(1·109101) xx

CINVESTAV - IPN
Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

Matriz de conmutación de alta velocidad: rediseño de algunos de sus elementos

Tesis que presenta

Julio Cesar Silva Briano

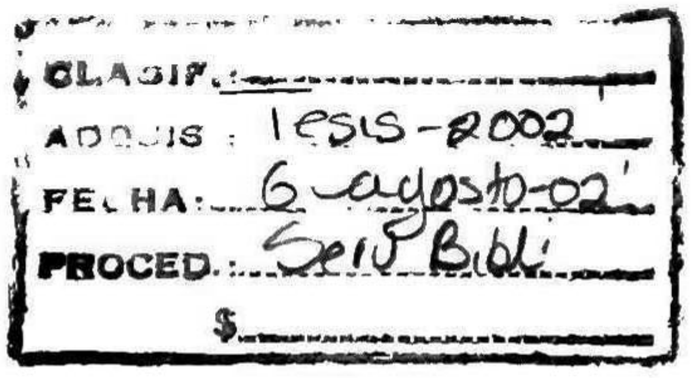
Para obtener el grado

Maestro en ciencias

En la especialidad de

Ingeniería Eléctrica

Guadalajara Jal, Abril del 2002.


CLASIF.:
ADQUIS.: IESIS-2002
FECHA: 6 agosto-02
PROCED.: Serie Bibl
$

Matriz de conmutación de alta velocidad: rediseño de algunos de sus elementos

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

Julio Cesar Silva Briano

Ingeniero en Sistemas Computacionales
Universidad Autónoma de Aguascalientes, 1992-1997

Becario del Conacyt, expediente No. 136570

Directores de Tesis:

Dr. Deni L. Torres Román.

Dr. Manuel E. Guzmán Rentería.

Cinvestav del IPN Unidad Guadalajara, Abril del 2002.

INDICE DE REFERENCIAS

INDICE DE TABLAS

INDICE DE FIGURAS

1 Introducción

Las redes ATM (Asynchronous Trasfer Mode) se han estandarizado por ITU-T, para la creación de nuevas redes de comunicaciones. La cantidad de usuarios impulsa a la creación de elementos de comunicaciones. Estas redes de alta velocidad necesitan de un desempeño en redes digitales. Para los estándares sobre ATM, ver [10] y ver [11], estos estándares incluyen redes basadas en ISDN banda estrecha y banda ancha.

Las matrices de conmutación (MC) son elementos de las redes de alta velocidad, las cuales manejan velocidades desde 25Mbps, 15Mbps, 622Mbps en sus entradas y anchos de banda en el orden de los Gbps.

En trabajos anteriores como el de [1] y [7] donde se diseñó y desarrolló una MC de alta velocidad, fue necesario el desarrollo de elementos de comunicación para propósitos de control y monitoreo. Es importante mencionar también el trabajo desarrollado por [2] y el de [3] para el análisis y diseño de elementos periféricos de la MC, respectivamente.

Esta tesis presenta la importancia del desarrollo de software para pruebas y monitoreo, y también el desarrollo de firmware y software.

El documento presenta los siguientes capítulos

Capítulo 1 Introducción

En este capitulo se da una vista preliminar del contenido del documento explicando brevemente su contenido y sus objetivos. También se prestan los objetivos particulares de la tesis presentada, la importancia de los objetivos radica en que muestran los elementos importantes que se desarrollaron, también los objetivos muestran lo que se logro durante el desarrollo de la tesis.

Capítulo 2 Agregar a la MC el soporte del protocolo UTOPIA nivel 1.

Explica la inclusión del protocolo UTOPIA nivel 1 para propósitos de prueba y de crecimiento de la funcionalidad de la matriz.

Capítulo 3 Desarrollo de un protocolo de comunicación entre firmware y una aplicación en Windows.

Se desarrolló de un protocolo de comunicaciones creado para la comunicación de la PC con la matriz de conmutación para propósitos de monitoreo y pruebas.

Capítulo 4 Plan de pruebas para la MC.

Se muestra el plan de pruebas y los elementos a probar junto con los algoritmos que se han escogido para verificar la MC.

Capítulo 5 Firmware con soporte del plan de pruebas.

Se desarrolló un firmware para el microcontrolador y su labor en el plan de pruebas.

Capítulo 6 Desarrollo de drivers de prueba, verificación y monitoreo de la MC.

Se explica la manera en que se desarrolla el driver que conecta la PC y el microcontrolador, también en que herramienta se implementó y la manera en que se desarrolló.

Capítulo 7 Realizar pruebas conjuntas con el CL interfaz E1/T1.

Se muestra las pruebas desarrolladas para el CL junto con la matriz de conmutación junto con las pruebas desarrolladas para el plan de pruebas, al igual que la funcionalidad y elementos probados.

Capítulo 8 Conclusiones.

Se da a conocer los resultados finales y las conclusiones encontradas durante el desarrollo de esta tesis.

1.1 Objetivos de la tesis

El modelo base de esta tesis presenta problemas de implementación y funcionalidad, lo cual es de nuestro interés resolverlos, los objetivos son:

- Agregar a la MC el soporte del protocolo UTOPIA nivel 1.
- Desarrollo de un protocolo de comunicación entre firmware y una aplicación en Windows.
- Realizar un plan de pruebas para la MC.
- Realizar firmware que soporte el plan de pruebas.
- Desarrollo de drivers para la comunicación y monitoreo de la MC desde los programas que corren en una PC.
- Realizar pruebas conjuntas con el CL interfaz E1/T1.

Otros objetivos.

- Considerados secundarios, estos objetivos no están contemplados para su desarrollo, pero sí se mostrará la documentación relacionada.
- Plantear otras opciones para el desarrollo de una matriz de conmutación compatible con UTOPIA.

2 Agregar a la MC el soporte del protocolo UTOPIA nivel 1

2.1 Panorama general

Una red de interconexión es un sistema con una MC que es controlada para el intercambio de información entre elementos de una red de interconexión. Por otra parte, la MC requiere de una red capaz de proveer las velocidades de transmisión requeridas. La dificultad en implementarla depende con la complejidad de los requerimientos del sistema.

2.2 Arquitectura de la matriz de conmutación, Generalidades

La arquitectura de la MC bajo estudio, abarca los siguientes puntos:

- Simplicidad.
- Modularidad.
- Matriz de conmutación de Orden (N,N), con autoenrutado de los paquetes.
- Soporte al servicio reproducción (multicast y broadcast).

Estas características han decidido la arquitectura actual. En resumen, la MC puede ser considerada un bloque básico de 4 puertos bidireccionales, escalable hasta 16. Cada puerto se conecta a un circuito de línea (**CL**) específico para protocolos como ATM, Fast-Ethernet, Ethernet, ISDN, E1/T1, etc.

A continuación se muestra el diagrama a bloques de la MC:

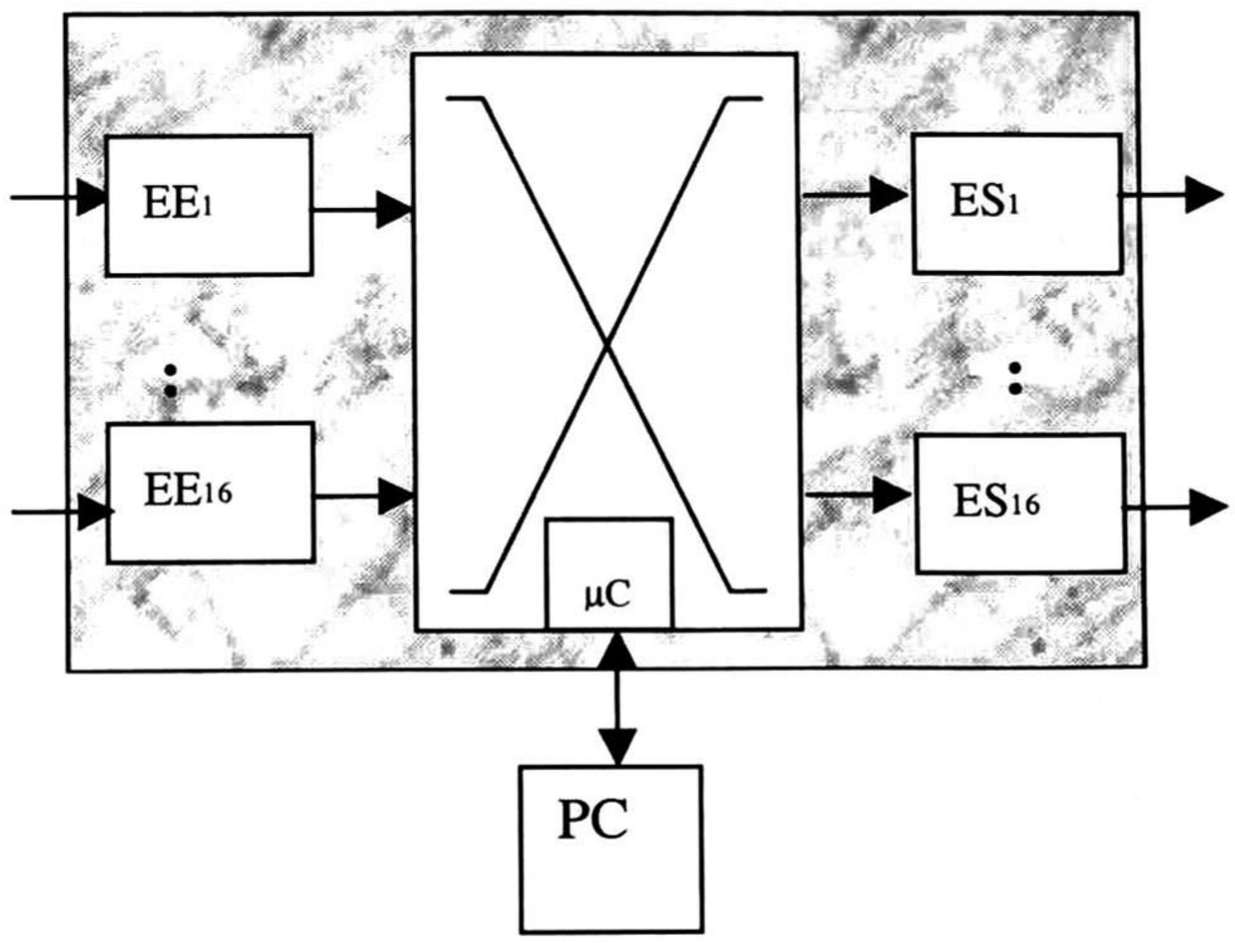


Figura 1 *Diagrama general de la MC.*

La MC está compuesta con un conjunto de componentes que permiten la transferencia de datos desde un elemento de entrada (**EE**), a uno o varios elementos de salida (**ES.**) La MC se puede dividir en cuatro bloques funcionales, para pruebas y monitoreo.

- Elementos de entrada.
- Bus común y su controlador.
- Elementos de salida.
- Microcontrolador y la computadora.

2.2.1Elemento de entrada

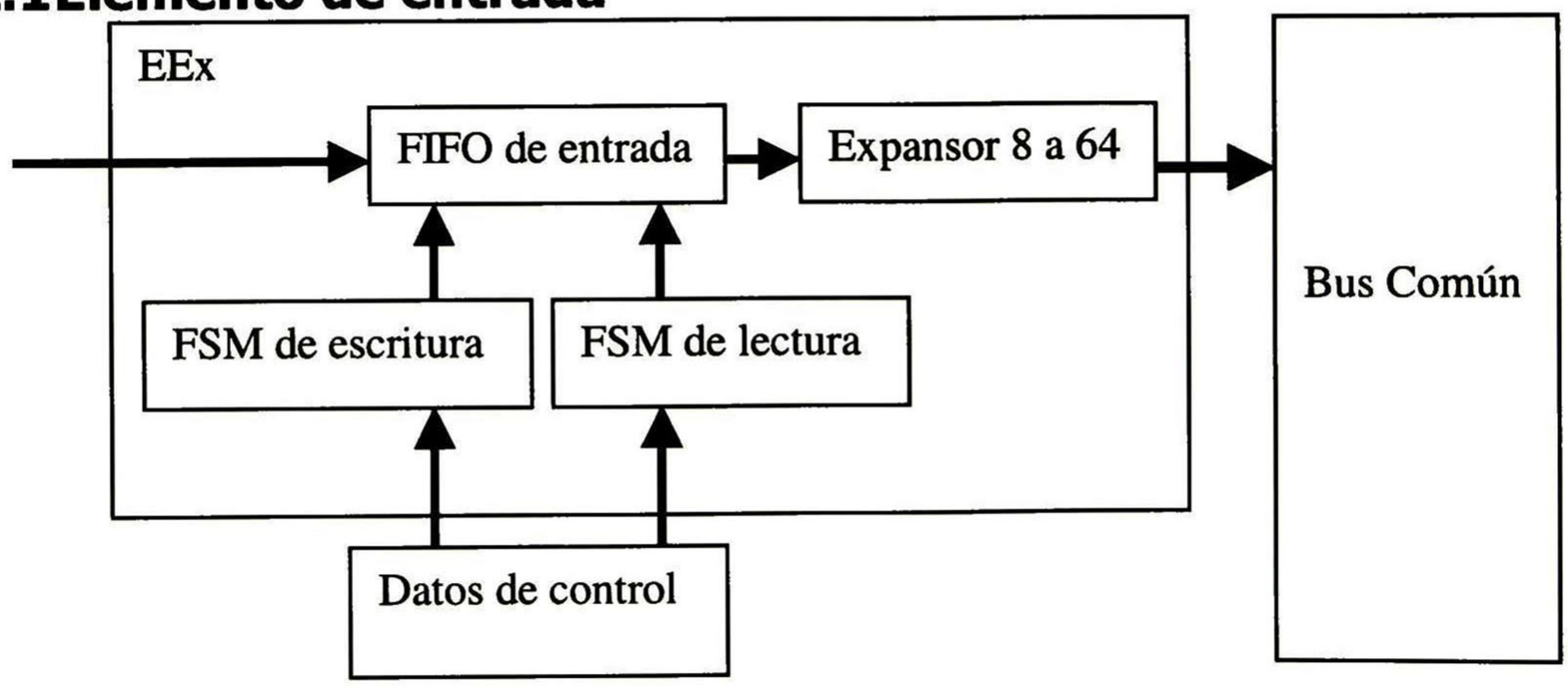


Figura 2 *Diagrama a bloques del elemento de entrada.*

Se compone de:

- FIFO de entrada.

- FSM de escritura.
- FSM de lectura.
- Expansor de 8 a 64 bits.

La *FIFO de entrada:* recibe los datos desde la interfaz conectada, controlada por la FSM de escritura quien recibe la señalización externa para aceptar una escritura a la FIFO.

La *FSM de escritura:* es quien, en asociación con la FSM de lectura, permite el almacenamiento de datos en la FIFO.

La *FSM de lectura:* es quien, una vez que se acepta una celda completa del exterior, la transfiere al expansor de 8 a 64 bits para posteriormente enviar datos a los ES.

Expansor de 8 a 64 bits: toma 8 bytes de información de la FIFO y los coloca a disponibilidad del bus común para la transferencia de datos por la MC.

Bus Común

El bus común es el medio de transferencia de datos a través de la MC, desde un EEx hasta un ESx. Tiene un ancho de 64 bits. La información que llega a cada EE tiene un formato específico. El formato de trama de entrada es el siguiente:

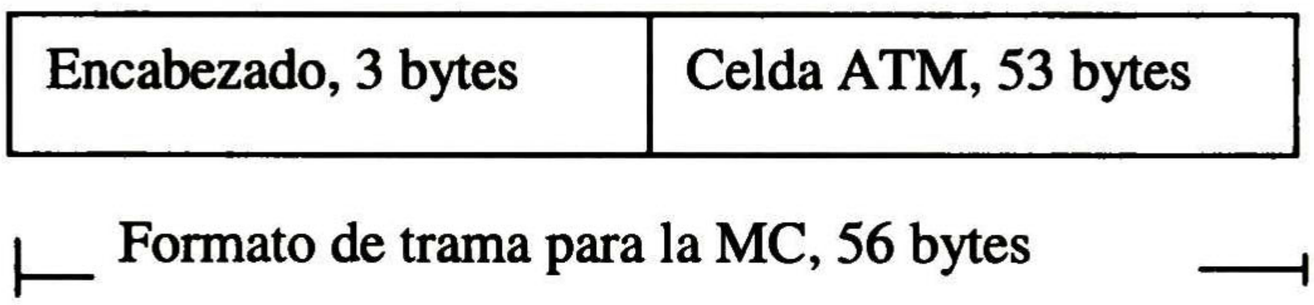


Figura 3 *Formato de trama para la MC*

Encabezado: tiene la información de enrutado, agregado por el CL.

Carga útil: Contiene la información a enrutar, por la MC.

Como se muestra en la Figura 3, la trama completa tiene 56 bytes (448 bits). Como se vio en la sección 2.2.1 la trama se descompone en secciones de 64 bits para ser transferido por el bus, se necesitan 7 ciclos de bus para transferir una celda completa desde un EE hasta un ES. A cada bloque de 64 bits se le conoce como *microcelda*. El controlador de bus asigna un espacio o ranura de tiempo a cada EE para colocar una microcelda en el bus, a este proceso se le conoce como *asignación rotativa del bus ver 2.2.2*. Esta asignación consiste en darle a cada EE acceso en tiempos diferentes; por ejemplo, el EE_i coloca, no siempre, su microcelda en la ranura de tiempo *i,* en cada ciclo de bus la ranura de tiempo se mueve un lugar hacia delante, realizando así la asignación rotativa del bus.

Datos de control

Estos datos son los cuales comunican al microcontrolador por medio de sus puertos de entrada/salida.

2.2.2Bus Común

Para transferir datos el bus común asigna ranuras de tiempo a cada EE. En este caso, cada trama se divide en 7 subtramas, cada subtrama a su vez se divide en 16 microceldas. Estas se transfieren en un tiempo igual a 17 microceldas, donde la 17 ranura de tiempo es para que los ES tomen la información que se dirija a ellos. Por esto se llama ranura de salida.

La razón de la 17 ranura de tiempo se da por la falta de memorias de doble puerto, con la utilización de memorias de doble puerto los ES podría tomar su información al mismo tiempo que el EE coloca su información.

La asignación de BUS se controla por un elemento llamado Bloque de control general (BCG) donde éste maneja 17 estados, 1 por microcelda. Cada EE lleva su control para acceder al bus común.

Asignación rotativa del bus.

Existe otro tipo de asignación de bus además del que se expone al principio de esta sub sección, a continuación se presenta un algoritmo diferente para la manera en que se comparte un medio compartido como lo es el bus de la MC.

El tipo de asignación del bus de la MC es una asignación estática, es decir, un espacio de tiempo es asignada de manera constante a un EE para transferir su información hasta el ES pero este tipo de asignación desperdicia espacios de tiempo, esto cuando un puerto no tiene datos para ser transferidos, sin embargo si la asignación de espacios fuese dinámica podrían ser utilizados para otros puertos estrictamente activos.

En [12] se muestran los siguientes algoritmos de asignación de bus, se hace referencia a ellos para dar una un ejemplo que existe para la asignación de bus, estática y dinámica. Sólo se mencionaran los siguientes dos:

Asignación rotativa y cíclica para todos los EE (estática)

Asignación dinámica y secuencial para todos los EE (dinámica)

Rotativa y cíclica para todos los EE (estática)

Este tipo de asignación se basa en el principio de que la probabilidad de otorgar un espacio de tiempo a un EE determinado no depende de donde se conecte este elemento en la MC, esta manera combina la forma *cíclica y rotativa* de asignación, así la probabilidad de entrar al bus y colocar datos en el ES es igual para todos los EE.

Este algoritmo se basa el algoritmo *round-robin* el cual asigna un espacio de tiempo *i* para tener acceso al medio, en siguientes oportunidades el EE coloca su información en el espacio de tiempo diferente, por lo general *i+1*.

Dinámica y secuencial para todos los EE (dinámica)

Este algoritmo tiene como meta distribuir los tiempos de acceso al bus sólo entre los elementos activos conectados a los EE.

Este algoritmo tiene la complejidad de la detección de los elementos conectados y la manera de distribuir los accesos al bus, en esto radica la complejidad del algoritmo.

En resumen el desempeño de la MC está en función de los puertos conectados y la eficiencia del bus sobre el algoritmo usado.

2.2.3Elemento de salida

El ES toma los datos del bus común en formato de microceldas en su ranura de tiempo, almacenando la información en colas temporales. Cuando más de un EE manda información a un mismo ES, se le conoce como el fenómeno de contención. El uso de memorias temporales resuelve el problema.

Cada ES tiene un identificador único con el cual comparar contra la etiqueta de enrutado de la primer microcelda de una trama de entrada, para así poder determinar sí debe tomarla, en caso contrario la ignora.

Se presenta a continuación un diagrama a bloques de la arquitectura del ES.

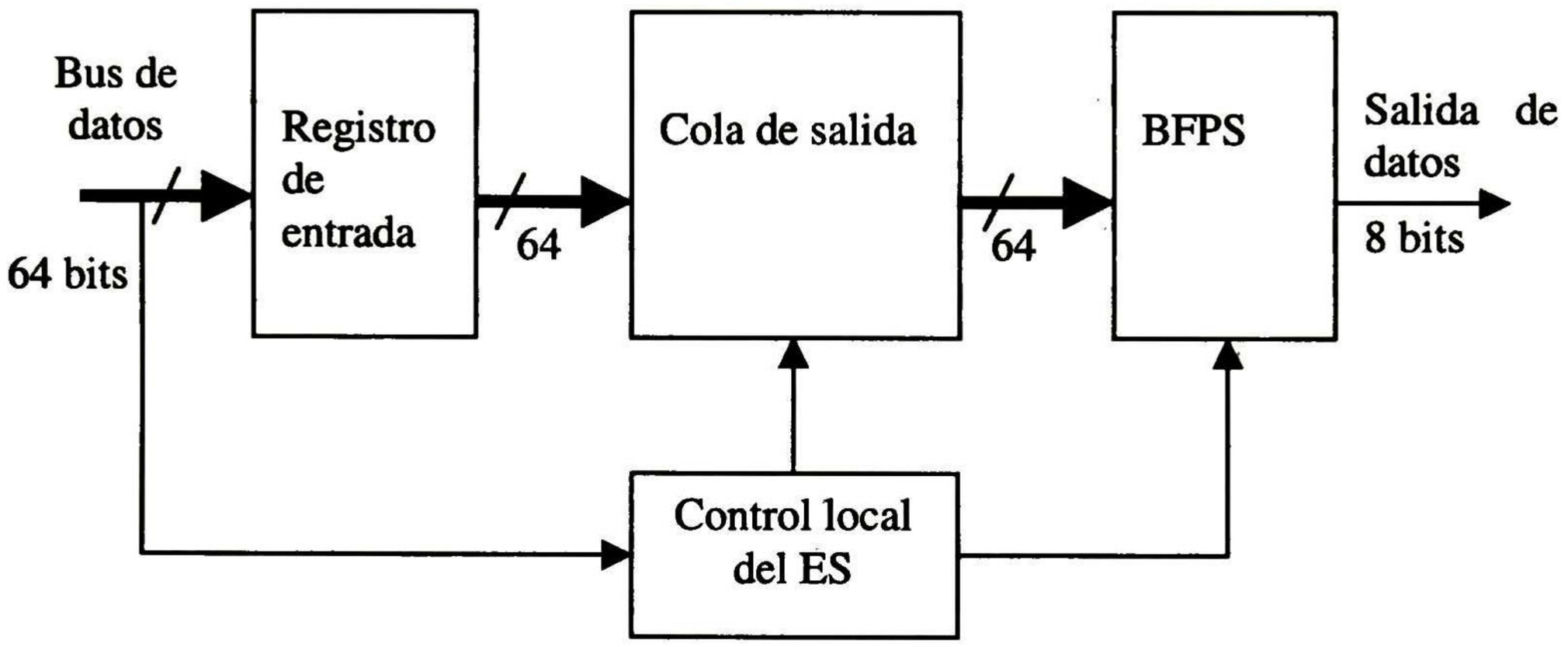


Figura 4 *Arquitectura del ES*

La arquitectura mostrada en Figura 4 muestra los 4 componentes principales, los cuales se describen a continuación:

El *registro de entrada* es el lugar donde se almacena temporalmente los datos que se reciben del bus común.

La *cola de salida* es una con una capacidad de 256 palabras de 64 bits. Se usa para el problema de la contención, presente en este tipo de arquitecturas.

El *BFPS* Consiste en dos registros de 64 bits, los cuales se configuran para lectura y escritura donde, en una transferencia, uno de ellos recibe los datos de la cola de salida al mismo tiempo que el segundo registro presenta los datos para su transferencia al exterior, de tal manera que en la próxima transferencia los registros intercambian su función.

El *control local del ES* consiste en dos FSM que permiten escribir a la cola de salida y habilitar y controlar el BFPS, de tal manera que pueda transferir los datos desde la cola de salida al CL que esta conectado al ES.

2.2.4 Microcontrolador y la computadora

El microcontrolador tiene las funciones de elemento de monitoreo y diagnóstico para la etapa de pruebas de la MC. Junto con el microcontrolador, la computadora completa esta etapa. Se utiliza un protocolo de comunicación y una interfaz visual para esta tarea.

En el capítulo 3 se presenta el protocolo.

Para poder hacer un rediseño correcto, es necesario su documentación. A continuación se presenta un resumen de lo más importante cuando se planea rediseño. El lector puede encontrar la información aquí presentada en el apéndice B. Además trataremos el punto de la metodología de la reingeniería y codiseño del hardware y software.

2.3 Análisis de factibilidad para el desarrollo de la interfaz UTOPIA

La inclusión de UTOPIA a esta MC involucra diferentes aspectos, tales como factibilidad económica, operativa y tecnológica, los cuales a continuación desglosaremos.

Factibilidad económica.

El diseño actual es de alrededor de $15,000 dólares, por lo que el costo no variará demasiado. Además, se tiene la mayor parte de los componentes necesarios para ser usados en una tesis futura, por lo que este proyecto es posible.

Factibilidad operativa.

Otro punto importante es la forma en que llega la información a la MC, el CL agrega 3 bytes de enrutado al paquete. La MC necesita de estos 3 bytes para poder hacer el enrutado de las celdas. UTOPIA es una interfaz de ATM, la cual consta de una trama fija de 53 bytes, lo que no permite la existencia de estos 3 bytes extras. Por lo tanto se tendría que calcular el enrutamiento por parte de la MC. Tal labor es complicada y el diseño de la MC no fue planeado para esta opción, la MC debería poder realizar el enrutamiento con los bytes de VPI y VCI, (parte del encabezado de la trama ATM.)

Aun así, el problema es cálculo del enrutamiento provocaría que se tuviera una memoria bastante grande (12 bits X 1024 como mínimo razonable), o bien una interfaz a la PC o cualquier otro dispositivo que realizara el trabajo, por lo que una nueva interfaz como se escoja, implica un cambio en el PCB.

Factibilidad técnica.

Sí es realizable, aún con todos estos cambios. Se especificará el diseño de una MC que involucre a la interfaz UTOPIA. Ver la siguiente sección donde se explica brevemente la interfaz.

2.3.1Interfaz UTOPIA nivel 1

UTOPIA (del inglés *Universal Test & Operations Physical interface for ATM*) es un estándar para poder conectar elementos que soportan ATM.

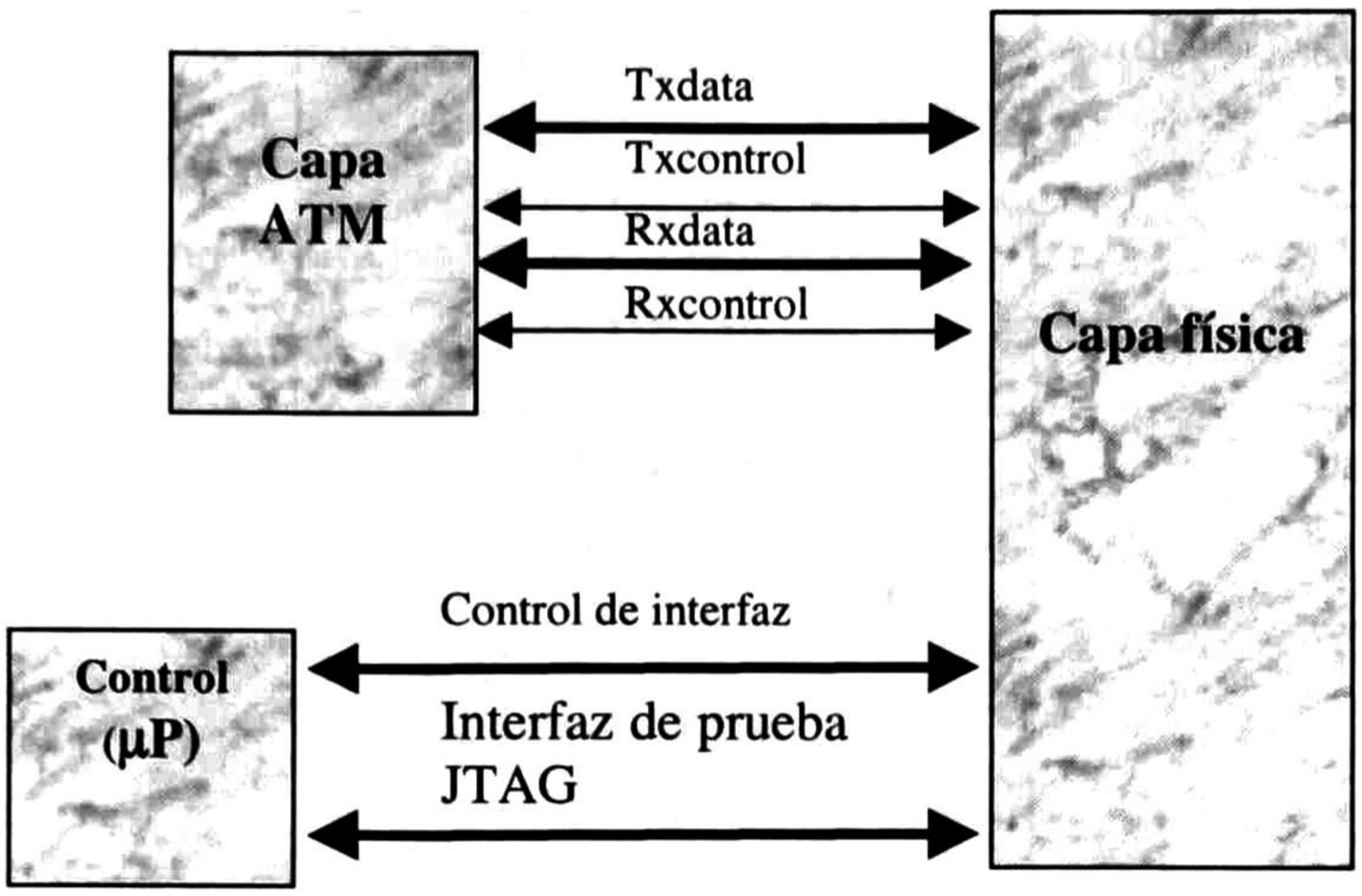


Figura 5. *Diagrama de interfaz de UTOPIA*

El objetivo fundamental de este estándar es definir una interfaz entre ATM y capas físicas de subsistemas de ATM. Existen muchas capas físicas definidas para ATM (definidas por ANSI, ITU y ATM forum) cada una de estas capas es potencialmente soportable por una capa común de ATM. Consecuentemente, es una motivación de apoyar el desarrollo de ATM a través de múltiples tipos de capas físicas.

La interfaz ATM-PHY no es una especificación de interface de usuario a red (*user network interface* UNI) ni tampoco interfaz de red a red (*network network interfaz* NNI). Por lo tanto las ventajas económicas de un soporte de múltiples capas físicas comunes, no pueden ser ignoradas. Correspondientemente, otra motivación para le desarrollo de esta interfaz en obtener una implementación de tal interfaz.

Una meta es soportar velocidades de 100Mbps hasta 155Mbps, con la misma interfaz común, requiriendo solo una sola trayectoria de datos de 8 bits. Esto con la intención de proveer soporte de para velocidades mayores, por ejemplo 622 Mbps, obteniéndose extendiendo el ancho del datapath e incrementar la velocidad.

Existen dos maneras de comunicación, por celdas y por bytes, con la posibilidad de aumentar a 16 bits de transferencia.

Formato de celda

El formato de celda es:

Header 1(1 byte)
Header 2(1 byte)
Header 3(1 byte)
Header 4(1 byte)
UDF (user defined) **(1 byte)**
Payload 1(1 byte)
:
Payload 48(1 byte)

Tabla 1 *Formato de celda de UTOPIA*

Interfaz de transmisión

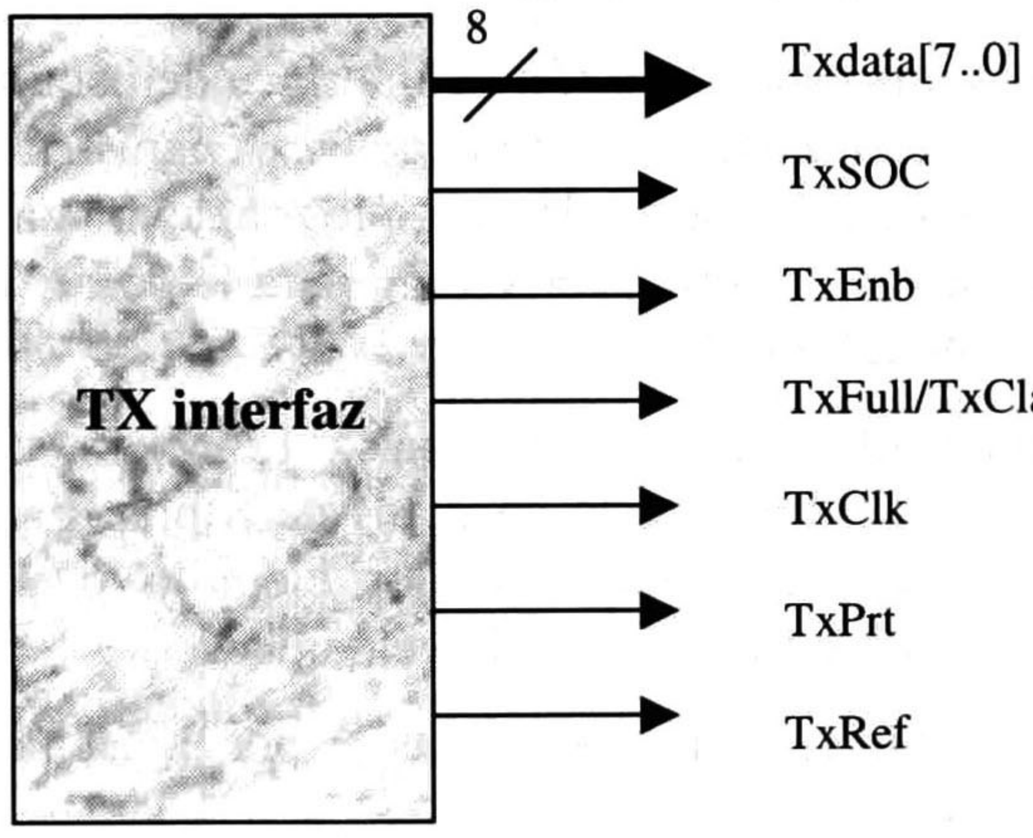


Figura 6 *Interfaz de transmisión para UTOPIA*

A continuación se describen las señales que interactúan en la interfaz de transmisión.

Señal	Descripción
TxData[7..0]	Dato proveniente de la capa física ATM.
TxSOC	Comienzo de Celda
TxEnb	Activa en bajo mientras se tengan datos válidos

TxFull/TxClav	Señal de control para definir si los datos son válidos o no.
TxClk	Reloj de datos de entrada.
TxPrty	Paridad par de los datos TxData
TxRef	Referencia de transmisión, para propósitos de sincronización.

Tabla 2 *Señales de transmisión.*

Interfaz de recepción

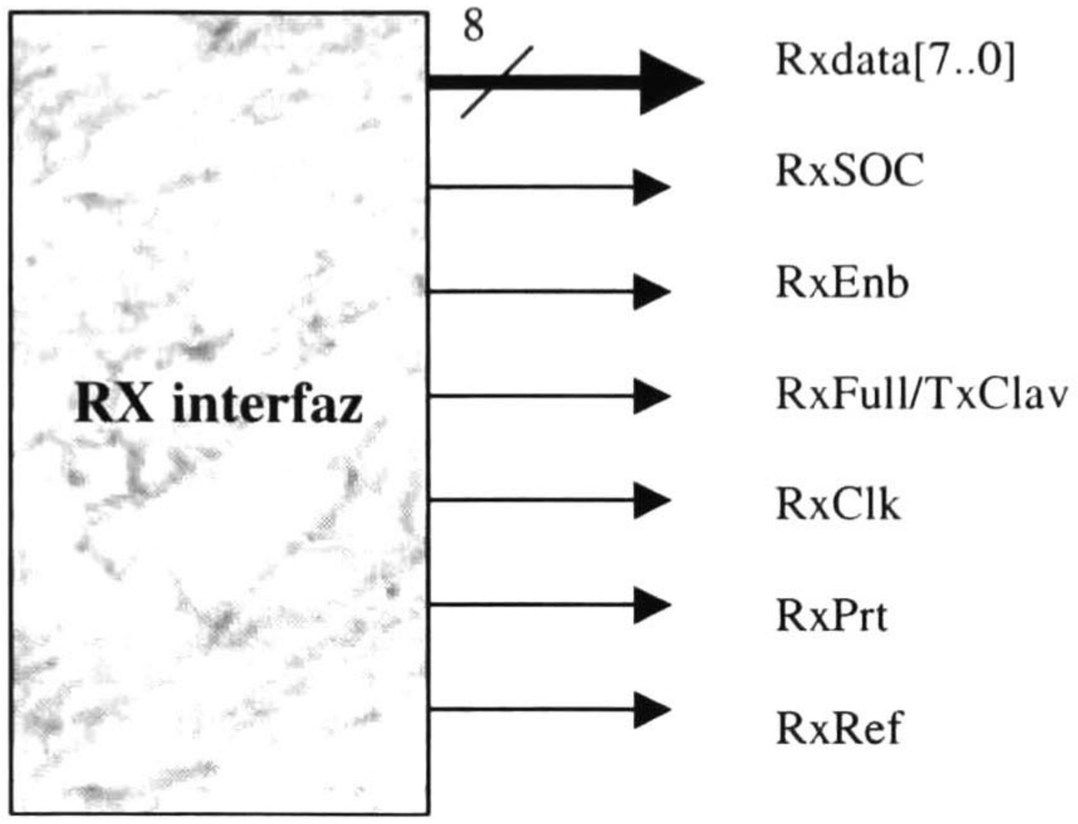


Figura 7 *Interfaz de recepción para UTOPIA*

A continuación se describen las señales que interactúan en la interfaz de Recepción.

Señal	Descripción
RxData[7..0]	Dato proveniente de la capa física ATM.
RxSOC	Comienzo de Celda
RxEnb	Activa en bajo mientras se tengan datos válidos
RxEmpty/TxClav	Señal de control para definir si los datos son válidos o no.
RxClk	Reloj de datos de entrada.
RxPrty	Paridad par de los datos RxData
RxRef	Referencia de transmisión, para propósitos de sincronización.

Tabla 3 *Señales de recepción*

2.4 CUT

El CUT (convertidor UTOPIA/TOPIA) La solución es integrar *UTOPIA* a la MC. Se ha logrado agregar un nuevo módulo que realice esta función. El problema es dónde colocar el módulo, lo cual es la parte central de lo que se presenta:

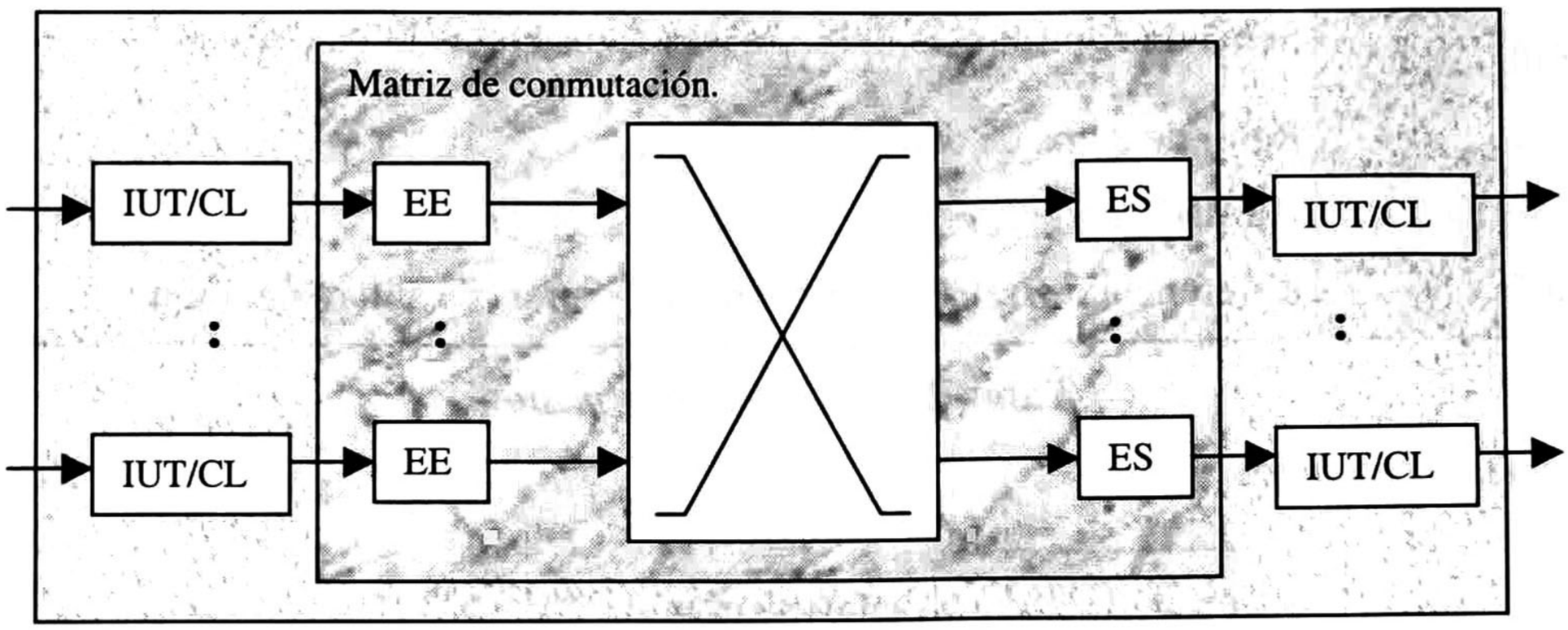


Figura 8 *CUT dentro del CL.*

La Figura 8 muestra la solución 1 "*inclusión de CL con soporte de UTOPIA*", la cual consiste en implementar al CUT dentro del diseño de un CL para que pueda ser conectado a la MC.

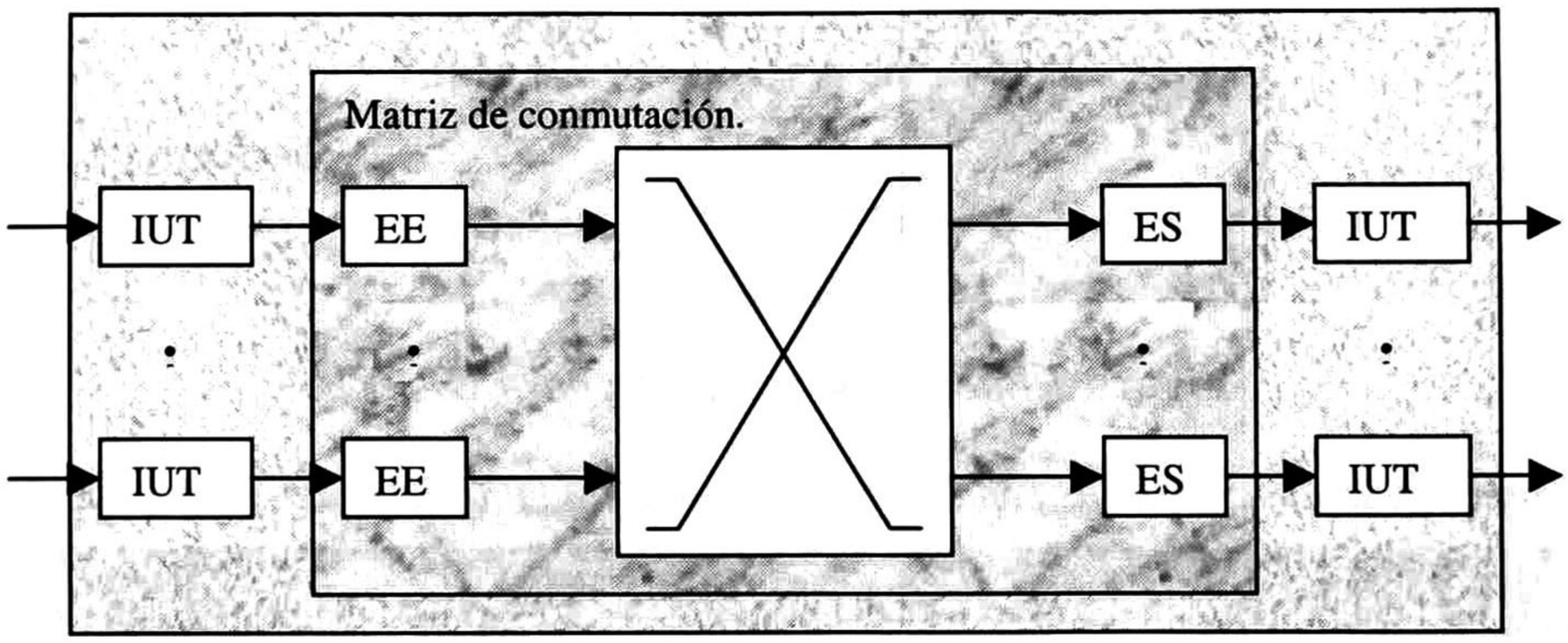


Figura 9 *CUT como interfaz.*

La Figura 9 muestra la solución 2, "*Integrar UTOPIA con mínimos cambios*". Se plantea una MC para que pueda soportar UTOPIA con la integración de nuevos dispositivos, pero sin hacer modificaciones considerables a la actual MC.

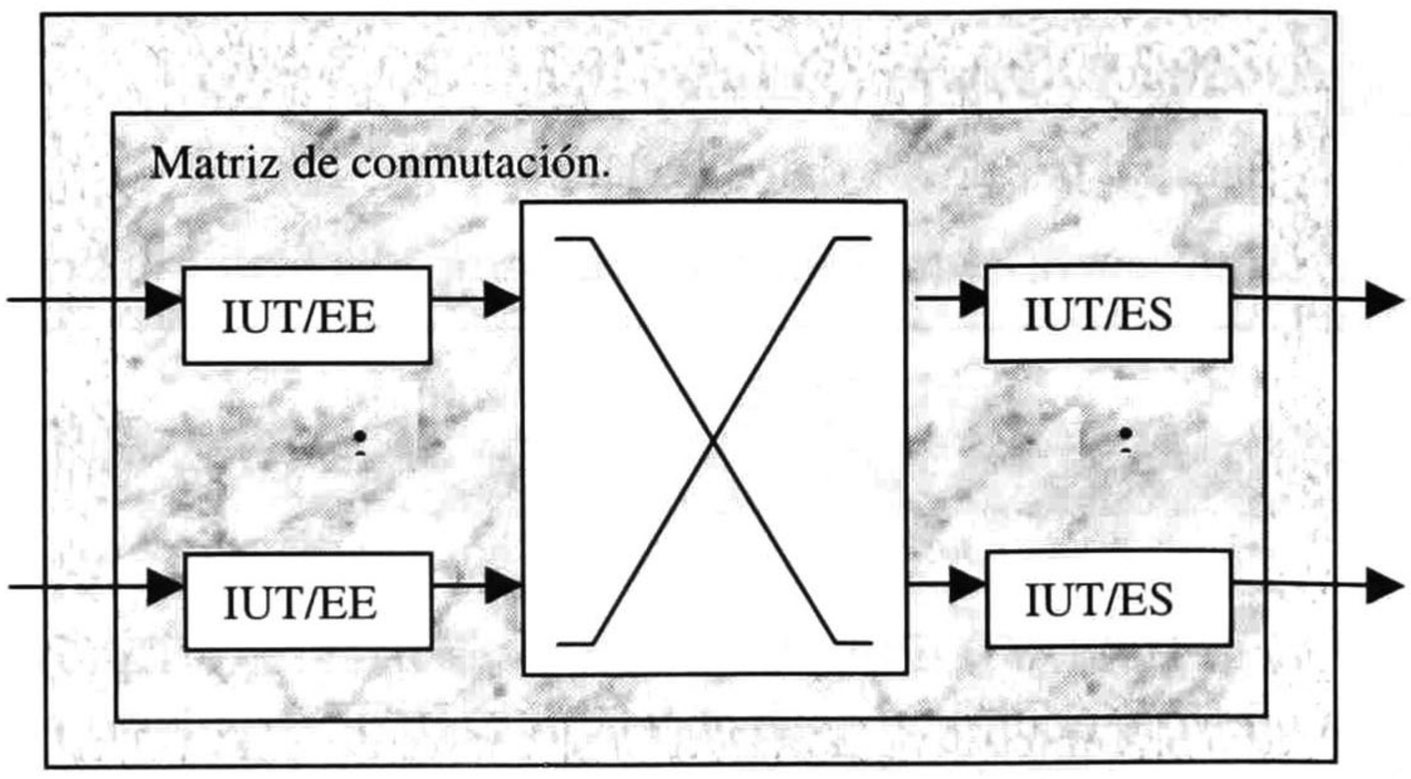


Figura 10 *CUT dentro del EE y ES.*

La Figura 10 muestra la solución 3, *"MC con soporte en UTOPIA"*. Esta consiste en implementar a CUT dentro de un nuevo diseño de MC.

En lo sucesivo se mostrará a detalle estas soluciones que son:

- Inclusión de CL con soporte de UTOPIA.
- Integración UTOPIA con mínimos cambios.
- MC con soporte en UTOPÍA.

2.5 Inclusión de CL con soporte de UTOPIA

La inclusión de un módulo para realizar la conversión de TOPIA a UTOPIA es la forma en que se ha desarrollado [3] Tal documento muestra un CL que puede realizar la tarea de conectar elementos que soporten UTOPIA nivel 2 al protocolo TOPIA. Basándose en esto se conecta la MC a elementos con este estándar. Ver [3] para más detalles.

2.6 Integración UTOPIA con mínimos cambios

Se pretende desarrollar un bloque que convierta de UTOPIA/TOPIA. Lo descrito en [3] utiliza este bloque, llamado CUT, para su funcionamiento.

Se expondrá esta opción mostrando el módulo que describe al CUT.

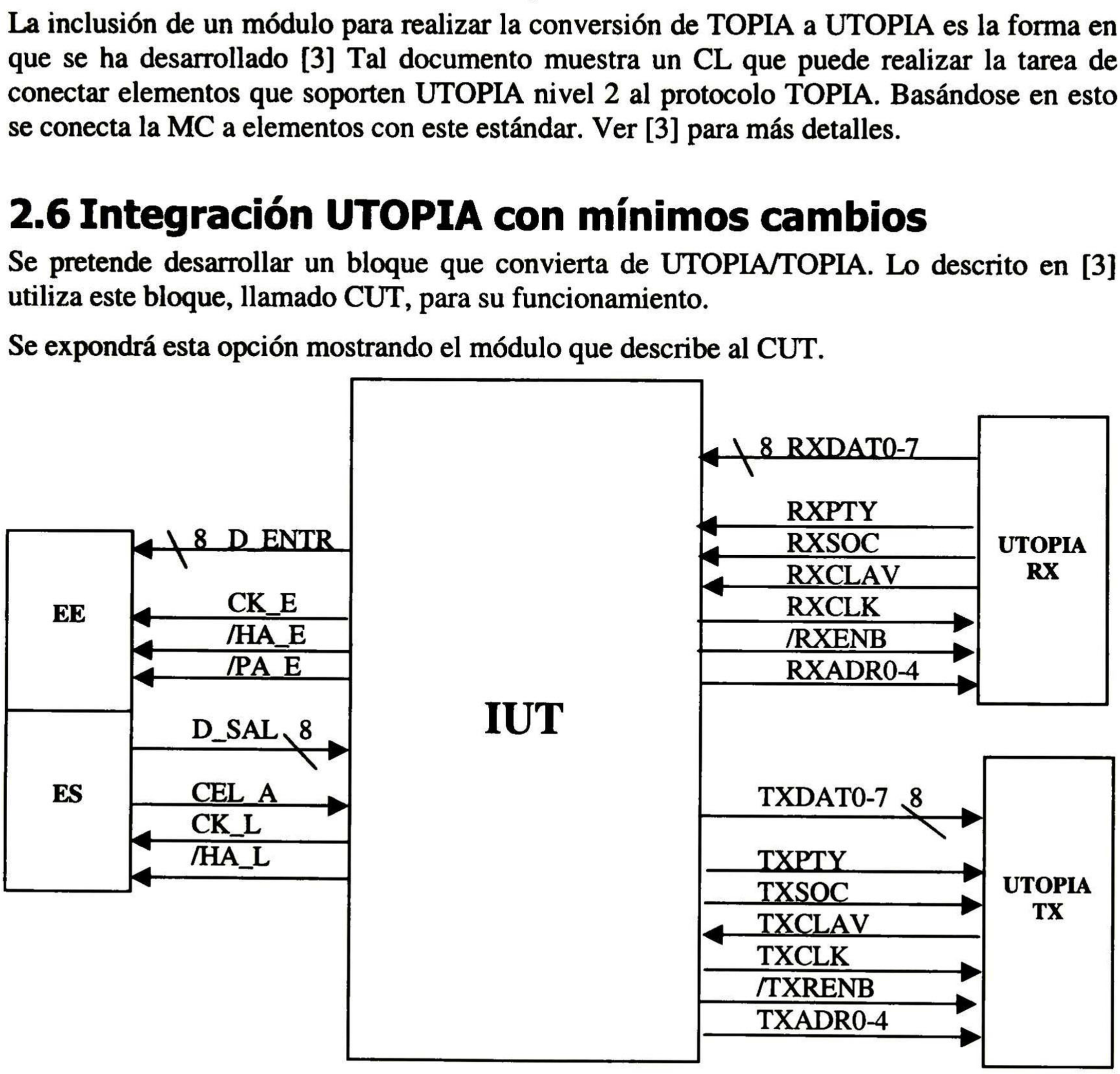


Figura 11 *Diagrama a bloques del CUT*

Este módulo se implementó en un FLEX 10K100, el cual trabaja con un reloj de 20 Mhz. Al ver cómo se ha especificado [1], se observa un microcontrolador que monitorea

elementos como registros, alarmas del circuito SAR, por mencionar algunos. Ahora se planea desarrollar un elemento CUT reduciendo sus funciones a:

- Pruebas BIST para la memoria interna, donde se encuentra la TR.
- Activación / desactivación de los LoopBacks.

A continuación se presenta una descripción de las líneas del chip CUT.

Linea	Tipo	Ancho de bus	Descripción
D_ENTR	Salida	8	Bus de datos de salida al EE
CK_E	Salida	1	Reloj proporcionado al interfaz TOPIA
/HA_E	Salida	1	Habilitador de escritura (CL)
/PA_E	Salida	1	Habilitador de escritura al puerto
D_SAL	Entrada	8	Bus de datos de entrada del ES
CEL_A	Entrada	1	Línea que indica cuando esta disponible una celda completa
CK_L	Salida	1	Reloj proporcionado al ES
/HA_L	Salida	1	Habilitador de lectura del CL
RX_DAT	Entrada	8	Bus de datos de entrada de UL1 (Utopía nivel 1)
RXPTY	Entrada	1	Línea que indica la paridad de los datos que provienen UL1
RXSOC	Entrada	1	Línea que indica el comienzo de la trama
RXCLAV	Entrada	1	Línea que indica que UL1 esta lista para transferir
RXCLK	Salida	1	Reloj con el que realiza las transmisiones
/RXENB	Salida	1	Línea que indica que CUT esta listo para transferir
RXADR0	Salida	5	Bus de direcciones para seleccionar la capa física que se maneja
TXDAT	Salida	8	Bus de datos de salida e la interfaz
TXPTY	Salida	1	Línea que indica la paridad de los datos que salen de CUT
TXSOC	Salida	1	Línea que indica el comienzo de la trama
TXCLAV	Entrada	1	Línea que indica que UL1 esta lista para transferir

TXCLK	Salida	1	Reloj con el que realiza las transmisiones
/TXRENB	Salida	1	Línea que indica que CUT esta listo para transferir
TXADR	Salida	5	Bus de direcciones para seleccionar la capa física que se maneja

Tabla 4 *Descripción de las líneas del chip CUT*

A continuación se puede observar los diagramas de tiempo para las diferentes interfaces usadas en el CUT.

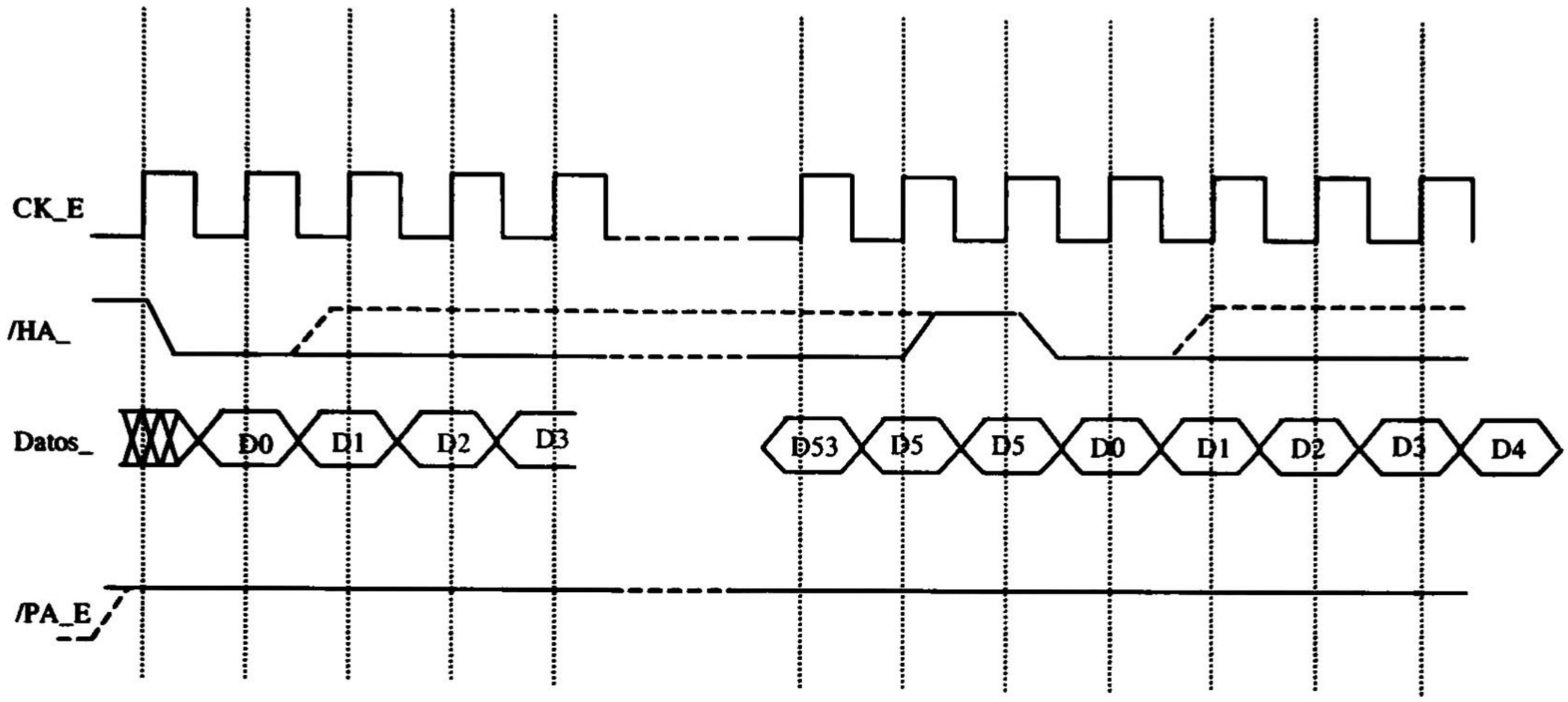


Figura 12 *Diagrama de tiempos de la interfaz CL-MC*

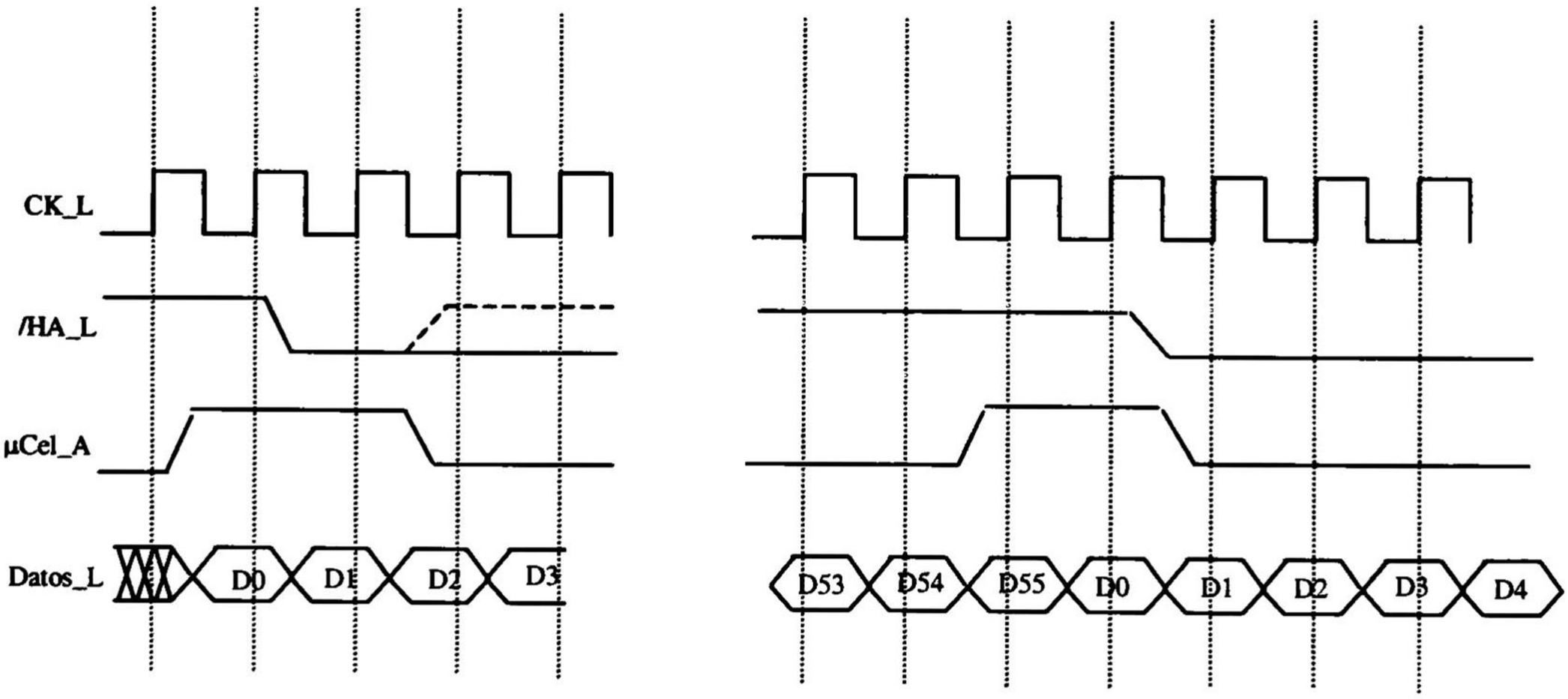


Figura 13 *Diagrama de tiempos de la interfaz MC-CL*

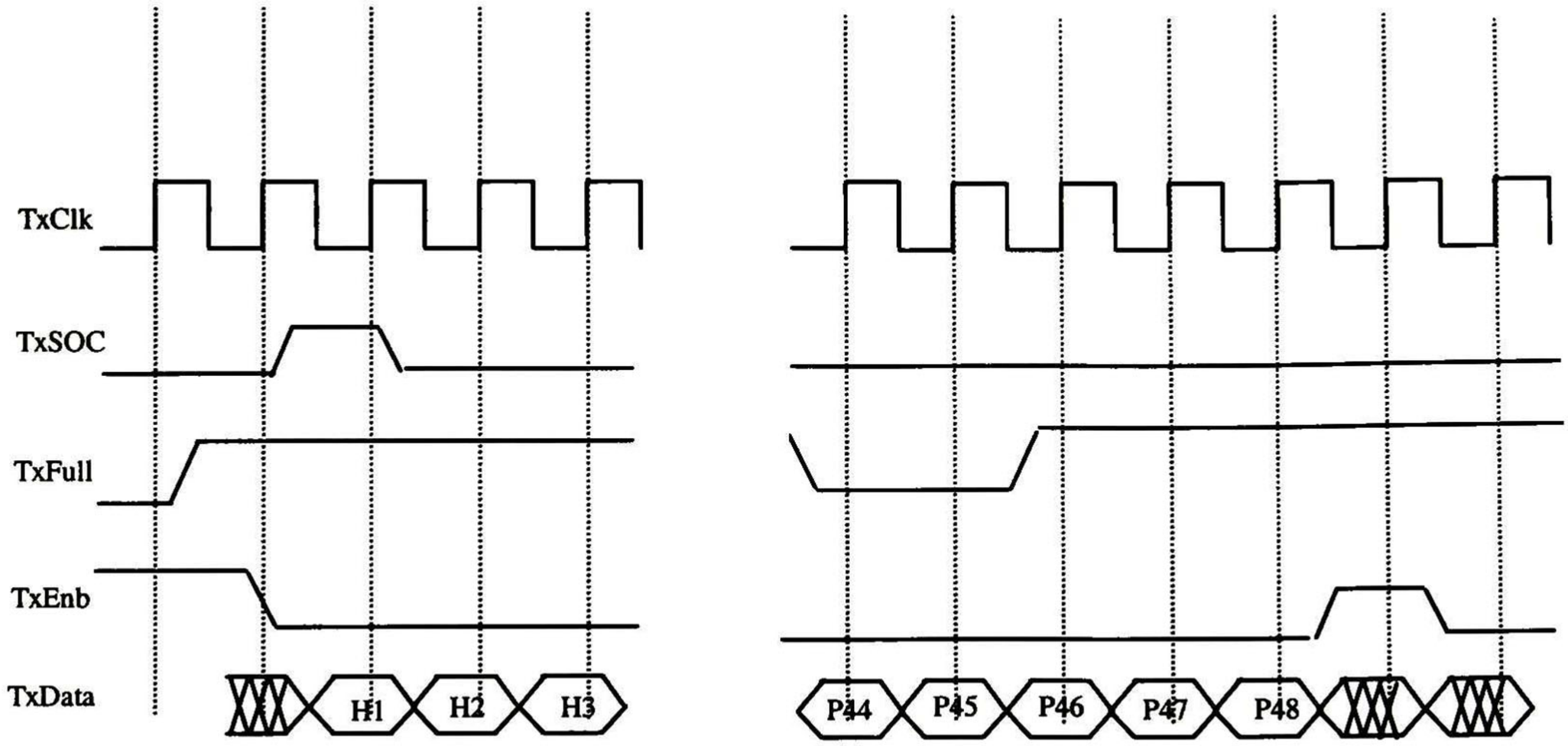


Figura 14 *Diagrama de tiempo para transmisión en Utopía nivel 1*

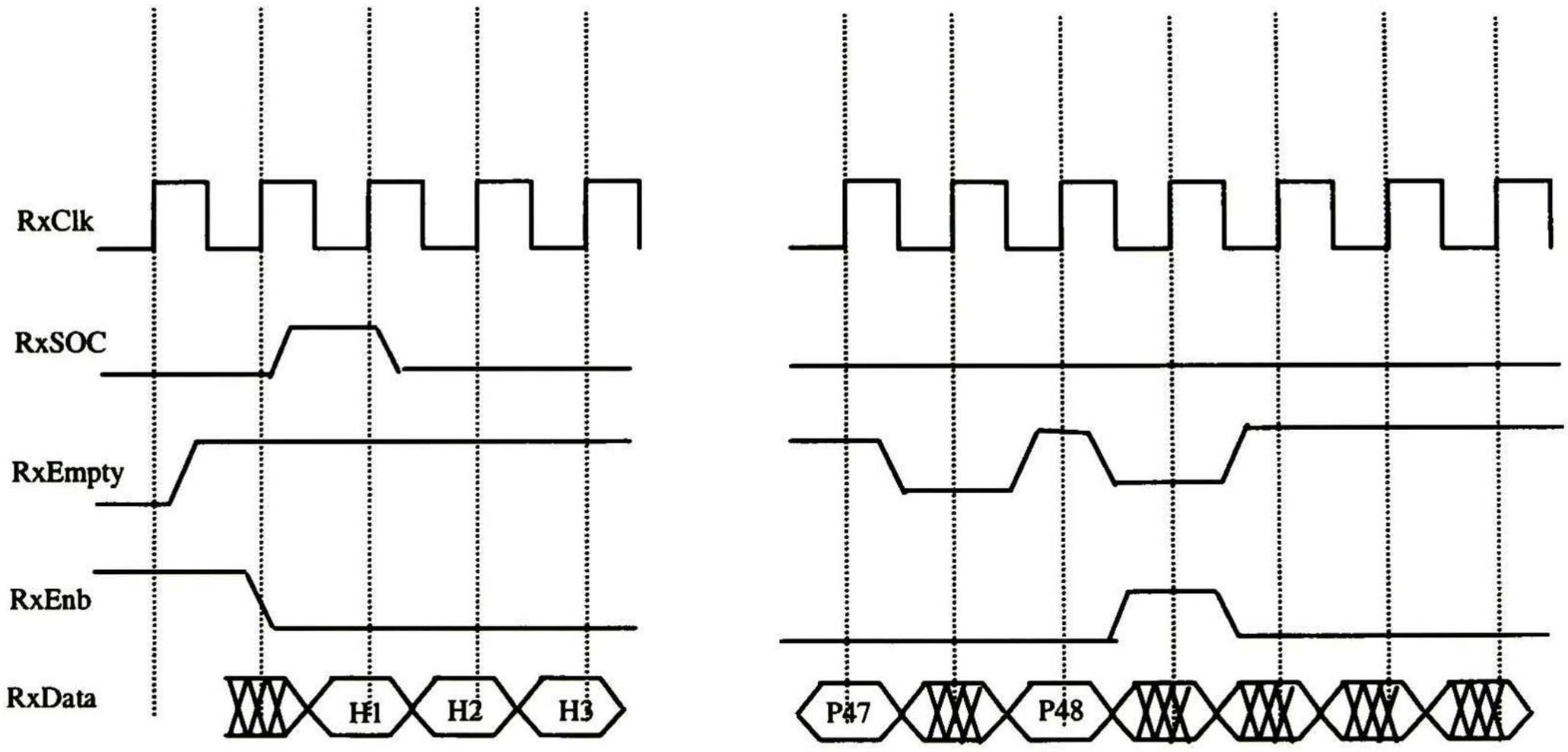


Figura 15 *Diagrama de tiempo para recepción en Utopía nivel 1*

2.7 MC con soporte en UTOPIA

Esta solución implica agregar el módulo de CUT a la MC. Como se sabe, los elementos EE y ES comparten el mismo PLD, el cual es un FLEX 10K100. Ver [1].

Se propone agregar este módulo al PLD. Actualmente el módulo se ha programado en VHDL y ocupa el 25% del mismo. Como se vio en [1], la ocupación del chip es del 30% por lo tanto es factible la inclusión de este en el FLEX 10K100. Ahora se plantea la nueva estructura, teniendo en cuenta que no deben afectarse las conexiones ya establecidas de la estructura actual.

2.8 Resumen del capítulo

Este capítulo se basa en la documentación realizada para llevar a cabo un rediseño. La información que se presenta en este capítulo se complementa con la presentada en el Apéndice B.

La inclusión de UTOPIA a esta MC involucra diferentes aspectos, tales como *Factibilidad económica,* que involucra lógicamente los costos del rediseño y todas sus implicaciones; *Factibilidad operativa,* donde el problema más complejo que se puede atacar es debido a las modificaciones al incluir UTOPIA al diseño (ejemplo, el incremento en el tamaño de la memoria usada); *Factibilidad técnica,* donde el problema fundamental trata de la posibilidad de realizar el diseño involucrando a la interfaz UTOPÍA. Para tal problema, la solución es integrar UTOPIA a la MC, agregando un nuevo módulo que realice esta función.

Se muestran soluciones donde se especifican:

- La inclusión de CL con soporte de UTOPIA.
- La integración UTOPIA con mínimos cambios.
- MC con soporte en UTOPIA.

La inclusión de un módulo para realizar la conversión de TOPIA a UTOPIA, es la forma en que se ha efectuado la solución. Se muestra un CL que puede realizar la tarea de conectar elementos que soporten UTOPIA nivel 2 al protocolo TOPIA.

Se pretende desarrollar un bloque que convierta de UTOPIA / TOPIA, el cual utiliza un bloque llamado CUT para su funcionamiento. Además, se expondrá esta opción mostrando el módulo que describe al CUT.

Se muestra un módulo implementado en un FLEX 10K100, del cual ahora se planea desarrollar un elemento CUT, reduciendo sus funciones a:

- Pruebas BIST para la memoria interna, donde se encuentra la TR.
- Activación de los LoopBacks.

Por lo tanto, se muestra que esta solución implica agregar el módulo de CUT a la MC, como se sabe los elementos EE y ES comparten el mismo PLD el cual es un FLEX 10K100.

Ahora se propone agregar este módulo al PLD, el cual actualmente se ha programado en VHDL y ocupa el 25% del mismo, lo cual es completamente factible. Se plantea la nueva estructura, teniendo en cuenta que no se debe de afectar las conexiones ya establecidas de la actual.

3 Desarrollo de un protocolo de comunicación entre firmware y una aplicación en Windows

En las siguientes secciones se muestra el protocolo desarrollado.

3.1 Protocolo de comunicación para el puerto serie

Es necesario reiterar que la necesidad de un mecanismo para asegurar la comunicación entre la PC y el microcontrolador es base del éxito de las pruebas, dado que el protocolo asegura que la comunicación es robusta y confiable.

Es importante que la información que se transmite entre aplicaciones de equipos distintos esté siempre libre de errores, lo más posible. En nuestro caso es necesario el desarrollo de protocolos de comunicación entre la capa física y una aplicación de Windows. Este protocolo se lleva a cabo principalmente añadiendo cierta redundancia controlada a la información a transferir, la misma que nos permite detectar errores cometidos en la recepción de la información. Una vez detectado un error, se pueden seguir dos estrategias para corregirlo: en la primera, la misma redundancia es quien permite la corrección de ciertos errores; en la segunda, el receptor solicita la retransmisión del bloque de datos recibido erróneamente.

El objetivo es diseñar un protocolo flexible y confiable para una comunicación punto a punto, que sea transparente tanto a su capa superior como a su capa inferior, y que además vea a esas capas de una forma igualmente transparente.

No confunda el concepto de redundancia con repetición, la redundancia de información es información adicional que permite control de la información, como puede ser detectar y/o corregir errores en la trama que se transmite, en este protocolo la redundancia de información se da con un código de CRC (será explicado posteriormente) el cual es información adicional para detección de errores.

3.1.1 Principios de diseño

La necesidad de un protocolo de comunicaciones se basa en la necesidad de confiabilidad de los datos a transferir. Este protocolo se aplica entre la comunicación de la PC y el microcontrolador, dando así una fortaleza y seguridad en el plan de pruebas. La necesidad de desarrollo considera que la comunicación serie tiene un factor de error considerable para ser tomado en cuenta.

El protocolo de comunicaciones que se diseñó puede volver a ser utilizado para otro tipo de aplicaciones, donde el medio físico y de usuario pueden cambiar, dependiendo de la aplicación. Se menciona como ejemplo la aplicación desarrollada para transmitir información por tonos DTMF (Dual Tone Multi Frecuency, utilizado comúnmente para servicios telefónicos digitales) desarrollado y expuesto en [13]. El protocolo muestra así que puede ser utilizado ampliamente.

Las capas de usuario y de medio físico así pues pueden cambiar haciendo transparente el uso del protocolo de comunicaciones.

Teniendo en cuenta que las aplicaciones de Internet facilitan y flexibilizan las tareas de control y monitoreo de tareas, se puede tener para futuras aplicaciones la facilidad de agregar un elemento de control de MODEM, para de esta manera tener la posibilidad de ejecutar el monitoreo y control a distancia. Esta posibilidad puede desarrollarse haciendo una capa extra que pueda manejar el MODEM de manera transparente para el medio físico.

El protocolo fue diseñado partiendo del modelo de referencia OSI, a un nivel de la capa de enlace de datos, como se muestra en la Figura 16. Este no es un diseño propiamente de OSI, sin embargo, se tomó la idea básica para el diseño, aunque no es un diseño complejo.

El protocolo tiene la capacidad de:

- Acuse de recibo en la recepción.
- Retransmisión de N veces (configurable) de la trama en caso de errores en la transmisión.
- Tiempo de espera (configurable) para establecer nuevamente la conexión.
- Retransmisión en caso de error en la recepción detectada por un código de CRC.
- Detección de tramas perdidas mediante un código de secuencia de transferencia.
- Buffer de 255 bytes en medio físico.
- La recepción es por un servicio de interrupción.

Para tener más en claro la conceptualización de las capas del protocolo, se muestra a continuación la siguiente figura.

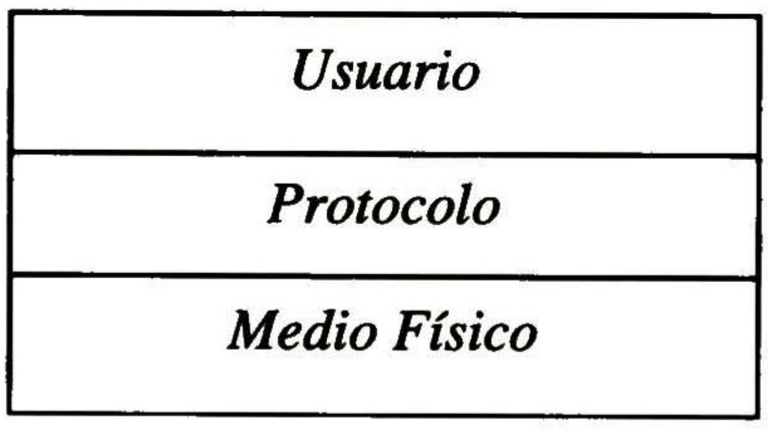


Figura 16 *Conceptualización en capas del protocolo y sus capas adyacentes*

Las funciones de cada una de estas capas son:

Medio físico: Se encarga de la transmisión física, aunque de manera no confiable, de los bits de información.

Protocolo: Se encarga de convertir a la capa física en un medio confiable de transmisión, llevando control de los errores y de la secuenciación de la información. Con este fin, arma tramas con la información a transferir y además le añade datos de control. Las funciones de esta información se explicarán más adelante.

Usuario: Es quien genera la información que debe ser transferida o recibida.

Se establecen los siguientes principios sobre los cuales descansa el desarrollo del protocolo:

- Confiabilidad de la información transferida.
- Flexibilidad del diseño y software.

- Simplicidad de procesamiento.
- Modularidad basada en programación orientada a objetos (**POO**.)

3.1.2Arquitectura

Se desarrolló con una visión de orientación a objetos, por la flexibilidad que presenta para su implementación y mantenimiento. Se desarrollaron tres clases abstractas, para el protocolo, capa física y quien hace uso de los servicios del protocolo.

Esta forma de visualizar el problema, presenta las siguientes ventajas:

- Independencia entre capas, al heredar estas clases abstractas y modificarlas para una aplicación en específico.
- Se logra una comunicación bidireccional entre las capas, al mantener referencias a cada capa superior e inferior.

Estas clases se heredan. Se trabaja sobre ellas para alguna aplicación; por ejemplo, con el mismo protocolo heredado de la clase DataLinkLayer. Se puede trabajar sobre medios físicos distintos, heredados de la clase PhysicalLayer, como un puerto serial. La aplicación se puede variar, heredando de la clase AplicationLayer. Las relaciones de herencia se pueden visualizar en la Figura 17.

Las referencias a las clases superiores e inferiores permiten que cada capa tenga conocimiento de quién le está prestando un servicio y a su vez, a quién le presta un servicio. De esta forma se comunican.

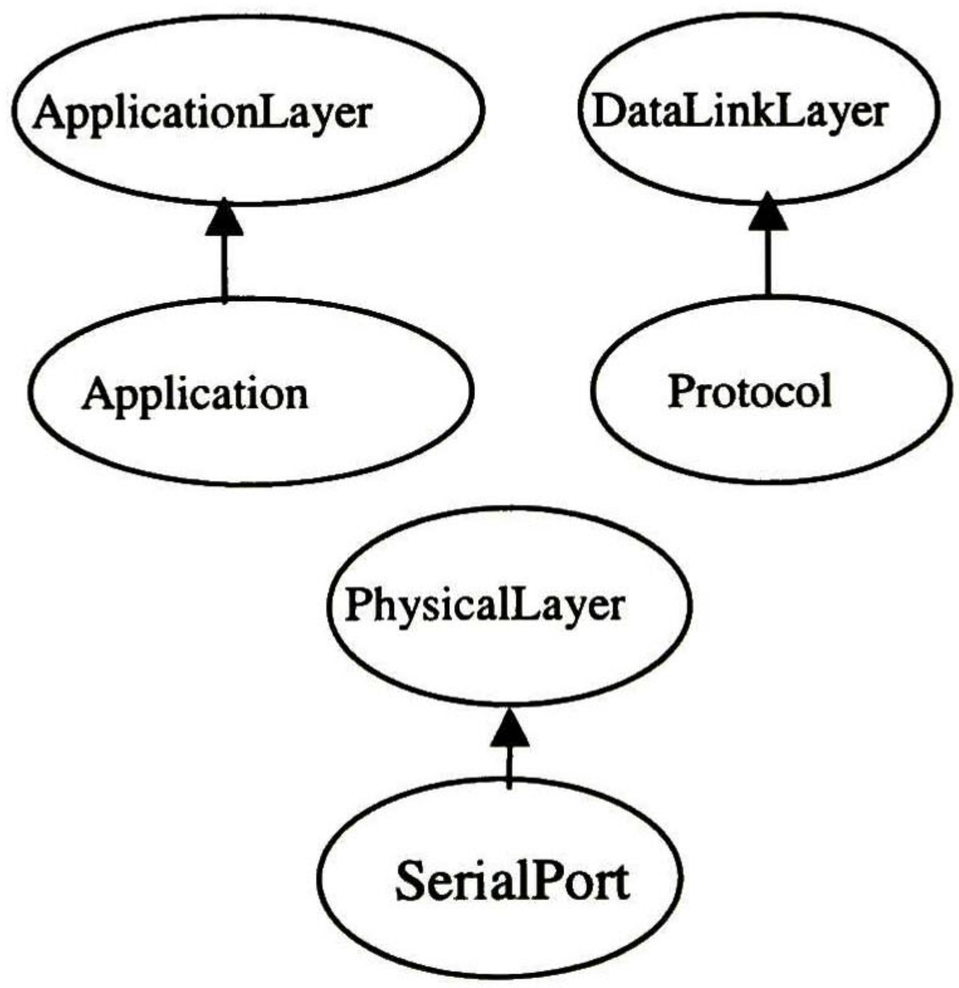


Figura 17 *Herencia de las clases abstractas*

Así, se tienen las siguientes clases abstractas:

```
class AplicationLayer{
        protected:
        DataLinkLayer *lower_layer;
};
```

```
class DataLinkLayer{
        protected:
        AplicationLayer *upper_layer;
        PhysicalLayer *lower_layer;
        public:
        virtual unsigned int SendFrame()=0;
        virtual unsigned int ReceiveFrame()=0;
};
class PhysicalLayer{
        protected:
        DataLinkLayer *upper_layer;
        public:
        virtual unsigned char GetStringRx(char *)=0;
        virtual unsigned char DataReceivedForProtocol(void)=0;
        virtual unsigned char WriteStringTx(char *, unsigned char)=0;
        virtual unsigned char CleanBuffers(void)=0;
        protected:
        virtual unsigned char CheckForData(void)=0;
};
```

La clase *Protocol*, que hereda de la clase *DataLinkLayer*, tiene la estructura mostrada a continuación.

```
class Protocol:DataLinkLayer{
class ApplicationLayer *upper_layer;
        class PhysicalLayer *lower_layer;
        // Apuntadores a las capas vecinas.
        public:
        unsigned char Protocol(int portnum, unsigned long speed,
                char parity, char stopbits, char wordlength);
        unsigned char Protocol();
        //Configura e inicializa la capa de protocolo. Hace uso de InitProtocol
        unsigned char ~Protocol(void);
         // Termina  la comunicación y finaliza el protocolo
        unsigned int SendFrame(char datacontrol,char *data);
        // Envía una trama de datos e informa si se logró exitosamente el envío
         unsigned int ReceiveFrame(unsigned char *command,char,
            unsigned char *receivestring);
// Recibe una trama de la capa inferior

private:
 unsigned long int CrcBit(short int bit,unsigned char reset);
 unsigned long int Crc(char *string,unsigned char action);
 // Funciones para el calculo de CRC
 unsigned int PutFrame(void);
 // Arma la trama y se la pasa a la capa física
 unsigned int SendWithAck(void);
```

```
//Lleva el control de envío de tramas, secuenciación y acuses de recibo
unsigned int GetAck(void);
// Realiza la recepción de los acuses de recibo
unsigned int ErrorInFrame(void); };
// Valida la trama recibida
```

3.1.3 Filosofía del protocolo

La filosofía es "para y espera" (stop and wait), por su simplicidad de procesamiento. Esta disciplina requiere del receptor acuse de recibo por cada trama de información transferida, y no envía una nueva trama hasta que le llegue acuse de recibo de la trama anterior. En caso de que este acuse de recibo no llegue, se retransmitirá la trama un número determinado de k veces, y si después de estas retransmisiones no se recibe el acuse, se concluye que el canal es inapropiado para la comunicación.

Por otro lado, cuando al receptor llega una trama, hace una verificación para detectar si hay errores. Si detecta alguno, solicita la retransmisión de la trama. Si nos los hay, envía el acuse de recibo al transmisor. La disciplina que el protocolo sigue para la corrección de los errores es la de la solicitud de retransmisión.

Ventajas

- Facilidad de implementación.
- Robusto.

Desventajas

- Deficiente para transmisiones cortas.
- Unidireccional.

3.1.4Formato de la trama

El formato define a la trama de longitud variable, desde 1 hasta 256 bytes, que consta de cinco campos:

- Longitud de la trama (un byte): contiene la longitud total de la trama.
- Consecutivo de trama (un byte): es el número de trama transferida.
- Comando (un byte): indica los comandos del protocolo (acuse de recibo, solicitud de retransmisión). Además se deja para uso de la capa superior.
- Datos (0 a 250 bytes): contiene la información a transmitir.
- Chequeo de errores (2 bytes): bits de redundancia para la verificación de errores (CRC.)

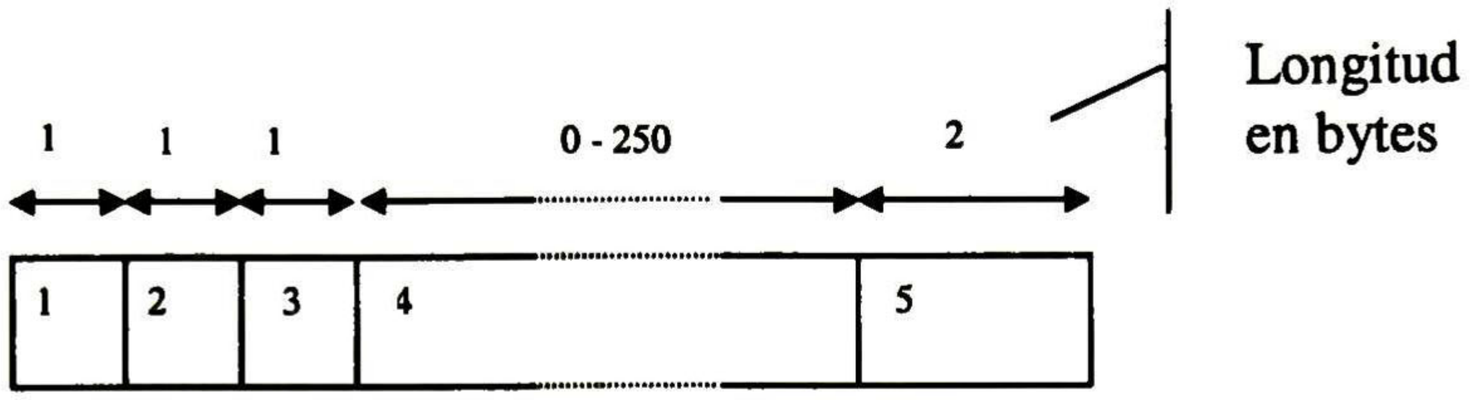


Figura 18 *Trama del protocolo*

El consecutivo de trama se usa para detectar tramas perdidas y pedir retransmisión. El campo de comandos es usado para su propio control, pero se deja también para uso de la capa superior. El chequeo de errores se lleva a cabo por un CRC de 16 bits. Para la elección del polinomio generador CRC, se analizó literatura correspondiente y se tenían como alternativas los polinomios $G(x)=x^{16} + x^{15} + x^2 + 1$ y $G(x)=x^{16} + x^{12} + x^5 + 1$, éste último, estándar del CCITT. La elección fue por el primero de ellos, ya que este código de redundancia cíclica permite detectar todos los errores simples y dobles, todos los errores de un número impar de bits, todos los errores en ráfagas de longitudes de 16 o menos, 99.997% de las ráfagas de errores de 17 bits y 99.998% de las ráfagas de 18 bits o mayores. Estas estadísticas fueron obtenidas [14] de donde es mencionado el estándar de la CCITT.

3.1.5Manejo de la capa física

Para la capa física, también se implementó una clase abstracta. Esta clase, además de incluir los métodos que le permiten comunicarse con la capa de protocolo, lleva el control de todos los dispositivos de la capa física que intervengan en una comunicación, creando una lista de objetos del tipo dispositivo por cada comunicación que se intente establecer. La detección del arribo de una nueva trama se lleva a cabo por medio de interrupción, para la cual se tiene una rutina de manejo.

Esta fue la filosofía que se utilizó, donde es necesario en ocasiones que una misma interrupción sea utilizada por varios dispositivos.

3.2 Resumen del capítulo

Es necesario reiterar que la necesidad de un mecanismo para asegurar la comunicación entre la PC y el microcontrolador es base del éxito de las pruebas, dado que el protocolo asegura que la comunicación es robusta y confiable.

En este capítulo se muestra el protocolo desarrollado. El objetivo es diseñar un protocolo flexible y confiable para una comunicación punto a punto transparente tanto a su capa superior como a su capa inferior, y que vea a esas capas de una forma igualmente transparente.

El protocolo de comunicaciones que se diseñó puede usarse en otras aplicaciones, y fue diseñado partiendo del modelo de referencia OSI, a un nivel de la capa de enlace de datos, además tiene características de Acuse de recibo en la recepción, Retransmisión de la trama, Tiempo de espera para establecer nuevamente la conexión, Retransmisión en caso de error en la recepción, detección de tramas perdidas, buffer de 255 bytes en medio físico y recepción controlada por un servicio de interrupción.

Las funciones de cada una de estas capas son:

- Medio Físico: Transmisión física de los bits de información.
- Protocolo: Conversión de la capa física en un medio confiable de transmisión.
- Usuario: Es quien genera la información que debe ser transferida.

Se establecen los siguientes principios sobre los cuales descansa el desarrollo del protocolo:

Confiabilidad, Flexibilidad, Simplicidad y Modularidad.

La filosofía del protocolo es "para y espera" (stop and wait) y requiere del receptor un acuse de recibo por cada trama de información transferida, y no envía una nueva trama hasta que le llegue acuse de recibo de la trama anterior. Este presenta sus ventajas y desventajas.

El formato define a la trama de longitud variable, desde 1 hasta 256 bytes, que consta de cinco campos:

- Longitud de la trama (un byte): contiene la longitud total de la trama.
- Consecutivo de trama (un byte): es el número de trama transferida.
- Comando (un byte): en él van los comandos del protocolo (acuse de recibo, solicitud de retransmisión). Además se deja para uso de la capa superior.
- Datos (0 a 250 bytes): contiene la información a transmitir.
- Chequeo de errores (2 bytes): bits de redundancia para la verificación de errores (CRC.)
- El chequeo de errores se lleva a cabo por un CRC de 16 bits.

Para la capa física se implementó una clase abstracta que incluye los métodos que le permiten comunicarse con la capa de protocolo, y además lleva el control de todos los dispositivos de la capa física que intervengan en una comunicación.

4 Plan de pruebas para la MC

En cualquier sistema o elemento que se diseña, la etapa de verificación usa una gran parte del tiempo de desarrollo, por esto un buen plan de pruebas ayuda a que este tiempo sea usado correctamente, algunos autores asignan el 70% del tiempo de verificación, el número de ingenieros de verificación se puede llegar a duplicar con respecto a la gente de diseño; también se calcula que el 80% del código desarrollado es de verificación. Esto hace que los planes de prueba tomen importancia considerablemente. Para mas detalles vea [15].

Ahora se muestra como se desarrolla un plan de pruebas, usando a BIST como idea principal.

4.1 Metodología de pruebas

Se plantea BIST para el desarrollo de las pruebas de la MC, como a continuación se explica.

4.1.1 Pruebas BIST

Las pruebas BIST ("Built-In Self-Test:" Autopruebas dentro del mismo diseño) se refieren a realizar elementos de diagnóstico para sistemas desarrollados, para las pruebas a sistemas de comunicaciones en nuestro caso la MC, se tiene un diagrama general que a continuación se presenta:

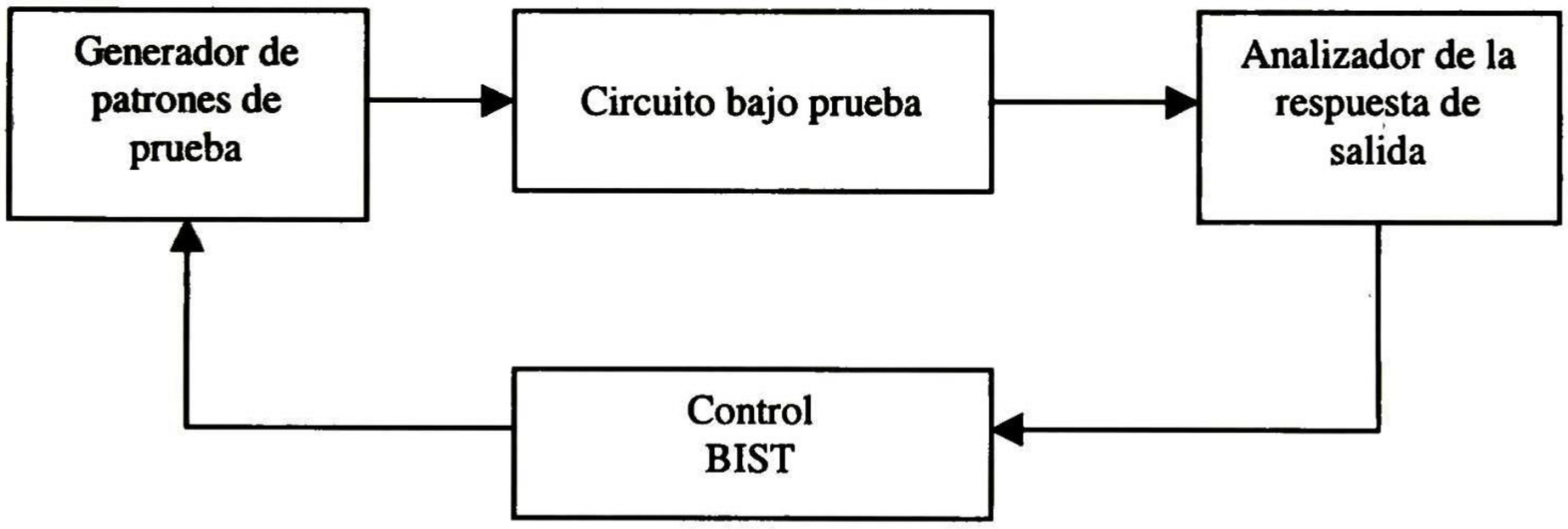


Figura 19 *Diagrama general del sistema BIST*

Ventajas del uso de BIST:

- Bajo tiempo de verificación del módulo.
- Elimina la necesidad de sistemas de diagnóstico costosos.
- Posibilidad de realizar pruebas de alta velocidad.
- Posibilidad de pruebas de campo.
- Alta cobertura de fallas.

La metodología de generación de pruebas puede ser determinística, exhaustiva o aleatoria. Las pruebas determinísticas generan pruebas con modelos de falla, basándose en procesamiento algorítmico de la estructura del circuito.

La generación de pruebas exhaustivas requiere la aplicación de todas las posibles combinaciones a las entradas del circuito y la verificación de las respuestas correspondientes.

La metodología de pruebas más ampliamente usada en el ambiente BIST está basada en pruebas pseudo aleatorias. En esta técnica, son generados y aplicados vectores de prueba aleatorios que forman parte del subconjunto de pruebas exhaustivas a las entradas del circuito. Son usados registros de corrimiento de retroalimentación lineal para generar vectores de prueba pseudo aleatorios.

BIST es básicamente una manera de saber que el circuito bajo prueba realiza lo planeado, usando un generador de patrones inyectados al circuito probando los elementos internos, como por ejemplo cadenas SCAN, y obteniendo a la salida los resultados que se obtiene de dicha prueba todo logrado desde el control de BIST.

BIST como se muestra en la Figura 19, tiene tres módulos como se muestran a continuación:

- Generador de patrones de prueba.
- Analizador de respuestas de salida.
- Control BIST.

4.1.1.1 Generador de patrones de prueba

La mayoría de las veces es un generador de patrones pseudo aleatorios, el cual trata de probar que bajo la mayoría de los patrones que se inyectan al circuito. Este debe de reaccionar de acuerdo a lo previsto. La filosofía de funcionamiento nos dice que BIST se aplica en manufactura. Desde este punto de vista, la funcionalidad del circuito bajo prueba se descarta por completo, quedando solamente las pruebas de los elementos internos y su funcionalidad base, como por ejemplo un conjunto de compuertas lógicas, FLIP-FLOP´s, etc.

4.1.1.2 Analizador de respuestas de salida

Se encarga de comprobar que los datos arrojados por el circuito sean correctos. Este módulo debe tener la característica de ser observable para poder ser un elemento funcional.

4.1.1.3 Control BIST

Este esquema de diagnóstico tiene un equivalente dentro de nuestro plan de pruebas, ya que como BIST exige que el control esté dentro del propio diseño y la tendencia actual es que así continúe, hablando de la MC, esta tiene en la propia tarjeta un sistema muy semejante al propuesto por BIST, donde se tiene un microcontrolador que realiza las tres funciones mencionadas (Generador de patrones de prueba, Analizador de respuestas de salida, Control BIST).

Así, el microcontrolador proporciona los patrones de prueba al introducir por los EE tramas de diagnóstico, y con señales de control para el BCG, transfiriendo los datos de los EE a los

ES. También el microcontrolador es el que captura los datos, analizándolos para informar que la MC funciona correctamente.

4.1.2 Unión entre el software y el hardware, por medio de BIST

BIST se enriquece al unirse con software para diagnóstico de elementos que son susceptibles a fallas. La inclusión de BIST en estos sistemas reduce la complejidad del software de pruebas, como lo propone [9] En nuestro caso, el microcontrolador realiza pruebas con una periodicidad programable para que la información sea procesada y se detecten las fallas que sucedieron, en lugar de la forma tradicional en la que el sistema de control activa alarmas para que el software dedicado los detecte en el momento de la falla.

Todo esto tiene un enfoque técnico, el cual nos dice que el saber dónde se origina una falla y qué falla ocurre, nos ayuda a que la reparación sea rápida y tenga un impacto menor o no considerable en el costo de operación del equipo.

Se aconseja un diseño modular para poder intercambiar bloques completos y así reducir tiempos de reparación y más si éste se encuentra en lugares lejanos o que presenten algún problema para poder acceder a ellos.

4.1.3 Implementación de BIST

Existen maneras en las cuales se pueden probar memorias. A continuación se explican dos esquemas de pruebas para memorias, que son:

- En línea
- Fuera de línea.

4.1.3.1 Pruebas fuera de línea

La arquitectura propuesta nos indica cómo agregar los elementos ya conocidos para BIST, donde el generador de patrones escribe a la memoria datos, los cuales son leídos y comparados en el analizador de respuestas, y el control del BIST ejecuta el algoritmo de prueba para la memoria.

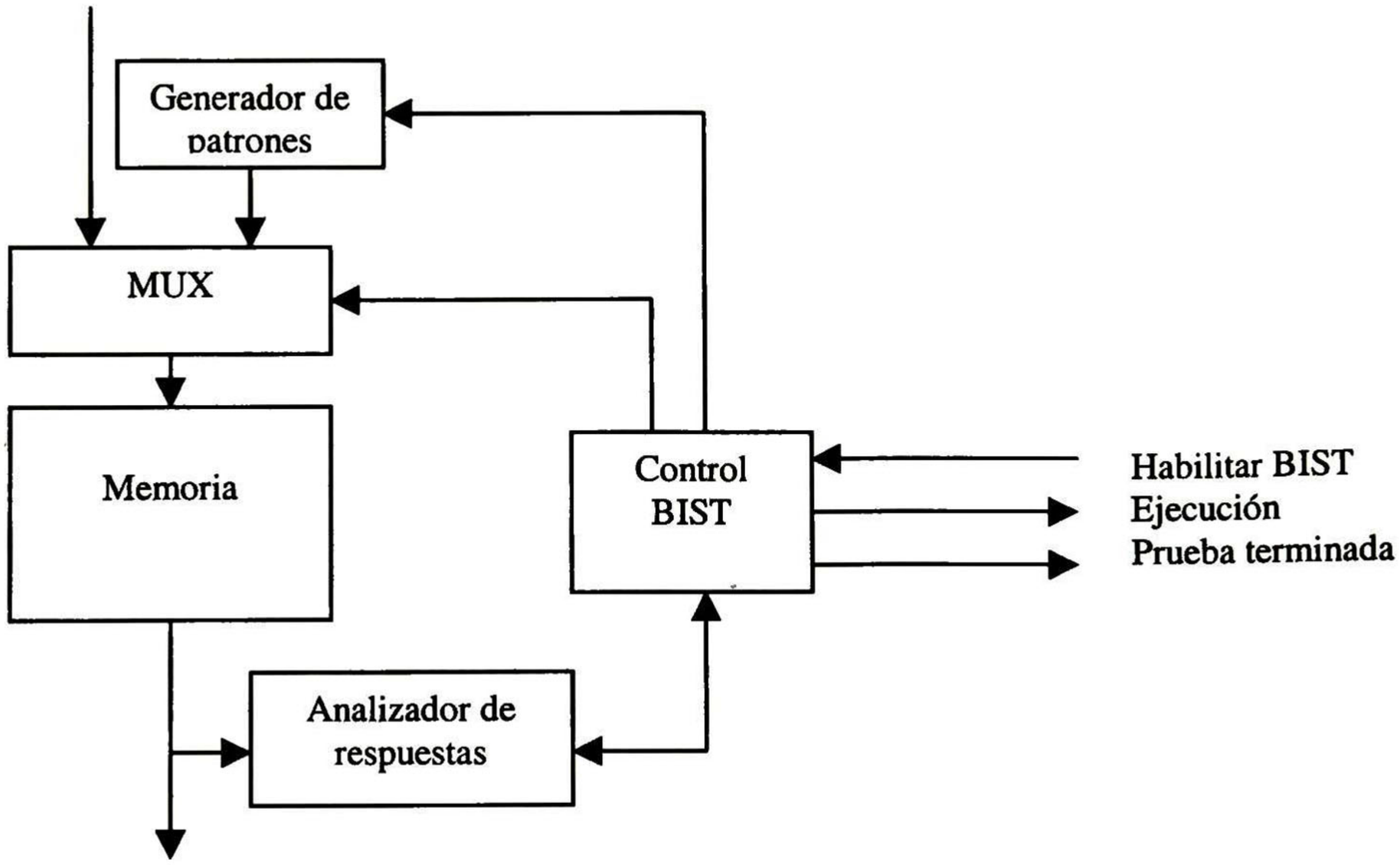


Figura 20 *Estructura general de la arquitectura BIST fuera de línea*

4.1.3.2 Pruebas en línea

Cuando las fallas son críticas y los errores deben ser detectados, el esquema de pruebas fuera de línea no tiene sentido. En este tipo de esquema es donde las pruebas en línea toman valor, en este documento se define como falla crítica aquella que provoca que el sistema deje de funcionar adecuadamente, hasta que un medio externo restaure la falla, se define como error a todos los eventos que provocan una falla crítica.

Se puede lograr protección contra fallas con el principio de redundancia de información, esto es, se puede escribir en dos o más módulos de memoria para que de esta manera la información se almacene en forma simultánea en diferentes memorias, para después, leer las diferentes fuentes y por votación decidir cual es la información real. De esta manera es más fácil detectar un error sin degradación significativa en el servicio del sistema.

Este tipo de redundancia se forma al introducir un código de detección y/o corrección de datos. Los datos son escritos en la memoria después de ser codificados, y en el momento de la lectura el dato original es decodificado nuevamente. Así, se extrae el dato original, y los códigos son comparados. Finalmente se decide si el dato es válido o no. Vea la Figura 21.

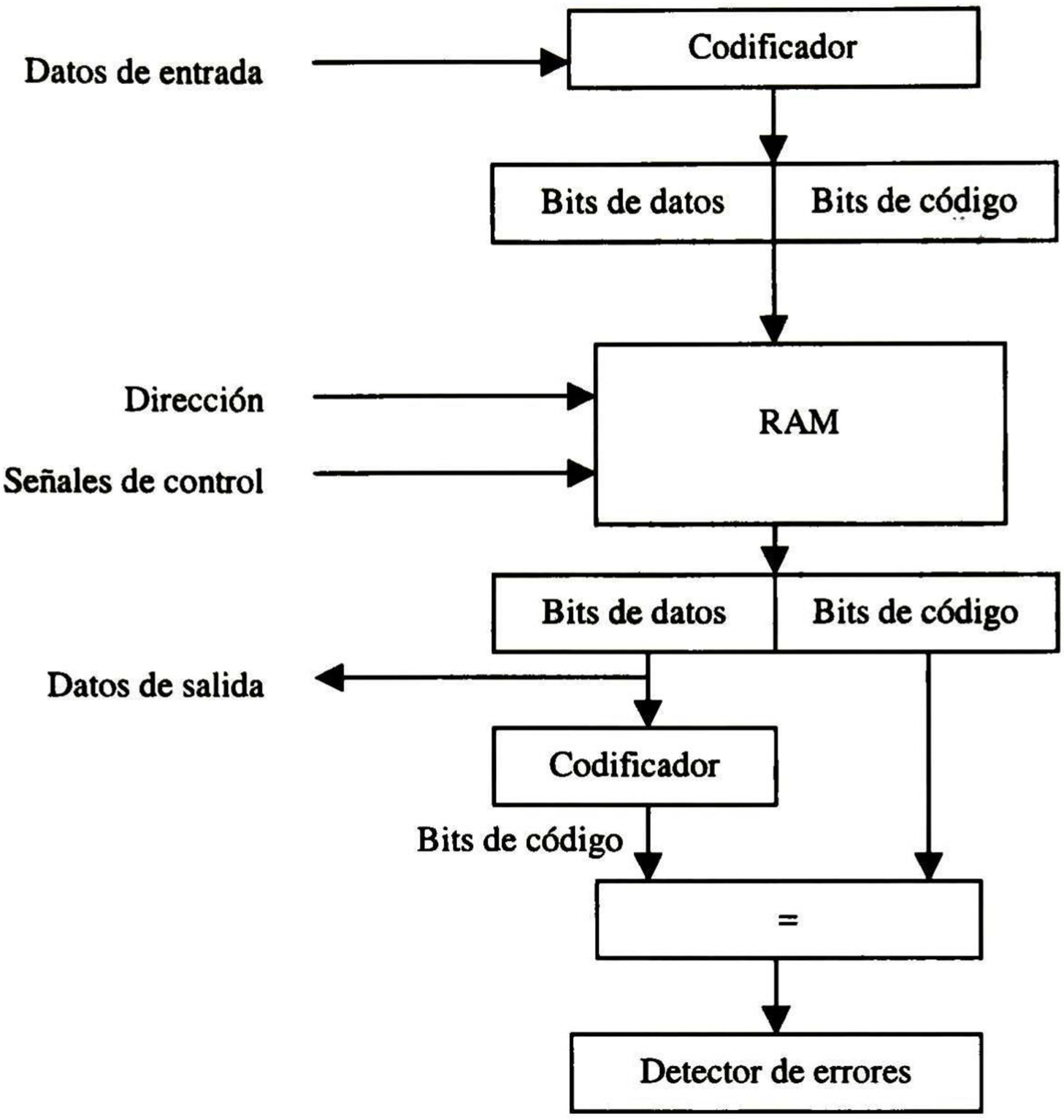


Figura 21 *Arquitectura de códigos separables para un BIST en línea*

Los elementos de memoria se prueban en una secuencia de pasos para abarcar a todos los puntos expuestos en el modelo de fallas.

Para probar elementos de memoria donde los datos no pueden ser modificados, como ocurre en nuestra arquitectura, no puede almacenarse información en otro lugar mientras estos dispositivos son examinados, por lo que las pruebas deberán de ser realizadas fuera de línea. Se aplicarán para ROM, RAM interna y externa, que no pueden ser modificadas dado que guardan datos importantes del programa mismo. Por esto ahora se expone un plan de pruebas que permita detectar errores sin afectar el contenido de las memorias.

4.2 Dispositivos a probar

Los elementos que se han probado se listan a continuación:

- Elementos de entrada
- Elementos de salida
- Tabla de direcciones
- Registros COSP y ACAP
- Circuito de línea E1/T1 junto con la MC.

4.2.1 Elementos internos

A continuación se presentan las pruebas que han sido desarrolladas.

4.2.1.1 Funciones a probar

Definiendo las funciones a probar de los dispositivos bajo prueba. Estas se muestran a continuación.

4.2.1.1.1 Elemento de entrada

Es un elemento de baja complejidad como ya se ha visto en [1]. Consiste básicamente en una FIFO y la máquina de estados, que realiza las operaciones de transferir al bus común los datos colocados por el microcontrolador. La rutina de diagnóstico se divide en:

- Verifica FIFO de entrada.
- Verifica banderas de la FIFO de entrada.
- Verifica FSM de escritura de los EE al bus común.

4.2.1.1.1.1 Verifica FIFO de entrada

La FIFO de entrada tiene importancia en la recepción de datos del exterior. Es por ello el desarrollo de una prueba para este fin.

4.2.1.1.1.1.1 Diseño de la prueba

La verificación de esta memoria se basa en escrituras y lecturas a ella, para comprobar que los datos que se leen son los que se escribieran. La escritura es realizada por la FSM de escritura del EE, donde el microcontrolador proporciona la señal de reloj para la FSM. La parte de lectura es similar, realizada por la FSM de lectura del EE, donde el reloj también es proporcionado por el microcontrolador. De esta manera, el microcontrolador en conjunto con el EE, realiza la prueba y verificación de este dispositivo.

4.2.1.1.1.1.2 Casos de prueba

Los casos de prueba se listan como sigue:

- Escritura de datos a la memoria.
- Lectura de datos de la memoria.

4.2.1.1.1.1.3 Entradas

Flujo de datos del microcontrolador a la FIFO bajo prueba.

4.2.1.1.1.1.4 Salidas

Flujo de datos de lectura de la FIFO al microcontrolador.

4.2.1.1.1.1.5 Criterio de aceptación de la prueba

Se acepta la prueba a la memoria si y sólo sí los datos escritos son los mismos que los leídos.

4.2.1.1.1.2 Verifica banderas de la FIFO de entrada

Las banderas de la FIFO son elementos de monitoreo de la FIFO, con los que se permite tener un reporte del estado de la FIFO bajo prueba.

4.2.1.1.1.2.1 Diseño de la prueba

Como se ha explicado en [1], la FIFO de entrada tiene cuatro banderas programables que nos dicen de su ocupación. Las banderas son FIFO vacía, FIFO Llena, FIFO casi vacía, FIFO casi llena. Las banderas se activan dependiendo de la ocupación. Escribiendo y leyendo datos estas banderas se actualizan y se puede probar que funcionan normalmente. Se puede configurar la FIFO para cambiar los límites para los cuales las banderas deben activarse o desactivarse.

4.2.1.1.1.2.2 Casos de prueba

Durante la fase de escritura de datos, es monitoreado el estado de las banderas, para así comprobar que estas trabajan correctamente.

4.2.1.1.1.2.2.1 Entradas

Flujo de datos del microcontrolador a la FIFO bajo prueba.

4.2.1.1.1.2.2.2 Salidas

Flujo de datos de lectura de la FIFO al microcontrolador.

4.2.1.1.1.2.3 Criterio de aceptación de la prueba

La prueba se acepta si y sólo sí las banderas se activan correctamente.

4.2.1.1.1.3 Verifica FSM de escritura de los EE al bus común

La FSM de escritura es un módulo el cual reacciona a un conjunto de señales provenientes tanto del BCG y la FIFO, que es importante para la correcta transferencia de los datos de entrada al registro de entrada del EE, visto en el *expansor de 8 a 64* de la Figura 2.

4.2.1.1.1.3.1 Diseño de la prueba

La FSM del EE es quien transfiere los datos al BUS, dentro de esta verificación se divide en tres pasos: colocar una celda completa en la FIFO de entrada, proporcionar el reloj a la FSM para que transfiera la celda escrita, y el microcontrolador realiza la labor del BCG leyendo la celda del ES, para ser verificado.

4.2.1.1.1.3.2 Casos de prueba

Escritura de datos del microcontrolador.

4.2.1.1.1.3.2.1 Entradas

Datos provenientes del microcontrolador para ser almacenados en la FIFO.

4.2.1.1.1.3.2.2 Salidas

Estado de las banderas de la FIFO.

4.2.1.1.1.3.3 Criterio de aceptación de la prueba

Los resultados de las banderas y contenido leído de memoria deben coincidir con el esperado.

4.2.1.1.2 Elemento de salida

En comparación con el EE es de mayor complejidad, contiene más bloques y una máquina de estados más compleja, su función es que recibe los datos que llegan al bus común, para después colocarlos en la cosa de salida, para después, transferirlos a los CL conectados a la MC, para mayor detalle ver [1].

La verificación de este módulo se divide en:

- Verifica cola de salida.
- Verifica tabla de direcciones TDE.
- Verifica registros COSP y ACAP.

Definiciones:

COSP: Registro de 6 bits para seleccionar la dirección base de escritura a la cola de salida, para cada ranura de ciclo de bus. Estas direcciones base se encuentran localizadas en la TDE (tabla de direcciones de escritura).

ACAP: Registro de 6 bits que sirve para pasar al registro COSP la dirección TDE, que apuntará a la dirección de escritura de la cola de salida durante la primer ranura del siguiente ciclo de bus.

4.2.1.1.2.1 Cola de salida

Este elemento se encarga de la transferencia de datos del bus común al ES en conjunción con las máquinas de estados de lectura y escritura.

4.2.1.1.2.1.1 Diseño de la prueba

Para la verificación de este elemento se requiere realizar lecturas y escrituras a la cola de salida. La verificación se divide en lectura y escritura de la memoria, en estos dos procesos se involucra el control de los registros COSP y ACAP, registros de control para la FSM de escritura y lectura, existen dos FSM para el proceso completo de transferencia de datos.

4.2.1.1.2.1.2 Casos de prueba

Transferencia de datos desde el EE pasando por el bus común hasta la cola de salida del ES.

4.2.1.1.2.1.2.1 Entradas

Flujo de una trama completa desde el elemento de entrada pasando por el bus.

4.2.1.1.2.1.2.2 Salidas

Estado de la ocupación de la cola de salida.

4.2.1.1.2.1.3 Criterio de aceptación de la prueba

La prueba se acepta si y sólo sí el estado de ocupación de la cola de salida corresponde con las tramas escritas desde el EE de la MC.

4.2.1.1.3 Tabla de direcciones

La Tabla de Direcciones de Escritura (TDE) contiene las direcciones de escritura para la cola de salida de 36 localidades de 64 bits de longitud, tales direcciones son los apuntadores de inicio para cada localidad. Una localidad se refiere a una palabra de 64 bits de ancho, por ejemplo una localidad puede almacenar el valor 0xFFFFFFFFFFFFFFFF.

4.2.1.1.3.1.1 Diseño de la prueba

Su verificación consiste en leer la tabla y comprobar que los valores que se encuentran en ella sean los correctos.

4.2.1.1.3.1.2 Casos de prueba

Verificar el contenido de la TDE.

4.2.1.1.3.1.2.1 Entradas

Dirección de memoria para la lectura de la TDE.

4.2.1.1.3.1.2.2 Salidas

Contenido de la TDE para análisis.

4.2.1.1.3.1.3 Criterio de aceptación de la prueba

La prueba se acepta si y sólo sí el contenido de la TDE corresponde con lo esperado, este arreglo es estático y es conocido de antemano por el firmware de prueba.

4.2.1.1.4 Registros COSP y ACAP

Estos registros son complementarios con las FSM de escritura y lectura de la cola de salida o del elemento de salida.

La siguiente figura muestra un poco mas como se relacionan estos dos registros.

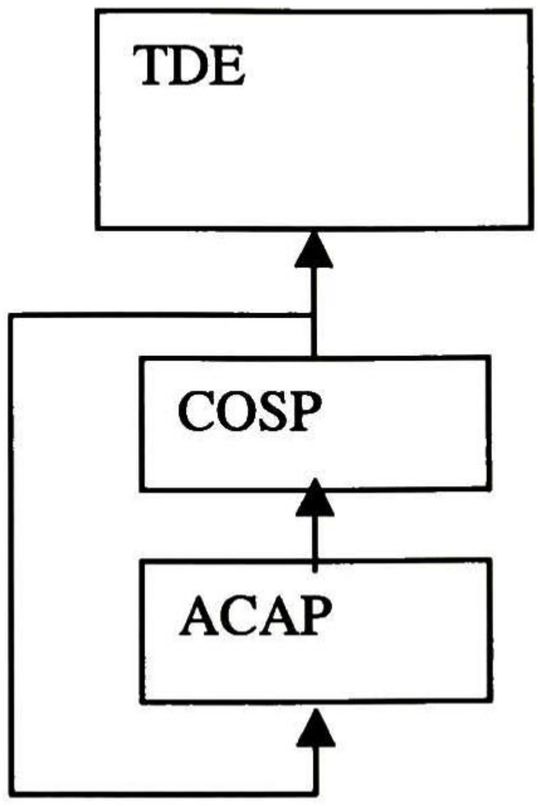


Figura 22 *Interconexión entre los registros COSP y ACAP*

4.2.1.1.4.1.1 Diseño de la prueba

Los registros COSP y ACAP ayudan al controlador BCG para tener un control de los punteros a la cola de salida, para escribir y leer, se han mapeado en el microcontrolador para tener acceso a los diferentes registros de la MC, así se simula el funcionamiento de la FSM de escritura y lectura para verificación y pruebas del ES.

4.2.1.1.4.1.2 Casos de prueba

Verificar el correcto funcionamiento de los registros, estos registros se modifican junto con la funcionalidad del TDE, es necesario primero inicializar la TDE para esta prueba de registros COSP y ACAP.

4.2.1.1.4.1.2.1 Entradas

Inicialización de la TDE.

4.2.1.1.4.1.2.2 Salidas

Valores de los registros.

4.2.1.1.4.1.3 Criterio de aceptación de la prueba

El valor de los registros deben de coincidir a los estímulos seleccionados.

4.2.2Circuito de línea

El circuito de línea es un dispositivo el cual se conecta a un puerto de entrada y salida de la MC transformando el protocolo TOPIA propia de esta, en otro protocolo; por ejemplo, Ethernet, ATM, UTOPIA, E1/T1, etc.

En esta tesis se probaron los siguientes circuitos de línea:

- Convertidor CUT.

- Convertidor CUU.

4.2.2.1 Convertidor CUT

4.2.2.1.1 Funciones a probar

Este convertidor puede transformar en ambos sentidos la información, de TOPIA a UTOPIA y viceversa, las pruebas son pasar tramas en los dos sentidos.

4.2.2.1.2 Descripción técnica de la prueba

Se necesita crear tramas de TOPIA y UTOPIA, para poder realizar la transferencia de información y así probar la bidirecionalidad del módulo.

4.2.2.1.3 Diseño de la prueba

Crear una trama de TOPIA, con valores pseudo aleatorios.

Crear una trama de UTOPIA, con valores pseudo aleatorios.

4.2.2.1.4 Caso de prueba

Se insertan las dos tramas creadas por ambos lados de la interfaz, primero uno después el segundo.

4.2.2.1.4.1 Entradas

Tramas de TOPIA, con valores pseudo aleatorios.

Tramas de UTOPIA, con valores pseudo aleatorios.

4.2.2.1.4.2 Salidas

Tramas transferidas y convertidas a su equivalente formato de salida.

4.2.2.1.5 Criterio de aceptación de la prueba

Se debe de obtener la señalización correspondiente y los datos esperados de las tramas transferidas.

4.2.2.2 Convertidor CUU

4.2.2.2.1 Funciones a probar

Este convertidor puede transformar en ambos sentidos la información, de UTOPIA nivel 1 a UTOPIA nivel 2, y viceversa, las pruebas son pasar tramas en los dos sentidos.

Señal	Ancho De bus	Tipo	Descripción
Loopback	1	E	Señal para habilitar y deshabilitar el modo de loopback
Reset	1	E	Reset asíncrono activo en bajo
CLk	1	E	Reloj general del bloque
Clockx2	1	E	Reloj secundario, se usa para las memorias doble puerto
U2_RxAddress	5	E	Dirección de la capa, U2, recepción
U2_RxData	8	S	Datos de la capa U2, recepción
U2_RxSoc	1	S	Señal de comienzo de trama para U2, recepción
U2_RxClav	1	S	Señal de habilitación celda lista para recepción, producida por la capa física, U2.
U2_RxEnb	1	E	Señal de habilitación que indica que la capa ATM puede recibir, U2 recepción
U1_RxData	8	E	Datos de entrada a la capa ATM
U1_RxSoc	1	E	Señal de comienzo de trama, U1, recepción
U1_RxEnb	1	S	Señal de habilitación de la capa ATM, U1, recepción
U1_RxClav	1	E	Señal de habilitación de la capa física, U1, recepción.
U2_TxAddress	5	E	Dirección de la capa física, U2, transmisión
U2_TxData	8	E	Datos de transmisión, U2, transmisión
U2_TxSoc	1	E	Inicio de trama, U2, transmisión
U2_TxClav	1	S	Señal de habilitación de la capa física, U2, transmisión
U2_TxEnb	1	E	Señal de habilitación de la capa ATM, U2, transmisión
U1_TxData	8	S	Datos de transmisión, U1, transmisión
U1_TxSoc	1	S	Inicio de trama, U1, transmisión
U1_TxEnb	1	S	Habilitación de la capa ATM, U1, transmisión

U1_TxClav	1	E	Habilitación de la capa física, U1, transmisión

Tabla 5 *Señales del CUU*

4.2.2.2.2 Descripción técnica de la prueba

Se necesita crear tramas de UTOPIA nivel 1 y nivel 2, para poder realizar la transferencia de información y así probar la bidirecionalidad del módulo.

4.2.2.2.3 Diseño de la prueba

Crear una trama de UTOPIA nivel 1, con valores pseudo aleatorios.

Crear una trama de UTOPIA nivel 2, con valores pseudo aleatorios.

4.2.2.2.4 Casos de prueba

Insertar las dos tramas creadas por ambos lados del interfaz, primero uno después el segundo.

4.2.2.2.4.1 Entradas

Tramas de UTOPIA nivel 1, con valores pseudo aleatorios.

Tramas de UTOPIA nivel 2, con valores pseudo aleatorios.

4.2.2.2.4.2 Salidas

Tramas transferidas y convertidas a su equivalente formato de salida.

4.2.2.2.5 Criterio de aceptación de la prueba

Se debe de obtener la señalización correspondiente y los datos esperados de las tramas transferidas.

4.2.3 Pruebas al microcontrolador

Como inicio para las pruebas a la MC, se tiene que garantizar el funcionamiento del microcontrolador.

La rutina de diagnóstico se divide en:

- Registros del microcontrolador.
- Memoria E^2PROM.
- Memoria RAM interna y externa.
- Puerto serie.

4.2.3.1 Registros del microcontrolador

Los registros del microcontrolador son en especial útiles para el dispositivo, son elementos indispensables para el microcontrolador. Son necesarias las pruebas de su buen funcionamiento.

4.2.3.1.1 Diseño de la prueba

Consiste en escribir datos para después ser leídos. Los datos deben asegurar que cualquier valor pueda ser guardado, para evitar que algún registro pueda estar sujeto a un valor determinado.

4.2.3.1.2 Casos de prueba

Se consideran los casos de escritura y lectura de valores en los registros del microcontrolador.

4.2.3.1.2.1 Entradas

Valores predefinidos para asegurar la integridad del registro.

4.2.3.1.2.2 Salidas

Valores leídos de cada registro.

4.2.3.1.3 Criterio de aceptación de la prueba

La prueba se acepta si y sólo sí, los valores leídos coinciden con los valores anteriormente escritos.

4.2.3.2 Memoria E^2PROM

Se requiere que la memoria del programa no tenga datos inválidos, por esto se requiere que el código se pueda probar para asegurar que el microcontrolador realice correctamente las funciones programadas.

4.2.3.2.1 Diseño de la prueba

Se muestra el algoritmo usado para su verificación:

1. Obtener CheckSum desde la PC del archivo HEX original.
2. Obtener el complemento a 2 del CheckSum.
3. Mandar el dato al microcontrolador.
4. Leer los 10Kb de la memoria E^2PROM y sumarlos al CheckSum obtenido de la PC.
5. Comprobar que el resultado final es 0. Si no alertar de error en la ROM.

4.2.3.2.2 Casos de prueba

Se considera que el caso de prueba consiste en que los datos grabados en la memoria del microcontrolador no sufran cambios. Esto es equivalente a probar que el código alojado en la memoria E^2PROM no ha sufrido cambios.

4.2.3.2.2.1 Entradas

Checksum desde la PC.

Checksum calculado por la función de lectura del microcontrolador.

4.2.3.2.2.2 Salidas

Estado de alerta o de código validado.

4.2.3.2.3 Criterio de aceptación de la prueba

La prueba se acepta si y sólo sí los Cheksum obtenidos coinciden correctamente.

4.2.3.3 Memoria RAM interna y externa

Este tipo de memorias en un microcontrolador se utiliza como elemento de almacenamiento, en las cuales se mantienen los datos del stack del microprocesador, junto con las variables globales y temporales del compilador usado.

4.2.3.3.1 Diseño de la prueba

Las rutinas de diagnostico se han desarrollado para memorias. Las pruebas que se presentan fueron tomadas de [4] Este artículo presenta una metodología para pruebas de memorias, asegurando su correcto funcionamiento.

Modelos de fallas comúnmente usados para representar defectos físicos en las memorias:

- Simple o múltiple amarrado a 1 ó 0: un valor específico de la memoria no cambia su valor binario.
- El valor de uno o más bits de una celda de memoria es cambiado.
- Acoplamiento: en una operación de escritura al escribir a una celda una o más celdas son accesadas simultáneamente.
- Falla de transición: Una celda o bit de una celda falla en la escritura o lectura de un cambio de valor de 1 a 0 o de 0 a 1.
- Para una celda direccionada el valor no es escrito o leído.
- Para una celda direccionada el valor es escrito en múltiples direcciones.
- Una celda nunca es direccionada.
- Una celda es direccionada con múltiples direcciones.

4.2.3.3.2 Casos de prueba

Los casos de prueba consisten en al escritura de datos vistos en la sección 4.2.3.3.1.

4.2.3.3.2.1 Entradas

Conjunto de datos de prueba.

4.2.3.3.2.2 Salidas

Conjunto de datos para análisis.

4.2.3.3.3 Criterio de aceptación de la prueba

La prueba se acepta si y sólo sí el algoritmo explicado en 4.2.3.3.1 decide que los datos pueden ser almacenados correctamente.

4.2.3.4 Puerto serie

El puerto serie es la interfaz física entre el firmware y el Software, es por ello importante asegurar que los datos se transfieren correctamente en ambos sentidos.

4.2.3.4.1 Diseño de la prueba

La prueba se basa en la detección del enlace entre el microcontrolador y la PC.

4.2.3.4.2 Casos de prueba

Como caso de prueba se considera la petición de acuse de recibo del microcontrolador a petición de la PC.

4.2.3.4.2.1 Entradas

Transferencia de una trama de prueba por parte de la PC al microcontrolador.

4.2.3.4.2.2 Salidas

Trama de acuse de recibo por parte del microcontrolador.

4.2.3.4.3 Criterio de aceptación de la prueba

El protocolo de comunicación usado en este enlace define un total de 3 reintentos para que el microcontrolador responda al pedido de la PC, a la vez de un tiempo de espera de 3 segundos entre reintento. La prueba se acepta si en esta cantidad de intentos el microcontrolador responde a la PC con un acuse de recibo.

4.3 Resumen del capítulo

En este capítulo se muestra cómo se desarrolla un plan de pruebas, usando BIST como idea principal.

Se plantea BIST para el desarrollo de las pruebas de la MC, y presenta ventajas como el bajo tiempo de desarrollo, corto tiempo de aplicación de pruebas, elude la necesidad de sistemas de diagnóstico costosos, la realización de pruebas de alta velocidad, realización de pruebas de campo y la alta cobertura de fallas.

La metodología de generación de pruebas puede ser determinística, exhaustiva o aleatoria. La más ampliamente usada en el ambiente BIST está basada en pruebas pseudo aleatorias, que implica la generación y aplicación de vectores de prueba generados de un subconjunto de pruebas exhaustivas.

BIST tiene tres módulos: Generador de patrones de prueba, Analizador de respuestas de salida y Control BIST.

La inclusión de BIST en los sistemas reduce la complejidad del software que se desarrolla, además de que saber donde se origina una falla y qué falla ocurre, ayuda a que la reparación sea rápida y tenga un impacto menor o no considerable en el costo de operación del equipo.

Algunas maneras de probar memorias son:

- En línea.
- Fuera de línea.

Los elementos que se prueban son:

- Elementos internos y funcionalidad de la matriz de conmutación.
- Circuito de línea E1/T1 junto con la MC.

Se presentan las pruebas que han sido desarrolladas para tanto los elementos internos y funcionalidad de la matriz de conmutación, como del circuito de línea E1/ T1 en conjunto con la MC.

De la matriz de conmutación se verifica: FIFO de entrada, banderas de la FIFO de entrada y FSM de escritura de los EE al bus común.

Los casos de prueba son: Escritura y lectura de datos de la memoria, y Flujo de datos del microcontrolador a la FIFO y de la FIFO al microcontrolador.

El circuito de línea es un dispositivo el cual se conecta a un puerto de entrada y salida de la MC transformando el protocolo TOPIA propia de esta, en otro protocolo; por ejemplo, Ethernet, ATM, UTOPIA, E1/T1, etc.

Se probaron los siguientes circuitos de línea:

- Convertidor CUT.
- Convertidor CUU.

Como complemento se mencionaran algunos aspectos importantes para el proceso de pruebas, los cuales pueden ayudar a diseñar las pruebas, los cuales son:

- Se debe tener un modelo para representar al elemento que se quiere probar.
- Identificar la funcionalidad a probar del modulo.
- Diseñar un conjunto de elementos que servirán como estímulos para las pruebas, de tal manera que cubra a todo o un porcentaje deseado del elemento bajo prueba.
- La verificación tiene que tener en cuenta si las pruebas se realizaran para cajas negras o cajas blancas.
- Asegurar que se obtienen los resultados esperados.

- Escoger la herramienta adecuada que servirá como cama de pruebas.

5 Firmware con soporte del plan de pruebas

El protocolo antes descrito ha sido implementado como medio de comunicación entre un microcontrolador AT89S8252 (compatible con el 8051 de INTEL) y una PC. El microcontrolador es usado en la MC para realizar las pruebas y diagnósticos del mismo. Esta aplicación es capaz de transferir sus resultados a la PC y también de recibir ordenes desde la PC para dichas rutinas, lo que nos da la flexibilidad de control desde software de la MC y sus periféricos CL, como se explicará más adelante.

5.1 Monitoreo de la matriz de conmutación

Las tareas que la MC realiza son operaciones que se requieren monitorear, en este caso, el microcontrolador puede realizar la verificación de los EE y ES, por lo cual tiene un módulo autónomo para esta tarea. Aun así, la PC puede no ordenar tarea alguna, siendo capaz de obtener datos provenientes del microcontrolador.

Los datos obtenidos son estadísticas de EE y ES. También se obtiene el desempeño de la MC como CL activos, flujo de información y celdas perdidas. La capacidad de observación es importante para que la MC se pueda probar, y poder realizar depuración y pruebas.

5.2 Control del circuito de línea

El control de los CL desde software es importante para que la MC se pueda monitorear y depurar desde un lugar remoto sin que el usuario tenga que observar el problema, ya que al obtener los resultados por medio de esta interfaz, se puede saber con anticipación los problemas que se presentan.

Así se ha logrado implementar un control de los CL, los cuales tienen que ser configurados por algún dispositivo externo. Para ahorrar el uso de otra computadora u otra interfaz que proporcione este servicio, se usa el microcontrolador para lograr el acceso a los CL por medio de los EE y ES. La tarea que se implementan es el llenado de la Tabla de Enrutado TR (elemento en el CL que nos permite el enrutado de la información para ser entregado a la MC, como se explica en [3]).

El acceso a los CL se realiza de la siguiente manera: la TR se localiza en la PC, se requiere hacer llegar al CL indicado estos datos. En general, se coloca la TR en todos los CL conectados para no tener ambigüedad en las direcciones origen y destino, se busca un EE no usado, en caso de no existir se usa el que tenga menor tráfico, sólo se necesita un EE para lograr hacer llegar la información por medio Multicast y Broadcast de la MC. La PC crea una trama de información que se manda al microcontrolador, el cual retransmite la trama a cada ES activo. El microcontrolador distingue esta trama para que el CL pueda extraer la información.

La Figura 23 muestra el formato de trama el cual se usa en todo este proceso.

Los elementos de la trama son:

- Header 1: Header de Autoenrutado, tamaño 3 bytes.
- Header 2: Header propio de ATM, tamaño 5 bytes.

- Payload : Carga útil con información de enrutado u operaciones de mantenimiento de la TR, tamaño 48 bytes.

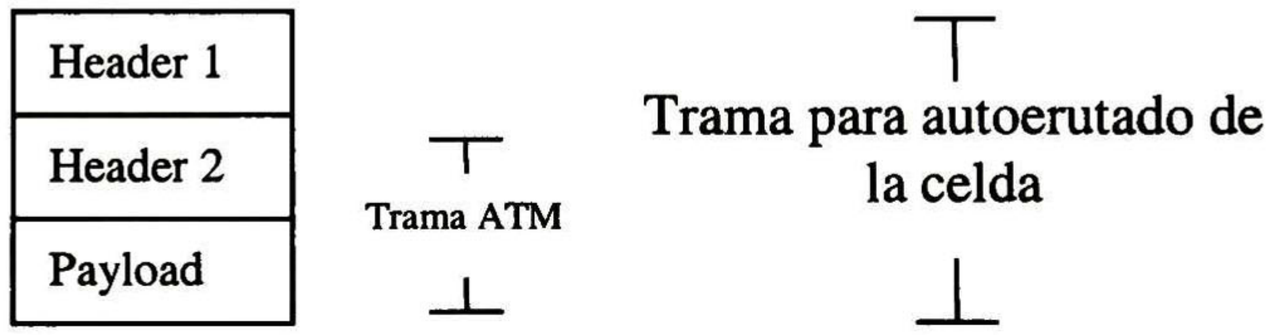


Figura 23 *Formato de trama para la comunicación con los CL desde la PC a través de la interfaz*

El acceso a los CL´s es unidireccional, es decir, sólo se puede mandar información del microcontrolador a los CL por medio de la MC, pero en sentido contrario no fue contemplado, ya que esto involucraba un EE adicional solamente para recibir, perdiendo una considerable cantidad de información al interrumpirse para una confirmación.

Los CL le dan compatibilidad a la MC con cualquier elemento que soporte el estándar UTOPIA nivel II ver [5].

El driver fue desarrollado en Builder C++ 3.0, dándonos acceso al puerto serie como uno de sus componentes.

5.3 Resumen del capítulo

En este capítulo se explica el firmware con soporte del plan de pruebas. Se explica que el protocolo descrito ha sido implementado como medio de comunicación entre un microcontrolador AT89S8252 y una PC, donde el microcontrolador es usado en la MC para realizar las pruebas y diagnósticos del mismo. Las tareas que la MC realiza son operaciones que se requiere monitorear, donde el microcontrolador puede realizar la verificación de los EE y ES.

Los datos obtenidos son estadísticas de EE y ES. También se obtiene el desempeño de la MC como CL activos, flujo de información y celdas perdidas. La capacidad de observación es importante para que la MC se pueda probar, y poder realizar depuración y pruebas.

El control de los CL desde software es importante para que la MC se pueda monitorear y depurar desde un lugar remoto sin que el usuario tenga que observar el problema, ya que al obtener los resultados por medio de esta interfaz, se pueda saber con anticipación los problemas que se presentan.

Así se ha logrado implementar un control de los CL los cuales tienen que ser configurados por algún dispositivo externo. Para ahorrar el uso de otra computadora u otra interfaz que proporcione este servicio, se usa el microcontrolador para lograr el acceso a los CL por medio de los EE y ES. La tarea que se implementan es el llenado de la Tabla de Enrutado TR (elemento en el CL que nos permite el enrutado de la información para ser entregado a la MC, como se explica en [3]).

El acceso a los CL se realiza de la siguiente manera; la TR se localiza en la PC, se coloca la TR en todos los CL conectados para no tener ambigüedad en las direcciones origen y destino, se busca un EE no usado, en caso de no existir se usa el que tenga menor tráfico,

sólo se necesita un EE para lograr hacer llegar la información por medio Multicast y Broadcast de la MC, la PC crea un trama de información que se manda al microcontrolador el cual retransmite la trama a cada ES activo, el microcontrolador distingue esta trama para que el CL pueda extraer la información.

Los elementos de la trama son:

- Header 1: Header de Autoenrutado.
- Header 2: Header propio de ATM.
- Payload : Carga útil con datos de enrutado u operaciones de mantenimiento de la TR.

El acceso a los CL's es unidireccional, es decir, sólo se puede mandar información del microcontrolador a los CL por medio de la MC, pero en sentido contrario no fue contemplado, ya que esto involucraba un EE adicional solamente para recibir, perdiendo una considerable cantidad de información al interrumpirse para una confirmación.

Los CL le dan compatibilidad a la MC con cualquier elemento que soporte el estándar UTOPIA nivel II.

6 Desarrollo de drivers de prueba, verificación y monitoreo de la MC

6.1 Utilización de los Drivers de Windows

Para el desarrollo del device driver se investigó la forma en que Windows toma el control de los periféricos. Existen dos maneras propuestas por las cuales se puede lograr usar los periféricos de la PC, que son:

- Control de periféricos por herramientas de Windows.
- Control de periféricos por medio de drivers de Windows.

6.1.1Control de periféricos por herramientas de Windows

Una manera de hacerlo, es obtener el control del periférico por un medio indirecto, como lo fue el Builder C++. La herramienta nos da la posibilidad de que el puerto sea transparente a nosotros, de esta manera se logra evitar el problema de usar el periférico de forma manual (device driver.) Los problemas de tener acceso a los puertos en forma manual, como sería el caso que explicamos a continuación, nos da como resultado una gran complejidad de implementación, ya que involucra el conocimiento profundo de Windows y su programación, así como tener que aprender paquetes nuevos, protocolos de comunicación entre las capas de Windows, etc. Estos problemas serán expuestos en la sección siguiente.

El driver se implementó por medio de "Componentes" de Builder C++ obtenidos de INTERNET ver el sitio de Internet [6], estos componentes nos dan la facilidad de tener un objeto visual el cual es interfaz para los drivers de Windows, y sólo se tiene que aprender sus propiedades y sus eventos relacionados.

El componente es un objeto de la clase derivada VCOMM, clase que se encarga de dar servicio a los puertos serie y paralelo, como se muestra en la Figura 24. Se muestra como Windows define la clase que se encarga del control del puerto serie.

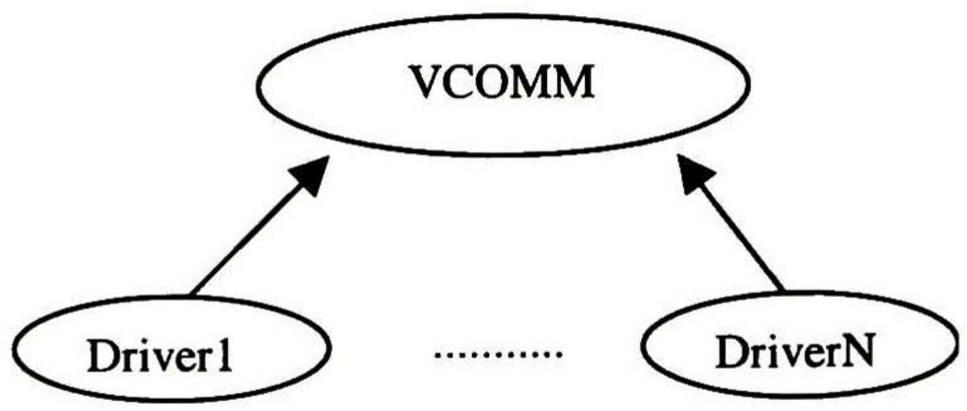


Figura 24 *Clase VCOMM*

Lo que podemos configurar del componente es velocidad de transmisión, bits de paro, paridad, tamaño de buffers de entrada y salida, y una función de interrupción que se genera al recibir un flujo de datos provenientes del puerto. La Figura 24 muestra las clases derivadas a partir de la clase base VCOMM.

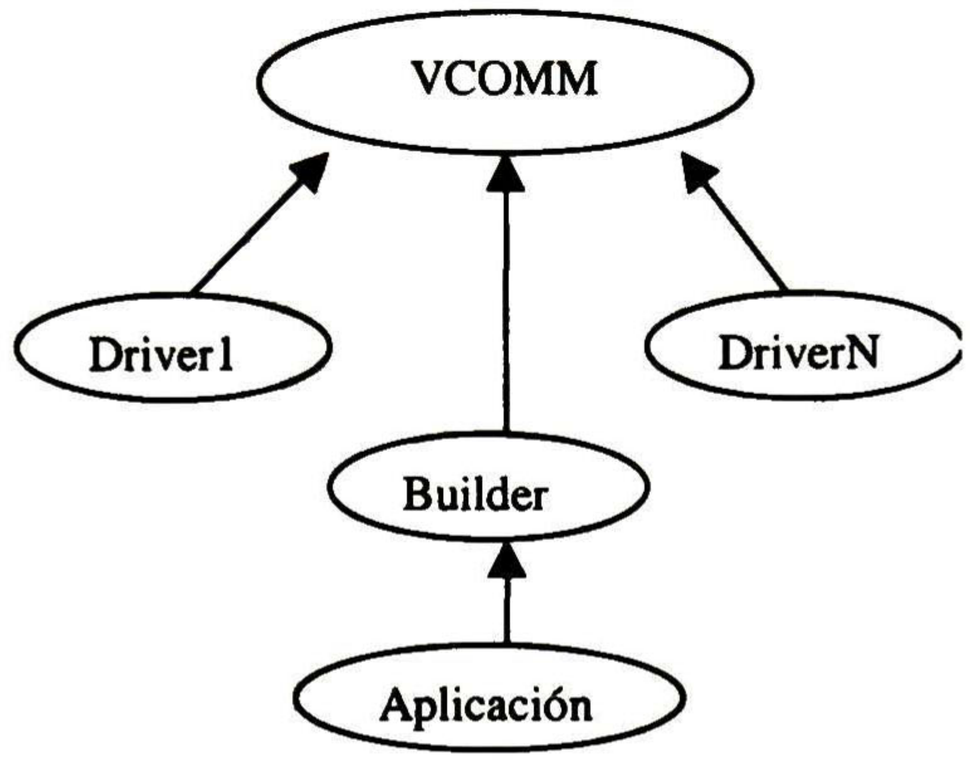


Figura 25 *Árbol de herencia entre las clases de Windows y Builder C++*

El elemento que llamamos *Builder* en la Figura 25 se refiere a los métodos del lenguaje para el control del puerto serie.

6.1.2Control de periféricos por medio de drivers de Windows

Una manera es usar VtoolsD para realizar un prototipo de un VxD (Virtual Device.) Esta alternativa genera problemas para el programador, de complejidad y desconocimiento de documentación de los dispositivos de Windows.

Desglosaremos cómo se implementa un driver al usar VtoolsD para este caso en particular:

VtoolsD nos deja realizar un driver llamado VxD, el cual es una aplicación que corre al nivel de sistema operativo. Existen dos tipos de VxD, que son estáticos y dinámicos. La diferencia esta en que el primero permanece en toda la sesión, y el segundo se puede activar y desactivar en cualquier instante.

El VxD encargado en los puertos es estático, denominado VCOMMPORT, que no permite que el usuario o programador pueda generar un driver especializado para el control de los puertos serie y paralelo, por lo que realizar un VxD estático de Windows no es fácil. La Figura 26 muestra el problema de la sustitución la clase base VCOMM por otra. Sin embargo es posible hacer un VxD que nos permita el acceso al puerto.

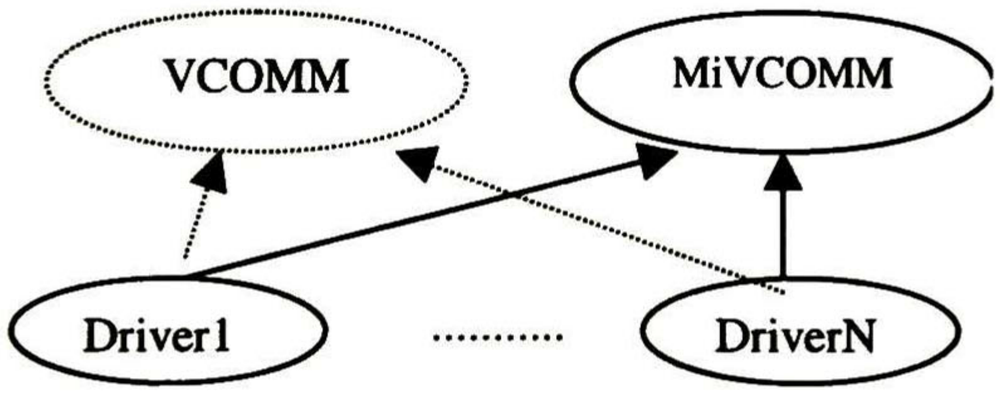


Figura 26 *Intercambio de la clase base por otra construida por el usuario*

La forma alternativa es realizar un VxD que tenga acceso a la clase derivada de VCOMM y al crear un objeto se pueda heredar las funciones de la clase base. Esta forma permite tener acceso al puerto de manera indirecta, pero es realizable por el uso del objeto que heredó los métodos de VCOMM. La Figura 27 muestra el árbol de herencia.

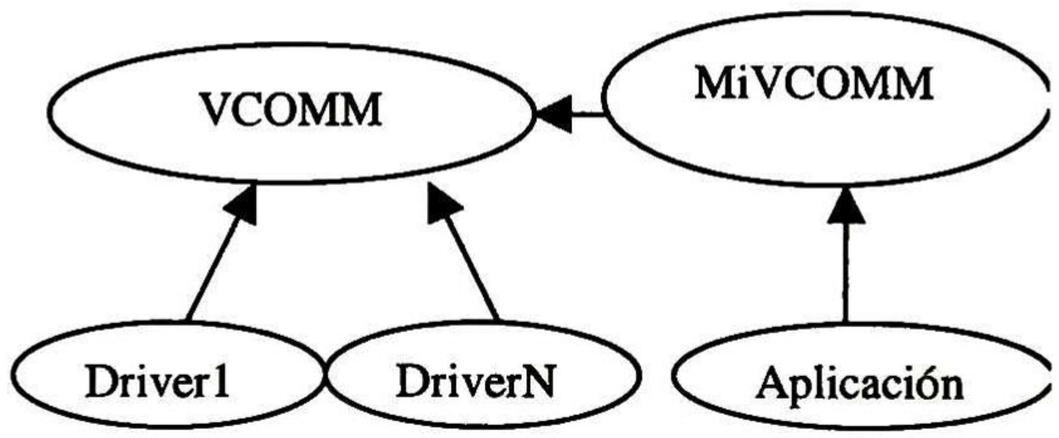


Figura 27 *Forma de cómo realizar un driver para el puerto serie*

Este esquema presenta el problema de la falta de conocimiento de los métodos de la clase base. El problema es mayor por el tiempo de recopilación de información que nos genera retardos considerables, que no son aceptables para seguir esta opción.

Sin embargo, la complejidad de ésta implementación es alta en comparación con la primera alternativa, así se optó por rechazar ésta alternativa.

6.1.3 Comparación entre las técnicas de realizar un driver para el puerto serie

Comparando ambas formas, la primera opción tiene una desventaja respecto a esta última, la velocidad. La primera opción se ejecuta a una prioridad baja, por ejemplo, la prioridad que tiene Word o aplicaciones como estas. El VxD de la segunda opción nos da mejor tiempo de ejecución compartiendo con el sistema operativo mismas prioridades.

Se optó por el componente de Builder C++ que es más fácil de usar e implementarlo.

6.2 Verificación de los drivers obtenidos

Se programó emuladores de los drivers en Turbo C. Se escogió esta herramienta por la facilidad de uso e implementación. Dada la modularidad y uso de un lenguaje estándar se puede reutilizar los códigos en otras aplicaciones como Builder C++ y el compilador de C para el 8051, herramientas finales de implementación de los drivers. En la Figura 28 se muestra cómo se han implementado los drivers y las diferentes pruebas desarrolladas para estas aplicaciones.

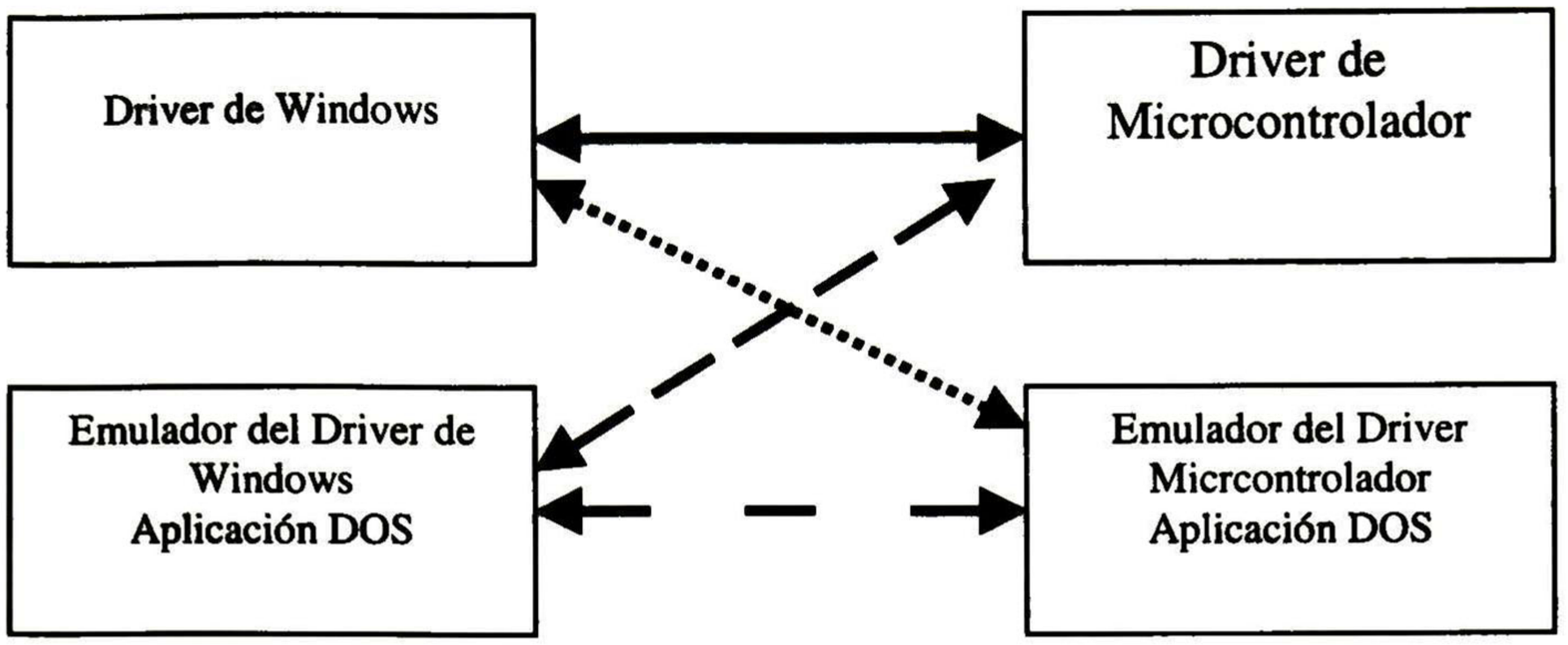


Figura 28 *Diferentes modelos de comunicación en este proyecto*

A continuación se muestra los pasos para el desarrollo de los drivers:

- Comunicación entre los emuladores.
- Comunicación entre el driver de Windows y el emulador del microcontrolador.
- Comunicación entre el emulador de Windows y el driver del microcontrolador.
- Comunicación final.

6.2.1Comunicación entre los emuladores

Teniendo desarrollado el protocolo descrito anteriormente, se realizó lo siguiente:

Figura 29 *Primer modelo de la comunicación entre dos elementos por medio del protocolo*

El protocolo se implementó en este ambiente. Después se desarrolló la funcionalidad que representa a los elementos finales, como los comandos a usar, qué mensajes se transmiten, formatos de trama etc.

6.2.2Comunicación entre el driver de Windows y el emulador del microcontrolador

El siguiente paso en el desarrollo de estos drivers es la creación de la aplicación. Se optó en realizar primero la parte de Windows, se reutilizó el código y se implementó un ambiente

visual en el cual se puede ver fácilmente los dispositivos conectados a la MC, así como las estadísticas de la MC que se han propuesto para observarse.

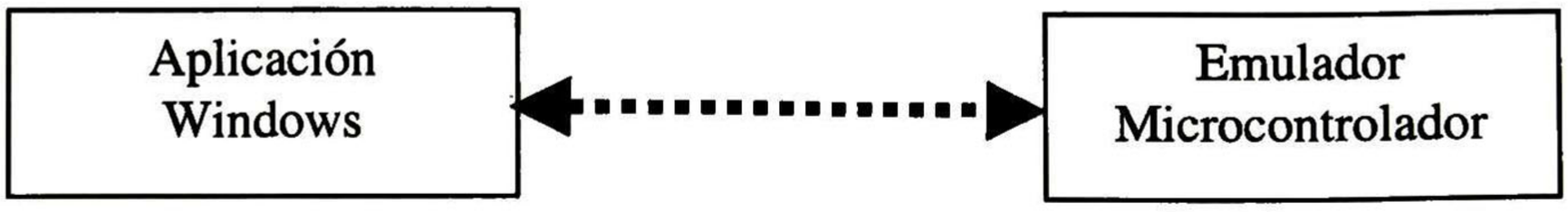


Figura 30 *Segundo modelo de comunicación entre la aplicación de Windows y el emulador del microcontrolador*

6.2.3Comunicación entre el emulador de Windows y el driver del microcontrolador

Esta opción presenta más dificultad para depuración por el uso de un compilador, ya que el programador no tiene el control de las variables, memoria, etc. que nos permite el ensamblador.

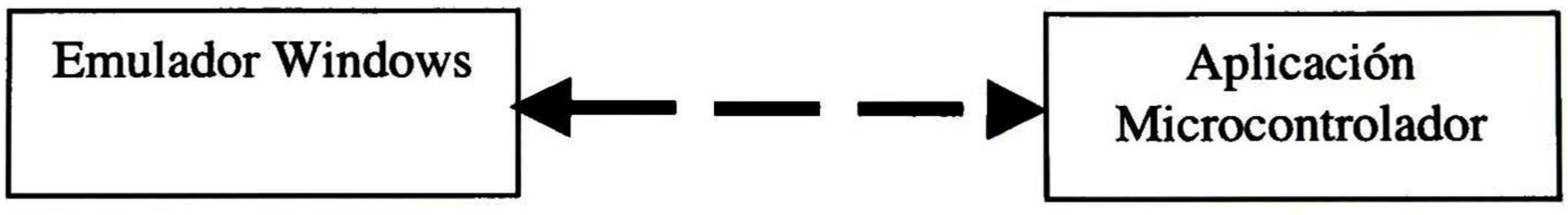


Figura 31 *Tercer modelo de comunicación entre el emulador de Windows y el driver del microcontrolador*

6.2.4Comunicación versión final

La versión final se logró dado que todas las anteriores funciones han realizado sus tareas adecuadamente. No se presentó ningún tipo de problema.

Con esta idea se desarrollo un driver que tiene las siguientes características:

- Contiene el protocolo descrito anteriormente en el capítulo 3.
- GUI para la interfaz con el usuario.
- Desarrollado para probar los diferentes elementos a probar.

6.3 Resumen del capítulo

En este capítulo se plantea el desarrollo de Drivers de prueba, la verificación y monitoreo de la MC, comenzando con el uso de los drivers de Windows, de lo cual se investiga la forma en que Windows toma el control de los periféricos. Existen dos maneras propuestas por las cuales se puede lograr usar los periféricos de la PC, que son:

- Control de periféricos por herramientas de Windows:

Obteniendo el control del periférico por un medio indirecto, como es por medio de Builder C++. Esto da la posibilidad de que el puerto sea transparente al usuario, evitando el uso de forma manual (device driver.)

- Control de periféricos por medio de drivers de Windows.

Una manera es usar VtoolsD para realizar un prototipo de un VxD (Virtual Device). Esta alternativa genera problemas de complejidad y desconocimiento de documentación de los dispositivos de Windows.

La manera en como Vindows nos deja hacer esto es que VtoolsD nos deja realizar un driver llamado VxD. Existen dos tipos de VxD, estáticos y dinámicos, de los cuales la diferencia estriba en que el primero está presente en toda la sesión y el segundo se puede activar en cualquier instante.

El VxD encargado en los puertos es estático, denominado VCOMMPORT, que no permite al usuario generar un driver especializado para el control de los puertos serie y paralelo.

Comparando ambas formas de crear drivers, la primera opción tiene la desventaja de la velocidad. La primera opción se ejecuta a una prioridad baja, el VxD de la segunda opción nos da mejor tiempo de ejecución compartiendo con el sistema operativo mismas prioridades. Se optó por el componente de Builder C++ que es más fácil de usar e implementar.

Se programaron emuladores de los drivers en Turbo C y se mostró cómo se implementan los drivers y las diferentes pruebas desarrolladas para estas aplicaciones.

A continuación se muestra los pasos para el desarrollo de los drivers:

- Comunicación entre los emuladores.
- Comunicación entre el driver de Windows y el emulador del microcontrolador
- Comunicación entre el emulador de Windows y el driver del microcontrolador.
- Comunicación final.

Se llevaron a cabo cada una de las implementaciones de comunicación antes mencionadas, donde la implementación de comunicación entre el emulador de Windows y el driver del microcontrolador presentó mayor complicación para depuración.

Como ventajas que se pueden pensar sobre el uso de drivers son:

- Facilidad para unir el driver con ambientes gráficos.
- Facilidad de entendimiento de la prueba, al usar GUIs.

Desventajas

- Conocimiento de la implementación de GUI en ambientes Windows.
- Mayor programación para el control de lo que se prueba.

7 Realizar pruebas conjuntas con el CL interfaz E1/T1

La interfaz CUU (Convertidor UTOPIA nivel 1- UTOPIA nivel 2) se desarrolló como elemento interno del desarrollo de [3], donde es necesario poder conectar una interfaz de capa física de UTOPIA nivel1 con una interfaz de capa ATM de UTOPIA nivel 2. Se muestra continuación en un modelo a bloques:

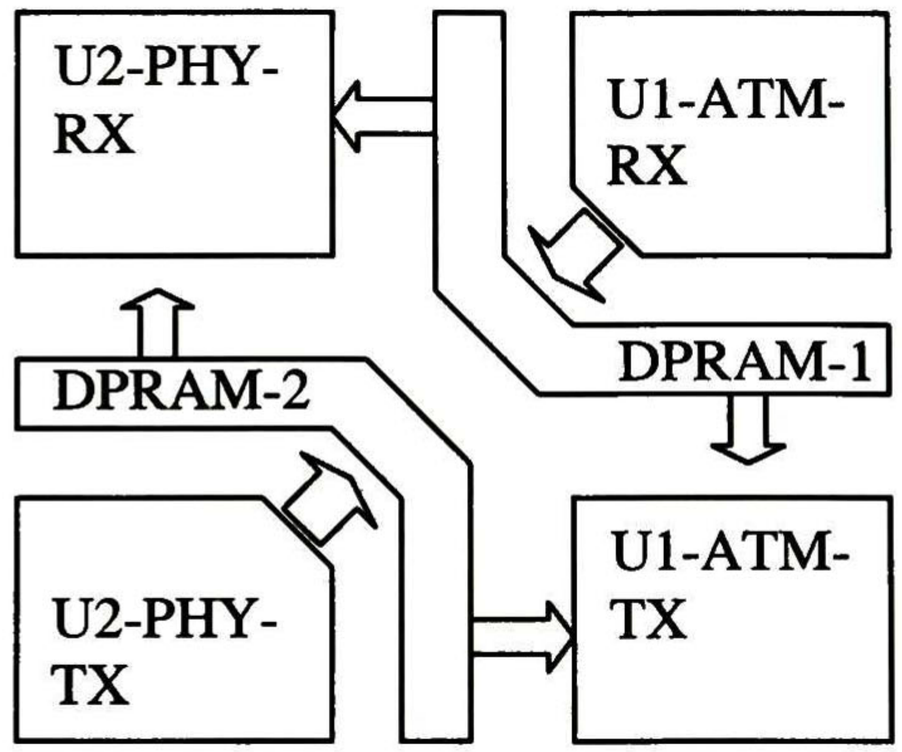


Figura 32 *Convertidor UTOPIA nivel 1 - UTOPIA nivel 2.*

Se explica a continuación cada elemento de la Figura 32.

- U2-PHY-RX.
- U2-PHY-TX.
- U1-ATM-RX.
- U1-ATM-TX.
- DPRAM1.
- DPRAM2.

U2-PHY-RX : Es el elemento de conexión de UTOPIA nivel 2 de la capa física de la etapa de recepción.

U2-PHY-TX : Es el elemento de conexión de UTOPIA nivel 2 de la capa física de la etapa de transmisión.

U1-ATM-RX : Es el elemento de conexión de UTOPIA nivel 1 de la capa ATM de la etapa de recepción.

U1-ATM-TX : Es el elemento de conexión de UTOPIA nivel 1 de la capa ATM de la etapa de transmisión.

DPRAM1 : Memoria que conecta opcionalmente la capa U1-ATM-RX con las capas U2-PHY-RX o con la capa U1-ATM-TX; la función principal es ser un banco de transferencia intermedio, esta memoria tiene la capacidad de almacenar hasta 2 celdas completas de información.

DPRAM2 : Memoria que conecta opcionalmente la capa U2-PHY-TX con las capas U2-PHY-RX o con la capa U1-ATM-TX; la función principal es ser un banco de transferencia intermedio, esta memoria tiene la capacidad de almacenar hasta 2 celdas completas de información.

7.1 Funcionalidad del CUU

El CUU está diseñado para unir una capa física de ATM de UTOPIA nivel 1, con una capa ATM de UTOPIA nivel 2. Como se sabe UTOPIA nivel 2 conecta una capa ATM con una o varias capas físicas. El CUU simula la presencia de varias capaz físicas.

El CUU puede configurarse de dos maneras:

- Modo normal de transferencia.
- Modo de Loopback.

7.1.1 Modo normal de transferencia

Este modo permite la transferencia de datos de una capa ATM a física o viceversa, en este caso la conexión de los elementos internos es como se muestra a continuación.

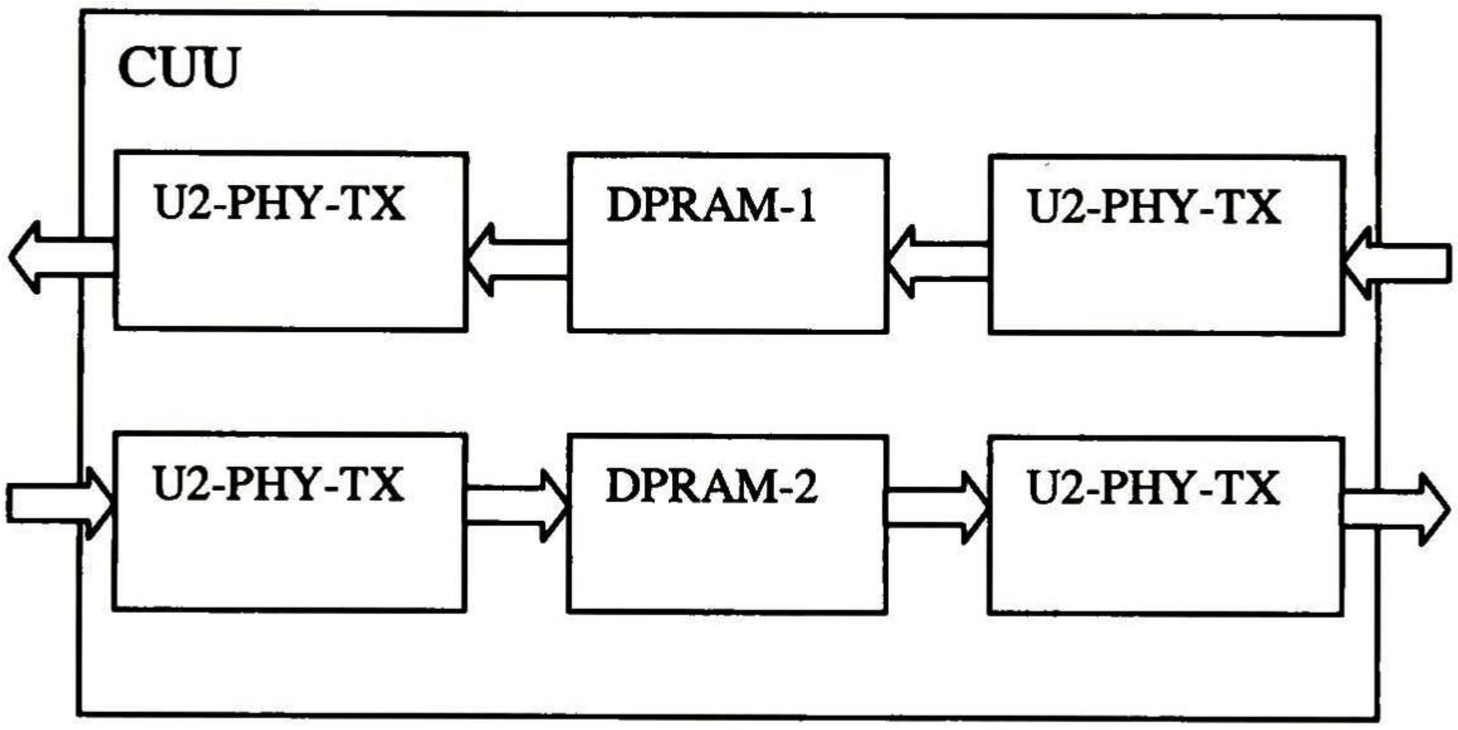


Figura 33 *CUU en modo de operación normal.*

En este modo las memorias se conectan de tal manera que el flujo de información pueda fluir desde UTOPIA nivel 2 hasta UTOPIA nivel 1 en ambos sentidos del flujo.

7.2 Pruebas para el microcontrolador y el CL

Existen dos variantes para la prueba del CL, consistiendo en dos modos de operación del CL, a continuación se exponen:

- Escritura a la TR.
- Modo de "loopbacks."

7.2.1Escritura a la TR

Consiste en realizar una escritura desde la PC hasta el CL. Los pasos considerados para esta prueba son:

- Conectarse al microcontrolador.
- Inicializar la comunicación.
- Transmitir los datos al microcontrolador.
- Retransmitir al CL por medio de la MC.

Una vez que la información llega al CL, se graba en la memoria la cual almacena el contenido de la TR, actualizando así la información en el CL.

7.2.2Modo de Loopbacks

La prueba conjunta entre estos dos módulos se basa en poder realizar un "loopback" con y sin el CL, estos dos métodos se describen ahora:

- "Loopback" sin el uso de un CL.
- "Loopback" con un CL conectado.

7.2.2.1 Loopback sin el uso de un CL

El realizar un "loopback" a la MC nos permite saber si la MC se comunica al exterior, debido a que no se puede realizar un lazo cerrado de un EE a un ES dado la configuración TOPIA, la solución es integrar a la lógica de cada puerto la Interfaz UTOPIA/TOPIA.

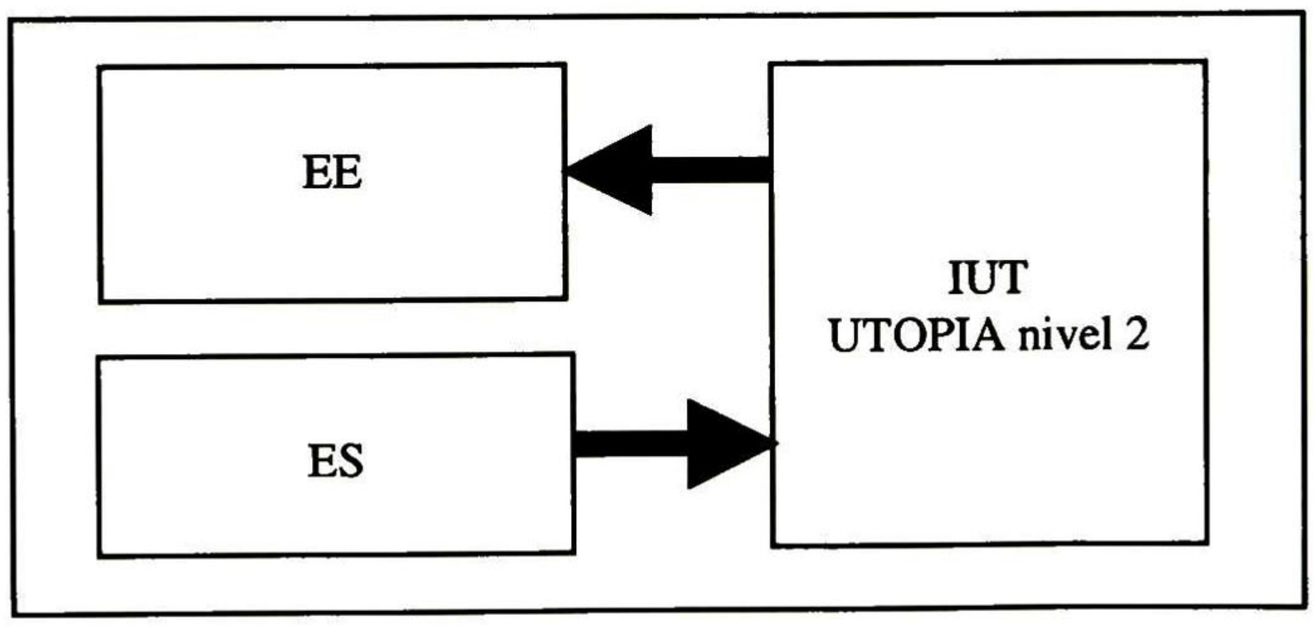


Figura 34 *Puerto de la MC con la inclusión del CUT.*

La conexión que se muestra se realizó para simular que un CL toma la información y la regresa de nuevo a la MC, ya que el bloque CUT tiene la capacidad de realizar un "loopback" de tal manera que regresen los datos de un EE a un ES. Para esta prueba se necesita un código en el que se unan la actual lógica de un puerto y la de CUT. En resumen, esta tarea es unir básicamente dos cajas negras.

7.2.2.2 Loopback con un CL conectado

Esta prueba consiste en realizar un "loopback" en el CL, el cual regresará las celdas escritas a la MC, así se puede comprobar que la información se conmuta por toda la MC hasta el CL. Este modo de prueba implica la CUT se encuentre externa a la MC. En ambos casos el bloque es idéntico son independientes del lugar de la implementación.

7.3 Resumen del capítulo

Se realizaron pruebas conjuntas con el CL interfaz E1 / T1 G. López.

Se explica cada elemento que interviene en el Convertidor UTOPIA nivel 1 – UTOPÍA nivel 2: U2-PHY-RX, U2-PHY-TX, U1-ATM-RX, U1-ATM-TX, DPRAM1 y DPRAM2.

Se explica la funcionalidad del CUU. Este está diseñado para unir una capa física de ATM de UTOPIA nivel 1, con una capa ATM de UTOPIA nivel 2. UTOPIA nivel 2 conecta una capa ATM con una o varias capas físicas; el CUU simula la presencia de varias capaz físicas. Puede configurarse de dos maneras:

Modo normal de transferencia: Este modo permite la transferencia de datos de una capa ATM a física o viceversa.

Modo de Loopback.

Existen dos variantes para la prueba del CL, consistiendo en dos modos de operación del CL:

Escritura a la TR

Consisten en realizar una escritura desde la PC hasta el CL, los pasos considerados para esta prueba son: a)Conectarse al microcontrolador, b)Inicializar la comunicación, c)Transmitir los datos al microcontrolador, d)Retransmitir al CL por medio de la MC. Una vez que la información llega al CL se graba en la memoria la cual almacena el contenido de la TR, actualizando así la información en el CL.

Modo de "loopbacks"

Para la prueba conjunta entre estos dos módulos se basa en poder realizar un "loopback" con y sin el CL, estos dos métodos son: a) "Loopback" sin el uso de un CL, que permite saber si el MC se comunica al exterior, debido a que no se puede realizar un lazo cerrado de un EE a un ES dado la configuración TOPIA, b)"Loopback" con un CL conectado, donde esta consiste en realizar un "loopback" en el CL, el cual regresará las celdas escritas a la MC, así comprobar que la información se conmuta por toda la MC hasta el CL, este modo de prueba implica la CUT se encuentre externa a la MC. En ambos casos el bloque es idéntico son independientes del lugar de la implementación.

8 Conclusiones

Al término del presente documento, se tiene los siguientes resultados y conclusiones que podrán servir como resumen; lo que se desarrollo y lo que se podría plantear para trabajo futuro.

Con respecto al objetivo agregar a la MC el soporte del protocolo UTOPIA nivel 1.

Se logró el desarrollo de un convertidor de TOPIA a UTOPIA y viceversa, el cual se puede usar en otros dispositivos o para aumentar la funcionalidad de la matriz de conmutación.

Con respecto al objetivo desarrollo de un protocolo de comunicación entre firmware y una aplicación en Windows.

Se logró el desarrollo de un protocolo de comunicaciones de propósito general para futuras aplicaciones, el cual es robusto y confiable.

Con respecto al objetivo realizar un plan de pruebas robusto para la MC.

Se pudo verificar el correcto funcionamiento de la matriz de conmutación asegurando que los elementos bajo prueba funcionan correctamente.

Con respecto al objetivo realizar firmware que soporte el plan de pruebas.

Se obtuvo un código que pude realizar el plan de pruebas especificado, este código es compatible con el protocolo de comunicaciones implementado.

Con respecto al objetivo desarrollo de drivers de prueba, verificación y monitoreo de la MC.

Se obtuvo un driver de propósito general que conecta a una PC con un microcontrolador con el protocolo anteriormente descrito, u otro similar.

Con respecto al objetivo realizar pruebas conjuntas con el CL interfaz E1/T1.

Se obtuvo un plan de pruebas que también fue implementado en el microcontrolador y que hace uso y es compatible con el driver y protocolo de comunicación, de tal manera que se puede probar en un futuro elementos que se conecten a algún puerto de entrada o salida de la matriz de conmutación y ser accesados por el microcontrolador, el driver y un software de aplicación especifica en una PC.

9 REFERENCIAS

[1] Núñez L. A, Matriz de conmutación de alta velocidad: Su verificación y su elemento de salida, Tesis de Cinvestav-IPN Gdl., 2000.

[2] Ruiz I. E, Evaluación del desempeño de un conmutador de banda amplia, Tesis de CICESE, Gdl., 1999.

[3] Lopéz L. G, Interfaz entre enlaces primarios E1/T1 y una matriz de conmutación de alta velocidad, Tesis de Cinvestav-IPN Gdl, Noviembre 2000.

[4] S. Bargagallo, M. Lobetti, A. Banso, S. Chiusano, P. Prientto, Testing embedded memories in Telecomunicacion Systems, IEEE Communications Magazine, June 1999.

[5] The ATM Forum technical Committee, Utopia Specification Level 1, version 2.01, march 21, 1994.

[6] Torry's Delphi Pages : http://www.torry.ru.

[7] González P. J., Arquitectura, Diseño e Implementación de una matriz de conmutación de alta velocidad, Tesis de Cinvestav-IPN Gdl., 1998.

[8] G. Broomell, J.R. Heath, Classification Categories and Historical Development of Circuits Switching Topologies, Computing surveys, Vol 15, No. 2, June 1983, pages 95-133.

[9] N. Mukherjee, T. J. Chakraborty, A complete test solutions for telecommunication systems, IEEE Communications Magazine June 1999.

10 BIBLIOGRAFÍA

[10] ITU-T recommendation (I.112-I.751) Integrate Services Digital Network (ISDN).

[11] ITU-T recommendation (Q.2021-Q2971) BroadBand ISDN.

[12] Torres, D., Gonzalez, J. and Guzman, M., A New Bus Assignment Algorithm for Shared Bus Switch Fabric, VLSI DESIGN, 2000, Vol. 11, No 4, pp. 339-351.

[13] *Artículo :* J. C. Silva Briano, J. Hermosillo Gutiérrez, M. A. Figueroa Salinas, D. Torres Román (Asesor), A. García García (Colaborador),"Desarrollo e implementación de un protocolo para ciertas aplicaciones de VOZ y datos", Congreso Internacional, Electro ´98, Chihuahua, Mex. Octubre 1998.

[14] Redes de computadoras, Autor: Tanenbaum.

[15] Writing Testbenches, Autor: Janick Bergeron.

11 Glosario de terminos

Bit : unidad binaria.

Byte : conjunto de 8 bits.

Gbps : Giga bytes por segundo.

Mbps : Mega bytes por segundo.

Mhz : Mega Hertz.

Multicast : (multidifusión) Modo de difusión de información que permite que ésta pueda ser recibida por múltiples nodos de la red y por lo tanto por múltiples usuarios.

Broadcast : (difusión) Tipo de comunicación en que todo posible receptor es alcanzado por una sola transmisión

Autoerutado : Es cuando un paquete de información contiene en si misma la información de su destino.

FIFO : first in, first out, esquema diseñado para memorias donde lo que primero entra es lo primero que sale.

FSM : Finite State Machine, Sistema dinámico discreto que traduce secuencias de vectores de entradas en secuencias de vectores de salida.

Interface : conexión entre dos componentes de "hardware", entre dos aplicaciones o entre un usuario y una aplicación

FLIP-FLOP : Elemento de almacenamiento de un bit.

CheckSum : suma consecutiva de bytes, para obtener un resultado que puede detectar errores en un flujo de datos.

Stack : También conocida como pila (LIFO), es un esquema donde el último dato que entra es el primero en salir.

Buffer : Memoria de almacenamiento intermedio, en ocaciones es usado para no perder continuidad en un flujo de datos.

GUI : Interfaz Gráfica de Usuario, Componente de una aplicación informática que visualiza el usuario y a través de la cual opera con ella. Está formada por ventanas, botones, menús e iconos, entre otros elementos.

INDICE

INDICE DE REFERENCIAS

INDICE DE FIGURAS

1 Arquitecturas de matrices de conmutación

El problema de la rigidez de las interconexiones es complejo, y el diseño del sistema de MC involucra los siguientes puntos.

- Método de transferencia de datos.
- Topología.

1.1 Métodos de trasferencia de datos

La transferencia de datos se refiere a los métodos usados en matrices de conmutación para lograr que un conjunto de datos se transfiera adecuadamente.

Autoenrutamiento:

- Cada control de entrada añade a cada celda entrante una extensión de enrutamiento que contiene información acerca del camino que debe tomar para llegar a su destino a través de la MC.
- La extensión tendrá tantos subcampos como etapas de conmutado a través de la MC.
- Permite a los elementos de conmutación llevar rápidamente las celdas entrantes a su destino.
- Permite que la matriz sea un medio transparente para las celdas.

Control por tablas:

- Es empleado para *indexar* las tablas dentro de los elementos de conmutación.
- Esta propiedad no descansa en las propiedades de la red de conmutación, así que una arbitraria interconexión de elementos de conmutación puede ser empleada.

  Permite la operación de multicast y broadcast, pero requiere de varias tablas de enrutamiento y traslación.

1.2 Topología

Cuando se interconectan diferentes sistemas para ser conmutados, se necesita minimizar las interconexiones. La topología de un sistema es un modelo geométrico usualmente definido por un algoritmo de interconexión que se usa generalmente para conectar varios sistemas, la misma generalmente influye en el desempeño del sistema.

Definiciones y funcionalidad.

Una MC es la que enruta los paquetes desde la entrada a la salida. Por simplicidad de representación asumiremos que todas las líneas tienen la misma capacidad, todos los paquetes son del mismo tamaño, y llegan de una manera sincrónica. Una MC ideal es la que puede enrutar todos los paquetes de entrada a salida sin perdidas y sin retraso por tráfico. La MC debe ser rápida, el reto es diseñar una MC que permita velocidades altas. Nuevas arquitecturas han surgido, y se clasifican como:

Por división Temporal.

- Memoria compartida
- Medio compartido

Por división espacial.

- Crossbar
- Clos

En secciones siguientes serán explicadas.

Estas arquitecturas tienen problemas de tamaño y velocidad, por este motivo se construyen grandes bloques de interconexión, a partir de bloques pequeños. Por ejemplo la Figura 1 es una matriz de 1024x1024 construida con bloques de 32x32.

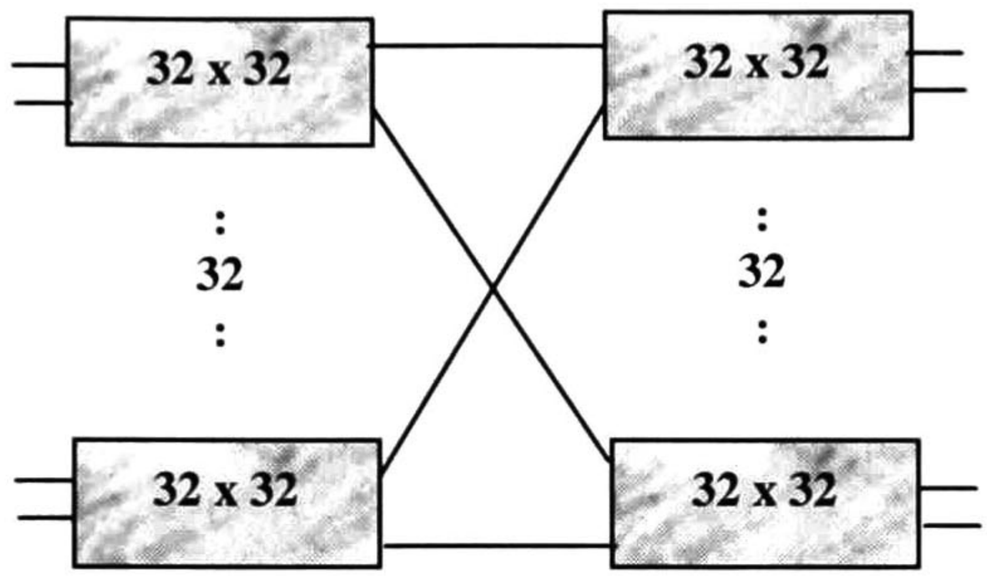


Figura 1 *Matriz 1024x1024 en bloques de 32x32*

Un factor importante para el desempeño es el tráfico de paquetes que llegan a la MC. El patrón de tráfico es determinado por:

- El proceso el cual describe la llegada de los paquetes a las entradas de la MC.
- El destino de las solicitudes de los paquetes que entran al circuito.

1.2.1Matrices de conmutación por división de temporal

1.2.1.1 MC de memoria compartida

Son muy usadas en LAN´s, consisten de una memoria de puerto dual para todas las entradas y salidas. Los paquetes llegan a todas las entradas los cuales son multiplexados en uno solo flujo de datos, y colocados en una memoria común, los paquetes son organizados en diferentes colas, una por cada salida.

Limitantes de esta arquitectura son:

- El tiempo de procesamiento requerido para determinar en que cola colocar el paquete y seleccionar la señal de control apropiada debe ser pequeña para mantener el flujo de datos.

  Los paquetes pueden perderse por el tamaño de la memoria utilizada, ésta se dimensiona para asegurar una determinada de probabilidad de pérdida. El tamaño de la memoria requerida es una función del tamaño de la MC, del tráfico ofrecido A, y del

patrón de tráfico. Como una muestra de esta dependencia, nosotros consideraremos un patrón de tráfico uniforme, generando posibles casos:

- *Partición completa*: la memoria es dividida en *n* secciones, cada una con una cola particular de salida; un paquete mandado a la salida *j*; se pierde sí la cola elegida esta llena.
- *Full sharing*: es cuando las colas pueden compartir la memoria, y un paquete se pierde si toda la memoria esta llena.

Tiempo de acceso: En tiempo de acceso es importante dada la existencia del fenómeno de cuello de botella, si el tiempo de acceso es inapropiado una opción es la arquitectura de bit-slice.

1.2.1.2 MC de medio compartido

En este tipo de MC todos los paquetes llegan a las entradas de una manera multiplexada a un medio común de alta velocidad, típicamente es utilizado un bus, pero en ocasiones se usa otro tipo de arquitectura como anillos, etc. De ancho de banda *n* veces el tamaño de cada una de las entradas.

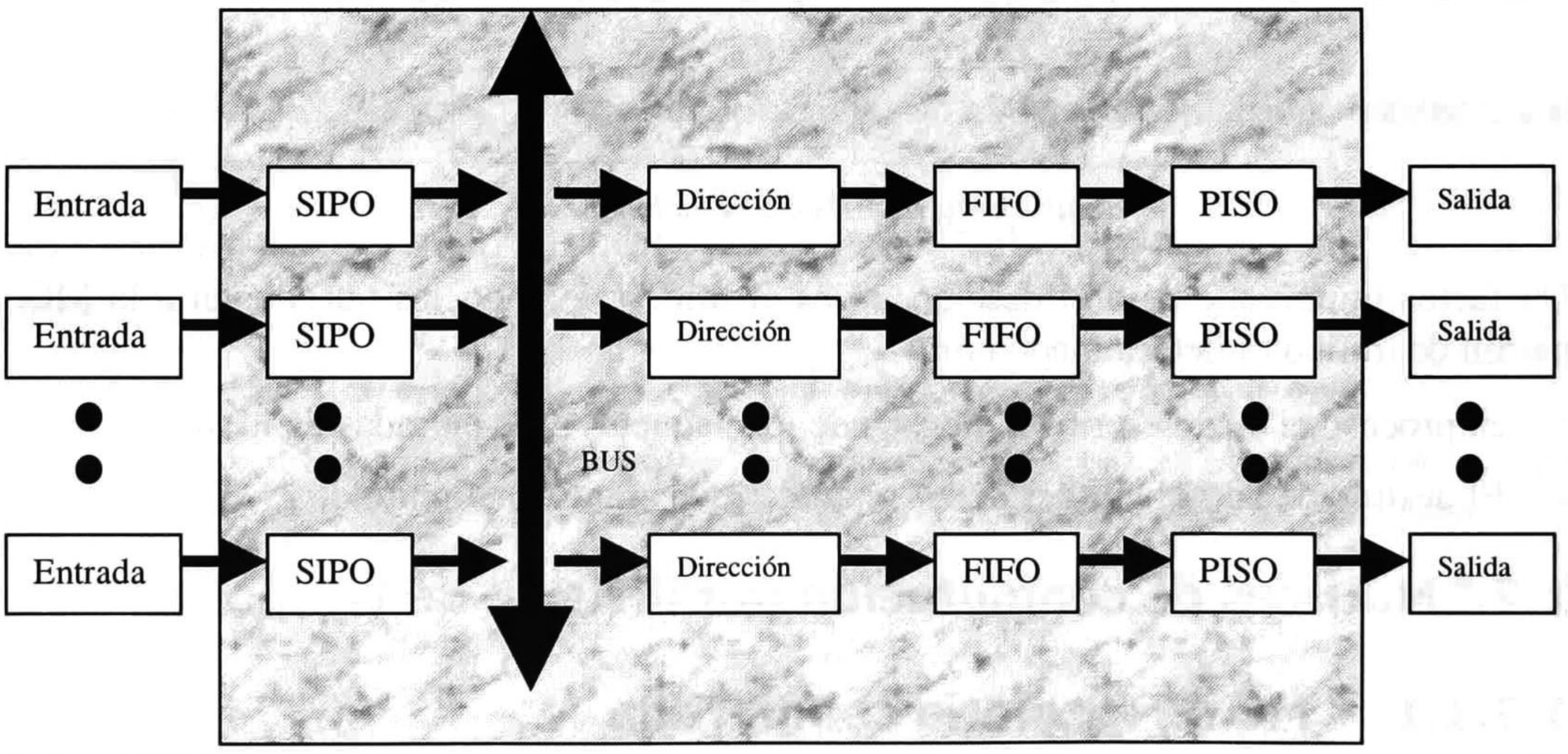


Figura 2. *Estructura básica de la arquitectura de bus compartido*

Cada salida se conecta al bus con una interfaz que filtra la información. La interfaz puede recibir todos los paquetes, y dependiendo de la dirección proporcionada determina si tomarla o no del bus, similarmente a la arquitectura memoria compartida y medio compartido, se basa en el multiplexaje de los paquetes que entran en el mismo flujo de datos, y se demultiplexa a un solo flujo de datos de salida.

La diferencia con shared-memory es que tiene particiones de memoria disponible para las colas de salida, y se organiza como FIFOs.

La restricción de este sistema es por la velocidad; se tiene que realizar paralelización de los datos para poder bajar las frecuencias de operación de los elementos de la MC.

1.2.2MC por división en espacio

Contrario a las dos arquitecturas anteriores, donde el tráfico de las entradas es multiplexado en un solo flujo de datos, en esta arquitectura, múltiples caminos son establecidos desde las entradas hasta las salidas cada uno con la misma velocidad. Como resultado no existen elementos de memoria en la MC que tengan que ser usados a velocidades mayores que la dos veces la velocidad de entrada. Otra característica es el control de la MC, no necesita ser centralizada, pero tal vez tenga que ser distribuida a través de la MC.

Este tipo de arquitectura presenta problemas consigo misma, dependiendo en particular de la MC usada y los recursos disponibles para establecer los caminos que la información recorrerá; es posible que para todos los que requieren una camino pueda ser asignado simultáneamente. Esta característica comúnmente referida como bloqueo interno, limita el flujo de salida de la MC.

Es importante colocar bancos de buffers en las MC que presentan bloqueo interno, éstos no pueden ser implementados a la salida, como en las arquitecturas anteriores. Sin embargo los buffers pueden ser colocados donde conflictos potenciales entre flujos de información puedan colisionar. Últimamente los estos bancos han sido colocados a las entradas de las MC. El colocarlos han ayudado al desempeño de esta arquitectura, así como en la implementación.

1.2.3Clasificación de topologías

Las topologías son la manera en que están organizadas en su geometría, a continuación se muestra una clasificación básica.

Basándose en su desarrollo se ha clasificado en:

- Basadas en puntas de cruce (Crossbars).
- Basadas en células de conmutación.
- Otras.

Una clasificación aún más general de las MC dependiendo de diferentes criterios son:

- Por capacidad de conexión
- Lado simple
- Doble lado
- Conector
- Concentrador
- Expansor
- Con bloqueo
- Rearreglable
- Sin bloqueo
- Sentido amplio

- Sentido débil

- Por geometría de su arquitectura
  - Una etapa
  - Multietapa
  - Regular
  - Irregular

Ahora hablaremos de estas clasificaciones, dando definiciones y características cada punto de la clasificación.

1.2.3.1 Capacidad de conexión

1.2.3.1.1 Lado simple y doble lado

La conexión es una relación entre las entradas y las salidas, pueden ser de dos maneras:

- Lado simple
- Lado doble

Lado simple: Cuando no existe diferencia entre puertos de entrada y salida, son generalmente bidireccionales ya que un puerto **i** puede ser conectado a un puerto **j**, como por ejemplo la red telefónica donde cualquier abonado es conectado a cualquier otro abonado.

Lado doble: Se distinguen 2 tipos de puertos, entradas y salidas; los puertos pueden ser unidirecionales o bidireccionales, en general estos son unión de dos tipos diferentes de dispositivos.

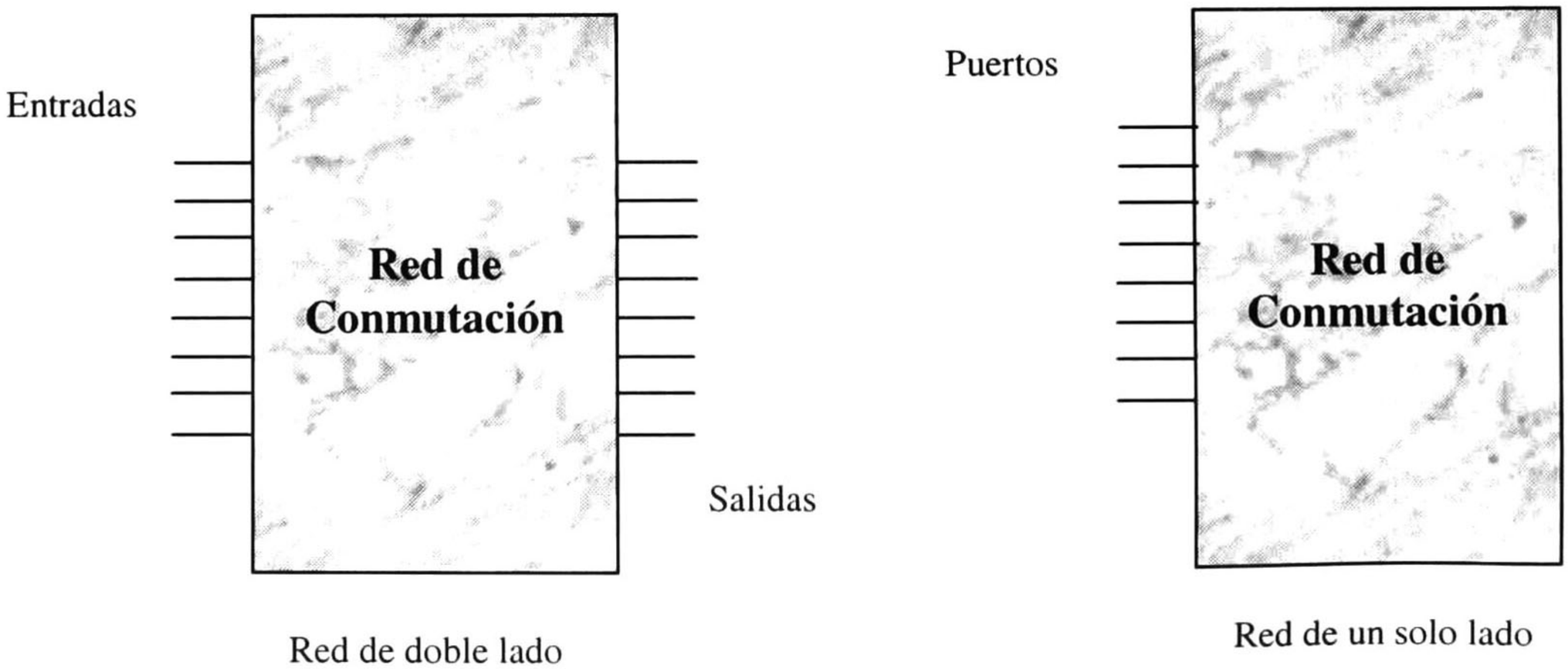


Figura 3 *Redes basadas en su capacidad de conexión*

En general esta diferencia en ocasiones no es fácil de distinguir.

Nomenclatura de conexión de red.

En general los siguientes tipos conectan una entrada libre una salida libre con siguiente notación (N, M, C).

N= Número de entradas.

M= Número de salidas.

C= Conectores.

Usando la notación anterior, se pueden representar los siguientes tipos de conectores:

Conectores de red: Asocian una entrada a una salida, llamada (i,i,i)c, donde el subíndice **c** describe al conector de red.

Concentradores de red: Asignan a una salida a varias entradas, llamada. (i,j,k)c, donde el subíndice **c** describe al concentrador de red.

Expansores de red: Asignan una entrada a varias salidas, tal y como una radiodifusora, llamada (i,j,k)e. Aun así la topología de la red es más importante que la habilidad de conectarse a la red para establecer conexiones, donde el subíndice **e** describe al expansor de red.

1.2.3.1.2 Habilidad para establecer conexiones

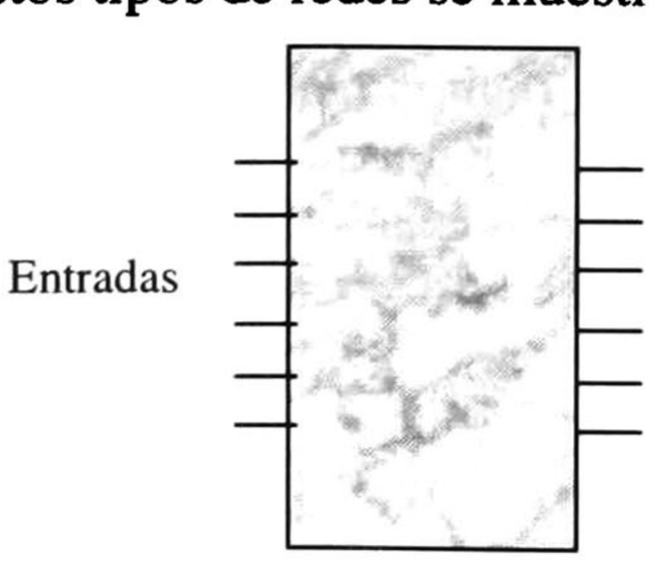
La habilidad para establecer conexiones trata sobre las restricciones que existen al intentar realizar una nueva conexión, en caso de que los recursos estén ocupados.

1.2.3.2 Relación geométrica

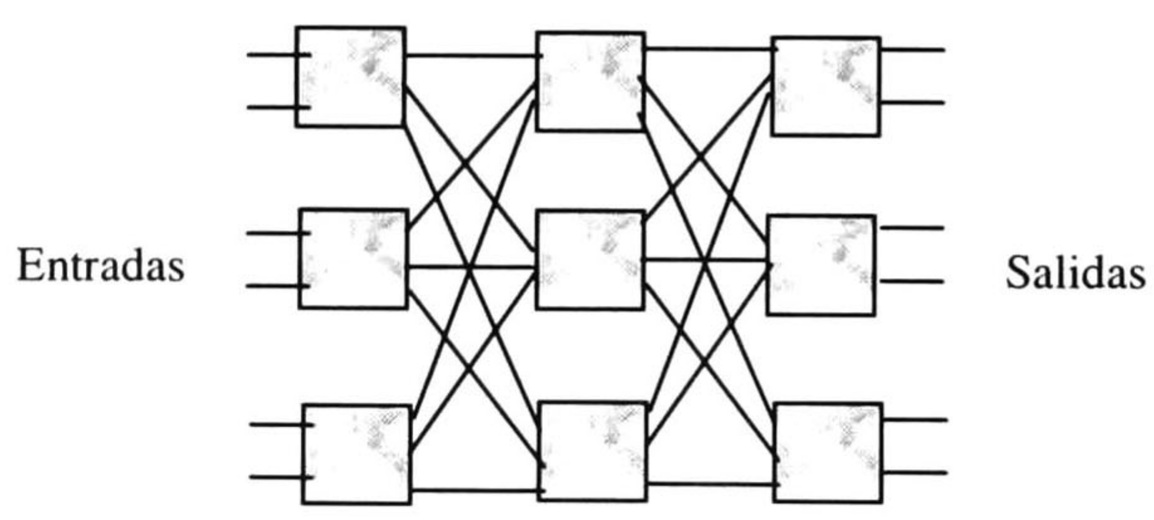
Las cualidades geométricas de las redes sirven para su clasificación. Los circuitos de conmutación de redes en general es un conjunto de elementos de conmutación interconectados bajo un patrón. Estos elementos son arreglados en dos o más etapas como se muestra en la **Figura 4**. En forma geométrica las columnas representan las etapas de los elementos del conmutador que realizan una conexión. En redes donde el *número de etapas* es diferente para cada flujo de información, es necesario usar el máximo requerido de ellas, además esta cantidad se relaciona con el retraso del circuito.

1.2.3.2.1 Redes de simple y múltiples etapas

Estos tipos de redes se muestran a continuación:



Una etapa



Múliple etapa

Figura 4. *Redes basándose en su geometría*

1.2.3.2.2 Redes regulares e irreglulares

Las redes consisten en elementos conectados siguiendo un patrón, éste puede ser irregular o regular.

Las redes *regulares* también son llamadas (redes uniformes) y se pueden clasificar como sigue simétricas horizontales o verticales, o ambas.

Las redes *irregulares* tienen a verse de alguna manera desordenada, pero siguen un patrón, las redes irregulares usan menos puntos de cruce en comparación con las redes regulares.

1.2.3.3 Bases de desarrollo

La mayoría de los puntos de inicio caen en unas cuantas categorías, las familias no muy bien definidas se traslapan, especialmente cuando se basan en algoritmos que se aplican de diferentes maneras. Siempre que se considera el análisis de redes de acuerdo a las bases de desarrollo.

En general, las redes se clasifican en dos grupos *Crossbar* y *Cell* (también llamada Nodo) a continuación se presenta algunas de ellas, basadas principalmente en propiedades matemáticas.

1.2.3.3.1 Redes crossbar

Cuando muchas entradas son interconectadas a un gran banco de puntos de cruce, Clos en 1953 mostró estas MC pueden ser interconectados en etapas con pocos puntos de cruce en un conmutador sin bloqueos como si fuera una MC completa.

Las redes de Clos han sido estudiadas, siguiendo la premisa de interconectar pequeños puntos de cruce como parte de una gran red de conmutación, como se ve en la Figura 5.

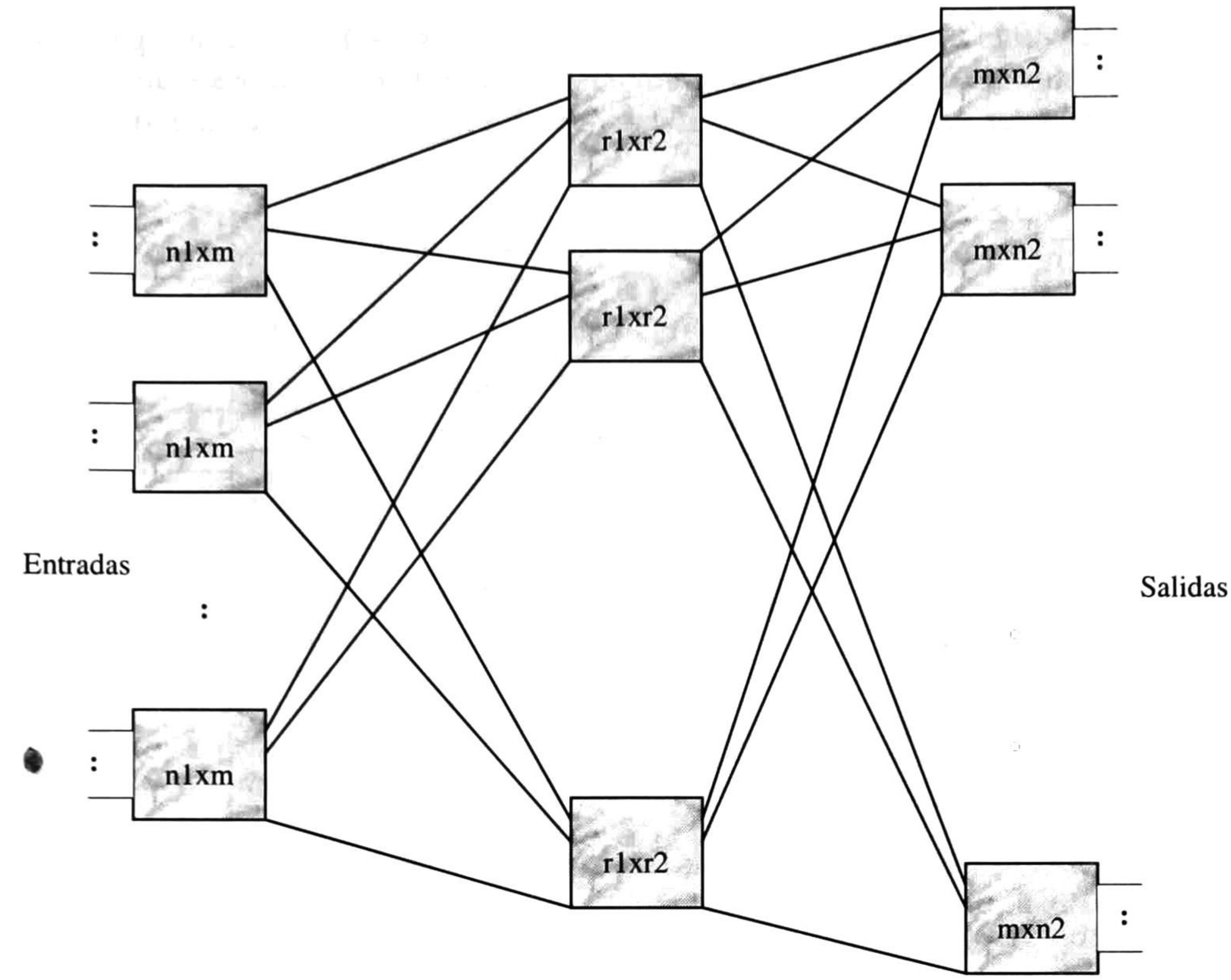


Figura 5. *MC de puntos de cruce*

A continuación se presenta algunos ejemplos de las redes basadas en puntos de cruce:

Crossbar de una etapa.

Es un arreglo donde se conecta N entradas con N salidas llamadas redes NxN, pero también existen redes NxM como se ilustra en forma general un conmutador crossbar de NxM en la Figura 6.

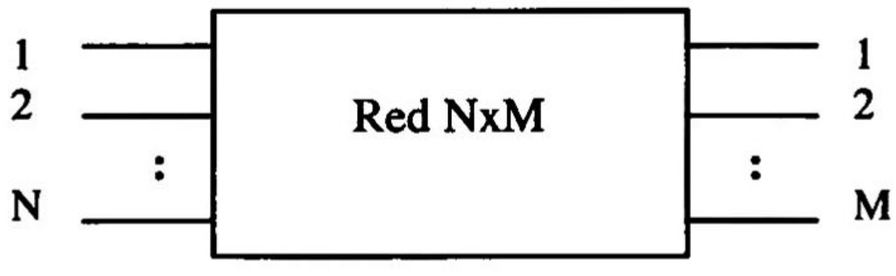


Figura 6. *Red NxM*

Llamada también arreglo rectangular, tiene la característica de no bloqueo, dado que es la misma conexión uno a uno entre entrada y salida en cualquier tiempo, no tiene posibilidad de bloqueo. En general ofrece funciones de concentración y expansión, además tiene un retardo uniforme. El costo es proporcional al número de elementos de conmutación comúnmente dicho orden N^2 aunque éste es en realidad NxM. En general, la medida de estos elementos requeridos para la topología de la red, se expresa en términos de las entradas y en ocasiones de sus salidas, estas redes son usadas para un número pequeño de entradas.

Redes de Clos.

Clos propuso en 1953 un arreglo de tres etapas relativamente pequeñas, el arreglo reduce el número de puntos de cruce para un gran número de entradas. Propuso un sistema de construcción de redes de diferentes especificaciones. La Figura 7 muestra una red de Clos.

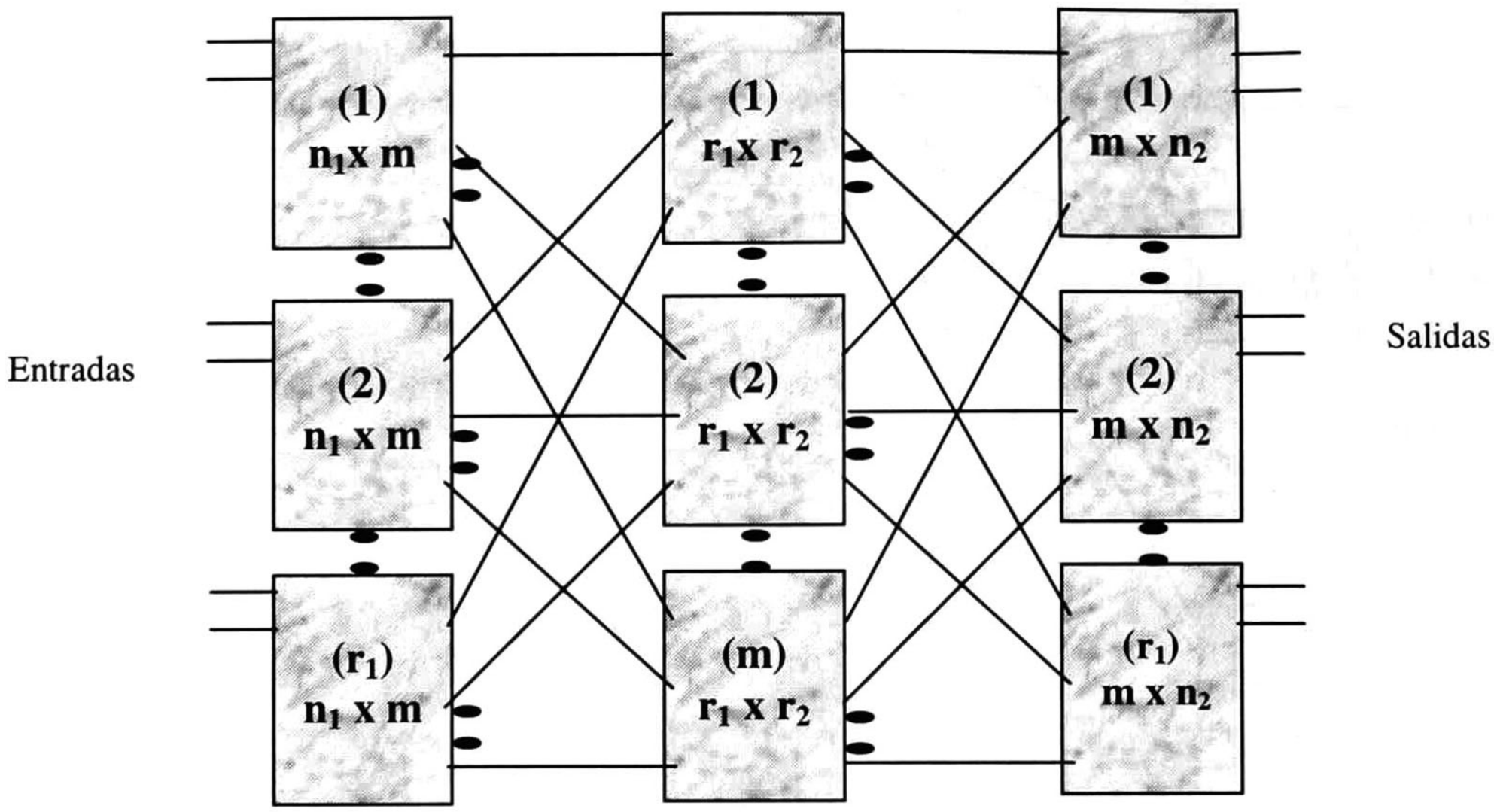


Figura 7. *Red de Clos de tres etapas* $(r_1 n_1 X\ r_1 n_2)$

Estas redes generalmente son de tres etapas, sin bloqueo y con arreglos rectangulares en todas las etapas.

1.2.3.3.2 Redes basadas elementos de conmutación

A los elementos de conmutación (**EC**) se les conoce como (células o nodos), existe un gran grupo de redes desarrolladas usando estos elementos de tamaño 2x2, como elementos básicos de construcción, ver la Figura 8.

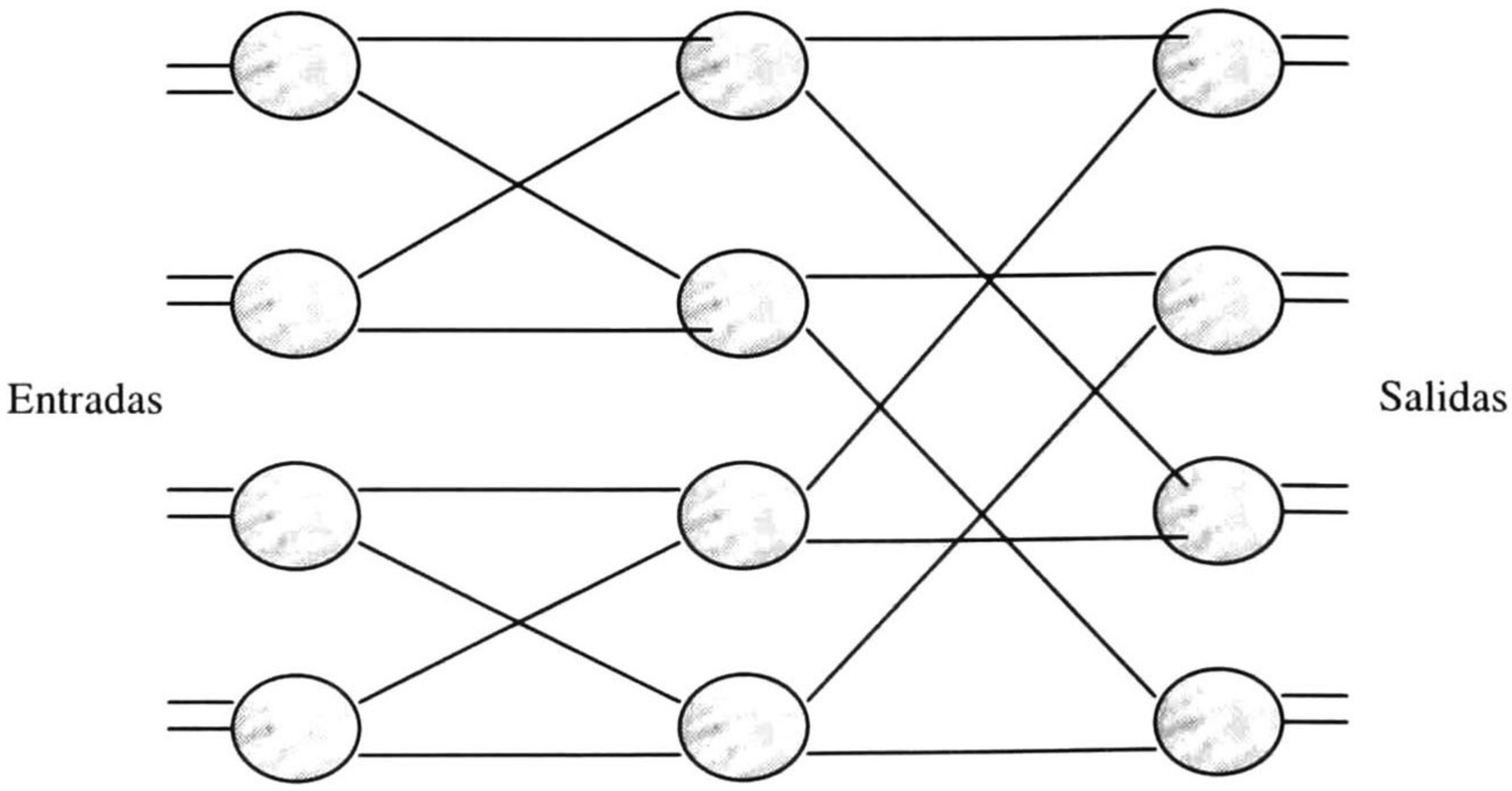


Figura 8. *Red típica de 2x2 basada en EC*

Existe una gran cantidad de redes que usan teoría de grafos para el análisis como se muestra en la Figura 9, donde se usa una red de conectores en un circuito y su representación en grafo del sistema de conexión. Entonces las líneas en el grafo representan los elementos del conmutador pero es similar al esquemático.

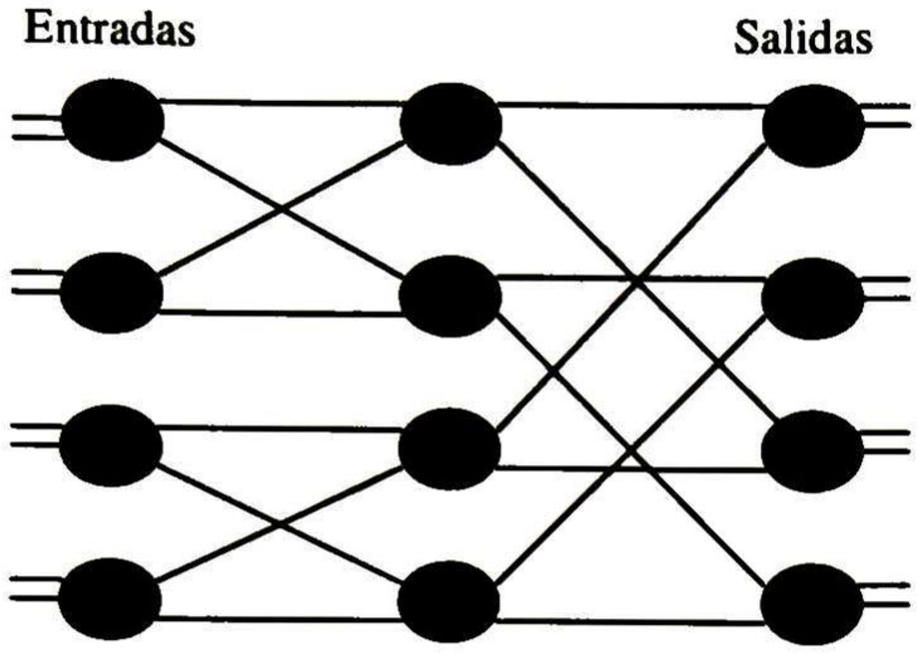


Figura 9a

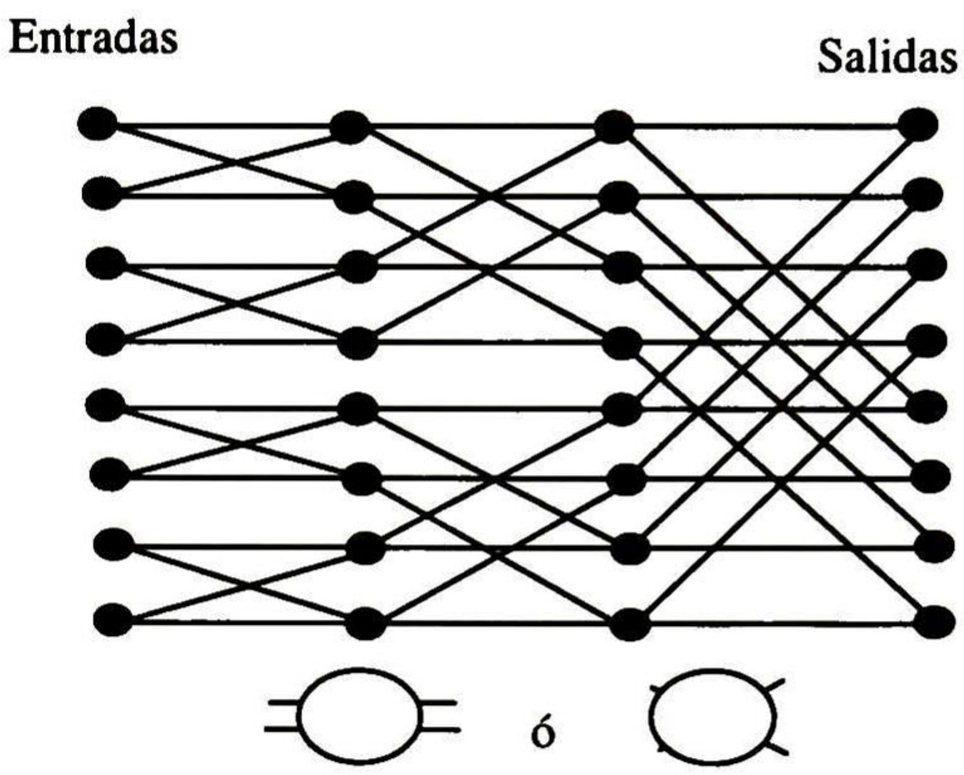


Figura 9b

Figura 9. *Relación entre topología y geometría*

En el grafo solo uno de los conductores porta el dato y en el circuito todas las líneas puede transferir datos. Es usado comúnmente el **EC** de tamaño 2x2, (Joel lo llamo elemento *Beta* en 1968 en sistemas electromecánicos, lo propuso como una eficiente simplificación para matrices de conmutación de puntos de cruce). En la construcción de redes el **EC** es útil porque tiene solo dos estados y requiere un solo bit de control, muchas redes se desarrollaron con él, sin embargo no es siempre de tamaño 2x2.

2 BIBLIOGRAFÍA Y REFERENCIAS

[1] G. Broomell, J.R. Heath, Clasification Categories and Historical Developement of Circuits Switching Topologies, Computing surveys, Vol 15, No. 2, June 1983, pages 95-133.

INDICE

INDICE DE TABLAS

1 Rediseño

Una vez definidos los objetivos de esta tesis, comenzaremos por el árbol de decisión, permitiéndonos tener formalidad para decidir que opción tomar.

1.1 Problemas

Identificar los problema que influyen en la MC es importante, se desglosa los problemas de la MC y como mejorar estos puntos críticos.

Dentro del plan de pruebas:

- Agregar elementos de software para la automatización y flexibilidad para implementar pruebas a la MC.

Frecuencia de operación del BCG:

- La elección del integrado no fue la correcta.
- El cálculo de la potencia para el BCG, no fue realizado correctamente.

Dentro de lo que es la revisión del código en AHDL, se tiene contemplado los siguientes puntos a tratar:

- No se ha modularizado correctamente.
- No existe una metodología formal de diseño para VHDL.
- El reuso de código no fue contemplado.
- La completa ausencia de métricas u otro método de medir el desempeño de códigos en VHDL.

Para medir el desempeño del código se han desarrollado métricas para lenguajes como C++ [2] basándose en ingeniería de software; hablando de VHDL no existen estudios similares, se deja a futuras investigaciones el establecer métricas de los códigos escritos para ambiente HDL.

1.2 Mejoras al árbol de decisión

Arbol de decisión, enfoque de programación lineal.

La investigación de operaciones es una rama de la ciencia relativamente joven desarrollada en la segunda guerra mundial, se utilizó para el análisis e investigación sobre la conducta de operaciones militares. Aun así el concepto de investigación es engañoso, ya que no se trata de una investigación científica, por su connotación de adelantar el conocimiento fundamental de alguna ciencia; en realidad no existe una definición aceptada para la investigación de operaciones. La visión aceptada en este texto es que se trata de una enfoque científico al análisis de muchos tipos de problemas complejos de toma de decisión (económicos, ingeniería, etc.).

Fases de un proyecto de investigación de operaciones:

- Planteamiento del problema.
- Construcción de un modelo matemático para representar la operación estudiada.
- Obtención de una solución para el modelo.
- Prueba de modelo y evaluación de la solución.
- Implantación y conservación de la solución.

La programación lineal, es una de las herramientas más importantes empleada en la investigación de operaciones. La programación lineal es una estructura matemática, que implica consideraciones específicas; y puede ser resuelta por medio de una técnica estándar de solución denominada método simplex. Cualquier problema que satisfaga las consideraciones de esta estructura puede plantearse como un programa lineal y resolverse. Por lo tanto, es un modelo de aplicación general.

A continuación se muestra la forma en que la programación lineal nos ayuda a tomar decisiones.

Según el árbol de decisión para la arquitectura (ver [1], árbol de decisión) y definamos el problema, función objetivo, variables de decisión y restricciones del sistema.

Definición del problema.

Dada la arquitectura, su desempeño está en función de un conjunto de variables que se mostrara más adelante, se pretende medir la eficiencia y como mejorar el desempeño basándose en el conjunto de variables para tomar una decisión.

Variables de decisión.

Definamos como sigue a cada elemento de desempeño de la arquitectura.

Variable de decisión	Descripción	Unidades
X_1	Cantidad de PCBs	Número de PCBs
X_2	Tamaño de PCBs	Cm^2
X_3	Conexiones de PCBs	Número de conexiones
X_4	Cantidad de componentes	Número de componentes
X_5	Frecuencia de los buses	Hz
X_6	Ancho de Buses	Bits
X_7	Escalabilidad	Si=1 , No=0
X_8	Tiempo de latencia	Seg, ms, ns
X_9	Probabilidad de pérdida de celdas	Sin unidad
X_{10}	Costo	Pesos o dolares
X_{11}	Complejidad de control	Sin definir
X_{12}	Complegidad de servicio de difusión	Sin definir
X_{13}	Tiempo de acceso a memoria	Ns

Tabla 1. Variables de decisión.

Por lo tanto nuestra función objetivo es.

$$Z = \sum_i a_i X_i$$

Donde i = 1..13, $\mathbf{a_i}$ se define como el factor de importancia de la variable en el desempeño de la arquitectura.

A continuación se muestra el factor de importancia de cada variable en el desempeño de la arquitectura, según se ha definido en [1].

Variable de decisión	Su peso en la arquitectura en %
X_1	2.5
X_2	2.5
X_3	2.5
X_4	6.75
X_5	6.75
X_6	4.5
X_7	15
X_8	5.25
X_9	9.75
X_{10}	20
X_{11}	10
X_{12}	5
X_{13}	10

Tabla 2. Pesos de las variables de decisión.

Restricciones

Restricción	Nota.
Xi>0	Todas la variables deben ser positivas.
Li<Xi<Hi	Cada variables debe tener cotas superiores e inferiores.

Tabla 3. Restricciones para las variables de decisión.

- $\mathbf{L_i}$ es el límite inferior aceptable para la variable de decisión **i**.
- $\mathbf{H_i}$ es el límite superior aceptable para la variable de decisión **i**.

Si resolvemos este sistema obtendremos la optimización de la arquitectura, en este caso la de bus común externo, este método no nos dice una comparativa entre cada arquitectura, sólo nos da un camino a seguir en su optimización, las conclusiones que se obtienen nos muestra que es lo que tenemos que hacer, para incrementar el desempeño de la arquitectura, a demás este sistema es complejo de resolver dadas la cantidad de variables, y restricciones. Sin embargo la forma que se obtiene el peso de cada variable para el desempeño de la arquitectura sigue siendo sumamente subjetiva.

Resumen matemático.

Función objetivo

Determine X_i tal que, Maximice $Z = \sum_i a_i X_i$

Restricciones

	Variables															
Res(1)	X_1														<	H_1
Res(2)		X_2													<	H_2
Res(3)			X_3												<	H_3
Res(4)				X_4											<	H_4
Res(5)					X_5										>	L_5
Res(6)						X_6									>	L_6
Res(7)							X_7								=	1
Res(8)								X_8							<	H_8
Res(9)									X_9						<	H_9
Res(10)										X_{10}					<	H_{10}
Res(11)											X_{11}				<	H_{11}
Res(12)												X_{12}			<	H_{12}
Res(13)													X_{13}		<	H_{13}
Res(14)														X_{14}	<	H_{14}
F.O.	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	=	Maximizar

Tabla 4. Resumen al problema de programación lineal.

Incluyendo las condiciones de no negatividad, $X_i>0$

Solución del problema de programación lineal.

Se puede considerar la solución de un problema de programación lineal (mediante el método simplex) es un sistema de N ecuaciones y M incógnitas. Por lo tanto en este preciso ejemplo tenemos que N=M en una matriz cuadrada diagonal. La solución de nuestro problema es el siguiente.

Variable	Solución
X_1	L_1
X_2	L_2

X_3	L_3
X_4	L_4
X_5	H_5
X_6	H_6
X_7	0
X_8	L_8
X_9	L_9
X_{10}	L_{10}
X_{11}	L_{11}
X_{12}	L_{12}
X_{13}	L_{13}
X_{14}	L_{14}

Tabla 5. Solución al problema de programación lineal.

2 Codiseño

El diseño de sistemas es una disciplina de ingeniería muy importante. Se diseñan comúnmente con un componente digital predominante y una aplicación de software. Codiseño de Hardware/software es encontrar los niveles del desarrollo de un sistema particionado adecuadamente en hardware y el software. Los problemas del codiseño se encuentran en la aplicación, tecnología de implementación, y metodología de diseño.

Se distinguen los siguientes problemas de codiseño:

- Sistemas dedicados.
- Autocontenidos (Self-contained).

En aplicaciones de sistemas dedicados en tiempo real como DSP se enfocan en la compilación de software para la arquitectura de la aplicación. Tales arquitecturas difieren desde procesadores estándares porque los recursos son diferentes. Así la tarea de compilación de software es más difícil, al mismo tiempo el desafío de compilación está en proveer alto desempeño a los programas, permitiendo la programación de estas arquitecturas con lenguajes de programación de alto nivel.

Tiene importancia el desarrollo concurrente de hardware y software dada las similitudes de enfoques en ambos. Debido al interés científico y comercial en este campo, grandes cambios tendrán lugar en los próximos años.

La amortización del hardware sugiere la idea de usar software con base en el hardware, también dado su complejidad el reuso es frecuente. Actualmente más software se encuentra sobre el mismo hardware, frecuentemente llamados sistemas sobre silicio. Así hardware y el software puede inspeccionarse como mercancías de gran valor de propiedad intelectual. Hoy ambos son sistemas cada vez más complejos y es importante su reusabilidad para construir bloques complejos.

El uso en la industria de los arreglos de compuertas de campos programables (Field Programable Gate Array FPGA) disminuye la diferencia entre la hardware y software. Con esta tecnología las funciones de un circuito de hardware son programados después de su fabricación, a diferencia donde la función de un circuito electrónico era fijado antes de su fabricación.

2.1 Distinción de aspectos de sistemas electrónicos

Se asocia el problema del codiseño con las clases de sistemas digitales de donde provienen. De Micheli intenta caracterizar sistemas usando algunos criterios generales tal como:

- Dominio de aplicación.
- Grado programabilidad.
- Implementación de características.

2.1.1Dominio de aplicación

Un sistema digital puede proveer un servicio interno (self-contained), o como parte de un sistema más grande, tradicionalmente la computadora es un ejemplo del primer tipo, los sistemas que caen en el segundo tipo son llamados sistemas dedicados (embedded systems). El término dedicado significa que se es parte de una unidad más grande y que provee un servicio dedicado a esa unidad.

Los sistemas digitales pueden clasificarse según su dominio de aplicación, los ejemplos de sistemas digitales self-contained son sistemas de procesamiento de información, abarcando desde computadoras portátiles a supercomputadoras, así como también emulación y creación de sistemas prototipos. Las aplicaciones de sistemas dedicados son usadas en la industria de fabricación, ejemplo, planta ensambladora, control de un robot, dispositivos domésticos, mantenimiento de automóviles, aviones, barcos, sistemas de defensa ambiental y territorial.

2.1.2Implementación de características

La implementación de sistema está ligado con el estilo del diseño del circuito, creando niveles de integración y tecnología. Se toca brevemente estos puntos, porque es necesario mantener una vista más general del problema que es bastante independiente de la implementación física.

Los sistemas digitales se basan en la tecnología VLSI. El estilo del diseño del circuito se relaciona con la selección de las primitivas del circuito (por ejemplo de biblioteca no especializada de una tecnología), estrategia de cronometraje (síncrono, asíncronos), y modo de operación de circuito (por ejemplo, estático y dinámico).

Un sistema puede tener componentes con diferentes escalas de integración y diferentes tecnologías de fabricación. La elección de una tecnología del hardware para los componentes del sistema afecta en el costo y el desempeño, y por lo tanto es de importancia primaria.

Cuando se considera los problemas de codiseño para agrupar sistemas, se distingue entre sistemas que consisten de componentes (como ASIC'S, procesadores, y memorias) colocados sobre una tarjeta, y el sistema consistente de un ASIC con uno o más procesadores y memorias. El núcleo programable es comúnmente un ASIC. Considerando un núcleo que puede proveer la misma funcionalidad como una parte estándar, costo y las consideraciones de desempeño estándar, puede predisponer la elección del nivel de integración. Las ventajas de VLSI son la confiabilidad, baja potencia, y aumento en el desempeño. Las desventajas son circuitos más grandes y más complejos en su verificación.

2.2 Problemas de codiseño y enfoques de diseño

Se considera diferentes facetas de diseño, específicamente los objetivos importantes en sistemas dedicados. Se describe a continuación a circuitos ISA y su uso en los sistemas

autónomos de procesamiento de información y sistemas dedicados, por último sistemas reconfigurables.

2.2.1Codiseño de sistemas dedicados

Los sistemas dedicados son elementos de sistemas más grandes, algunos sistemas proveen control a funciones del sistema total, mientras otros desempeñan funciones de información. Los sistemas dedicados de control comúnmente regulan componentes mecánicos por medio de actuadores y reciben los datos de aporte proveídos por sensores. Los sistemas de control son los sistemas reactivos, porque ellos se aplican para reaccionar a estímulos proporcionados por el ambiente. Sistemas de control de tiempo real realizan funciones que deben ser ejecutados dentro de ciertas ventanas de tiempo. Las funciones requeridas y clasificación de los controladores dedicados pueden variar ampliamente. Los más comunes son los microcontroladores y microprocesadores que proveen soluciones de bajo costo y flexibilidad a una amplia gama de problemas. No obstante, sistemas de control que son más complejos o requiere más información, necesitan diseños específicos como los DSP.

Cuando se habla de diseños dedicados, puede ser considerando el desempeño como un importante criterio de diseño para sistemas de procesamiento, la confiabilidad, disponibilidad, y la seguridad son sumamente importantes para los sistemas de control. La confiabilidad del sistema mide la probabilidad de que la operación correcta se realice por el control en presencia de algún error en algún componente, la disponibilidad contempla la reparación de componentes defectuosos, sin tener que detener el sistema, reparación in-line y la seguridad mide la probabilidad de evitar errores fatales.

Desde que las funciones de control se han implementado en el hardware y software, las disciplinas específicas de diseño deben usarse para asegurar confiabilidad, disponibilidad, y seguridad.

El problema de codiseño para sistemas dedicados incluyen modelado, validación, e implementación. Estas tareas son complejas porque la función de sistema puede ser desempeñada por componentes de naturaleza heterogénea.

2.2.2Codiseño de ISA

El concepto de ISA juega un papel fundamental en el diseño de sistemas digitales. ISA provee a la vista del programador de hardware el apoyo de un específico modelo de programa. La definición de un conjunto de instrucciones permite el desarrollo paralelo del hardware y del compilador correspondiente. Los componentes con base en ISA, es decir, por ejemplo, los microprocesadores, DSP'S, y microcontroladores se usan en los sistemas self-contained. Por lo tanto un buen diseño de ISA es crítico para lograr la facilidad de uso de un sistema.

El desarrollo de compiladores debería comenzar casi al mismo tiempo como la definición de ISA. Los compiladores necesitan el conjunto de instrucciones y la organización total de procesador, a fin de averiguar si se logran las metas del desempeño del sistema.

De manera diferente a los de propósito general y procesadores digitales de señales, los Applicaation-specific instruction set processors ASIP, pueden diseñarse para soportar un conjunto diferente de instrucciones, porque los requerimientos de compatibilidad son menos importantes y soportar un conjunto de instrucciones especificas es la meta deseada. Desgraciadamente el precio de la flexibilidad en escoger un conjunto de instrucciones crean la necesidad de hacer compiladores para aplicaciones específicas.

2.2.3Codiseño de sistemas reconfigurables

Sistemas que explotan la tecnología FPGA, donde los sistemas involucran una fase de configuración seguida por una de ejecución se les nombra reconfigurables (*evolvable,* que significa que evoluciona o sufre cambios); existen también sistemas que no son reconfigurables y sus aplicaciones son dirigidas a la aceleración de cómputo. En ambos casos, los sistemas digitales pueden incluir un subsistema reconfigurable que emula el software o la ejecución de hardware, y a veces una combinación de ambos.

En los problemas importantes del codiseño de hardware y software se necesita identificar los segmentos críticos de los programas de software y recopilarlos eficientemente para correr sobre el hardware programable. La tarea posterior es basada en algoritmos de síntesis de hardware, y mejores técnicas de optimización del desempeño de hardware.

La tolerancia a fallas en un sistema reconfigurable puede ser obtenida para detectar una unidad funcional con fallas, y reconfigurar una parte del sistema para quitar la falla en una unidad defectuosa. Esto puede lograrse bajo la suposición de que el diseño es tolerante a fallas y existen unidades bloques de respaldo disponibles.

2.3 Diseño de sistema de hardware y software

Se considera aquí el alto nivel que va dirigido al diseño de sistemas de hardware y software, también al problema en su total, más que en su profundidad para destacar similitudes y diferencias en codiseño de sistemas digitales de naturaleza distinta.

El diseño de hardware y software involucra modelado, validación, e implementación. Llamamos modelado al proceso de conceptuar y refinar las especificaciones para producir un modelo de software y hardware, validación al proceso de lograr un nivel razonable de confianza para que el sistema trabaje como se diseñó, e implementación a la realización física del hardware mediante la síntesis, y de la ejecución del software mediante la compilación.

Cuando se considera sistemas dedicados diferentes estrategias de modelado, implementación y paradigmas pueden seguirse, por lo tanto, el modelo total de un sistema dedicado involucra hardware y software. En el modelado puede obtenerse un sistema homogéneo o heterogéneo. En el caso anterior, un lenguaje de modelado como C++ o un formalismo gráfico, se usa para representar ambas porciones de software y hardware. El problema de una partición del hardware y software puede informar que partes del modelo se pueden implementaron en el hardware y que en software. El particionamiento del

sistema puede ser decidido por el diseñador, con una acotación y refinamiento consecutivo del modelo inicial, o determinado por una herramienta CAD.

Cuando se usa un modelado heterogéneo, la partición de hardware y software es planteada frecuentemente por el modelo mismo, porque los componentes de software y hardware pueden expresarse en los lenguajes correspondientes. No obstante, los diseñadores de sistema pueden querer explorar implementaciones alternativas de algunos componentes. Por ejemplo, una parte de un producto puede contener un componente desarrollado en software luego después para elevar el desempeño y/o razones de costo puede implementarse en hardware. Las herramientas de implementación para apoyo al redireccionamiento, ayudan al diseñador para evitar la traducción manual de los sistemas o subsistemas. Existen herramientas CAD para el diseño heterogéneo y, reorientación de los sistemas

ISA se modela a niveles diferentes, estos proveen la información esencial sobre la arquitectura, apoyando el desarrollo de software y hardware. La organización de procesador se describe comúnmente en un lenguaje de descripción de hardware, hardware description languaje HDL para propósitos de síntesis, mientras que el modelo del procesador se usan frecuentemente para cosimulation.

2.3.1 Particionamiento del Hardware y Software

La partición de un sistema en hardware o software es la causa de la relación costo/desempeño del diseño final. Por lo tanto cualquier decisión de particionamiento, hecha por un diseñador o por una herramienta CAD debe tomar en cuenta las propiedades de los bloques resultantes de software y hardware.

La formulación del problema de particionamiento del hardware y software difiere según el problema de codiseño que se enfrenta. En el caso de sistemas dedicados, una partición del hardware o software representa una partición física de la funcionalidad del sistema de una aplicación de hardware y software que se ejecuta sobre uno o más procesador(es). Diversas fórmulas a este problema de partición pueden compararse en base la arquitectura propuesta, metas de partición y estrategia de solución.

Cuando consideramos sistemas de cómputo de propósito general, una partición representa una división lógica de la funcionalidad del sistema, donde el hardware se diseña para apoyar la implementación del software en la funcionalidad completa del sistema. Esta división es hecha por el conjunto de instrucciones, así la selección de instrucciones afecta fuertemente la organización del hardware y software del sistema.

Para sistemas que solo consisten de arreglos de FPGA la partición del sistema en componentes corresponden al desarrollo del tecnologing mapping. De otra manera para sistemas que consisten de FPGA y procesadores la partición involucra ambos tipos.

3 Reingeriería

Al conocer más el codiseño encontramos con este nuevo término, la reingeniería, surge para plantear nuevos paradigmas aplicables a un sin número de aplicaciones, aunque su desarrollo se ha extendido por la rama de la industria y la empresa, puede ser aplicable a otras áreas como software y hardware. A continuación daremos una definición formal de reingeniería.

Definición formal de "Reingenieria".

"Reingeniería es la *revisión fundamental* y el *codiseño radical* de procesos para alcanzar mejoras espectaculares en *medidas críticas* y contemporáneas del rendimiento, tales como costo, calidad, servicio y rapidez"

Dentro de esta definición se encuentran cuatro palabras que se distinguen, a continuación explicaremos su significado.

Revisión Fundamental.

Al encargado del mejoramiento del sistema, se debe hacer la pregunta más básica en su diseño, ¿Por qué hacemos lo que estamos haciendo? ¿Por qué lo hacemos de esa manera? Hacer estas preguntas nos obliga ha examinar las reglas tácticas y los supuestos en que descansa el manejo de sus sistemas, a menudo esas reglas resultan anticuadas, equivocadas o inapropiadas.

La reingeniería empieza sin ningún concepto previo, sin nada por sentado; en efecto, quien emprende la reingeniería debe cuidarse de los supuestos que la mayoría de los procesos ya han arraigado en ellas, la reingeniería determina primero qué debe hacer un sistema; luego cómo debe hacerlo. No da nada por sentado, se olvida por completo de lo que es y se concentra en lo que debe ser.

Codiseño Radical.

Rediseñar radicalmente significa llegar hasta la raíz de las cosas; no efectuar cambios superficiales ni tratar de arreglar lo que ya está instalado sino abandonar lo viejo. Al hablar de reingeniería, rediseñar radicalmente significa descartar todas las estructuras y los procedimientos existentes e inventar maneras enteramente nuevas de realizar el trabajo.

Medidas criticas.

La reingeniería no es cuestión de hacer mejoras marginales o incrementales sino de dar saltos gigantescos en rendimiento. Si un sistema se encuentra al 10% por debajo del desempeño que se supone tendría, este sistema no necesita reingeniería, con los métodos convencionales se puede lograr este fin.

Procesos.

Aunque es la más importante de las cuatro, también es la que les da más trabajo a los encargados del sistema, se puede pensar que los procesos implica dividir las tareas en cosas más simples y asignar a cada una de ellas un especialista.

4 BIBLIOGRAFÍA Y REFERENCIAS

[1] Núñez L. A, Matriz de conmutación de alta velocidad: Su verificación y su elemento de salida, Tesis de Cinvestav-IPN Gdl., 2000.

[2] Lopéz L. G, Interfaz entre enlaces primarios E1/T1 y una matriz de conmutación de alta velocidad, Tesis de Cinvestav-IPN Gdl, Noviembre 2000.

Centro de Investigación y de Estudios Avanzados del IPN

Unidad Guadalajara

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: MATRIZ DE CONMUTACIÓN DE ALTA VELOCIDAD: REDISEÑO DE ALGUNOS DE SUS ELEMENTOS del(a) C. Julio Cesar SILVA BRIANO el día 30 de Abril de 2002

Dr. Arturo Veloz Guerrero
Investigador Cinvestav 3A
CINVESTAV GDL
Guadalajara

Dr. Deni Librado Torres Román
Profesor Investigador 3A
CINVESTAV GDL
Guadalajara

Mc. Lino Evgueni Coria Mendoza
Encargado del área de electrónica industrial, Dpto. de Electrónica
Instituto Tecnologico de Estudios Superiores de Occidente
Guadalajara