, tanto de unidades de procesamiento como de la unidad de detección de fallas y control. También se menciona la necesidad de utilizar tarjetas de aislamiento para permitir el retiro de módulos en mal estado y la adición de tarjetas en buen estado con el fin de lograr la operación continua del equipo, aun ante la presencia de fallas importantes en el sistema. En la parte final de este capítulo se describe la arquitectura y el uso de una unidad de interfaz múltiple integrada en la CTF, en la cual se conectan los periféricos esenciales del sistema.

En los siguientes capítulos se describe detalladamente el diseño modular de las tarjetas objetivo así como la interacción y protocolos de comunicación entre ellas.

2.2 Unidades de Procesamiento (UPs)

En la Figura 2.1 se muestra un diagrama de bloques de la arquitectura de la CTF, la cual utiliza tres unidades de procesamiento operando en forma concurrente. La razón de incluir tres unidades de procesamiento radica en contar con un número mínimo suficiente de procesadores, entre los cuales se realizan comparaciones dinámicas de datos para detectar por medio de votación mayoritaria algún tipo de falla generada en alguna unidad de procesamiento. En nuestro caso, como se verá en los siguientes capítulos, se toman los datos generados en cada UP (asociados con transferencias a memoria y accesos a puertos) y se comparan estos con aquellos obtenidos de las UPs restantes en busca de posibles diferencias, las que de existir, indicarían fallas de operación en alguna de las unidades. Con la electrónica que se discutirá en el capítulo IV es posible detectar con exactitud la tarjeta que originó la falla así como el autodiagnóstico del mismo circuito electrónico. Si bien se compara exclusivamente la información proveniente del ducto (*bus*) de datos, ello redunda en la detección indirecta de fallas en memoria, líneas de direcciones y en señales de control; pues, ante la falla de alguna de ellas se originan accesos incorrectos que se manifiestan en diferencias entre los datos durante el proceso de comparación. Es por esto que la detección y el diagnóstico de fallas de la arquitectura propuesta, se extiende a todos y cada uno de los componentes integrados en las UPs.

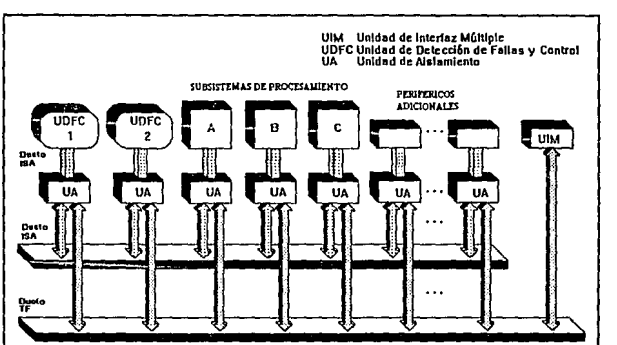


Figura 2.1 Diagrama de bloques de la arquitectura TF

Cada UP contiene un microprocesador 80386SX (32 bits de datos internos y 16 bits externos), coprocesador matemático (opcional), EPROM BIOS, manejador de teclado, bases de memoria que pueden contener desde 1 MB hasta 16 MB de memoria DRAM y un juego de circuitos integrados VLSI que auxilian al procesador para interactuar con la memoria dinámica, al multiplexar los ductos de datos internos y para aceptar/generar las señales del ducto estándar conocido como ISA (*Industry Standard Architecture*).

2.3 Unidades de Detección de Fallas y Control (UDFCs)

La arquitectura concurrente y tolerante a fallas se supervisa en tiempo real con el objeto de detectar posibles fallas de operación en cualquiera de sus módulos. De la correcta detección de anomalías depende el diagnóstico de la falla, la reconfiguración de la arquitectura y, por consiguiente, la esperada operación ininterrumpida. Para este propósito, en el desarrollo de la presente tesis se diseño una tarjeta que persigue como meta la detección de fallas y el control del sistema TF. Como se verá en el Capítulo IV, la tarjeta contiene dos autómatas electrónicos que operan a 16MHz, los cuales se encargan de adquirir los datos provenientes de los diferentes procesadores, de realizar la comparación de información y, dependiendo del diagnóstico de los resultados, permitir continuar sin modificaciones, o bien, realizar los cambios requeridos en la arquitectura para sostener su operación continua y por consiguiente permitir la ejecución de programas de aplicación sin interrupción alguna. Como se verá, esta tarjeta contiene además un microcontrolador (μC) y una serie de registros de control que facilitan la inicialización de la arquitectura así como las tareas de mantenimiento en línea.

2.4 Unidades de Aislamiento

Desde el inicio del proyecto se estableció la necesidad de incorporar al diseño el atributo de mantenimiento en línea, el cual está relacionado con la extracción e inserción de las diferentes tarjetas que componen al equipo, sin tener que desenergizarlo. Para permitir estas acciones, se debieron resolver dos problemas, primero, el control del suministro de energía hacia las diferentes tarjetas objetivo del

sistema y, segundo, asegurar que durante la inserción o extracción de tarjetas no existan cortos circuitos ni interferencias con los demás módulos del equipo.

Por otro lado, la operación concurrente de las tres tarjetas UPs de la computadora necesita forzosamente de algún medio que impida la colisión de señales provenientes de los tres subsistemas de procesamiento. Es decir, para que cada UP pueda leer la misma información, procesar los mismos datos y generar las mismas salidas, se necesita de un módulo intermedio que permita controlar dinámicamente el flujo de datos que va desde las UPs hacia los periféricos de la computadora, de tal forma que se eviten conflictos en los ductos de la computadora. De la misma manera, dicho módulo debe permitir el paso de información proveniente de cualquier periférico hacia los procesadores, para que estos reciban los mismos datos y puedan generar los mismos resultados.

En cuanto a las unidades de detección de fallas, también se necesita otro módulo intermedio que permita en primera instancia el mantenimiento en línea y que además permita el aislamiento de alguna tarjeta con falla, para proceder a su sustitución por una tarjeta en buen estado.

Para resolver todos estos problemas se construyeron dos tipos de tarjetas de aislamiento, las cuales se desarrollaron previamente como un trabajo de tesis [Rivera 1993]. Razón por la cual en el presente trabajo se le cita como infraestructura de apoyo. También se hace notar, que el autor del presente trabajo generó contribuciones para el desarrollo de las tarjetas mencionadas.

2.5 Unidad de Interfaz Múltiple

Para facilitar el manejo del teclado y la bocina compartidos por las UPs, así como para agilizar el mantenimiento de una pantalla de despliegue alfanumérico y de un teclado de membrana utilizados para controlar la arquitectura TF durante labores de mantenimiento correctivo o preventivo, se decidió instalar todos los conectores de periféricos en una tarjeta, permitiendo esto el aislamiento de sus señales y la modularidad de las tarjetas objetivo. Esta tarjeta contiene un número mínimo de componentes electrónicos (4), por lo cual, se le considera un apéndice de la

arquitectura y de muy baja probabilidad de falla. La tarjeta se describe a detalle en el Capítulo IV.

2.6 Ductos utilizados

El comportamiento externo de la CTF con todas sus tarjetas integradas, es similar al de una computadora personal convencional, pero con atributos de autodiagnóstico y de autoreconfiguración en caso de fallas. Debido a esto la CTF es compatible en circuitería y en programación con la amplia disponibilidad de periféricos y de paquetes de programación que hay en el mercado. Para lograr estas dos importantes características (compatibilidad y tolerancia a fallas) la CTF utiliza dos ductos, los cuales conforman la columna vertebral del sistema. Uno de los ductos es el PC-AT y el otro es un ducto propietario denominado TF.

2.6.1 Ducto PC-AT

Este ducto es completamente compatible con el estándar ISA, el cual contiene 24 líneas de direcciones, 16 de datos, líneas de control y protocolo, ver Figura 2.2. Para mayores detalles sobre el estándar ISA ver [Solari 1991]

2.6.2 Ducto TF

Este ducto se diseñó para propósitos de detección, diagnóstico y reconfiguración de la arquitectura, ver Figura 2.3. El ducto propietario TF fue otro factor que hizo posible la partición de la arquitectura total en módulos, lo cual facilitó la definición de zonas de contención de fallas y redujo la complejidad del diseño total del sistema.

Entre las señales más importantes del ducto TF tenemos 16 líneas para el multiplexaje de datos desde UPs hacia UDFCs, líneas de estado y categoría para UPs, señales de protocolo entre autómatas, señales de teclado y bocina, así como líneas de datos y control para el despliegue alfanumérico y teclado de membrana utilizados en tareas de mantenimiento. También se encuentran en este ducto las líneas de protocolo y control entre UDFCs, que permiten la sustitución automática de una UDFC anómala por su

contraparte redundante en buen estado y, finalmente, líneas de control entre UAs y UDFCs.

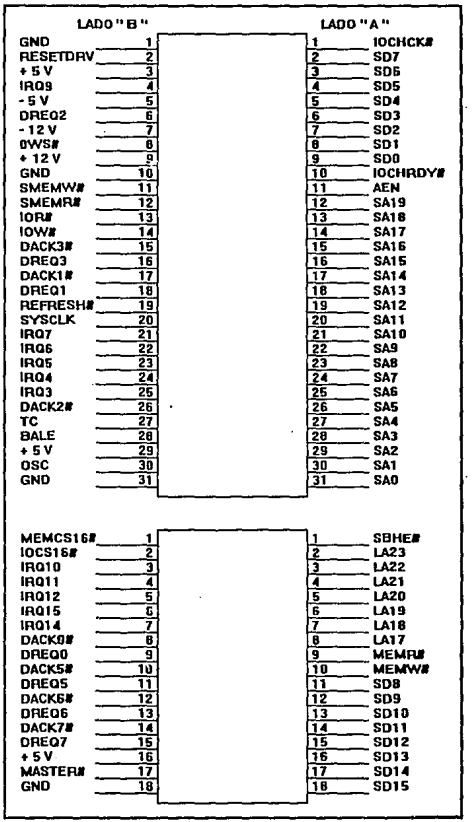


Figura 2.2 Terminales del Ducto PC-AT

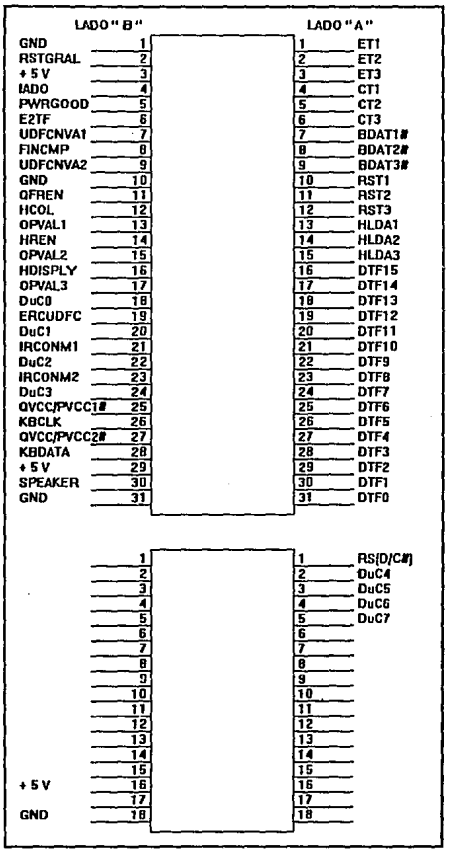


Figura 2.3 Terminales del Ducto TF

Como se mencionó anteriormente existen TAs dedicadas para UPs y para UDFCs. Las TAs se conectan directamente al ducto TF, pero no todas las señales contenidas en él llegan a las UPs o a las UDFCs. Dentro de las TAs se generan diferentes señales para UPs y para UDFCs, de ello se desprende la existencia de dos ductos intermedios que denominamos: ducto TA-UPs y ducto TA-UDFCs. Estos ductos conducen líneas de datos y control específicas para cada tarjeta objetivo. En las Figuras 2.4 y 2.5 se muestra la disposición de las terminales del ducto TA-UPs y el ducto TA-UDFCs, respectivamente.

A continuación se da una breve descripción de las señales que integran el ducto propietario:
IADO Inicio del Autómata de Detección de Operaciones. Inicializa al autómata que se encarga de decodificar señales que indican lectura o escritura a puerto o memoria.

PWRGOOD Power Good. Señal proporcionada por la fuente de alimentación que llega a la unidad manejadora del ducto AT (ATU, TACT83443).

E2TF Existen dos Tarjetas con Falla. Línea necesaria para proporcionar dicha información al Autómata de Detección de Operaciones.

FINCMP Fin de Comparación. Indica el final del proceso de comparación de los datos multiplexados.

QFREN Quita Freno. Señal que libera a los procesadores (cuando se encuentran detenidos) después de efectuarse la comparación de datos en la UDFC.

OPVALi Operación Válida de la UPi. Indica que se efectuó una lectura o escritura en la iésima UP.

KBCLK Keyboard Clock. Señal de reloj del teclado.

KBDATA Keyboard Data. Señal de datos del teclado.

SPEAKER Bocina

ETi Estado de la Tarjeta i. Señala el estado (encendido o apagado) de la tarjeta correspondiente.

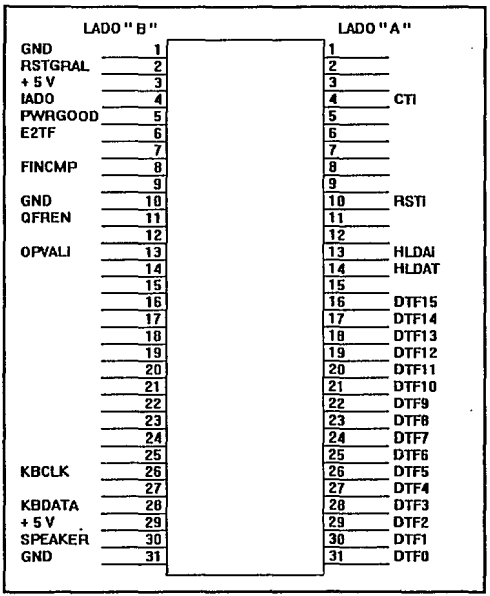


Figura 2.4 Terminales del Ducto TA-UPs

CTi Categoría de la Tarjeta i. Indica la categoría (principal o redundante) de la UP correspondiente.

BDATI Habilitador de Datos Multiplexados. Habilita las salidas del *buffer* i de la UP i para multiplexar los datos.

RSTi Reset i. Es la línea de RESET para la UP i.

HLDAI Hold Acknowledge i. Línea de reconocimiento de DMA en la UP i.

DTF0-15 Ducto de datos multiplexados por comparar.

UDFCNVA1,2 UDFC Nueva 1 y 2. Señal que indica a la UDFC redundante que la principal falló.

HCOL Habilita Columna. Señal que habilita la columna del teclado de la UIM.

HREN Habilita Renglón. Señal que habilita el renglón del teclado de la UIM.

HDISPLY Habilita Display. Señal para habilitar la pantalla de cristal líquido de la UIM.

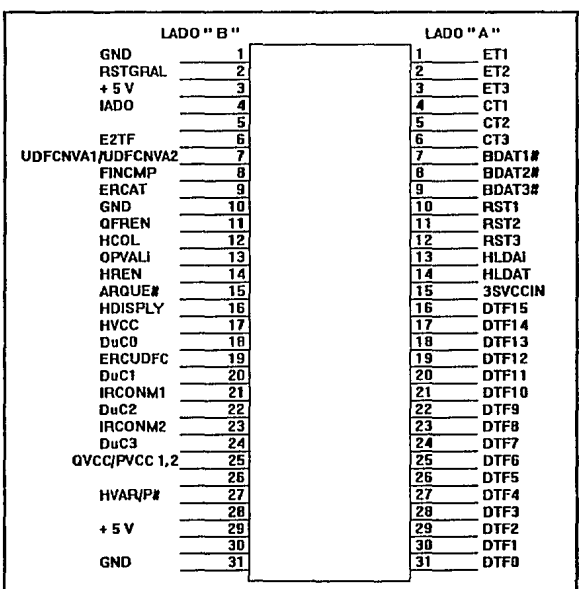


Figura 2.5 Terminales del Ducto TA-UDFCs

DuCO-7 Ducto de datos de la UIM.

ERCUDFC Error de Categoría en UDFC. Indica un error en la asignación de categoría de las dos UDFCs instaladas cuando arranca el sistema.

IRCONM1,2 Interrupción de Conmutación. Cuando falla la UDFC principal, esta señal indica a la UDFC redundante que debe iniciar operaciones.

QVCC/PVCC1,2# Quita Vcc/ Pon Vcc 1 y 2. Señal que se encarga de conmutar los voltajes de alimentación de las UDFCs, en caso de que se requiera mantenimiento en línea.

3SVCCIN Tres Segundos de Vcc In. Señal que permite el suministro de energía a las UDFCs durante tres segundos para que se autoconfiguren al arrancar el sistema.

ERCAT Error de Categoría en UPs. Línea que se activa en caso de detectarse dos o más UPs declaradas como principales.

ARQUE Arranque. Señal que permite detectar en las UDFCs un arranque en frío o en tibio.

HVCC Habilita Vcc. Esta señal habilita el voltaje de alimentación de las UDFCs.

RS(D/C#) Habilitador que indica transferencia de datos o comando de control para la pantalla de cristal líquido.

HUAR/P# Habilitador de unidades de aislamiento que define su operación como redundante o como principal.

TRES

Diseño de Unidades de Procesamiento

3.1 Introducción

En esta sección se describe la arquitectura diseñada de las tarjetas de procesamiento para la CTF. Se detalla la electrónica utilizada para permitir el proceso de sincronización de los tres procesadores de la CTF, la cual realiza también el frenado de los procesadores cuando así se requiere para que las UPs puedan procesar información concurrente. Durante este proceso se utilizan estados de espera exclusivamente cuando alguna UP finaliza sus procesos antes que las restantes.

Se describe la forma en que la arquitectura desarrollada puede compartir el mismo teclado junto con las UPs restantes. Además, se mencionan las dos formas de utilización de la tarjeta diseñada: como uniprocesadores sin redundancias y como UP para propósitos de tolerancia a fallas. Finalmente se describe el proceso de desarrollo del circuito impreso.

En el siguiente capítulo se detalla el diseño de la tarjeta UDFC, la cual permite la supervisión, el diagnóstico y la reconfiguración de la arquitectura en tiempo real; como se verá, ésta tiene capacidad de autodiagnóstico en las partes más críticas de su arquitectura.

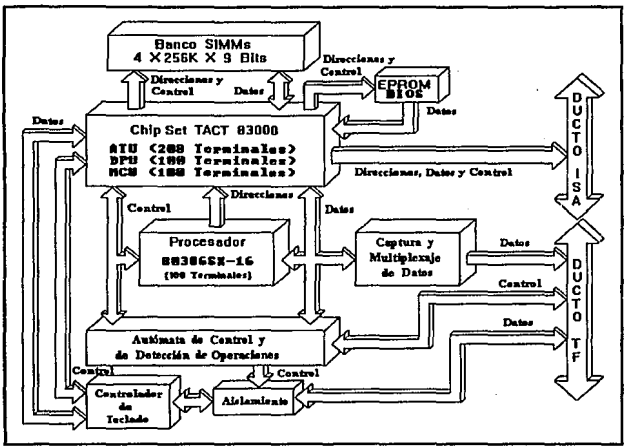


Figura 3.1 Diagrama de bloques de las Unidades de Procesamiento (UPs)

3.2 Arquitectura de UPs

Cada UP está constituida por:

- Un procesador 80386SX-16
- Un circuito integrado (CI) VLSI denominado ATU que genera señales compatibles con el estándar ISA.
- Un circuito VLSI denominado DPU encargado de controlar los diferentes ductos de datos.
- Un CI VLSI denominado MCU que interacciona con la memoria dinámica.
- Cuatro bases que forman un banco de memoria dinámica tipo SIMM con las cuales se pueden formar bancos de 1 a 16 Mbytes.

- EPROM BIOS de 64Kb, expandible a 128 ó 256Kb, para aplicaciones de tipo uniprocesador autosostenido.

- Un autómata electrónico para detectar datos válidos por comparar.

- Electrónica adicional para que el sistema triplex comparta el mismo teclado.

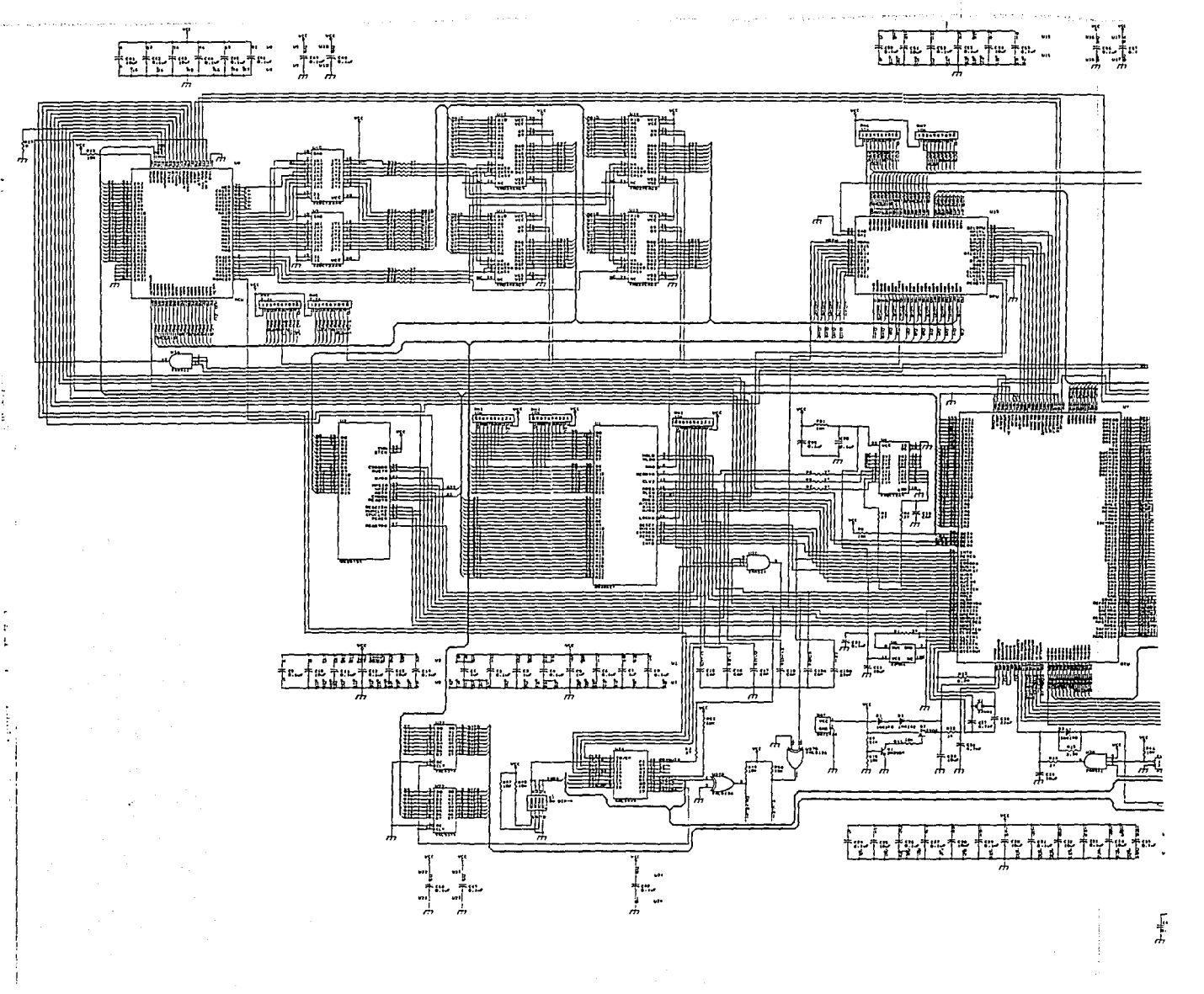
En la Figura 3.1 se observa el diagrama de bloques de UPs, en donde además de los módulos citados anteriormente resaltan los ductos utilizados por la CTF. Uno es el ducto ISA, el cual le confiere al sistema completa compatibilidad con periféricos y paquetes elaborados para PCs comerciales, y el otro, el ducto Tolerante a Fallas por medio del cual, entre otras cosas, se le enviará una señal a la UDFC cuando se haya detectado una operación válida, implicando ello que el dato generado se ha capturado y está listo para ser transmitido y proceder a la detección de posibles fallas. El ducto TF también se utiliza para enviar y recibir otras señales de protocolo entre UPs y UDFC principal.

En la Figura 3.2 se presenta el diagrama electrónico de la tarjeta desarrollada, en donde merecen especial atención los interruptores agrupados como S1 y los CIs U27, U25, U24, U23, U22. En particular, el interruptor S1 permite definir el modo de operación: simple (uniprocesador sin redundancias) o modo concurrente (esquema triplex para detección de fallas). El CI U27 (74LS136) se usa para evitar cortos circuitos en caso de que se realice una configuración incorrecta de los interruptores en la UA respectiva. Básicamente se protege a los CIs VLSI, al GAL (U24) y a las señales HLDA y OPVALi.

El CI U25 en combinación con el controlador de teclado y su lógica asociada, permiten que el sistema triplex comparta el mismo teclado y la misma bocina, o bien, que en modo simple se utilice un teclado (el cual deberá insertarse en el conector CNKB) y una bocina a través del conector CN7.

Adicionalmente, en la Figura 3.2 destaca la presencia del microprocesador etiquetado como U1, la Unidad de Interfaz con el ducto AT (U7), la Unidad de Control de Memoria (U8), las bases para alojar el banco de SIMMs (U11, U12, U13 y U14), el coprocesador matemático (U2) y el EPROM BIOS (U16).

Figura 3.2 Diagrama Electrónico de UPs



Los diferentes cristales de los relojes del sistema se indican en la Figura 3.2 con las etiquetas U6 (reloj del sistema de 32Mhz), Y1 (reloj de tiempo real) y Y2 (reloj del ducto ISA). En la misma figura se indican los conectores que permiten el tráfico de señales entre UAs y UPs, por un lado el ducto AT representado por los conectores CN10, CN11, CN12 y CN13 y por otro, el ducto TF indicado con las etiquetas CN14 y CN15. Debe recordarse que las tarjetas objetivo del sistema TF, es decir, tanto las UPs como UDFCs, se conectan a los ductos del sistema por medio de unidades de aislamiento, como se mostró en la Figura 1.1.

En vista de que la CTF basa sus actividades en un sistema triplex, es necesario contar con indicadores que establezcan la categoría (principal o redundante) y el estado (encendido o apagado) adjudicados a cada UP. En el diseño realizado se utiliza la señal CTi y la fuente de alimentación de 5 volts para identificar la categoría y el estado, respectivamente, para cada UP en cuestión. Debe señalarse que el sistema TF cuenta en realidad con tres bits alojados en el ducto TF para indicar la categoría asignada a cada UP, además de tener otros tres bits, también incluidos en el ducto TF, para indicar el estado de las mismas. En la tarjeta aquí descrita se hizo mención únicamente a la línea CTi y a la fuente de alimentación para definir la categoría y el estado, respectivamente, debido a que las tarjetas de aislamiento para UPs contienen interruptores que permiten elegir alguna de las tres líneas de categoría. Cuando en las tarjetas de aislamiento se detecta que el bit de estado respectivo indica la acción de apagado, se procede a interrumpir el suministro de alimentación a la UP, por lo cual en la UP la señal de estado encendido se toma directamente del voltaje de alimentación.

La operación concurrente del sistema triplex en combinación con las unidades de aislamiento permiten que cada UP adquiera y procese la misma información, y que por tanto todas generen los mismo resultados. Como ya se mencionó en la sección II.4, las unidades de aislamiento efectúan el control dinámico del flujo de datos en el ducto AT, de tal forma que cuando las UPs envían datos hacia el ducto se eviten posibles colisiones; en este caso, las señales que alcanzan el ducto AT provienen exclusivamente de la UP etiquetada como principal. Con el objeto de conocer dinámicamente los cambios de categoría producidos o inducidos en las UPs se colocaron diodos luminosos, que en la Figura 3.2 aparecen con las etiquetas D6 y D7, los cuales indican respectivamente el estado de cada UP en color verde y la categoría en color rojo, durante la operación del equipo. En el circuito impreso diseñado los LEDs anteriores no aparecen físicamente, en su lugar se colocaron conectores, de tal

forma que los LEDs se puedan instalar en la estructura del gabinete, unidos mediante cables.

3.3 Técnica de Sincronización para Trabajo Concurrente

La arquitectura triplex de la CTF basa su operación concurrente en la posibilidad de frenar a los procesadores cuando así se requiere. Una vez que se ha instalado el sistema operativo en cada uno de los procesadores, da inicio la detección de fallas en tiempo real; a partir de ese momento un autómata electrónico detecta aquellas operaciones (realizadas por el procesador) que interactúan con memoria o con puertos. Una vez detectada la operación, también se encarga de capturar el dato generado para que sea transferido a la UDFC, cuando ésta lo solicite. Estas operaciones se ejecutan concurrentemente en cada una de las UPs lo que implica que los datos capturados y por tanto el proceso realizado en cada UP se lleva a cabo en tiempos muy cercanos, pero no necesariamente iguales debido a las diferencias de comportamiento eléctrico de los CIs. Debido a esto y considerando que la UDFC requiere de cierto tiempo para efectuar la lectura multiplexada de los datos capturados en cada UP, se necesita frenar momentáneamente a todas las UPs instaladas, debiendo permanecer congeladas en tiempos muy cortos y sin perder su estado de operación. Durante este tiempo, la UDFC lee los datos de cada UP, uno a uno, los guarda en registros de propósito especial y posteriormente compara los datos entre sí, como se verá con mayor detalle en el siguiente capítulo.

Una de las formas de frenar la operación del 80386, sin que éste pierda su estado, y que posteriormente pueda continuar con sus tareas encomendadas, consiste en controlar la línea READY#, la cual se utiliza normalmente para que el procesador pueda interaccionar con periféricos lentos, como pueden ser los accesos a memoria de baja velocidad. En estos casos, el microprocesador sostiene sus líneas de direcciones y control hasta que ha transcurrido el tiempo necesario para que la memoria esté en condiciones de recibir o de entregar un dato; al terminar este proceso se libera la señal READY#, para indicar al procesador que el periférico está en condiciones de continuar y que por tanto se libera al procesador. En la Figura 3.3 se muestran dos ciclos de escritura y uno de lectura, cada uno consiste de dos estados llamados T1 y T2. Cuando los periféricos son suficientemente rápidos, cualquier localidad de memoria o puerto puede accesarse en los dos ciclos de *bus* indicados. Cada acceso a memoria o

puerto continua hasta que se detecta el reconocimiento del periférico por medio de la entrada READY# del 80386SX. Al hacer este reconocimiento, al final del primer ciclo del estado T2, se obtiene el ciclo de acceso a periférico más corto posible, el cual requiere únicamente a los estados T1 y T2 indicados en la figura.

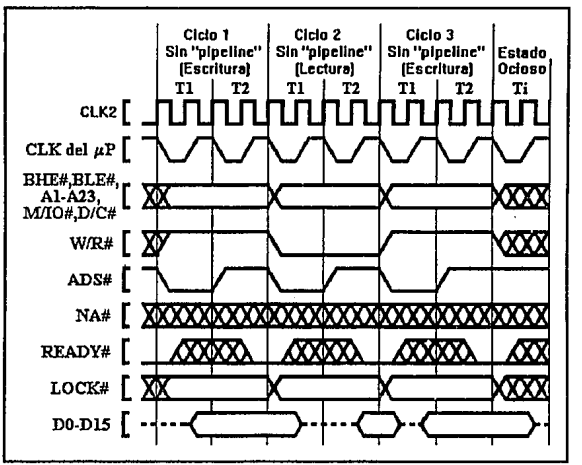


Figura 3.3 Diagrama de tiempos para lecturas/escrituras en el 386 sin estados de espera

En el caso en que la señal READY# no se genere al final del primer ciclo del estado T2, se genera una repetición indefinida del estado T2 hasta que se active la señal READY#, ver Figura 3.4.

Desde el inicio del proyecto se realizaron pruebas exhaustivas para comprobar la posibilidad de frenar repetitivamente y por tiempos variables la operación del 80386SX. En la Figura 3.5 se muestra la disposición del equipo utilizado para realizar tales pruebas. Se utilizó una tarjeta madre (TM) que contiene al procesador 80386SX y a los CIs VLSI TACT83000; se instalaron conexiones de prueba en la tarjeta indicada

para tener acceso a las señales: READY#, HLDA, TC y CLK2, se diseño y construyó en *wire-wrap* un autómata electrónico de prueba, el cual aceptaba como entrada, la salida digital de un generador de señales, al cual, durante las pruebas realizadas, se le varió su frecuencia desde 0.5 Hz hasta 10 KHz. El autómata al detectar un nivel alto proveniente del generador de señales procedía a desactivar la señal READY# de la TM; durante los niveles bajos se permitía la operación normal del procesador. Como se comprenderá al variar la frecuencia en el generador de señales se producía el frenado repetitivo de las operaciones del 80386 en tiempos proporcionales a la frecuencia utilizada.

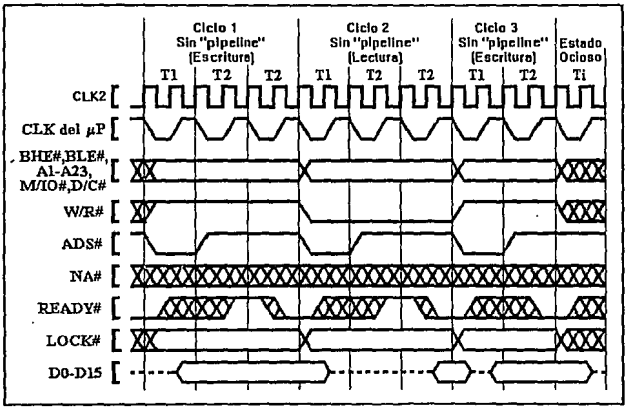


Figura 3.4 Diagrama de tiempos para lecturas/escrituras con estados de espera

El autómata previó la detección de procesos DMA, utilizados por el 80386 para realizar accesos ya sea a disco flexible o a disco duro. Durante tales eventos impedía que la señal del generador de señales produjera el frenado del procesador. El diseño se basa en la premisa de que los datos se almacenan y se recuperan correctamente desde disco. Por lo cual esta forma de tratar el DMA no afecta el comportamiento tolerante a

fallas de la CTF, debido a que el esquema TF asegura la veracidad de los datos en RAM.

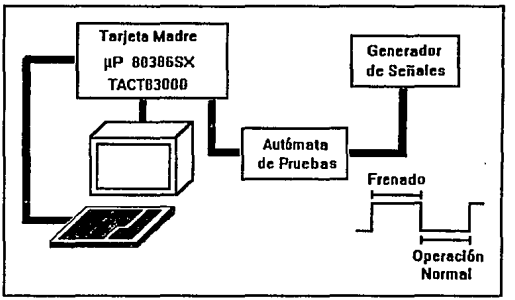


Figura 3.5 Diagrama de conexiones para pruebas de frenado

De las pruebas realizadas concluimos la factibilidad de realizar frenados repetitivos sin producir la pérdida del estado ni la continuidad de los programas ejecutados en la TM. Para las pruebas realizadas se elaboró un programa que lee repetitivamente de disco una imagen desplegándola en el monitor de la computadora.

3.4 Autómata de Detección de Operaciones

Como se mencionó en la sección anterior cada UP diseñada contiene electrónica dedicada para detectar aquellas operaciones realizadas por el 386 que involucran accesos a memoria o a puertos. En nuestro caso se desarrolló un autómata electrónico, mostrado en la Figura 3.6, encargado de realizar la tarea mencionada. Para iniciar la detección de operaciones el autómata espera que se active la señal IADO (Inicia el Autómata de Detección de Operaciones), la cual es generada por la UDFC principal del sistema. Posteriormente, el autómata pregunta por el inicio de una operación DMA, en cuyo caso se acude y se espera en el estado D hasta que dicho proceso haya culminado. Adicionalmente, en el estado B el autómata permite frenar a la UP respectiva siempre y cuando se decodifique una operación válida y que además

no existan dos tarjetas desactivadas en el sistema. De cumplirse esta condición se envía la señal RDYOUT# para iniciar el protocolo de generación de la señal READY# y en consecuencia activar la generación de estados de espera de la UP respectiva. En el estado E se activa la señal ALDATi (Activa *Latch* de Datos i) para retener el dato por comparar en un *latch* 74LS374, y en este estado se permanece hasta que se activa la señal FINCMP (Fin de Comparaciones) generada por los autómatas contenidos en la UDFC principal. La generación de la señal anterior indica que la UDFC ha recabado los datos generados en cada UP, los ha comparado entre sí y ha tomado alguna de las dos decisiones siguientes:

a) Generar la reconfiguración necesaria para resolver algún tipo de falla encontrada en el sistema TF. En este caso, la UDFC procede a aislar alguna tarjeta defectuosa (a través de la actualización del registro de estado) y posteriormente libera al (los) procesador(es) restante(s).

b) Terminar el proceso de comparación al no detectar anomalías en la comparación de datos, liberando en este caso a los procesadores instalados.

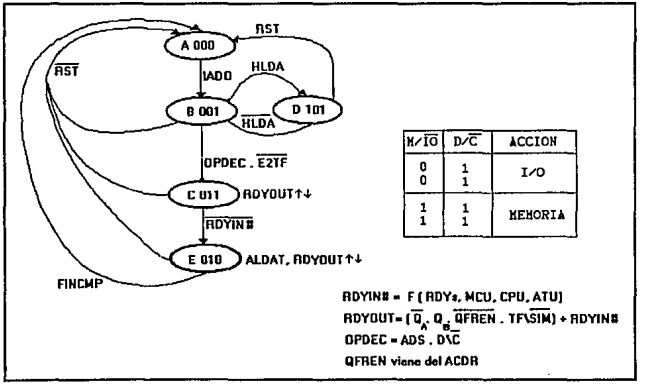


Figura 3.6 Autómata de Detección de Operaciones para UPs

Para lograr la reducción del número de componentes electrónicos utilizados por los autómatas, se recurrió al uso de GALs (*Generic Array Logic*), consiguiendo mejoras notables. Los CIs U22, U23 y U24, de la Figura 3.2, conforman el autómata de detección de operaciones; como se observa, el uso de GALs simplifica en buen grado el número de componentes del autómata, los que de haberse realizado con componentes tradicionales hubieran requerido de *flip-flops*, lógica para el decodificador de entradas y lógica para la generación de las salidas. Con el objeto de reducir los tiempos de respuesta del autómata se utiliza como señal de disparo el reloj de 16 MHz empleado por el 80386. En el Apéndice C se incluyen los datos técnicos de los GALs utilizados.

En la Tabla 3.1 se dan las ecuaciones de estado para programar el CI U24.

Para programar los GALs se utilizó el paquete PLAN que permite generar un archivo de datos por programar a partir de las ecuaciones derivadas de algún problema en particular. Deben de enfatizarse dos formas para la declaración de ecuaciones en el paquete, una utilizada para la programación de lógica combinacional y otra para programar circuitos secuenciales. La diferencia estriba básicamente en la sintaxis para declarar las ecuaciones, por ejemplo para programar la función combinacional

$$A = X + Y + Z,$$

la función se declara sin cambio alguno, en tanto que para declarar la ecuación de estado de un circuito secuencial, por ejemplo

$$Q_A = Var_1 + D_B + D_C$$

la ecuación debe expresarse como:

$$Q_A := Var_1 + Q_B + Q_C$$

lo cual corresponde a las sintaxis requerida por el paquete. Durante la edición de las ecuaciones, se deben declarar las variables del problema asociadas con cada una de las líneas del GAL utilizado. Esta operación constituye la declaración de variables asociadas a las terminales del GAL. A manera de ejemplo, en la Figura 3.7 se muestra el listado del programa para el Autómata de Detección de Operaciones.

$$Q_A = \overline{Q}_B \cdot Q_C \cdot RSTGRAL \cdot HLDAT$$

$$Q_B = \overline{Q}_A \cdot \overline{Q}_B \cdot Q_C \cdot RSTGRAL \cdot \overline{HLDAT} \cdot E2TF \cdot \overline{ADS\#} \cdot D/C\#$$

$$Q_C = (\overline{Q}_B \cdot Q_C \cdot RSTGRAL) + (\overline{Q}_A \cdot Q_B \cdot Q_C \cdot RSTGRAL \cdot RDYIN\#)$$
$$(\overline{Q}_A \cdot \overline{Q}_B \cdot \overline{Q}_C \cdot RSTGRAL \cdot IADO)$$

$$RDYOUT\# = (\overline{Q}_A \cdot Q_B \cdot \overline{QFREN} \cdot TF/S\#) + RDYIN\#$$

$$ALDAT = OFVALI = \overline{Q}_A \cdot Q_B \cdot \overline{Q}_C$$

Tabla 3.1 Ecuaciones de Estado y Salidas del Autómata de Detección de Operaciones

```
GAL16V8; Autómata de Detección de Operaciones
;Cambios de variables
;TF/S# = TFS,  ADS# = ADS, D/C# = DC
;RDYIN# = RDYIN, RDYOUT# = RDYOUT
;Inicio del Programa
CLK TFS DC ADS RDYIN HLDAT IADO E2TF FINCMP GND
RSTGRAL ALDAT QFREN OFVALI NC QC QB QA RDYOUT VCC
QA := /QB*QC*RSTGRAL*/HLDAT
QB := /QA*/QB*QC*RSTGRAL*/HLDAT*E2TF*/ADS*DC
QC := /QB*QC*RSTGRAL + /QA*QB*QC*RSTGRAL*RDYIN +
      /QA*/QB*/QC*RSTGRAL*IADO
RDYOUT = /QA*QB*/QFREN*TFS + RDYIN
ALDAT = /QA*QB*/QC
OFVALI = /QA*QB*/QC
;Fin del Programa
```

Figura 3.7 Listado del programa del Autómata de Detección de Operaciones

Debe señalarse que las ecuaciones se pueden escribir mediante el uso de cualquier editor, por ejemplo, Turbo C, Turbo Pascal, etc. Una vez generado el archivo que describe las ecuaciones requeridas, se invoca al programa PLAN y se le indican tanto el archivo de entrada (de ecuaciones) como el archivo de salida (formato JEDEC). Para la presente tesis se utilizó el GAL 16V8-12 que emulando el PAL16H8 cuenta con nueve entradas, seis líneas bidireccionales, dos salidas y una entrada que puede

emplearse como referencia de reloj o bien como una entrada adicional. Para programar los GALs se utilizó el equipo TUP400 de Tribal Systems, que permite programar memorias, microcontroladores, PALs y GALs de diversos tipos y de diferentes fabricantes.

3.5 Teclado Compartido en la CTF

Como ya se mencionó anteriormente, las UPs se diseñaron para ser usadas como uniprocesador (sin redundancias), o bien, para utilizarse en una arquitectura redundante y tolerante a fallas. En el diseño electrónico la adición de variantes de operación y las mejoras de versatilidad, conllevan compromisos, los que en este caso se relacionan con el uso del teclado y de la bocina. Esto es, en modo uniprocesador se utiliza la tarjeta UP junto con periféricos adicionales para conformar la arquitectura de una PC industrial convencional, en la que deben existir conectores directos para la bocina y el teclado; mientras que para la arquitectura tolerante a fallas se requiere que un máximo de tres UPs puedan compartir el mismo teclado y la misma bocina, lo cual es totalmente congruente con los objetivos de operación concurrente mencionados en la sección III.3. Para resolver este problema se decidió incluir lógica electrónica para permitir a cada UP las dos formas de operación explicadas. En vista del espacio disponible tan reducido en cada UP se dificultó la integración de un conector tipo *Phillips* para soportar el teclado en modo uniprocesador, por tal razón se sustituyó por un conector plano tipo *molex*. Para permitir la concurrencia de los procesadores es necesario que estos puedan leer al mismo tiempo los datos entregados por el usuario desde el teclado, para que los procesadores adquieran la misma información, y en consecuencia puedan efectuar el mismo proceso y generar las mismas salidas. Durante este proceso (lectura) no hay diferencias entre el modo uniprocesador y el modo TF. Sin embargo, la operación de un teclado no se restringe a la lectura de información relacionada con las teclas presionadas, sino que existe un protocolo de comunicación bidireccional entre los teclados y el controlador de teclado instalado en las UPs. Debido a esto, la operación puede entenderse como un puerto serie en el que circulan datos hacia las UPs cuando existen teclas oprimidas, pero también existe retroalimentación. Debido a estas condiciones el manejo del teclado en la CTF no es tan simple como parece. Para conferir la realimentación debe establecerse la comunicación con una sola tarjeta, pues de otra forma, de conectarse directamente las señales de las tres UPs, se producirían colisiones durante el proceso de

realimentación citado. Se instaló el CI U25 (interruptor analógico) que establece la ruta de retroalimentación únicamente para la UP declarada como principal, razonamiento muy parecido al utilizado para el control de las Unidades de Aislamiento mencionado en la sección II.4. Como puede observarse en la Figura 3.2 el conector del teclado (CNKB) y las líneas del teclado TF (KBCLK y KBDATA) están unidas a los mismos puntos; estas últimas líneas corren hasta el conector TF etiquetado como CN15, desde donde las diferentes UPs se conectan a las líneas del teclado compartido.

La solución para la bocina es similar a la del teclado, por lo cual sólo la UP principal será la encargada de enviar los pulsos eléctricos hacia la bocina. El control de la salida para esta señal lo realiza también el CI U25. En modo uniprocesador la conexión a la bocina es directa a través del conector CN7.

3.6 Elaboración del Circuito Impreso

Una vez generado el circuito esquemático de las UPs y hechas las revisiones pertinentes, se procedió a realizar el diseño del circuito impreso con el auxilio de un paquete de diseño asistido por computadora, el TANGO PCB PLUS. Este paquete permite el diseño de impresos constituidos de hasta 23 capas, incluyendo las capas de componentes, de soldadura, ocho capas intermedias, planos de alimentación y tierras, máscaras de antisoldado para el lado de componentes y de soldadura, capa de referencias de componentes y soldadura, capa de perforaciones, etc. Para definir trazos entre componentes permite editar líneas, arcos, círculos, polígonos para el trazo de planos, etc., todos ellos de dimensiones variables. Permite además la escritura de textos utilizados como referencias particulares, el establecimiento de bloques para copiado y rotación; además de facilitar la creación de diferentes tipos de bases para CIs ya sean convencionales o de montado superficial. Respecto a este último punto debe subrayarse el trabajo realizado para diseñar las bases de los CIs de montado superficial incluidos en las tarjetas UPs. No obstante que algunos de los CIs tienen el mismo número de contactos, al tener ellos diferentes dimensiones obligó el diseño particular de tres plantillas para montado superficial (CPU, DPU y ATU) y una especial para el coprocesador.

El paquete utilizado ofrece la posibilidad de enriquecer su base de datos a través de las nuevas plantillas generadas, por lo cual el trabajo desarrollado ha contribuido a incrementar la biblioteca de componentes del paquete.

Uno de los detalles más importantes del paquete TANGO es la generación de salidas denominadas *nido de ratas*, producidas con la información de los circuitos esquemáticos generados por el paquete OrCAD. Este tipo de salida permite visualizar a todos los componentes que forman el diseño junto con todas sus interconexiones declaradas; la importancia de esta opción radica en que el usuario puede modificar la ubicación de cualquiera de los componentes y en tiempo real (graficación animada) observar los cambios producidos en las interconexiones. Esta es una de las herramientas más importantes que ofrece este paquete, la que en el caso de la presente tesis se utilizó ampliamente para encontrar las posiciones preliminares de los componentes, las que en un 70% aproximadamente resultaron ser las posiciones definitivas, al 30% restante se le hicieron modificaciones de tipo rotación y traslado de componentes para que fuese posible realizar el trazo de las líneas.

Debe enfatizarse el hecho de que se escogieron aquellos arreglos que implicaran el menor número de cruces entre las líneas de interconexión. Otro punto importante que se debe resaltar es que el impreso de las UPs se elaboró en dos caras únicamente, lo cual tiene razones de índole económico y de fabricación; debido a que los costos para construir circuitos multicapa son elevados por un lado, y por otro, el que las compañías nacionales que dominan este tipo de tecnologías son muy pocas, repercutiendo en el costo de sus servicios. La decisión de realizar el impreso en dos capas tuvo consecuencias muy importantes en la complejidad del impreso diseñado, esto es, el contar con menos superficie para el trazo de líneas resultó en la saturación de líneas en la capa de componentes del impreso. En la capa de soldadura se trató de reducir el número de trazos con el objeto de dejar amplios espacios para planos de tierra, necesarios para disminuir problemas de ruido ocasionado por el tránsito de señales de alta frecuencia. Para este propósito, se instalaron también, suficientes capacitores de buena calidad y se dejo espacio para colocar filtros en líneas consideradas de alta prioridad en el diseño.

Otro de los atractivos del paquete TANGO radica en que permite generar salidas para *photoplotter*, pudiendo ser de tipo *gerber*, *postcript* y *epostcript*, que son formatos estándar; los dos últimos son formatos para impresoras láser especiales, las cuales ofrecen salidas en película, ya sea en positivo o en negativo. A este respecto, el

paquete utilizado permite construir impresos de hasta 32X32 pulgadas. De ello se desprende que el paquete demanda gran cantidad de memoria y de cálculo intensivo; dependiendo del tipo de aplicación el paquete puede usarse eficientemente en sistemas con RAM reducida (para diseños pequeños), o bien, en sistemas medianos y grandes. En nuestro caso en una máquina 386SX-33MHz con 4 Mbytes de RAM, sin coprocesador, la labor de diseño se llegó a dificultar debido a la alta densidad de gráficos implicados.

En la Figura 3.8 se observa la salida *nido de ratas* para las tarjetas de UPs. No obstante que el diseño involucra impresos de únicamente dos caras, para generar impresos de alta calidad es necesario diseñar al menos seis capas por circuito impreso, las cuales son:

- Capa de componentes: donde van montados los CIs.
- Capa de soldadura: donde se sueldan los componentes.
- Capa de referencias: donde se incluyen textos y gráficos que aparecerán pintados en el mismo impreso.
- Máscara de antisoldado del lado de componentes: sirve para impedir que la soldadura se propague en lugares indeseados e impide cortos circuitos.
- Máscara de antisoldado del lado de soldadura: es el mismo caso mencionado anteriormente.
- Capa de perforaciones: indica los lugares en que el impreso deberá ser perforado y por donde se insertarán los componentes para su soldado.

En las Figuras 3.9, 3.10, 3.11 y 3.12 se muestran las diversas capas diseñadas, mismas que integran el diseño total del impreso para UPs.

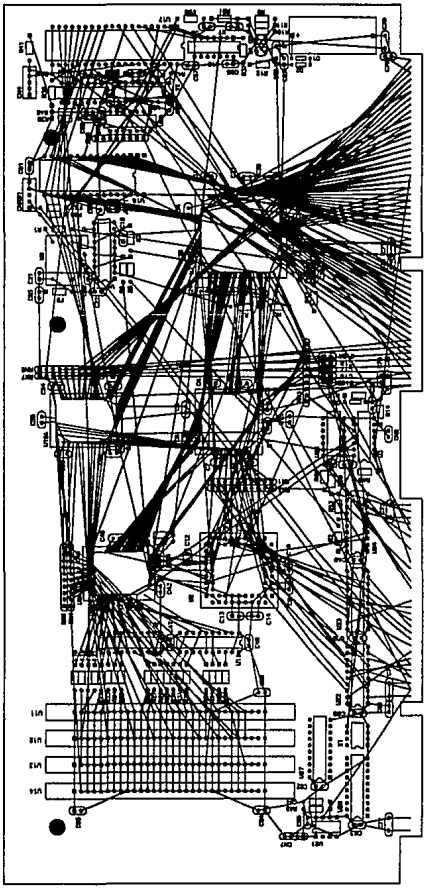


Figura 3.8 Salida *nido de ratas* de las UPs

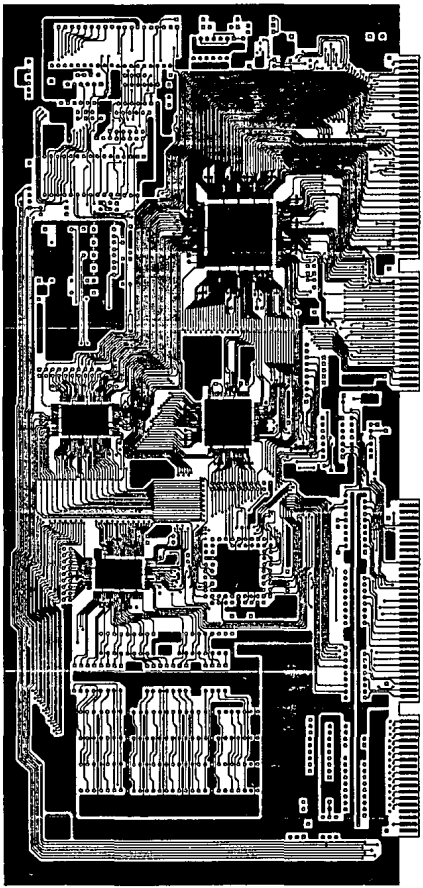


Figura 3.9 Capa de componentes de la UP

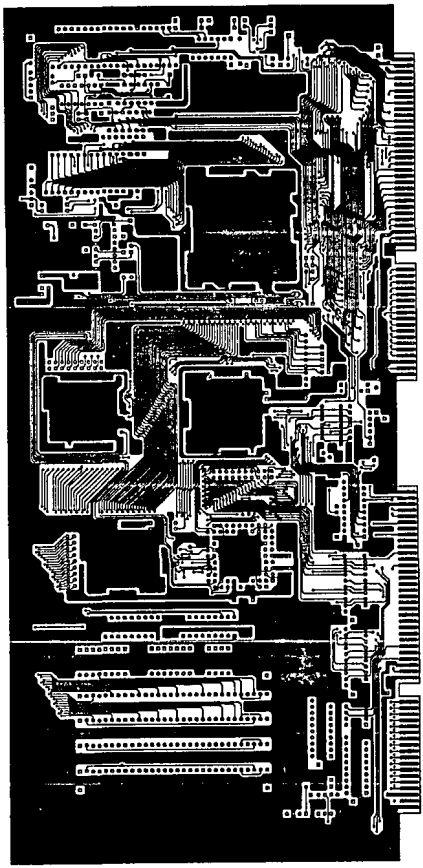


Figura 3.10 Capa de soldadura de la UP

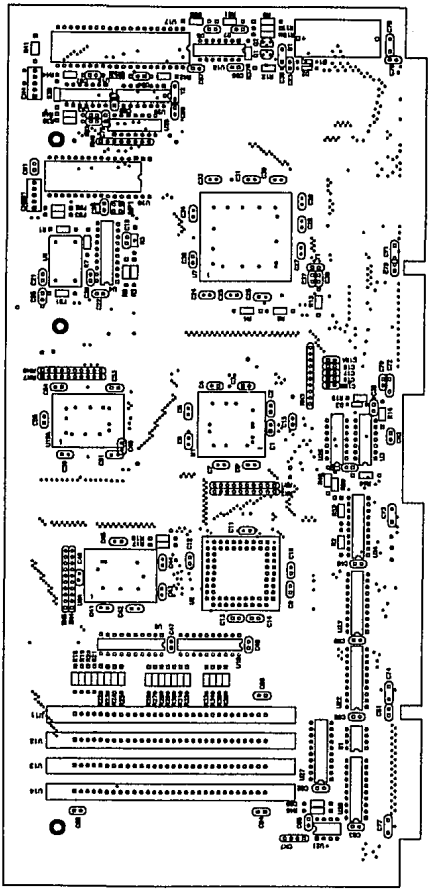


Figura 3.11 Capa de referencias de la UP

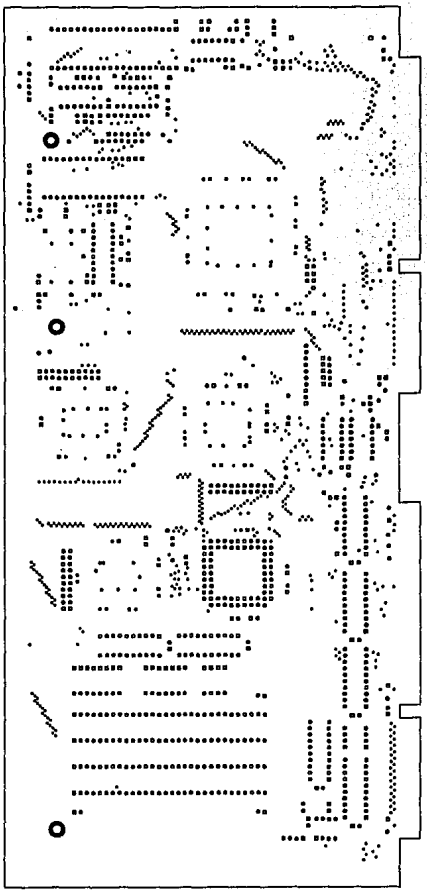


Figura 3.12 Capa de perforaciones de la UP

Diseño de la Unidad de Detección de Fallas y Control

4.1 Introducción

En las secciones anteriores se ha descrito la arquitectura y los modos especiales de trabajo diseñados para integrar una computadora tolerante a fallas. Como se explicó, la computadora utiliza redundancias y diversos medios para detectar y diagnosticar anomalías de operación; además de ejecutar procedimientos de reconfiguración para lograr el atributo de operación continua, el cual se obtiene incluso durante labores de mantenimiento, sea este correctivo, preventivo o para propósitos de demostración. También se hizo mención de algunas de las principales responsabilidades de la UDFC, por lo cual el lector ya estará familiarizado con algunos de los puntos a tratar.

En este capítulo se describe la arquitectura, modos de operación y los principios de programación requerida para integrar una tarjeta electrónica autodiagnosticable, con redundancia del 100% (duplex) que en caso de fallas conmuta automáticamente hacia su contraparte. De la correcta operación de esta unidad dependen las labores de detección y diagnóstico de fallas de la computadora, así como la reconfiguración del sistema que es indispensable para tolerar fallas en sus diferentes tarjetas electrónicas. Se describen también los protocolos de comunicación utilizados para sincronizar las diversas fases de operación de la CTF.

El capítulo incluye también el diseño de la tarjeta más pequeña utilizada por la CTF, la unidad de interfaz múltiple, que como se verá, contiene básicamente los conectores para los periféricos que permiten interactuar con el usuario. Finalmente se describe el desarrollo del circuito impreso de la UDFC.

4.2 Arquitectura de las UDFCs

Esta tarjeta está dividida en dos secciones, ver Figura 4.1, la primera compuesta por autómatas electrónicos que en tiempo real se encargan de realizar labores de detección, diagnóstico y reconfiguración (si es requerida ésta) de las diversas tarjetas que conforman la CTF; la segunda, compuesta por un microcontrolador y componentes adicionales, se encarga de interactuar con el usuario durante procedimientos de mantenimiento del sistema y para el envío de alarmas cuando se presentan fallas.

La sección de autómatas se encarga de multiplexar los datos provenientes de las UPs instaladas, así como de comparar los diversos datos para detectar fallas, las que de suceder, se diagnostican para proceder a efectuar las reconfiguraciones necesarias, obteniendo así la operación continua del sistema. Su electrónica está compuesta por dos autómatas que operan a 16 MHz, ambos vigilados por electrónica dedicada para detectar sus posibles fallas, las que de presentarse, obligan por medios electrónicos la conmutación y delegación de responsabilidad de control hacia la tarjeta UDFC redundante.

La sección de interacción con el usuario la integra un μC 68HC11 (modelo E1) junto con una memoria de programa (EPROM) y su *latch* para demultiplexar direcciones; adicionalmente cuenta con registros para leer variables de entrada y retener variables de salida, así como otros componentes que permiten interactuar con la Unidad de Interfaz Múltiple, con las UPs (para propósitos de inicialización y sincronización) y con los registros de estado que controlan al sistema TF. La comunicación entre el μC y los autómatas se efectúa mediante variables contenidas en los registros de entrada y de salida y también por medio de los registros de estado; debe enfatizarse que la prioridad más alta para accesar los registros de estado de la tarjeta, la tienen los autómatas. Por eso, para que el μC tenga acceso a tales registros debe solicitar permiso al autómata de reconfiguración, lo cual se gestiona mediante variables contenidas en los registros de entrada y de salida.

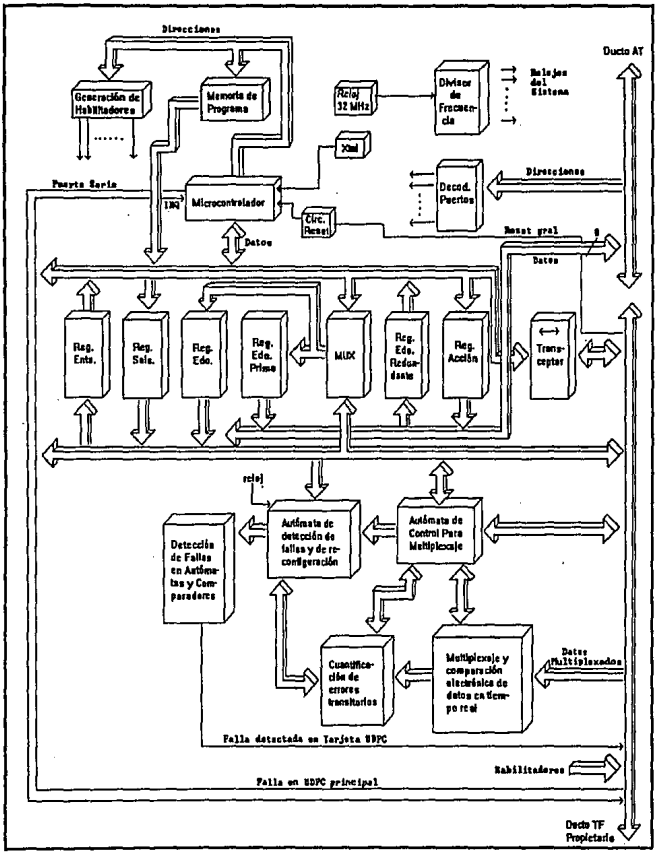
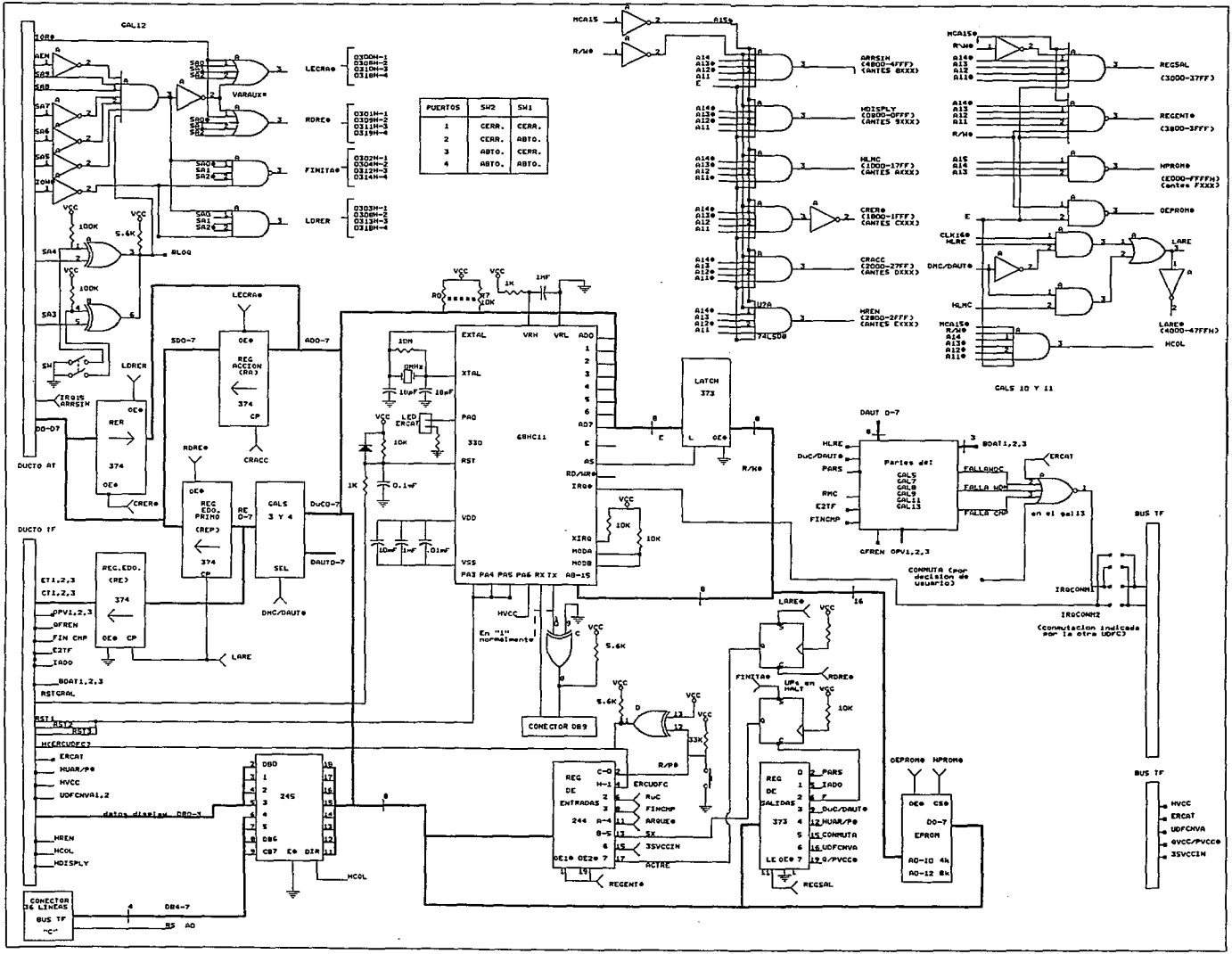


Figura 4.1 Diagrama de bloques de la UDFC

Figura 4.2 Diagrama de bloques de la electrónica que rodea al 68HC11


GAL12
LECRA*
RDRE*
FINITA*
LDRER
VARAUX*
SW1
SW2
SW3
SW4
PUERTOS
1 CERR. CERR.
2 CERR. ABTO.
3 ABTO. CERR.
4 ABTO. ABTO.
BLOQ
IRQ16 ARRSIN
DO-D7
DUCTO AT
DUCTO IF
REG ACCION (RA)
REG EDO. PRINC (REP)
REG.EDO. (RE)
RER
CRACC
CRERE*
ADO-7
SDO-7
DUCO-7
DAUTO-7
DMC/DAUT*
GAL5 3 Y 4
SEL
LARE
LECRA*
LDRER
RDRE*
68HC11
EXTAL
XTAL
PA0
RST
VDD
VSS
VRH
VRL
ADD
AD7
E
AS
RD/WR*
IRQ*
XIRQ
MODA
MODB
PA3 PA4 PA5 PA6 RX TX AB-15
LATCH 373
R/W*
LED ERCAT
HVCC
En "1" normalmente
CONECTOR DB9
R/P*
ERCUDFC
RUC
FINCMP
ARQUE*
SX
3SVCCIN
ACTRE
REGENT*
LARE*
FINITA*
RDRE*
UPA en HALT
REGSAL
OEPROM*
HPROM*
EPROM
D0-7
A0-10 4k
A0-12 8k
DAUT 0-7
HLRE
DUC/DAUT*
PARS
RMC
E2TF
FINCMP
Partes de:
GAL5
GAL7
GAL8
GAL9
GAL11
GAL13
FALLAHDC
FALLA HDM
FALLA CMP
en el gal13
GFREN OPV1,2,3
BDAT1,2,3
ERCAT
CONMUTA (por decision de usuario)
IRQCONM*
(conmutacion indicada por la otra UDFC)
BUS TF
HVCC
ERCAT
UDFCNVA
5VCC/PVCC*
3SVCCIN
MCA15*
R/W*
A15*
A14
A13
A12
A11
A10
ARRSIN (ANTES 8XXX)
HDISPLY (ANTES 9XXX)
HLMC (ANTES AXXX)
CREA* (1800-1FFF) (ANTES CXXX)
CRACC (2000-27FF) (ANTES DXXX)
HREN (2800-2FFF) (ANTES EXXX)
74LS08
U2A
REGSAL (3000-37FF)
REGENT* (3800-3FFF)
HPROM* (E000-FFFFH) (antes FXXX)
OEPROM*
LARE
LARE* (4000-47FFH)
HCOL
GAL5 10 Y 11
DBO
DB7
245
Eo
DIR
datos display DBO-7
HREN
HCOL
HDISPLY
HCERCUDFC?
ERCAT
HUAR/P*
HVCC
UDFCNVA1,2
RSTGRAL
RST1
RST2
CONECTOR 36 LINEAS
BUS TF "C"
DB4-7
R5 AO
ET1,2,3
CF1,2,3
OPV1,2,3
GFREN
FIN CMP
E2TF
IADO
BDAT1,2,3
REG EDO. (RE)
REG DE ENTRADAS 244
REG DE SALIDAS 373
PARS
IADO
DUC/DAUT*
HUAR/P*
CONMUTA
UDFCNVA
5VCC/PVCC*
3SVCCIN
LE OE* 7

En la Figura 4.2 se presenta el diagrama de bloques de la electrónica que rodea al 68HC11. En la parte superior izquierda se encuentra la decodificación de puertos utilizada por las UPs para leer y escribir registros de la UDFC; en la parte superior derecha están las funciones lógicas requeridas por el 68HC11 para generar los habilitadores de escritura y lectura de registros. En la parte inferior izquierda se tiene un CI 74LS245 con el cual el 68HC11 se interconecta al teclado y a la pantalla de cristal líquido (con los que el usuario coordina labores de mantenimiento) ubicados en la tarjeta UIM, es por ello que las líneas del 74LS245 se conducen hacia el ducto TF para establecer el contacto desde ahí, con la UIM.

A la izquierda del 68HC11 encontramos a los siguientes registros:

- **Registro de Acción (RA).** Aquí el μC deposita un dato de protocolo previamente al envío de una señal de interrupción a las UPs, el dato especifica el tipo de acción que los procesadores deberán realizar.

- **Registro de Estado (RE).** Este registro es actualizado ya sea por el Autómata de Comparación de Datos y Reconfiguración (ACDR) o por el μC. Sus bits ET1, ET2, ET3, CT1, CT2 y CT3 definen el estado y la categoría, respectivamente, de las UPs instaladas. Sus salidas se envían directamente al ducto TF y de ahí cada UP toma sus bits de control asignados.

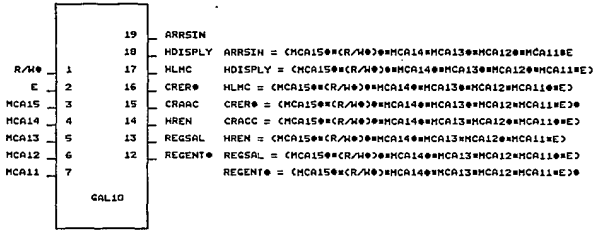
- **Registro de Estado Primo (REP).** Contiene la misma información que el registro de estado, sólo que sus salidas se leen exclusivamente por las UPs a través del ducto AT. Cuando existen cambios en el registro de estado se envía una interrupción a las UPs junto con la acción requerida y ellas, entre otras cosas, proceden a leer el cambio efectuado en el registro de estado primo. Posteriormente las UPs actualizan el dato leído en el registro de estado redundante de la UDFC de respaldo; este proceso se realiza para lograr que la UDFC de respaldo pueda iniciar operaciones (en caso de que la UDFC principal presente alguna falla) conservando el estado y la categoría de las UPs instaladas.

- **Registro de Estado Redundante (RER).** Como se mencionó en el punto anterior este registro lo actualizan las UPs siempre y cuando la UDFC que lo contiene sea redundante. Con el registro actualizado y en caso de requerirse una conmutación de UDFC, la tarjeta redundante conservará las asignaciones

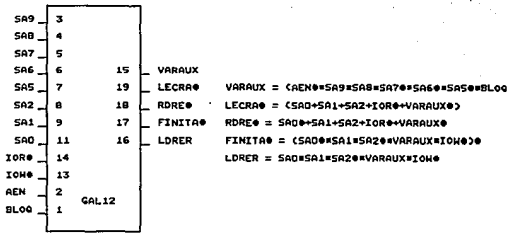
de categoría y estado de los procesadores debido a que una de las tareas iniciales del 68HC11 (durante la transición de UDFC redundante a principal) consiste en leer este registro y copiar su contenido en el RE y en el REP para entonces iniciar operaciones.

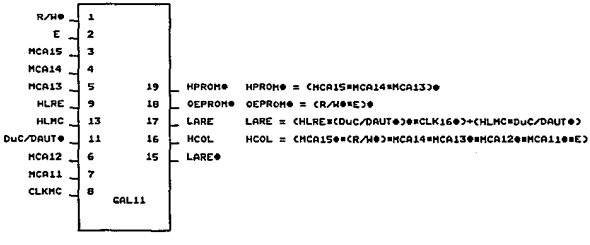
Figura 4.3 Banco de GALs del 68HC11

DEC. DE PUERTOS DEL uC



DEC. DE PUERTOS DE LAS UP*





En la Figura 4.3 se muestran los GALs utilizados por la UDFC para generar los diversos habilitadores y la lógica requerida en esta tarjeta, junto a ellos se indican las funciones lógicas programadas. Una vez más reiteramos la importancia del uso de GALs, pues de otra forma hubiera implicado incluir demasiados componentes electrónicos que impedirían la construcción de esta tarjeta. Los GALs 10 y 11 generan los habilitadores del EPROM externo, del teclado de membrana, de la pantalla de cristal líquido y los habilitadores de lectura y escritura de registros. El GAL 12 decodifica los puertos de comunicación con las UPs.

4.3 Autómata de Multiplexaje

El autómata está compuesto por un decodificador de entradas, *flip-flops* y un decodificador de salidas, los cuales se integraron en GALs para optimizar espacio, potencia y número de componentes electrónicos. La máquina secuencial se programó de acuerdo al procedimiento indicado en la sección III.5.

Como su nombre lo indica, el circuito electrónico se encarga de solicitar datos válidos provenientes de las UPs, uno a uno, de forma multiplexada. Para iniciar operaciones espera la indicación de disponibilidad de datos de cada uno de los autómatas de detección de operaciones contenidos en las UPs. Una vez satisfecha esta condición ordena la liberación del dato de alguna UP y procede a retenerlo en registros especiales de la UDFC. Esta operación se repite tres veces, independientemente del número de UPs instaladas. Cuando una UP no se encuentra en buen estado, o bien, cuando por razón alguna se desinstala una de ellas, la operación de este autómata no sufre alteración, lo cual no afecta los resultados del voteo; pues en este caso quien toma las decisiones de comparación y analiza los resultados es el ACDR, el cual prevé las diferentes combinaciones de procesadores instalados; este autómata se verá en la siguiente sección.

En la Figura 4.4 se presenta el autómata de multiplexaje de datos (AMD), se debe resaltar que las señales de solicitud (BDATi) y la señal de retención de datos en registros (LDCi) se generan al mismo tiempo para cada UP en particular. En la Figura 4.5 se muestran los diagramas de tiempos de estas señales, se observa que durante un ciclo de la máquina secuencial, se puede habilitar la lectura de un registro ubicado en alguna tarjeta UP y capturarlo en la tarjeta UDFC, sin que ello ocasione problemas

por tiempos de propagación en los ductos de la computadora. Como también se puede observar en el autómata, una vez finalizadas las transacciones con las UPs se activa el ACDR por medio de la señal IACOM. Posteriormente el AMD espera la señal FINCMP, la cual será generada por el ACDR después de haber analizado el voteo y realizado los cambios necesarios (en caso de ser requeridos).

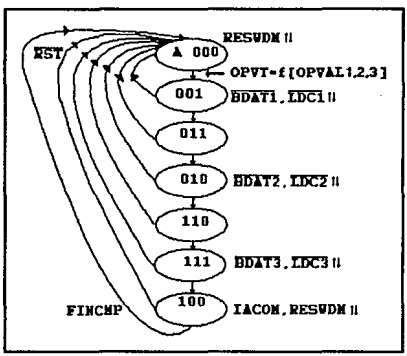


Figura 4.4 Autómata de Multiplexaje de Datos

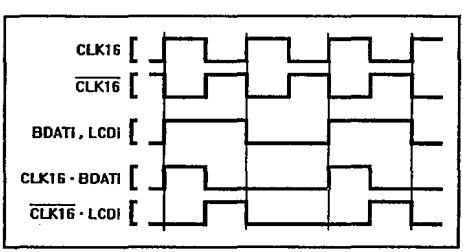
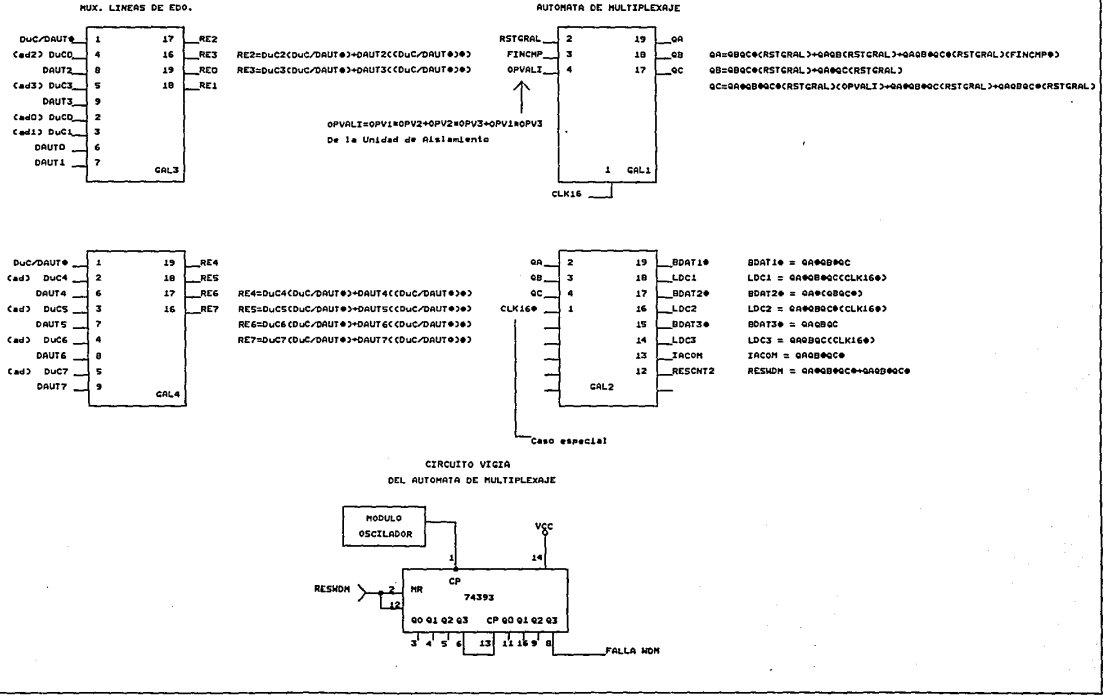


Figura 4.5 Diagrama de tiempos de las señales BDATi vs LDCi

Otro de los aspectos interesantes de diseño de este autómata radica en la posibilidad de autodetectar sus anomalías de operación por medio de un vigía electrónico, al cual se le programa en el estado A el tiempo máximo permisible de operación del autómata. En el caso de que éste exceda el tiempo permisible se envía una señal de control que genera el reemplazo automático de la UDFC actual por su contraparte redundante.

Figura 4.6 Diagrama de GALs del Autómata de Multiplexaje de Datos


MUX. LINEAS DE EDO.
DuC/DAUT0
(ad2) DuC0
DAUT2
(ad3) DuC3
DAUT3
(ad0) DuC0
(ad1) DuC1
DAUT0
DAUT1
GAL3
RE2
RE3
RE0
RE1
RE2=DuC2(DuC/DAUT0)+DAUT2((DuC/DAUT0)0)
RE3=DuC3(DuC/DAUT0)+DAUT3((DuC/DAUT0)0)
AUTOMATA DE MULTIPLEXAJE
RSTGRAL
FINCMP
OPVALI
GAL1
CLK16
QA
QB
QC
QA=QBQC0(RSTGRAL)+QAQB(RSTGRAL)+QAQB0QC0(RSTGRAL)(FINCMP0)
QB=QBQC0(RSTGRAL)+QA0QC(RSTGRAL)
QC=QA0QB0QC0(RSTGRAL)(OPVALI)+QA0QB0QC(RSTGRAL)+QAQB0C0(RSTGRAL)
OPVALI=OPV1*OPV2+OPV2*OPV3+OPV1*OPV3
De la Unidad de Aislamiento
DuC/DAUT0
(ad) DuC4
DAUT4
(ad) DuC5
DAUT5
(ad) DuC6
DAUT6
(ad) DuC7
DAUT7
GAL4
RE4
RE5
RE6
RE7
RE4=DuC4(DuC/DAUT0)+DAUT4((DuC/DAUT0)0)
RE5=DuC5(DuC/DAUT0)+DAUT5((DuC/DAUT0)0)
RE6=DuC6(DuC/DAUT0)+DAUT6((DuC/DAUT0)0)
RE7=DuC7(DuC/DAUT0)+DAUT7((DuC/DAUT0)0)
QA
QB
QC
CLK160
GAL2
BDAT10
LDC1
BDAT20
LDC2
BDAT30
LDC3
IACOM
RESCNT2
Caso especial
BDAT10 = QA0QB0QC
LDC1 = QA0QB0QCCLK160
BDAT20 = QA0QBQC0
LDC2 = QA0QBC0(CLK160)
BDAT30 = QAQBQC
LDC3 = QAQBQCCLK160
IACOM = QAQB0QC0
RESWDM = QA0QB0QC0+QAQB0QC0
CIRCUITO VIGIA
DEL AUTOMATA DE MULTIPLEXAJE
MODULO
OSCILADOR
VCC
CP
MR
74393
Q0 Q1 Q2 Q3
CP Q0 Q1 Q2 Q3
RESWDM
FALLA WDM

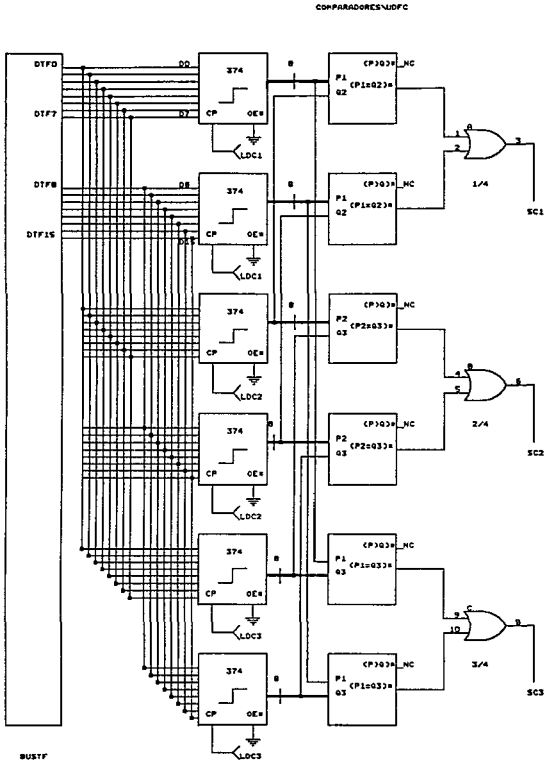
En la Figura 4.6 se observa el GAL1 y el GAL2, que constituyen el AMD, en tanto que el GAL3 y el GAL4 forman un multiplexor cuyas salidas conforman las entradas para el RE y el REP, a través de ellas, se controla la arquitectura y se permite la lectura de los bits respectivos desde cualquiera de las UPs. En la parte derecha del mismo diagrama se observa el circuito vigía del autómata de multiplexaje. Para clarificar la operatividad del multiplexor véase la Figura 4.1.

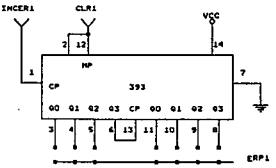
4.4 Autómata de Comparación y de Reconfiguración de la Arquitectura TF

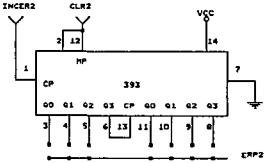
El principio de detección de fallas utilizado en la CTF se basa en el uso de una técnica de voteo mayoritario (enmascaramiento de fallas), la cual se presenta en el Apéndice A. En nuestro caso utilizamos tres registros de 16 bits, cada uno compuesto por dos CIs 74LS374 cuyas entradas provienen de las líneas de datos del ducto TF (datos multiplexados). Sus salidas se conectan a los CIs 74LS688 de operación continua, quienes realizan las comparaciones posibles entre datos provenientes de hasta tres UPs, como se muestra en la Figura 4.7. En el circuito mostrado, cuando se presenta un dato anómalo de cualquier UP, se puede detectar tanto la falla como la UP que la generó. Por ejemplo, en caso de presentarse un error en la UP1, las salidas SC1 y SC3 del circuito presentan niveles altos en tanto que SC2 permanece en bajo; cuando se tiene este tipo de relación, se busca el factor común entre las líneas que generan niveles altos. Para nuestro ejemplo, SC1 se deriva de las comparaciones entre datos de UP1 y UP2, en tanto que SC3 depende de los datos provenientes de UP1 y UP3, por lo que se deduce que la falla proviene de la UP1 (el factor común).

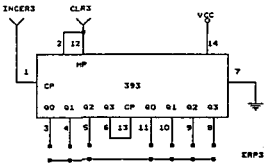
Un detalle importante de este circuito es el permitir la identificación de fallas en los mismos comparadores, pues como se vio en el ejemplo anterior la falla de alguna UP se manifiesta en dos de las tres salidas SC1, SC2 y SC3; por lo cual, de presentarse un sólo nivel alto en alguna de las señales anteriores indicaría la presencia de una falla en alguno de los comparadores o en los registros vinculados a ellos. Aprovechando esta cualidad del circuito, el ACDR integra la posibilidad de detectar este tipo de falla, la que de producirse, generaría también la conmutación automática hacia la UDFC redundante, para aislar el problema y permitir la operación continua del sistema.

Figura 4.7 Registros de Captura y Comparadores utilizados por el ACDR









SC1	SC2	SC3	DIAGNOSTICO
0	0	0	SIN FALLA
0	0	1	FALLA COMP3
0	1	0	FALLA COMP2
0	1	1	FALLA UP3
1	0	0	FALLA COMP1
1	0	1	FALLA UP1
1	1	0	FALLA UP2
1	1	1	FALLA UDFC

COMPARACION BUENA = SCi=0
COMPARACION MALA = SCi=1

Como se observa en el extremo derecho de la Figura 4.7, aparecen tres CIs 74LS393, los cuales se utilizan por el ACDR para cuantificar el número de fallas detectadas en cada UP. Estos CIs se añadieron con el fin de investigar la posibilidad de detectar errores transitorios y de evaluar sus repercusiones en diferentes aplicaciones. Como se puede deducir, una vez que una UP genera un error, generalmente se debe de producir un encadenamiento de errores, por lo que se antoja pensar que cualquier UP que genere una primera falla, debe ser aislada completamente del sistema. Sin embargo, pretendemos estudiar este fenómeno debido a que no existen reportes, hasta donde conocemos, acerca de este problema.

En la Figura 4.8 se presentan las diversas decisiones implantadas en el ACDR, vistas en forma de árbol. Mientras no existan anomalías en la CTF la ruta de decisiones corre a través de las señales IACOM (Inicia Autómata de Comparación, ACDR), ETS (Estado Total del Sistema Bueno), VBE (Voteo en Buen Estado) y NR (No hay Reconfiguración). Cuando aparece la primera falla, esta podrá provenir de alguna UP o de los comparadores de la UDFC activa, en este caso se pueden presentar las siguientes rutas:

- IACOM-ETS-VBE#-FCOMP (Falla en comparador): culminando con la conmutación de la UDFC.

- IACOM-ETS-VBE#-FTAR (Falla de alguna tarjeta UP): con alguna de las siguientes rutas:

- FT1 (Falla de UP1) en cuyo caso se declara a la UP1 como redundante (y se asigna la categoría de principal a alguna de las UPs restantes) en espera de nuevas fallas transitorias las que al llegar a un máximo programado ocasionan su completa desactivación, seguida de alarmas enviadas al usuario.

- FT2 (Falla en UP2) ocasionando análogamente al punto anterior el que se ceda la categoría principal a UP1 o UP3 y se esperan nuevas fallas transitorias.

- FT3 (Falla de UP3) se procede de manera similar a los dos puntos anteriores.

En caso de que el sistema incurra en una segunda falla sin que se haya restaurado la primera, la ruta de operación del sistema sería por IACOM, ETS# (que indica la existencia de una falla previa) y alguna de las siguientes posibilidades:

- ET1#, ERP2. En cuyo caso se aísla a la UP1 y UP3.

- ET2#, ERP3. En cuyo caso se aísla a UP2 y a UP3.
- ET3#, ERP1. En cuyo caso se aísla a UP2 y a UP3.
- En las rutas restantes no se realiza ninguna reconfiguración.

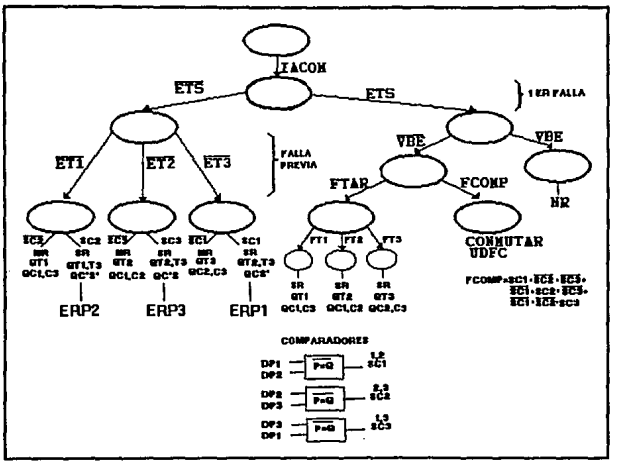


Figura 4.8 Árbol de decisiones para el ACDR

En el párrafo anterior se explicaron a grandes rasgos las reglas de detección de fallas y de reconfiguración. En la Figura 4.9 se muestra el autómata del ACDR que se implantó electrónicamente en la UDFC, en él se indican las siguientes rutas:

- Sistema sin Fallas (estados A, B y M)
- Detección de Fallas Transitorias en UP1, UP2 ó UP3 (estados A, B, C, D y E)

- Diagnóstico de Fallas Permanentes así como la reconfiguración utilizada (estados I, J, K y L).

- Detección y Diagnóstico de una segunda falla (sin haber dado atención a la primera) estados A, B, F, G y H.

- Encuesta al μC 68HC11 para cederle la posibilidad de reconfigurar el sistema a sugerencia del usuario. Esta opción se utiliza para dar mantenimiento preventivo, correctivo o bien para propósitos de demostración (estados M, N, O y P).

Antes de que el autómata llegue al fin de alguna de las rutas, genera las señales QFREN (Quita Freno) y FINCMP para liberar a los demás autómatas y para que la CTF continúe sus tareas encomendadas.

De este autómata también debe subrayarse su circuito de autodiagnóstico, el cual es similar al utilizado por el AMD. En este caso en el estado A se arranca a un circuito vigía programado con un tiempo relativamente mayor al máximo tiempo permisible de operación de este autómata, de ser superado indicará la imposibilidad del autómata para terminar alguna de sus rutas. En este caso se produciría, al igual que en el AMD, la generación de una señal que acciona la conmutación automática de la UDFC activa a la UDFC redundante.

En la Figura 4.10 se muestran las diferentes GALs requeridas para integrar el ACDR. Las GALs 5, 6, 7, 8, 9 y 13 conforman el ACDR. Con las GALs indicadas se logró integrar todas las entradas y todas las salidas de control de este autómata; junto a cada una de ellas se indican las funciones lógicas programadas. Como puede desprenderse, de no haber utilizado dispositivos programables, el número de componentes en esta tarjeta haría poco práctica su construcción. En la misma Figura 4.10 en la parte inferior izquierda se presenta el circuito vigía que permite detectar fallas operativas del ACDR.

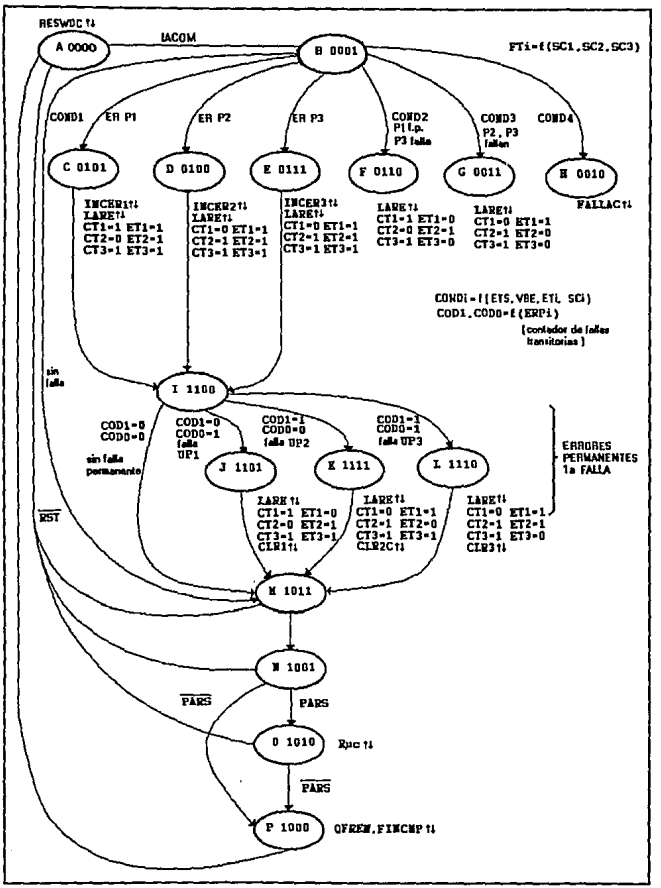
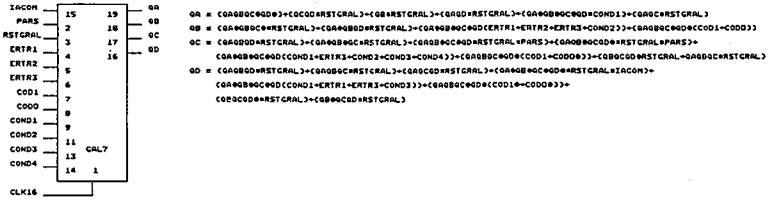
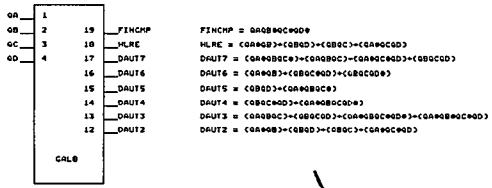
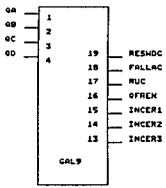


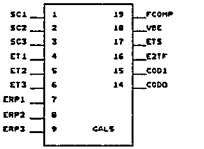
Figura 4.9 Diagrama de estados del ACDR

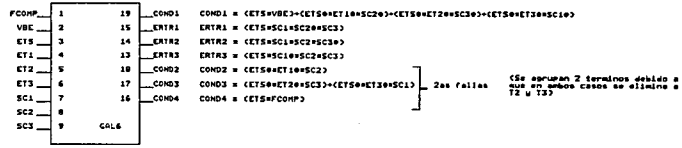
Figura 4.10 Banco de GALs del ACDR

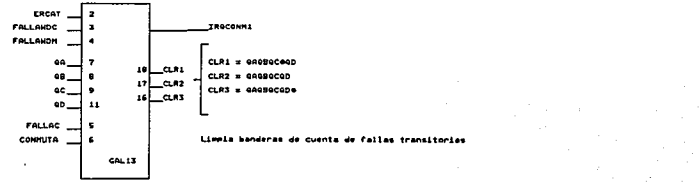




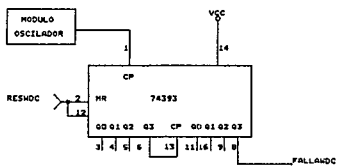








VIGIA DEL AUTOMATA DE COMPARACION Y RECONFIGURACION



4.5 Inicialización y Mantenimiento en Línea de la Arquitectura TF.

Como se ha establecido la UDFC, a grandes rasgos, está compuesta por dos bloques: 1) El de autómatas de multiplexaje de datos, detección y diagnóstico de fallas y,

2) La electrónica de interacción con el usuario para labores de mantenimiento, la cual se basa en un μC 68HC11.

Como también se ha mencionado, el propósito de integrar a un microcontrolador dentro de la UDFC se debió a la necesidad de contar con un dispositivo de control versátil y de amigable programación para desarrollar diversos procedimientos de control imprescindibles en la CTF, los cuales se enumeran a continuación:

- **Inicialización de UPs con Sistema Operativo (SO) y programa de aplicación**, así como el establecimiento de la sincronización requerida para permitir la operación concurrente de hasta tres UPs.

- **Envío de alarmas visuales y auditivas** después de la ocurrencia de algún tipo de falla en alguna de las tarjetas objetivo (UPs o UDFCs) del sistema TF.

- **Mantenimiento preventivo**. En cualquier instante de operación de la CTF, el usuario puede solicitar, a través del teclado de mantenimiento del STF, diferentes tipos de reconfiguración del sistema como pueden ser:

- Cambio de UDFC

- Cambio de categoría de alguna UP

- Cambio de estado de alguna UP

Durante el mantenimiento preventivo de alguna tarjeta objetivo, el usuario debe solicitar la desactivación de la tarjeta indicada, ante lo cual el sistema le indica cuando desenergiza al módulo respectivo, para que entonces el usuario proceda a remover del sistema la tarjeta en cuestión. Durante este procedimiento, generalmente el usuario deberá contar con una tarjeta de repuesto para insertarla al sistema. En estos casos, a través del teclado de mantenimiento se debe indicar al sistema que se procederá a instalar un módulo nuevo, después de ser insertado y declarado se energiza y se

procede a inicializarlo. Es aquí donde cobra aún más relevancia el trabajo del μC, pues además de interactuar con el usuario permite interaccionar con la arquitectura para declarar y para inicializar nuevos módulos.

- **Mantenimiento Correctivo.** A diferencia del punto anterior este tipo de tarea es detectada y solicitada automáticamente por la UDFC a través de sus autómatas que operan en tiempo real. Sin embargo, el μC es el encargado de gestionar el envío de avisos y alarmas al usuario para señalar la falla ocurrida. Además, cuando el personal de mantenimiento reacciona ante la falla, el μC interacciona con el usuario para conducirlo en los diferentes pasos para realizar el mantenimiento requerido.

Como ya se ha mencionado en otros capítulos, el haber integrado un sistema triplex monitoreado por una UDFC duplex además de incluir subsistemas de aislamiento para UPs y UDFCs, permitió conferirle a la arquitectura del sistema un atributo muy importante, el mantenimiento en línea, mediante el cual se pueden remover o adicionar tarjetas a la arquitectura sin que ello afecte el desarrollo de los programas de aplicación ni los datos generados o recién procesados en el sistema. Cuando se ha mencionado en los párrafos anteriores algún tipo de mantenimiento, debe recordarse que es posible agregar o quitar tarjetas a la CTF sin riegos de producir cortos circuitos debido a que la interacción usuario-68HC11-UAs hace posible la ejecución de procedimientos ordenados para efectuar los cambios ya indicados.

En las dos secciones siguientes se comentarán los detalles técnicos integrados en el diseño que permiten realizar las funciones descritas anteriormente.

4. 6 Mecanismos de Autodiagnóstico de la Tarjeta

Durante las diferentes fases de diseño electrónico realizadas para la tarjeta UDFC se idearon e integraron circuitos susceptibles de ser evaluados de forma automática; como es de pensarse se utilizaron técnicas de redundancia para contar con información veraz que pudiera respaldar la toma de decisiones.

Como ya se ha mencionado, esta tarjeta está compuesta por dos grandes bloques, el de autómatas y el del μC, siendo la primer parte las más crítica de ellas debido a que

opera y realiza evaluaciones en tiempo real. Por esta razón se acentuó la detección de fallas en tal bloque; respecto al μC, por el momento se piensa realizar algún tipo de detección de fallas que involucre la ejecución de programas de autodiagnóstico, los cuales quedan fuera del alcance de la presente tesis.

Como se recordará, esta tarjeta contiene dos autómatas, el primero realiza el multiplexaje de los datos generados en el sistema y el segundo, reconfigura la arquitectura cuando es requerido. En cuanto al multiplexaje poco se puede hacer para detectar posibles fallas durante la transmisión de datos, por lo cual se delega parte de esta responsabilidad al ACDR, el resto de las funciones de este autómata se diagnostica y evalúa por medio de circuitos vigía, los cuales se explicaron en la sección IV.3. El autodiagnóstico del autómata de multiplexaje se realiza con un circuito que genera el pulso FALLAWDM cuando la electrónica que compone al autómata deja de operar por alguna razón, ver Figura 4.2.

La parte de detección de fallas y su diagnóstico se desarrolla en el ACDR por lo cual contiene circuitos electrónicos para detectar anomalías tanto en la fase de detección de fallas como en la parte operativa del autómata en sí. Al ser fundamental la comparación de datos del sistema triplex se utilizó un esquema tal que permite diagnosticar fallas con relativa facilidad, como se observa en la Figura 4.7. Existen condiciones específicas que indican algún tipo de falla en cualquiera de las UPs instaladas, como también existen condiciones únicas para determinar la falla de un comparador. La electrónica diseñada para diagnosticar los comparadores produce la señal FALLACOMP cuando ellas fallan, el ACDR incluye también un circuito vigía que genera un pulso en la señal FALLAWDC cuando el autómata presenta algún tipo de anomalía, de forma similar a la generada por el AMD.

Como se puede observar en la Figura 4.7, cada una de las señales de falla mencionadas anteriormente se conducen a una compuerta NOR, cuya salida constituye la señal de solicitud de conmutación de UDFC. Cada vez que se genera alguno de estos pulsos de falla se considera que el error que se ha presentado es de alta magnitud, por lo cual se hace indispensable la conmutación de UDFC para continuar la supervisión de la CTF. La orden de conmutación se canaliza a la UDFC redundante, a través del ducto TF, llegando directamente a la línea de solicitud de interrupción del μC, forzándolo a que inicie actividades como UDFC principal y a desactivar la UDFC recién fallada. Uno de los detalles interesantes de esta tarjeta tiene relación con el diseño del circuito impreso el cual es el mismo para la tarjeta

UDFC principal y para la tarjeta redundante, esto se logró al incluir interruptores que se programan manualmente, consiguiendo así que una salida particular de una de las tarjetas se convierta en una entrada en la tarjeta complementaria. De esta forma, al insertarse una nueva UDFC de reemplazo con la misma posición de los interruptores (que los de la UDFC fallada) es posible continuar la detección de fallas y la autoconmutación hacia otra UDFC de respaldo de forma cíclica. Este detalle de diseño que involucra tanto electrónica como programas constituye una de las características más importantes de esta tarjeta, pues mientras exista la posibilidad de realizar mantenimiento correctivo se asegura la continuidad de la detección de fallas de forma totalmente transparente para el usuario, a quién sólo se le exige (cuando se producen fallas) reemplazar una tarjeta fallada por otra en buen estado con los interruptores en la misma posición que los de la tarjeta por reemplazar. Se enfatiza este diseño debido a que rompe la necesidad de contar con otra Unidad de Detección de Fallas encargada de supervisar a las mismas UDFCs. De ahí la importancia del autodiagnóstico y de la autoreconfiguración, pues de no haber sido posible el diseño con tales atributos habría resultado impráctica la construcción de la CTF. También reenfatizamos el atributo de mantenimiento en línea de esta tarjeta, el cual es posible gracias a la presencia y al trabajo de las unidades de aislamiento las cuales fueron especialmente diseñadas para interactuar con las UDFCs, para mayores detalles sobre UAs ver la sección II.4.

Adicionalmente, en la Figura 4.7 aparecen las señales ERCAT y FALLAC. ERCAT se genera en la UA para UDFCs cuando existen dos UPs declaradas con categoría principal. Los resultados producidos por este tipo de falla serían evidentemente desastrosos para la arquitectura, por lo que, en caso de presentarse tal situación se obliga la conmutación de UDFC; para esto debe recordarse que el RE de la UDFC principal es quien fija las señales de categoría y estado para UPs, por lo que la presencia de la señal ERCAT implicaría ya sea un error en el RE o en alguna parte de su electrónica asociada y; por tal razón, el cambio a una nueva UDFC solucionaría el problema.

La señal G(conmuta) es generada por el μC cuando el usuario desea realizar una conmutación de UDFC, la cual ha sido planeada principalmente para propósitos de demostración o en el caso de que el μC autodetecte problemas en su funcionamiento (opción a futuro).

Debe recalcarse que las acciones de conmutación automática o solicitada se ejecutan en tiempo real con lo cual se asegura una amplia cobertura de fallas, redundando en la operación continua del sistema. En la siguiente sección se comentarán los tipos de protocolos y operaciones que se realizan en la programación de las UDFCs y UPs para permitir la continuidad de operaciones entre perturbaciones originadas por fallas. Esta parte es de fundamental importancia pues de la correcta inicialización y puesta en marcha de la nueva UDFC depende la continuidad esperada, a nivel de instrucción, de UPs.

Cuando se inserta una nueva UDFC en el sistema, existe un pequeño protocolo de arranque, el cual tiene por objeto eliminar inconsistencias en la asignación de categoría de UDFCs instaladas. En este caso las inconsistencias aparecen cuando el usuario fija erróneamente la posición de los interruptores en las UDFCs, los cuales indican el tipo de categoría adjudicada a cada tarjeta durante el arranque. Como es lógico pensar los errores se presentan cuando se asignan categorías iguales (principal-principal o redundante-redundante). En la Figura 4.2 se observa al interruptor R/P# el cual se utiliza para definir la categoría de la UDFC. Aquí las rutinas ejecutadas por el μC son diferentes a las de autoinicialización por arranque del sistema, pues en el primer caso se detecta que ya están inicializadas todas las UPs y que por tanto existe ya una UDFC principal, por lo cual los algoritmos la llevan a declararse como UDFC de respaldo. Por esta razón al instalar una nueva UDFC en el sistema ya no importa la posición del interruptor R/P#, pues la autoinicilización de la tarjeta se salta la lectura de esta señal.

Para determinar si se trata del arranque inicial de todo el sistema o de la simple inicialización de una nueva UDFC, el 68HC11 evalúa el estado de la señal ARQUE = RST1+RST2+RST3 (donde RSTi es la línea de RESET aplicada a la UPi) la cual tiene nivel bajo durante la inicialización del sistema, pues cuando aún no se carga el SO en ninguna UP la UDFC aplica RESET (nivel bajo) a cada UP. En caso de tratarse de la inicialización de una nueva UDFC al muestrearse la señal ARQUE, ésta contiene un nivel alto lo cual implica que al menos existe una UP que no está en estado de RESET, obteniéndose así la posibilidad de detectar algunos de estos estados de funcionamiento y en consecuencia obligar en las UDFCs la inicialización deseada.

4.7 Protocolo de Comunicación entre UDFCs y UPs

En las secciones y capítulos anteriores se han descrito particularidades del diseño de la CTF y en algunos de ellos se describió la necesidad de interacción entre UDFC y UPs para permitir actualizar datos y variables de inicialización de tarjetas recién instaladas en el sistema TF.

El protocolo básico de comunicación entre UDFCs y UPs consiste en el envío de una señal de interrupción desde el μC hacia las UPs instaladas para captar la atención de todas ellas, quienes responderán leyendo el RA de la UDFC principal. En este registro la UDFC coloca previamente información que define el tipo de acción solicitada, así como a la(s) UP(s) a quien(es) va dirigida. En la Tabla 4.1 se observan las diferentes acciones utilizadas en el sistema y la forma de definir a las UPs involucradas. Se pueden hacer combinaciones utilizando las listas indicadas. Como se observa existen combinaciones no utilizadas las cuales pueden servir para modificaciones o bien para integrar nuevos atributos o formas de interacción entre UDFCs y UPs. Enseguida se explican brevemente las acciones programadas:

Acción 1 Arranque Síncrono. Se utiliza durante el arranque del sistema triplex o bien cuando por alguna razón se ha procedido a detener la operación del sistema y se desea reanudar nuevamente la operación concurrente síncrona.

Acción 2. Ordena que se cargue el archivo CTFBACK.DAT en un nuevo procesador por declarar en el sistema.

Acción 3. Indica respaldar toda la información contenida en RAM del procesador principal en el archivo CTFBACK.DAT de disco duro. Esta opción se usa en combinación con la acción anterior para inicializar a una nueva UP con SO, programa de aplicación y datos de la aplicación.

Acción 4. Se le avisa a todas las UPs que ha ocurrido una conmutación de UDFC y por tanto se les obliga a frenar operaciones hasta que lo ordene la nueva UDFC. Es un paso adicional para asegurar la sincronización después de la falla de la UDFC principal.

Acción 5. Esta acción le indica a las UPs activas que ha habido algún cambio en el RE de la UDFC principal. Concretamente indica que se ha modificado el valor de

alguno de los bits de categoría o estado contenidos en dicho registro, señalando la detección de una falla o bien cambios realizados por el usuario (demostración).

Acción 10. Asigna número de UP al procesador corriente. Esta opción se utiliza para retroalimentar a la UP respectiva un número con el que será identificada por las UDFCs. Se debe recordar que algunas acciones van dirigidas hacia procesadores específicos, para ello, cada UP debe saber qué número o clave se le ha asignado (1, 2 ó 3). Esta acción se le envía a las UPs durante el arranque del sistema TF y durante la inicialización de una UP recién instalada.

Acción	RA	
	No. UP	Código Acción
1	0XXX	0001
2	0XXX	0010
3	0XXX	0011
4	0XXX	0100
5	0XXX	0101
10	0XXX	1010
donde XXX = 001 ==> UP1 010 ==> UP2 011 ==> UP3 100 ==>Todas		

Tabla 4.1 Codificación del Registro de Acción

En el siguiente capítulo se mostrarán los diagramas de flujo que integran las diversas fases de comunicación entre UDFCs y UPs, ellos contienen los procedimientos detallados para efectuar el control de la arquitectura concurrente.

Para cerrar el lazo de comunicación entre UDFCs y UPs, el diseño electrónico de la UDFC prevé accesos desde las unidades de procesamiento mediante puertos digitales de entrada. En este sentido se incluyeron en la tarjeta cuatro habilitadores de puertos: LECRA#, RDRE#, FINITA# y LDRER#. Estos puertos se pueden observar en el extremo superior izquierdo de la Figura 4.2. Sus funciones se comentan enseguida:

LECRA# (Lectura del Registro de Acción). Por medio de este puerto las UPs realizan la lectura de la acción de interrupción solicitada por la UDFC principal.

RDRE# (Lectura del Registro de Estado). Por medio de este puerto las UPs actualizan su información acerca del estado corriente de la arquitectura TF.

FINITA# (Fin de Inicialización de Tarjeta). Con este puerto cada procesador emite una respuesta de fin de inicialización durante el arranque de las UPs (una a la vez). Durante la operación del sistema este puerto es utilizado por la UP principal para indicar que los procesadores instalados se encuentran detenidos (estado *HALT*). Esta opción se utiliza como un medio para conservar la sincronización de los procesadores en los siguientes casos: cuando se requiere dar de alta a un nuevo procesador, después de la conmutación de una UDFC, después de que las UPs realizan la lectura ya sea del RE o del RA, etcétera.

LDRER# (Carga el Registro de Estado Redundante). Cada vez que existe algún proceso de reconfiguración en la arquitectura TF, a las UPs se le ordena leer el RER de la UDFC corriente y de almacenarlo en el RER de la UDFC redundante. La escritura al RER de la UDFC redundante también se realiza después de dar de alta a una UDFC recién instalada.

4.8 Diseño del Circuito Impreso de la UDFC

Para desarrollar el circuito impreso de esta tarjeta, se utilizó también el paquete de diseño asistido por computadora denominado TANGO PCB PLUS, del cual se han mencionado ya sus características generales en la sección III.6. El impreso se diseño en dos caras, para lo cual se elaboraron cuatro capas (soldadura, componentes, referencias y perforaciones) las cuales se muestran en las Figuras 4.11, 4.12, 4.13 y 4.14.

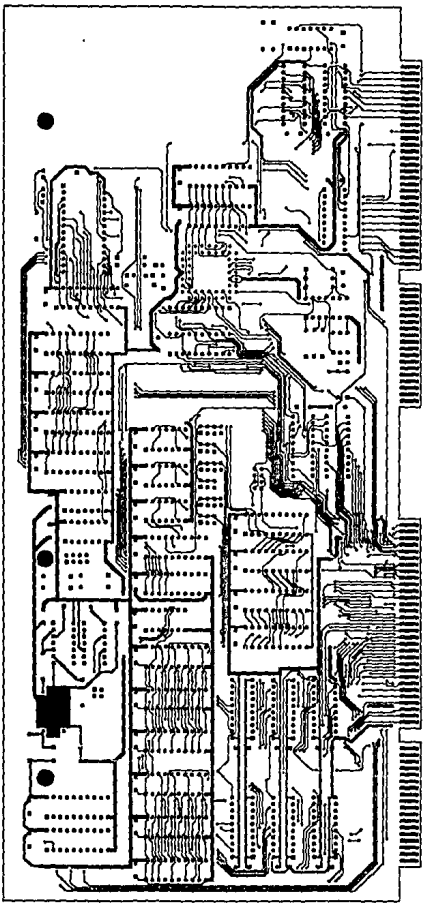


Figura 4.11 Capa de componentes de la UDFC

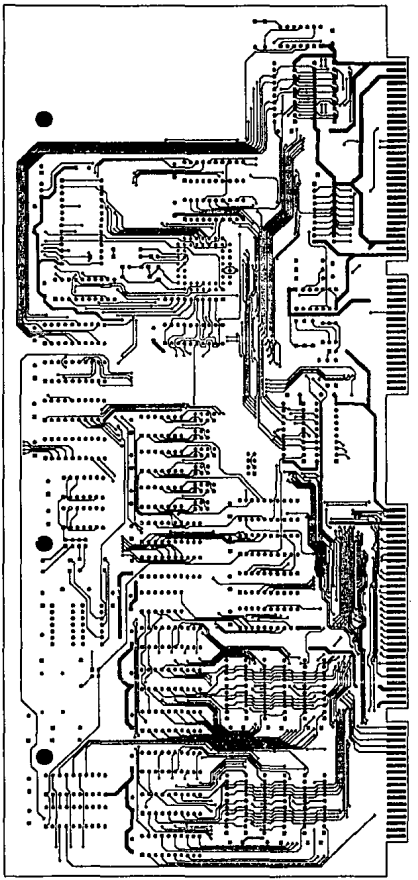


Figura 4.12 Capa de soldadura de la UDFC

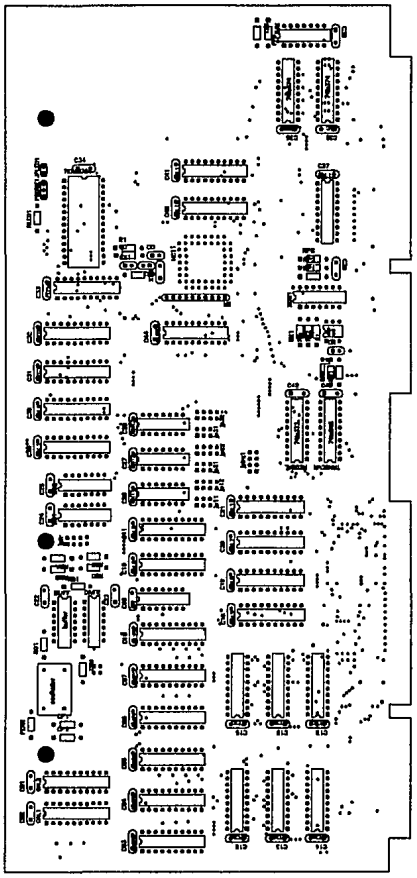


Figura 4.13 Capa de referencias de la UDFC

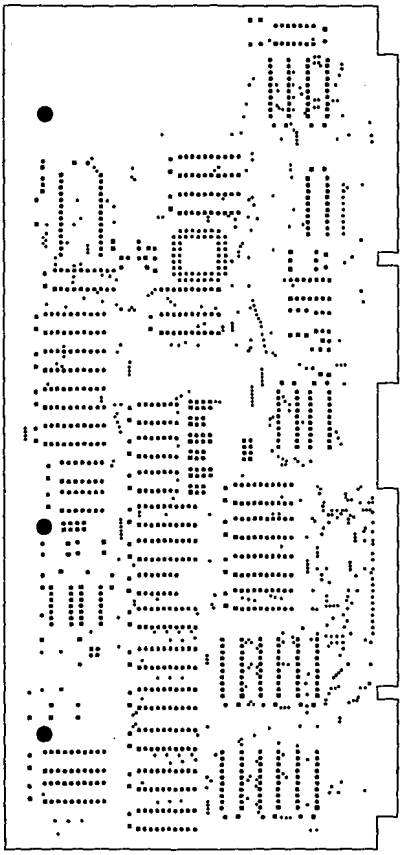


Figura 4.14 Capa de perforaciones de la UDFC

Como también se ha mencionado en secciones anteriores, el trazado de líneas de los circuitos impresos elaborados para esta tesis se ha efectuado en su totalidad por el usuario y, en vista de las características operativas de la electrónica contenida en la UDFC cobró mayor relevancia el trazado de líneas de la sección de autómatas de multiplexaje, de comparación de datos y de reconfiguración de la arquitectura, pues como se recordará este módulo electrónico trabajará a una frecuencia de **16 MHz**, y de resultar exitosas las pruebas se procederá a incrementar la frecuencia al doble del valor inicial. Por esta razón, se trató de que el diseño de estas etapas tuviera el menor número de conexiones y las distancias mínimas de recorrido para reducir efectos de ruido, usual en circuitos digitales de alta velocidad.

Una vez terminada la fase de mayor prioridad del impreso se procedió al trazado de las líneas restantes de la tarjeta, las que como se recordará pertenecen al μC 68HC11, a registros y a la lógica digital de control, las que operan a frecuencias relativamente bajas si se compara con la frecuencia propuesta para la electrónica de autómatas. No obstante el tamaño de la tarjeta, la parte final de la misma se complicó de sobremanera debido al espacio disponible tan reducido y por la gran cantidad de líneas trazadas. Como es lógico pensar, a medida que se dibujan más líneas, disminuyen los espacios de tránsito para la ubicación de nuevas líneas de conexión requeridas por la electrónica.

Nuevamente, como ya se ha mencionado, afirmamos que es más sencillo realizar circuitos impresos de tipo multicapa debido a que se cuenta con bastante área disponible y en consecuencia es menos laborioso el problema del trazado de líneas, sin embargo, también afirmamos que el desarrollo de circuitos impresos de dos caras resulta atractivo por su bajo costo de producción y sobre todo por la posibilidad de poder realizar mantenimiento correctivo de las tarjetas; pues como se puede imaginar resulta prácticamente imposible realizar modificaciones a líneas residentes en capas internas de circuitos impresos multicapa. En el Apéndice C se dan las estadísticas de los diversos trazos utilizados para el desarrollo de la tarjeta. Los números son indicadores reales de la actividad tan laboriosa que resulta en este tipo de diseños, cuando se hacen manualmente, y también debe recordarse que gran parte de este trabajo puede evitarse mediante el uso de paquetes de CAD que tengan la capacidad de autorutear (trazar automáticamente) líneas que componen al circuito impreso. Pero también debe recordarse que los precios de este tipo de paquetes (alrededor de los 9000 dólares) los hace poco accesibles para este tipo de proyectos.

Como ya se mencionó en secciones anteriores el paquete TANGO permite generar archivos de salida compatibles con impresoras láser tipo Linotronic, de tal forma que por medio de este periférico es posible generar copias de alta resolución en película fotográfica de superficie amplia. Dichos negativos se envían a compañías privadas para la fabricación de impresos de alta calidad.

IV.9 Unidad de Interfaz Múltiple

La UDFC contiene electrónica que le permite interaccionar con el usuario de forma totalmente dedicada para que en cualquier momento de las diversas etapas de operación de la arquitectura TF el usuario pueda solicitar algún tipo de cambio o de reconfiguración en el sistema. Para este propósito la UDFC posee una interfaz para un teclado de membrana exclusivo para labores de mantenimiento correctivo-preventivo-demostrativo; y adicionalmente cuenta con una interfaz para manejar una pantalla de cristal líquido utilizada para enviar diversos tipos de mensajes durante alguna de las actividades citadas.

Cuando se diseñó la tarjeta UDFC se contempló la posibilidad de incluir las interfaces citadas dentro de la misma tarjeta; sin embargo, se visualizó un problema relevante, el cual tiene que ver con el hecho de que la UDFC se encuentra duplicada en el sistema y en caso de falla de la UDFC principal se conmuta a la tarjeta redundante. El problema encontrado radicó en que para poder tener ambas UDFCs trabajando, sería necesario, ante la falla de la principal, el tener que realizar conexiones externas para conmutar el teclado y la pantalla de cristal líquido, solución que resulta inadecuada debido a que durante el reinicio de gestiones de la tarjeta de respaldo, después de una falla en la UDFC principal, no tendría conectados los periféricos citados. Una segunda solución planteada fue la de utilizar dos UDFCs cada una con su respectivo teclado y su despliegue de cristal líquido, solución que como podemos concluir resulta poco atractiva por los espacios utilizados en el chasis del equipo y por las redundancias excesivas de periféricos que normalmente no requieren de contrapartes de respaldo.

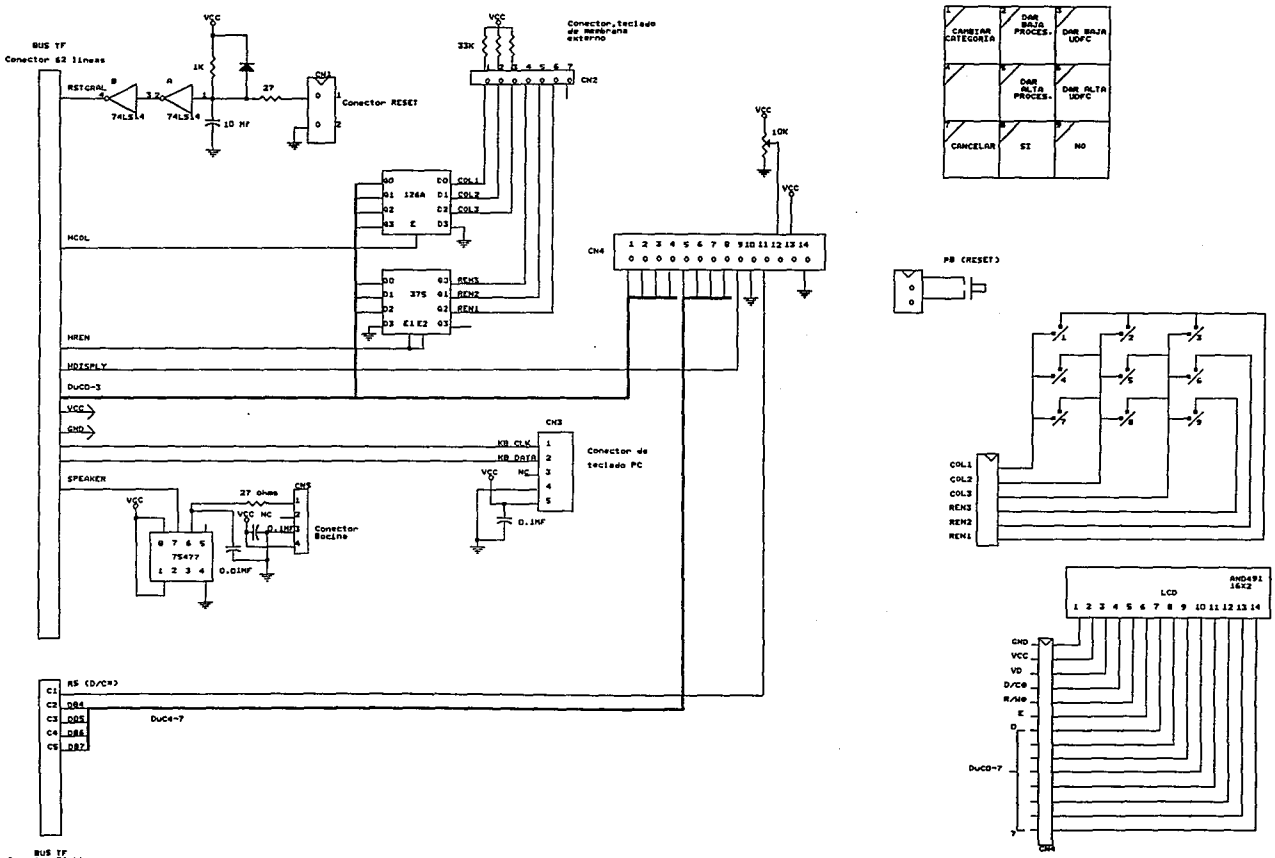
La solución que se propuso para este problema conserva fragmentos de las soluciones arriba mencionadas, para esto se proyectó una UDFC cuya operación es completamente independiente de su parte redundante y que por tanto cualquiera de ellas se puede extraer o incorporar sin necesidad de realizar conexiones externas para

teclado y pantalla de cristal líquido. Para lograr este objetivo se trasladaron los conectores de ambas interfaces hacia otra tarjeta, la cual contiene únicamente cuatro CIs y que por tanto tiene muy baja probabilidad de falla debido a que sus componentes son de muy baja escala de integración. Adicionalmente, en esta tarjeta se instalaron los conectores para teclado y pantalla, los cuales se conectarán con los periféricos colocados en una de las paredes del mueble de la computadora. A esta tarjeta la denominamos Unidad de Interfaz Múltiple, la cual se utiliza además para contener los conectores de RESET, bocina y teclado convencional de PC, estos últimos utilizados en el modo TF. De los conectores de esta tarjeta salen los cables respectivos hacia los conectores instalados en las paredes del gabinete de la CTF y de ahí se conectan a los periféricos en cuestión. En la Figura 4.15 se observa el diagrama de bloques de la UIM, como se nota, está formada principalmente por conectores.

El diseño de los pocos componentes electrónicos integrados en esta tarjeta se realizó de tal forma que incluso sea posible la extracción en línea de la UIM con posibilidades mínimas de ocasionar cortos circuitos. Aunque esta decisión no es recomendable y además poco probable de suceder debido al número reducido de componentes que pueden fallar. Esta prueba será recomendable realizarla para conocer las repercusiones de esta acción.

De esta forma ha sido posible el diseño de la UDFC para que se le pueda dar mantenimiento en línea sin importar cuantas UDFCs se encuentren instaladas, de tal forma que la UDFC principal siempre puede hacer uso del teclado y la pantalla de cristal líquido.

Figura 4.15 Diagrama eléctrico de la UIM


BUS TF
Conector 62 lineas
RSTGRAL
74LS14
74LS14
1K
10 MF
27
CN1
Conector RESET
VCC
33K
Conector,teclado
de membrana
externo
CN2
1OK
HCOL
HREN
HDISPLY
DuCD-3
VCC
GND
SPEAKER
CN4
PB (RESET)
CN3
KB CLK
KB DATA
VCC
NC
Conector de
teclado PC
27 ohms
CN5
Conector
Bocina
75477
0.01MF
0.1MF
COL1
COL2
COL3
REN3
REN2
REN1
CAMBIAR CATEGORIA
DAR BAJA PROCES.
DAR BAJA UDFC
DAR ALTA PROCES.
DAR ALTA UDFC
CANCELAR
SI
NO
LCD
AND491
16X2
GND
VCC
VD
D/CO
R/WO
E
DuCD-7
CN6
RS (D/CN)
DB4
DB5
DB6
DB7
DuC4-7
BUS TF
Conector 36 lineas

Programación de Unidades de Procesamiento y de UDFCs

5.1 Introducción

En este capítulo se describen los procedimientos generales de programación diseñados para UPs y para los microcontroladores de las UDFCs, los cuales realizan el trabajo de detección, diagnóstico y reconfiguración de la arquitectura TF en casos de fallas en cualquiera de las tarjetas objetivo. Para esta tesis se desarrollaron básicamente los algoritmos de comunicación y los protocolos necesarios entre UDFCs y UPs. Los algoritmos están estrechamente relacionados con el diseño de la electrónica para la arquitectura, por esta razón se dedicó especial atención al desarrollo de éstos. En trabajos posteriores que tendrán que ver con la validación de la electrónica, la integración de los diversos impresos de la arquitectura y estos algoritmos, se reportarán resultados específicos.

Enseguida se explicarán los algoritmos y se enfatizarán ideas ya mencionadas en la sección IV.7. En el siguiente y último capítulo se discuten las conclusiones y recomendaciones, que a juicio del autor permitirán concluir el desarrollo de la computadora tolerante a fallas.

5.2 Rutinas de Control para el microcontrolador 68HC11 contenido en la UDFC

En vista de que los algoritmos de control diseñados para el μC de las UDFCs son de gran importancia, se han desarrollado éstos hasta sus últimos detalles; sería muy extenso describirlos en forma escrita, por lo cual, se darán sus detalles generales y sus aspectos relevantes, remitiendo al lector a la consulta de los diagramas de flujo elaborados para obtener información detallada de la programación.

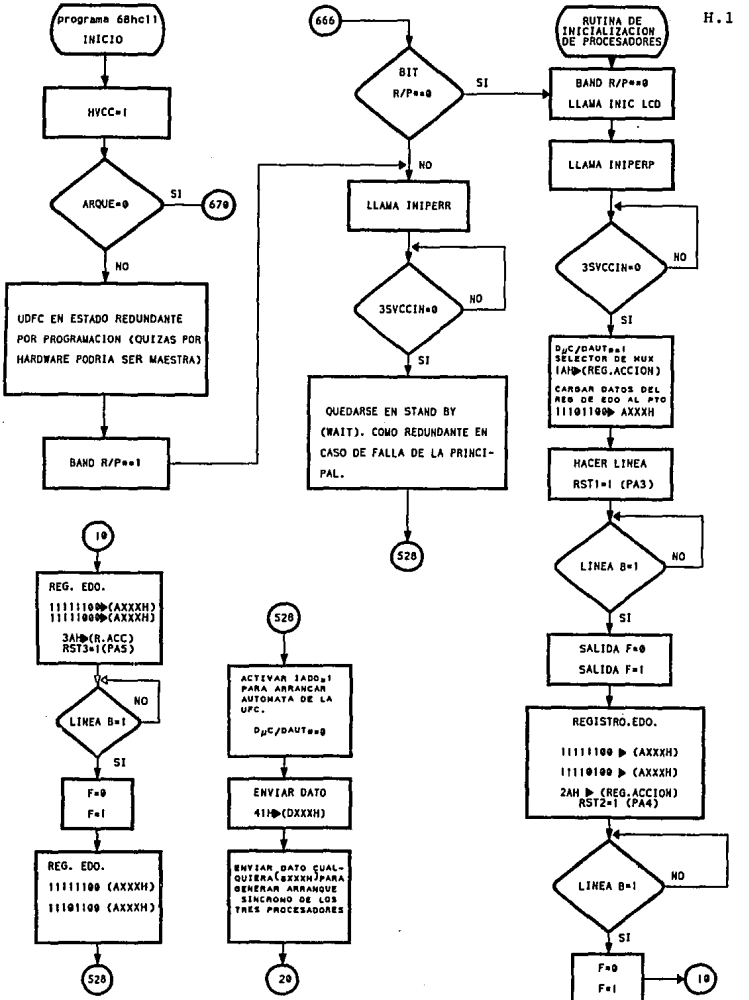
Después del arranque, el 68HC11 pregunta por una bandera para saber si se trata de una inicialización en frío (*cold booting*) o una en tibio (*warm booting*). Dependiendo de esta condición se procede a detectar la posición del interruptor de categoría de la UDFC para arrancar como tarjeta principal o para iniciar actividades como UDFC de respaldo. Esto es, cuando se trata de un arranque en tibio se presupone la existencia de una UDFC principal instalada, por lo cual al insertarse una nueva UDFC, ésta necesariamente se declarará como redundante independientemente de la posición del interruptor que define su categoría. Durante un arranque en frío, el μC (si existen dos UDFCs instaladas) detecta la posición de su interruptor de categoría y posteriormente pregunta por la bandera ERCUDFC, la cual se genera por medios electrónicos y avisa cuando dos UDFCs han sido declaradas con la misma categoría, en estas circunstancias ambos μC generan una señal de error indicada por un LED y detienen todo tipo de operaciones. En este caso, cuando el usuario se percata de este aviso sabe que se trata de un error en la asignación de categorías de las UDFCs.

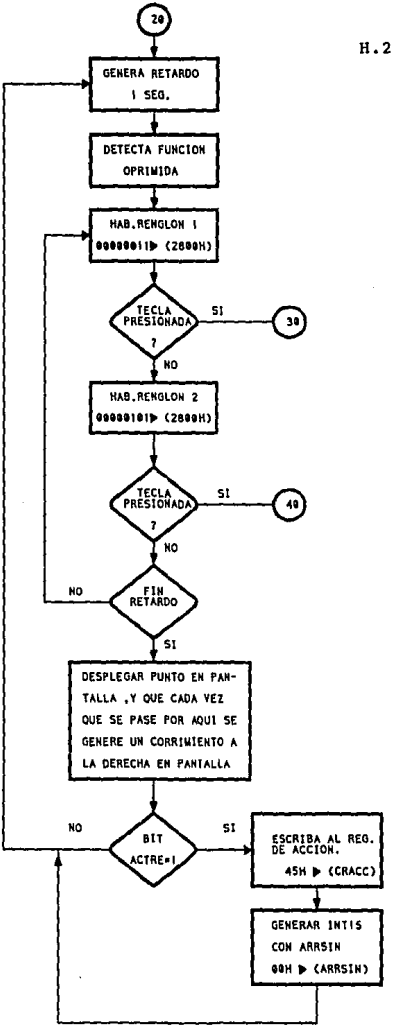
En caso de no existir conflictos de categoría, esto es, cuando existe una UDFC programada como principal y una como redundante, cada UDFC ejecuta un programa diferente, la redundante inicializa sus puertos de salida y espera a que desaparezca el pulso 3SVCCIN, el cual forza que durante el arranque las UDFCs sean energizadas y tomen la categoría redundante por 3 segundos, hasta que las tarjetas identifiquen su categoría y dependiendo de ello establezcan sus condiciones iniciales. Una vez finalizado este pulso, el μC de la UDFC redundante entra en un estado de espera (*WAIT* de bajo consumo de energía) del cual saldrá sólo si le llega una solicitud de interrupción producida automáticamente al detectarse una falla en la UDFC principal.

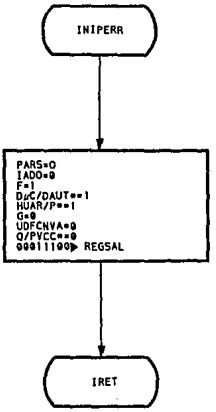
Cuando el μC detecta categoría principal, ejecuta los procedimientos necesarios para inicializar las tres UPs, una a una, cargándoles en primera instancia el SO y posteriormente el programa de aplicación. Entre la inicialización de una y otra UP existe un protocolo de comunicación entre el μC y la UP, el cual permite al 68HC11

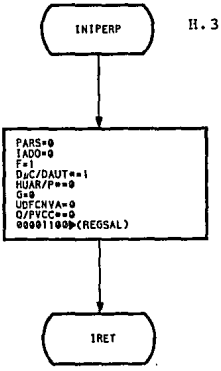
detectar cuando una UP ha quedado inicializada y además detectar cuando se encuentra la UP en estado estacionario (*HALT*), es decir, alineada correctamente para comenzar actividades de manera concurrente. Este protocolo, como ya se ha mencionado en la sección IV.7 consiste en liberar a la UP (desactivando su línea de RESET) y dejarla efectuar su inicialización habitual. Posteriormente se le obliga a ejecutar un programa tipo BATCH que carga el programa de aplicación; una de las primeras líneas contiene una escritura a uno de los puertos de la UDFC, con lo cual se indica que la UP corriente ha terminado su inicialización. Después ejecuta una instrucción *HALT* de la cual saldrá sólo cuando la UDFC desee arrancar a las tres UPs de manera sincronizada para operar concurrentemente. Al recibir la respuesta el μC procede a inicializar la siguiente UP hasta terminar con la tercera. Posteriormente, el μC genera un llamado de interrupción a las UPs instaladas, cargando previamente en el RA el tipo de tarea ordenada a las UPs. Las UPs, al recibir la interrupción acuden a la rutina de servicio, iniciando con la lectura del RA para conocer el tipo de tarea por realizar, que en este caso correspondería a un llamado a todos los procesadores para arranque síncrono, y como consecuencia abandonan el estado *HALT* y proceden a ejecutar el programa de aplicación de manera concurrente. A partir de este momento entran en operación los autómatas de detección de operaciones, el AMD y el ACDR.

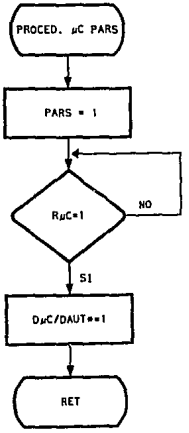
En los siguientes diagramas de flujo se encuentra la información detallada de estos algoritmos y de otros que integran los procedimientos necesarios para el control de la arquitectura TF.

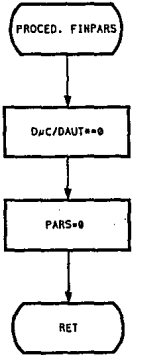

programa 68hc11
INICIO
HVCC=1
ARQUE=0
SI
670
NO
UDFC EN ESTADO REDUNDANTE POR PROGRAMACION (QUIZAS POR HARDWARE PODRIA SER MAESTRA)
BAND R/P=1
666
BIT R/P=0
SI
NO
LLAMA INIPERR
3SVCCIN=0
NO
SI
QUEDARSE EN STAND BY (WAIT). COMO REDUNDANTE EN CASO DE FALLA DE LA PRINCIPAL.
528
RUTINA DE INICIALIZACION DE PROCESADORES
BAND R/P=0
LLAMA INIC LCD
LLAMA INIPERP
3SVCCIN=0
NO
SI
DμC/DAUT=1
SELECTOR DE MUX
1AH▶(REG.ACCION)
CARGAR DATOS DEL REG DE EDO AL PTO
11101100▶ AXXXH
HACER LINEA
RST1=1 (PA3)
LINEA B=1
NO
SI
SALIDA F=0
SALIDA F=1
REGISTRO.EDO.
11111100 ▶ (AXXXH)
11101100 ▶ (AXXXH)
2AH ▶ (REG.ACCION)
RST2=1 (PA4)
LINEA B=1
NO
SI
F=0
F=1
10
10
REG. EDO.
11111100▶(AXXXH)
11110000▶(AXXXH)
3AH▶(R.ACC)
RST3=1(PA5)
LINEA B=1
NO
SI
F=0
F=1
REG. EDO.
11111100 (AXXXH)
11101100 (AXXXH)
528
528
ACTIVAR IADO=1 PARA ARRANCAR AUTOMATA DE LA UFC.
DμC/DAUT=0
ENVIAR DATO
41H▶(DXXXH)
ENVIAR DATO CUALQUIERA(0XXXH)PARA GENERAR ARRANQUE SINCRONO DE LOS TRES PROCESADORES
20


20
GENERA RETARDO
1 SEG.
DETECTA FUNCION
OPRIMIDA
HAB.RENGLON 1
00000011▶ (2800H)
TECLA
PRESIONADA
?
SI
30
NO
HAB.RENGLON 2
00000101▶ (2800H)
TECLA
PRESIONADA
?
SI
40
NO
NO
FIN
RETARDO
SI
DESPLEGAR PUNTO EN PAN-
TALLA ,Y QUE CADA VEZ
QUE SE PASE POR AQUI SE
GENERE UN CORRIMIENTO A
LA DERECHA EN PANTALLA
NO
BIT
ACTRE=1
SI
ESCRIBA AL REG.
DE ACCION.
45H ▶ (CRACC)
GENERAR INTIS
CON ARRSIN
00H ▶ (ARRSIN)

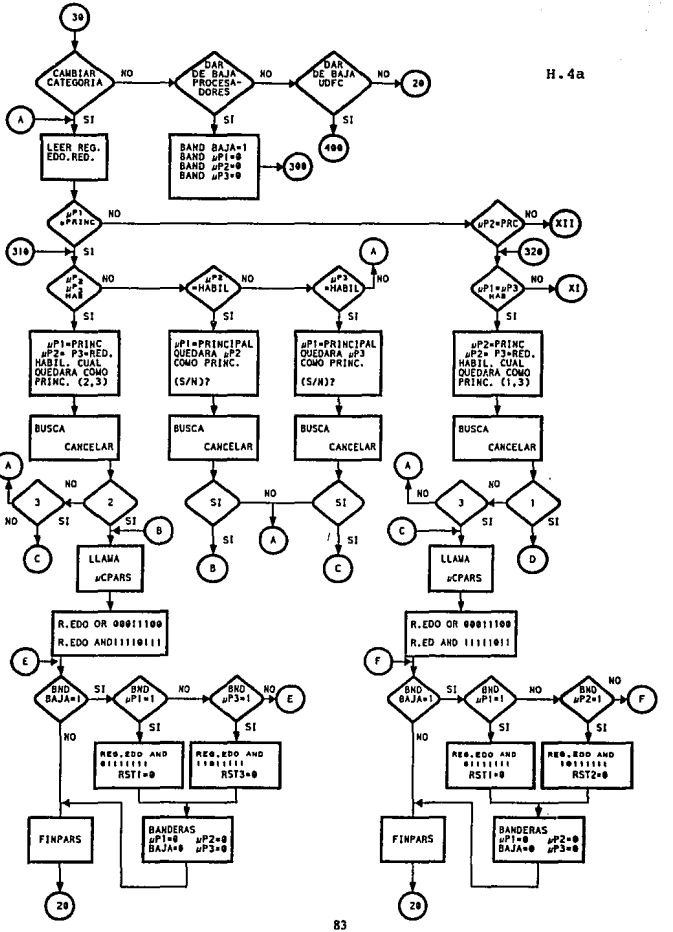

INIPERR
PARS=0
IADO=0
F=1
DμC/DAUT=*1
HUAR/P=*1
G=0
UDFCNVA=0
O/PVCC=*0
00001100▶ REGSAL
IRET


INIPERP
PARS=0
IADO=0
F=1
DμC/DAUT=*1
HUAR/P=*0
G=0
UDFCNVA=0
O/PVCC=*0
00001100▶(REGSAL)
IRET

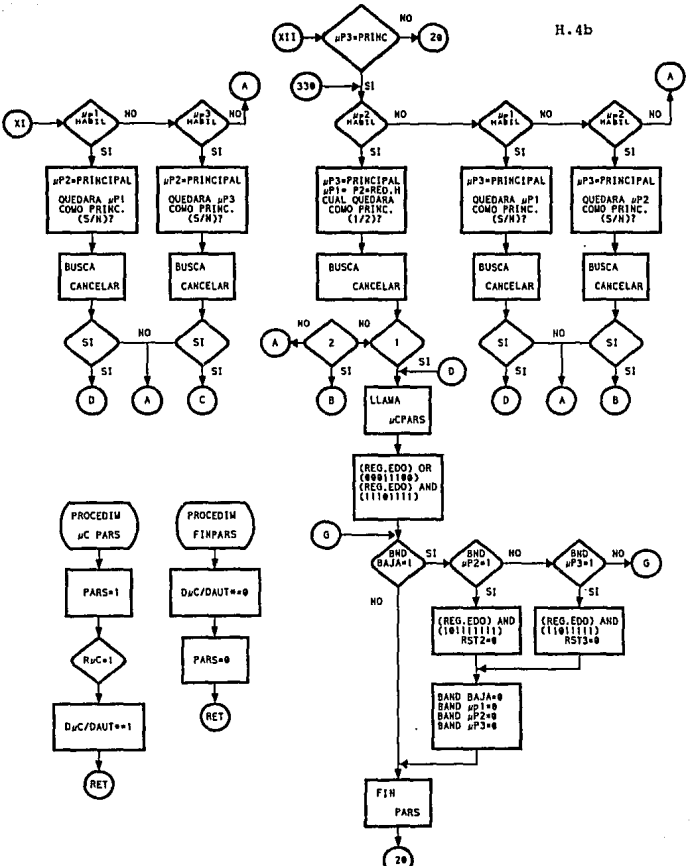

PROCED. μC PARS
PARS = 1
RμC=1
NO
SI
DμC/DAUT=*1
RET

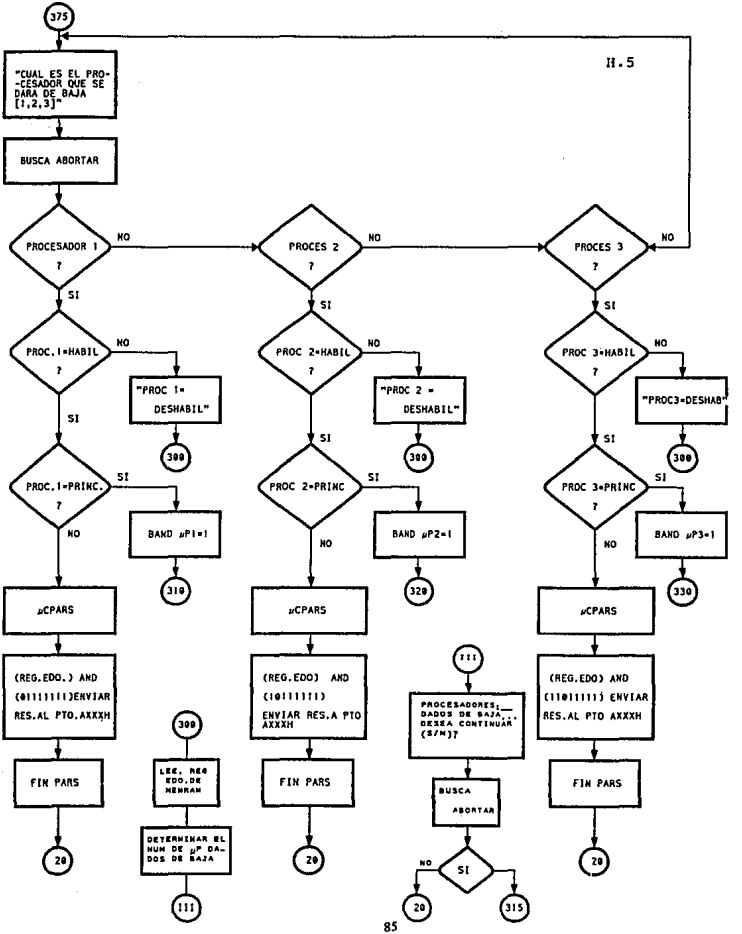

PROCED. FINPARS
DμC/DAUT=*0
PARS=0
RET

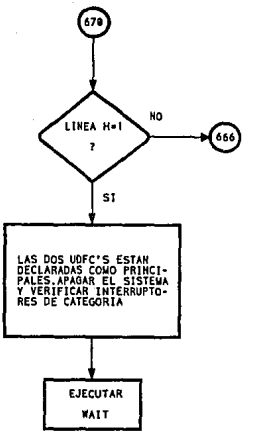
H.4a

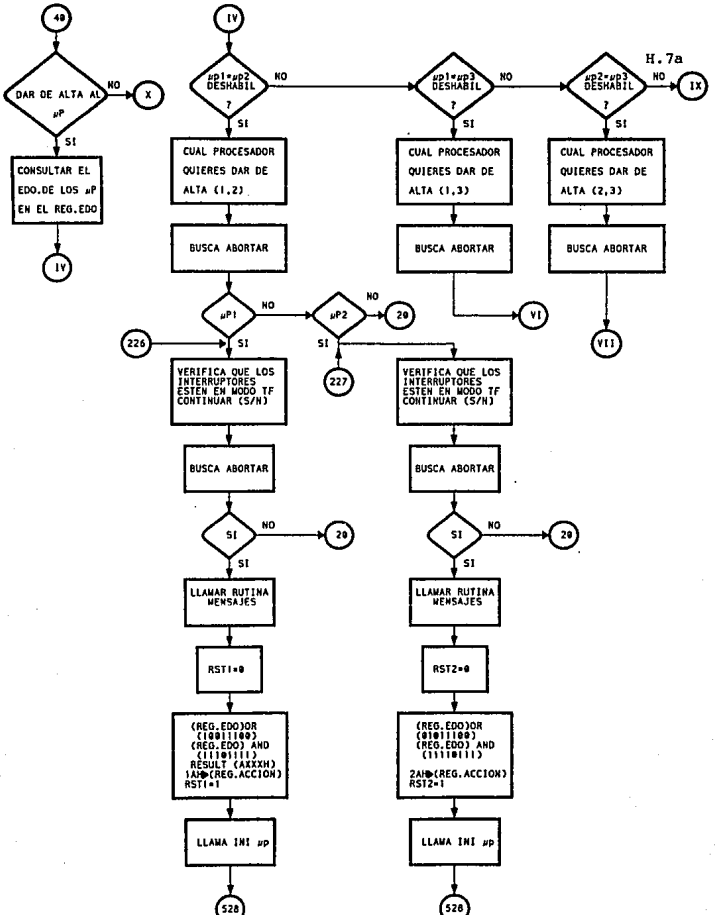

30
CAMBIAR CATEGORIA
NO
SI
DAR DE BAJA PROCESA-DORES
NO
SI
DAR DE BAJA UDFC
NO
SI
20
400
A
LEER REG. EDO.RED.
BAND BAJA=1
BAND μP1=0
BAND μP2=0
BAND μP3=0
300
μP1 =PRINC
NO
SI
310
μP2 =PRC
NO
XII
320
μP2 =PRINC
NO
SI
μP2 =HABIL
NO
SI
μP3 =HABIL
NO
SI
A
μP1 =μP3 HAB
NO
XI
SI
μP1=PRINC μP2= P3=RED. HABIL. CUAL QUEDARA COMO PRINC. (2,3)
μP1=PRINCIPAL QUEDARA μP2 COMO PRINC. (S/N)?
μP1=PRINCIPAL QUEDARA μP3 COMO PRINC. (S/N)?
μP2=PRINC μP2= P3=RED. HABIL. CUAL QUEDARA COMO PRINC. (1,3)
BUSCA
CANCELAR
3
2
1
SI
NO
B
C
D
LLAMA μCPARS
R.EDO OR 00011100
R.EDO AND11110111
R.ED AND 11111011
E
F
BND BAJA=1
BND μP1=1
BND μP2=1
BND μP3=1
REG.EDO AND 01111111 RST1=0
REG.EDO AND 10111111 RST2=0
REG.EDO AND 11011111 RST3=0
FINPARS
BANDERAS μP1=0 μP2=0 BAJA=0 μP3=0
20

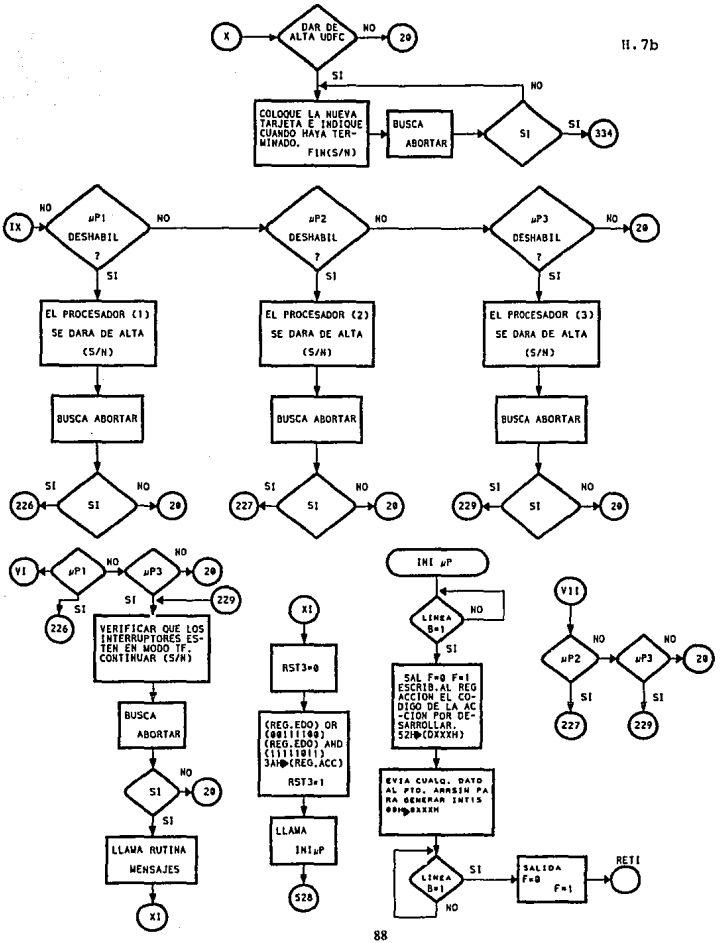
H.4b

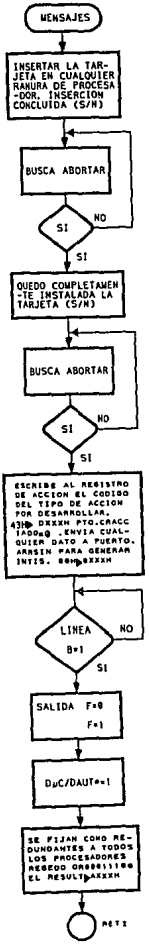

XII
μP3=PRINC
NO
20
SI
330
A
XI
μP1 HABIL
μP2 HABIL
μP2=PRINCIPAL
QUEDARA μP1 COMO PRINC. (S/N)?
QUEDARA μP3 COMO PRINC. (S/N)?
μP3=PRINCIPAL
μP1= P2=RED.H CUAL QUEDARA COMO PRINC. (1/2)?
BUSCA
CANCELAR
2
1
D
B
C
LLAMA
μCPARS
(REG.EDO) OR (00011100)
(REG.EDO) AND (11101111)
G
BND BAJA=1
BND μP2=1
BND μP3=1
(REG.EDO) AND (11011111) RST2=0
(REG.EDO) AND (11011111) RST3=0
BAND BAJA=0
BAND μP1=0
BAND μP2=0
BAND μP3=0
FIN PARS
PROCEDIM μC PARS
PARS=1
RμC=1
DμC/DAUT=1
RET
PROCEDIM FINPARS
DμC/DAUT=0
PARS=0

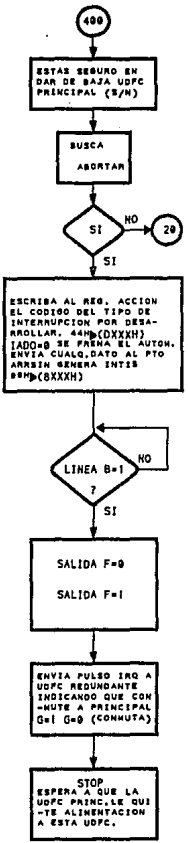

375
"CUAL ES EL PRO-CESADOR QUE SE DARA DE BAJA [1,2,3]"
BUSCA ABORTAR
PROCESADOR 1 ?
NO
SI
PROC.1=HABIL ?
NO
"PROC 1= DESHABIL"
300
SI
PROC.1=PRINC. ?
SI
BAND μP1=1
310
NO
μCPARS
(REG.EDO.) AND (01111111)ENVIAR RES.AL PTO.AXXXH
FIN PARS
20
PROCES 2 ?
NO
SI
PROC 2=HABIL ?
NO
"PROC 2 = DESHABIL"
300
SI
PROC 2=PRINC
SI
BAND μP2=1
320
NO
μCPARS
(REG.EDO) AND (10111111) ENVIAR RES.A PTO AXXXH
FIN PARS
20
PROCES 3 ?
NO
SI
PROC 3=HABIL ?
NO
"PROC3=DESHAB"
300
SI
PROC 3=PRINC ?
SI
BAND μP3=1
330
NO
μCPARS
(REG.EDO) AND (11011111) ENVIAR RES.AL PTO AXXXH
FIN PARS
20
300
LEE. REG EDO.DE MEMRAM
DETERMINAR EL NUM DE μP DA-DOS DE BAJA
111
111
PROCESADORES DADOS DE BAJA, DESEA CONTINUAR (S/N)?
BUSCA ABORTAR
NO
SI
20
315

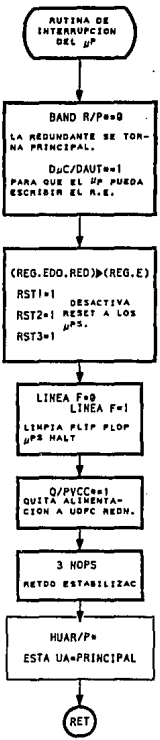

670
LINEA H=I
?
NO
666
SI
LAS DOS UDFC'S ESTAN DECLARADAS COMO PRINCIPALES.APAGAR EL SISTEMA Y VERIFICAR INTERRUPTORES DE CATEGORIA
EJECUTAR
WAIT


H.7a
40
DAR DE ALTA AL μP
NO
X
SI
CONSULTAR EL EDO.DE LOS μP EN EL REG.EDO
IV
IV
μP1+μP2 DESHABIL ?
NO
SI
CUAL PROCESADOR QUIERES DAR DE ALTA (1,2)
BUSCA ABORTAR
μP1
NO
μP2
NO
20
SI
SI
226
227
VERIFICA QUE LOS INTERRUPTORES ESTEN EN MODO TF CONTINUAR (S/N)
BUSCA ABORTAR
SI
NO
20
SI
LLAMAR RUTINA MENSAJES
RST1=0
(REG.EDO)OR (10011100) (REG.EDO) AND (11101111) RESULT (AXXXH) 1AH→(REG.ACCION) RST1=1
LLAMA INI μP
528
μP1+μP3 DESHABIL ?
NO
SI
CUAL PROCESADOR QUIERES DAR DE ALTA (1,3)
BUSCA ABORTAR
VI
VERIFICA QUE LOS INTERRUPTORES ESTEN EN MODO TF CONTINUAR (S/N)
BUSCA ABORTAR
SI
NO
20
SI
LLAMAR RUTINA MENSAJES
RST2=0
(REG.EDO)OR (01011100) (REG.EDO) AND (11110111) 2AH→(REG.ACCION) RST2=1
LLAMA INI μP
526
μP2+μP3 DESHABIL ?
NO
IX
SI
CUAL PROCESADOR QUIERES DAR DE ALTA (2,3)
BUSCA ABORTAR
VII


X
DAR DE ALTA UDFC
NO
20
SI
COLOQUE LA NUEVA TARJETA E INDIQUE CUANDO HAYA TERMINADO. FIN(S/N)
BUSCA ABORTAR
NO
SI
SI
334
IX
NO
μP1 DESHABIL ?
NO
μP2 DESHABIL ?
NO
μP3 DESHABIL ?
NO
20
SI
SI
SI
EL PROCESADOR (1) SE DARA DE ALTA (S/N)
EL PROCESADOR (2) SE DARA DE ALTA (S/N)
EL PROCESADOR (3) SE DARA DE ALTA (S/N)
BUSCA ABORTAR
BUSCA ABORTAR
BUSCA ABORTAR
SI
226
SI
NO
20
SI
227
SI
NO
20
SI
229
SI
NO
20
VI
μP1
NO
μP3
NO
20
SI
226
SI
229
VERIFICAR QUE LOS INTERRUPTORES ESTEN EN MODO TF. CONTINUAR (S/N)
BUSCA ABORTAR
SI
NO
20
SI
LLAMA RUTINA MENSAJES
XI
XI
RST3=0
(REG.EDO) OR (00111100) (REG.EDO) AND (11111011) 3A1►(REG.ACC)
RST3=1
LLAMA INIμP
528
INI μP
LINEA B=1
NO
SI
SAL F=0 F=1 ESCRIB.AL REG ACCION EL CODIGO DE LA ACCION POR DESARROLLAR. 521►(OXXXH)
ENVIA CUALQ. DATO AL PTO. ARBSIN PARA GENERAR INTIS 0D1►0XXXH
LINEA B=1
SI
NO
SALIDA F=0 F=1
RETI
VII
μP2
NO
μP3
NO
20
SI
227
SI
229

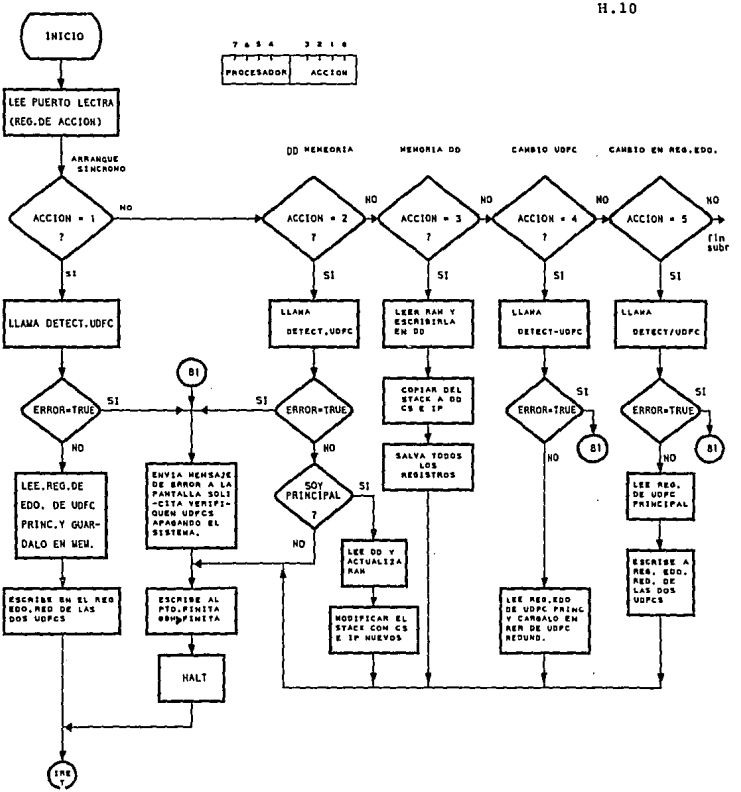

MENSAJES
INSERTAR LA TARJETA EN CUALQUIER RANURA DE PROCESADOR. INSERCION CONCLUIDA (S/N)
BUSCA ABORTAR
SI
NO
SI
QUEDO COMPLETAMENTE INSTALADA LA TARJETA (S/N)
BUSCA ABORTAR
SI
NO
SI
ESCRIBE AL REGISTRO DE ACCION EL CODIGO DEL TIPO DE ACCION POR DESARROLLAR.
43H DXXH PTO.CRACCIADO=0 .ENVIA CUALQUIER DATO A PUERTO. ARRSIN PARA GENERAR INTIS. 80H=0XXXH
LINEA
B=1
NO
SI
SALIDA F=0
F=1
DμC/DAUT=1
SE FIJAN COMO REDUNDANTES A TODOS LOS PROCESADORES REGEDO OR8011100 EL RESULTADXXXH
RETI
33H
UDFCNVA=1
QVCC/PVCC=0
ESPERAR 3 SEG
QVCC/PVCC=1
UDFCNVA=0
2B

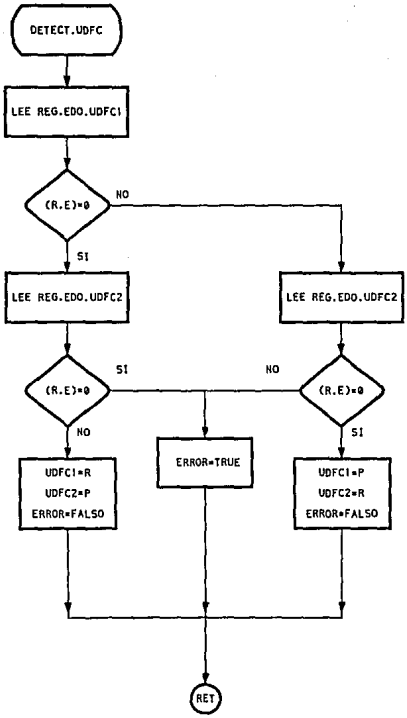

400
ESTAS SEGURO EN DAR DE BAJA UDFC PRINCIPAL (S/N)
BUSCA
ABORTAR
SI
NO
20
SI
ESCRIBA AL REG. ACCION EL CODIGO DEL TIPO DE INTERRUPCION POR DESARROLLAR. 44H▶(DXXXH)
IADO=0 SE FRENA EL AUTOM. ENVIA CUALQ.DATO AL PTO ARRBIN GENERA INTIS
88H▶(8XXXH)
LINEA B=1 ?
NO
SI
SALIDA F=0
SALIDA F=1
ENVIA PULSO IRQ A UDFC REDUNDANTE INDICANDO QUE CON-MUTE A PRINCIPAL G=1 G=0 (CONMUTA)
STOP
ESPERA A QUE LA UDFC PRINC.LE QUI-TE ALIMENTACION A ESTA UDFC.

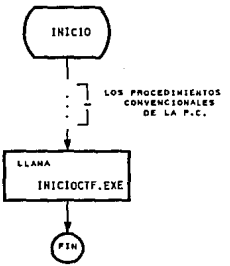

RUTINA DE INTERRUPCION DEL μP
BAND R/P=0
LA REDUNDANTE SE TORNA PRINCIPAL.
DμC/DAUT=1
PARA QUE EL μP PUEDA ESCRIBIR EL R.E.
(REG.EDO.RED)▶(REG.E)
RST1=1
RST2=1
RST3=1
DESACTIVA RESET A LOS μPS.
LINEA F=0
LINEA F=1
LIMPIA FLIP FLOP μPS HALT
Q/PVCC=1
QUITA ALIMENTACION A UDFC REDN.
3 NOPS
RETDO ESTABILIZAC
HUAR/P=
ESTA UA=PRINCIPAL
RET

5.3 Programación de Interrupciones en UPs

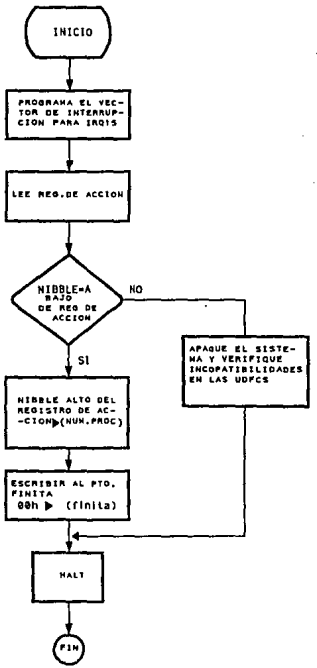
Como se ha mencionado y como se puede observar en los diagramas de flujo anteriores, existen protocolos de comunicación entre UDFC principal y UPs, basados principalmente en el uso de interrupciones. En los siguientes diagramas de flujo se esquematizan las diferentes rutinas de comunicación diseñadas para UPs, las cuales se complementan con información de la sección IV.7.


INICIO
7 6 5 4
3 2 1 0
PROCESADOR
ACCION
LEE PUERTO LECTRA (REG.DE ACCION)
ARRANQUE SINCRONO
DD MEMORIA
MEMORIA DD
CAMBIO UDFC
CAMBIO EN REG.EDO.
ACCION = 1 ?
NO
ACCION = 2 ?
NO
ACCION = 3 ?
NO
ACCION = 4 ?
NO
ACCION = 5 ?
NO
fin subr
SI
LLAMA DETECT.UDFC
LLAMA DETECT,UDFC
LEER RAM Y ESCRIBIRLA EN DD
LLAMA DETECT-UDFC
LLAMA DETECT/UDFC
ERROR=TRUE
SI
NO
B1
COPIAR DEL STACK A DD CS E IP
SALVA TODOS LOS REGISTROS
LEE.REG.DE EDO. DE UDFC PRINC.Y GUARDALO EN MEM.
ENVIA MENSAJE DE ERROR A LA PANTALLA SOLICITA VERIFIQUEN UDFCS APAGANDO EL SISTEMA.
SOY PRINCIPAL ?
LEE DD Y ACTUALIZA RAM
MODIFICAN EL STACK CON CS E IP NUEVOS
LEE REG.EDO DE UDFC PRINC Y CARGALO EN REG DE UDFC REDUND.
LEE REG. DE UDFC PRINCIPAL
ESCRIBE EN EL REG EDO.REG DE LAS DOS UDFCS
ESCRIBE AL PTO.FINITA 89H FINITA
ESCRIBE A REG. EDO. REG. DE LAS DOS UDFCS
HALT
INT


DETECT.UDFC
LEE REG.EDO.UDFC1
(R.E)=0
NO
SI
LEE REG.EDO.UDFC2
LEE REG.EDO.UDFC2
(R.E)=0
SI
NO
(R.E)=0
NO
SI
UDFC1=R
UDFC2=P
ERROR=FALSO
ERROR=TRUE
UDFC1=P
UDFC2=R
ERROR=FALSO
RET



NOTA : AL ENCENDER EL SISTEMA, AL USUARIO SE LE MANDA AL "MS DOS"



SEIS

Conclusiones y Recomendaciones

En los capítulos pasados se ha detallado el diseño, el desarrollo y la construcción de los circuitos impresos centrales, de una computadora autoconfigurable para procesamiento concurrente. Debido a la complejidad del proyecto y a las limitaciones de tiempo para desarrollar una tesis de licenciatura, al trabajo realizado sólo le faltó llegar a la etapa de construcción y de validación de la electrónica desarrollada. Sin embrago, debe subrayarse la gran cantidad de trabajo elaborado y los resultados obtenidos hasta este momento, los cuales definen puntos cruciales en el desarrollo del proyecto citado. En el Instituto de Ingeniería el trabajo continua y serán otras tesis, junto con otros logros, los que permitirán cristalizar esta laboriosa arquitectura, la cual se ha gestado en el trabajo aquí expuesto.

En este capítulo se señalarán las conclusiones que emanan de la experiencia acumulada y los resultados obtenidos en la presente tesis. Así mismo y con el objeto de vincular el trabajo desarrollado con el trabajo por continuar, se mencionan diversas recomendaciones que a juicio del autor constituyen partes sustanciales en la integración de la CTF.

6.2 Conclusiones

De la tesis desarrollada se presentan las siguientes conclusiones:
a) Se diseñó la electrónica así como los respectivos circuitos impresos para integrar las dos tarjetas objetivo fundamentales de la arquitectura de una CTF. La primera es la Unidad de Procesamiento (UP), de la cual la CTF integrará tres tarjetas similares (para formar un sistema concurrente triplex o RMT), en tanto que la segunda constituye la Unidad de Detección de Fallas y Control (UDFC), de las cuales la CTF utiliza dos tarjetas iguales (para formar un sistema duplex). El trabajo constituye una de las partes centrales del proyecto para construir una computadora con altos índices de confiabilidad orientada a aplicaciones de alto riesgo.

b) Las tarjetas construidas satisfacen los requerimientos del proyecto de construcción de una CTF desarrollada en el Instituto de Ingeniería. Entre ellos podemos citar la modularidad, el confinamiento de fallas, la posibilidad de realizar mantenimiento preventivo y correctivo de las tarjetas objetivo diseñadas (sin tener que desenergizar el equipo y sin detener la ejecución de programas de aplicación), el poder trabajar con varias tarjetas objetivo al mismo tiempo, la incorporación de los microprocesadores de Intel de mayor uso nacional, la amplia utilización de dispositivos lógicos programables, etcétera.

c) Las tarjetas objetivo diseñadas forman parte de una arquitectura de procesamiento concurrente controlada principalmente mediante lógica electrónica. Los autómatas de control se integraron de forma distribuida en las tarjetas para desarrollar tareas de detección de fallas, ubicación de las mismas y la reconfiguración automática, cuando ésta sea requerida.

d) Las características de modularidad de las tarjetas diseñadas facilitan las labores de mantenimiento del sistema TF.

e) Los circuitos impresos de ambas tarjetas objetivo se realizaron en dos caras únicamente, lo cual complicó enormemente el trazo de líneas de conexión entre dispositivos, debido a la gran cantidad de pistas integradas; sin embargo, se tendrán repercusiones favorables en el costo de producción del equipo y además hará posible el mantenimiento correctivo y por tanto el reuso de tarjetas después de aplicarles mantenimiento.

f) La lógica de control utilizada en las tarjetas permite desactivarlas remotamente desde otras tarjetas (tanto UPs como UDFCs) para que en caso de que se presenten anomalías operativas, las tarjetas redundantes procedan a tomar el mando respectivo.

g) Se diseñaron autómatas optimizados que con un número mínimo de estados, permiten controlar dinámicamente y en tiempo real la arquitectura TF, ellos son: autómata de detección de operaciones (instalados en cada UP), autómata de multiplexaje de datos (AMD) y autómata de comparación de datos y reconfiguración (ACDR), estos últimos integrados en cada UDFC diseñados para operar a 16 MHz.

h) Como medio de detección de fallas se incluyeron comparadores digitales, cuyos resultados se utilizan para diagnosticar fallas permanentes y fallas transitorias en la arquitectura, además de permitir automáticamente su autodiagnóstico.

i) Las UPs emplean el microprocesador 80386SX e integran además al juego de CIs TACT8300 de Texas Instruments, 1Mb de memoria dinámica expandible a 4, 8 y 16 Mb, EPROM BIOS, controlador de teclado y un autómata de detección de operaciones. La tarjeta incluye a cuatro CIs de montado superficial, todos ellos VLSI, de los cuales el más grande tiene 208 terminales.

j) El diseño de las UPs contempla la posibilidad de utilizarlas para integrar una CTF, o bien, para usarlas como procesadores independientes. Además se previó la eventualidad de poder sustituir el EPROM BIOS de 64Kb por uno de hasta 256Kb. Este detalle se podrá explotar, junto con un paquete recién adquirido por nuestro laboratorio, para generar tarjetas de procesamiento autosostenidas (*stand-alone*). Con esto se amplía el espectro de aplicaciones del diseño desarrollado.

k) En cuanto a la UDFC, su lógica de control se dividió en dos partes, la primera que contiene autómatas de alta velocidad para el control dinámico de la arquitectura concurrente; la segunda, constituida por un μC 68HC11, que tiene como funciones principales la inicialización de la arquitectura, el envío de mensajes al usuario en caso de fallas y la asistencia interactiva con el usuario durante labores de mantenimiento preventivo, correctivo, o bien, con fines de demostración.

l) Se proyectó y diseñó un ducto propietario para propósitos de detección y diagnóstico de fallas, así como para apoyar las tareas de reconfiguración en la arquitectura diseñada; este ducto está compuesto por dos conectores, uno de 62 líneas y otro

principalmente para propósitos de expansión que contiene 36 líneas adicionales. Estas últimas podrán utilizarse en el momento en que sea requerido.

m) Se utilizó un programa de diseño asistido por computadora para elaborar los circuitos impresos, contribuyendo a generar un producto de alta calidad. Durante el desarrollo de los impresos quedó de manifiesto la necesidad de contar con un paquete aún más versátil, principalmente en lo que se refiere a las salidas de archivos para producir impresiones en película de alta resolución. Para resolver este problema se ha gestionado la adquisición de la nueva versión de este paquete.

n) Las experiencias obtenidas en el desarrollo del presente trabajo se comparten y seguirán difundiéndose hacia los nuevos colabores del Laboratorio de Automatización del II.

o) Durante el desarrollo de la tesis y con el objeto de hacer uso de algunos recursos de otras instituciones, se fomentaron convenios con dependencias cómo : el IPN, Motorola, Texas Instruments, Intel, etc.

p) En el desarrollo del trabajo expuesto en esta tesis intervinieron ideas y apoyo de los diversos colaboradores del proyecto, y de la misma manera algunos temas no incluidos en esta tesis, constituyeron tópicos en los que el presente autor tuvo colaboración, obteniendo también otras experiencias, lo cual refleja el trabajo en equipo que es común denominador en este proyecto.

6.3 Recomendaciones

Para continuar el trabajo y mejorar las herramientas de desarrollo del Laboratorio de Automatización se pueden mencionar las siguientes recomendaciones:

a) Para reducir el tamaño de las tarjetas diseñadas se puede realizar una nueva versión de circuito impreso de tipo multicapa, pudiendo obtener tarjetas de dimensiones mínimas; esto repercutiría en las dimensiones finales de la computadora en desarrollo. Como es de imaginarse, el costo de desarrollo y construcción de esta nueva versión de tarjeta representa más que nada compromisos económicos.

b) En relación al punto anterior, la reducción del tamaño de las tarjetas sería factible si se utilizara algún tipo de ducto compatible con el empleado en este caso pero con mayor capacidad de líneas, por lo cual se hace recomendable el uso del ducto y conectores tipo EISA. Esta opción también podría usarse en el ducto propietario.

Debe subrayarse que esta recomendación es válida en caso de que se deseen reducir las dimensiones de las tarjetas y de que se requiera utilizar la CTF en aplicaciones críticas, que sean sensibles al volumen y al peso del equipo, como es el caso de aplicaciones móviles en general.

c) Para agilizar y hacer más eficiente el diseño de circuitos impresos, es altamente recomendable contar con un paquete CAD para estaciones de trabajo, como las que actualmente tiene el Laboratorio de Automatización. Esta recomendación se hace en virtud de que el cálculo intensivo requerido por TANGO, en especial para el circuito impreso de UPs hace lentas las operaciones de las PCs aun tratándose de computadoras 486. Además, el paquete CAD debería contar con un módulo de trazado automático de líneas, capaz de realizar al 100% el ruteo de las conexiones requeridas, ya que el dibujar manualmente las pistas es una tarea tediosa que requiere mucho tiempo y esfuerzo.

d) Respecto al juego de CIs (TACT83000) utilizado en las UPs, sería muy recomendable sustituirlo por una versión más reciente, lo cual redundaría en sustituir tres CIs VLSI de montado superficial, por uno solo.

e) En lo referente a los dispositivos lógicos programables es posible migrar de los GALs adoptados en el diseño, hacia nuevos PLDs de mayor capacidad. Con esto se obtendrían reducciones adecuadas en cuanto al número de componentes y número de líneas trazadas en las tarjetas UPs y UDFCs.

f) Por el lado de la UDFC es muy recomendable sustituir el 68HC11 por un modelo que incluya una EEPROM, evitando el uso de una memoria externa y de un *latch* para mantener direcciones. Con esta nueva versión, sería posible incluso, tener un 68HC11 redundante en la tarjeta, cuyo principio de detección de fallas podría basarse en protocolos de comunicación vía puerto serie. Con esta adición, la UDFC quedaría respaldada en todos sus componentes y con la posibilidad de detectar fallas en todos ellos.

g) Debido a la gran cantidad de pistas trazadas y a la complejidad del diseño, el circuito impreso de las UPs es sumamente difícil de fabricar en el país, debido a las limitaciones tecnológicas de los procesos de producción de la industria nacional (sus límites se presentan a partir de separaciones entre pistas de 8 a 10 milésimas de pulgada, punto en que comienzan a aparecer cortos circuitos aleatorios en sus tarjetas producidas).

Por esta razón, se sugiere que de ser posible esta tarjeta en particular se envíe a fabricar al extranjero, debido a que bastantes de sus separaciones entre líneas son de 8 milésimas de pulgada.

h) En cuanto a los autómatas AMD y ACDR, una vez que se obtengan pruebas experimentales exitosas, sería importante efectuar pruebas de operación a una frecuencia de 32 MHz, para mejorar el desempeño del sistema.

APÉNDICE A

Filosofías de Diseño y Técnicas de Tolerancia a Fallas

A.1 Filosofías de Diseño para Combatir Fallas

Existen tres metodologías básicas para mejorar o mantener el buen desempeño de un sistema cuyo ambiente no está exento de fallas: la prevención de fallas, el enmascaramiento de fallas y la tolerancia a fallas.

- **Prevención de Fallas.** Esta técnica se usa para prevenir fallas en primer término. Ésta puede incluir cuestiones como: revisión de diseños, blindaje, verificación y otros métodos de control de calidad de los componentes que integren el sistema.

- **Enmascaramiento de Fallas.** Este proceso evita la introducción de fallas en la estructura de la información de un sistema. Por ejemplo, la corrección de errores en datos de memorias antes de ser usados, es una manera de eludir el uso de datos erróneos en el sistema. Otro ejemplo de enmascaramiento de fallas son los sistemas de voteo (votación) mayoritario. En un sistema con tres módulos de procesamiento, si al menos dos de ellos coinciden en sus resultados, esto determina la decisión final del resultado válido.

- **Tolerancia a Fallas.** Este tipo de diseño se puede lograr mediante varias formas. En efecto, el enmascaramiento de fallas es uno de los criterios, otro es

el detectar y localizar la falla ocurrida para reconfigurar el sistema y así remover el módulo o componente dañado. El proceso de reconfiguración elimina del sistema aquellas entidades con falla y lleva al mismo a un estado de continuidad operativa. Cuando se usa la reconfiguración hay algunos conceptos que deben tomarse en consideración, tales como:

- **Detección de Fallas.** Se trata de un proceso para determinar que ha ocurrido una falla en el sistema.

- **Localización de la Falla.** Sirve para determinar el sitio donde ocurrió la falla, para así aplicar los procedimientos de recuperación adecuados.

- **Confinamiento de Fallas.** Cuando ha ocurrido alguna falla, es preciso aislarla del resto del sistema para prevenir su propagación a través del mismo. El aislamiento de fallas es un elemento indispensable en todos los STF.

- **Recuperación de Fallas.** Este proceso se refiere a permanecer en un estado operacional o a recuperar un estado operante aun ante la ocurrencia de fallas.

A.2 Técnicas de Diseño en Tolerancia a Fallas

Todas las filosofías de diseño requieren el uso de elementos redundantes para lograr su cometido. Los primeros diseños de STF tomaban el concepto de redundancia únicamente a nivel físico, esto es, se empleaban replicas de componentes o módulos de *hardware* para lograr los atributos de tolerancia a fallas. En nuestros días, se tiene un mejor entendimiento del concepto de redundancia y de los diferentes tipos de ésta. Una redundancia es simplemente la adición de información, recursos, o tiempo, necesarios para la operación normal del sistema. Existen cuatro tipos de redundancias: redundancia por *hardware*, por *software*, redundancia de información y redundancia en el tiempo.

En seguida se ampliarán detalles de las redundancias por *hardware* y por *software*, puesto que dichos conceptos son de mayor trascendencia para los fines de este

trabajo y permitirán tener una mejor comprensión del mismo. Con el propósito de simplificar las explicaciones, sólo se dan breves definiciones de la redundancia por tiempo y la redundancia de información, para profundizar en estas técnicas ver [Johnson W. 1989].

La redundancia por tiempo emplea tiempo extra en la ejecución de alguna tarea con el objeto de cumplir con las labores de detección y tolerancia a fallas.

La redundancia de información implica la incorporación de información complementaria, excediendo la necesaria para desarrollar una determinada función; por ejemplo, el uso de códigos de detección de errores es una forma de redundancia de información.

A.2.1 Redundancia por *Hardware*

Este tipo de redundancia se refiere a la adición de componentes complementarios, con el propósito de ayudar en las tareas de detectar o tolerar fallas. Esta repetición física de componentes es tal vez la forma más común de redundancia que se usa en los sistemas digitales. Resulta una práctica usual, puesto que los costos de repetición se abaten a cierto grado, con la reducción de precios de los semiconductores.

En el caso de la redundancia por *hardware*, existen tres formas básicas de llevarla a cabo: pasiva, activa e híbrida. Las técnicas pasivas se emplean para enmascarar fallas ocultando así la ocurrencia de fallas. El enmascaramiento consiste en tener N módulos idénticos operando concurrentemente y comparando sus salidas para detectar algún módulo con falla y para no permitir que las salidas de éste lleguen al exterior. Esta aproximación se usa en aquellos sistemas donde la participación de operadores es indeseable para sustituir partes en caso de presentarse la primera falla.

Las técnicas activas (o dinámicas) logran el atributo tolerante a fallas teniendo dos módulos iguales cuyas salidas se comparan continuamente y al haber discrepancias en los datos generados se ejecutan rutinas de diagnóstico para ubicar la falla y remover dicho módulo del sistema. Esto es, se hace uso intensivo de las técnicas de detección de fallas, localización de las mismas y recuperación, las cuales ya fueron mencionadas.

La combinación de los procedimientos pasivos y activos deriva en la obtención de criterios híbridos, donde el enmascaramiento de fallas, se usa para prevenir la generación de resultados erróneos; mientras que la detección, localización y recuperación de fallas actuarán para confinar (aislar) la parte dañada y en la reconfiguración del sistema.

Existe una serie de arquitecturas para cada una de las técnicas mencionadas, las cuales serán descritas en breve.

A.2.1.1 Redundancia Pasiva por *Hardware*

Esta técnica se basa en mecanismos de enmascaramiento de fallas. La mayoría de los enfoques pasivos se desarrollan alrededor del concepto de voteo mayoritario. Como se mencionó anteriormente, los criterios pasivos efectúan la tolerancia a fallas sin requerir la ejecución de rutinas de detección de fallas, ni la reconfiguración del sistema; el diseño pasivo tolera las fallas de manera inherente.

Dentro de esta categoría se tienen dos arquitecturas, la redundancia modular triple y la redundancia modular N, las cuales se describen a continuación.

Redundancia Modular Triple

La forma más común de redundancia pasiva por *hardware* es la llamada Redundancia Modular Triple (RMT). El concepto básico (mostrado en la Figura A.1) es triplicar los módulos electrónicos (ya sean procesadores, memoria, etc.), para efectuar un voteo mayoritario que determine la salida del sistema. En caso de que algún módulo presente una falla, ella será enmascarada por los dos módulos restantes en buen estado. Cabe mencionar, que es posible aplicar esta misma técnica en programación, donde se tienen tres versiones diferentes de programas que efectúan el mismo proceso, esto es con la finalidad de proteger al sistema en caso de que alguno de los módulos de programación falle, más adelante se ampliará la información sobre la redundancia por programación.

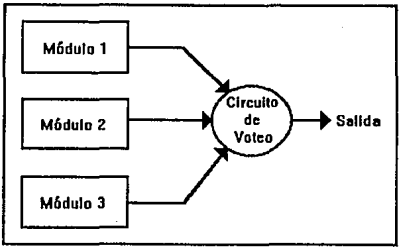


Figura A.1 Esquema de Redundancia Modular Triple (RMT).

El punto débil en esta arquitectura es el circuito de voteo, pues de presentarse una falla en él, el sistema completo se avería. Esto quiere decir que, la confiabilidad del sistema depende de la calidad del circuito de voteo. Cualquier componente dentro de un sistema que conduzca al sistema entero a un estado de avería, se le conoce como **punto único de avería.** Se pueden usar diferentes técnicas para corregir los efectos por la falla del circuito de voteo. Una alternativa es triplicar el circuito de voteo y proporcionar tres salidas diferentes, como se muestra en la Figura A.2. En dicha figura, se tienen tres módulos funcionalmente independientes que realizan idénticas operaciones, cuyos resultados se insertan en los circuitos de voteo generando tres resultados. Cada resultado es correcto mientras no haya un solo módulo con falla.

Redundancia Modular N

Se trata de la generalización del criterio RMT. Basado en el mismo principio de voteo mayoritario, se tienen N módulos idénticos, donde N se recomienda sea un número impar. El concepto de la Redundancia Modular N (RMN) se muestra en la Figura A.3. La ventaja de usar N módulos, donde N>3, radica en el hecho de que es factible tolerar más módulos fallados. Por ejemplo, si N=5, el esquema RMN es capaz de tolerar hasta dos módulos con fallas. Esto es importante remarcarlo, porque existen muchas aplicaciones criticas donde se requiere tolerar dos fallas a la vez, pues sus índices de confiabilidad así lo exigen.

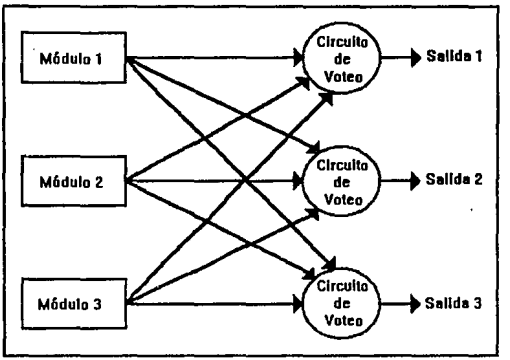


Figura A.2 Esquema RMT con los Circuitos de Voteo triplicados

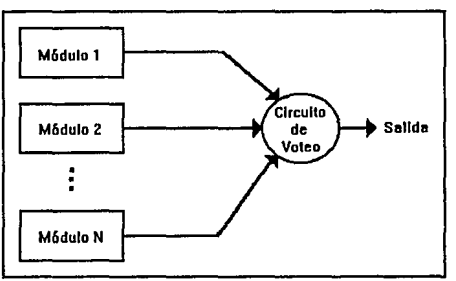


Figura A.3 Esquema de Redundancia Modular N (RMN)

A.2.1.2 Redundancia Activa por *Hardware*

Las técnicas de redundancia activa logran el atributo de tolerancia a fallas por medio de la detección, localización y recuperación de fallas. Debido a que las fallas en los

diseños se detectan basándose en los errores producidos, es a veces más adecuado el uso de la terminología: detección, localización y restablecimiento de errores. Este esquema no es capaz de prevenir la presencia de errores que conducirán a averías dentro del sistema; pues no tiene la propiedad de enmascaramiento de errores. En consecuencia, las aproximaciones activas se usan comúnmente en aplicaciones que temporalmente pueden tolerar resultados erróneos, mientras el sistema se reconfigura y recupera su estado operativo en períodos de tiempo razonables. Los sistemas de satélites son un buen ejemplo de aplicación de este tipo de redundancias activas.

En cuanto a las arquitecturas dinámicas los esquemas más conocidos son: la duplicación con comparación, el repuesto de apoyo, la técnica de par y repuesto y los vigías de tiempo, todas ellas se explican en los próximos renglones.

Duplicación con Comparación

El primer esquema de redundancia dinámica es el de duplicación con comparación mostrado en la Figura A.4. El concepto básico de la duplicación con comparación se refiere a contar con dos módulos de *hardware* idénticos, ejecutando en paralelo los mismos cálculos y comparando los resultados de dichas operaciones. En caso de tener discrepancias, se genera un mensaje de error. Este esquema es capaz únicamente de detectar la presencia de fallas, pero no las tolera, porque no cuenta con un método que determine cuál de los dos módulos tiene la falla. Sin embargo, este planteamiento es de gran utilidad como técnica de detección de fallas en los sistemas basados en redundancias activas.

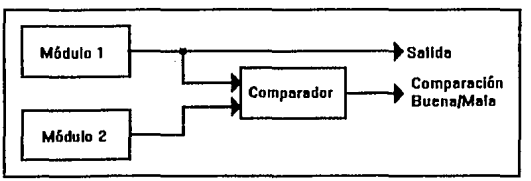


Figura A.4 Esquema de la técnica de Duplicación con Comparación.

Repuesto de Apoyo

Una segunda forma de redundancia activa es la llamada repuesto de apoyo ilustrada en la Figura A.5. Esta técnica contempla un módulo en operación y uno o más módulos que sirvan como repuestos o refacciones. El funcionamiento de este procedimiento requiere el uso de métodos de detección y localización de fallas, para conocer el módulo que ha fallado, con la finalidad de removerlo del sistema y reemplazarlo con uno de los repuestos. La operación de reconfiguración se puede ver conceptualmente como un conmutador cuya salida se toma de uno, y sólo uno, de los módulos que conforman el sistema. El conmutador examina los reportes de los circuitos de detección de fallas asociados a cada módulo para decidir cual salida de los módulos emplear.

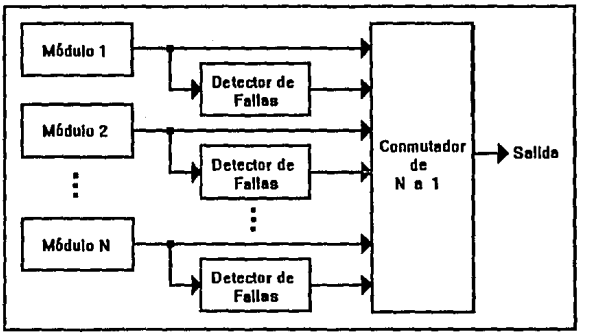


Figura A.5 Técnica de Repuesto de Apoyo.

Técnica de Par y Repuesto

La combinación de las dos técnicas mencionadas anteriormente, la de duplicación con comparación y la de repuesto de apoyo, son los elementos constitutivos del planteamiento mostrado en la Figura A.6. En esencia, se tienen dos módulos trabajando todo el tiempo y sus resultados se comparan para lograr la detección de fallas requerida en la técnica de repuesto de apoyo. El circuito de comparación genera

la señal de error que inicia el proceso de reconfiguración para remover los módulos con falla y reemplazarlos con los repuestos existentes.

La reconfiguración se puede ver simplemente como un conmutador, ver Figura A.6, que acepta las salidas de todos los módulos instalados en el sistema junto con los reportes de errores y proporciona al comparador las salidas de los dos módulos activos en ese momento, una de estas salidas es la salida del sistema. Mientras las dos salidas elegidas sean iguales, no se usan las refacciones. Pero cuando hay diferencias, el conmutador verifica el reporte de errores para identificar al módulo que falló y después selecciona un módulo suplente.

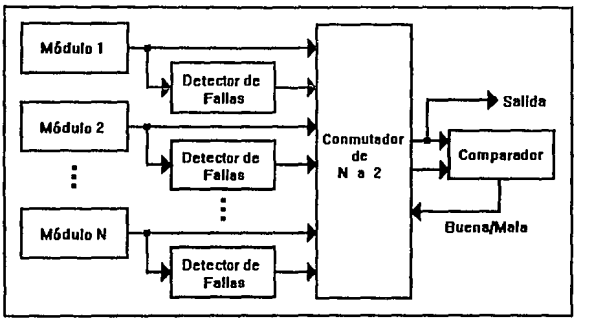


Figura A.6 Esquema de la Técnica de Par y Repuesto

Vigías de Tiempo

Una técnica ampliamente usada para detectar fallas en un sistema, es precisamente la de los vigías de tiempo. Estos vigías se consideran en la categoría de las redundancias activas debido a que se solicita la participación del sistema para determinar estados sin falla. La función de un guardián de tiempo es considerar la duración de una acción como un indicador de falla. Esto implica que el vigía debe ser reinicializado repetitivamente, pues en caso de que el sistema dejara de hacerlo, el circuito vigía indica la presencia de una falla.

Cabe señalar que esta técnica de redundancia puede ser usada tanto en sistemas electrónicos como en programación.

A.2.1.3 Redundancia Híbrida por *Hardware*

La idea básica de la redundancia híbrida es la de incorporar las bondades de las técnicas activas y pasivas. El enmascaramiento de fallas se aprovecha para evitar que el sistema produzca resultados erróneos; mientras que la detección, ubicación y reemplazo de fallas son útiles para las operaciones de reconfiguración, en caso de presentarse alguna falla. Las técnicas de redundancia híbrida se utilizan en aplicaciones que requieren alta integridad en las operaciones.

Dentro de las técnicas híbridas se pueden mencionar la redundancia modular N con repuestos, la redundancia de autodepuración, la redundancia modular selectiva y la arquitectura triplex-duplex, que en breve se describirán.

Redundancia Modular N con Repuestos

El paradigma de esta técnica es contar con un esquema RMN en configuración de voteo, de tal manera que cuando ocurra una falla, se le enmascare y de inmediato se remplace el módulo fallado por uno de los repuestos del sistema. El gran beneficio de este modelo radica en que se restituye la configuración de voteo original después de que haya ocurrido una falla.

La técnica de RMN con Repuestos se muestra en la Figura A.7. El sistema permanece en la configuración básica RMN hasta que el detector de discrepancias determina que existe una unidad con falla. Un criterio para la detección de fallas, ver Figura A.7, es el comparar la salida del circuito de voteo con las salidas individuales de los módulos. Una unidad que difiera con la mayoría se le marca como fallada y se remueve del esquema RMN, es entonces cuando se conmuta a un módulo de repuesto para sustituir a la unidad que falló. La confiabilidad del sistema RMN se mantiene en tanto existan repuestos en el sistema.

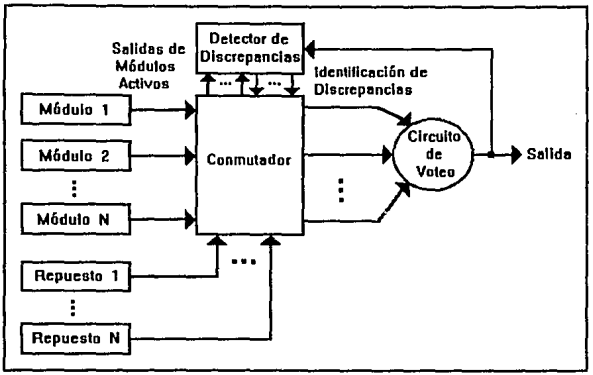


Figura A.7 Esquema de la técnica de Redundancia Modular N con Repuestos

Redundancia de Autodepuración

Este concepto es muy similar al del punto anterior, la diferencia radica en el uso que se le da a los módulos de repuesto. En el caso anterior, las refacciones están desactivadas hasta que ocurre una falla, en tanto que en el presente modelo, los módulos de repuesto participan en el voteo. En la Figura A.8 se ilustra el concepto de la técnica de redundancia de autodepuración. El circuito de voteo recibe las salidas de los N módulos que conforman el sistema (incluyendo los módulos de refacción), cada módulo tiene asociado un interruptor (para su propia desconexión en caso de tener alguna falla) cuyo funcionamiento depende de la salida del circuito de voteo y de la salida del módulo al que está asociado. Es pertinente señalar que el circuito de voteo basa su operación en umbrales ponderados e involucra el uso de elementos analógicos.

Sin embargo, este esquema es extremadamente atractivo por sus cualidades de autodepuración que facilitan el mantenimiento y la reparación, puesto que no es necesaria la interrupción de la operación del equipo para el reemplazo de módulos dañados.

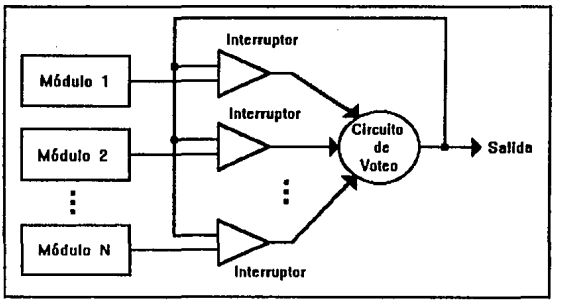


Figura A.8 Diagrama de la Redundancia de Autodepuración.

Redundancia Modular Selectiva

En este modelo, se tienen N módulos idénticos que componen el sistema junto con tres circuitos especiales denominados: comparadores, detectores y colectores. La estructura básica de esta técnica se muestra en la Figura A.9.

La función del circuito comparador es equiparar la salida de un módulo con las salidas del resto de los módulos, produciendo una salida por cada comparación realizada, para N=3 se tienen tres salidas del circuito comparador.

El detector determina las discrepancias reportadas por el comparador y deshabilita aquella unidad que difiera de la mayoría restante. Este circuito produce una salida por cada módulo del sistema, indicando su condición operativa (falla o buen estado).

El circuito colector se encarga de generar la salida del sistema, esto lo hará en función de las salidas de cada módulo y de los parámetros proporcionados por el detector, indicando los módulos con falla.

La diferencia principal entre el modelo de redundancia autodepurativa y el presente, radica en la forma de realizar las comparaciones. Mientras que en el anterior se compara una salida general con la propia salida del módulo, en este último esquema las comparaciones se realizan de forman centralizada.

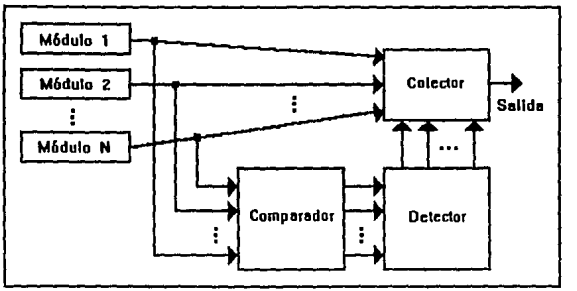


Figura A.9 Estructura del modelo de Redundancia Modular Selectiva

Arquitectura Triplex-Duplex

La última técnica híbrida es la denominada arquitectura triplex-duplex porque combina la redundancia modular triple y la duplicación con comparación. El esquema RMT permite enmascarar fallas y la continuidad en la operación, logrando con ello el buen desempeño de al menos un módulo sin falla. La técnica de duplicación con comparación facilita la detección de fallas y la remoción de los módulos con falla, que el proceso de voteo del esquema RMT indique. Esta arquitectura se usa típicamente en aplicaciones de control donde el arreglo de suma de corriente actúe como mecanismo de voteo. El esquema básico de la arquitectura triplex-duplex empleando técnicas de suma de corriente se muestra en la Figura A.10. El proceso de suma de corriente es capaz de tolerar cualquier falla única que ocurra en el sistema. Para detectar fallas, cada módulo se constituye mediante la técnica de duplicación con comparación. Si durante el proceso de comparación se detecta una falla, se remueve el módulo con falla del arreglo de suma de corriente. Si el esquema de duplicación con comparación detecta las posibles fallas que ocurran, la arquitectura triplex-duplex con el sumador de corriente puede tolerar hasta dos módulos con falla.

La salida del proceso de comparación se puede usar para desconectar un módulo del sumador de corriente, como se muestra en la Figura A.10. Si en la comparación se detecta una falla se abre el interruptor para remover el módulo con falla.

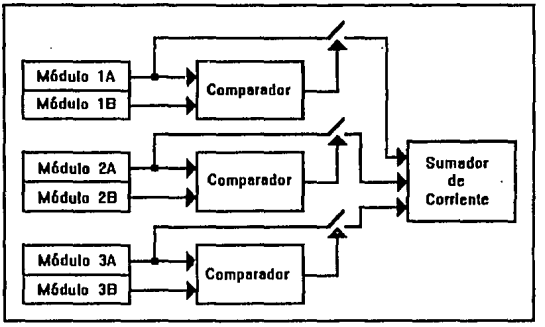


Figura A.10 Bosquejo de la arquitectura triplex-duplex.

A.2.2 Redundancia por *Software*

Agregar *software* extra a las rutinas de un sistema se hace con la finalidad de detectar y posiblemente tolerar algunas fallas. En aquellas aplicaciones que utilizan computadoras convencionales, las técnicas de detección y tolerancia a fallas se pueden implantar por programación, en cuyo caso las redundancias de componentes electrónicos pueden ser mínimas, mientras que la redundancia por *software* es notablemente mayor. La programación redundante se puede realizar de varias maneras; no se requiere duplicar programas completos para tener un *software* redundante. La redundancia por *software* puede ser un conjunto de líneas adicionales de programación para verificar la magnitud de una señal o una pequeña rutina que periódicamente pruebe una memoria por medio de accesos (lecturas y escrituras) a localidades específicas. A continuación se presentan las tres técnicas de redundancia por *software* mas usadas: examinador de consistencia, examinador de capacidad y los métodos de repetición de *software*.

A.2.2.1 Examinador de Consistencia

Un examinador de consistencia usa el conocimiento *a priori* de las características de la información para verificar su validez. Por ejemplo, en una cierta aplicación, se puede saber por adelantado que una cantidad digital nunca puede exceder una determinada magnitud.

A.2.2.2 Examinador de Capacidad

Para comprobar que un sistema posee la capacidad esperada es necesario aplicar algunas pruebas. Por ejemplo, para verificar la memoria, el procesador escribe y lee patrones de datos en ciertas localidades para saber si los datos se escriben y leen correctamente.

Otra técnica más, consiste en ejecutar periódicamente instrucciones específicas con datos particulares, para comparar sus resultados con los valores esperados que se almacenan en una ROM. De esta manera se puede definir el comportamiento de la unida de cálculo.

Otra forma de examinar un equipo, consiste en verificar que todos los procesadores de un sistema de múltiples procesadores, sean capaces de comunicarse con los demás procesadores. Esto se puede lograr si se hace pasar regularmente información específica de un procesador a otro.

A.2.2.3 Programación de N versiones

Las técnicas consideradas anteriormente se basan en el uso extra o redundante de *software* para detectar fallas en la electrónica. Sin embargo, no se han mencionado técnicas que permitan detectar y posiblemente tolerar fallas de programación. De hecho las fallas en programación se deben al mal diseño del *software* o a la incorrecta codificación. La simple duplicación de programas idénticos es un esquema pobre, pues no hay certeza de que el programa esté libre de fallas de programación.

La técnica de programación de N versiones [Chen & Avizienis 1978] permite detectar ciertas fallas en la programación. El principio es el siguiente, se requieren N versiones diferentes de un mismo programa y se comparan los resultados de los N módulos desarrollados. Cada uno de los N módulos debe ser diseñado y codificado por grupos distintos, basándose en el mismo conjunto de especificaciones, para que los N módulos (programas) tengan el mismo funcionamiento.

El Microprocesador 80386SX y el juego de CIs TACT83000

B.1 El Microprocesador 80386SX

El microprocesador 80386SX es una versión avanzada del microprocesador 80286, que conserva características de este último y mejora algunos detalles, entre los cuales se encuentra la conmutación de modo protegido a modo real. En el 286, esto era posible sólo a través de un *reset* por *hardware*, mientras que el 386 lo realiza por *software*. Esta es una de las grandes mejoras que incorpora el 386 respecto a su antecesor. El diagrama de bloques del 386 se muestra en las hojas de especificaciones (Apéndice D). Entre sus principales características se encuentran las siguientes:

- Capacidad de multitarea y multiprocesamiento.
- Direcciona hasta 16 Mbytes de memoria física y 1 Gbyte de memoria virtual.
- Memoria virtual con o sin paginación.
- Protección por *software*.
- Compatible con programas desarrollados para sus antecesores (8086/88, 80186/88 y 286).

El diagrama de terminales se muestra en las hojas de especificaciones técnicas. A continuación se dará una breve descripción de las señales de control que incorpora el 386.

- $A_2 - A_{23}$: Son las terminales de direcciones usadas para accesar los 16 Mbytes de memoria del sistema. Debe observarse que $A_0 - A_1$ aparecen como BLE# y BHE# , las cuales se describirán más adelante. Cabe recordar que las líneas de direcciones no están multiplexadas con los datos, como sucedía en el 8086.

- $D_0 - D_{15}$: Es el ducto de datos usado para intercambiar información con los sistemas de entrada/salida (I/O) y la memoria. Estas señales son bidireccionales.

- **BLE#, BHE#** : Estas señales son las encargadas de habilitar apropiadamente los bancos de memoria, para un dato de 16 bits, logrando tener transferencias de 8 ó 16 bits.

- **M/IO#** : La indicación de una transferencia a puerto (I/O) o a memoria se hace por medio de esta señal. Cuando tiene un 1 lógico es un operación con memoria y en caso contrario se trata de una interacción con puertos.

- **W/R#** : Esta línea indica si los accesos a puerto o a memoria son de lectura o escritura. La operación escritura se conoce cuando la señal toma un nivel alto y la lectura por un nivel bajo.

- **ADS#** : Esta señal (nivel bajo) indica la presencia de direcciones válidas durante transferencias a puerto o a memoria. Además, junto con W/R# y M/IO# definen los diferentes tipos de accesos.

- **RESET** : Se trata de una línea de entrada para indicar al procesador que debe inicializar su operación, ocasionando que empiece la ejecución de programa en la localidad FFFFF0h. El inicio se da siempre en modo real.

- **CLK2** : La señal de reloj debe ser del doble de la frecuencia de operación nominal del 386. Por ejemplo, el 80386SX-16 debe usar un reloj de 32MHz.

- **READY#** : La activación de esta señal se usa para controlar el número de estados de espera por insertar, requeridos cuando se interacciona con periféricos lentos.

- **LOCK#** : Esta salida se emplea para impedir que controladores de DMA u otros microprocesadores en el sistema, se apoderen del ducto ISA.

- **D/C#** : La señal datos/control# es una salida que indica cuando el ducto de datos contiene información para puertos o memoria (nivel alto) y también señala (en nivel bajo) una acción de control, como pueden ser la ejecución de una instrucción *HALT*, reconocimiento de interrupción y durante *fetchs* a memoria. En el caso de indicación de datos, se activa simultáneamente con las señales W/R#, M/IO# y ADS# para determinar los diferentes accesos a periféricos.

- **NA#** : Esta entrada se utiliza para forzar al microprocesador a proporcionar la siguiente dirección durante el ciclo de *bus* corriente, para lograr un esquema de *pipeline*.

- **HOLD** : Con esta entrada los periféricos piden un proceso de DMA al microprocesador.

- **HLDA** : Complementando la señal anterior, la línea HLDA (salida) indica que el 386 ha cedido el control del ducto del sistema.

- **PEREQ#** : Esta entrada permite al coprocesador 80387 pedir datos a través del 386.

- **BUSY#** : Esta entrada indica al 80386SX que debe esperar puesto que el 80387SX está ejecutando una instrucción numérica.

- **ERROR#** : El coprocesador indica al 386, por medio de esta entrada, la ocurrencia de un error.

- **INTR** : Por medio de esta línea los periféricos solicitan interrupciones mascarables (se enmascaran con el bit del registro de banderas I).

- **NMI** : Esta entrada es para la petición de interrupciones no mascarables. Como en los otros miembros de la familia, dicha entrada usa el vector de interrupción 2.

Para profundizar sobre el funcionamiento del microprocesador 80386, consultar [Intel 1990].

B.2 El Juego de CIs TACT83000

El juego de CIs (*chip set*) TACT8300 AT de Texas Instruments fue diseñado para lograr un alto desempeño de los sistemas PC-AT basados en el 386. El *chip set* está dividido funcionalmente en tres dispositivos: el TACT83443 que es la Unidad de Interfaz con el Ducto AT (ATU), el TACT83442 que es la Unidad de Control de Memoria (MCU) y el TACT83441 que es la Unidad de Ruteo de Datos (DPU). Como un DPU proporciona una palabra de 16 bits al sistema, se utilizan dos conectadas en cascada para los sistemas de 32 bits (en nuestro caso sólo se utiliza uno). Las características generales del TACT83000 son:

- Las principales funciones se programan por *software*.
- Se pueden conectar en cascada hasta ocho MCUs.
- En los modos normal, de paginación y paginación entrelazada se pueden emplear DRAMs de 256K, 1M y 4Mbits.
- Soporta hasta dos bancos de memoria para sistemas de 32 bits y cuatro para sistemas de 16 bits.
- Se dispone de "Shadow RAM" entre las localidades 0C 0000h y 0F FFFFh.
- La MCU contiene el mapeo global de páginas de RAM (PMRAM) que permite:
- El remapeo de cada bloque de 16Kbytes de memoria abajo de 1Mbyte.

- El remapeo de cada bloque de 64Kbytes de memoria arriba de 1Mbyte.

- Los relojes internos trabajan con dos frecuencias diferentes.

- La interfaz con el ducto AT trabaja en forma asíncrona con el *buffer* de escritura.

- El reloj de tiempo real (RTC) tiene 128 bytes de RAM tipo CMOS.

- Tiene la capacidad de realizar accesos directos a memoria (DMA) de 32 bits.

- Posee una total compatibilidad para manejar el ducto PC-AT.

- Emplea tecnología CMOS de alta velocidad (1-μm).

La arquitectura del TACT83000 está diseñada para usarse en sistemas con o sin memoria caché. En un sistema 386 con memoria caché, el TACT83000 emplea tres ductos. Estos ductos son el ducto local o de CPU, el ducto local extendido (EL), y el ducto AT/DX. En las hojas de especificaciones anexas se muestra la estructura de los tres ductos utilizados por el TACT8300 en un sistema con memoria caché.. La arquitectura también proporciona un ducto para periféricos de 8 bits, el ducto XD, que opera con los tiempos del ducto AT.

En los sistemas con memoria caché, la interfaz entre el ducto local del CPU y el ducto EL se regula con el controlador de memoria caché. El TACT83000 sustenta el uso del controlador de memoria caché 82385 (o alguno compatible), el DPU proporciona el control de los *buffers* de dirección externa y de tranceptores de datos entre el ducto local del CPU y el ducto EL. En un sistema donde no se cuenta con memoria caché, el ducto EL se convierte esencialmente en el ducto del CPU.

El ducto EL incluye líneas de datos (ED) y de direcciones (EA). El MCU se encarga de convertir las direcciones mapeadas (enviadas al final de cada evento) en direcciones de memoria física (MA). De la misma manera, el ducto AT agrupa las líneas de direcciones del sistema (SA) y las líneas de direcciones sostenidas (LA), además de los datos del sistema (SD). Las líneas de control quedan incluidas en los ductos, a pesar de no aparecer en la figura. Todo el tráfico entre los ductos de datos se controla

por medio del (de los) DPU(s), agrupando la lógica de conversión y ajuste de tamaños de palabras, además de realizar la verificación y generación de los bits de paridad.

El ATU, localizado entre el ducto EL y el ducto del sistema AT, proporciona todas las líneas de control y la capacidad de corriente necesaria (24 mA) para controlar los canales de Entrada/Salida. El ATU además contiene los siguientes periféricos esenciales para el sistema AT:

- Dos controladores de DMA compatibles con el 8237A.
- Dos controladores de interrupciones compatibles con el 8259.
- Tres temporizadores/contadores compatibles con el 8254.
- Un reloj de tiempo real compatible con el 146818.
- Un registro de 128 bytes de respaldo por batería para la RAM CMOS y el RTC.

El MCU controla la interfaz con el sistema de memoria y se localiza en el ducto EL. Este MCU es un controlador de DRAM que tiene amplias capacidades para el mapeo de memoria y funciona en diferentes modos y configuraciones, de acuerdo a las necesidades del diseño. Otras características del MCU son:

- Cada MCU puede direccionar hasta 32Mbytes.
- Es posible conectar en cascada hasta 8 MCUs.
- Las configuraciones se programan por *software*.
- Completa programación de los parámetros para controlar una DRAM:
- Duración de los pulsos de CAS# y RAS#.
- Retraso de RAS# y la dirección de memoria.
- Retraso en la dirección de memoria y la lectura del CAS#.

- Tiempo de precarga del CAS# y RAS#.

- Los modos de acceso a las DRAMs son:
- Modo normal.

- Modo página.

- Modo entrelazado, modo por palabra/palabra doble, o modo página limitado para configuraciones de dos vías (32 bits) y de cuatro vías (16 bits).

- Refrescamiento mediante la técnica de CAS# antes de RAS#.

El DPU ofrece el enrutamiento de datos para todo el tráfico de datos en el sistema. Dicho enrutamiento se proporciona por dos secciones del DPU: la que controla los ductos de datos del CPU y de EL (las líneas de datos D/ED) y el ducto de datos AT (las líneas de datos SD/XD). Para operaciones de lectura y escritura, los datos de ambas secciones se transfieren mediante un *latch-buffer* del ducto AT. El DPU tiene un *buffer* para escrituras AT, controlado por el ATU. El DPU también posee lógica para la conversión y el ajuste de los accesos al ducto AT de 8 y 16 bits, así como también incluye lógica para la generación y verificación de la paridad.

Para mayores detalles acerca del funcionamiento del TACT83000 ver [TACT8300 1990].

Estadísticas de los circuitos impresos diseñados

En capítulos anteriores se explicó el diseño electrónico de las tarjetas objetivo de la CTF, además de proporcionar detalles de los circuitos impresos desarrollados, las dimensiones de las tarjetas son 13.827 X 6.347 pulgadas. A continuación se presentan las estadísticas de cada tarjeta.

EstadÀsticas del circuito impreso de la UP

Arcs:	223	Components:	200	Pads:	1633	Polygons:	400
Lines:	12220	Text strings:	446	Vias:	758	Holes:	1687

Text Summary -

T_0060_010_0:	182
T_0060_010_1:	99
T_0060_010_2:	48
T_0060_010_3:	55
T_0080_010_0:	28
T_0120_010_0:	22
T_0120_010_1:	1
T_0120_010_3:	3
T_0148_010_0:	8

Line Summary -

L_008:	590
L_010:	1293
L_012:	6668
L_014:	3669
L_020:	222
L_040:	1

Pad Summary -

P_EL_0040_0040_022_AL:	4
P_EL_0050_0050_028_AL:	3
P_EL_0050_0050_030_AL:	27
P_EL_0060_0060_034_AL:	8
P_EL_0060_0060_038_AL:	693
P_EL_0070_0070_046_AL:	1
P_EL_0250_0250_125_AL:	3
P_RR_0050_0280_000_BL:	98
P_RR_0050_0280_000_TL:	98
P_SQ_0008_0070_000_TL:	1
P_SQ_0008_0078_000_TL:	1
P_SQ_0008_0080_000_TL:	2
P_SQ_0008_0090_000_TL:	1

EstadÀsticas del circuito impreso de la UP

EstadÀsticas del circuito impreso de la UP

	P_SQ_0008_0100_000_TL:	7
	P_SQ_0008_0110_000_TL:	2
	P_SQ_0008_0120_000_TL:	90
	P_SQ_0012_0090_000_TL:	50
	P_SQ_0014_0080_000_TL:	1
	P_SQ_0014_0090_000_TL:	2
	P_SQ_0014_0100_000_TL:	3
	P_SQ_0014_0120_000_TL:	74
	P_SQ_0040_0040_022_AL:	2
	P_SQ_0050_0050_028_AL:	1
	P_SQ_0050_0050_030_AL:	27
	P_SQ_0060_0060_034_AL:	2
	P_SQ_0060_0060_038_AL:	157
	P_SQ_0070_0012_000_TL:	1
	P_SQ_0070_0070_046_AL:	1
	P_SQ_0080_0008_000_TL:	1
	P_SQ_0080_0014_000_TL:	1
	P_SQ_0090_0008_000_TL:	2
	P_SQ_0090_0012_000_TL:	49
	P_SQ_0090_0014_000_TL:	1
	P_SQ_0100_0008_000_TL:	10
	P_SQ_0110_0014_000_TL:	2
	P_SQ_0120_0008_000_TL:	91
	P_SQ_0120_0014_000_TL:	116
Via Summary -	V_CR_040_028:	677
	V_CR_050_028:	57
	V_CR_050_034:	1
	V_CR_060_038:	23
Hole Summary -	H_022:	6
	H_028:	738
	H_030:	54
	H_034:	11
	H_038:	873

EstadÀsticas del circuito impreso de la UP

	H_046:	2
	H_125:	3
Plane Connections -	Power Thermals:	0
	Power Directs:	0
	Ground Thermals:	0
	Ground Directs:	0

Estadàsticas del circuito impreso de la UDFC

Arcs:	148	Components:	141	Pads:	1314	Polygons:	1
Lines:	6211	Text strings:	270	Vias:	530	Holes:	1648

Text Summary -

T_0060_010_0:	134
T_0060_010_1:	39
T_0060_010_2:	16
T_0060_010_3:	65
T_0060_012_0:	7
T_0080_010_0:	6
T_0148_010_0:	3

Line Summary -

L_010:	628
L_012:	5013
L_014:	17
L_016:	12
L_020:	233
L_024:	5
L_030:	133
L_040:	92
L_050:	226

Pad Summary -

P_EL_0050_0050_022_AL:	1
P_EL_0050_0050_028_AL:	3
P_EL_0060_0060_038_AL:	927
P_EL_0250_0250_125_AL:	3
P_OV_0050_0050_034_AL:	51
P_RR_0050_0280_000_BL:	98
P_RR_0050_0280_000_TL:	98
P_SQ_0050_0050_022_AL:	1
P_SQ_0050_0050_028_AL:	1
P_SQ_0050_0050_034_AL:	1
P_SQ_0060_0060_038_AL:	130

Via Summary - V_CR_040_028: 519

Estadàsticas del circuito impreso de la UDFC

EstadÀsticas del circuito impreso de la UDFC

==========

V_CR_050_028: 11

Hole Summary -
H_022: 2
H_028: 534
H_034: 52
H_038: 1057
H_125: 3

Plane Connections -
Power Thermals: 0
Power Directs: 0
Ground Thermals: 0
Ground Directs: 0

APÉNDICE D

Hojas de Especificaciones

Lattice Semiconductor Corporation

ORIGINAL

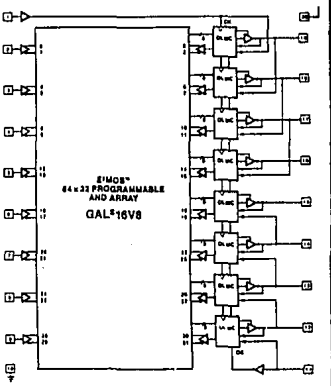
GAL® 16V8

GENERIC ARRAY LOGIC

FEATURES

- ELECTRICALLY ERASABLE CELL TECHNOLOGY
  - — Reconfigurable Logic
  - — Reprogrammable Cells
  - — Guaranteed 100% Yields
- HIGH PERFORMANCE E²CMOS™ TECHNOLOGY
  - — Low Power: 45mA Max Active
  - — High Speed: 15ns Access Max
    25ns Access Max
- EIGHT OUTPUT LOGIC MACROCELLS
  - — Maximum Flexibility for Complex Logic Designs
  - — Also Emulates 20-pin PAL® Devices with Full Function/Fuse Map/Parametric Compatibility
- PRELOAD AND POWER-ON RESET OF ALL REGISTERS
  - — 100% Functional Testability
- HIGH SPEED PROGRAMMING ALGORITHM
- SECURITY CELL PREVENTS COPYING LOGIC
- DATA RETENTION EXCEEDS 20 YEARS
- ELECTRONIC SIGNATURE FOR IDENTIFICATION

FUNCTIONAL BLOCK DIAGRAM



DESCRIPTION

The LATTICE E²CMOS™ GAL® device combines a high performance CMOS process with electrically erasable floating gate technology. This programmable memory technology applied to array logic provides designers with reconfigurable logic and bipolar performance at significantly reduced power levels.

The 20-pin GAL®16V8 features 8 programmable Output Logic Macrocells (OLMCs) allowing each output to be configured by the user. Additionally, the GAL®16V8 is capable of emulating, in a functional/fuse map/parametric compatible device, all common 20-pin PAL® device architectures.

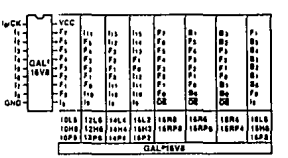
Programming is accomplished using readily available hardware and software tools. LATTICE guarantees a minimum 100 erase/write cycles and that data retention exceeds 20 years.

Unique test circuitry and reprogrammable cells allow complete AC, DC, cell and functionality testing during manufacture. Therefore, LATTICE guarantees 100% field programmability and functionality of the GAL® devices. In addition, electronic signature is available to provide positive device ID. A security circuit is built-in, providing proprietary designs with copy protection.

PIN NAMES

I_0-I_{15}	INPUT	$\overline{OE}$	OUTPUT ENABLE
CK	CLOCK INPUT	V_{CC}	POWER (+5V)
B_0-B_7	BIDIRECTIONAL	GND	GROUND
F_0-F_7	OUTPUT		

GAL®16V8 EMULATING PAL® DEVICES



FEBRUARY, 1988

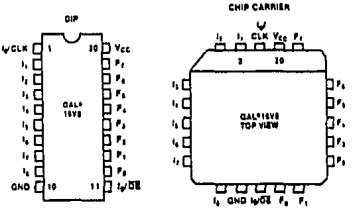
GAL® 16V8 HIGH PERFORMANCE E²CMOS GENERIC ARRAY LOGIC

ABSOLUTE MAXIMUM RATINGS(1)

Supply voltage V_{CC} ..... −.5 to +7V
Input voltage applied ..... −2.5 to V_{CC} +1.0V
Off-state output voltage applied .. −2.5 to V_{CC} +1.0V
Storage temperature ..... −65 to 125°C

1. Stresses above those listed under the "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied (while programming, follow the programming specifications).

PIN CONFIGURATION



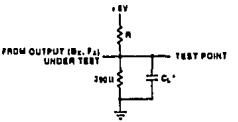
OPERATING RANGE

SYMBOL	PARAMETER	TEMPERATURE RANGE									UNIT
		MILITARY			INDUSTRIAL			COMMERCIAL			
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.5	5	5.5	4.75	5	5.25	V
T_A	Ambient temperature				−40		85	0		75	°C
T_C	Case temperature	−55		125							°C

SWITCHING TEST CONDITIONS

Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	3ns 10% - 90%
Input Timing Reference Levels	1.5V
Output Timing Reference Levels	1.5V
Output Load	See Figure

3-state levels are measured 0.5V from steady-state active level.



*C_L INCLUDES JIG AND PROBE TOTAL CAPACITANCE

CAPACITANCE (T_A = 25°C, f = 1.0 MHz)

SYMBOL	PARAMETER	MAXIMUM*	UNITS	TEST CONDITIONS
C_I	Input Capacitance	12	pF	V_{CC} = 5.0V, V_I = 2.0V
C_F	Output Capacitance	15	pF	V_{CC} = 5.0V, V_F = 2.0V
C_B	Bidirectional Pin Cap	15	pF	V_{CC} = 5.0V, V_B = 2.0V

*Guaranteed but not 100% tested.

D3612 AUGUST 1990

- High-Speed 1-µm CMOS Technology Supports System Speeds up to 33 MHz
- Fully AT-Compatible 386 Three-Chip SX, Four-Chip DX Solutions
- Only Four Additional Logic Chips Needed
- Major Features Programmable Through Software
- TACT83442 Memory Control Unit (MCU)
  - Cascadable up to Eight Devices
  - Address Range of up to 32M Byte Per Device, 256M Byte Fully Cascaded
  - Supports 256K-, 1M-, and 4M-Bit DRAMs in Normal, Page, Word-Interleave, and Page Block-Interleave Modes
  - Programmable DRAM Timing Parameters
  - Supports up to Two Memory Banks for 32-Bit Systems and Four Banks for 16-Bit Systems
  - Can Directly Drive up to 36 DRAM Devices
  - Shadow RAM Available Between 0C 0000h and 0F FFFFh
  - Contains Global Page Mapping RAM Allowing Remap of
    - 64K-Byte Memory Blocks Above 1M Byte
    - 16K-Byte Memory Blocks Below 1M Byte
- TACT83443 AT Bus Interface Unit (ATU)
  - Internal Clock Switching Between Two Independent Frequencies Controlled by Software
  - Asynchronous AT Bus Interface With Write Buffer Option
  - Full AT Direct-Drive Capability
  - Extended Direct Memory Access Mode for 32-Bit Operation
  - Fast CPU Reset and A20GATE Modification
  - Numeric Processor Interface for 387SX, 387DX, and Weitek 3167
  - Integrates All Essential AT Peripherals
  - Real Time Clock With 128-Byte CMOS RAM
- TACT83441 Data Path Unit (DPU)
  - 8- and 16-Bit Data Bus Sizing
  - Data Path Cascadable to 32 Bits
  - Write Buffer Capability for AT Bus Access
  - Supports Posted Write Operations From Cache Controller
  - Parity Generation and Checking Logic

description

The Texas Instruments TACT83000 AT Chip Set is designed for cached and noncached 386™-based PC-AT™ compatible systems running at speeds up to 33 MHz. Manufactured with high-speed 1-µm CMOS EPIC™ technology, the chip set is functionally partitioned into three devices: the TACT83443 AT Bus Interface Unit (ATU), the TACT83442 Memory Control Unit (MCU), and the TACT83441 Data Path Unit (DPU). The ATU is packaged in a 208-lead plastic quad flatpack (QFP), while the MCU and DPU are packaged in 100-lead plastic QFPs.

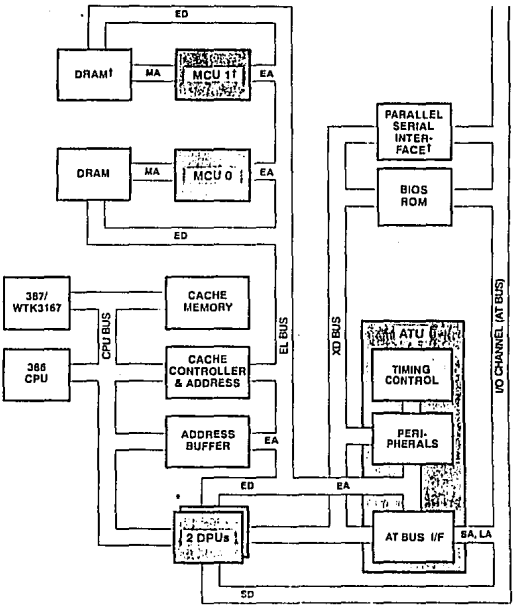
These three chips, along with four other logic chips, comprise all the logic necessary for a fully compatible 16-bit 386SX-based system. Since one DPU provides a 16-bit data path, a 32-bit 386DX-based system requires an additional DPU.

With software-controlled configuration registers on board the ATU and MCU, the chip set supports a wide variety of PC system configurations. For complete programming details, see the *TACT83000 AT Chip Set User's Guide*, literature number SRZU001.

PC-AT is a trademark of International Business Machines.
386 and 387 are trademarks of Intel Corporation.
EPIC is a trademark of Texas Instruments Incorporated.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

functional block diagram – 386DX cache-based system



† Optional

architecture

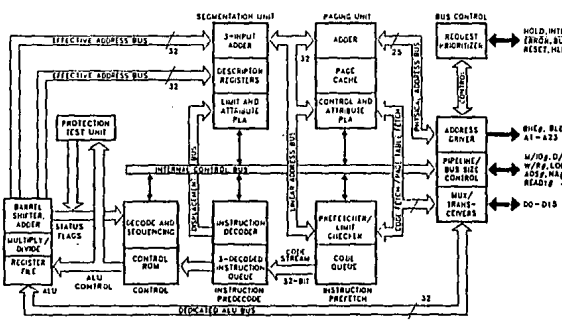
In a 386 cache-based system, the TACT83000 chip set architecture uses three tiers of buses. From the system core outward, these buses are the local or CPU bus, the extended local (EL) bus, and the AT bus. The architecture also provides an 8-bit peripheral bus, the XD bus, which operates with AT bus timing. The three-tiered bus structure used by the TACT83000 in a cache-based system can be seen in the functional block diagram.

386™ SX MICROPROCESSOR

- **Full 32-Bit Internal Architecture**
  - 8-, 16-, 32-Bit Data Types
  - 8 General Purpose 32-Bit Registers
- **Runs Intel386™ Software in a Cost Effective 16-Bit Hardware Environment**
  - Runs Same Applications and O.S.'s as the 386™ DX Processor
  - Object Code Compatible with 8086, 80186, 80286, and 386 Processors
  - Runs MS-DOS*, OS/2* and UNIX**
- **Very High Performance 16-Bit Data Bus**
  - 20 MHz Clock
  - Two-Clock Bus Cycles
  - 20 Megabytes/Sec Bus Bandwidth
  - Address Pipelining Allows Use of Slower/Cheaper Memories
- **Integrated Memory Management Unit**
  - Virtual Memory Support
  - Optional On-Chip Paging
  - 4 Levels of Hardware Enforced Protection
  - MMU Fully Compatible with Those of the 80286 and 386 DX CPUs
- **Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System**
- **Large Uniform Address Space**
  - 16 Megabyte Physical
  - 64 Terabyte Virtual
  - 4 Gigabyte Maximum Segment Size
- **High Speed Numerics Support with the 387™ SX Coprocessor**
- **On-Chip Debugging Support Including Breakpoint Registers**
- **Complete System Development Support**
  - Software: C, PL/M, Assembler
  - Debuggers: PMON-386 DX, ICE™-386 SX
  - Extensive Third-Party Support: C, Pascal, FORTRAN, BASIC, Ada*** on VAX, UNIX**, MS-DOS*, and Other Hosts
- **High Speed CHMOS III Technology**
- **100-Pin Plastic Quad Flatpack Package**

(See Packaging Outlines and Dimensions #231369)

The 386™ SX Microprocessor is a 32-bit CPU with a 16-bit external data bus and a 24-bit external address bus. The 386 SX CPU brings the high-performance software of the Intel386™ Architecture to midrange systems. It provides the performance benefits of a 32-bit programming architecture with the cost savings associated with 16-bit hardware systems.



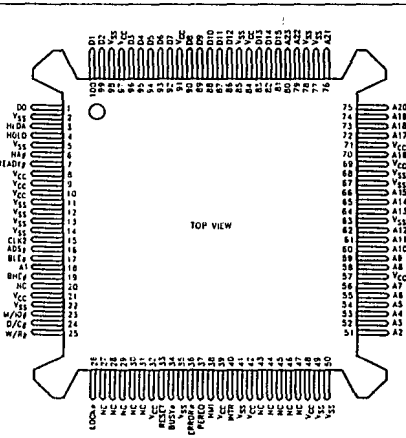
240187-47

386™ SX Pipelined 32-Bit Microarchitecture

*MS-DOS and OS/2 are trademarks of Microsoft Corporation.
**UNIX is a trademark of AT&T.
***Ada is a trademark of the Department of Defense.

October 1989
Order Number: 240187-004

1.0 PIN DESCRIPTION



NOTE:
NC = No Connect

Figure 1.1. 386™ SX Microprocessor Pin out Top View

Table 1.1. Alphabetical Pin Assignments

Address		Data		Control		N/C	V_{CC}	V_{SS}
A_1	18	D_0	1	ADS#	16	20	8	2
A_2	51	D_1	100	BHE#	19	27	9	5
A_3	52	D_2	99	BLE#	17	28	10	11
A_4	53	D_3	96	BUSY#	34	29	21	12
A_5	54	D_4	95	CLK2	15	30	32	13
A_6	55	D_5	94	D/C#	24	31	39	14
A_7	56	D_6	93	ERROR#	36	43	42	22
A_8	58	D_7	92	HLDA	3	44	48	35
A_9	59	D_8	90	HOLD	4	45	57	41
A_{10}	60	D_9	89	INTR	40	46	69	49
A_{11}	61	D_{10}	88	LOCK#	26	47	71	50
A_{12}	62	D_{11}	87	M/IO#	23		84	63
A_{13}	64	D_{12}	86	NA#	6		91	67
A_{14}	65	D_{13}	83	NMI	38		97	68
A_{15}	66	D_{14}	82	PEREQ	37			77
A_{16}	70	D_{15}	81	READY#	7			78
A_{17}	72			RESET	33			85
A_{18}	73			W/R#	25			98
A_{19}	74							
A_{20}	75							
A_{21}	76							
A_{22}	79							
A_{23}	80							

[Aréchiga et al. 1980] Aréchiga, R. et al."Fundamentos de computación," Limusa, México 1980.

[Avizienis et al. 1971] Avizienis, A, Gilley, G.C., F.P. Mathur, D.A. Rennels, J.A. Rohr, and D.K. Rubin. "The STAR (Self-Testing And Repairing) computer: An investigation of the theory and practice of fault tolerant computer design," *IEEE Transactions on Computers*, Vol. C-20, No. 11, November 1971, pp. 1312-1321.

[Avizienis et al. 1990] Avizienis, A. et al. "Lectures notes on the course the applications of fault tolerant technology," UCLA, August 1990.

[Chen & Avizienis 1978] Chen, L., y A. Avizienis. "N-version programming: A fault tolerant approach to reliability of software operation," *Proceedings of the International Symposium on Fault Tolerant Computing*, 1978, pp 3-9.

[Intel 1990] 386SX Hardware Reference Manual, Intel 1990.

[Johnson 1984] Johnson D. "The Intel 432: A VLSI architecture for fault-tolerant computer systems," *Computer*, August 1984, pp 40-48.

[Johnson W. 1989] Johnson, Barry W. *Design and analysis of fault-tolerant digital systems*. Addison-Wesley Inc., 1989.

[Koczela 1968] Koczela, L. J. and G. J. Burnett, "Advanced space missions and computer systems," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-4, No. 3, May 1968, pp. 456-467.

[Kuehn 1969] Kuehn, R. E. "Computer redundancy: design, performance and future," *IEEE Transactions on Reliability*, Vol. R-18, No. 1, February 1969.

[Lerner 1983] Lerner, E. J. "Crossroads in space," *Spectrum*, Vol. 20, No. 9, September 1983, pp. 28-55.

[Moore & Shannon 1956] Moore, E. F. and C. E. Shannon, "Reliable Circuits Using Less Reliable Relays, pt. 1," J. Franklin Inst., Vol. 262.

[Ng & Avizienis 1980] Ng Y. and A. Avizienis "A unified reliability model for fault-tolerant computers," *IEEE Transactions on Computers*, Vol. C-29, November 1980, pp 1002-1011.

[Pierce 1965] Pierce, W. H. "Failure-Tolerant Computer Desig," Academic Press, NY and London, 1965.

[Rennels 1980] Rennels D. A. "Distributed fault-tolerant computer systems," *IEEE Computer*, Vol. 13, No. 3, March 1980, pp. 55-64.

[Rennels 1984] Rennels D.A. "Fault-tolerant computing -Concepts and examples," *IEEE Transactions on Computers*, Vol. C-33, No. 12, December 1984, pp 1116-1129.

[Rivera 1993] Rivera M. R. Tesis de Licenciatura, Facultad de Ingeniería, UNAM; "Desarrollo de una tarjeta electrónica para controlar dinámicamente el flujo de información en los ductos de una computadora autoreconfigurable", 1993.

[Serlin 1984] Serlin O. "Faul-tolerant systems in commercial applications," *Computer*, Vol. 17, No. 8, August 1984, pp. 19-30.

[Sklaroff 1976] Sklaroff, J.R. "Redundacy managment technique for the space shuttle computers," *IBM Journal of Research and Development*, Vol. 20, No. 1, January 1976, pp. 20-28

[Solari 1991] Solari Ed., *AT Bus Design*, Annabooks, 1991

[Shladover 1993] Shladover, E. S. " Research an d development needs for adevenced vehicle control systems," IEEE Micro, February 1993, pp. 11-19.

[TACT83000 1990] TACT83000 AT Chip Set User's Guide. Texas Instruments, 1990.

[Toy 1978] Toy, W.N. "Fault-tolerant design of local ESS processor," *Proceedings of the IEEE*, Vol. 66, No. 10, October 1978, pp. 1126-1145.

[Vicente et al. 1986] Vicente-Vivas E. et al., "Instrumentación de los experimentos espaciales automáticos de la UNAM," Memoria del XII Congreso de la ANIAC, Septiembre 1986.

[Vicente 1993] Vicente Vivas E., "Computadoras tolerantes a fallas: evolución, análisis y oportunidades," *Memoria del XIX Congreso de la ANIAC*, Septiembre 1993, pp. 67-71.

[Vicente et al. 1993] Vicente Vivas E., Mejía-Galeana J. A., Rivera M. R. "Diseño de una computadora de procesamiento concurrente para aplicaciones de alto riesgo," *Memoria del II Congreso Nacional de Investigación en Ciencias Computacionales*, Octubre 1993, pp. 137-148.

[Von Neumann 1956] Von Neumann, J. "Probabilistic logics and the synthesis of the reliable organisms from unreliable components," *Automata Studies*, Annals of Mathematical Studies, Princeton University Press, No. 34, pp. 43-98, 1956.

[Zanoni 1993] Zanoni, E. "Improving the reliability and safety of automotive electronics," IEEE Micro, February 1993, pp. 30-48.

1. Asser S. M., Stigliano V. J. & Bahrenburg R. F., Microcomputer theory and servicing, Merril, 1990.

2. Brey B. B., The Intel Microprocessors 8086/8088, 80186, 80286, 80386 and 80486., architecture, programming and interfacing, Merril, 1991.

3. CMOS Logic Databook, National Semiconductors, 1988.

4. Di Giacomo J., Digital bus handbook, Mc Graw-Hill, 1990.

5. FAST and LS TTL Data, Motorola, 1992.

6. Fletcher W. I., An engineering approach to digital design, Prentice Hall, 1980.

7. Ginsberg, G. L., Printed Circuit Design, Mc Graw-Hill, 1991.

8. Peripherals, Intel 1990.

9. Tango PCB PLUS Reference Manual, Accel Technologies, 1990