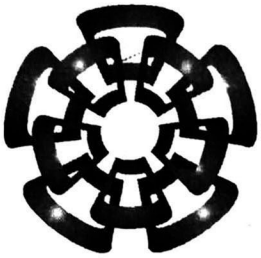
xx(86709.1)

CINVESTAV I.P.N.
SECCION DE INFORMACION
Y DOCUMENTACION



CINVESTAV - IPN

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

Interfaz entre enlaces E1/T1 y una matriz de conmutación ATM de alta velocidad

Tesis que presenta
Guillermo Alejandro López López

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal. Noviembre de 2000.

CINVESTAV
IPN
ADQUISICION
DE LIBROS

CLASIF.:
ADQUIS.: TESIS 2001
FECHA: 29-III-01
PROCED.: Depto. Servicios Bibliográficos
$:

Interfaz entre enlaces E1/T1 y una matriz de conmutación ATM de alta velocidad

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

por:

Guillermo Alejandro López López

Ingeniero en Electrónica
Instituto Tecnológico de La Laguna, 1992-1996

Becario del CONACYT, expediente no. 121161

Director de Tesis:
Dr. Deni Librado Torres Román

CINVESTAV del IPN Unidad Guadalajara, Noviembre de 2000.

A Dios, por darme la Vida y brindarme siempre Amor de una u otra forma.

A la memoria de mi madre que está siempre conmigo en todo momento.

Nunca se me olvida la anécdota de los "carneros" papá, gracias por ser mi mejor amigo, por apoyarme, por tus consejos y por estar siempre a mi lado.

A Angel, Meliza y Nancy que fueron mi principal motivación en mis estudios; espero haberles dado un buen ejemplo.

A toda mi familia aquí en Guadalajara, que siempre me han apoyado desde el primer día de mi llegada; siempre estaré en deuda con ustedes.

A Manuel, Carlos y Adolfo; gracias por todo su apoyo, valoro mucho su amistad.

A todos mis compañeros y profesores del CINVESTAV Gdl., especialmente al Dr. Deni (mi asesor de tesis), porque confió siempre en mí a pesar de todo.

A CONACYT por el apoyo económico que me brindó.

A ti Sara que sin tu apoyo, comprensión y Amor; no estaría escribiendo estas palabras. Gracias por darle sentido a mi Vida, Chiquita...

No tengo palabras para agradecer todo lo que ustedes han hecho por mí. Solo espero que Dios me dé la oportunidad de corresponder todo el apoyo y cariño que me han brindado.

Guillermo Alejandro López López

Que la lucha por la vida
No te asombre,
Que no es hombre
Quien luchar no sabe.

Porque el hombre
Nació para luchar,
Como nació
Para volar el ave.

-Amado Nervo-

CINVESTAV IPN, Gdl.

Índice

Índice de figuras

Índice de tablas

1 Introducción

El objetivo de los sistemas actuales de comunicación es el realizar la interconexión entre dos puntos para la transferencia de información entre estos, independientemente del sistema y protocolo de comunicación que maneje cada uno de ellos. Una buena solución a esto es *B-ISDN* (*Broadband Integrated Services Digital Network*, red digital de servicios integrados de banda ancha) [6][7]. Actualmente se hacen esfuerzos para contar con redes globales de servicios integrados; para esto, es necesario contar con sistemas de comunicación que hagan posible transferir la información de redes en donde el ancho de banda de información es pequeño, a redes en las cuales el ancho de banda es grande. Para lograr esto, son necesarios sistemas de comunicación (sistemas de interconexión) entre estas diferentes redes. Entonces, estos sistemas son capaces de transferir información entre una red de gran ancho de banda y una con menor ancho, pero también entre redes de gran ancho de banda.

Este sistema de comunicación es el *conmutador de alta velocidad* (*SW*) [1][2][5]. Este es un dispositivo que permite cursar tráfico de un lugar a otro a altas velocidades (Gigabits/s). Estos son empleados en redes de computadoras o en redes donde el ancho de banda de información es muy grande (servicios de vídeo digital, videoconferencia, multimedia, etc.).

En la figura 1-1 se muestra un esquema de comunicación en el que se interconectan todos estos tipos de redes entre si empleando *SWs*.

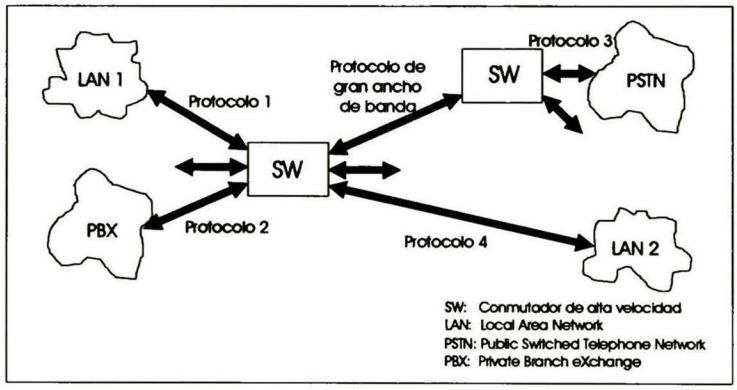


Figura 1-1 Esquema general de un sistema de comunicación

En esta figura se ilustra la conexión de *SWs* entre si, empleando el mismo protocolo de red de gran ancho de banda; esto es, un *SW* tiene la capacidad de conectar: diferentes redes que manejan ancho de banda reducidos (empleando diferentes protocolos), diferentes *SWs* entre sí empleando el mismo protocolo (gran ancho de banda) y diferentes redes con un ancho de banda limitado a un mismo protocolo de red de un amplio ancho de banda.

1.1. Planteamiento del problema

Un *SW* puede ser dividido en tres partes:

1. Los protocolos de comunicación de las diferentes redes[1] (amplio o reducido ancho de banda) de información que se desean conectar aunque estas cuenten con diferentes protocolos de comunicación.
2. La *matriz de conmutación* de la información (*MC*), que tiene la función de mover la información empaquetada de un puerto de entrada a un puerto de salida (ver detalles en [1][2][5]).
3. Los *circuitos de línea* (*CL*), que tienen la función de convertir de protocolos de redes de gran ancho de banda (protocolo empleado en la *MC*) a protocolos de redes de reducido ancho de banda y viceversa. También es posible que el *CL* haga la conexión entre diferentes protocolos de redes de gran ancho de banda (el empleado por la *MC* internamente y el de otra *MC* de otro *SW* o simplemente el de otra red de ancho de banda amplio). Ver figura 1-1.

Es necesario contar con un *CL* por cada protocolo específico de red de ancho de banda reducido que se quiera tener conectada al *SW*. En la figura 1-2 se muestran estas tres partes integrantes de un *SW*.

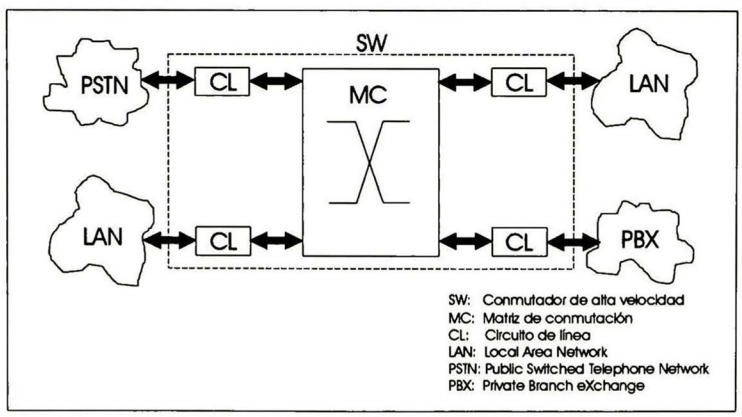


Figura 1-2 Esquema general de un conmutador de alta velocidad (*SW*)

El flujo de la información en este *SW* es explicado en los siguientes pasos:

1. La información de las diferentes redes como *LANs* (*Local Area Networks*, redes de área local), *PBX* (*Private Branch eXchange*, conmutadores privados), *PSTN* (*Public Switched Telephone Network*, red telefónica conmutada); es entregada a los *CLs* en el protocolo de

[1] El término *red* empleado en este documento no necesariamente describe redes de computadoras, también abarca otros tipos de redes (otros protocolos de comunicación).

la red de ancho de banda reducido a la que está conectado este *CL* (*Ethernet*, *Fast-Ethernet*, *E1*, *T1*, etc.).

2. Los circuitos de línea (*CLs*) se encargan de trasladar la información de estas redes (que viene en protocolo de red ancho de banda reducido) a los protocolos de red de gran ancho de banda, ya que los elementos de conmutación (*MC*) no manejan los protocolos de redes de ancho de banda reducido. Además, los *CLs* también proporcionan información adicional a los protocolos de redes de ancho de banda limitado (cuando esta información ya es convertida al protocolo de gran ancho de banda empleado por el *SW*) para que esta sea dirigida correctamente por medio de la *MC* a alguno de los puertos de salida de esta. Para esto, los *CLs* cuentan con una *tabla de enrutado* (*TR*), que consiste de una memoria que contiene el puerto de salida específico de la *MC* al cual será enviada la información de la red específica. Ver detalles sobre la *TR* en [4].
3. La *MC* recibe la información de la red de ancho de banda limitado pero ya en un protocolo de red de ancho de banda amplio como lo puede ser *ATM* (*Asynchronous Transfer Mode*, modo de transferencia asíncrono) [6][7]. A esta información el *CL* le agregó información que es empleada para el enrutado de esta dentro de la *MC*. Entonces, la *MC* solo revisa esta información adicional y basándose en ella, la *MC* enruta (dirige) la información de la red específica hacia un puerto de salida al cual está conectado otra red a la cual se le entregará esta información.
4. La información es entregada al *CL* que se encarga de cambiar del protocolo de red de gran ancho de banda en el que viene la información (*ATM*) al protocolo que emplea la red específica (de ancho de banda limitado) que recibirá la información.

En estos pasos anteriores es importante observar que:

- Los *CLs* tiene la capacidad de conversión de protocolos en ambas direcciones. Esto es, del protocolo de red de gran ancho de banda empleado por el *SW*, al protocolo que emplea la red específica (de reducido ancho de banda) y viceversa.

  Con este esquema de comunicación se envía información entre los "mismos" protocolos de redes de ancho de banda reducido, pero también es posible la transferencia de información entre protocolos de redes "diferentes"

En el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV) Unidad Guadalajara, se cuenta con una *MC* que fue diseñada con propósitos académicos y de investigación. Esta emplea como protocolo de red global a *ATM*, pero la celda (paquete de información) no es una celda *ATM* "pura" ya que el *CL* pone información adicional de enrutado en esta para la transferencia de la información dentro de la *MC*.

En el protocolo *ATM*, se tienen diferentes tipos de servicios para el flujo de la información [6][7]. Esta es organizada en paquetes de tamaño fijo llamados celdas. Cada celda tiene 53 bytes de los cuales 48 son de información del usuario y 5 son de encabezado, que es utilizado para el control del flujo de esta celda en la red.

La *MC* diseñada necesita 3 bytes aparte de los 53 bytes de una celda *ATM* "pura", para el enrutado de la información dentro de esta. De estos 3 bytes, los dos primeros son empleados para el enrutado de la información dentro de la *MC* y el tercero (que no contiene información de enrutado) no es empleado por la *MC*, ya que esta reservado para uso de los *CLs*. Para mayor información sobre los 3 bytes de enrutado adicionales a la celda *ATM* "pura", ver detalles en [1][2][4].

Entonces, surge la necesidad de construcción de un *CL* para la interconexión de la *MC* y diferentes enlaces *PCM* (*Pulse Code Modulation*, modulación por pulsos codificados) primarios *E1* (un flujo de información de 2.048 Mbits por segundo, dividido en 32 canales *PCM* de 64 Kbits por segundo cada uno) y *T1* (un flujo de información de 1.544 Mbits por segundo, dividido en 24 canales *PCM* de 64 Kbits por segundo cada uno). Para mayor detalle sobre *E1* y *T1* ver [15][16].

1.2. Objetivos de la tesis

Los objetivo de esta tesis son:

- La especificación y diseño de la arquitectura de una "Interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad" (*IE1-T1/MC*).
- Esta interfaz (*CL*) deberá ser capaz de interconectarse con diferentes enlaces *E1/T1*.
- Este *CL* deberá tener una conexión hacia un administrador del sistema para funciones de control y monitoreo.
- La simulación de los bloques diseñados que formarán parte del *CL*, así como las pruebas de estos.

1.3. Organización de este documento de tesis

Este documento está dividido en 6 capítulos en donde se describe la propuesta de solución, así como el diseño de algunos de los componentes de funciones específicas que no se encuentran en el mercado.

La organización de los capítulos de este documento es como sigue:

El segundo capítulo describe a "groso modo" el esquema propuesto de solución. Aquí se mencionan los componentes que forman a la *IE1-T1/MC* y el porque fueron elegidos.

- El tercer capítulo muestra a detalle el diseño de la interfaz (*IT/UL2*) entre el protocolo propio de red empleado por la *MC* (*TOPIA*) y el protocolo de comunicación *UTOPIA Level 2* (en el capítulo 2 se describe el porque se emplea este protocolo físico de *ATM*).

Aquí se muestra este diseño hasta un determinado nivel de abstracción, ya que si se llegara al nivel de abstracción más bajo (código *VHDL; Very high speed integrated circuits Hardware Description Lenguaje*, lenguaje de descripción de hardware), se tendría un documento muy voluminoso que no sería fácil ni práctico de leer. El nivel de abstracción empleado en este capítulo es la descripción a bloques del diseño; describiendo la función de cada bloque, su interacción con los otros y las señales empleadas por éste para comunicarse con el exterior.

El cuarto capítulo muestra a detalle el diseño de la interfaz de 8 a 16 bits y viceversa (*I8/16b*). Para la descripción del diseño de esta interfaz se emplea el mismo nivel de abstracción empleado para la descripción de la *IT/UL2* (capítulo 3).

El quinto capítulo describe una arquitectura diseñada con el objetivo de probar el diseño de la *IT/UL2*. Se describe el diseño de esta arquitectura (*cama de prueba*) con el mismo nivel de abstracción empleado en los capítulos 3 y 4. Además, en este capítulo se dan algunos resultados obtenidos de la implementación de los diseños *IT/UL2*, *I8/16b* y de la *cama de prueba* incluida a la *IT/UL2*.

En el sexto capítulo se mencionan las conclusiones del desarrollo de esta tesis, así como el trabajo futuro propuesto.

Finalmente, se tienen 4 apéndices. En el apéndice A se describe a detalle (con un nivel de abstracción mayor al empleado en los capítulos 3, 4 y 5) la implementación en código *VHDL* de uno de los bloques del la *IT/UL2*. En el apéndice B se ilustran los esquemáticos de la *IE1-T1/MC*. En el apéndice C se muestran algunas simulaciones temporales de las señales empleadas en los diseños *IT/UL2*, *I8/16b y la cama de prueba*; realizadas con la herramienta *Max+plus II* de la compañía *ALTERA*. Y finalmente en el Apéndice D se tiene un resumen de *ATM* describiendo las capas que forman el modelo de referencia de *ATM* así como los tipos de servicio que se manejan.

1.4. Terminología

Las siguientes abreviaturas serán utilizadas a lo largo de este documento.

Abreviatura	Significado
Ana_UL1	Bloque analizador de celdas *ATM* en protocolo *UTOPIA Level 1.*
ATM	Modo de transferencia asíncrona (*Asynchronous Transfer Mode*).
BIST	Prueba construida internamente (*Built In-Self Test*).
CAP	Bloque de control automático de pruebas.
CL	Circuito de línea.
FPGA	Dispositivo de arreglo de compuertas programables (*Field Programmable Gate Array*).
FSM	Máquina de estados finitos (*Finite State Machine*).
Gen_UL1	Bloque generador de celdas *ATM* en protocolo *UTOPIA Level 1.*
IE1-T1/MC	Interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad.

ITR	Interfaz de la tabla de enrutado.
IT/UL2	Interfaz entre el protocolo de comunicación *TOPIA* y el protocolo *UTOPIA Level 2.*
I8/16b	Interfaz entre un protocolo de comunicación de 8 bits y otro de 16 bits.
I_Rx	Interfaz de recepción del la *IT/UL2.*
I_Tx	Interfaz de transmisión de la *IT/UL2.*
IWE	Elemento de interfaz entre el protocolo *UTOPIA Level 2* y el protocolo empleado por los *framers* empleados en el *CL* (*InterWorking Element*).
LBT	Loopback por el lado del protocolo *TOPIA.*
LBU	Loopback por el lado del protocolo *UTOPIA Level 2*
LB1,LB2,LB3	Pruebas hechas a diferentes niveles de un diseño.
MC	Matriz de conmutación *ATM* de alta velocidad.
PC	Computadora personal (*Personal Computer*).
PCB	Tarjeta de circuito impreso (*Printed Circuit Board*).
PCM	Modulación por pulsos codificados (*Pulse Code Modulation*).
R1a,R2a,R1b,R2b	Bytes de enrutado de la celda normal y especial de 56 bytes del protocolo *TOPIA.*
SW	Sistema de comunicación formado por diferentes protocolos de red, una *MC* y diferentes *CL* (*Switch*).
SAR	Circuito de segmentación y ensamble de celdas *ATM* (*Segmentation and Reassembling*).
TOPIA	Protocolo propio del *switch ATM* de alta velocidad.
TR	Tabla de enrutado.
UTOPIA	Interfaz física de prueba y operación de *ATM* (*Universal Test and Operation Physical Interface of ATM*).
VCI	Identificador del camino virtual empleado en el encabezado de la celda *ATM* (*Virtual Channel Identifier*).
VHDL	Lenguaje de descripción de hardware para circuitos integrados de muy alta velocidad *(Very high speed integrated circuits Hardware Description Lenguaje).*
VPI	Identificador de la trayectoria virtual empleado en el encabezado de la celda *ATM* (*Virtual Path Identifier*).

2 Esquema general de solución

El objetivo de esta tesis es el diseño de un circuito de línea (interfaz) entre diferentes enlaces *E1* o *T1* y una matriz de conmutación *ATM* de alta velocidad. Además, este circuito de línea debe contar con comunicación hacia un administrador del sistema para sus funciones de control y monitoreo. Por el lado de la matriz de conmutación *ATM* de alta velocidad (*MC*) se cuenta con el protocolo propio de esta que es *TOPIA*[2] (ver sección 3.2 de este documento). El flujo de información que puede manejar cada puerto de la *MC* es de 155.52Mbits/s [1][2], así es que si se toma en cuenta que se manejan solo flujos primarios *E1* que son de 2.048 Mbits/s [16], el ancho de banda de cada puerto de la *MC* podría llevar hasta 68 diferentes enlaces *E1*, en caso de que la *MC* empleara celdas *ATM* "puras" (53 bytes de información: 5 bytes de encabezado y 48 bytes de carga útil). Pero al convertir los enlaces primarios *E1* o *T1* al protocolo empleado por la *MC* (*ATM* "puro" más 3 bytes adicionales de enrutado) el número máximo de estos enlaces *E1* que se pueden transportar por este flujo de 155.52 Mbits/s baja ya que además de la información (carga útil) de estos enlaces primarios se agregan 5 bytes de encabezado de la celda *ATM* [6][7] y 3 bytes de enrutado. Entonces, el circuito de línea podría manejar hasta 64 posibles enlaces primarios *E1* para un flujo de uno de los puerto de la *MC*. En la figura 2-1 se muestra la interfaz como una caja negra con las entradas y salidas requeridas.

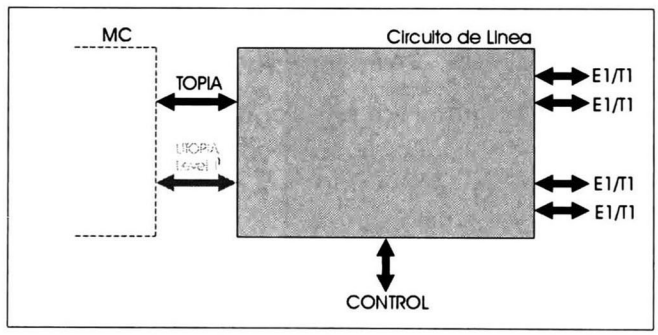


Figura 2-1 Esquema de caja negra del circuito de línea

Para los fines de esta interfaz (circuito de línea) es necesario un circuito que construya celdas *ATM* de enlaces primarios (*E1* o *T1*); esto es, un circuito que genere celdas *ATM* con la información contenida en estos enlaces y además genere el encabezado de estas celdas. Y en la otra dirección, el mismo circuito debe de generar los enlaces primarios de las celdas provenientes de la *MC*, generando los canales especiales de sincronía y señalización de estos enlaces [16].

[2] Inicialmente la *MC* fue diseñada e implementada con el protocolo *TOPIA* para la comunicación con los *CLs*. En [5] se diseño una interfaz entre *TOPIA* y *UTOPIA Level 1* [13] de tal forma que la *MC* contará con estas dos opciones de comunicación hacia los *CLs*. La interfaz entre la *MC* y este *CL* (diseño de esta tesis) es mediante el protocolo *TOPIA*, ya que cuando se inició el diseño la *MC* no contaba con la opción de *UTOPIA Level 1*.

En el mercado se encuentran circuitos que fabrican diferentes compañías llamados *SAR* (Segmentation & Reassembling, segmentación y ensamble de celdas *ATM*). Estos circuitos se encargan de construir celdas *ATM* de información que proviene de un protocolo especifico, este generalmente es propio del fabricante de este circuito. A la vez, este también extrae la información de las celdas *ATM* poniéndola en el protocolo empleado por este circuito.

El circuito *SAR* que se empleó en el diseño de esta interfaz fue el *PXB-4220* del fabricante *SIEMENS* (para detalles ver [22]). Este circuito *SAR* tiene la capacidad de manejar hasta 8 posibles enlaces primarios *E1* o *T1*, empleando para esto circuitos denominados "*framer*".

Estos circuitos "*framer*" son los encargados de la conversión del protocolo del enlace primario al protocolo específico que emplea el circuito *SAR*. En este diseño se emplean dos de estos circuitos que tienen la capacidad cada uno de ellos de manejar 4 enlaces primarios, formando así los 8 enlaces primarios que máximo puede manejar el circuito *SAR*. El circuito *framer* empleado en el diseño es el *PEB 22554* que es también del fabricante *SIEMENS*. Este circuito además de las características mencionadas anteriormente, también cuenta internamente con una interfaz de línea (*LIU*), esta interfaz es la encargada de ajustar los distintos niveles de voltaje empleados por estos enlaces primarios [15]. En la figura 2-2 se muestran estos circuitos, que son la base del diseño del circuito de línea.

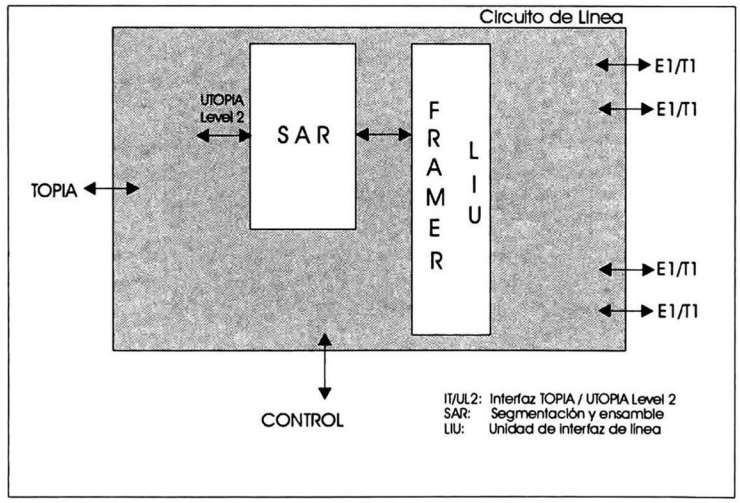


Figura 2-2 Circuitos *SAR*, *Framer* y *LIU* del circuito de línea

Se observa en la figura que el circuito *SAR* emplea el protocolo *UTOPIA Level 2* [14]. Este es un protocolo de capa fisica empleado en *ATM* para la conexión de diferentes circuitos que manejan *ATM*. Existe también el protocolo *UTOPIA Level 1* [13] y la diferencia de este con el manejado por el circuito *SAR*, es que el *Level 1* solo puede hacer una conexión uno a uno, mientras que el *Level 2* puede hacer una conexión de hasta 31 (capas *PHY*) a uno (capa *ATM*)[3]. En nuestro caso, el circuito de línea podría manejar hasta 8 circuitos *SAR*, cada uno de ellos con sus respectivos *framers-LIU*; esto nos daría la posibilidad de emplear hasta 64 enlaces *E1* o *T1*. Por razones practicas de espacio y tomando en cuenta que el diseño de esta interfaz es un

[3] En el Apéndice D se describe cada una de las capas del modelo de referencia de *ATM*.

prototipo, el diseño se hizo con la intención de solo manejar un solo circuito *SAR*; esto es, manejar hasta un máximo de 8 enlaces primarios *E1* o *T1*.

Por el lado de los enlaces primarios todavía hacen falta circuitos de protección contra transitorios y descargas que podrían existir en los enlaces primarios y que dañarían a la interfaz si son conectados directamente al circuito *framer-LIU*.

Por el lado de la *MC* el circuito *SAR* entrega y recibe celdas *ATM* "puras" (53 bytes) en el protocolo *UTOPIA Level 2* mientras que la *MC* emplea su propio protocolo de comunicación (*TOPIA*), además de manejar celdas de 56 bytes (53 bytes de la celda *ATM* "pura" y 3 bytes adicionales de enrutado). Entonces, es necesario contar con una interfaz entre el protocolo *TOPIA* y *UTOPIA Level 2*. Esta interfaz (*IT/UL2*) no es posible de conseguir en el mercado ya que el protocolo *TOPIA* no es un protocolo estandarizado, por lo que es necesario el diseño e implementación de esta interfaz (ver detalles de esta interfaz en el capitulo 3 de este documento).

Los circuitos *SAR*, *framer-LIU* y la *IT/UL2* necesitan ser inicializados para realizar su función dentro del circuito de línea (*IE1-T1/MC*), además estos circuitos pueden ser monitoreados y así reportar el estado de la *IE1-T1/MC* (como puede ser cualquier alarma que ocurra en alguno de los enlaces primarios). Para estas funciones surge la necesidad de un administrador del sistema, el cual tendrá la posibilidad de inicializar la interfaz, así como el monitorear el estado actual de esta.

Se toma en cuenta que este administrador contará con una interfaz en una computadora personal (*PC*) para realizar todas estas funciones. Por tanto, se escogió un microcontrolador de 8 bits (ver detalles en capitulo 4) para la comunicación de la *PC* (por medio del protocolo *Rs-232*) con los diferentes circuitos de la *IE1-T1/MC*.

Los diferentes circuitos de la *IE1-T1/MC* cuentan con una interfaz de bus *Intel* de 16 bits para esta comunicación (inicialización, control y monitoreo) y por otro lado se cuenta con un microcontrolador de 8 bits. Entonces, es necesario una interfaz de 8 a 16 bits y viceversa (*I8/16b*).

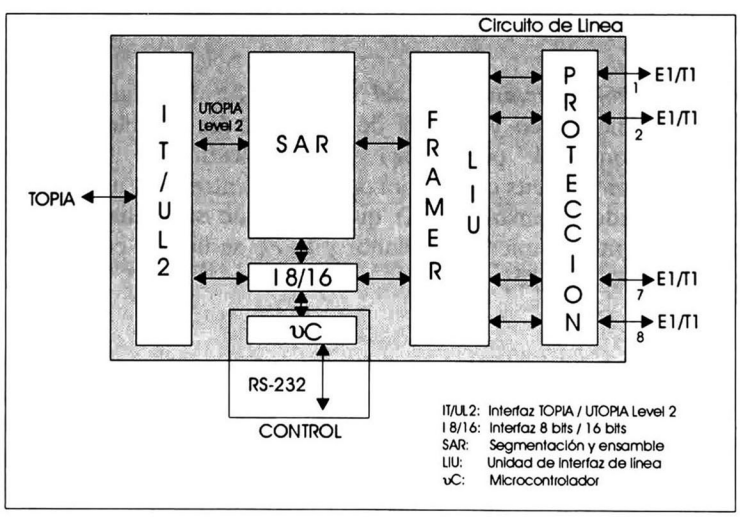


Figura 2-3 Esquema general de solución

Finalmente, la arquitectura propuesta (seleccionada) de la *IE1-T1/MC* (*CL*) consiste de los siguientes 6 bloques (ver figura 2-3):

1. ***Bloque de protección***: Este bloque contiene los circuitos de protección de sobrecargas é inmunidad a transitorios en los enlaces primarios *E1* o *T1*.

2. ***Bloque de interfaz de línea y framer***: La función de la interfaz de línea es adecuar las señales empleadas entre los enlaces primarios *E1* o *T1* y el circuito *framer*, recobrando datos y señales de reloj del enlace troncal primario [15][16]. La función del *framer* es la alineación y síntesis de la trama del enlace primario, así como el monitoreo del *performance* y alarmas del enlace [23].

3. ***Bloque de segmentación y ensamble de celdas ATM***: Este bloque es el encargado de formar la celda *ATM* con la información proveniente del *framer* y enviarla a la *IT/UL2* en el protocolo *UTOPIA Level 2* [14]. A su vez, este bloque recibe las celdas *ATM* en el protocolo *UTOPIA Level 2* y convierte esta información al protocolo que maneja el circuito *framer*. Además, este bloque tiene comunicación con el bloque de control permitiendo así funciones de inicialización y monitoreo.

4. ***Bloque de interfaz entre protocolos UTOPIA level 2 y TOPIA***: Este bloque es el encargado de transferir la información que viene en el protocolo *UTOPIA level 2* al protocolo propio de la *MC* (protocolo *TOPIA* [1][2]) y viceversa. Este bloque recibe del bloque de control la orden del modo en el que debe de funcionar.

5. ***Bloque de interfaz de 16 bits / 8 bits***: La función de este bloque es la de conversión del protocolo utilizado por el control que es de 8 bits de datos al protocolo propio de 16 bits de datos empleado por los bloques *SAR, FRAMER/LIU* y *IT/UL2*.

6. ***Bloque de control (administrador del sistema)***: La función de este bloque es la inicialización, monitoreo y control de cada uno de los bloques de la *IE1-T1/MC* (excepto el bloque de protección). Para realizar esta función se emplea un microcontrolador de 8 bits que servirá de interfaz entre cada uno de los bloques del *CL* y una computadora personal (*PC*) que es donde se realizan estas funciones. La comunicación entre el microcontrolador y la *PC* se lleva a cabo mediante la interfaz serie asíncrona *RS-232* [25].

3 Interfaz *TOPIA / UTOPIA Level 2*

En el esquema de la figura 3-1, está enfatizada la interfaz entre protocolos de comunicación *TOPIA* y *UTOPIA Level 2* (*IT/UL2*), dentro de la interfaz entre enlaces *E1/T1* y un matriz de conmutación *ATM* de alta velocidad (*IE1-T1/MC*). Por un lado, esta interfaz (*IT/UL2*) es capaz de recibir información en el protocolo de comunicación *UTOPIA Level 2* [14] desde el circuito que ensambla y desensambla celdas *ATM* (circuito *SAR*) [18] y mandar esta misma información en el protocolo propio de comunicación de la *MC* (*TOPIA*) [1][2][5], con información adicional de enrutado; ayudando así a la matriz de conmutación *ATM* de alta velocidad (*MC*) a dirigir esta información a su puerto destino. Por otro lado, esta interfaz será capaz de recibir la información de la *MC* con el protocolo de comunicación *TOPIA*, mandando esta misma al circuito *SAR* con el protocolo de comunicación *UTOPIA Level 2.*

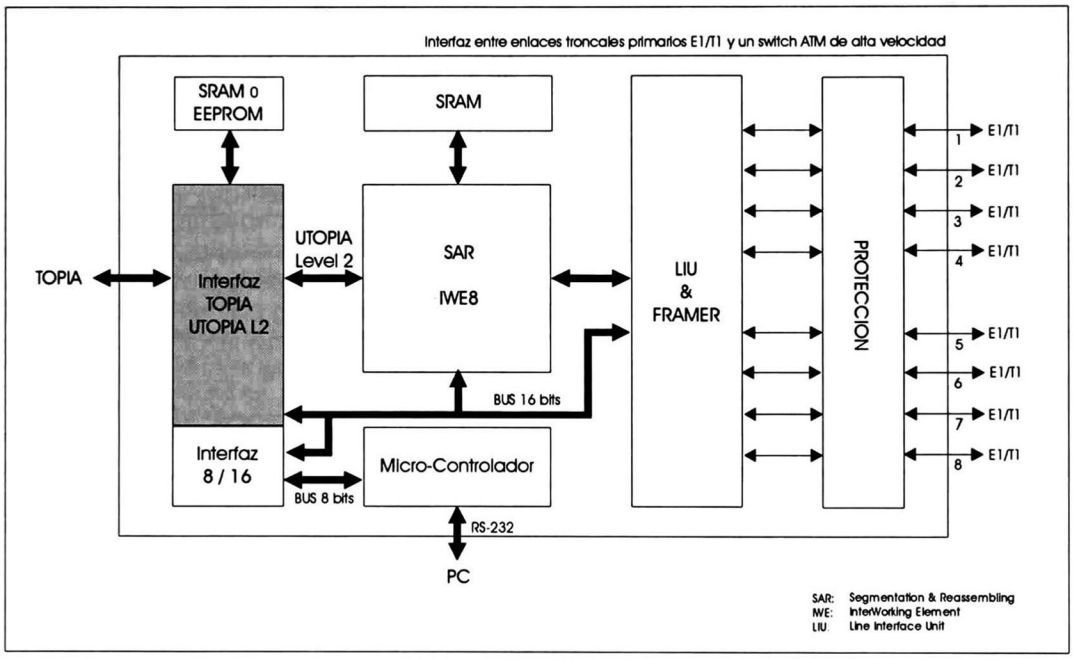


Figura 3-1 Interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad

Para hacer posible el cambio de protocolo de *UTOPIA Level 2* a *TOPIA*, es necesario agregar información adicional de enrutado. Esta información es tomada de una memoria (tabla de enrutado [4]) que está dentro de esta interfaz; de tal manera que, esta también cuenta con las funciones necesarias para la actualización de la tabla de enrutado (*TR*). Esta tabla es considerada de cierto modo como una *TR "estática"*, ya que la actualización de esta se realiza cuando el administrador de este sistema de comunicación así lo desea. Esta característica es debida a que se está conectando la información de enlaces *E1/T1* a la *MC* y se considera que

en los canales de estos enlaces no viene incluida la información para la actualización de la *TR* [15][16].

3.1. Descripción funcional de la interfaz *TOPIA / UTOPIA Level 2*

La *IT/UL2* está constituida por dos áreas principales que son: la interfaz de recepción (*I_Rx*), que es la encargada de hacer el cambio de protocolo de comunicación *UTOPIA Level 2* al protocolo de comunicación *TOPIA*; y la interfaz de transmisión (*I_Tx*), que es la encargada de hacer el cambio del protocolo de comunicación *TOPIA* al protocolo de comunicación *UTOPIA Level 2*. En la *IT/UL2* se cuenta con una memoria que puede ser *SRAM* o *EEPROM* [26], para el almacenamiento de la tabla de enrutado (*TR*). Como sólo se cuenta con una *TR* y con dos áreas en el diseño que tienen acceso a esta memoria, es necesario un bloque de control para decidir cual de las dos áreas tendrá el acceso a la memoria, ya que puede haber algún conflicto si ambas áreas quisieran tener acceso a la *TR* al mismo tiempo. Este bloque de control se encuentra incluido en el bloque de interfaz de la tabla de enrutado (*ITR*). Además, la interfaz cuenta con dos bloques adicionales: *loopback*[4] del lado del protocolo *UTOPIA Level 2* (*LBU*) y *loopback* del lado del protocolo *TOPIA* (*LBT*). La función de estos bloques es la de prueba de esta interfaz. Ver secciones 3-8 y 3-9.

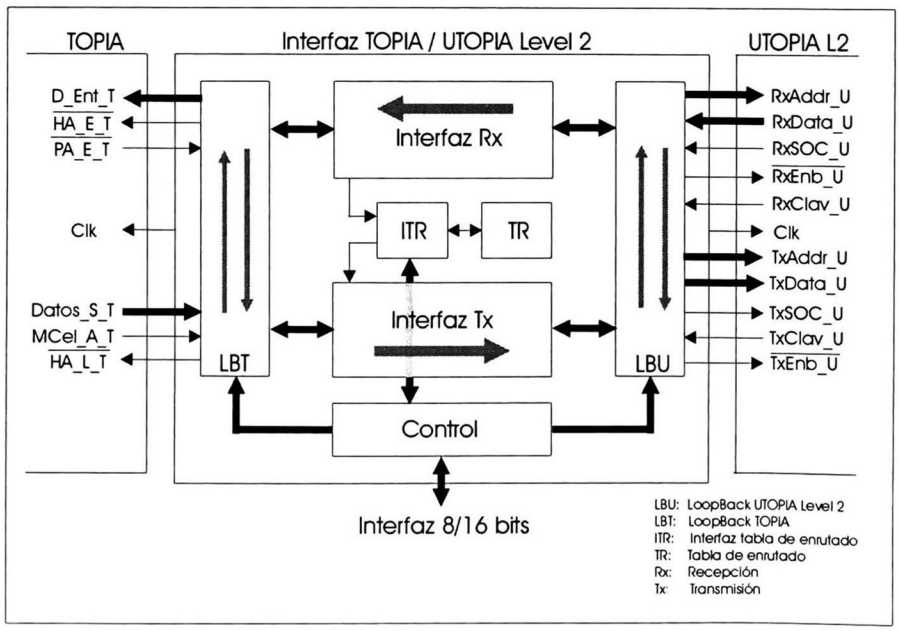


Figura 3-2 Interfaz *TOPIA/UTOPIA Level 2*

[4] En este documento es empleada la palabra inglesa "*loopback*" cuando se habla de un "lazo de retroalimentación", con el fin de reducir espacio.

La *IT/UL2* tiene dos modos de operación: el primero es el modo de trabajo normal, esto es, la conversión de protocolos *TOPIA* a *UTOPIA Level 2* y viceversa; el segundo es el de pruebas, el cual es especial ya que comprueba el correcto funcionamiento de la *I_Rx*, *I_Tx* y *TR*.

Finalmente, la *IT/UL2* cuenta con un bloque de control que es el encargado de poner en alguno de los modos mencionados a esta interfaz. Este bloque recibe la información del modo de trabajo mediante la interfaz 8/16 bits (*I8/16b*). La *I8/16b* es vista a detalle en el capítulo 4 de este documento. En la figura 3-2, se muestra el diagrama principal de la *IT/UL2* así como las señales que interactúan con ella, tanto del protocolo *UTOPIA Level 2* como *TOPIA*.

La *I_Rx* recibe una celda *ATM* (5 bytes de encabezado y 48 bytes de carga útil) [6][7], de la cual extrae la información de enrutado del protocolo de comunicación *ATM*; que son los identificadores del camino virtual y del canal virtual (*VPI* y *VCI*), ambos del encabezado de la celda. Con esta información, la *I_Rx* se encarga de encontrar una dirección de enrutado (que está localizada en la *TR*) para que esta información sea dirigida a algún puerto de salida de la *MC*. Esto es, la *MC* a la cual está conectada esta interfaz necesita 2 bytes de información para enrutar esta celda y así hacerla llegar al puerto correspondiente. Ver mayores detalles en [1][2][4][5].

La *I_Tx* recibe una celda en formato *TOPIA*, esto es, 56 bytes de información. Esta celda recibida puede ser información para convertirse al protocolo *UTOPIA Level 2* o también puede ser información para actualizar la *TR*. La *I_Tx* debe distinguir entre estas dos opciones y hacer lo necesario para realizar estas funciones. En caso de que se tenga que convertir al formato *UTOPIA Level 2*, la *I_Tx* debe quitar los tres primeros bytes de la celda de 56 bytes, ya que estos únicamente sirvieron para enrutar la información dentro de la *MC*. En caso de que la celda sea para la actualización de la *TR*, la información contenida en esta celda de 56 bytes solo actualizará una sola posición de esta tabla; esto es, un solo dato (de dos bytes) para un especifico *VPI* y *VCI*. En la figura 3-3, se muestra el formato de la celda que recibe la *I_Tx* de la *MC*.

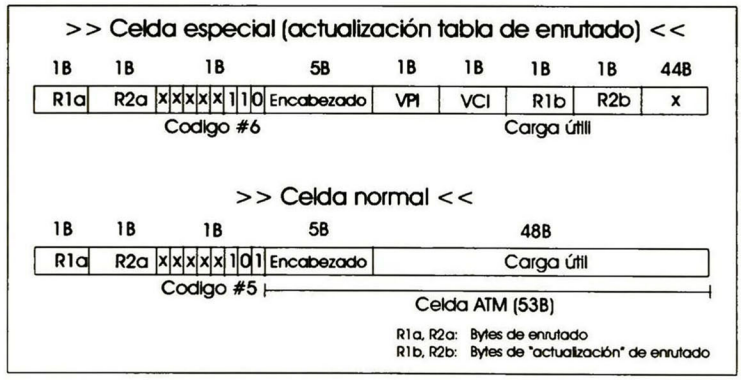


Figura 3-3 Formato de celda *TOPIA* (56 bytes)

3.2. Protocolo de comunicación *TOPIA*

Por el lado de la *MC*, la *IT/UL2* se comunica con esta por medio del protocolo de comunicación *TOPIA*, que es el medio de comunicación empleado para la conexión de la *MC* con los diferentes *CLs*. La diferencia principal entre este y *ATM* "puro", es que *TOPIA* añade 3 bytes a los 53 bytes de *ATM*. En estos 3 bytes se tiene información adicional, que le sirve a la *MC* para el enrutado de la información dentro de esta.

3.2.1. Señales del protocolo de comunicación *TOPIA*

En la tabla 3-1, se describen las señales empleadas en el protocolo de comunicación *TOPIA*, así como el origen, destino y descripción de cada una de estas señales.

Señal	Origen	Destino	Descripción
D_Ent_T[7-0]	*ITUL2*	*MC*	Consiste de un bus de 8 líneas de datos de entrada al protocolo *TOPIA* proveniente la *IT/UL2*.
HA_E_T	*ITUL2*	*MC*	Señal activa en bajo. Sirve para iniciar la transferencia de un ciclo de celda de la *IT/UL2* hacia el protocolo *TOPIA*.
PA_E_T	*MC*	*ITUL2*	Señal activa en bajo. Sirve para detener la transmisión de la información de la *IT/UL2* hacia el protocolo *TOPIA*.
Clk	*ITUL2*	*MC*	Reloj provisto por la *IT/UL2* para la sincronización de la transferencia de los datos. La frecuencia máxima de esta señal es de 25 *MHz*. El trabajo es con flancos positivos.
Datos_S_T[7-0]	*MC*	*ITUL2*	Consiste de un bus de 8 líneas de salida del protocolo de comunicación *TOPIA* hacia la *I_Tx* contenida en la *IT/UL2*.
MCel_A_T	*MC*	*ITUL2*	Señal activa en alto. Sirve para indicarle a la interfaz *IT/UL2* los instantes cuando hay una microcelda [1][2][5] (8 bytes de información) disponible para ser leída por esta interfaz.
HA_L_T	*ITUL2*	*MC*	Señal activa en bajo. Señal que sirve para iniciar la transferencia de una celda hacia la *IT/UL2*.

Tabla 3-1 Señales del protocolo de comunicación *TOPIA*

3.2.2. Operación y temporizado del protocolo de comunicación *TOPIA*

3.2.2.1. Flujo de celdas de la interfaz TOPIA/UTOPIA Level 2 *hacia la matriz de conmutación* ATM *de alta velocidad*

La *IT/UL2* debe avisar el instante en que se va a iniciar la transferencia de una celda mediante la señal *HA_E_T* debiéndose cumplir el siguiente procedimiento:

1. Es necesario que la señal *PA_E_T* que provee la *MC* se encuentre en estado alto para el inicio de la transferencia de celdas. Esta señal es adecuada en situaciones en las cuales la *MC* no pueda aceptar celdas (periodo de inicialización y/o diagnóstico del sistema).

2. Antes del comienzo de la transferencia de una celda, la señal *HA_E_T* debe permanecer en estado alto al menos un ciclo de reloj *Clk*, de tal forma que la *MC* se cerciore de que es el inicio de una transferencia. El pulso positivo de la señal *HA_E_T* puede traslaparse con la captura del último byte del ciclo de celda actual, logrando así transferir una celda detrás de otra.

3. La transferencia de la celda se iniciará con la transición negativa de la señal *HA_E_T*, antes del siguiente flanco activo de reloj. Junto a estas dos transiciones deberá estar presente el primer byte de datos de la celda, de tal manera que la *MC* pueda capturar este dato en el flanco activo del reloj (flanco positivo).

En la figura 3-4, se muestra el diagrama de tiempos de las señales que interactúan en el flujo de celdas de la *IT/UL2* hacia la *MC*.

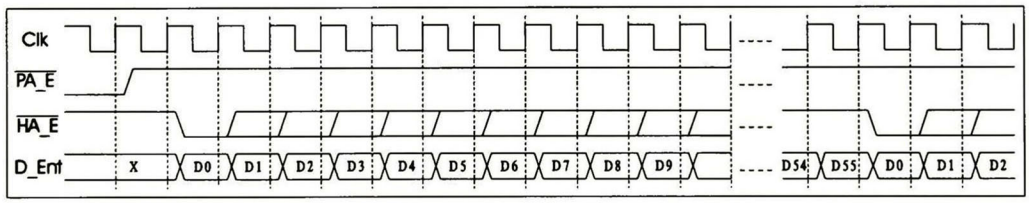


Figura 3-4 Diagrama de tiempos del flujo de celdas de la interfaz *TOPIA/UTOPIA Level 2* hacia la matriz de conmutación *ATM* de alta velocidad

3.2.2.2. *Flujo de celdas de la matriz de conmutación* ATM *de alta velocidad hacia la interfaz* TOPIA/UTOPIA Level 2

En este sentido del flujo de celdas, se dispone de la señal *Mcel_A_T* para que la *IT/UL2* pueda dosificar el flujo efectivo de celdas que los puertos de salida de la *MC* producen. Se debe cumplir el siguiente procedimiento:

1. La señal *MCel_A* le indica a la *IT/UL2* que hay una microcelda (8 bytes de información) disponible en la cola de salida de ese puerto al cual está conectada la *IT/UL2* y que en el momento en que esta interfaz lo desee pueda extraerla poniendo en un estado bajo a la señal *HA_L_T*. Después que la señal *HA_L_T* se activa (se pone en estado bajo), la *MC* puede entregar la microcelda a la *IT/UL2*. La señal *HA_L_T* es el medio por el cual la *IT/UL2* regula la emisión de las microceldas del puerto de la *MC* al cual está conectada esta interfaz.

2. Para comenzar la transferencia de una celda, la *IT/UL2* debe generar un pulso negativo en la línea *HA_L_T*, lo suficientemente ancho (mínimo dos ciclos de reloj) para que este estado bajo se mantenga hasta el flanco activo más próximo de *Clk*; con lo que se inicia una transferencia.

Es responsabilidad de la *IT/UL2* el llevar la cuenta de los bytes extraídos y del monitoreo continuo de la señal *Mcel_A_T*.

En la figura 3-5, se muestra el diagrama de tiempos de las señales que interactúan en el flujo de celdas de la *MC* hacia la *IT/UL2*.

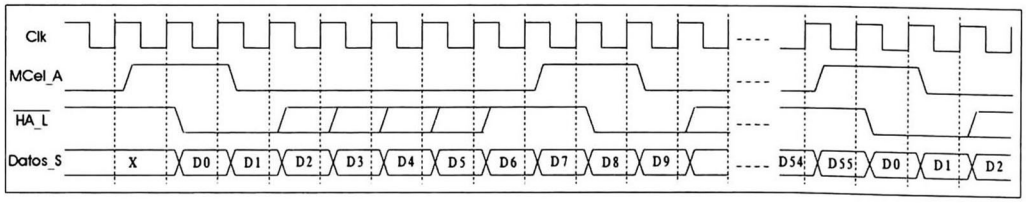


Figura 3-5 Diagrama de tiempos del flujo de celdas de la matriz de conmutación *ATM* de alta velocidad hacia la interfaz *TOPIA/UTOPIA Level 2*

3.3. Protocolo de comunicación *UTOPIA Level 2*

La *IT/UL2* por el lado de *UTOPIA Level 2*, tiene la capacidad de conectarse con múltiples circuitos. Estos circuitos son las capas *PHY* (fisicas), siendo la capa *ATM*[5] la *IT/UL2*. Por razones de implementación fisica de la *IE1-T1/MC* (circuito de línea), esta interfaz solo cuenta con una sola capa *PHY* que es el circuito *SAR* [22].

En las secciones siguientes se describen las señales empleadas del protocolo *UTOPIA Level 2*, así como un procedimiento conciso de las señales empleadas en este protocolo. Para mayor detalle sobre el protocolo *UTOPIA Level 2*, véase [13][14].

3.3.1. Señales del protocolo *UTOPIA Level 2*

En la tabla 3-2, se describen las señales empleadas en el protocolo de comunicación *UTOPIA Level 2*; así como el origen, destino y descripción de cada una de estas señales.

Señal	Origen	Destino	Descripción
TxAddr_U[4-0]	*ITUL2*	*SAR*	Consiste en un bus de 5 líneas de dirección de salida de la *IT/UL2* hacia la capa *PHY* (circuito SAR). Esta señal es utilizada para seleccionar hasta 31 posibles capas *PHY* (el circuito *SAR* en nuestro caso). El valor de "*1Fh*" en este bus, indica una capa *PHY* nula.
TxData_U[7-0]	*ITUL2*	*SAR*	Consiste de un bus de 8 líneas de datos de salida de la *IT/UL2* hacia la capa *PHY* (circuito SAR).
TxSOC_U	*ITUL2*	*SAR*	Señal activa en alto cuando *TxData_U* contiene el primer byte válido de la celda.
TxEnb_U	*ITUL2*	*SAR*	Señal activa en bajo cuando *TxData_U* contiene datos válidos de la celda.
TxClav_U	*SAR*	*ITUL2*	Señal tri-estado activa en alto, que se utiliza para indicar que se acepta la transferencia de una celda completa hacia cualquiera de las 31 posibles capas *PHY*.
Clk	*ITUL2*	*SAR*	Reloj provisto por la *IT/UL2* para la sincronización de la transferencia de los datos. La frecuencia máxima de esta señal es de 25 *MHz*. El trabajo es con flancos positivos.
RxAddr_U[4-0]	*ITUL2*	*SAR*	Consiste en un bus de 5 líneas de dirección de salida de la *IT/UL2* hacia la capa *PHY* (circuito SAR). Esta señal es utilizada para seleccionar hasta 31 posibles capas *PHY*. El valor de "*1Fh*" en este bus, indica una capa *PHY* nula.

[5] La capa *PHY* y la *ATM* están definidas por el *ATM Forum* en el estándar de *UTOPIA Level 2*. Para detalles ver [14].

RxData_U[7-0]	*SAR*	*ITUL2*	Consiste de un bus de 8 líneas de datos de entrada de alguna capa *PHY* hacia la *IT/UL2.*
RxSOC_U	*SAR*	*ITUL2*	Señal activa en alto cuando *RxData_U* contiene el primer byte válido de la celda.
RxEnb_U	*ITUL2*	*SAR*	Señal tri-estado activa en bajo cuando *RxData_U* contiene datos válidos de la celda
RxClav_U	*SAR*	*ITUL2*	Señal tri-estado activa en alto, que se utiliza para indicar que se acepta la transferencia de una celda completa hacia la *IT/UL2*, de cualquiera de las 31 posibles capas *PHY.*

Tabla 3-2 Señales del protocolo de comunicación *TOPIA/UTOPIA Level 2*

3.3.2. Operación y temporizado del protocolo de comunicación *UTOPIA Level 2*

3.3.2.1. *Flujo de celdas de la interfaz* TOPIA/UTOPIA Level 2 *hacia el circuito* SAR

El dato a transmitir es transferido de la *IT/UL2* hacia el circuito *SAR*, empleando el siguiente procedimiento:

1. La *IT/UL2* supervisa constantemente el estado de la señal *TxClav_U*, poniendo la dirección correspondiente de cualquiera de las posibles capas *PHY* en el bus *TxAddr_U* (para mayor detalle ver [14]).
2. El circuito *SAR* responde en el siguiente ciclo de reloj con la señal *TxClav_U* (cuando *TxClav_U* está en estado alto, significa que el circuito *SAR* responde que es capaz de aceptar una celda completa).
3. La *IT/UL2* selecciona una de las posibles capas *PHY* para la transferencia de una celda completa, poniendo la dirección del circuito SAR (una de las posibles capas *PHY*) en el bus *TxAddr_U* y la señal *TxEnb_U* en estado alto, en el ciclo actual de *Clk* y pone en estado bajo esta misma señal (*TxEnb_U*) en el siguiente ciclo de *Clk.*

Todas las capas *PHY* (circuito *SAR*) solo examinan el valor de *TxAddr_U* para propósitos de selección cuando la señal *TxEnb_U* está en estado alto. El circuito *SAR* es seleccionado iniciando desde el ciclo siguiente en el que su dirección estuvo en el bus *TxAddr_U* y *TxEnb_U* estuvo en estado alto. La selección finaliza en el ciclo en el que un nuevo circuito *SAR* (una nueva capa *PHY*) es direccionado para su selección y la señal *RxEnb_U* está en estado alto.

3.3.2.2. *Flujo de celdas del circuito* SAR *hacia la interfaz* TOPIA/UTOPIA Level 2

El dato a recibir es transferido de alguna capa *PHY* (circuito *SAR*) hacia la *IT/UL2*, empleando el siguiente procedimiento:

1. La *IT/UL2* supervisa constantemente el estado de la señal *RxClav_U*, poniendo la dirección correspondiente de cualquiera de las posibles capas *PHY* (del circuito *SAR*) en el bus *RxAddr_U*.
2. La capa *PHY* correspondiente (en nuestro caso, el circuito *SAR*), responde en el siguiente ciclo de *Clk* con la señal *RxClav_U* (cuando *RxClav_U* está en estado alto significa que la capa *PHY* responde que es capaz de transmitir una celda completa).
3. La *IT/UL2* selecciona una de las capas *PHY* para la recepción de una celda completa, poniendo la dirección del circuito *SAR* en el bus *RxAddr_U* y la señal *RxEnb_U* en estado alto en el ciclo actual de *Clk*. Después pone en estado bajo esta misma señal (*RxEnb_U*) en el siguiente ciclo de *Clk*.

Todas las capas *PHY* solo examinan el valor de *RxAddr_U*, para propósitos de selección, cuando la señal *RxEnb_U* está en estado alto. La capa *PHY* es seleccionada iniciando desde el ciclo siguiente en el que su dirección estuvo en el bus *RxAddr_U* y al mismo tiempo *RxEnb_U* en estado alto. La selección finaliza en el ciclo en el que una nueva capa *PHY* se le direcciona para su selección y la señal *TxEnb_U* se encuentra en estado alto.

3.4. Interfaz de recepción

La *I_Rx* (dentro de la *IT/UL2*) es la encargada de realizar el cambio del protocolo de comunicación *UTOPIA Level 2* al protocolo *TOPIA* (que es el protocolo propio de la *MC*). En la figura 3-6, se muestra esta interfaz, así como los bloques y las señales de comunicación entre ésta y otros bloques que interactúan.

Los bloques que interactúan con la *I_Rx* son: la interfaz con la tabla de enrutado (*ITR*), el bloque de *loopback* del lado de *UTOPIA Level 2* (*LBU*) y el bloque de *loopback* del lado de *TOPIA* (*LBT*). Los bloques *LBT* y *LBU* manejan las señales ya descritas en las secciones anteriores (ver secciones 3.2 y 3.3 respectivamente). El bloque *ITR* se encarga de recibir la petición que hace la *I_Rx* para hacer una consulta a la *TR* y entregar el resultado de esta consulta a la *I_Rx*; así esta interfaz forma los 3 bytes extras (protocolo *TOPIA*) de la celda de 56 bytes. Además, el bloque *ITR* tiene otras funciones que serán descritas en secciones posteriores. En la tabla 3-3, se describen las señales del bloque *ITR* que interactúan con la *I_Rx*.

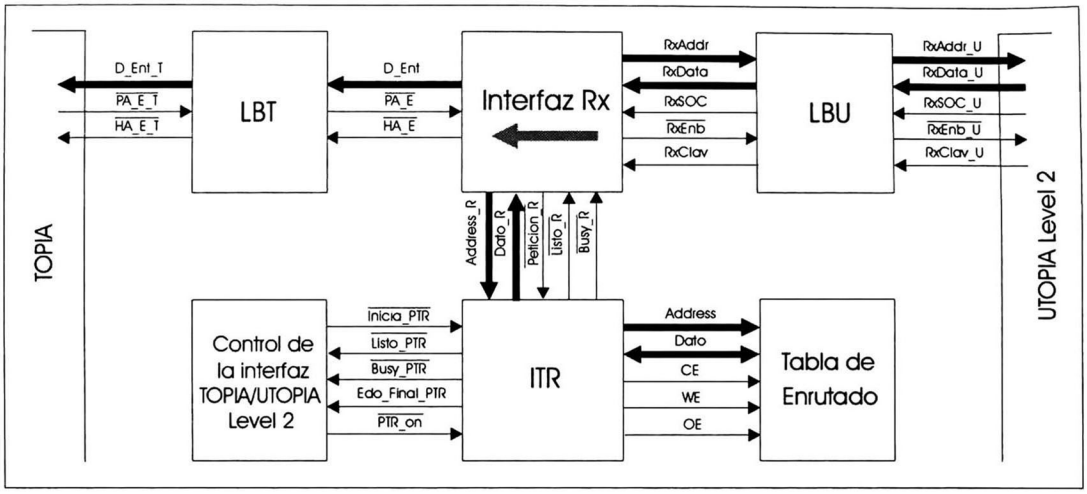


Figura 3-6 Interfaz de recepción de la interfaz *TOPIA/UTOPIA Level 2*

Señal	Origen	Destino	Descripción
Address_R[15-0]	*I_Rx*	*ITR*	Consiste de un bus de 16 líneas formadas por el *VPI* y el *VCI* de encabezado de la celda *ATM*, para formar una dirección de 16 líneas. Estas 16 líneas forman una dirección hacia la *TR*. Nota: Para cada combinación de *VPI* y *VCI* se cuenta con 2 bytes de datos, dentro de la *TR*.
Dato_R[15-0]	*ITR*	*I_Rx*	Consiste de un bus de 16 líneas que forman los dos bytes necesarios para enrutar la información dentro de la *MC*. Estos 2 bytes se obtienen de la *TR* mediante la dirección *Address_R*.
Peticion_R	*I_Rx*	*ITR*	Señal activa en bajo empleada por la *I_Rx*. Esta señal indica a la *ITR* que existe una petición de lectura hacia esta tabla.
Listo_R	*ITR*	*I_Rx*	Señal activa en bajo empleada por la *ITR*. Esta señal indica a la *I_Rx* que la petición de lectura ya fue hecha y que el resultado de la lectura de la dirección *Address_R* (par *VPI* y *VCI*) se encuentra en el bus *Dato_R*.
Busy_R	*ITR*	*I_Rx*	Señal activa en bajo empleada por la *ITR*. Esta señal indica a la *I_Rx* que aún no se cuenta con el resultado de la lectura a la *TR*, pero que está en proceso.

Tabla 3-3 Señales empleadas entre la interfaz de recepción y la interfaz de la tabla de enrutado

3.4.1. Esquema general de solución para la interfaz de recepción

La idea básica en este esquema de solución es tener una memoria dividida a la mitad, con capacidad de dos celdas *ATM* (106 bytes). Ver figura 3-7.

Se recibe una celda del circuito *SAR* y esta es guardada en una de las 2 mitades de la memoria. Mientras una mitad de la memoria es utilizada para guardar la celda que está enviando el circuito *SAR*, la otra mitad de la memoria es utilizada para transmitir a la *MC* la celda recibida anteriormente.

También se tienen tres registros *(H1, H2* y *H3)* en los que se almacenan los tres primeros bytes del encabezado de la celda *ATM*, ya que de estos se toma la información de *VPI* y *VCI* porque con esta información obtenemos de la *TR* los dos primeros bytes de la celda de 56 bytes (del protocolo *TOPIA*), que son necesarios para el enrutado de la información por la *MC*.

Entonces, mientras se está recibiendo una celda *ATM*, paralelamente se consulta en la *TR* los dos primeros bytes con la dirección formada del par *VPI* y *VCI* extraídos de esta celda. Estos dos bytes leídos de la *TR* se almacenan en dos registros (*R1* y *R2*) para cuando sea oportuno ser leídos y así ser transmitidos hacia la *MC*.

Es necesario tener 2 pares de registros *R1* y *R2*. Uno por mitad de memoria (esto es, un par de registros por celda); ya que de no ser así se puede perder la información de *R1* y *R2* si solo se cuenta con un solo par de estos registros.

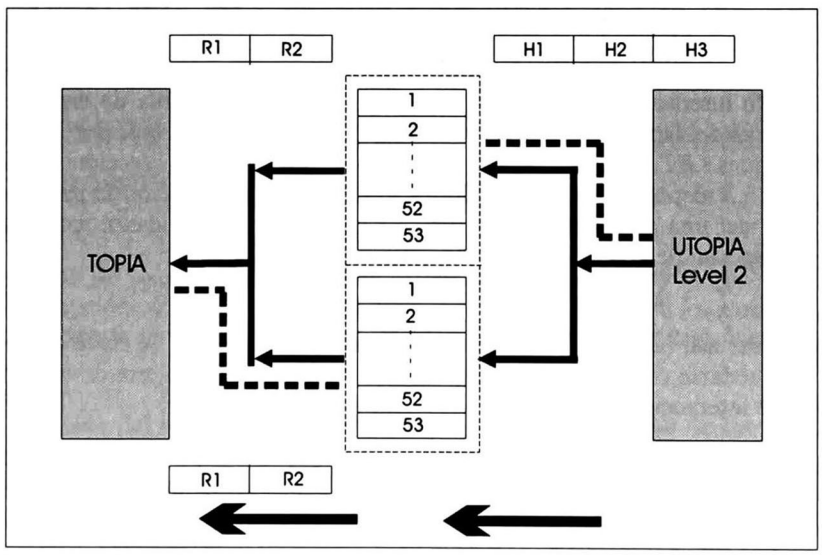


Figura 3-7 Esquema general de solución para la interfaz de recepción

3.5. Interfaz de transmisión

La interfaz de transmisión (*I_Tx*) que está dentro de la *IT/UL2*, es la encargada de realizar el cambio del protocolo de comunicación *TOPIA* (que es propio de la *MC*) a *UTOPIA Level 2*. En la figura 3-8 se muestra esta interfaz, los bloques que interactúan con esta así como las señales de comunicación entre esta interfaz y los otros bloques.

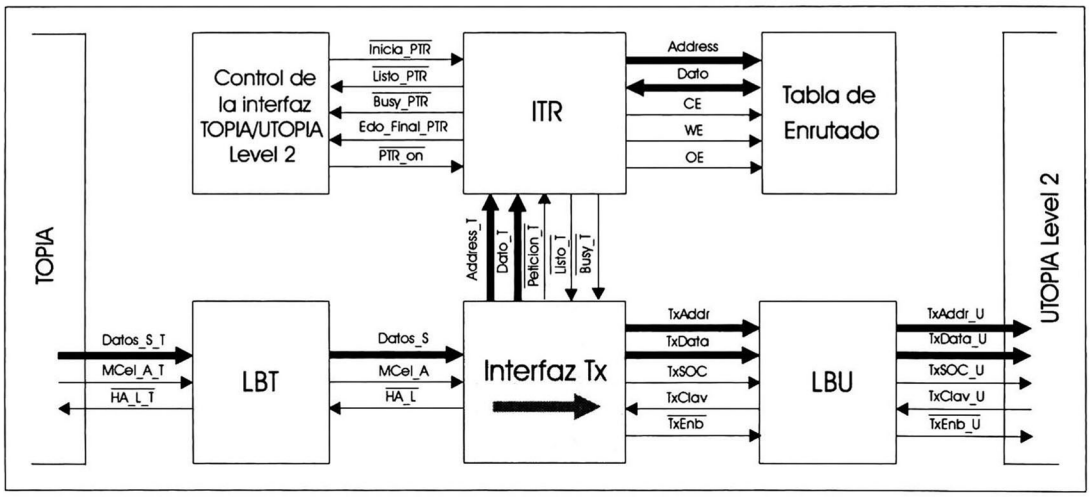


Figura 3-8 Interfaz de transmisión de la interfaz *TOPIA/UTOPIA Level 2*

Los bloques que interactúan con la *I_Tx* son: la interfaz con la tabla de enrutado (*ITR*), el bloque de *loopback* del lado de *UTOPIA Level 2* (*LBU*) y el bloque de *loopback* del lado de *TOPIA* (*LBT*). Los bloques *LBT* y *LBU* manejan las señales ya descritas en las secciones anteriores (ver secciones 3.2 y 3.3 respectivamente). El bloque *ITR* se encarga de recibir la petición (que hace la *I_Tx*) para hacer una escritura a la *TR* (en caso de que la celda que esté recibiendo la *I_Tx*, sea una celda especial de actualización de la *TR*).

En caso de no ser así, la *I_Tx* solo desecha los 3 primeros bytes (de la celda con el protocolo *TOPIA*) y así quedarse con los 53 bytes de la celda *ATM* "pura". En este caso (celda normal), la *I_Tx* no tiene interacción con la *ITR*.

En la tabla 3-4, se describen las señales del bloque *ITR*, que interactúan con la interfaz de transmisión.

Señal	Origen	Destino	Descripción
Address_T[15-0]	*I_Tx*	*ITR*	Consiste de un bus de 16 líneas formadas por los dos siguientes bytes al encabezado de una celda especial de actualización de la *TR*. Estos dos bytes serán el par *VPI* y el *VCI* de encabezado de la celda *ATM* que tendrán asignado un par de bytes de enrutado (en la *TR*); estos le sirven a la *MC* para enrutar la información dentro de esta. Nota: Para cada combinación de *VPI* y *VCI* se cuenta con 2 bytes de datos, dentro de la *TR*.
Dato_T[15-0]	*I_Tx*	*ITR*	Consiste de un bus de 16 líneas que forman los dos bytes necesarios para enrutar la información dentro de la *MC*. Estos 2 bytes se obtienen de los bytes *R1b* y *R2b* de la celda especial de actualización de la *TR*.
Peticion_T	*I_Tx*	*ITR*	Señal activa en bajo empleada por la *I_Tx*. Esta señal indica a la *ITR* que existe una petición de escritura hacia esta tabla.
Listo_T	*ITR*	*I_Tx*	Señal activa en bajo empleada por la *ITR*. Esta señal indica a la *I_Tx* que la petición de escritura ya fue hecha y que el dato *Dato_T* (bytes *R1b* y *R2b*) está almacenado en la *TR* en la dirección *Address_T* (par *VPI* y *VCI*).
Busy_T	*ITR*	*I_Tx*	Señal activa en bajo empleada por la *ITR*. Esta señal indica a la *I_Tx* que está en proceso la escritura a la *TR*, pero que aún no se finaliza con esta.

Tabla 3-4 Señales empleadas entre la interfaz de transmisión y la interfaz de la tabla de enrutado

3.5.1. Esquema general de solución para la interfaz de transmisión

El procedimiento de solución para esta interfaz tiene la misma filosofía que se empleó en la interfaz de recepción. La idea básica en este esquema de solución es también tener una memoria dividida a la mitad con capacidad de dos celdas *ATM* (106 bytes). Ver figura 3-9.

Se recibe una celda del protocolo *TOPIA* (*MC*) y se almacena en una de las 2 mitades de la memoria. Mientras una de las mitades de esta memoria es empleada para almacenar la celda que envía la *MC*, la otra se emplea para transmitir a alguna de las capas *PHY* (circuito *SAR*) la celda recibida anteriormente.

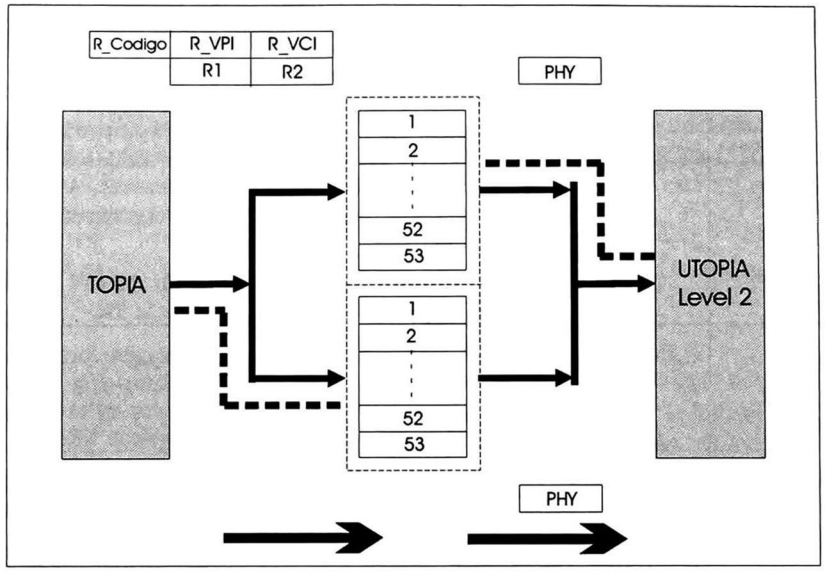


Figura 3-9 Esquema general de solución para la interfaz de transmisión

En esta interfaz se tienen cinco registros (*R_Código*, *R_VPI*, *R_VCI*, *R_R1* y *R_R2*) en los que se almacenan los datos requeridos en caso de hacer una actualización de la *TR* (Ver figura 3-3, formato de celda *TOPIA* de 56 bytes). En caso de que la celda que se recibe por esta interfaz de transmisión no sea una celda especial de actualización de la *TR*, se recibe la celda *ATM* en una de las mitades de la memoria; mientras que la otra sirve para transmitir la celda anteriormente almacenada hacia alguna de las capas *PHY* (circuito *SAR*, en nuestro caso).

Es necesario tener un registro *PHY* por celda, para almacenar en este la dirección de la capa *PHY* a la cual se tiene que enviar esta celda. En este registro se almacena la dirección cuando se reciben los bytes del encabezado de la celda *ATM* (véase figura 3-10, encabezado de la celda *ATM*).

Es necesario tener 2 registros *PHY* uno por mitad de memoria, ya que de no ser así, se puede perder la información de la capa *PHY* a la cual va dirigida la celda (si solo se cuenta con un solo registro).

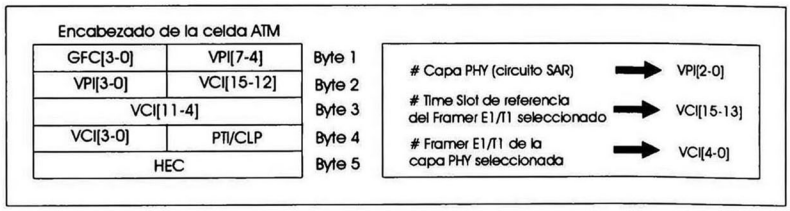


Figura 3-10 Encabezado de la celda *ATM*

3.6. Tabla de enrutado

La tabla de enrutado (*TR*) es una memoria *SRAM* o *EEPROM* externa a la *IT/UL2* pero manejada por esta interfaz. La capacidad de esta memoria es de 128 Kbytes de información. En esta memoria se almacena la información necesaria para el enrutado de las celdas *ATM* dentro de la *MC*.

La dirección de la memoria es formada por el *VPI* y *VCI* del encabezado de la celda *ATM*. Entonces, para cada combinación de *VPI* y *VCI* se cuenta con un par de bytes (*R1a* y *R2a*, ver figura 3-3). Este par de bytes forman la información adicional a la celda *ATM* "pura" (53 bytes) para así formar la celda *ATM* de 56 bytes (protocolo *TOPIA*).

En la *IT/UL2* se cuenta (fisicamente) con una memoria *EEPROM* de 128 Kbytes * 8 bits (*AT28LV010*) de la compañía *ATMEL* [26]. Para el almacenamiento de esta información (2 bytes por par de *VPI* y *VCI*) se emplean los 16 bits (64 Kbytes) más significativos de los 17 bits de dirección totales (128 Kbytes) de esta memoria; y el bit menos significativo de este bus de direcciones indica si es el byte *R1a* o *R2a*. De esta forma, se tienen 8 bits de direcciones en *VPI* (los más significativos del bus de 16 bits) y los otros 8 bits de direcciones en *VCI* (los menos significativos del bus de 16 bits). En la figura 3-11 se muestran los bloques y las señales que interactúan con la *TR*.

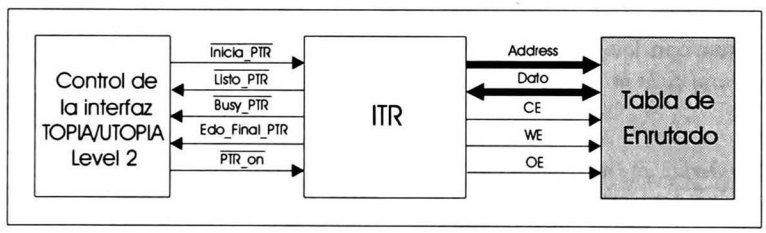


Figura 3-11 Tabla de enrutado

Señal	Origen	Destino	Descripción
Address[16-0]	*ITR*	*TR*	Consiste de un bus de 17 líneas de direcciones. Este bus es manejado por la *ITR*.
Dato[7-0]	*ITR* *TR*	*TR* *ITR*	Consiste de un bus de 8 líneas de datos bidireccionales. Este bus contiene el dato a ser almacenado por la *TR* (memoria *AT28LV010*) en una operación de escritura. También, este bus contiene el dato resultante de una operación de lectura a esta memoria. La operación de escritura o lectura de una localidad específica de esta memoria depende de la dirección del bus *Address*. El bus *Dato* es puesto en alta impedancia cuando la señal *CE* o la señal *OE* están en estado alto.
CE	*ITR*	*TR*	Señal activa en bajo empleada por la *ITR* para activar o desactivar alguna operación (lectura o escritura) de la *TR*.

WE	*ITR*	*TR*	Señal activa en bajo empleada por la *ITR* para una operación de escritura (estado bajo de la señal) o una operación de lectura (estado alto de la señal).
OE	*ITR*	*TR*	Señal activa en bajo empleada por la *ITR* para indicarle a la *TR* (memoria *AT28LV010*) que puede poner en el bus de datos *Dato* el resultado de una lectura.

Tabla 3-5 Señales empleadas entre la tabla de enrutado y la interfaz de la tabla de enrutado

3.7. Interfaz de la tabla de enrutado

La interfaz de la tabla de enrutado (*ITR*) que está dentro de la *ITU/L2*, es la encargada de realizar las consultas y las actualizaciones a la *TR*. Esta interfaz también es capaz de realizar una prueba en línea (ver "*on-line BIST*" en [17][18][19][20][21]) de la *TR* si así es solicitada por el administrador del sistema (el administrador de la *IE1-T1/MC*). Para llevar estas funciones a cabo, la *ITR* cuenta con tres bloques principales: el bloque de prueba de la tabla de enrutado (*PTR*), *control tabla de enrutado* y el bloque de selección de la tabla de enrutado (*SEL_PTR*).

La *ITR* interactúa con los siguientes bloques: *tabla de enrutado*, *interfaz de recepción*, *interfaz de transmisión* y *control de la interfaz TOPIA/UTOPIA Level 2*.

Del bloque de *interfaz de recepción* recibe peticiones de lectura de la *TR*. En caso de ser posible (si otro bloque no esta empleando la *TR*) la interfaz hace la lectura de la *TR* y entrega el resultado al bloque de *interfaz de recepción*.

Del bloque de *interfaz de transmisión* recibe peticiones de escritura a la *TR* (cuando la *interfaz de transmisión* recibe una celda especial de actualización de la *TR*). En caso de ser posible (si no se esta empleando la *TR* por otro bloque) la interfaz hace la escritura hacia esta tabla.

Del bloque *control de la interfaz TOPIA/UTOPIA Level 2* recibe peticiones (enviadas por el administrador del sistema) para hacer la prueba de la *TR*. Al recibir esta petición y en caso de ser posible (si no la está empleando otro bloque), la interfaz realiza la prueba de la *TR* y reporta los resultados de esta prueba al administrador del sistema.

En la figura 3-12, se muestra la *ITR*, así como los bloques y las señales de comunicación entre esta interfaz y otros bloques que interactúan con esta.

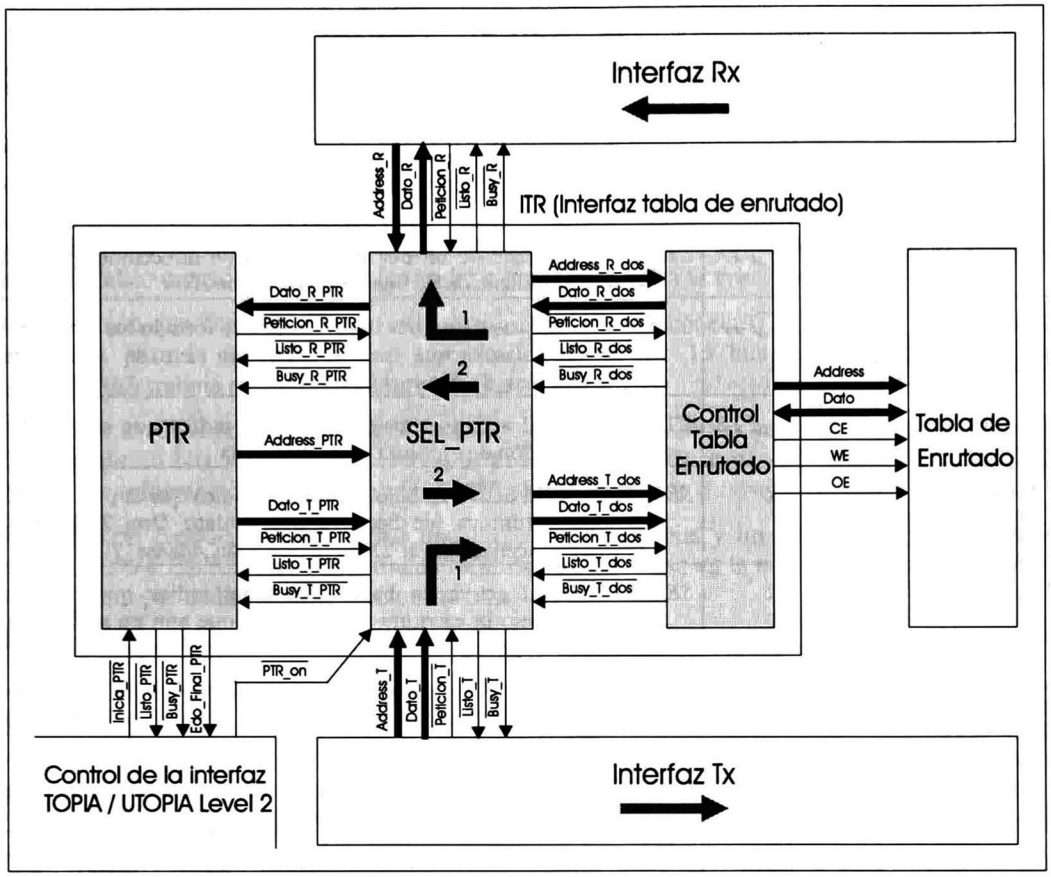


Figura 3-12 Interfaz de la tabla de enrutado

En la tabla 3-6 se describen las señales que interactúan entre los bloques: *PTR*, *SEL_PTR* y *control tabla de enrutado*.

Las señales que interactúan con la *ITR* y los bloques: *interfaz de recepción*, *interfaz de transmisión*, *tabla de enrutado* y *control de la interfaz TOPIA/UTOPIA Level 2*; se describen en las secciones de estos bloques.

Señal	Origen	Destino	Descripción
Address_R_dos[15-0]	*SEL_PTR*	*CTR*	Consiste de un bus de 16 líneas de direcciones para la consulta de una localidad de la *TR*.
Dato_R_dos[15-0]	*CTR*	*SEL_PTR*	Consiste de un bus de 16 líneas que forman los dos bytes de datos que se leen de la dirección *Address_R_dos* de la *TR*.
Peticion_R_dos	*SEL_PTR*	*CTR*	Señal activa en bajo. Esta señal indica que existe una petición de lectura hacia esta tabla.

Listo_R_dos	*CTR*	*SEL_PTR*	Señal activa en bajo. Esta señal indica que la petición de lectura ya fue hecha y que el resultado de la lectura de la dirección *Address_R_dos* se encuentra en el bus *Dato_R_dos.*
Busy_R_dos	*CTR*	*SEL_PTR*	Señal activa en bajo. Esta señal indica que aún no se cuenta con el resultado de la lectura a la *TR* pero que está en proceso.
Address_T_dos[15-0]	*SEL_PTR*	*CTR*	Consiste de un bus de 16 líneas de direcciones para la actualización de una localidad de la *TR.*
Dato_T_dos[15-0]	*SEL_PTR*	*CTR*	Consiste de un bus de 16 líneas que forman los dos bytes de datos que se almacenarán en la dirección *Address_T_dos.*
Peticion_T_dos	*SEL_PTR*	*CTR*	Señal activa en bajo. Esta señal indica que existe una petición de escritura hacia esta tabla.
Listo_T_dos	*CTR*	*SEL_PTR*	Señal activa en bajo. Esta señal indica que la petición de escritura ya fue hecha y que el dato *Dato_T_dos* está almacenado en al *TR* en la dirección *Address_T_dos.*
Busy_T_dos	*CTR*	*SEL_PTR*	Señal activa en bajo. Esta señal indica que está en proceso la escritura a la *TR* pero que aún no se finaliza con esta.
Dato_R_PTR[15-0]	*SEL_PTR*	*PTR*	Consiste de un bus de 16 líneas que forman los dos bytes de datos que se leen de la dirección *Address_PTR* de la *TR.*
Petición_R_PTR	*PTR*	*SEL_PTR*	Señal activa en bajo. Esta señal indica que existe una petición de lectura hacia esta tabla.
Listo_R_PTR	*SEL_PTR*	*PTR*	Señal activa en bajo. Esta señal indica que la petición de lectura ya fue hecha y que el resultado de la lectura de la dirección *Address_PTR* se encuentra en el bus *Dato_R_PTR.*
Busy_R_PTR	*SEL_PTR*	*PTR*	Señal activa en bajo. Esta señal indica que aún no se cuenta con el resultado de la lectura a la *TR* pero que está en proceso.
Address_PTR[15-0]	*PTR*	*SEL_PTR*	Consiste de un bus de 16 líneas de direcciones para la lectura o escritura de una localidad de la *TR.*
Dato_T_PTR[15-0]	*PTR*	*SEL_PTR*	Consiste de un bus de 16 líneas que forma el dato que se almacenará en la dirección *Address_PTR.*
Petición_T_PTR	*PTR*	*SEL_PTR*	Señal activa en bajo. Esta señal indica que existe una petición de escritura hacia esta tabla.
Listo_T_PTR	*SEL_PTR*	*PTR*	Señal activa en bajo. Esta señal indica que la petición de escritura ya fue hecha y que el dato *Dato_T_PTR* está almacenado en la *TR* en la dirección *Address_PTR.*
Busy_T_PTR	*SEL_PTR*	*PTR*	Señal activa en bajo. Esta señal indica que está en proceso la escritura a la *TR* pero que aún no se finaliza con esta.

Tabla 3-6 Señales de la interfaz de la tabla de enrutado

3.7.1. Bloque "control de la tabla de enrutado"

El bloque de control de la tabla de enrutado (*CTR*) es el encargado de hacer las consultas y las actualizaciones de la *TR*. Este bloque recibe peticiones de consulta o de actualización del bloque *SEL_PTR* (selección prueba tabla de enrutado).

Este bloque puede recibir la petición de consulta cuando se está en proceso de actualización de la *TR* y viceversa. Esto es, recibir una petición de actualización mientras se está en proceso de consulta de la *TR*. La función de este bloque es la de servir como "árbitro"; de tal forma que, solo se realice una sola operación (consulta o actualización) a la vez.

Otra función que tiene este bloque es la de adecuar las señales de la *TR*. A este bloque le llegan los datos para la actualización o consulta en modo de 16 bits, pero la *TR* (memoria *AT28LV010*) trabaja en modo de 8 bits.

El bloque de *CTR* recibe el bus de direcciones (para la consulta o actualización de la *TR*) en formato de 16 bits y por el otro lado (por el lado de la *TR*) se tiene un bus de 17 bits (128 Kbytes de información en la memoria *AT28LV010*).

Entonces, el bloque de *CTR* recibe una dirección de 16 bits y un dato de 16 bits. Y para almacenar o consultar una determinada localidad de memoria en la tabla de enrutado hace dos operaciones por cada consulta o actualización.

En el caso de consulta:

1. Recibe una dirección de 16 bits del bloque *SEL_PTR*.
2. Hace una lectura a la *TR* formando una dirección de 17 bits con los 16 bits que recibe del bloque *SEL_PTR*, poniéndolos como los más significativos y poniendo "cero lógico" en el bit menos significativo.
3. Este byte leído de la *TR* formará el byte más significativo del resultado de la consulta.
4. Hace una lectura a la *TR* formando una dirección de 17 bits con los 16 bits que recibe del bloque *SEL_PTR*, poniéndolos como los más significativos y poniendo "uno lógico" en el bit menos significativo.
5. Este byte leído de la *TR* formará el byte menos significativo del resultado de la consulta.

En el caso de la actualización:

1. Recibe una dirección de 16 bits del bloque *SEL_PTR*.
2. Hace una escritura a la *TR* formando una dirección de 17 bits con los 16 bits que recibe del bloque *SEL_PTR*, poniéndolos como los más significativos y poniendo "cero lógico" en el bit menos significativo. Después, tomando el byte menos significativo del dato recibido del bloque *SEL_PTR*, que será el dato que se almacenará en dicha localidad de 17 bits de dirección.
3. Hace una escritura a la *TR* formando una dirección de 17 bits con los 16 bits que recibe del bloque *SEL_PTR*, poniéndolos como los más significativos y poniendo "uno lógico" en el bit menos significativo. Después, tomando el byte más significativo del dato recibido del bloque *SEL_PTR*, que será el dato que se almacenará en dicha localidad de 17 bits de dirección.

Una vez hechas estas dos operaciones tanto para consulta como para actualización, el bloque de *CTR* avisa al bloque *SEL_PTR* que fue realizada la operación indicada y en caso de consulta, el resultado de esta se encuentra disponible para el bloque *SEL_PTR*.

3.7.2. Bloque *PTR* (prueba de la tabla de enrutado)

Este bloque de prueba de la tabla de enrutado (*PTR*) es el encargado de realizar la prueba de la memoria en donde se almacena la *TR*. El bloque *PTR* recibe la petición de inicio de la prueba del bloque *control de la interfaz TOPIA/UTOPIA Level 2* que fue enviada por el administrador del sistema.

La filosofía empleada en este bloque para la ejecución de la prueba de la *TR* fue basada en "*on-line BIST*". De forma general, este tipo de prueba está basado en 5 bloques principales: un *generador de patrones de prueba*, un *analizador de resultados*, el *control de pruebas*, un *circuito multiplexor* y el *circuito bajo prueba*. Ver figura 3-13.

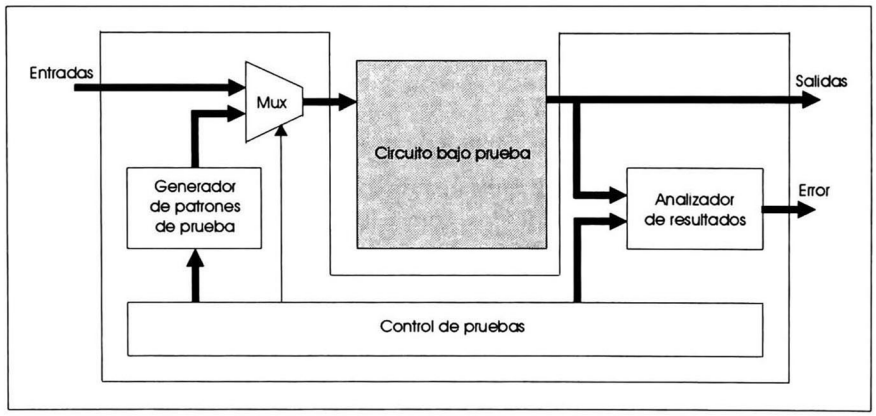


Figura 3-13 Esquema general de "*on-line BIST*"

El bloque *circuito bajo prueba* en nuestro caso es la memoria en la que se encuentra almacenada la *TR*. Los bloques: *generador de patrones de prueba*, *analizador de resultados* y *control de pruebas*; se encuentran contenidos en el bloque *PTR* del bloque *ITR*. Y por último, el bloque *circuito multiplexor* (mux) es el bloque de selección de prueba de la tabla de enrutado (*SEL_PTR*) que también está contenido en el bloque *ITR*.

Una vez recibida la petición de inicio de la prueba de la *TR*, el primer paso es el ajustar el *circuito multiplexor* para que acepte las señales del *generador de patrones de prueba* y no las señales de entrada normales. Después de esto, el *generador de patrones de prueba* "inyecta" al *circuito bajo prueba* las señales o patrones a los que este circuito deberá de responder de una forma única y conocida por el bloque *analizador de resultados*. El bloque *analizador de resultados*, una vez que recibe y analiza las salidas del *circuito bajo prueba*, reporta al bloque *control de pruebas* si los resultados fueron satisfactorios o no lo fueron. Y finalmente, el *bloque de control de pruebas* reporta los resultados obtenidos de las pruebas realizadas al *circuito bajo prueba*. Finalizada la prueba, el *circuito multiplexor* es ajustado para dejar pasar solo las señales normales de entrada.

Entonces, una vez que el bloque *PTR* recibió la petición del inicio de la prueba de la *TR*, este bloque procede a la ejecución de esta prueba.

3.7.2.1. Pseudocódigo de prueba para la tabla de enrutado

Enseguida se describe en pseudocódigo los pasos que realiza este bloque (*PTR*) para determinar si la memoria donde está almacenada la *TR* es o no confiable.

1. Leer dato (almacenado en la *TR*) de la localidad de memoria que se está probando.
2. Guardar temporalmente este dato en registros (de la localidad de memoria que se está probando).
3. Escribir el dato "AAAAh" (1010101010101010b) en la localidad de memoria que se está probando.
4. Leer el dato almacenado de la localidad de memoria que se está probando.
5. Comparar este dato leído con "AAAAh"
   - En caso de no coincidir el dato leído con "AAAAh", se aborta la prueba y se reporta el fallo de la memoria en donde está almacenada la *TR*.

     En caso de coincidir el dato leído con "AAAAh" se continúa con el paso 6.
6. Escribir el dato "5555h" (0101010101010101b) en la localidad de memoria que se está probando.
7. Leer el dato almacenado de la localidad de memoria que se está probando.
8. Comparar este dato leído con "5555h"
   - En caso de no coincidir el dato leído con "5555h", se aborta la prueba y se reporta el fallo de la memoria en donde está almacenada la *TR*.
   - En caso de coincidir el dato leído con "5555h", se continúa con el paso 9.
9. Guardar el dato que estaba originalmente almacenado (y que se tiene disponible en registros) de la localidad de memoria que se está probando.
10. Leer el dato almacenado de la localidad de memoria que se está probando y así cerciorarse de que este quedó como originalmente estaba almacenado.
    - En caso de no coincidir el dato leído con el almacenado en registros, se aborta la prueba y se reporta el fallo de la memoria donde está almacenada la *TR*.
    - En caso de coincidir el dato leído con el dato almacenado en registros, se continúa con el paso 11.
11. Se repiten todos los pasos hasta no acabar con todas las localidades de memoria de la *TR* (64 Kwords).

Una vez realizada esta prueba de la memoria en donde se almacena la *TR*, el bloque *PTR* reporta al bloque *control de la interfaz TOPIA/UTOPIA Level 2* los resultados de esta prueba, para que a su vez, este último bloque reporte al administrador del sistema estos mismos resultados.

3.7.3. Bloque de selección de la prueba de la tabla de enrutado (*SEL_PTR*)

Este bloque (*SEL_PTR*) es el encargado de la selección entre las señales provenientes del bloque *PTR* y de los bloques: *interfaz de recepción* e *interfaz de transmisión*. En el esquema general de "*on-line BIST*" (ver figura 3-13) este bloque (*SEL_PTR*) es el bloque descrito como *circuito multiplexor*.

La decisión de que señales debe dejar pasar hacia el bloque *control tabla de enrutado* es hecha mediante la señal *PTR_on* (descrita a detalle en la sección 3-10), que es manejada por el bloque *control de la interfaz TOPIA/UTOPIA Level 2*.

Existen 2 posibilidades de operación de este bloque (ver figura 3-12).

1. Cuando la señal *PTR_on* está en estado alto, las señales que dejará pasar este bloque (*SEL_PTR*) serán las señales provenientes de los bloques: *Interfaz de recepción* e *interfaz de transmisión*; para una operación normal de la *IT/UL2*.

2. Cuando la señal *PTR_on* está en estado bajo, las señales que dejará pasar este bloque (*SEL_PTR*) serán las señales provenientes del bloque *PTR*, para que de esta forma se lleve a cabo la prueba de la *TR*.

3.8. Loopback del lado del protocolo *TOPIA*

El bloque *LBT* (*loopback* del lado del protocolo *TOPIA*) forma parte de la *IT/UL2*. La función de este bloque es la de prueba de la *IT/UL2* por el lado del protocolo *TOPIA*. En la figura 3-14, se muestra el bloque *LBT*, así como los bloques y las señales de comunicación entre este y los otros bloques que interactúan con él.

El bloque *LBT* se encuentra físicamente (dentro de la *IT/UL2*) entre la *MC* y la interfaz de recepción como de transmisión.

En operación normal el bloque *LBT* se comporta como un "cable de conexión" entre la *MC* (protocolo *TOPIA*) y las interfaces: *I_Rx* e *I_Tx*. Esto es, cuando las señales *LBT1_on* y *LBT2_on* están en estado alto (ambas); esto indica que las señales del lado del protocolo *TOPIA* se conectan a las señales del lado de estas interfaces. En la figura 3-14 se ilustra este modo de operación con las flechas marcadas con el número 1 dentro del bloque *LBT*. Las flechas marcadas con los números 2 y 3 (que indican también el sentido del flujo de la información) no existen en este modo de operación.

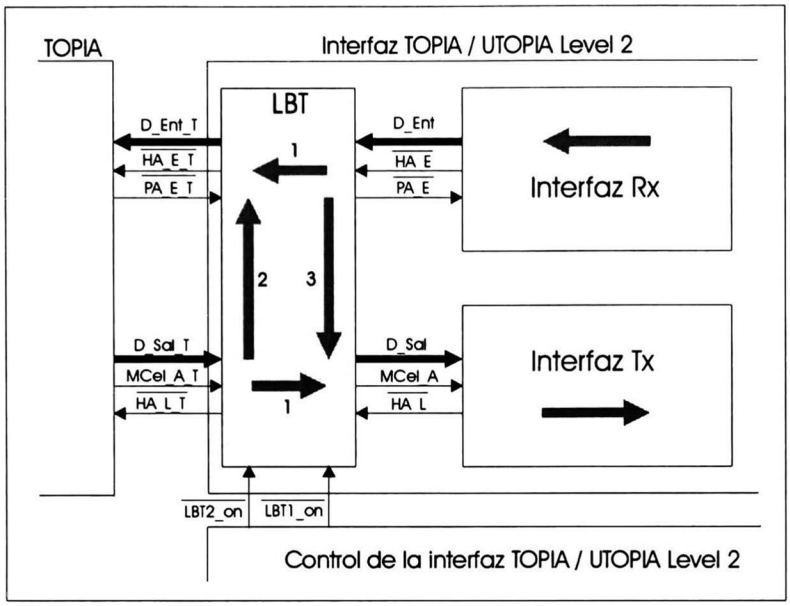


Figura 3-14 *Loopback* del lado del protocolo *TOPIA*

Cuando se pone la *IT/UL2* en modo de prueba por el lado del protocolo *TOPIA*, alguna de las señales *LBT1_on* y *LBT2_on* está en estado bajo.

Si la señal *LBT2_on* está en estado bajo, la información proveniente de la *MC* (protocolo *TOPIA*) no llega a la *interfaz de transmisión*, sino que se regresa a la misma *MC*. Entonces, el sentido del flujo de la información se ilustra con la flecha marcada con el número 2 (ver figura 3-14). Las flechas marcadas con los números 1 y 3 no existen en este modo de operación.

Si la señal *LBT1_on* está en estado bajo, la información proveniente de la *interfaz de recepción* no es entregada a la *MC* (protocolo *TOPIA*); si no que es entregada a la *interfaz de transmisión* para su regreso al protocolo *UTOPIA Level 2*. Entonces, el sentido del flujo de la información se ilustra con la flecha marcada con el número 3 (ver figura 3-14). Las flechas marcadas con los números 1 y 2 no existen en este modo de operación.

Si las señales *LBT1_on* y *LBT2_on* están en estado bajo (ambas), la información proveniente de la *MC* (protocolo *TOPIA*) no llega a la *interfaz de transmisión*, sino que se regresa a la misma *MC*. También, la información proveniente de la *interfaz de recepción* no es entregada a la *MC;* si no que es entregada a la *interfaz de transmisión* para su regreso al protocolo *UTOPIA Level 2*. Entonces, el sentido del flujo de la información se ilustra con las flechas marcada con los números 2 y 3 (ver figura 3-14). Las flechas marcadas con el número 1 no existen en este modo de operación.

En caso de que alguna de las señales *LBT1_on* y *LBT2_on* se encuentre en estado bajo y la otra señal se encuentre en estado alto, la interfaz *IT/UL2* se encontrará en cualquiera de los dos primeros modos de prueba descritos anteriormente.

En la sección 3.2, se describieron las señales del protocolo *TOPIA* y las señales *LBT1_on* y *LBT2_on* se describirán más a detalle en la sección 3.10.

En la tabla 3-7, se describen las señales empleadas entre el bloque *LBT* y las interfaces tanto de recepción como de transmisión.

Señal	Origen	Destino	Descripción
D_Ent[7-0]	*I_Rx*	*LBT*	Consiste de un bus de 8 líneas de datos de entrada al bloque *LBT* proveniente la *interfaz de recepción.*
HA_E	*I_Rx*	*LBT*	Señal activa en bajo. Sirve para iniciar la transferencia de un ciclo de celda de la *interfaz de recepción* hacia el bloque *LBT.*
PA_E	*LBT*	*I_Rx*	Señal activa en bajo. Sirve para detener la transmisión de la información de la *interfaz de recepción* hacia el bloque *LBT.*
Datos_S[7-0]	*LBT*	*I_Tx*	Consiste de un bus de 8 líneas de salida del bloque *LBT* hacia la *interfaz de transmisión* de la *IT/UL2.*
MCel_A	*LBT*	*I_Tx*	Señal activa en alto. Sirve para indicarle a la *interfaz de transmisión* de la *IT/UL2,* los instantes cuando hay una microcelda [1][2] (8 bytes de información) disponible para ser leída por esta interfaz.
HA_L	*I_Tx*	*LBT*	Señal activa en bajo. Señal que sirve para iniciar la transferencia de una celda hacia la *interfaz de transmisión* de la *IT/UL2.*

Tabla 3-7 Señales que interactúan entre el bloque *LBT* y las interfaces *I_Rx* e *I_Tx*

3.9. Loopback del lado del protocolo *UTOPIA Level 2*

El bloque *LBU* (*loopback* del lado del protocolo *UTOPIA Level 2*) forma parte de la *IT/UL2.* La función de este bloque es la de prueba de la *IT/UL2* por el lado del protocolo *UTOPIA Level 2.* En la figura 3-15, se muestra el bloque *LBU,* así como los bloques y las señales de comunicación entre este y otros bloques que interactúan con él.

El bloque *LBU* se encuentra físicamente (dentro de la *IT/UL2*) entre las capas *PHY* (en nuestro caso el circuito *SAR*) y las interfaces tanto de recepción como de transmisión.

En operación normal, el bloque *LBU* se comporta como un "cable de conexión" entre las capas *PHY* y las interfaces de recepción/transmisión (esto es, cuando las señales *LBU1_on* y *LBU2_on* están en estado alto). Esto indica que las señales del lado de las capas *PHY* se conectan a las señales del lado de las interfaces de recepción/transmisión. En la figura 3-15, se ilustra este modo de operación con las flechas marcadas con el número 1, dentro del bloque *LBU.* Las flechas marcadas con los números 2 y 3 (que indican también el sentido del flujo de la información) no existen en este modo de operación.

Cuando se pone la *IT/UL2* en modo de prueba por el lado de las capas *PHY,* alguna de las señales *LBU1_on* y *LBU2_on* está en estado bajo.

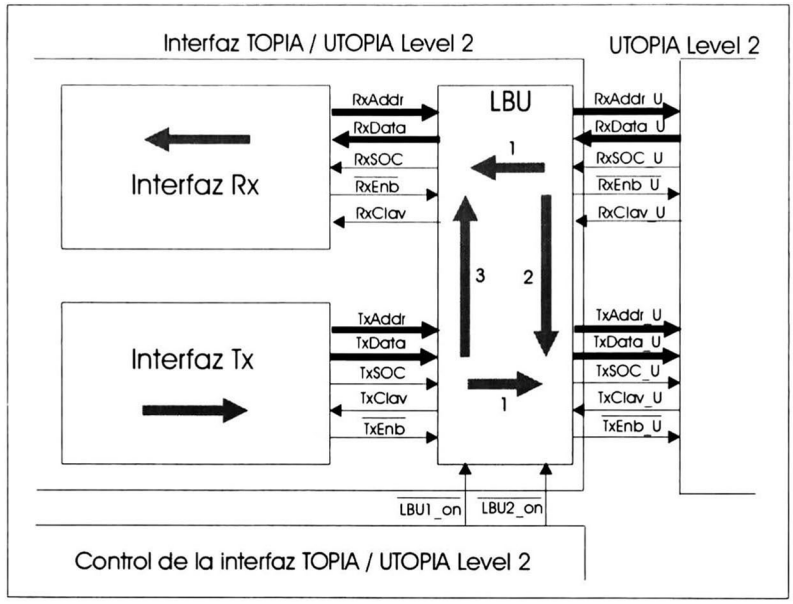


Figura 3-15 *Loopback* del lado del protocolo *UTOPIA Level 2*

Si la señal *LBU2_on* está en estado bajo, la información proveniente de las capas *PHY* no llega a la *interfaz de recepción*, sino que se regresa al mismo circuito *SAR*. Entonces, el sentido del flujo de la información se ilustra con la flecha marcada con el número 2 (ver figura 3-15). Las flechas marcadas con los números 1 y 3 no existen en este modo de operación.

Si la señal *LBU1_on* está en estado bajo, la información proveniente de la *interfaz de transmisión* no es entregada a las capas *PHY*, sino que es entregada a la *interfaz de recepción* para su regreso al protocolo *TOPIA* (*MC*). Entonces, el sentido del flujo de la información se ilustra con la flecha marcada con el número 3. En este modo las flechas 1 y 2 no existen.

Si las señales *LBU1_on* y *LBU2_on* están en estado bajo (ambas), la información proveniente de las capas *PHY* no llega a la *interfaz de recepción*, sino que se regresa al mismo circuito *SAR*. La información proveniente de la *interfaz de transmisión* no es entregada a las capas *PHY*, sino que es entregada a la *interfaz de recepción* para su regreso al protocolo *TOPIA* (*MC*). Entonces, el sentido del flujo de la información se ilustra con las flechas marcadas con los números 2 y 3. No se toman en cuenta en este modo las flechas marcadas con el número 1.

En caso de que alguna de las señales *LBU1_on* y *LBU2_on* se encuentre en estado bajo y la otra señal se encuentre en estado alto, la *IT/UL2* se encontrará en cualquiera de los dos primeros modos de prueba descritos anteriormente.

En la sección 3.3 se describieron las señales del protocolo *UTOPIA Level 2* y las señales *LBU1_on* y *LBU2_on* se describirán más a detalle en la sección 3.10.

En la tabla 3-8 se describen las señales empleadas entre el bloque *LBU* y las interfaces tanto de recepción como de transmisión.

Señal	Origen	Destino	Descripción
TxAddr[4-0]	*I_Tx*	*LBU*	Consiste en un bus de 5 líneas de dirección de salida de la *interfaz de transmisión* hacia el bloque *LBU*. Esta señal es utilizada para seleccionar hasta 31 posibles capas *PHY* (el circuito *SAR* en nuestro caso). El valor de "*1Fh*" en este bus, indica una capa *PHY* nula.
TxData[7-0]	*I_Tx*	*LBU*	Consiste de un bus de 8 líneas de datos de salida de la *interfaz de transmisión* hacia el bloque *LBU*.
TxSOC	*I_Tx*	*LBU*	Señal activa en alto cuando *TxData* contiene el primer byte válido de la celda.
TxEnb	*I_Tx*	*LBU*	Señal activa en bajo cuando *TxData* contiene datos válidos de la celda.
TxClav	*LBU*	*I_Tx*	Señal activa en alto, que se utiliza para indicar que se acepta la transferencia de una celda completa hacia cualquiera de las 31 posibles capas *PHY* (el circuito *SAR* en nuestro caso).
RxAddr[4-0]	*I_Rx*	*LBU*	Consiste en un bus de 5 líneas de dirección de salida de la *interfaz de recepción* hacia el bloque *LBU*. Esta señal es utilizada para seleccionar hasta 31 posibles capas *PHY*. El valor de "*1Fh*" en este bus, indica una capa *PHY* nula.
RxData[7-0]	*LBU*	*I_Rx*	Consiste de un bus de 8 líneas de datos de entrada del bloque *LBU* hacia la *interfaz de recepción*.
RxSOC	*LBU*	*I_Rx*	Señal activa en alto cuando *RxData* contiene el primer byte válido de la celda.
RxEnb	*I_Rx*	*LBU*	Señal activa en bajo cuando *RxData* contiene datos válidos de la celda
RxClav	*LBU*	*I_Rx*	Señal activa en alto, que se utiliza para indicar que se acepta la transferencia de una celda completa hacia la *interfaz de recepción* del bloque *LBU*.

Tabla 3-8 Señales que interactúan entre el bloque *LBU* y las interfaces *I_Rx* e *I_Tx*

3.10. Bloque de control de la interfaz *TOPIA/UTOPIA Level 2*

La función del bloque de control de la *IT/UL2* es la de hacer la interfaz entre el protocolo de comunicación empleado por la *interfaz 8/16 bits* y las señales que controlan la operación de esta interfaz. En la figura 3-16 se muestra el bloque de control de la *IT/UL2*, así como los bloques y las señales de comunicación entre éste y los otros bloques que interactúan con él.

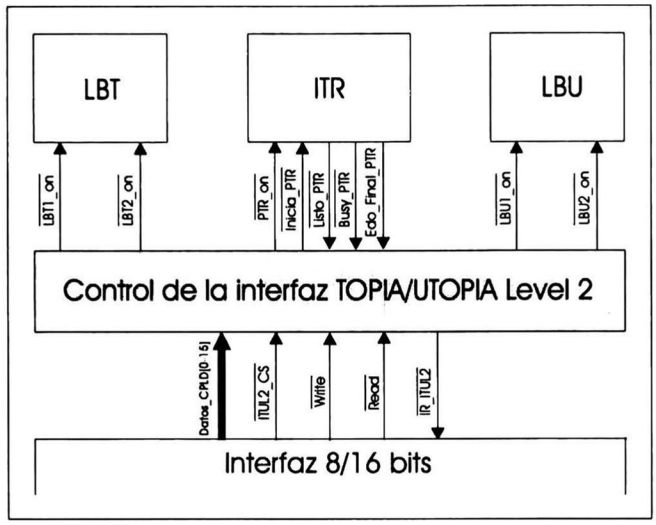


Figura 3-16 Bloque de control de la interfaz *TOPIA/UTOPIA Level 2*

Con el bloque de control de la *IT/UL2* interactúan los bloques: *loopback* del lado del protocolo *TOPIA* (*LBT*), *loopback* del lado del protocolo *UTOPIA Level 2* (*LBU*), la interfaz con la *TR* (*ITR*); todos estos bloques internos a la *IT/UL2*. Y la *interfaz 8/16 bits* (*I8/16b*) que también interactúa con el bloque de control de la *IT/UL2* pero que es externa a esta interfaz.

Este bloque recibe información de la *I8/16b* (proveniente del administrador del sistema) para poner el modo de operación de la *IT/UL2*. También, por medio de este bloque la *IT/UL2* reporta el estado en que se encuentra esta interfaz cuando es solicitado por la *I8/16b*.

La *I8/16b* no utiliza un bus de direcciones para la comunicación con la *IT/UL2*. Esto es debido a que la información (señales en este caso) de comunicación no excede el ancho del bus de 16 bits de datos de la *I8/16b*. Por esto, las señales en el bloque de control de la *IT/UL2* son almacenadas en un solo registro de 8 bits (ver tabla 3-9).

Las señales que interactúan en el bloque de control de la *IT/UL2* se describen en la tabla 3-9.

Señal	Origen	Destino	Descripción
LBT1_on	*C_ITUL2*	*LBT*	Señal activa en bajo que indica un lazo de retroalimentación entre la *I_Rx* y la *I_Tx*; por el lado del protocolo *TOPIA*.
LBT2_on	*C_ITUL2*	*LBT*	Señal activa en bajo que indica un lazo de retroalimentación entre las señales del protocolo *TOPIA*.
LBU1_on	*C_ITUL2*	*LBU*	Señal activa en bajo que indica un lazo de retroalimentación entre la *I_Rx* y la *I_Tx*; por el lado del protocolo *UTOPIA Level 2*.
LBU2_on	*C_ITUL2*	*LBU*	Señal activa en bajo que indica un lazo de retroalimentación entre las señales del protocolo *UTOPIA Level 2*.

<table>
<tr><td>PTR_on</td><td>C_ITUL2</td><td>ITR</td><td>Señal activa en bajo que es utilizada por el bloque ITR para ponerse en modo de prueba de la TR.</td></tr>
<tr><td>Inicia_PTR</td><td>C_ITUL2</td><td>ITR</td><td>Señal activa en bajo que indica el inicio de la prueba de la memoria en donde está almacenada la TR.</td></tr>
<tr><td>Listo_PTR</td><td>ITR</td><td>C_ITUL2</td><td>Señal activa en bajo que indica que la prueba a la memoria en donde está almacenada la TR ha finalizado.</td></tr>
<tr><td>Busy_PTR</td><td>ITR</td><td>C_ITUL2</td><td>Señal activa en bajo que indica que la prueba a la memoria en donde está almacenada la TR está en proceso.</td></tr>
<tr><td>Edo_Final_PTR</td><td>ITR</td><td>C_ITUL2</td><td>Señal que indica el resultado de la prueba a la memoria donde se encuentra almacenada la TR.
<table>
<tr><th>Edo_Final_PTR</th><th>Resultado</th></tr>
<tr><td>0</td><td>Favorable</td></tr>
<tr><td>1</td><td>Desfavorable</td></tr>
</table></td></tr>
<tr><td>Datos_CPLD[15-0]</td><td>I_8/16b</td><td>C_ITUL2</td><td>Bus bidireccional de datos de 16 bits. Este bus es manejado por las señales Read, Write y la señal ITUL2_CS.
En una operación de escritura la configuración empleada en los bits de datos es la que se muestran a continúación:
<table>
<tr><th>Bits</th><th>Señal de ITUL2</th></tr>
<tr><td>D0</td><td>LBU1_on</td></tr>
<tr><td>D1</td><td>LBU2_on</td></tr>
<tr><td>D2</td><td>LBT1_on</td></tr>
<tr><td>D3</td><td>LBT2_on</td></tr>
<tr><td>D4</td><td>PTR_on</td></tr>
<tr><td>D5</td><td>Inicia_PTR</td></tr>
<tr><td>D6</td><td>Velocidad[0]</td></tr>
<tr><td>D7</td><td>Velocidad[1]</td></tr>
<tr><td>D[8-15]</td><td>X</td></tr>
</table>
En una operación de lectura, la configuración empleada en los bits de datos es la que se muestra a continúación:
<table>
<tr><th>Bits</th><th>Señal de ITUL2</th></tr>
<tr><td>D0</td><td>Busy_PTR</td></tr>
<tr><td>D1</td><td>Listo_PTR</td></tr>
<tr><td>D2</td><td>Edo_Final_PTR</td></tr>
<tr><td>D[3-15]</td><td>X</td></tr>
</table></td></tr>
<tr><td>ITUL2_CS</td><td>I_8/16b</td><td>C_ITUL2</td><td>Señal activa en bajo manejada por la I8/16b. Un estado bajo en esta señal selecciona a la IT/UL2 para una operación de escritura o lectura.</td></tr>
</table>

Write	*I_8/16b*	*C_ITUL2*	Señal activa en bajo manejada por la *I8/16b*. Cuando la señal *Write* se activa (estado bajo) se realiza una operación de escritura si la señal *ITUL2_CS* está activa.
Read	*I_8/16b*	*C_ITUL2*	Señal activa en bajo manejada por la *I8/16b*. Cuando la señal *Read* se activa (estado bajo) se realiza una operación de lectura si la señal *ITUL2_CS* está activa
IR_ITUL2	*C_ITUL2*	*I_8/16b*	Señal activa en bajo manejada por la *IT/UL2*. Indica cuando este circuito finalizó algún proceso y está listo para reportar resultados.

Tabla 3-9 Señales que interactúan con el bloque de control de la interfaz *TOPIA/UTOPIA Level 2*

4 Interfaz 8/16 bits

La interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad (*IE1-T1/MC*) para realizar su función cuenta con circuitos que realizan tareas específicas como lo son: la interfaz de línea - *framers PEB-22554* [23], el circuito *SAR PXB-4220* [22] y la interfaz *TOPIA/UTOPIA Level 2* (*IT/UL2*), ver capítulo 3 de este documento. Estos circuitos específicos necesitan de una programación inicial (configuración) para realizar su función. También, estos circuitos son capaces de activar alarmas que notifican el estado en que se encuentra el sistema (*IE1-T1/MC*).

La forma en que se configuran estos circuitos es mediante un esquema de memoria de registros. Estos circuitos cuentan con un número finito de registros, estos pueden ser de actualización de funcionamiento o de reporte del estado del circuito. Todos estos están organizados uno tras otro asociándosele a cada uno de ellos una dirección específica. De este modo, la forma de acceder a alguno es mediante su dirección en el mapa de memoria de registros. Entonces, la filosofía para tener acceso a ellos es que este circuito específico se puede ver como una memoria externa a un microprocesador o microcontrolador, pero realizando funciones específicas.

Estos circuitos específicos (antes mencionados) poseen una interfaz con un microprocesador de bus *Intel* de 16 bits [22][23] no multiplexado (esto es, diferentes líneas para el bus de datos y para el bus de direcciones) para la comunicación. Por esta interfaz estos circuitos reciben y envían información. Además, estos circuitos cuentan con una o dos señales de interrupción que notifican que se tiene una alarma o que se acabó de realizar un proceso específico.

Para realizar estas tareas de programación y monitoreo del estado del sistema, el circuito de línea cuenta con una interfaz *RS-232* [25] hacia el exterior. Así, un administrador del sistema con acceso a esta interfaz puede cambiar el modo de operación de la *IE1-T1/MC* o simplemente monitorearla.

Por un lado se tienen cuatro circuitos específicos diferentes que cuentan con una interfaz con un microprocesador de bus *Intel* de 16 bits no multiplexado cada uno de ellos y por otro lado se tiene la interfaz *RS-232*. Entonces, es necesario un sistema intermedio que realice la función de interfaz entre estos dos protocolos de comunicación.

Se tuvieron dos opciones para el diseño de esta interfaz. Una de ellas fue el usar un microprocesador y la otra usar un microcontrolador. Se rechazó la primera opción debido a que el circuito de línea se localiza fuera de una *PC* y si se tuviera un microprocesador, se tendría que adicionar circuitos especiales (memoria *RAM*, *UART*, circuitería de reloj, etcétera) para su correcto funcionamiento; lo que encarece el circuito de línea además de hacerlo más complejo. Entonces se aceptó el usar un microcontrolador ya que este cuenta internamente con estos circuitos especiales además de su costo accesible.

Primeramente, la intención fue utilizar un microcontrolador que tuviera una interfaz de bus *Intel* de 16 bits no multiplexado, capaz de manejar un bus de direcciones lo suficientemente grande para poder tener mapeados los cuatro circuitos específicos (aproximadamente 1 millón de localidades; o sea, 20 líneas de direcciones). Además, que contara con una interfaz *RS-232* y una memoria *EEPROM* o *FLASH* para el almacenamiento del programa del microcontrolador. En la industria se cuenta con algunos microcontroladores que cubren estas características pero se encontró otra solución.

Se tomó la decisión de emplear un microcontrolador sencillo que contara con una interfaz *RS-232*, con una memoria *EEPROM* o *FLASH* interna para el almacenamiento del programa de este (eliminando la necesidad de memoria externa) y con suficientes líneas para realizar una interfaz de 8 bits de datos. Este microcontrolador se conecta por medio de esta interfaz de 8 bits y algunas líneas de control hacia una interfaz (interfaz 8/16 bits) que convierta el flujo de 8 bits de datos al flujo de 16 bits de datos que manejan los circuitos específicos del circuito de línea.

El microcontrolador escogido fue el *AT89S8252* del fabricante *ATMEL* [25], debido a la arquitectura sencilla (arquitectura *8051*), bajo precio y disponibilidad en el mercado.

4.1. Arquitectura de la interfaz 8/16 bits

La interfaz 8/16 bits (*I8/16b*) como su nombre lo indica, es la encargada de recibir información (mediante un protocolo) del microcontrolador en un formato de 8 bits de datos y enviar esta información a cualquiera de los cuatro circuitos específicos (en un formato de 16 bits de datos) que se tienen en la *IE1-T1/MC*. A la vez, esta información realiza el mismo flujo de información pero en sentido inverso; esto es, recibe información con un formato de 16 bits de datos de alguno de los cuatro circuitos específicos y envía esta información hacia el microcontrolador con un formato de 8 bits de datos.

El microcontrolador será el encargado de recibir de la *PC* o enviar hacia esta, la información con el formato del protocolo *RS-232*.

La *I8/16b* por el lado del microcontrolador cuenta (en dirección del microcontrolador hacia la *I8/16b*) con un bus de 8 bits (*Data_Tx[7-0]*) y dos señales de control (*Busy_Tx* y *Peticion_Tx*). Y tiene (en dirección de la *I8/16b* hacia el microcontrolador) un bus de 8 bits (*Data_Rx[7-0]*) y dos señales de control (*Busy_Rx* y *Peticion_Rx*). Ver figura 4-1.

Esta misma interfaz por el lado de los circuitos específicos cuenta con un bus de direcciones de 18 bits (*Dir[17-0]*), un bus de datos bidireccional de 16 bits (*Datos_CPLD[0-15]*) y con señales de control en las cuales están incluidas: las señales básicas para el control de bus de lectura y escritura (*Read* y *Write*), las señales de selección del circuito específico con el que se está en comunicación (*ITUL2_CS*, *IWE8_CS*, *PEB1_CS* y *PEB2_CS*) y las señales de interrupción de los circuitos específicos (*IR_ITUL2*, *MPIR1_IWE8*, *MPIR2_IWE8*, *INT_PEB1* y *INT_PEB2*).

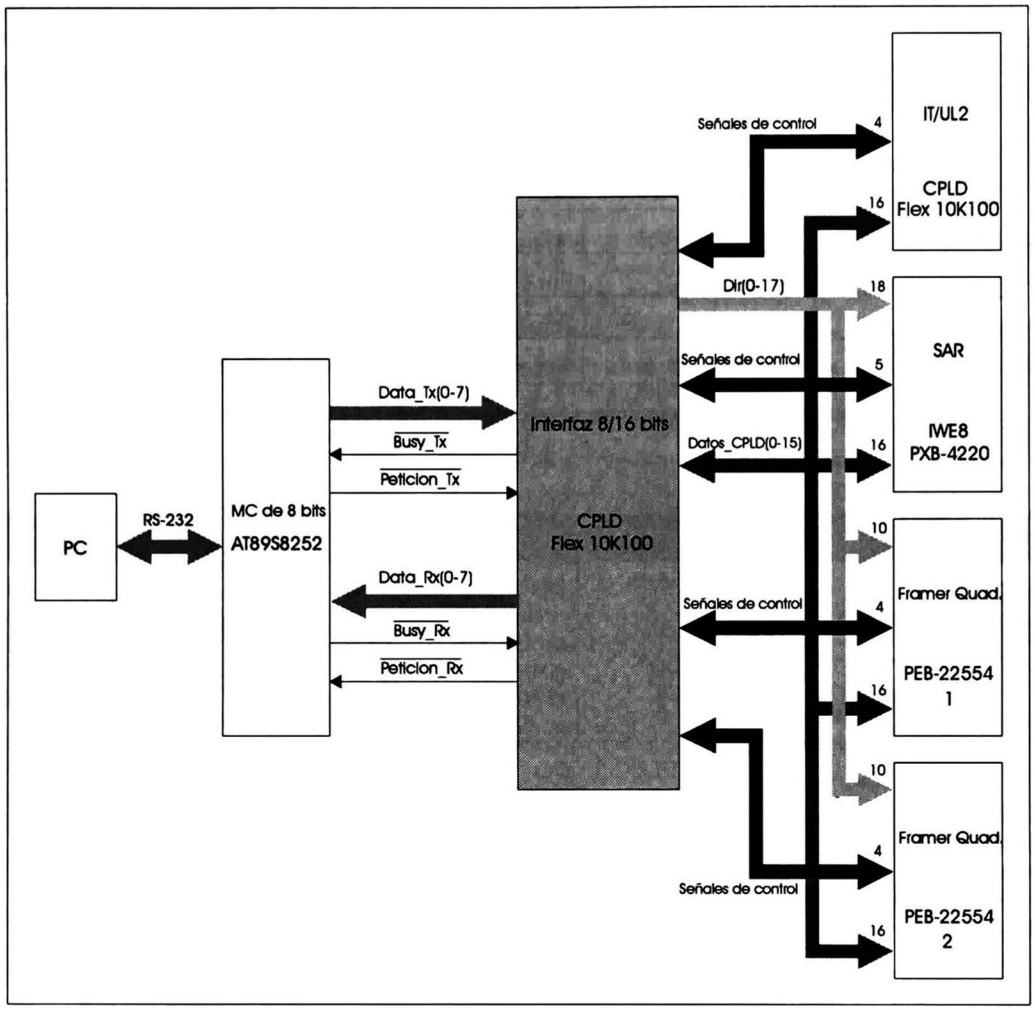


Figura 4-1 Interfaz 8/16 bits

La *I8/16b* fue diseñada en código *VHDL* [8][9][11][12] para ser sintetizada dentro del chip *CPLD Flex 10K100* [24] utilizado para la *IT/UL2*. El agregar el diseño de esta interfaz dentro del chip *Flex 10K100* se facilitó debido a que no se empleó toda su capacidad en la *IT/UL2*, dando la ventaja de no utilizar otro chip *CPLD* para la *I8/16b*.

En las tablas 4-1 y 4-2 se describen cada una de las señales empleadas entre el microcontrolador *AT89S8252* y la *I8/16b*, también son descritas las señales empleadas entre la *I8/16b* y los 4 circuitos específicos, respectivamente.

Señal	Origen	Destino	Descripción
Data_Tx[7-0]	*MC*	*I 8/16b*	Bus de datos de 8 bits.
Busy_Tx	*I 8/16b*	*MC*	Señal activa en bajo manejada por la *I8/16b*.
Peticion_Tx	*MC*	*I 8/16b*	Señal activa en bajo manejada por el microcontrolador.
Data_Rx[7-0]	*I 8/16b*	*MC*	Bus de datos de 8 bits
Busy_Rx	*MC*	*I 8/16b*	Señal activa en bajo manejada por el microcontrolador.
Peticion_Rx	*I 8/16b*	*MC*	Señal activa en bajo manejada por la *I8/16b*.

Tabla 4-10 Señales que interactúan entre el microcontrolador *AT89S8252* y la interfaz 8/16 bits

Señal	Origen	Destino	Descripción
Dir[17-0]	*I 8/16b*	*IWE8, PEB1, PEB2*	Bus de direcciones para el manejo de los registros de los circuitos específicos. El circuito *SAR* (*IWE8*) maneja las 18 líneas de este bus (*Dir[17-0]*). El circuito *PEB1* y *PEB2* manejan sólo 10 líneas de este bus (*Dir[9-0]*). Nota: La *IT/UL2*, no maneja ninguna línea de este bus puesto que solo cuenta con un solo registro de operación.
Datos_CPLD[15-0]	*I 8/16b* *ITUL2, IWE8, PEB1, PEB2*	*ITUL2, IWE8, PEB1, PEB2* *I 8/16b*	Bus bidireccional de datos de 16 bits. Este bus es manejado por las señales *Read*, *Write* y las señales de *CS* (*ITUL2_CS, IWE8_CS, PEB1_CS y PEB1_CS*)
Write	*I 8/16b*	*ITUL2, IWE8, PEB1, PEB2*	Señal activa en bajo manejada por la *I8/16b*. Cuando la señal *Write* se activa (estado bajo) se realiza una operación de escritura al circuito cuya señal *CS* (*ITUL2_CS*, *IWE8_CS*, *PEB1_CS* y *PEB2_CS)* esté activa. El registro que modifica su contenido por el valor del bus de datos (*Datos_CPLD[15-0]*) es el que concuerde con la dirección del bus de direcciones (*Dir[17-0]*).
Read	*I 8/16b*	*ITUL2, IWE8, PEB1, PEB2*	Señal activa en bajo manejada por la *I8/16b*. Cuando la señal *Read* se activa (estado bajo) se realiza una operación de lectura al circuito cuya señal *CS* (*ITUL2_CS*, *IWE8_CS*, *PEB1_CS* y *PEB2_CS)* esté activa. El registro del cual se obtendrá su contenido en el bus de datos (*Datos_CPLD[15-0]*) es el que concuerde con la dirección del bus de direcciones (*Dir[17-0]*).

ITUL2_CS	*I8/16b*	*ITUL2*	Señal activa en bajo manejada por la *I8/16b*. Un estado bajo en esta señal selecciona al circuito *IT/UL2* para una operación de escritura o lectura a alguno de sus registros.
IWE8_CS	*I8/16b*	*IWE8*	Señal activa en bajo manejada por la *I8/16b*. Un estado bajo en esta señal selecciona al circuito *SAR* para una operación de escritura o lectura a alguno de sus registros.
PEB1_CS	*I8/16b*	*PEB1*	Señal activa en bajo manejada por la interfaz 8/16 bits. Un estado bajo en esta señal selecciona al circuito *PEB-22554_1*, para una operación de escritura o lectura a alguno de sus registros.
PEB2_CS	*I8/16b*	*PEB2*	Señal activa en bajo manejada por la *I8/16b*. Un estado bajo en esta señal selecciona al circuito *PEB-22554_2* para una operación de escritura o lectura a alguno de sus registros.
MPRDY	*IWE8*	*I8/16b*	Señal activa en alto manejada por el circuito *SAR*. Indica cuando el dispositivo *SAR* ha finalizado con la operación de lectura o escritura.
IR_ITUL2	*ITUL2*	*I8/16b*	Señal activa en bajo manejada por la *IT/UL2*. Indica cuando este circuito finalizó algún proceso y está listo para reportar resultados.
MPIR1_IWE8	*IWE8*	*I8/16b*	Señal activa en bajo manejada por el circuito *SAR*. Indica cuando este circuito finalizó algún proceso y está listo para reportar resultados o cuando reporta alguna alarma ocurrida en el sistema.
MPIR2_IWE8	*IWE8*	*I8/16b*	Señal activa en bajo manejada por el circuito *SAR*. Indica cuando este circuito finalizó algún proceso y está listo para reportar resultados, o cuando reporta alguna alarma ocurrida en el sistema. Esta es una señal auxiliar de interrupción del dispositivo *SAR*.
INT_PEB1	*PEB1*	*I8/16b*	Señal activa en bajo manejada por el circuito *PEB-22554_1*. Indica cuando este circuito finalizó algún proceso y está listo para reportar resultados o cuando reporta alguna alarma ocurrida en el sistema.
INT_PEB2	*PEB2*	*I8/16b*	Señal activa en bajo manejada por el circuito *PEB-22554_2*. Indica cuando este circuito finalizó algún proceso y está listo para reportar resultados o cuando reporta alguna alarma ocurrida en el sistema.

Tabla 4-11 Señales que interactúan entre la interfaz 8/16 bits y los 4 circuitos específicos

4.2. Protocolo de comunicación entre el microcontrolador *AT89S9252* y la interfaz 8/16 bits

Para la comunicación entre el microcontrolador *AT89S8252* y la *I8/16b* se diseñó un protocolo de comunicación que cuenta con un bus de datos de 8 bits y dos señales de control. El mismo se divide en dos dependiendo del flujo de información.

El protocolo empleado es idéntico en cuanto al manejo de las señales empleadas; esto es, la funcionalidad de las señales es la misma en ambos flujos solo que en sentido contrario. En lo único que difieren es en el número de bytes que se transmiten de un lugar a otro.

4.2.1. Protocolo con flujo de información de microcontrolador *AT89S8252* hacia la interfaz 8/16 bits

En este protocolo de comunicación se envían 5 bytes de información del microcontrolador *AT89S8252* hacia la *I8/16b*. En estos va contenida información que le indica a la *I8/16b* primeramente si es una operación de lectura o escritura de un registro, que circuito específico es el que realizará esta operación, la dirección del registro de este circuito específico y el dato a ser escrito en este registro en caso de ser una operación de escritura. En la tabla 4-3 y 4-4 se describe la información contenida en cada uno de estos 5 bytes.

No. Byte	MSB Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB bit 0
B1	R/W	O2	O1	X	X	X	A17	A16
B2	A15	A14	A13	A12	A11	A10	A9	A8
B3	A7	A6	A5	A4	A3	A2	A1	A0
B4	D15	D14	D13	D12	D11	D10	D9	D8
B5	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 4-12 Información de los 5 bytes del protocolo microcontrolador *AT89S8252* hacia la interfaz 8/16 bits

<table>
<tr><th>Símbolo</th><th>Descripción</th></tr>
<tr><td>B[1-5]</td><td>Indica el número de byte del flujo de información (ejemplo, B4 es el cuarto byte en el flujo de información).</td></tr>
<tr><td>R/W</td><td>Este bit indica si es una operación de lectura (R) o de escritura (W).
<table>
<tr><th>R/W</th><th>Acción</th></tr>
<tr><td>0</td><td>Escritura</td></tr>
<tr><td>1</td><td>Lectura</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>O[2-1]</td><td>Estos bits indican el circuito específico al cual se hará una lectura o escritura.
<table>
<tr><th>O2</th><th>O1</th><th>Circuito específico</th></tr>
<tr><td>0</td><td>0</td><td>Interfaz TOPIA/UTOPIA Level 2</td></tr>
<tr><td>0</td><td>1</td><td>Circuito SAR (IWE8)</td></tr>
<tr><td>1</td><td>0</td><td>Framer Quad. PEB-22554 1</td></tr>
<tr><td>1</td><td>1</td><td>Framer Quad. PEB-22554 2</td></tr>
</table></td></tr>
<tr><td>A[0-17]</td><td>Estos bits indican la dirección del registro específico al cual se hará una operación de lectura o escritura.
<table>
<tr><th>Dispositivo</th><th>Bits empleados</th></tr>
<tr><td>Interfaz TOPIA/UTOPIA Level 2</td><td>No usado ninguno</td></tr>
<tr><td>Circuito SAR (IWE8)</td><td>A[0-17]</td></tr>
<tr><td>Framer Quad. PEB-22554 1</td><td>A[0-9]</td></tr>
<tr><td>Framer Quad. PEB-22554 2</td><td>A[0-9]</td></tr>
</table></td></tr>
<tr><td>D[0-15]</td><td>Estos bits son el dato que se pondrá en el registro específico, en una operación de escritura.
<table>
<tr><th>Dispositivo</th><th>Bits empleados</th></tr>
<tr><td>Interfaz TOPIA/UTOPIA Level 2</td><td>D[0-7] *Ver Nota</td></tr>
<tr><td>Circuito SAR (IWE8)</td><td>D[0-15]</td></tr>
<tr><td>Framer Quad. PEB-22554 1</td><td>D[0-15]</td></tr>
<tr><td>Framer Quad. PEB-22554 2</td><td>D[0-15]</td></tr>
</table>
*Nota: Este byte es empleado por la IT/UL2 para el manejo de las señales de modo de operación de esta interfaz.
<table>
<tr><th>Bits</th><th>Señal de la ITUL2</th></tr>
<tr><td>D0</td><td>LBU1_on</td></tr>
<tr><td>D1</td><td>LBU2_on</td></tr>
<tr><td>D2</td><td>LBT1_on</td></tr>
<tr><td>D3</td><td>LBT2_on</td></tr>
<tr><td>D4</td><td>PTR_on</td></tr>
<tr><td>D5</td><td>Inicia_PTR</td></tr>
<tr><td>D6</td><td>Velocidad[0]</td></tr>
<tr><td>D7</td><td>Velocidad[1]</td></tr>
<tr><td>D[8-15]</td><td>X</td></tr>
</table></td></tr>
</table>

Tabla 4-13 Descripción de los símbolos empleados en la tabla 4-3

4.2.1.1. *Procedimiento empleado por el protocolo de transmisión de información del microcontrolador AT89S8252 hacia la interfaz 8/16 bits*

A continuación se describe el procedimiento para el protocolo de transmisión de información del microcontrolador *AT89S8252* hacia la *I8/16b*. Cada uno de los puntos de este están numerados, permitiendo así su localización en la figura 4-2.

1. El microcontrolador *AT89S8252* pone el dato del byte en *Data_Tx*.
2. El microcontrolador *AT89S8252* pone la señal *Peticion_Tx* en estado bajo.
3. La *I8/16b* detecta que la señal *Peticion_Tx* se encuentra en estado bajo y toma el byte del bus de datos *Data_Tx*. Así mismo, pone la señal *Busy_Tx* en estado bajo para indicarle al microcontrolador *AT89S8252* que ya se tomó el byte del bus de datos *Data_Tx*.
4. El microcontrolador *AT89S8252* detecta que la señal *Busy_Tx* se encuentra en estado bajo indicándole que la *I8/16b* ya tomó el byte del bus *Data_Tx*. Enseguida, el microcontrolador *AT89S8252* pone la señal *Peticion_Tx* en estado alto.
5. La *I8/16b* detecta que la señal *Peticion_Tx* está en estado alto. Enseguida, la *I8/16b* pone la señal *Busy_Tx* en estado alto.

Y así se repiten estos pasos hasta que se transmiten los 5 bytes de datos; esto es, el paso 6 es el mismo que el 1, el paso 7 es el mismo que el 2 y así sucesivamente.

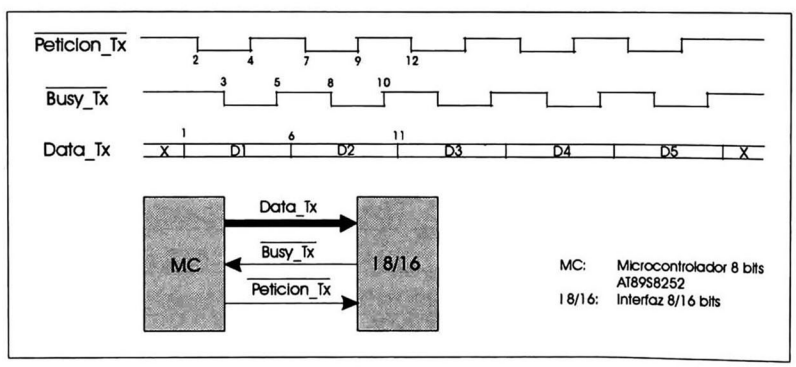


Figura 4-2 Protocolo de transmisión de información del microcontrolador *AT89S8252* hacia la interfaz 8/16 bits

4.2.2. Protocolo con flujo de información de la interfaz 8/16 bits hacia el microcontrolador *AT89S8252*

En este protocolo de comunicación se envían 3 bytes de información de la *I8/16b* hacia el microcontrolador *AT89S8252*. En estos va contenida información que es el resultado de una operación de lectura de un circuito específico. También, estos bytes son empleados para indicar que está presente alguna interrupción en alguno de estos circuitos específicos, dando a conocer cual de estos 4 es el que la tiene activa. En caso de ser una interrupción, el administrador del sistema recibirá esta notificación y deberá proceder con una operación de lectura (al circuito específico que tiene activa esta), para conocer la alarma ocurrida o percatarse si finalizó algún proceso del circuito específico. En la tabla 4-5 y 4-6 se describe la información contenida en cada uno de estos 3 bytes.

No. Byte	MSB Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB bit 0
B1	R/I	O2	O1	X	X	X	X	X
B2	D15	D14	D13	D12	D11	D10	D9	D8
B3	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 4-14 Información de los 3 bytes del protocolo de la interfaz 8/16 bits hacia el microcontrolador *AT89S8252*

<table>
<tr><th>Símbolo</th><th>Descripción</th></tr>
<tr><td>B[1-3]</td><td>Indica el número de byte del flujo de información (ejemplo, B2 es el segundo byte en el flujo de información).</td></tr>
<tr><td>R/I</td><td>Este bit indica si el dato contenido en estos bytes es el resultado de una operación de lectura (R) o si ocurrió alguna interrupción en cualquiera de los 4 circuitos específicos.
<table>
<tr><th>R/I</th><th>Acción</th></tr>
<tr><td>0</td><td>Ocurrió interrupción</td></tr>
<tr><td>1</td><td>Resultado de lectura</td></tr>
</table></td></tr>
<tr><td>O[2-1]</td><td>Estos bits indican el circuito específico que tiene activa alguna interrupción.
*Nota: En un proceso de lectura los bits O2 y O1 no tienen sentido, ya que el microcontrolador AT89S8252 está esperando el resultado de la operación de lectura que este mismo inició.
<table>
<tr><th>O2</th><th>O1</th><th>Circuito específico</th></tr>
<tr><td>0</td><td>0</td><td>Interfaz TOPIA/UTOPIA Level 2</td></tr>
<tr><td>0</td><td>1</td><td>Circuito SAR (IWE8)</td></tr>
<tr><td>1</td><td>0</td><td>Framer Quad. PEB-22554 1</td></tr>
<tr><td>1</td><td>1</td><td>Framer Quad. PEB-22554 2</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>D[0-15]</td><td>Estos bits son el dato que es el resultado de la operación de lectura. Estos bits no tienen ningún sentido cuando se está notificando que algún circuito específico cuenta con una interrupción.
<table>
<tr><th>Dispositivo</th><th>Bits empleados</th></tr>
<tr><td>Interfaz TOPIA/UTOPIA Level 2</td><td>D[0-2] *Ver Nota</td></tr>
<tr><td>Circuito SAR (IWE8)</td><td>D[0-15]</td></tr>
<tr><td>Framer Quad. PEB-22554 1</td><td>D[0-15]</td></tr>
<tr><td>Framer Quad. PEB-22554 2</td><td>D[0-15]</td></tr>
<tr><td colspan="2">*Nota: Estos bits son empleados por la IT/UL2 para el reporte del estado de la "prueba de tabla de enrutado (PTR)", mediante señales específicas.
<table>
<tr><th>Bits</th><th>Señal de IT/UL2</th></tr>
<tr><td>D0</td><td>Busy_PTR</td></tr>
<tr><td>D1</td><td>Listo_PTR</td></tr>
<tr><td>D2</td><td>Edo_Final_PTR</td></tr>
<tr><td>D[3-15]</td><td>X</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Tabla 4-15 Descripción de los símbolos empleados en la tabla 4-5

4.2.2.1. Procedimiento empleado por el protocolo de transmisión de información de la interfaz 8/16 bits hacia el microcontrolador AT89S8252

A continuación se describe el procedimiento empleado por el protocolo de transmisión de información de la *I8/16b* hacia el microcontrolador *AT89S8252*. Cada uno de los puntos de este están numerados permitiendo así su localización en la figura 4-3.

1. La *I8/16b* pone el dato del byte en *Data_Rx*.
2. La *I8/16b* pone la señal *Peticion_Rx* en estado bajo.
3. El microcontrolador *AT89S8252* detecta que la señal *Peticion_Rx* se encuentra en estado bajo y toma el byte del bus de datos *Data_Rx*. Así mismo, pone la señal *Busy_Rx* en estado bajo para indicarle a la *I8/16b* que ya se tomó el byte del bus de datos *Data_Rx*.
4. La *I8/16b* detecta que la señal *Busy_Rx* se encuentra en estado bajo indicándole que el microcontrolador *AT89S8252* ya tomó el byte del bus *Data_Rx*. Enseguida, la *I8/16b* pone la señal *Peticion_Rx* en estado alto.
5. El microcontrolador *AT89S8252* detecta que la señal *Peticion_Rx* está en estado alto. Enseguida, el microcontrolador *AT89S8252* pone la señal *Busy_Rx* en estado alto.

Y así se repiten estos pasos, hasta que se transmiten los 3 bytes de datos; esto es, el paso 6 es el mismo que el 1, el paso 7 es el mismo que el 2 y así sucesivamente.

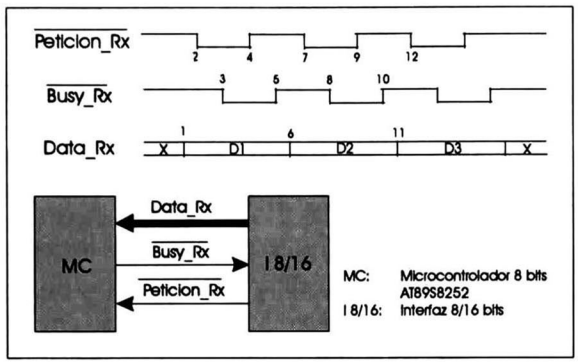


Figura 4-3 Protocolo de transmisión de información de la interfaz 8/16 bits hacia el microcontrolador *AT89S8252*

5 Pruebas y su implementación

El trabajo de esta tesis se enfocó además del diseño de la interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad (*IE1-T1/MC*); principalmente en el diseño e implementación de la interfaz entre el protocolo de comunicación *TOPIA* y el protocolo de comunicación *UTOPIA Level 2* (*IT/UL2*).

En el capítulo 3 de este documento, se explica en detalles el diseño de cada uno de los bloques que intervienen en *IT/UL2*. También se describió en ese capítulo el diseño de dos bloques importantes que son el bloque de *loopback* del lado del protocolo *TOPIA* (*LBT*) y el bloque de *loopback* del lado del protocolo *UTOPIA Level 2* (*LBU*); ambos diseñados para la prueba de los bloques: interfaz de recepción (*I_Rx*) e interfaz de transmisión(*I_Tx*).

Primeramente, la metodología de prueba empleada siguió la filosofía de primero simular (código *VHDL* de cada uno de los bloques diseñados) el correcto funcionamiento del código que forma los bloques de hardware que están en el nivel más bajo (*bottom level*) del diseño. Enseguida, la simulación del código de los bloques que están en el siguiente nivel (un nivel más arriba del *bottom level*), y así sucesivamente hasta llegar al nivel más alto del diseño (*top level*).

Estas simulaciones de código *VHDL*, fueron desarrolladas en la herramienta de software *Max2Win* distribuido por la compañía *ALTERA* [24]. Este software tiene la característica de hacer la simulación después de realizar la síntesis del código *VHDL*, esto significa que la simulación contará con los tiempos de retardo de las señales empleadas dentro del *FPGA*.

Después de realizadas estas simulaciones y habiendo obtenido los resultados satisfactorios de estas, se realizó también el diseño e implementación de una arquitectura de prueba (una "*cama de prueba*", ver [10]) empleada para comprobar el correcto funcionamiento de *IT/UL2*.

Todos estos diseños fueron sintetizados ("*bajados*") en el *FPGA Flex10K100* para la prueba física en *PCB* (*Printed Circuit Board*, tarjeta de circuito impreso). El *PCB* empleado para estas pruebas físicas fue el desarrollado en [3]. Esto ayuda a comprobar (además de las simulaciones realizadas) el correcto funcionamiento de las interfaces diseñadas.

Finalmente, se mencionan algunos resultados de espacio empleado en el *FPGA Flex10K100*, para la implementación de *IT/UL2*, *I8/16b* y *IT/UL2* con la *cama de prueba*.

5.1. Pruebas de la interfaz *TOPIA/UTOPIA Level 2*

Como se mencionó anteriormente, la *IT/UL2* cuenta con dos bloques extras a su función principal (la conversión del protocolo *TOPIA* al protocolo *UTOPIA Level 2* y viceversa). Estos dos bloques son *LBT* y *LBU*, que permiten a la *IT/UL2* hacer lazos de retroalimentación de la información. Estos lazos ayudan a los circuitos externos de la *IT/UL2* a hacer un diagnóstico de esta interfaz; esto es, detectar si la *IT/UL2* está realizando adecuadamente la conversión de protocolos.

La forma en que estos circuitos externos a la *IT/UL2* detectan si está realizando o no la conversión adecuada de protocolos, es poniendo a esta interfaz en modo de prueba (ver capítulo 3) para después activar los diferentes "*loopbacks*" con los que esta cuenta.

Una vez activado el "*loopback*" que se empleará en la prueba, el circuito externo a la *IT/UL2* alimenta celdas *ATM* con formato de 56 bytes por el lado de *TOPIA*, o con formato de 53 bytes por el lado de *UTOPIA Level 2* dependiendo el "*loopback*" activado. Estas celdas alimentadas a la interfaz se regresarán al circuito externo, permitiendo a este hacer una comparación de la celda *ATM* enviada con la celda *ATM* recibida.

Existen dos diferentes posibilidades de retroalimentación de la información dentro de la *IT/UL2*:

- La primera es inmediata a la salida del circuito externo que está conectado a esta interfaz. Esto es, la información no cambia de protocolo (del protocolo *TOPIA* al protocolo *UTOPIA Level 2* o viceversa), sino que es inmediatamente enviada al circuito externo del cual proviene. La función principal de esta retroalimentación (este *loopback*) es la de prueba de las conexiones entre la *IT/UL2* y el circuito externo.

  La segunda está hasta el extremo contrario del que se encuentra el circuito externo conectado a la *IT/UL2*. Esto es, la información cambia en los dos diferentes protocolos.

  Si el circuito externo está conectado a la interfaz por el lado del protocolo *TOPIA*, la información pasa por medio de la interfaz de transmisión (convirtiéndose del protocolo *TOPIA* a *UTOPIA Level 2*). Enseguida, la información se regresa pasando por la interfaz de recepción (convirtiéndose del protocolo *UTOPIA Level* 2 a *TOPIA* de nuevo) para después entregarla al circuito externo.

  Si el circuito externo está conectado a la interfaz por el lado del protocolo *UTOPIA Level* 2, la información pasa por medio de la interfaz de recepción (convirtiéndose del protocolo *UTOPIA Level* 2 a *TOPIA*). Entonces, la información regresa al circuito externo, convirtiéndose del protocolo *TOPIA* a *UTOPIA Level 2* (de nuevo) al pasar por la interfaz de transmisión.

  La función de esta retroalimentación es la prueba de la conversión de protocolos en ambas direcciones.

Estas dos diferentes retroalimentaciones permiten que el circuito externo a la *IT/UL2* (después de hecha la comparación de las celdas enviadas con las celdas recibidas), determine si la *IT/UL2* está realizando adecuadamente la conversión de protocolos y que además, no existan problemas con la conexión entre este circuito externo y la interfaz.

Para la prueba física de la *IT/UL2* se siguió con la metodología de pruebas explicada anteriormente. Por el lado del protocolo *TOPIA* se empleó como circuito externo a la *MC*, configurándola (programándola) de tal forma que envíe celdas *ATM* con formato *TOPIA* (conocido perfectamente el contenido de los 56 bytes de estas celdas); para después de ser recibidas estas celdas (provenientes de la *IT/UL2* "al dar la vuelta"), sean comparadas con las celdas generadas por la *MC* (para mayores detalles acerca de la programación de esta prueba, ver [5]).

Por el lado del protocolo *UTOPIA Level 2* se diseñó una "*cama de prueba*" basada en la filosofía de "*on-line BIST*". Esto es, se diseñó un circuito generador de celdas, un circuito analizador de celdas y un control automático de activación de los diferentes *loopbacks* de la *IT/UL2*; empleando a la *IT/UL2* como el circuito bajo prueba (ver figura 3-13). La activación y monitoreo de esta prueba puede ser hecha por una *PC*, ya que las señales de control y monitoreo de esta prueba pueden ser manejadas por el puerto paralelo de alguna *PC* (para mayor detalle de la programación del manejador del puerto paralelo así como de la interfaz de software, ver [5]).

5.1.1. Prueba física de la Interfaz TOPIA/UTOPIA Level 2 por el lado del protocolo TOPIA

En esta prueba se emplean dos diferentes "*loopbacks*" contenidos en la *IT/UL2*. El primero de ellos (*LB1*) es utilizado para cerciorarse de que en las conexiones entre la *MC* y la *IT/UL2* no existe ningún problema (corto circuito o circuito abierto) En esta primera prueba (*LB1*) las celdas enviadas por la *MC* no sufren ninguna modificación en su contenido, de esta forma la *MC* sólo tiene que hacer una comparación "*uno a uno*" de cada uno de los bytes de las celdas *ATM* enviadas con las celdas recibidas por esta.

El segundo "*loopback*" (*LB2*) sirve para comprobar que la *IT/UL2* está realizando bien la conversión de protocolos de comunicación. Primeramente de *TOPIA a UTOPIA Level 2* por la interfaz de transmisión y posteriormente de *UTOPIA Level 2 a TOPIA* por la interfaz de recepción. En la figura 5-1, se muestran estos dos diferentes "*loopbacks*", así como el sentido del flujo de la información de esta prueba.

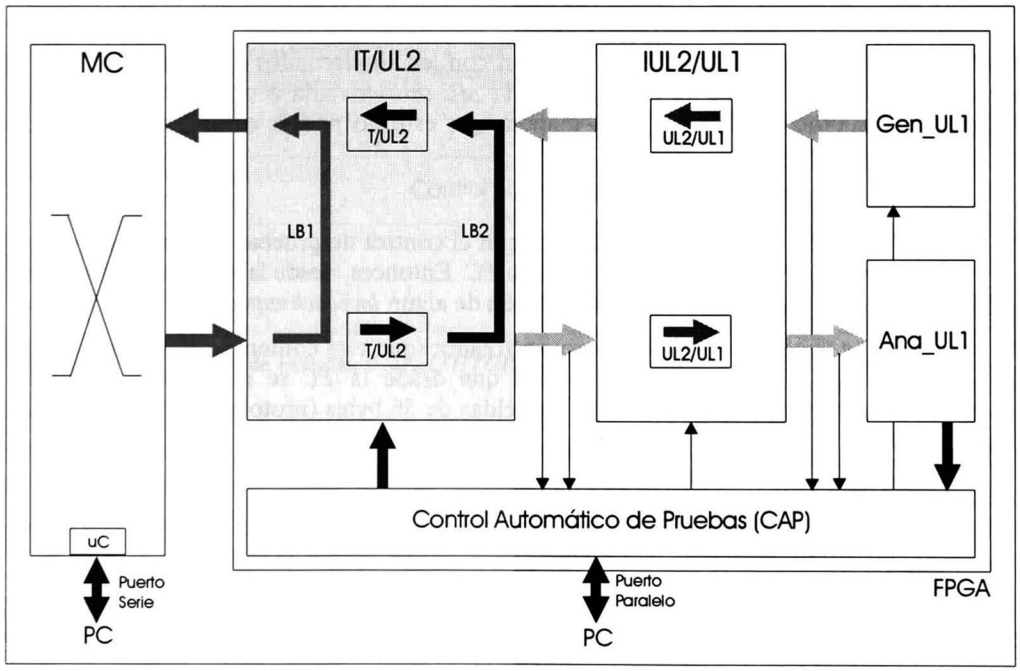


Figura 5-1 Prueba física de la interfaz *TOPIA/UTOPIA Level 2* por el lado del protocolo *TOPIA*

Cuando se realiza la prueba con el segundo *loopback* (*LB2*), es necesario que la *MC* envíe primeramente una celda de actualización de la *TR* (tabla de enrutado); ya que por características propias de la *MC*, el microcontrolador contenido en este (que es el encargado del modo de prueba de la *MC*) extraerá la celda de un puerto diferente al puerto en el que está conectada la *IT/UL2* (para detalles de este modo de prueba, ver [5]).

Entonces la primera celda *ATM* de 56 bytes enviada a la *IT/UL2*, debe ser una celda especial de actualización de la *TR*. Esta tendrá un determinado par *VPI* y *VCI*, relacionados a los bytes *R1b* y *R2b* en donde se asigna el puerto específico de la *MC* (ver figura 3-3) al cual serán dirigidas de regreso las celdas enviadas por la *MC*. Este puerto denotado por los bytes *R1b* y *R2b*, debe estar conectado (internamente) al microcontrolador de la *MC*, para que este reciba las celdas que serán comparadas con las enviadas anteriormente.

Las celdas que la *MC* envíe hacia la *IT/UL2* después de que se envió la celda especial de actualización de la *TR*, deberán ser "normales"; esto es, que se conviertan del protocolo *TOPIA* a *UTOPIA Level 2*, que pasen por el *LB2* para que finalmente se conviertan del protocolo *UTOPIA Level 2* a *TOPIA* y así ser entregadas de vuelta a la *MC* para su posterior comparación por medio del microcontrolador.

Es importante mencionar que estas celdas "normales", deben de tener el mismo par *VPI* y *VCI* (en el encabezado de la celda *ATM*) que el de la especial de actualización de la *TR*; para cuando esta pase del protocolo *UTOPIA Level 2* a *TOPIA*, se le sea asignado el puerto de la *MC* (bytes *R1a* y *R2a* de la figura 3-3) al cual está conectado el microcontrolador.

En caso de que los bytes de enrutado (*R1a* y *R2a*) correspondientes al par *VPI* y *VCI* del encabezado de la celda "normal" no coincidan con los almacenados anteriormente en la *TR* por la especial; la celda que va de regreso a la *MC* será enviada a un puerto que no estará conectado al microcontrolador, de tal manera que este no podría comparar ya que nunca recibiría la celda de regreso.

Por un lado, se tiene conectada a la *IT/UL2* con el control de pruebas automáticas, que a su vez está conectado por el puerto paralelo a una *PC*. Entonces, desde la *PC* se controla el modo de funcionamiento (modo normal o la activación de algún *loopback* específico) de la *IT/UL2*.

Por otro lado, se tiene conectado el microcontrolador (que está contenido en la *MC*) a una *PC* por medio del puerto serie. De tal manera que desde la *PC* se controla y supervisa la generación, recepción y comparación de las celdas de 56 bytes (protocolo *TOPIA*) empleadas en esta prueba.

Entonces, es posible tener una *PC* empleando el puerto serie y el paralelo, de tal forma que se tenga el control y supervisión de esta prueba por completo dentro de esta.

5.1.2. Prueba física de la Interfaz *TOPIA/UTOPIA Level 2* por el lado del protocolo *UTOPIA Level 2*

Para la prueba física de la *IT/UL2* por el lado del protocolo *UTOPIA Level 2*, se diseñó una "*cama de prueba*" basada en la filosofía de "*on-line BIST*". La función de esta es alimentar celdas *ATM* a la *IT/UL2* por el lado de *UTOPIA Level 2* activado los "*loopbacks*" correspondientes, de tal manera que sean probadas primeramente con el *LB2* las líneas de conexión entre la *IT/UL2* (capa *ATM* del protocolo *UTOPIA Level 2*, ver [14]) y el circuito conectado a esta interfaz (capa *PHY* de *UTOPIA Level 2*). Enseguida (si la prueba con *LB2* es favorable) con el "*loopback*" *LB3* se probarán las conversiones de protocolo de *UTOPIA Level 2* al protocolo *TOPIA* y viceversa. En la figura 5-2, se muestra el sentido del flujo de la información para esta prueba.

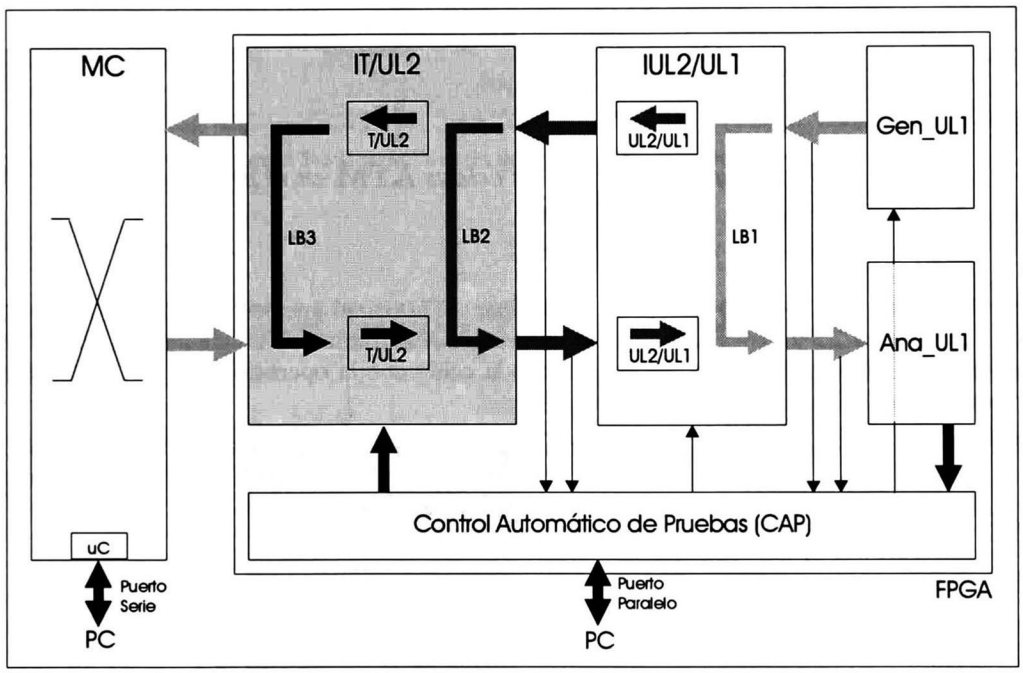


Figura 5-2 Prueba física de interfaz *TOPIA/UTOPIA Level 2* por el lado del *protocolo UTOPIA Level 2*

Para la realización de las pruebas de estos dos *loopbacks* (*LB2* y *LB3*) fue necesario el diseño de los bloques: generador de celdas *ATM* en protocolo *UTOPIA Level 1* (*Gen_UL1*), analizador de celdas *ATM* en protocolo *UTOPIA Level 1* (*Ana_UL1*) y control automático de pruebas (*CAP*). Estos bloques siguen la filosofía de prueba empleada en "*on-line BIST*" [17][19]. Los bloques *Gen_UL1* y *Ana_UL1* emplean el protocolo *UTOPIA Level 1*, debido a que se diseñó en [5] una interfaz entre el protocolo de comunicación *UTOPIA Level 2* y *Level 1* (*IUL2/UL1*) y se decidió emplearla en esta *cama de prueba* con el fin de prueba de esta misma empleando *LB1*.

5.1.2.1. *El bloque generador de celdas* ATM *en el protocolo* UTOPIA Level 1 *(Gen_UL1)*

Este bloque se encarga de generar celdas *ATM* con el protocolo *UTOPIA Level 1* y entregarlas a la *IUL2/UL1*. Las celdas *ATM* generadas por este bloque, tienen un patrón fijo en los 53 bytes de la celda *ATM*. Este patrón es un número consecutivo desde el 0 hasta el 51 para cada uno de los bytes de la celda. Esto es, el número 0 está en el byte 1, el número 1 está en el 2 y así sucesivamente hasta llegar al número 51 en el byte 52. Finalmente el byte 53 de la celda *ATM* cuenta con el byte de *ChkSum*, que es una operación de *OR-EXCLUSIVO* entre los 52 primeros bytes de la celda *ATM*, comenzando con el byte 1. Este patrón lo tienen todas las celdas que son generadas por este bloque (*Gen_UL1*).

Este bloque empieza a generar estas celdas y las entrega a la *IUL2/UL1* cuando la señal *Empieza_GenCell* (que es activada por el bloque *CAP*) está en estado bajo (ver tabla 5-1). Y no dejará de generarlas hasta que esta señal se encuentre en estado alto.

5.1.2.2. *El bloque analizador de celdas* ATM *en el protocolo* UTOPIA Level 1 *(Ana_UL1)*

Este bloque es el encargado de analizar las celdas *ATM* con el protocolo *UTOPIA Level 1* que recibe de la *IUL2/UL1* y reportar al bloque *CAP*, si la celda fue recibida completa y que concuerde el dato *ChkSum* del último byte de la celda con la operación *OR-EXCLUSIVO* de todos los otros primeros 52 bytes.

Este bloque cuenta con dos señales que le indican al bloque *CAP* que se recibió una celda *ATM* completa (*Listo_AnaCell*) y si el resultado de la operación *OR-EXCLUSIVO* es correcta o no lo es (*Resultado_AnaCell*). Para mayor detalle de estas señales, ver tabla 5-1.

5.1.2.3. *El bloque control automático de pruebas (*CAP*)*

El bloque *CAP* es el que tiene el control de la generación de la celdas *ATM* del bloque *Gen_UL1* y es el que recibe el resultado de la celda recibida por el bloque *Ana_UL1*.

Este bloque recibe las señales de control por medio del puerto paralelo de la *PC*. Enseguida, analiza estas señales para dar inicio a las pruebas *LB1*, *LB2* y *LB3*. Finalmente, recibe resultados de estas pruebas individuales y reporta estos resultados mediante señales a la *PC* por el puerto paralelo de esta. Las señales del bloque *CAP* que recibe y envía por el puerto paralelo de la *PC*, así como las señales de comunicación de este bloque con los otros de la *cama de prueba*, se describen a detalle en la tabla 5-1.

En la figura 5-3 se muestran los bloques y las señales que intervienen con el bloque *CAP*.

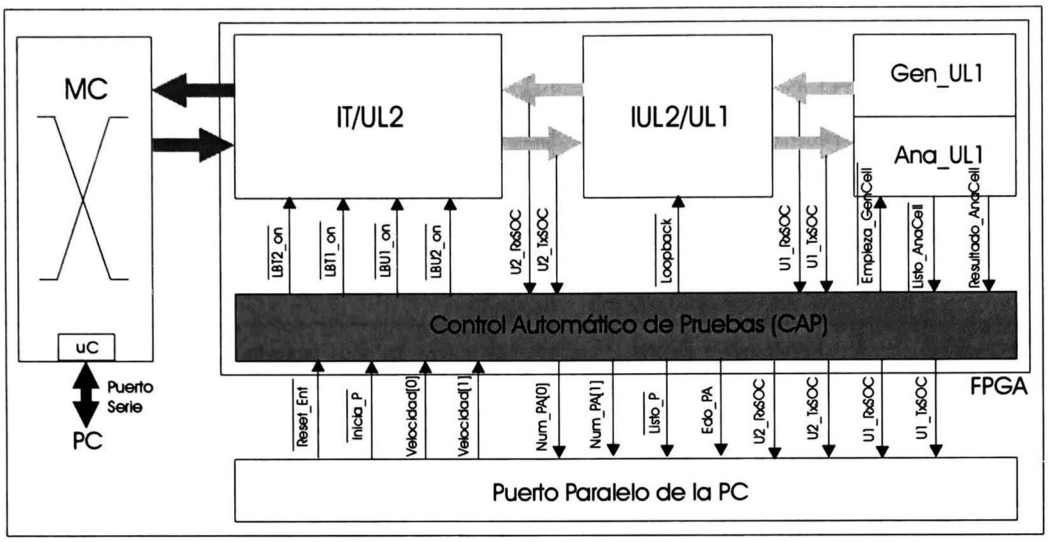


Figura 5-3 Bloque control automático de pruebas (*CAP*)

Señal	Origen	Destino	Descripción
LBT2_on	*CAP*	*IT/UL2*	Señal activa en bajo, que indica un lazo de retroalimentación entre las señales del protocolo *TOPIA*, dentro de *IT/UL2*. Esta señal se activa dependiendo el número de prueba activa.
LBT1_on	*CAP*	*IT/UL2*	Señal activa en bajo, que indica un lazo de retroalimentación entre la interfaz de recepción y la interfaz de transmisión, por el lado del protocolo *TOPIA*, dentro de *IT/UL2*. Esta señal se activa dependiendo el número de prueba activa.
LBU2_on	*CAP*	*IT/UL2*	Señal activa en bajo, que indica un lazo de retroalimentación entre las señales del protocolo *UTOPIA Level 2*, dentro de *IT/UL2*. Esta señal se activa dependiendo el número de prueba activa.
LBU1_on	*CAP*	*IT/UL2*	Señal activa en bajo, que indica un lazo de retroalimentación entre la interfaz de recepción y la interfaz de transmisión, por el lado del protocolo *UTOPIA Level 2*, dentro de *IT/UL2*. Esta señal se activa dependiendo el número de prueba activa.
Loopback	*CAP*	*IUL2/UL1*	Señal activa en bajo, que indica un lazo de retroalimentación entre las señales del protocolo *UTOPIA Level 1*, dentro de *IUL2/UL1*. Esta señal se activa dependiendo el número de prueba activa. Para mayor detalle, ver [5].

U1_RxSOC	*Gen_UL1*	*CAP* *PC*	Señal activa en alto cuando *U1_RxData* (ver [5]) contiene el primer byte válido de la celda del protocolo *UTOPIA Level 1*, en el sentido de recepción de información por parte de *IUL2/UL1*. Esta señal también es transmitida hacia el puerto paralelo de la *PC* para monitoreo de pruebas.
U2_RxSOC	*IUL2/UL1*	*CAP* *PC*	Señal activa en alto cuando *RxData_U* contiene el primer byte válido de la celda del protocolo *UTOPIA Level 2*, en el sentido de recepción de información por parte de *IT/UL2*. Esta señal también es transmitida hacia el puerto paralelo de la *PC* para monitoreo de pruebas.
U2_TxSOC	*IT/UL2*	*CAP* *PC*	Señal activa en alto cuando *TxData_U* contiene el primer byte válido de la celda del protocolo *UTOPIA Level 2*, en el sentido de transmisión de información por parte de *IT/UL2*. Esta señal también es transmitida hacia el puerto paralelo de la *PC* para monitoreo de pruebas.
U1_TxSOC	*IUL2/UL1*	*CAP* *PC*	Señal activa en alto cuando *U1_TxData* (ver [5]) contiene el primer byte válido de la celda del protocolo *UTOPIA Level 1*, en el sentido de transmisión de información por parte de *IUL2/UL1*. Esta señal también es transmitida hacia el puerto paralelo de la *PC* para monitoreo de pruebas.
Empieza_GenCell	*CAP*	*Gen_UL1*	Señal activa en bajo que le indica a *Gen_UL1* que inicie la generación de celdas *ATM* en el protocolo de comunicación *UTOPIA Level 1*, para ser entregadas a *IUL2/UL1*.
Listo_AnaCell	*Ana_UL1*	*CAP*	Señal activa en bajo que le indica a *CAP* que se ha recibido una celda *ATM* completa en el protocolo de comunicación *UTOPIA Level 1*.
Resultado_AnaCell	*Ana_UL1*	*CAP*	Señal que indica, una vez que se recibió una celda *ATM* completa en el protocolo de comunicación *UTOPIA Level 1*; si esta celda concuerda o no con el *ChkSum*. Si concuerda, el resultado de esa prueba es favorable. **Resultado_AnaCell** / **Significado** 0 / *ChkSum* desfavorable 1 / *ChkSum* favorable
Reset_Ent	*PC*	*CAP*	Señal activa en bajo que le indica al bloque *CAP* generar un reset global de toda la lógica en el *FPGA*.
Inicia_P	*PC*	*CAP*	Señal activa en bajo que le indica al bloque *CAP* que inicie con la prueba automática del bloque *IT/UL2* por el lado del protocolo *UTOPIA Level 2*.
Velocidad[1-0]	*PC*	*CAP*	Este bus de 2 señales, indica la frecuencia de *Clk*, a la cual trabajará la lógica contenida en el *FPGA*. La señal de *Clk* se obtiene del cristal de la tarjeta de prueba.

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Velocidad</th><th>Frecuencia de Clk</th></tr>
<tr><td>00</td><td>Cristal / 16</td></tr>
<tr><td>01</td><td>Cristal /8</td></tr>
<tr><td>10</td><td>Cristal / 4</td></tr>
<tr><td>11</td><td>Cristal /2</td></tr>
</table>
</td></tr>
<tr><td>Num_PA[1-0]</td><td>CAP</td><td>PC</td><td>Este bus de 2 señales, indica la prueba actual del sistema. Esta prueba actual está codificada como se muestra en la siguiente tabla:
<table>
<tr><th>Num_PA</th><th>Prueba</th></tr>
<tr><td>00</td><td>---</td></tr>
<tr><td>01</td><td>LB1</td></tr>
<tr><td>10</td><td>LB2</td></tr>
<tr><td>11</td><td>LB3</td></tr>
</table>
</td></tr>
<tr><td>Listo_P</td><td>CAP</td><td>PC</td><td>Señal activa en bajo que indica que se ha concluido la prueba de IT/UL2.</td></tr>
<tr><td>Edo_PA</td><td>CAP</td><td>PC</td><td>Señal que indica (una vez que la señal Listo_P está en bajo) si la prueba de IT/UL2 es favorable o desfavorable.
<table>
<tr><th>Edo_PA</th><th>Significado</th></tr>
<tr><td>0</td><td>Resultado desfavorable</td></tr>
<tr><td>1</td><td>Resultado favorable</td></tr>
</table>
</td></tr>
</table>

Tabla 5-1 Señales de bloque de control automático de pruebas (*CAP*)

5.1.2.3.1. Procedimiento de prueba que realiza el bloque de control automático de pruebas (*CAP*)

El procedimiento que realiza el bloque *CAP* para la prueba de la *IT/UL2* se puede describir en los siguientes pasos:

1. Para dar inicio con las diferentes pruebas (*LB1*, *LB2* y *LB3*), las señal *Reset_Ent* debe estar en estado alto. Previamente, el bus *Velocidad[1-0]* configura la frecuencia de trabajo de la señal *Clk* (que es la señal de reloj de todo el sistema).
2. Hasta que el bloque *CAP* detecte que la señal *Inicia_P* está en estado bajo, en ese momento se inicia con el procedimiento de prueba de la *IT/UL2*.
3. El bloque *CAP* pone la señal *Loopback* en estado bajo. Esto hace que se active el "*loopback*" *LB1* de la *IUL2/UL1*, permitiendo que la información (celdas *ATM* con protocolo *UTOPIA Level 1*) proveniente del bloque *Gen_UL1* sea enviada de regreso al bloque *Ana_UL1*. La función de este loopback (*LB1*) es probar la conexión entre la *IUL2/UL1* y los bloques *Gen_UL1* / *Ana_UL1*. En este punto la señal *Listo_P* esta en estado alto, indicando que no ha finalizado la prueba de la *IT/UL2*; también, el bus *Num_PA* indica que se está en la prueba del "*loopback*" *LB1*.

4. Enseguida, el bloque *CAP* pone en estado bajo la señal *Empieza_GenCell*, lo que hace que el bloque *Gen_UL1* empiece a generar celdas *ATM* con protocolo *UTOPIA Level 1* y estas sean entregadas a la *IUL2/UL1*. Debido a que está activo el "*loopback*" *LB1*, la celda que recibe esta interfaz es inmediatamente enviada al bloque *Ana_UL1* para su posterior comparación.
5. Una vez que la celda *ATM* es recibida por el bloque *Ana_UL1*, este bloque compara el último byte de esta (byte 53) para comprobar si el byte de *ChkSum* recibido concuerda con el calculado por este bloque. Una vez hecha esta comparación, este bloque reporta al bloque *CAP* que terminó de recibir la celda *ATM* por medio de la señal *Listo_AnaCell* y el resultado de la comparación es reportado con la señal *Resultado_AnaCell*.
6. Si el resultado que reporta el bloque *Ana_UL1* es desfavorable, el bloque *CAP* detiene la prueba de la *IT/UL2*, reporta a la *PC* (por medio del puerto paralelo) que finalizó la prueba a la *IT/UL2* (poniendo en bajo la señal *Listo_P*) y que el resultado de la prueba es desfavorable (poniendo la señal Edo_PA en bajo). En este punto, el bus *Num_PA* indica que se está en la prueba del "*loopback*" *LB1*; esto le indica a la *PC*, que el problema se encuentra entre la *IUL2/UL1* y los bloques *Gen_UL1 / Ana_UL1*.
7. Si el resultado reportado por el bloque *Ana_UL1* es favorable, entonces el bloque *CAP* pone la señal *Loopback* en estado alto y pone la señal *LBU2_on* en estado bajo (*Loopback*, *LBU1_on*, *LBT1_on* y *LBT2_on* en estado alto). Esto hace que se desactive el *loopback LB1* de la *IUL2/UL1* y se active el *loopback LB2* de la *IT/UL2*; permitiendo que la información (celdas *ATM* con protocolo *UTOPIA Level 1*) proveniente del bloque *Gen_UL1* pase a través de la *IUL2/UL1*, convirtiéndose al protocolo *UTOPIA Level 2* e ingrese a la *IT/UL2*. Esta información que ingresa a la *IT/UL2* es inmediatamente enviada de regreso por el *loopback LB2* a la *IUL2/UL1* para ser convertida ahora del protocolo *UTOPIA Level 2* a *UTOPIA Level 1* y ya en este protocolo ser transmitida al bloque *Ana_UL1*. La función de este *loopback* (*LB2*) es primeramente, probar la conversión de *UTOPIA Level 1* a *UTOPIA Level 2* hecha por la *IUL2/UL1*, después la conexión entre las interfaces *IT/UL2* y *IUL2/UL1*, y finalmente la prueba de la conversión de *UTOPIA Level 2* a *UTOPIA Level 1* hecha también por la *IUL2/UL1*. Este "*loopback*" (*LB2*) prueba principalmente la correcta funcionalidad de la *IUL2/UL1*. En este punto la señal *Listo_P* esta en estado alto, indicando que no ha finalizado la prueba de la *IT/UL2*; también, el bus *Num_PA* indica que se está en la prueba del "*loopback*" *LB2*.
8. Enseguida, el bloque *CAP* pone en estado bajo la señal *Empieza_GenCell*, lo que hace que el bloque *Gen_UL1* empiece a generar celdas *ATM* con protocolo *UTOPIA Level 1*. Estas celdas son entregadas a la *IUL2/UL1* y convertidas por esta a *UTOPIA Level 2*. Debido a que está activo el "*loopback*" *LB2* las celdas que recibe esta interfaz son inmediatamente enviadas de regreso a la *IUL2/UL1* para ser convertidas al protocolo *UTOPIA Level 1* de nuevo y finalmente ser entregadas al bloque *Ana_UL1* para su posterior comparación.
9. Una vez que la primera celda *ATM* es recibida por el bloque *Ana_UL1*, este bloque compara el byte 53 de la celda y así comprobar si el byte de *ChkSum* recibido concuerda con el calculado por este bloque. Una vez hecha esta comparación, este bloque reporta al bloque *CAP* el fin de la recepción de la celda *ATM* por medio de la señal *Listo_AnaCell* y que tiene el resultado de la comparación, reportándolo con la señal *Resultado_AnaCell*.

10. Si el resultado que reporta el bloque *Ana_UL1* es desfavorable, el bloque *CAP* detiene la prueba de la *IT/UL2*, reporta a la *PC* (por medio del puerto paralelo) que finalizó la prueba a la *IT/UL2* (poniendo en bajo la señal *Listo_P*) y que el resultado de la prueba es desfavorable (poniendo la señal *Edo_PA* en bajo). En este punto, el bus *Num_PA* indica que se está en la prueba del *"loopback" LB2*; esto le indica a la *PC*, que el problema se encuentra en la *IUL2/UL1* o entre la *IT/UL2* y la *IUL2/UL1*.

11. Si el resultado reportado por el bloque *Ana_UL1* es favorable, entonces el bloque *CAP* pone la señal *Loopback* en estado alto y pone la señal *LBT1_on* en estado bajo (*Loopback*, *LBU1_on*, *LBU2_on* y *LBT2_on* en estado alto). Esto hace que se desactive el *"loopback" LB1* de la *IUL2/UL1*, el *"loopback" LB2* de la *IT/UL2* y se active el *"loopback" LB3* de la *IT/UL2*; permitiendo que la información (celdas *ATM* con protocolo *UTOPIA Level 2*) proveniente de la *IUL2/UL1* pase a través de la *IT/UL2* convirtiéndose a *TOPIA*. Esta información es inmediatamente enviada de regreso por el *"loopback" LB3* (convirtiéndose de nuevo a *UTOPIA Level 2*) a la *IUL2/UL1* para ser convertida ahora de *UTOPIA Level 2* a *UTOPIA Level 1* y ya en este protocolo ser transmitida al bloque *Ana_UL1*. La función de este *"loopback" (LB3)* es la de probar la conversión de *UTOPIA Level 2* a *TOPIA* hecha por la *IT/UL2* y la prueba de la conversión de *TOPIA* a *UTOPIA Level 2* hecha también por esta interfaz. En este punto, la señal *Listo_P* esta en estado alto indicando que no ha finalizado la prueba de la *IT/UL2*; también, el bus *Num_PA* indica que se está en la prueba del *loopback LB3*.

12. Enseguida, el bloque *CAP* pone en estado bajo la señal *Empieza_GenCell*, lo que hace que el bloque *Gen_UL1* empiece a generar celdas *ATM* con protocolo *UTOPIA Level 1*. Estas celdas son entregadas a la *IUL2/UL1* y convertidas por esta a *UTOPIA Level 2*, para después ser entregadas a la *IT/UL2* y convertidas por esta a *TOPIA*. Debido a que está activo el *"loopback" LB3*, las celdas que recibe esta interfaz son inmediatamente enviadas de regreso a la *IUL2/UL1* para ser convertidas de *TOPIA* a *UTOPIA Level 2* (por la *IT/UL2*), después de *UTOPIA Level 2* a *UTOPIA Level 1* (por la *IUL2/UL1*) y finalmente ser entregadas al bloque *Ana_UL1* para su posterior comparación.

13. Una vez que la primera celda *ATM* es recibida por el bloque *Ana_UL1*, este bloque compara el último byte de la celda (byte 53) para comprobar si el byte de *ChkSum* recibido concuerda con el calculado por este bloque. Una vez hecha esta comparación, este bloque reporta al bloque *CAP* el fin de recepción de la celda *ATM* por medio de la señal *Listo_AnaCell* y que se tiene el resultado de la comparación, reportándolo con la señal *Resultado_AnaCell*.

14. Si el resultado que reporta el bloque *Ana_UL1* es desfavorable, el bloque *CAP* detiene la prueba de la *IT/UL2*, reporta a la *PC* (por medio del puerto paralelo) que finalizó la prueba a la *IT/UL2* (poniendo en bajo la señal *Listo_P*) y que el resultado de la prueba es desfavorable (poniendo la señal *Edo_PA* en bajo). En este punto, el bus *Num_PA* indica que se esta en la prueba del *"loopback" LB3*; esto le indica a la *PC* que el problema se encuentra en la *IT/UL2*.

15. Si el resultado reportado por el bloque *Ana_UL1* es favorable, el bloque *CAP* detiene la prueba de la *IT/UL2*, reporta a la *PC* (por medio del puerto paralelo) que finalizó la prueba a la *IT/UL2* (poniendo en bajo la señal *Listo_P*) y que el resultado de la prueba completa a la *IT/UL2* es favorable (poniendo la señal *Edo_PA* en alto).

5.2. Implementación

Las interfaces *IT/UL2*, *I8/16b* y la *cama de pruebas* completa (*IT/UL2*, *IUL2/UL1*, *Gen_UL1*, *Ana_UL1* y *control automático de pruebas*) fueron diseñadas e implementadas en código *VHDL*. Todos estos bloques fueron simulados y sintetizados en la herramienta *Max2Win* de la compañía *ALTERA*, para el *FPGA EPF10k100GC503-3* de la familia *Flex10K100* (ver detalles de este chip en [24]).

Los resultados de los componentes empleados del *FPGA*, así como el porcentaje de la lógica empleada por cada diseño, se describe a continuación:

Para la *IT/UL2* se cuenta con dos resultados de síntesis. Uno de ellos toma en cuenta que la *TR* se encuentra en un memoria externa al *FPGA*. En el otro, se implementó esta tabla en memoria *RAM* interna al *FPGA*. En este último se planeó de esta forma, ya que las pruebas realizadas solo necesitan algunos localidades de memoria de esta *TR*; lo que nos permite aprovechar los recursos del *FPGA* para no tener más dispositivos (chips) externos en estas pruebas y reducir así el riesgo de problemas de conexiones. El tamaño de la memoria interna para la implementación de la *TR* es de 512 localidades de 8 bits cada una. Esto permite tener una tabla con 256 pares de *VPI* y *VCI*.

La síntesis de la *IT/UL2* tomando en cuenta que la *TR* se encuentra fuera del *FPGA*, tiene las siguientes características (resultados):

Característica *EPF10k100GC503-3*	Resultado de la síntesis
Total de entradas y salidas empleadas	91/400 (22%)
Total de *flipflops* empleados	556
Total de memoria interna (*EABs*) empleada	4/12 (33%)
Total de celdas lógicas empleadas	1288/4992 (**25%**)

Tabla 5-2 Resultados de la síntesis de la *IT/UL2* sin la tabla de enrutado

La síntesis de la *IT/UL2* tomando en cuenta que la *TR* se encuentra dentro del *FPGA*, tiene las siguientes características (resultados):

Característica *EPF10k100GC503-3*	Resultado de la síntesis
Total de entradas y salidas empleadas	63/400 (15%)
Total de *flipflops* empleados	536
Total de memoria interna (*EABs*) empleada	5/12 (41%)
Total de celdas lógicas empleadas	1215/4992 (**24%**)

Tabla 5-3 Resultados de la síntesis de la *IT/UL2* con la tabla de enrutado interna

La síntesis de la *I8/16b* tiene las siguientes características (resultados):

Característica *EPF10k100GC503-3*	Resultado de la síntesis
Total de entradas y salidas empleadas	66/400 (16%)
Total de *flipflops* empleados	123
Total de memoria interna (*EABs*) empleada	0/12 (0%)
Total de celdas lógicas empleadas	185/4992 (**3%**)

Tabla 5-4 Resultados de la síntesis de la *I8/16b*

Para la *cama de prueba* completa (*IT/UL2*, *IUL2/UL1*, *Gen_UL1*, *Ana_UL1* y *control automático de pruebas*) tomando en cuenta que la *IT/UL2* cuenta internamente con la *TR*. La síntesis del *FPGA* de este diseño, tiene las siguientes características (resultados):

Característica *EPF10k100GC503-3*	Resultado de la síntesis
Total de entradas y salidas empleadas	13/400 (3%)
Total de *flipflops* empleados	753
Total de memoria interna (*EABs*) empleada	8/12 (66%)
Total de celdas lógicas empleadas	1516/4992 (**30%**)

Tabla 5-5 Resultados de la síntesis de la *cama de prueba* completa

Finalmente, se diseñaron los esquemáticos completos de la *IE1-T1/MC*, estos se muestran en el apéndice B de este documento.

6 Conclusiones

En esta tesis se desarrolló el diseño de la arquitectura de la interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad; empleando tanto circuitos comerciales (dispositivos de diferentes fabricantes que realizan funciones particulares), así como circuitos diseñados bajo un lenguaje de descripción de *hardware* (*VHDL*) empleando un circuito *FPGA*.

Se diseñó e implementó una interfaz entre el protocolo de comunicación propio (*TOPIA*) de una matriz de conmutación *ATM* de alta velocidad y el protocolo de comunicación *UTOPIA Level 2*. A esta interfaz se le añadieron "*loopbacks*" para la comprobación del correcto funcionamiento del cambio de protocolos de comunicación. Además, esta interfaz también cuenta con una prueba de la tabla de enrutado (*TR*), para asegurar que la información de enrutado sea la correcta ya que la *TR* es una memoria externa a la interfaz *TOPIA/UTOPIA Level 2*. Todo este diseño fue implementado en un dispositivo *FPGA Flex 10K100* de la compañía *Altera*.

Así mismo, se hizo el diseño y la implementación de una interfaz de comunicación entre flujos de 8 bits y 16 bits de datos, sintetizando el diseño en un dispositivo *FPGA Flex 10K100* de la compañía *Altera*.

También se diseñó e implementó una plataforma de prueba de la interfaz entre el protocolo de comunicación propio (*TOPIA*) de una matriz de conmutación *ATM* de alta velocidad y el protocolo de comunicación *UTOPIA Level 2*; con interfaz hacia una computadora personal (por medio del puerto paralelo), de tal modo que esta prueba sea manejada y supervisada desde un *software* en la *PC*. Todo este diseño abarca:

- La interfaz entre el protocolo *TOPIA* (de la matriz de conmutación *ATM* de alta velocidad) y el protocolo *UTOPIA Level 2*.
- La interfaz entre los protocolos de comunicación *UTOPIA Level 1* y el protocolo *UTOPIA Level 2* (descrito en [5]).

  En generador y analizador de celdas con el protocolo *UTOPIA Level 1*.

  El control automático de estas pruebas.

Todos estos diseños fueron implementados en un dispositivo *FPGA Flex 10K100* de la compañía *Altera*, interactuando y realizando sus funciones específicas, todos ellos juntos.

Se realizaron los esquemáticos completos de la interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad.

Como trabajo futuro, se propone:

La realización del *PCB* de la interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad.

La programación (*firmware*) del microcontrolador *AT89S8252* para la comunicación entre la interfaz 8/16 bits y el puerto serie asíncrono de la computadora personal.

Una interfaz de *software* (en la computadora personal) para la inicialización, control y monitoreo de la interfaz entre enlaces *E1/T1* y una matriz de conmutación *ATM* de alta velocidad.

Bibliografía y documentación relacionada

[1] González P. J., "Arquitectura, Diseño e Implementación de una Matriz de Conmutación de Alta Velocidad", Tesis del CINVESTAV-IPN, Gdl., 1998.

[2] Núñez R. A., "Matriz de Conmutación de Alta Velocidad: Su Verificación y su Elemento de Salida", Tesis del CINVESTAV-IPN, Gdl., 2000.

[3] Medina T. R., "Interfaz entre una Red Ethernet 100BaseTx y una Matriz de Conmutación de Celdas", Tesis del CINVESTAV-IPN, Gdl., 1999.

[4] Larios G. A., "Arquitectura, Diseño e Implementación de una Tabla de Enrutado para un Conmutador de Alta Velocidad", Tesis del CINVESTAV-IPN, Gdl., 1999.

[5] Silva B. J., "Rediseño de la Arquitectura de una Matriz de Conmutación de Alta Velocidad con interfaces UTOPIA Level 1", Tesis del CINVESTAV-IPN, Gdl., 1998.

[6] Tanenbaum A., "Redes de Computadoras" Tercera Edición, Prentice Hall Hispanoamérica S. A., 1997.

[7] Bellamy, J., "Digital Telephony" Second Edition, John Wiley & Sons, Inc.,1991.

[8] Pardo C. F., "VHDL Lenguaje para descripción y modelado de circuitos", Universidad de Valencia, 1997.

[9] Gajski, D., "Principles of Digital Design", Prentice Hall, 1997.

[10] Bergeron, J., "Writing Testbenches, Functional Verification of HDL Models", Kluwer Academic Publishers, 2000.

[11] Skahill, K., "VHDL for Programmable Logic", Addison-Wesley, 1996.

[12] Ashenden, P., "The Designer's Guide to VHDL", Morgan Kaufmann Publishers, Inc., 1996.

[13] The ATM Forum, Technical Committee, "UTOPIA Specification Level 1, Version 2.01, af-phy-0017.000), March 21, 1994.

[14] The ATM Forum, Technical Committee, "Utopia Level 2, Version 1.0, af-phy-0039.000), June 1995.

[15] ITU-T, Telecommunication Standarization Sector of ITU, "G.703, Características físicas y eléctricas de las interfaces digitales jerárquicas", 10/98.

[16] ITU-T, Telecommunication Standarization Sector of ITU, "G.704, Synchronous Frame Structures Used at 1544, 6312, 2048, 8488 and 44736 kbit/s Hierarchical Levels", 07/95.

[17] Mukherjee, N., Chakraborty, T., Karri, R., "Built-In Self-Test: A Complete Test Solution for Telecommunication Systems", IEEE Communications Magazine, Vol. 37 No. 6, June 1999, pp. 72-78.

[18] Barbagallo, S., Lobetti Bodoni, M., Benso, A., Chiusano, S., Prinetto, P., "Testing Embedded Memories in Telecommunication Systems", IEEE Communications Magazine, Vol. 37 No. 6, June 1999, pp. 84-89.

[19] Al-Assad, H., Murray, B., Hayes, J., "Online BIST for Embedded Systems", IEEE Design & Test of Computers Magazine, October-December 1998, pp. 17-24.

[20] Miyano, S., Sato, K., Numata, K., "Universal Test Interface for Embedded-DRAM Testing", IEEE Design & Test of Computers Magazine, January-March 1999, pp. 53-58.

[21] Huang, C., Huang, J., Wu, C., Chang, T., "A Programmable BIST Core for Embedded DRAM", IEEE Design & Test of Computers Magazine, January-March 1999, pp. 59-70.

[22] Siemens ICs for Communications, "Interworking Element IWE8, PXB 4220 Version 3.1, PXB 4221 Version 3.1", Data Sheet, 03.99

[23] Siemens ICs for Communications, "Quad Framing and Line Interface Component for E1/T1/J1 QuadFALC™, PEB 22554 Version 1.1", Preliminary Data Sheet, 09.98

[24] Altera Corporation, "Flex 10K Embedded Programmable Logic Family, Version 4.02", Data Sheet, May 2000.

[25] ATMEL, "8-Bit Microcontroller with 8K Bytes Flash, AT89LS8252" Data Sheet, 12/97.

[26] ATMEL, "1-Megabit (128K x 8) Low Voltage Paged Parallel EEPROMs, AT28LV010", Data Sheet, 10/98.

[27] Samsung Electronics, "64Kx36-Bit Synchronous Burst SRAM 3.3V Power, KM736V687", Datasheets for 100TQFP, Rev 1.0, May 1997.

Apéndice A. Interfaz de recepción de la *IT/UL2*

En este apéndice se describe la implementación de la arquitectura de la *interfaz de recepción* (*I_Rx*) en la *IT/UL2*. Antes de comenzar con la descripción de esta interfaz, es necesario la descripción de la estructura de los archivos (*el árbol*) del diseño completo de la *IT/UL2*. En la figura A-1 se muestra a la *IT/UL2* a bloques. Cada uno de estos bloques tiene asociado un archivo en código *VHDL* (archivo con extensión *.vhd*) que agrupa el diseño completo de este bloque en particular (el *package* para *VHDL*, ver [8][11][12]).

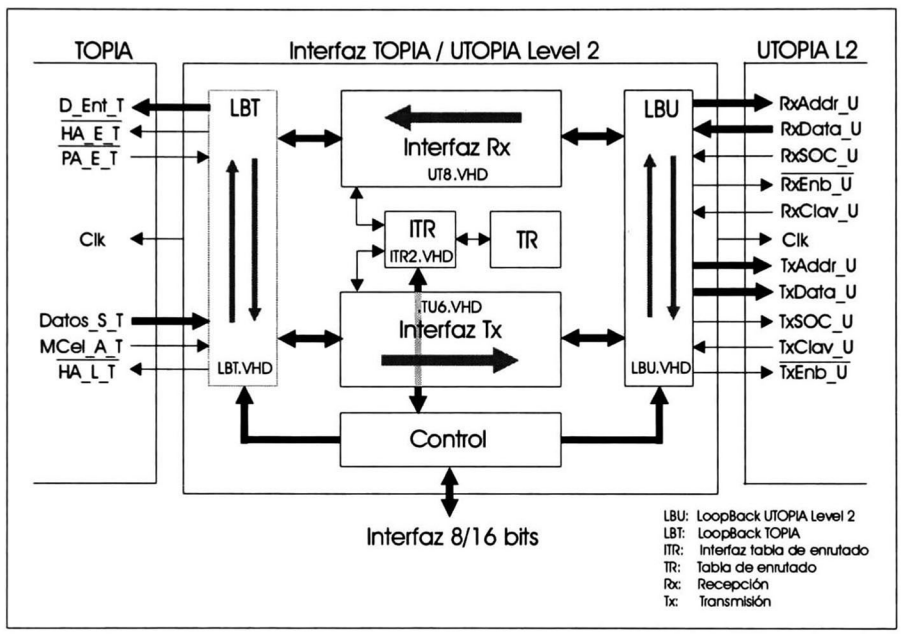


Figura A - 1 Interfaz entre protocolos de comunicación *TOPIA* y *UTOPIA Level 2*

El árbol de los archivos de la *IT/UL2* está descrito en la figura A-2 y la estructura de archivos del diseño de la *I_Rx* es mostrada en la figura A-3.

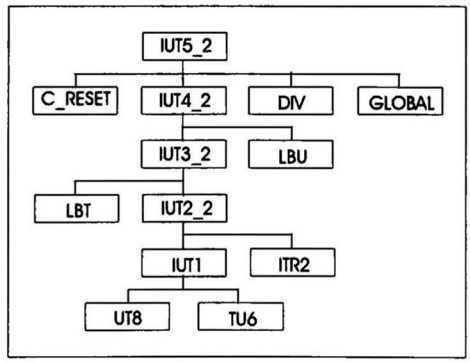


Figura A - 2 Estructura de los archivos de la *IT/UL2*

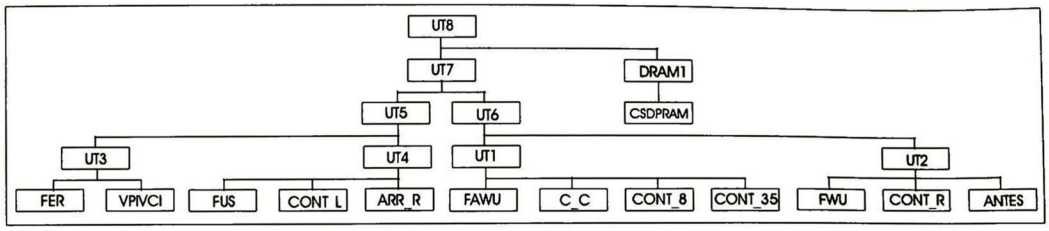


Figura A - 3 Estructura de los archivos de la interfaz de recepción

Descripción de la implementación de la arquitectura de la interfaz de recepción (*I_Rx*)

Para la implementación de la arquitectura de la *I_Rx* son necesarias algunas *máquinas de estados finitos* (*FSMs*), diferentes contadores, registros, lógica combinacional, así como la implementación de una memoria *RAM* de doble puerto [9]. En la figura A-8 se muestra un diagrama a bloques de la solución a la arquitectura de la *I_Rx*, así como de las señales que intervienen en cada uno de los bloques mostrados. También, en cada uno de estos bloques se ilustra el archivo que contiene el código *VHDL* (archivo con extensión *.vhd*) de este bloque en específico. En esta figura no se incluyen las señales de reloj y de reset, ya que estas son señales de entrada a la mayoría de los bloques mostrados en el diseño. Los bloques remarcados de un color oscuro se encuentran fuera de la *I_Rx*. Para detalles sobre el diseño de la *I_Rx*, ver capítulo 3 de este documento.

Descripción de los bloques de la interfaz de recepción (*I_Rx*)

A continuación se describen las funciones de cada uno de los bloques internos de la *I_Rx*, así como cada una de las *FSMs* (en formato de tabla) contenidas en esta interfaz.

FSM Aleatorio SAR>>IT/UL2 (fawu.vhd)

La función de esta *FSM* es la de controlar el bloque de *conteo cíclico* (*c_c.vhd*) para monitorear cual de las 8 posibles capas *PHY* (circuitos *SAR*), tiene una celda disponible para ser entregada a la *IT/UL2*. Esta FSM maneja dos diferentes contadores: *contador 0-8* (cont_8.vhd) y *contador 0-35* (*cont_35.vhd*). En la tabla A-1 se describe esta *FSM* mostrando además de los estados actuales y siguientes, la condición de salto del estado actual al siguiente; así como las señales de salida en cada estado actual.

Estado actual	Estado siguiente		Salida actual						
	Condición	Estado	A	B	C	D	E	F	G
S1	*reset=1*	S1	0	11	1	0	0	0	0
	Fin_Tx=0	S2							
S2	*RxClav=1*	S3	1	01	1	0	0	0	0
S3		S4	0	00	1	0	0	0	0
S4	*RxSOC=1*	S5	0	11	0	0	0	0	0
	RxSOC=0	S42							
S42	*RxSOC=1*	S5	0	11	0	0	0	0	0
	RxSOC=0	S3							
S5	*F_Cuenta_35=1 & clk_2_aux=1*	S6	0	11	0	0	1	1	1
	F_Cuenta_35=1 & clk_2_aux=0	S52							
S52		S6	0	11	0	0	0	1	1
S6	*RxClav=1*	S8	0	01	0	1	0	0	0
	RxClav=0	S7							
S7	*F_Cuenta_8=0 & RxClav=1*	S8	1	01	0	1	0	0	0
	F_Cuenta_8=1 & RxClav=0	S10							
	F_Cuenta_8=1 & RxClav=1 & Fin_Tx=1	S9							
	F_Cuenta_8=1 & RxClav=1 & Fin_Tx=0 & clk_2_aux=1	S3							
S8	*F_Cuenta_8=1 & Fin_Tx=0*	S3	0	11	0	1	0	0	0
	F_Cuenta_8=1 & Fin_Tx=1	S11							
S9	*Fin_Tx=0 & clk_2_aux=1*	S3	0	11	1	1	0	0	0
S10	*RxClav=1 & Fin_Tx=0 & clk_2_aux=1*	S3	1	01	1	0	0	0	0
	RxClav=1 & [Fin_Tx=1 OR clk_2_aux=0]	S11							
S11	*Fin_Tx=0 & clk_2_aux=1*	S3	0	11	1	0	0	0	0
A=*C_C_Cuenta_8*; B=*Escoge*; C=*RxEnb*; D=*Cuenta_8*; E=*Cuenta_35*; F=*resta*; G=*Inicia_Cuenta_8*									

Tabla A - 1 ***FSM Aleatorio SAR>>IT/UL2 (fawu.vhd)***

Conteo cíclico (c_c.vhd)

Este bloque es controlado por la *FSM Aleatorio SAR >> IT/UL2 (fawu.vhd)* y su función es la de proveer del bus de dirección *RxAddr* a las capas *PHY* (circuitos *SAR*). Cuando se hace el monitoreo de las capas *PHY* para saber cual de estos dispositivos tiene una celda disponible, la salida de este bloque será una dirección de 1 a 8 (restricción del diseño de la *IT/UL2*) de lo contrario, la dirección será de 31 (1Fh). Para mayor detalle sobre la señal *RxAddr*, ver detalles en [14].

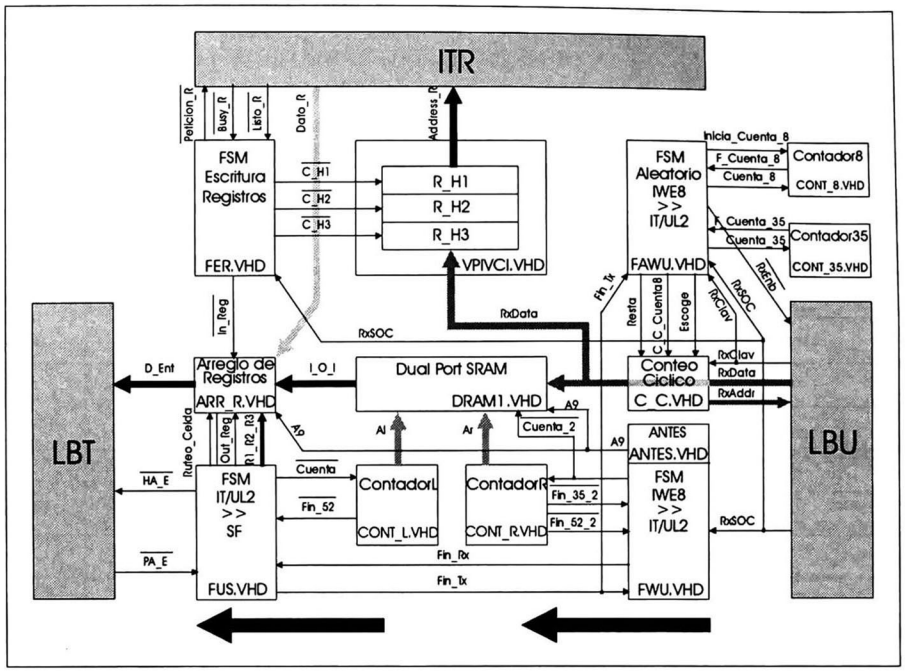


Figura A - 4 Diagrama del diseño de la arquitectura de la interfaz de recepción

FSM SAR>>IT/UL2 (fwu.vhd)

La función de esta *FSM* es la de almacenar los 53 bytes de la celda *ATM* en la *memoria RAM de doble puerto* (*dram1.vhd*). Para realizar esta función, la FSM cuenta con el contador de 0 a 52 (*cont_r.vhd*). Esta *FSM* maneja la señal *A9* en conjunto con el bloque *antes.vhd* controlando la señal *cambia*. Esta señal (*A9*) es importante porque con esta se controla si la información será almacenada en la primera o segunda mitad de la *memoria RAM de doble puerto*. Si la primer mitad de esta memoria está siendo usada para guardar información proveniente de una capa *PHY*, entonces la segunda está siendo empleada para transferir la celda (anteriormente guardada en esta segunda mitad de la *memoria RAM de doble puerto*) hacia la *MC*.

Cuando se finaliza con la recepción de una celda completa, esta *FSM* activa la señal *Fin_Tx* para hacer el cambio de mitad de la memoria e iniciar la recepción de la nueva celda *ATM*, pero ahora en la otra mitad de la memoria. En la tabla A-2 se describe esta *FSM* mostrando además de los estados actuales y siguientes, la condición de salto del estado actual al siguiente; así como las señales de salida en cada estado actual.

Estado	Estado siguiente		Salida actual		
actual	Condición	Estado	A	B	C
S1	*reset=1*	S1	0	1	1

	Fin_Tx=0	S2			
S2	*RxSOC=1*	S3	0	1	1
S3	*Fin_35_2=0*	S4	0	0	1
S4	*Fin_52_2=0*	S5	0	0	1
S5	*Fin_Tx=0*	S7	0	0	1
	Fin_Tx=1	S6			
S6	*Fin_Tx=0*	S7	0	1	1
S7	*RxClav=1*	S8	1	1	0
	RxClav=0	S7			
A=*cambia*; B=*Cuenta_2*; C=*Fin_Rx*					

Tabla A - 2 ***FSM SAR>>IT/UL2 (fwu.vhd)***

FSM Escritura Registros (fer.vhd)

Esta *FSM* es la encargada de tomar la información de *VPI* y *VCI* del encabezado de la celda *ATM* y con esta información obtener los dos primeros bytes de la celda del protocolo *TOPIA*, que son necesarios para el enrutado de la información dentro del *SF*. Para desarrollar esta función, la *FSM* controla la carga de los registros *H1*, *H2* y *H3* (contenidos en el archivo *vpivci.vhd*) que es donde se guardan los 3 primeros bytes del encabezado de la celda *ATM*. También está conectada al bloque *ITR* que es el bloque encargado de la escritura y la lectura de la tabla de enrutado. Una vez que este bloque cuenta con lo dos bytes de enrutado (*R1a* y *R2a*), activa la señal *In_Reg* para que la información de enrutado sea almacenada en el bloque *arreglo de registros* (*arr_r.vhd*). En la tabla A-3 se describe esta *FSM*, mostrando además de los estados actuales y siguientes, la condición de salto del estado actual al siguiente; así como las señales de salida en cada estado actual.

Estado	Estado siguiente		Salida actual				
actual	Condición	Estado	A	B	C	D	E
S1	*reset=1*	S1	1	1	1	1	1
	RxSOC=1	S2					
S2		S3	0	1	1	1	1
S3		S4	1	0	1	1	1
S4		S5	1	1	0	1	1
S5	*Busy_R=0*	S6	1	1	1	1	0
S6	*Listo_R=0*	S7	1	1	1	1	1
S7		S1	1	1	1	0	1
A=*C_H1*; B=*C_H2*; C=*C_H3*; D=*In_Reg*; E=*Peticion_R*							

Tabla A - 3 ***FSM Escritura Registros (fer.vhd)***

Arreglo de registros (arr_r.vhd)

Este bloque cuenta con dos registros con capacidad de 2 bytes cada uno (un byte para *R1a* y otro para *R1b*) y lógica combinacional con el fin de saber cuando un registro se emplea para almacenar información o para leerla. Para realizar esta función este bloque necesita de la señal *A9*. Y para el almacenado o la escritura este bloque recibe las señales *In_Reg* y *Out_Reg*.

Además, recibe la información de salida de la *memoria RAM de doble puerto* y junto con las señales de control *R1_R2_R3* y *Ruteo_Celda* (provenientes del bloque *FSM IT/UL2>>SF*), decide el envío hacia la *MC* de la información de enrutado (3 primeros bytes de la celda de 56 bytes del protocolo *TOPIA*) o de la información de la celda *ATM* de 53 bytes.

FSM IT/UL2>>SF (fus.vhd)

Esta *FSM* es la encargada del control de la transferencia de información hacia la *MC* (protocolo *TOPIA*). Para lograr esta función, este bloque se encarga del control de la lectura de la *memoria RAM de doble puerto* y para esto maneja al contador *cont_1.vhd* el cual proporciona la dirección de lectura de esta memoria. Además, envía señales de control al bloque *arreglo de registros* para el control del tipo de información que se envía a la *MC* (información de enrutado o información de la celda *ATM* de 53 bytes). Cuando finaliza la transmisión de una celda completa (56 bytes), se activa la señal *Fin_Tx* y espera a que se active la señal *Fin_Rx* para iniciar la transmisión de una nueva celda pero ahora de la otra mitad de la memoria. En la tabla A-4 se describe la *FSM,* mostrando además de los estados actuales, siguientes y la condición de salto del estado actual al siguiente; así como las señales de salida en cada estado actual.

Estado actual	Estado siguiente		Salida actual					
	Condición	Estado	A	B	C	D	E	F
S1	*reset=1*	S1	1	1	1	01	1	0
	Fin_Rx=0 & PA_E=1	S3						
	Fin_Rx=0 & PA_E=0	S2						
S2	*PA_E=1*	S3	1	1	1	01	1	0
S3		S4	0	1	0	01	1	0
S4		S5	1	1	1	00	1	1
S5		S6	1	0	1	11	1	1
S6	*Fin_52=0*	S7	1	0	1	01	0	1
S7		S1	1	1	1	01	0	1
A=*HA_E*; B=*Cuenta*; C=*Out_Reg*; D=*R1_R2_R3*; E=*Ruteo_Celda*; F=*Fin_Tx*								

Tabla A - 4 *FSM IT/UL2>>SF (fus.vhd)*

Señales del diseño de la arquitectura de la interfaz de recepción

En la tabla A-5 se describe cada una de las señales empleadas en el diseño de la arquitectura de la *I_Rx*.

Señal	Origen	Destino	Descripción
Clk	*IT/UL2*	*I_Rx*	Reloj provisto por la *IT/UL2* para la sincronización de la transferencia de los datos. La frecuencia máxima de esta señal es de 25 *MHz*. El trabajo es con flancos positivos.
reset	*IT/UL2*	*I_Rx*	Esta señal lleva a las *FSMs* a su estado inicial, así como al estado inicial de todos los contadores.
RxAddr[4-0]	*c_c.vhd*	*LBU*	Consiste en un bus de 5 líneas de salida. Esta señal es utilizada para seleccionar alguna de las capas *PHY*. El valor de "1Fh" en este bus, indica una capa *PHY* nula.
RxData[7-0]	*LBU*	*dram1.vhd* *arr_r.vhd*	Consiste de un bus de 8 líneas de entrada. Este bus es utilizado para la transferencia de los bytes de la celda *ATM*.
RxSOC	*LBU*	*fwu.vhd*, *fawu.vhd*, *fer.vhd*	Señal activa en alto cuando *RxData* contiene el primer byte válido de la celda *ATM*.
RxEnb	*fawu.vhd*	*LBU*	Señal activa en bajo cuando *RxData* contiene datos válidos de la celda *ATM*.
RxClav	*LBU*	*c_c.vhd*, *fawu.vhd*	Señal activa en alto, que se utiliza para indicar que se acepta la transferencia de una celda completa hacia la *I_Rx* de cualquiera de las posibles capas *PHY*.
D_Ent[7-0]	*arr_r.vhd*	*LBT*	Consiste de un bus de 8 líneas de salida. Este bus es utilizado para la transferencia de los bytes de la celda *ATM*.
HA_E	*fus.vhd*	*LBT*	Señal activa en bajo. Sirve para iniciar la transferencia de un ciclo de celda.
PA_E	*LBT*	*fus.vhd*	Señal activa en bajo. Sirve para detener la transmisión de la información de la *I_Rx* hacia el protocolo *TOPIA*.
Fin_Rx	*fwu.vhd*	*fus.vhd*	Señal activa en bajo. Indica que se acabó de recibir una celda completa del protocolo de comunicación *UTOPIA Level 2*.
Fin_Tx	*fus.vhd*	*fwu.vhd*	Señal activa en bajo. Indica que se acabó de transmitir una celda completa hacia el protocolo *TOPIA*.

A9	*antes.vhd*	*dram1.vhd, arr_r.vhd*	Esta señal indica cuando está en estado alto, que la primera mitad de la *memoria RAM de doble puerto* está siendo utilizada para guardar la celda que llega del protocolo de comunicación *UTOPIA Level 2* (alguna capa *PHY*); y la segunda mitad de esta memoria está siendo utilizada para transmitir la celda guardada anteriormente hacia el protocolo *TOPIA*. Cuando esta señal está en estado bajo sucede lo contrario.
Cuenta_2	*fwu.vhd*	*dram1.vhd, cont_r.vhd*	Esta señal se utiliza para incrementar la posición de la *memoria RAM de doble puerto* cuando se está guardando en esta la celda proveniente del protocolo de comunicación *UTOPIA Level 2*.
Fin_52_2	*cont_r.vhd*	*fwu.vhd*	Señal activa en bajo que indica el término de la cuenta de 0 a 52. Cuando se activa esta señal indica que se terminó de guardar el último byte de la celada *ATM*.
Ar[5-0]	*cont_r.vhd*	*dram1.vhd*	Este bus maneja valores desde 0 a 52, que es la dirección donde se van guardando los bytes de la celda *ATM* dentro de la *memoria RAM de doble puerto*.
Al[5-0]	*cont_l.vhd*	*dram1.vhd*	Este bus maneja valores desde 0 a 52, que es la dirección donde se leen los bytes de la celda *ATM* dentro de la *memoria RAM de doble puerto*.
Cuenta	*fus.vhd*	*cont_l.vhd*	Esta señal se utiliza para incrementar la posición de la *memoria RAM de doble puerto* cuando se está leyendo en esta memoria la celda hacia el protocolo de comunicación *TOPIA*.
Fin_52	*cont_l.vhd*	*fus.vhd*	Señal activa en bajo que indica el término de la cuenta de 0 a 52. Cuando se activa esta señal indica que se termino de leer el último byte de la celada *ATM*.
R1_R2_R3	*fus.vhd*	*arr_r.vhd*	Esta señal es utilizada para distinguir entre la lectura de los 3 primeros bytes de la celda del protocolo *TOPIA*. *R1a=00*, *R2a=01* y *R3a=10*.
Out_Reg	*fus.vhd*	*arr_r.vhd*	Señal activa en bajo. Esta señal se activa cuando se quiere obtener del bloque *Arreglo de Registros* alguno de los 2 primeros bytes de la celda del protocolo *TOPIA*.
Ruteo_Celda	*fus.vhd*	*arr_r.vhd*	Esta señal sirve para decidir si el dato que se va a enviar hacia el protocolo *TOPIA* es uno de los 3 primeros bytes (enrutado de la *MC*) de la celda de 56 bytes, o es un dato de la celda *ATM* de 53 bytes.
I_O_l[7-0]	*dram1.vhd*	*arr_r.vhd*	Esta señal es un bus de datos de 8 líneas de salida hacia el protocolo *TOPIA*.
In_Reg	*fer.vhd*	*arr_r.vhd*	Señal activa en bajo. Esta señal se activa cuando se quiere guardar en el bloque *Arreglo de Registros* los 2 primeros bytes de la celda del protocolo *TOPIA*.
Dato_R[15-0]	*ITR*	*arr_r.vhd*	Esta señal es un bus de datos en el cual vienen los dos primeros bytes de la celda del protocolo *TOPIA*.

C_H1, C_H2, C_H3	*fer.vhd*	*vpivci.vhd*	Señales activas en bajo que sirven para hacer la carga del dato hacia los registros *R_H1*, *R_H2* y *R_H3* respectivamente.
Peticion_R	*fer.vhd*	*ITR*	Señal activa en bajo que indica que se quiere hacer una petición de lectura hacia la tabla de enrutado.
Listo_R	*ITR*	*fer.vhd*	Señal activa en bajo. Indica cuando se finalizó con la operación de lectura y que el dato leído de la tabla de enrutado ya está disponible en el bus *Dato_R[15-0]*.
Busy_R	*ITR*	*fer.vhd*	Señal activa en bajo. Indica que se encuentra en proceso la lectura de la tabla de enrutado.
Address_R[15-0]	*vpivci.vhd*	*ITR*	Consiste de un bus de 16 líneas que forman la dirección del dato que se va a leer de la tabla de enrutado.
Cuenta_8	*fawu.vhd*	*cont_8.vhd, c_c.vhd*	Esta señal se utiliza para incrementar la dirección del bus *RxAddr* que sirve para hacer el monitoreo de los dispositivos físicos.
Cuenta_35	*fawu.vhd*	*cont_35.vhd*	Señal activa en alto que se utiliza para la cuenta de 0 a 35. Cuando se activa esta señal, comienza el monitoreo de las capas *PHY*, para saber cual de ellas tiene una celda disponible para ser transmitida.
F_Cuenta_8	*cont_8.vhd*	*fawu.vhd*	Señal activa en alto que indica el término de la cuenta de 0-8.
F_Cuenta_35	*cont_35.vhd*	*fawu.vhd*	Señal activa en alto que indica el término de la cuenta de 0 a 35. Cuando se activa esta señal, comienza el monitoreo de las capas *PHY*, para saber cual de ellos tiene una celda disponible para ser transmitida.
Escoge	*fawu.vhd*	*c_c.vhd*	Esta señal cuando está en estado alto indica que el valor de *RxAddr* será un valor entre el 1 y el 8 (8 posibles capas *PHY*); y cuando está en estado bajo, el valor de *RxAddr* será "1Fh" (capa *PHY* nula).

Tabla A - 5 Señales del diseño de la arquitectura de la interfaz de recepción

Código en *VHDL* de los archivos de la interfaz de recepción

Enseguida se tiene un listado de cada uno de los archivos (bloques) de la interfaz de recepción. Estos archivos están codificados en *VHDL*; fueron compilados, simulados y sintetizados en la herramienta *Max2Win* (*Max+Plus II 9.21*) de la compañía *ALTERA*.

ut8.vhd

```
--*************************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Package:                  ut8
--Packages Incluidos:       ut7, y dram1.
--Autor:                    Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                  ut8.vhd
--Fecha de Realizacion:     10/Agosto/1999.
--Fecha de Ultima Revision: 10/Agosto/1999.
--*************************************************************************

library ieee;
use ieee.std_logic_1164.all;

package ut8_package  IS
    component ut7
        port (clk,clk_2,reset,RxSOC,
            Busy_R,Listo_R,PA_E,RxClav:      in  std_logic;
            HA_E,RxEnb,A9,
            Cuenta_2,Peticion_R:             out std_logic;
            A1:                              out std_logic_vector(5 downto 0);
            Ar:                              out std_logic_vector(5 downto 0);
            D_Ent:                           out std_logic_vector(7 downto 0);
            I_O_1:                           in  std_logic_vector(7 downto 0);
            RxAddr:                          out std_logic_vector(4 downto 0);
            RxData:                          in  std_logic_vector(7 downto 0);
            Dato_R:                          in  std_logic_vector(15 downto 0);
            Address_R:                       out std_logic_vector(15 downto 0));
    end component;
    component dram1
        port (clockx2,clk,Cuenta_2,A9:       in  std_logic;
        RxData:                              in  std_logic_vector(7 downto 0);
            A1,Ar:                           in  std_logic_vector(5 downto 0);
            I_O_1:                           out std_logic_vector(7 downto 0));
    end component;
end ut8_package;

library work;
use work.ut8_package.all;
library ieee;
use ieee.std_logic_1164.all;

entity ut8 is
    port (clockx2,clk,clk_2,reset,RxSOC,
        Busy_R,PA_E,RxClav,Listo_R:          in  std_logic;
        HA_E,RxEnb,Peticion_R:               out std_logic;
        D_Ent:                               out std_logic_vector(7 downto 0);
        RxAddr:                              out std_logic_vector(4 downto 0);
        RxData:                              in  std_logic_vector(7 downto 0);
        Dato_R:                              in  std_logic_vector(15 downto 0);
        Address_R:                           out std_logic_vector(15 downto 0));
end ut8;

architecture descripcion_ut8 of ut8 is

signal  A9_signal,Cuenta_2_signal:       std_logic;
signal  A1_signal,Ar_signal:             std_logic_vector(5 downto 0);
signal  I_O_1_signal:                    std_logic_vector(7 downto 0);

begin

    ut7_i:ut7
        port map (clk=>clk,clk_2=>clk_2,reset=>reset,RxSOC=>RxSOC,Busy_R=>Busy_R,
                Listo_R=>Listo_R,PA_E=>PA_E,RxClav=>RxClav,HA_E=>HA_E,RxEnb=>RxEnb,
                A9=>A9_signal,Cuenta_2=>Cuenta_2_signal,Peticion_R=>Peticion_R,
                A1=>A1_signal,Ar=>Ar_signal,D_Ent=>D_Ent,I_O_1=>I_O_1_signal,
                RxAddr=>RxAddr,RxData=>RxData,Dato_R=>Dato_R,Address_R=>Address_R);
```

```
    dram1_i:dram1
        port map (clockx2=>clockx2,clk=>clk,Cuenta_2=>Cuenta_2_signal,A9=>A9_signal,
                RxData=>RxData,A1=>A1_signal,Ar=>Ar_signal,I_O_1=>I_O_1_signal);

end descripcion_ut8;
```

ut7.vhd

```
--*************************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Package:                  ut7
--Packages Incluidos:       ut5 (ut3 y ut4), y ut6 (ut1 y ut2).
--Bloques Incluidos:        FSM Escritura Registros (fer.vhd),
--                          VPI y VCI (vpivci.vhd),
--                          FSM I U/T>>SF (fus.vhd),
--                          Contador_L 0-52 (cont_1.vhd),
--                          Arreglo de Registros (arr_r.vhd),
--                          FSM Aleatorio IWE8>>I U/T (fawu.vhd),
--                          Conteo Ciclico (c_c.vhd),
--                          Contador 0-8 (cont_8.vhd),
--                          Contador 0-36 (cont_36.vhd),
--                          FSM IWE8>>I U/T (fwu.vhd),
--                          Contador_R 0-52 (cont_r.vhd) y
--                          Antes (antes.vhd).
--Autor:                    Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                  ut7.vhd
--Fecha de Realizacion:     10/Agosto/1999.
--Fecha de Ultima Revision: 10/Agosto/1999.
--*************************************************************************

library ieee;
use ieee.std_logic_1164.all;

package ut7_package  IS
    component ut5
        port (clk,reset,RxSOC,Busy_R,Listo_R,
            Fin_Rx,PA_E,A9:                 in  std_logic;
            Fin_Tx,HA_E,Peticion_R:         out std_logic;
            A1:                             out std_logic_vector(5 downto 0);
            D_Ent:                          out std_logic_vector(7 downto 0);
            I_O_1:                          in  std_logic_vector(7 downto 0);
            RxData:                         in  std_logic_vector(7 downto 0);
            Dato_R:                         in  std_logic_vector(15 downto 0);
            Address_R:                      out std_logic_vector(15 downto 0));
    end component;
    component ut6
        port (clk,clk_2,reset,RxSOC,RxClav,
            Fin_Tx:                         in  std_logic;
            RxEnb,A9,Cuenta_2,Fin_Rx:       out std_logic;
            Ar:                             out std_logic_vector(5 downto 0);
            RxAddr:                         out std_logic_vector(4 downto 0));
    end component;
END ut7_package;

library work;
use work.ut7_package.all;
library ieee;
use ieee.std_logic_1164.all;

entity ut7 is
    port (clk,clk_2,reset,RxSOC,Busy_R,
        Listo_R,PA_E,RxClav:            in  std_logic;
        HA_E,RxEnb,A9,Cuenta_2,
        Peticion_R:                     out std_logic;
        A1:                             out std_logic_vector (5 downto 0);
        Ar:                             out std_logic_vector(5 downto 0);
        D_Ent:                          out std_logic_vector (7 downto 0);
        I_O_1:                          in  std_logic_vector (7 downto 0);
        RxAddr:                         out std_logic_vector(4 downto 0);
        RxData:                         in  std_logic_vector (7 downto 0);
```

```
        Dato_R:                                 in  std_logic_vector (15 downto 0);
        Address_R:                              out std_logic_vector (15 downto 0));
end ut7;

architecture descripcion_ut7 of ut7 is

signal  Fin_Rx_signal,Fin_Tx_signal,A9_signal: std_logic;

begin

    ut5_i:ut5
        port map (clk=>clk,reset=>reset,RxSOC=>RxSOC,Busy_R=>Busy_R,Listo_R=>Listo_R,
                RxData=>RxData,Fin_Rx=>Fin_Rx_signal,Fin_Tx=>Fin_Tx_signal,PA_E=>PA_E,
                HA_E=>HA_E,A9=>A9_signal,Al=>Al,D_Ent=>D_Ent,I_O_l=>I_O_l,
                Peticion_R=>Peticion_R,Dato_R=>Dato_R,Address_R=>Address_R);

    ut6_i:ut6
        port map (clk=>clk,clk_2=>clk_2,reset=>reset,RxSOC=>RxSOC,RxClav=>RxClav,
                RxEnb=>RxEnb,A9=>A9_signal,Fin_Tx=>Fin_Tx_signal,
                Fin_Rx=>Fin_Rx_signal,Cuenta_2=>Cuenta_2,RxAddr=>RxAddr,Ar=>Ar);

    A9<=A9_signal;

end descripcion_ut7;
```

dram1.vhd

```
library ieee;
use ieee.std_logic_1164.all;
library lpm;
use lpm.lpm_components.ALL;

package dram1_package  IS
    component csdpram
        generic (LPM_WIDTH: INTEGER := 8;
            LPM_WIDTHAD:    INTEGER := 7);
            --LPM_NUMWORDS: STRING := "UNUSED";
            --FILE:         STRING := "UNUSED");
        port (dataa, datab:             in  std_logic_vector(LPM_WIDTH-1 downto 0);
            addressa, addressb:         in  std_logic_vector(LPM_WIDTHAD-1 downto 0);
            clock, clockx2, wea, web:   in  std_logic;
            qa, qb:                     out std_logic_vector(LPM_WIDTH-1 downto 0);
            busy:                       out std_logic);
    end component;
end dram1_package;

library work;
use work.dram1_package.all;
library ieee;
use ieee.std_logic_1164.all;
LIBRARY lpm;
USE lpm.lpm_components.ALL;

entity dram1 is
    port (clockx2,clk,Cuenta_2,A9:      in  std_logic;
    RxData:                             in  std_logic_vector(7 downto 0);
        Al,Ar:                          in  std_logic_vector(5 downto 0);
        I_O_l:                          out std_logic_vector(7 downto 0));
end dram1;
```

ut5.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Package:                  ut5
--Packages Incluidos:       ut3 y ut4.
--Bloques Incluidos:        FSM Escritura Registros (fer.vhd),
--                          VPI y VCI (vpivci.vhd),
--                          FSM I U/T>>SF (fus.vhd),
```

```
--                          Contador_L 0-52 (cont_l.vhd) y
--                          Arreglo de Registros (arr_r.vhd).
--Autor:                   Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                 ut5.vhd
--Fecha de Realizacion:    10/Agosto/1999.
--Fecha de Ultima Revision: 10/Agosto/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;

package ut5_package  IS
    component ut3
        port (clk,reset,RxSOC,Busy_R,Listo_R:    in  std_logic;
            In_Reg,Peticion_R:                   out std_logic;
            RxData:                              in  std_logic_vector (7 downto 0);
            Address_R:                           out std_logic_vector (15 downto 0));
    end component;
    component ut4
        port (clk,reset,Fin_Rx,PA_E,In_Reg,A9:   in  std_logic;
            HA_E,Fin_Tx:                         out std_logic;
            Dato_R:                              in  std_logic_vector (15 downto 0);
            I_O_1:                               in  std_logic_vector (7 downto 0);
            D_Ent:                               out std_logic_vector (7 downto 0);
            Al:                                  out std_logic_vector (5 downto 0));
    end component;
end ut5_package;

library work;
use work.ut5_package.all;
library ieee;
use ieee.std_logic_1164.all;

entity ut5 is
    port (clk,reset,RxSOC,Busy_R,Listo_R,
        Fin_Rx,PA_E,A9:                      in  std_logic;
        Fin_Tx,HA_E,Peticion_R:              out std_logic;
        Al:                                  out std_logic_vector (5 downto 0);
        D_Ent:                               out std_logic_vector (7 downto 0);
        I_O_1:                               in  std_logic_vector (7 downto 0);
        RxData:                              in  std_logic_vector (7 downto 0);
        Dato_R:                              in  std_logic_vector (15 downto 0);
        Address_R:                           out std_logic_vector (15 downto 0));
end ut5;

architecture descripcion_ut5 of ut5 is

signal  In_Reg_signal: std_logic;

begin

    ut3_i:ut3
        port map (clk=>clk,reset=>reset,RxSOC=>RxSOC,RxData=>RxData,In_Reg=>In_Reg_signal,
            Busy_R=>Busy_R,Listo_R=>Listo_R,Peticion_R=>Peticion_R,Address_R=>Address_R);

    ut4_i:ut4
        port map (clk=>clk,reset=>reset,Fin_Rx=>Fin_Rx,PA_E=>PA_E,In_Reg=>In_Reg_signal,

    A9=>A9,HA_E=>HA_E,Fin_Tx=>Fin_Tx,Dato_R=>Dato_R,I_O_1=>I_O_1,D_Ent=>D_Ent,Al=>Al);

end descripcion_ut5;
```

ut6.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Package:                  ut6
--Packages Incluidos:       ut1 y ut2.
--Bloques Incluidos:        FSM Aleatorio IWE8>>I U/T (fawu.vhd),
```

```
--                          Conteo Ciclico (c_c.vhd),
--                          Contador 0-8 (cont_8.vhd),
--                          Contador 0-36 (cont_36.vhd),
--                          FSM IWE8>>I U/T (fwu.vhd),
--                          Contador_R 0-52 (cont_r.vhd) y
--                          Antes (antes.vhd).
--Autor:                    Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                  ut6.vhd
--Fecha de Realizacion:     10/Agosto/1999.
--Fecha de Ultima Revision: 10/Agosto/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;

package ut6_package  is
    component ut1
        port (clk,clk_2,reset,RxClav,RxSOC,Fin_Tx:    in  std_logic;
            RxAddr:                                   out std_logic_vector(4 downto 0);
            RxEnb:                                    out std_logic);
    end component;
    component ut2
        port (clk,reset,Fin_Tx,RxSOC:                 in  std_logic;
            Ar:                                       out std_logic_vector(5 downto 0);
            A9,Cuenta_2,Fin_Rx:                       out std_logic);
    end component;
end ut6_package;

library work;
use work.ut6_package.all;
library ieee;
use ieee.std_logic_1164.all;

entity ut6 is
    port (clk,clk_2,reset,RxSOC,RxClav,Fin_Tx:        in  std_logic;
        RxEnb,A9,Cuenta_2,Fin_Rx:                     out std_logic;
        Ar:                                           out std_logic_vector(5 downto 0);
        RxAddr:                                       out std_logic_vector(4 downto 0));
end ut6;

architecture descripcion_ut6 of ut6 is

begin

    ut1_i:ut1
        port map (clk=>clk,clk_2=>clk_2,reset=>reset,RxSOC=>RxSOC,RxClav=>RxClav,
                Fin_Tx=>Fin_Tx,RxAddr=>RxAddr,RxEnb=>RxEnb);

    ut2_i:ut2
        port map (clk=>clk,reset=>reset,Fin_Rx=>Fin_Rx,Fin_Tx=>Fin_Tx,
                RxSOC=>RxSOC,Ar=>Ar,A9=>A9,Cuenta_2=>Cuenta_2);

end descripcion_ut6;
```

ut3.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Package:                  ut3
--Bloques Incluidos:        FSM Escritura Registros (fer.vhd) y
--                          VPI y VCI (vpivci.vhd).
--Autor:                    Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                  ut3.vhd
--Fecha de Realizacion:     10/Agosto/1999.
--Fecha de Ultima Revision: 10/Agosto/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;
```

```
package ut3_package  IS
    component fer
        port (Busy_R,Listo_R,RxSOC,clk,reset:     in  std_logic;
            C_H1,C_H2,C_H3,In_Reg,Peticion_R:     out std_logic);
    end component;
    component vpivci
        port (C_H1,C_H2,C_H3,clk:                 in  std_logic;
            RxData:                               in  std_logic_vector (7 downto 0);
            Address_R:                            out std_logic_vector (15 downto 0));
    end component;
END ut3_package;

library work;
use work.ut3_package.all;
library ieee;
use ieee.std_logic_1164.all;

entity ut3 is
    port (clk,reset,RxSOC,Busy_R,Listo_R:         in  std_logic;
        In_Reg,Peticion_R:                        out std_logic;
        RxData:                                   in  std_logic_vector (7 downto 0);
        Address_R:                                out std_logic_vector (15 downto 0));
end ut3;

architecture descripcion_ut3 of ut3 is

signal  C_H1_signal,C_H2_signal,C_H3_signal: std_logic;

begin

    fer_i:fer
        port map (Busy_R=>Busy_R,Listo_R=>Listo_R,clk=>clk,reset=>reset,
                RxSOC=>RxSOC,C_H1=>C_H1_signal,C_H2=>C_H2_signal,C_H3=>C_H3_signal,
                In_Reg=>In_Reg,  Peticion_R=>Peticion_R);

    vpivci_i:vpivci
        port map (clk=>clk,C_H1=>C_H1_signal,C_H2=>C_H2_signal,C_H3=>C_H3_signal,
                RxData=>RxData,Address_R=>Address_R);

end descripcion_ut3;
```

ut4.vhd

```
--*********************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Package:                  ut4
--Bloques Incluidos:        FSM I U/T>>SF (fus.vhd),
--                          Contador_L 0-52 (cont_1.vhd) y
--                          Arreglo de Registros (arr_1.vhd).
--Autor:                    Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                  ut4.vhd
--Fecha de Realizacion:     10/Agosto/1999.
--Fecha de Ultima Revision: 10/Agosto/1999.
--*********************************************************************

library ieee;
use ieee.std_logic_1164.all;

package ut4_package  IS
    component fus
        port (Fin_Rx,PA_E,Fin_52,clk,reset:          in  std_logic;
            HA_E,Cuenta,Out_Reg,Ruteo_Celda,Fin_Tx:  out std_logic;
            R1_R2_R3:                                out std_logic_vector(1 downto 0));
    end component;
    component cont_1
        port (Cuenta,reset,clk:      in  std_logic;
            Fin_52:                  out std_logic;
            A1:                      out std_logic_vector (5 downto 0));
```

```
    end component;
    component arr_r
        port (In_Reg,Out_Reg,A9,Ruteo_Celda:      in  std_logic;
            R1_R2_R3:                             in  std_logic_vector (1 downto 0);
            Dato_R:                               in  std_logic_vector (15 downto 0);
            I_O_l:                                in  std_logic_vector (7 downto 0);
            D_Ent:                                out std_logic_vector (7 downto 0));
    end component;
end ut4_package;

library work;
use work.ut4_package.all;
library ieee;
use ieee.std_logic_1164.all;

entity ut4 is
    port (clk,reset,Fin_Rx,PA_E,In_Reg,A9:    in  std_logic;
        HA_E,Fin_Tx:                          out std_logic;
        Dato_R:                               in  std_logic_vector (15 downto 0);
        I_O_l:                                in  std_logic_vector (7 downto 0);
        D_Ent:                                out std_logic_vector (7 downto 0);
        Al:                                   out std_logic_vector (5 downto 0));
end ut4;

architecture descripcion_ut4 of ut4 is

signal  Fin_52_signal,Cuenta_signal,Out_Reg_signal,Ruteo_Celda_signal: std_logic;
signal  R1_R2_R3_signal: std_logic_vector (1 downto 0);
begin

    fus_i:fus
        port map (clk=>clk,reset=>reset,Fin_Rx=>Fin_Rx,PA_E=>PA_E,
                Fin_52=>Fin_52_signal,HA_E=>HA_E,Cuenta=>Cuenta_signal,
                Out_Reg=>Out_Reg_signal,Ruteo_Celda=>Ruteo_Celda_signal,
                Fin_Tx=>Fin_Tx,R1_R2_R3=>R1_R2_R3_signal);

    cont_l_i:cont_l
        port map (clk=>clk,reset=>reset,Cuenta=>Cuenta_signal,
                Fin_52=>Fin_52_signal,Al=>Al);

    arr_r_i:arr_r
        port map (In_Reg=>In_Reg,Out_Reg=>Out_Reg_signal,A9=>A9,
                Ruteo_Celda=>Ruteo_Celda_signal,R1_R2_R3=>R1_R2_R3_signal,
                Dato_R=>Dato_R,I_O_l=>I_O_l,D_Ent=>D_Ent);

end descripcion_ut4;
```

ut1.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Package:                  ut1
--Bloques Incluidos:        FSM Aleatorio IWE8>>I U/T (fawu.vhd),
--                          Conteo Ciclico (c_c.vhd),
--                          Contador 0-8 (cont_8.vhd) y
--                          Contador 0-35 (cont_35.vhd).
--Autor:                    Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                  ut1.vhd
--Fecha de Realizacion:     10/Agosto/1999.
--Fecha de Ultima Revision: 10/Agosto/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;

package ut1_package  IS
    component fawu
        port (Fin_Tx,F_Cuenta_8,RxClav,F_Cuenta_35,
            RxSOC,clk,clk_2,reset:                    in  std_logic;
```

```
            Cuenta_35,Cuenta_8,RxEnb,C_C_Cuenta_8,resta,
            Inicia_Cuenta_8:                          out std_logic;
            Escoge:                                   out std_logic_vector (1 downto
0));
    end component;
    component c_c
        port (C_C_Cuenta_8,RxClav,
            reset,clk,clk_2,resta:      in  std_logic;
            Escoge:                     in  std_logic_vector (1 downto 0);
            RxAddr:                     out std_logic_vector (4 downto 0));
    end component;
    component cont_8
        port (Cuenta_8,reset,clk_2,Inicia_Cuenta_8:      in  std_logic;
            F_Cuenta_8:                                  out std_logic);
    end component;
    component cont_35
        port (Cuenta_35,reset,clk:      in  std_logic;
            F_Cuenta_35:                out std_logic);
    end component;
end ut1_package;

library work;
use work.ut1_package.all;
library ieee;
use ieee.std_logic_1164.all;

entity ut1 is
    port (clk,clk_2,reset,RxClav,RxSOC,Fin_Tx:    in  std_logic;
        RxAddr:                                   out std_logic_vector(4 downto 0);
        RxEnb:                                    out std_logic);
end ut1;

architecture descripcion_ut1 of ut1 is

signal  F_Cuenta_8_signal,F_Cuenta_35_signal,Cuenta_35_signal,Cuenta_8_signal,
        C_C_Cuenta_8_signal,resta_signal,
        Inicia_Cuenta_8_signal: std_logic;
signal  Escoge_signal: std_logic_vector (1 downto 0);

begin

    fawu_i:fawu
        port map (Fin_Tx=>Fin_Tx,RxClav=>RxClav,clk=>clk,clk_2=>clk_2,reset=>reset,
                F_Cuenta_8=>F_Cuenta_8_signal,F_Cuenta_35=>F_Cuenta_35_signal,
                RxSOC=>RxSOC,Cuenta_35=>Cuenta_35_signal,Cuenta_8=>Cuenta_8_signal,
                Escoge=>Escoge_signal,RxEnb=>RxEnb,C_C_Cuenta_8=>C_C_Cuenta_8_signal,
                resta=>resta_signal,Inicia_Cuenta_8=>Inicia_Cuenta_8_signal);

    c_c_i:c_c
        port map (C_C_Cuenta_8=>C_C_Cuenta_8_signal,Escoge=>Escoge_signal,clk=>clk,
                reset=>reset,RxClav=>RxClav,RxAddr=>RxAddr,
                resta=>resta_signal,clk_2=>clk_2);

    cont_8_i:cont_8
        port map (Cuenta_8=>Cuenta_8_signal,F_Cuenta_8=>F_Cuenta_8_signal,
                clk_2=>clk_2,reset=>reset,
                Inicia_Cuenta_8=>Inicia_Cuenta_8_signal);

    cont_35_i:cont_35
        port map (Cuenta_35=>Cuenta_35_signal,F_Cuenta_35=>F_Cuenta_35_signal,
                clk=>clk,reset=>reset);

end descripcion_ut1;
```

ut2.vhd

```
--*********************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Package:                    ut2
```

```
--Bloques Incluidos:     FSM IWE8>>I U/T (fwu.vhd),
--                           Contador_R 0-52 (cont_r.vhd) y
--                           Antes (antes.vhd).
--Autor:                   Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                 ut2.vhd
--Fecha de Realizacion:    10/Agosto/1999.
--Fecha de Ultima Revision: 10/Agosto/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;

package ut2_package  IS
    component fwu
        port (Fin_Tx,Fin_35_2,Fin_52_2,RxSOC,clk,reset:       in  std_logic;
            cambia,Cuenta_2,Fin_Rx:                           out std_logic);
    end component;
    component cont_r
        port (Cuenta_2,reset,clk:    in  std_logic;
            Fin_35_2,Fin_52_2:       out std_logic;
            Ar:                      out std_logic_vector (5 downto 0));
    end component;
    component antes
        port (cambia,reset,clk:      in  std_logic;
            A9:                      out std_logic);
    end component;
end ut2_package;

library work;
use work.ut2_package.all;
library ieee;
use ieee.std_logic_1164.all;

entity ut2 is
    port (clk,reset,Fin_Tx,RxSOC:            in  std_logic;
        Ar:                                  out std_logic_vector(5 downto 0);
        A9,Cuenta_2,Fin_Rx:                  out std_logic);
end ut2;

architecture descripcion_ut2 of ut2 is

signal  Fin_35_2_signal,Fin_52_2_signal,cambia_signal,
        Cuenta_2_signal: std_logic;

begin

    fwu_i:fwu
        port map (Fin_Tx=>Fin_Tx,Fin_35_2=>Fin_35_2_signal,clk=>clk,reset=>reset,
                Fin_52_2=>Fin_52_2_signal,RxSOC=>RxSOC,cambia=>cambia_signal,
                Cuenta_2=>Cuenta_2_signal,Fin_Rx=>Fin_Rx);

    cont_r_i:cont_r
        port map (Cuenta_2=>Cuenta_2_signal,clk=>clk,reset=>reset,
                Fin_35_2=>Fin_35_2_signal,Fin_52_2=>Fin_52_2_signal,Ar=>Ar);

    antes_i:antes
        port map (cambia=>cambia_signal,clk=>clk,reset=>reset,A9=>A9);

    Cuenta_2<=Cuenta_2_signal;
end descripcion_ut2;
```

fer.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                  FSM Escritura Registros
--Autor:                   Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                 fer.vhd
--Fecha de Realizacion:    30/Junio/1999.
```

```
--Fecha de Ultima Revision:  07/Julio/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;

entity fer is
    port (Busy_R,Listo_R,RxSOC,clk,reset:        in  std_logic;
        C_H1,C_H2,C_H3,In_Reg,Peticion_R:        out std_logic);
end fer;

architecture descripcion_fer of fer is
    type estado is (S1,S2,S3,S4,S5,S6,S7);
    signal presente: estado:=S1;

begin
    maquina_fer:
    process(clk,reset)
    begin
    if reset='1' then
        presente<=S1;
    elsif (clk='1' and clk'event) then
        case presente is
            when S1=>
                if (RxSOC='1') then
                    presente<=S2;
                end if;
            when S2=>
                presente<=S3;
            when S3=>
                presente<=S4;
            when S4=>
                presente<=S5;
            when S5=>
                if (Busy_R='0') then
                    presente<=S6;
                end if;
            when S6=>
                if (Listo_R='0') then
                    presente<=S7;
                end if;
            when S7=>
                presente<=S1;
        end case;
    end if;
    end process maquina_fer;

    salida_fer:
    process(presente)
    begin
    case presente is
        when S1=>
            C_H1<='1';
            C_H2<='1';
            C_H3<='1';
            In_Reg<='1';
            Peticion_R<='1';
        when S2=>
            C_H1<='0';
            C_H2<='1';
            C_H3<='1';
            In_Reg<='1';
            Peticion_R<='1';
        when S3=>
            C_H1<='1';
            C_H2<='0';
            C_H3<='1';
            In_Reg<='1';
            Peticion_R<='1';
        when S4=>
            C_H1<='1';
```

```
            C_H2<='1';
            C_H3<='0';
            In_Reg<='1';
            Peticion_R<='1';
        when S5=>
            C_H1<='1';
            C_H2<='1';
            C_H3<='1';
            In_Reg<='1':
            Peticion_R<='0';
        when S6=>
            C_H1<='1';
            C_H2<='1';
            C_H3<='1';
            In_Reg<='1';
            Peticion_R<='1';
        when S7=>
            C_H1<='1';
            C_H2<='1';
            C_H3<='1';
            In_Reg<='0':
            Peticion_R<='1';
    end case;
    end process salida_fer;

end descripcion_fer;
```

vpivci.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    VPI y VCI
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   vpivci.vhd
--Fecha de Realizacion:      01/Julio/1999.
--Fecha de Ultima Revision:  07/Julio/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

entity vpivci is
    port (C_H1,C_H2,C_H3,clk:    in  std_logic;
        RxData:                  in  std_logic_vector (7 downto 0);
        Address_R:               out std_logic_vector (15 downto 0));
    end vpivci;

architecture descripcion_vpivci of vpivci is
signal RxData_aux:   std_logic_vector (7 downto 0);

begin
    retraso:
    process (clk)
    begin
        if (clk'EVENT and clk = '1') then
            RxData_aux<=RxData;
        end if;
    end process retraso;

    asignar:
    process (clk)
    begin
        if (clk'EVENT and clk = '0') then
            if (C_H3='0') then
                Address_R(7 downto 0)<=RxData_aux;
            end if;
            if (C_H1='0') then
```

```
                Address_R(15 downto 12)<=RxData_aux(3 downto 0);
            end if;
            if (C_H2='0') then
                Address_R(11 downto 8)<=RxData_aux(7 downto 4);
            end if;
        end if;
    end process asignar;

end descripcion_vpivci;
```

fus.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    FSM I U/T>>SF
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   fus.vhd
--Fecha de Realizacion:      30/Junio/1999.
--Fecha de Ultima Revision:  26/Julio/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;

entity fus is
    port (Fin_Rx,PA_E,Fin_52,clk,reset:          in  std_logic;
        HA_E,Cuenta,Out_Reg,Ruteo_Celda,Fin_Tx:  out std_logic;
        R1_R2_R3:                                out std_logic_vector(1 downto 0));
end fus;

architecture descripcion_fus of fus is
    type estado is (S1,S2,S3,S4,S5,S6,S7);
    signal presente: estado:=S1;

begin
    maquina_fus:
    process(clk,reset)
    begin
    if reset='1' then
        presente<=S1;
    elsif (clk='1' and clk'event) then
        case presente is
            when S1=>
                if (Fin_Rx='0' and PA_E='1') then
                    presente<=S3;
                elsif (Fin_Rx='0' and PA_E='0') then
                    presente<=S2;
                end if;
            when S2=>
                if (PA_E='1') then
                    presente<=S3;
                end if;
            when S3=>
                presente<=S4;
            when S4=>
                presente<=S5;
            when S5=>
                presente<=S6;
            when S6=>
                if (Fin_52='0') then
                    presente<=S7;
                end if;
            when S7=>
                presente<=S1;
        end case;
    end if;
    end process maquina_fus;

    salida_fus:
```

```
    process(presente)
    begin
    case presente is
        when S1=>
            HA_E<='1';
            Cuenta<='1';
            Out_Reg<='1';
            R1_R2_R3<="01";
            Ruteo_Celda<='1';
            Fin_Tx<='0';
        when S2=>
            HA_E<='1';
            Cuenta<='1';
            Out_Reg<='1';
            R1_R2_R3<="01";
            Ruteo_Celda<='1';
            Fin_Tx<='0';
        when S3=>
            HA_E<='0';
            Cuenta<='1';
            Out_Reg<='0';
            R1_R2_R3<="01";
            Ruteo_Celda<='1';
            Fin_Tx<='0';
        when S4=>
            HA_E<='1';
            Cuenta<='1';
            Out_Reg<='1';
            R1_R2_R3<="00";
            Ruteo_Celda<='1';
            Fin_Tx<='1';
        when S5=>
            HA_E<='1';
            Cuenta<='0';
            Out_Reg<='1';
            R1_R2_R3<="11";
            Ruteo_Celda<='1';
            Fin_Tx<='1';
        when S6=>
            HA_E<='1';
            Cuenta<='0';
            Out_Reg<='1';
            R1_R2_R3<="01";
            Ruteo_Celda<='0';
            Fin_Tx<='1';
        when S7=>
            HA_E<='1';
            Cuenta<='1';
            Out_Reg<='1';
            R1_R2_R3<="01";
            Ruteo_Celda<='0';
            Fin_Tx<='1';
    end case;
    end process salida_fus;

end descripcion_fus;
```

cont_1.vhd

```
--*********************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    Contador_L 0-52
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   cont_1.vhd
--Fecha de Realizacion:      08/Julio/1999.
--Fecha de Ultima Revision:  08/Julio/1999.
--*********************************************************************

library ieee;
```

```
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

entity cont_1 is
    port (Cuenta,reset,clk:      in  std_logic;
          Fin_52:                out std_logic;
          A1:                    out std_logic_vector (5 downto 0));
end cont_1;

architecture descripcion_cont_1 of cont_1 is

begin
    process (clk)
        variable cnt: integer range 0 to 52;
    begin
        if (reset='1') then
            cnt:=0;
            Fin_52<='1';
        elsif (clk'EVENT and clk='1') then
            if (Cuenta='0') then
                if (cnt=52) then
                    cnt:=0;
                    Fin_52<='1';
                elsif (cnt=51) then
                        cnt:=cnt+1;
                        Fin_52<='0';
                    else
                        cnt:=cnt+1;
                        Fin_52<='1';
                end if;
            end if;
        end if;
        A1<=CONV_STD_LOGIC_VECTOR(cnt,6);
    end process;
end descripcion_cont_1;
```

arr_r.vhd

```
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                  Arreglo de Registros
--Autor:                   Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                 arr_r.vhd
--Fecha de Realizacion:    08/Julio/1999.
--Fecha de Ultima Revision: 08/Julio/1999.
--****************************************************************

library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

entity arr_1 is
    port (In_Reg,Out_Reg,A9,Ruteo_Celda:        in  std_logic;
          R1_R2_R3:                             in  std_logic_vector (1 downto 0);
          Dato_R:                               in  std_logic_vector (15 downto 0);
          I_O_1:                                in  std_logic_vector (7 downto 0);
          D_Ent:                                out std_logic_vector (7 downto 0));
    end arr_r;

architecture descripcion_arr_r of arr_1 is

signal  Dato_1_signal,Dato_2_signal,
        Sal_Reg_signal:                 std_logic_vector (15 downto 0);
signal  Ruteo_signal:                   std_logic_vector (7 downto 0);
signal  Reg_1,Reg_2:                    std_logic_vector (15 downto 0);

begin
    Reg_1<=Dato_R when (A9='1' and In_Reg='0') else Reg_1;
    Reg_2<=Dato_R when (A9='0' and In_Reg='0') else Reg_2;
    Dato_1_signal<=Reg_1 when (A9='0' and Out_Reg='0') else Dato_1_signal;
```

```
    Dato_2_signal<=Reg_2 when (A9='1' and Out_Reg='0') else Dato_2_signal;

    Sal_Reg_signal<=Dato_1_signal when (A9='0') else Dato_2_signal;

    Ruteo_signal<=  Sal_Reg_signal(15 downto 8) when (R1_R2_R3="01") else
                    Sal_Reg_signal(7 downto 0) when (R1_R2_R3="00") else
                    "00000101";

    D_Ent<=Ruteo_signal when (Ruteo_Celda='1') else I_O_1;

end descripcion_arr_r;
```

fawu.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    FSM Aleatorio IWE8>>I U/T
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   fawu.vhd
--Fecha de Realizacion:      30/Junio/1999.
--Fecha de Ultima Revision:  26/Julio/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;

entity fawu is
    port (Fin_Tx,F_Cuenta_8,RxClav,F_Cuenta_35,
        RxSOC,clk,clk_2,reset:                          in  std_logic;
        Cuenta_35,Cuenta_8,RxEnb,C_C_Cuenta_8,resta,
        Inicia_Cuenta_8:                                out std_logic;
        Escoge:                                         out std_logic_vector (1 downto 0));
end fawu;

architecture descripcion_fawu of fawu is
    type    estado is (S1,S2,S3,S4,S42,S5,S52,S6,S7,S8,S9,S10,S11);
    signal  presente: estado:=S1;
    signal  clk_2_aux:  std_logic;
begin
    maquina_fawu:
    process(clk,reset)
    begin
    if reset='1' then
        presente<=S1;
    elsif (clk='1' and clk'event) then
        case presente is
            when S1=>
                if (Fin_Tx='0') then
                    presente<=S2;
                end if;
            when S2=>
                if (RxClav='1') then
                    presente<=S3;
                end if;
            when S3=>
                presente<=S4;
            when S4=>
                if (RxSOC='1') then
                    presente<=S5;
                else
                    presente<=S42;
                end if;
            when S42=>
                if (RxSOC='1') then
                    presente<=S5;
                else
                    presente<=S3;
                end if;
```

```
        when S5=>
            if (F_Cuenta_35='1') then
                if (clk_2_aux='1') then
                    presente<=S6;
                else
                    presente<=S52;
                end if;
            end if;
        when S52=>
            presente<=S6;
        when S6=>
            if (RxClav='1') then
                presente<=S8;
            else
                presente<=S7;
            end if;
        when S7=>
            if (F_Cuenta_8='0') then
                if (RxClav='1') then
                    presente<=S8;
                end if;
            elsif (RxClav='0') then
                    presente<=S10;
                elsif (Fin_Tx='1') then
                        presente<=S9;
                    elsif (clk_2_aux='1') then
                        presente<=S3;
            end if;
        when S8=>
            if (F_Cuenta_8='1') then
                if (Fin_Tx='0') then
                    presente<=S3;
                else
                    presente<=S11;
                end if;
            end if;
        when S9=>
            if (Fin_Tx='0' and clk_2_aux='1') then
                presente<=S3;
            end if;
        when S10=>
            if (RxClav='1') then
                if (Fin_Tx='0' and clk_2_aux='1') then
                    presente<=S3;
                else
                    presente<=S11;
                end if;
            end if;
        when S11=>
            if (Fin_Tx='0' and clk_2_aux='1') then
                presente<=S3;
            end if;
        when OTHERS=>
            presente<=S1;
    end case;
end if;
end process maquina_fawu;

salida_fawu:
process(presente)
begin
case presente is
    when S1=>
        C_C_Cuenta_8<='0';
        Escoge<="11";
        RxEnb<='1';
        Cuenta_8<='0';
        Cuenta_35<='0';
        resta<='0':
        Inicia_Cuenta_8<='0':
    when S2=>
```

```
        C_C_Cuenta_8<='1';
        Escoge<="01";
        RxEnb<='1';
        Cuenta_8<='0';
        Cuenta_35<='0';
        resta<='0';
        Inicia_Cuenta_8<='0';
    when S3=>
        C_C_Cuenta_8<='0';
        Escoge<="00";
        RxEnb<='1';
        Cuenta_8<='0';
        Cuenta_35<='0';
        resta<='0';
        Inicia_Cuenta_8<='0';
    when S4=>
        C_C_Cuenta_8<='0';
        Escoge<="11";
        RxEnb<='0';
        Cuenta_8<='0';
        Cuenta_35<='0';
        resta<='0';
        Inicia_Cuenta_8<='0';
    when S42=>
        C_C_Cuenta_8<='0';
        Escoge<="11";
        RxEnb<='0';
        Cuenta_8<='0';
        Cuenta_35<='0';
        resta<='0';
        Inicia_Cuenta_8<='0';
    when S5=>
        C_C_Cuenta_8<='0';
        Escoge<="11";
        RxEnb<='0';
        Cuenta_8<='0';
        Cuenta_35<='1';
        resta<='1';
        Inicia_Cuenta_8<='1';
    when S52=>
        C_C_Cuenta_8<='0';
        Escoge<="11";
        RxEnb<='0';
        Cuenta_8<='0';
        Cuenta_35<='0';
        resta<='1';
        Inicia_Cuenta_8<='1';
    when S6=>
        C_C_Cuenta_8<='0';
        Escoge<="01";
        RxEnb<='0';
        Cuenta_8<='1';
        Cuenta_35<='0';
        resta<='0';
        Inicia_Cuenta_8<='0';
    when S7=>
        C_C_Cuenta_8<='1';
        Escoge<="01";
        RxEnb<='0';
        Cuenta_8<='1';
        Cuenta_35<='0';
        resta<='0';
        Inicia_Cuenta_8<='0';
    when S8=>
        C_C_Cuenta_8<='0';
        Escoge<="11";
        RxEnb<='0';
        Cuenta_8<='1';
        Cuenta_35<='0';
        resta<='0';
        Inicia_Cuenta_8<='0';
```

```
        when S9=>
            C_C_Cuenta_8<='0';
            Escoge<="11";
            RxEnb<='1';
            Cuenta_8<='1';
            Cuenta_35<='0';
            resta<='0';
            Inicia_Cuenta_8<='0';
        when S10=>
            C_C_Cuenta_8<='1';
            Escoge<="01";
            RxEnb<='1';
            Cuenta_8<='0';
            Cuenta_35<='0';
            resta<='0';
            Inicia_Cuenta_8<='0';
        when S11=>
            C_C_Cuenta_8<='0';
            Escoge<="11";
            RxEnb<='1';
            Cuenta_8<='0';
            Cuenta_35<='0';
            resta<='0';
            Inicia_Cuenta_8<='0';
    end case;
    end process salida_fawu;

    sincro:
    process (clk)
    begin
        if (clk='0' and clk'EVENT) then
            clk_2_aux<=clk_2;
        end if;
    end process sincro;

end descripcion_fawu;
```

c_c.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    Conteo Ciclico
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   c_c.vhd
--Fecha de Realizacion:      01/Julio/1999.
--Fecha de Ultima Revision:  20/Agosto/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity c_c is
    port (C_C_Cuenta_8,RxClav,
        reset,clk,clk_2,resta:      in  std_logic;
        Escoge:                     in  std_logic_vector (1 downto 0);
        RxAddr:                     out std_logic_vector (4 downto 0));
    end c_c;

architecture descripcion_c_c of c_c is

signal  sal1:                   std_logic_vector (4 downto 0);
signal  Addr,Addr_a1,sal2:      std_logic_vector (2 downto 0);

begin

    registro_1:
    process (clk)
    begin
```

```
        if (clk'EVENT and clk='1') then
            if (RxClav='0') then
                Addr_a1<=Addr;
            end if;
        end if;
    end process registro_1;

    registro_2:
    process (clk)
    begin
        if (clk'EVENT and clk='1') then
            if (RxClav='1') then
                sal2<=Addr_a1;
            end if;
        end if;
    end process registro_2;


    conteo:
    process (clk_2)
        variable cnt: integer range 1 to 8;
    begin
        if (reset='1') then
            cnt:=8;
        elsif (clk_2'EVENT and clk_2='0') then
            if (C_C_Cuenta_8='1') then
                if (resta='0') then
                    if cnt=8 then
                        cnt:=1;
                    else
                        cnt:=cnt+1;
                    end if;
                end if;
                Addr<=CONV_STD_LOGIC_VECTOR(cnt,3);
            end if;
        end if;
    end process conteo;


    sal1 <= "11111" when (clk_2='1') else
            "00" & Addr when (clk_2='0') else
            "11111";

    RxAddr<="00" & sal2 when (Escoge="00") else
            sal1 when (Escoge="01")  else
            "11111";

end descripcion_c_c;
```

cont_8.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    Contador 0-8
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   cont_8.vhd
--Fecha de Realizacion:      01/Julio/1999.
--Fecha de Ultima Revision:  05/Julio/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

entity cont_8 is
    port (Cuenta_8,reset,clk_2,Inicia_Cuenta_8:      in  std_logic;
          F_Cuenta_8:                                out std_logic);
end cont_8;
```

```
architecture descripcion_cont_8 of cont_8 is

begin

    process (clk_2)
        variable cnt: integer range 0 to 8;
    begin
        if (reset='1') then
            cnt:=0;
        elsif (clk_2'EVENT and clk_2='1') then
            if (Inicia_Cuenta_8='1') then
                cnt:=0;
                F_Cuenta_8<='0';
            elsif (Cuenta_8='1') then
                if (cnt=8) then
                    cnt:=0;
                    F_Cuenta_8<='0';
                elsif (cnt=7) then
                        cnt:=cnt+1;
                        F_Cuenta_8<='1';
                    else
                        cnt:=cnt+1;
                        F_Cuenta_8<='0';
                end if;
            end if;
        end if;
    end process;
end descripcion_cont_8;
```

cont_35.vhd

```
--*****************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    Contador 0-35
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   cont_35.vhd
--Fecha de Realizacion:      01/Julio/1999.
--Fecha de Ultima Revision:  05/Julio/1999.
--*****************************************************************

library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

entity cont_35 is
    port (Cuenta_35,reset,clk:   in  std_logic;
        F_Cuenta_35:             out std_logic);
end cont_35;

architecture descripcion_cont_35 of cont_35 is

begin
    process (clk)
        variable cnt: integer range 0 to 35;
    begin
        if (reset='1') then
            cnt:=1;
        elsif (clk'EVENT and clk='1') then
            if (Cuenta_35='1') then
                if (cnt=35) then
                    cnt:=1;
                    F_Cuenta_35<='0';
                elsif (cnt=34) then
                        cnt:=cnt+1;
                        F_Cuenta_35<='1';
                    else
                        cnt:=cnt+1;
                        F_Cuenta_35<='0';
                end if;
```

```
            end if;
        end if;
    end process;
end descripcion_cont_35;
```

fwu.vhd

```
--*********************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    FSM IWE8>>I U/T
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   fwu.vhd
--Fecha de Realizacion:      07/Julio/1999.
--Fecha de Ultima Revision:  26/Julio/1999.
--*********************************************************************

library ieee;
use ieee.std_logic_1164.all;

entity fwu is
    port (Fin_Tx,Fin_35_2,Fin_52_2,RxSOC,clk,reset:      in  std_logic;
        cambia,Cuenta_2,Fin_Rx:                          out std_logic);
end fwu;

architecture descripcion_fwu of fwu is
    type estado is (S1,S2,S3,S4,S5,S6,S7);
    signal presente: estado:=S1;

begin
    maquina_fwu:
    process(clk,reset)
    begin
    if reset='1' then
        presente<=S1;
    elsif (clk='1' and clk'event) then
        case presente is
            when S1=>
                if (Fin_Tx='0') then
                    presente<=S2;
                end if;
            when S2=>
                if (RxSOC='1') then
                    presente<=S3;
                end if;
            when S3=>
                if (Fin_35_2='0') then
                    presente<=S4;
                end if;
            when S4=>
                if (Fin_52_2='0') then
                    presente<=S5;
                end if;
            when S5=>
                if (Fin_Tx='0') then
                    presente<=S7;
                else
                    presente<=S6;
                end if;
            when S6=>
                if (Fin_Tx='0') then
                    presente<=S7;
                end if;
            when S7=>
                if (RxSOC='1') then
                    presente<=S3;
                else
                    presente<=S2;
                end if;
        end case;
```

```
    end if;
    end process maquina_fwu;

    salida_fwu:
    process(presente)
    begin
    case presente is
        when S1=>
            cambia<='0';
            Cuenta_2<='1';
            Fin_Rx<='1';
        when S2=>
            cambia<='0';
            Cuenta_2<='1';
            Fin_Rx<='1';
        when S3=>
            cambia<='0';
            Cuenta_2<='0';
            Fin_Rx<='1';
        when S4=>
            cambia<='0';
            Cuenta_2<='0';
            Fin_Rx<='1';
        when S5=>
            cambia<='0';
            Cuenta_2<='0';
            Fin_Rx<='1';
        when S6=>
            cambia<='0';
            Cuenta_2<='1';
            Fin_Rx<='1';
        when S7=>
            cambia<='1';
            Cuenta_2<='1';
            Fin_Rx<='0';
    end case;
    end process salida_fwu;

end descripcion_fwu;
```

cont_r.vhd

```
--*********************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    Contador_R 0-52
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   cont_r.vhd
--Fecha de Realizacion:      07/Julio/1999.
--Fecha de Ultima Revision:  07/Julio/1999.
--*********************************************************************

library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

entity cont_r is
    port (Cuenta_2,reset,clk:    in  std_logic;
          Fin_35_2,Fin_52_2:     out std_logic;
          Ar:                    out std_logic_vector (5 downto 0));
end cont_r;

architecture descripcion_cont_r of cont_r is

begin
    process (clk)
        variable cnt: integer range 0 to 52;
    begin
        if (reset='1') then
            cnt:=0;
```

```
            Fin_35_2<='1';
            Fin_52_2<='1';
        elsif (clk'EVENT and clk='1') then
            if (Cuenta_2='0') then
                if (cnt=52) then
                    cnt:=0;
                    Fin_35_2<='1';
                    Fin_52_2<='1';
                elsif (cnt=50) then
                        cnt:=cnt+1;
                        Fin_35_2<='1';
                        Fin_52_2<='0';
                    elsif (cnt=34) then
                            cnt:=cnt+1;
                            Fin_35_2<='0';
                            Fin_52_2<='1';
                        else
                            cnt:=cnt+1;
                            Fin_35_2<='1';
                            Fin_52_2<='1';
                end if;
            end if;
        end if;
        Ar<=CONV_STD_LOGIC_VECTOR(cnt,6);
    end process;
end descripcion_cont_r;
```

antes.vhd

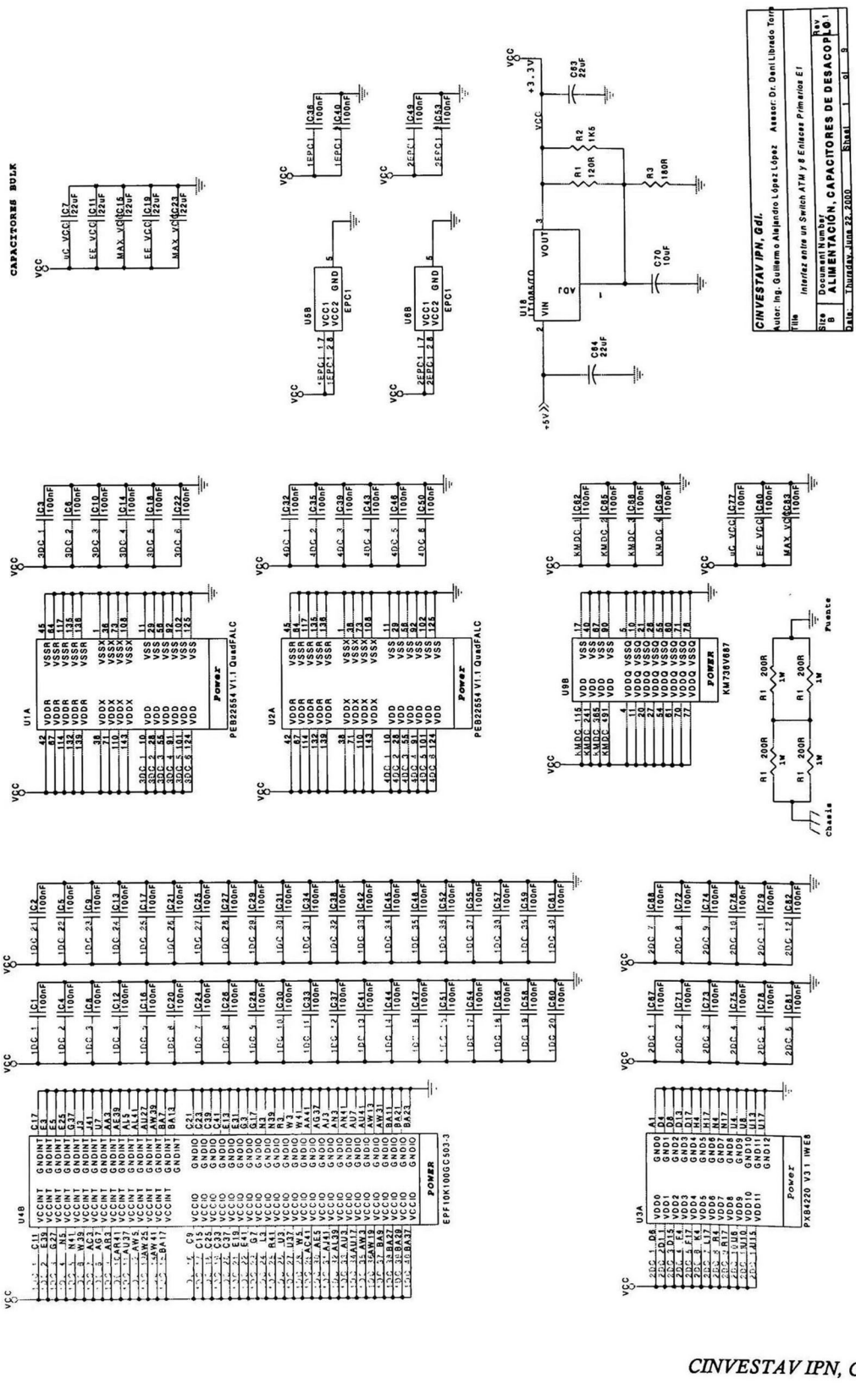
```
--*********************************************************************
--Interfaz entre protocolos de comunicacion UTOPIA y TOPIA
--Interfaz de Recepcion (IWE8>>SF)
--Bloque:                    Antes
--Autor:                     Ing. Guillermo Alejandro Lopez Lopez
--Archivo:                   antes.vhd
--Fecha de Realizacion:      07/Julio/1999.
--Fecha de Ultima Revision:  07/Julio/1999.
--*********************************************************************

library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

entity antes is
    port (cambia,reset,clk:      in  std_logic;
        A9:                      out std_logic);
    end antes;

architecture descripcion_antes of antes is
signal A9_aux:           std_logic;

begin
    cambio:
    process (clk)
    begin
        if (reset='1') then
            A9_aux<='0';
        elsif (clk'EVENT and clk   '1') then
            if (cambia='1') then
                A9_aux<=not A9_aux;
            end if;
        end if;
    end process cambio;
    A9<=A9_aux;
end descripcion_antes;
```

Apéndice B. Esquemáticos de la *IE1-T1/MC*

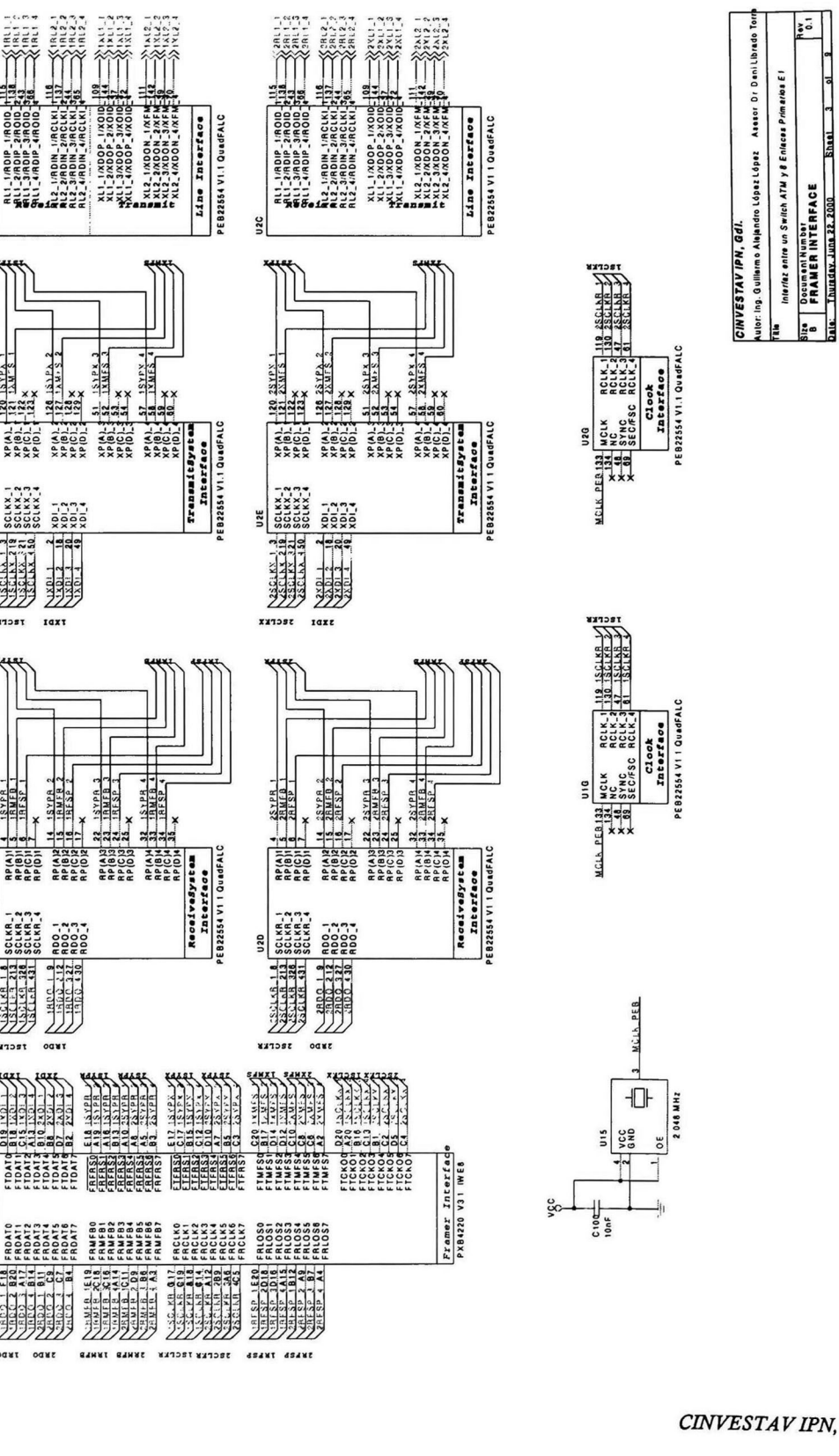
En este apéndice se muestran los esquemáticos de la *IE1-T1/MC*. Estos fueron diseñados con la herramienta *OrCAD Capture Ver. 9.0* de la compañía *OrCAD, Inc.*

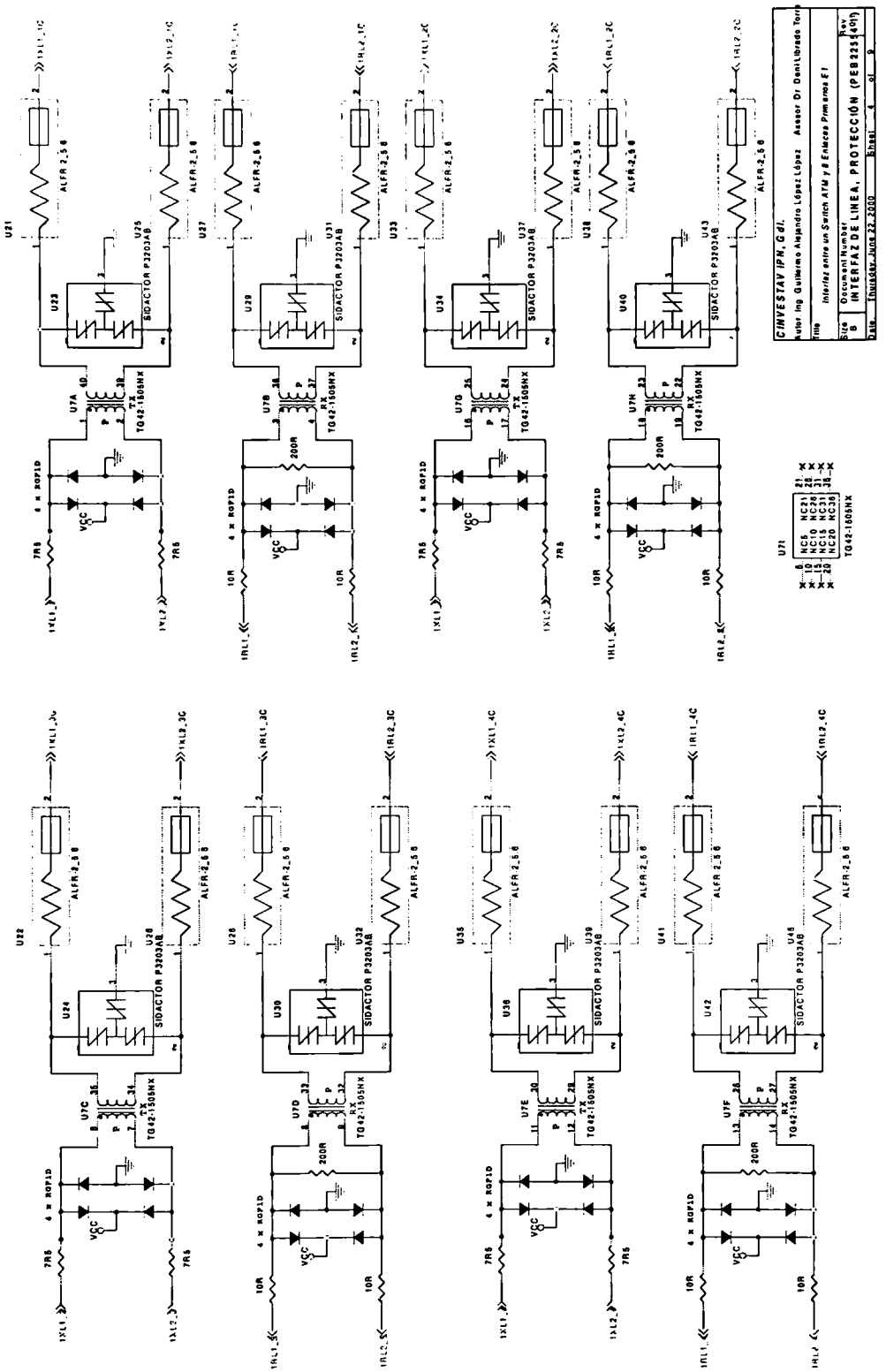
Los esquemáticos están divididos en nueve páginas y los esquemas contenidos en cada una de estas se menciona a continuación:

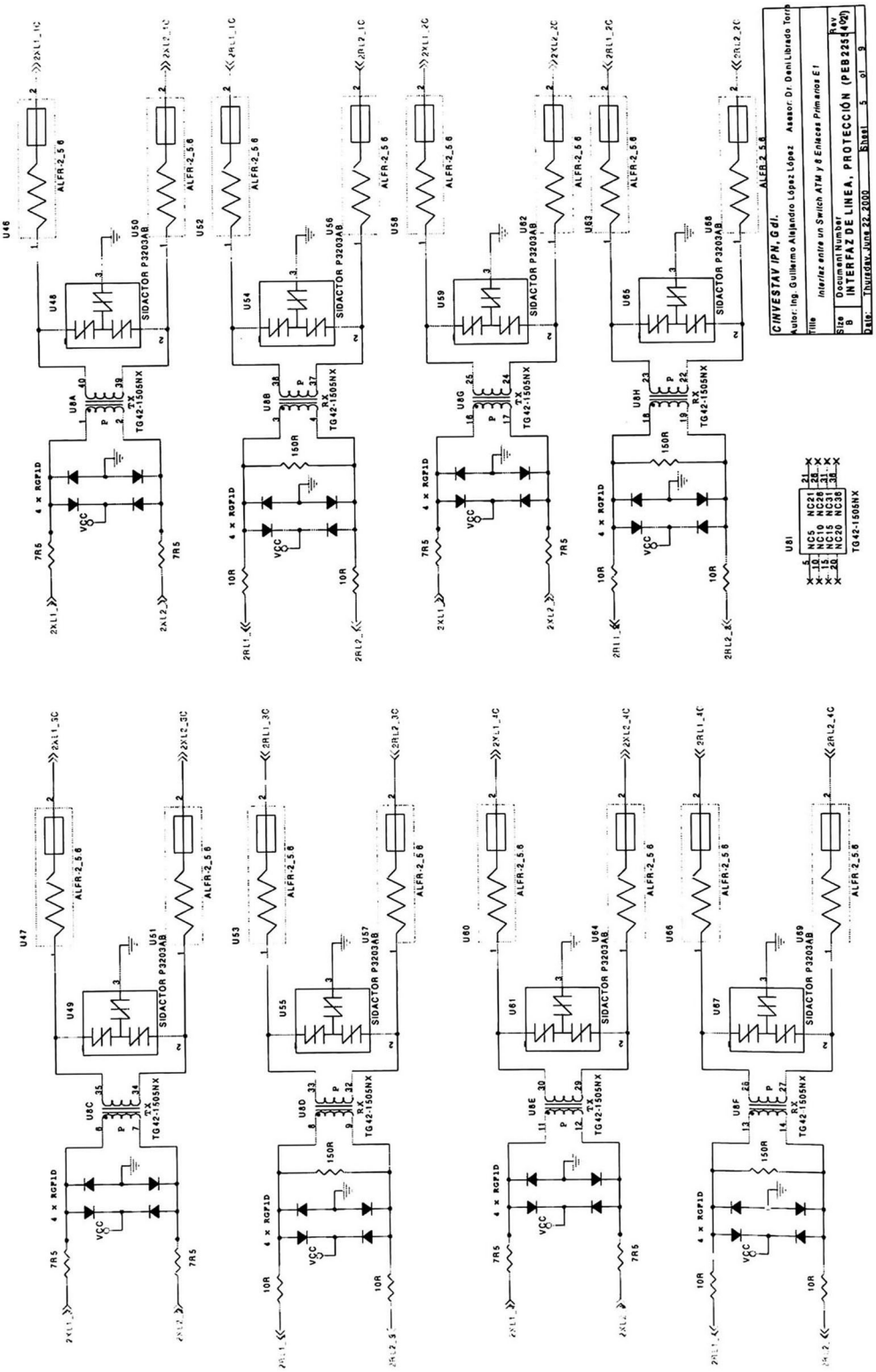
1. Alimentación y capacitores de desacoplo.
2. Conectores y *headers*.
3. Interfaz con los *framers*.
4. Interfaz de línea y circuitos de protección para *framer* 1.
5. Interfaz de línea y circuitos de protección para *framer* 2.
6. Interfaz de prueba y conexión con *JTAG*.
7. Pines de no conexión del *FPGA EPF10K100GC503-3*.
8. Interfaz a memoria *SRAM*, interfaz con *uP* y circuito microcontrolador.
9. Interfaz *TOPIA*, interfaz *UTOPIA Level 2* y Relojes del circuito *IWE* (*SAR*).

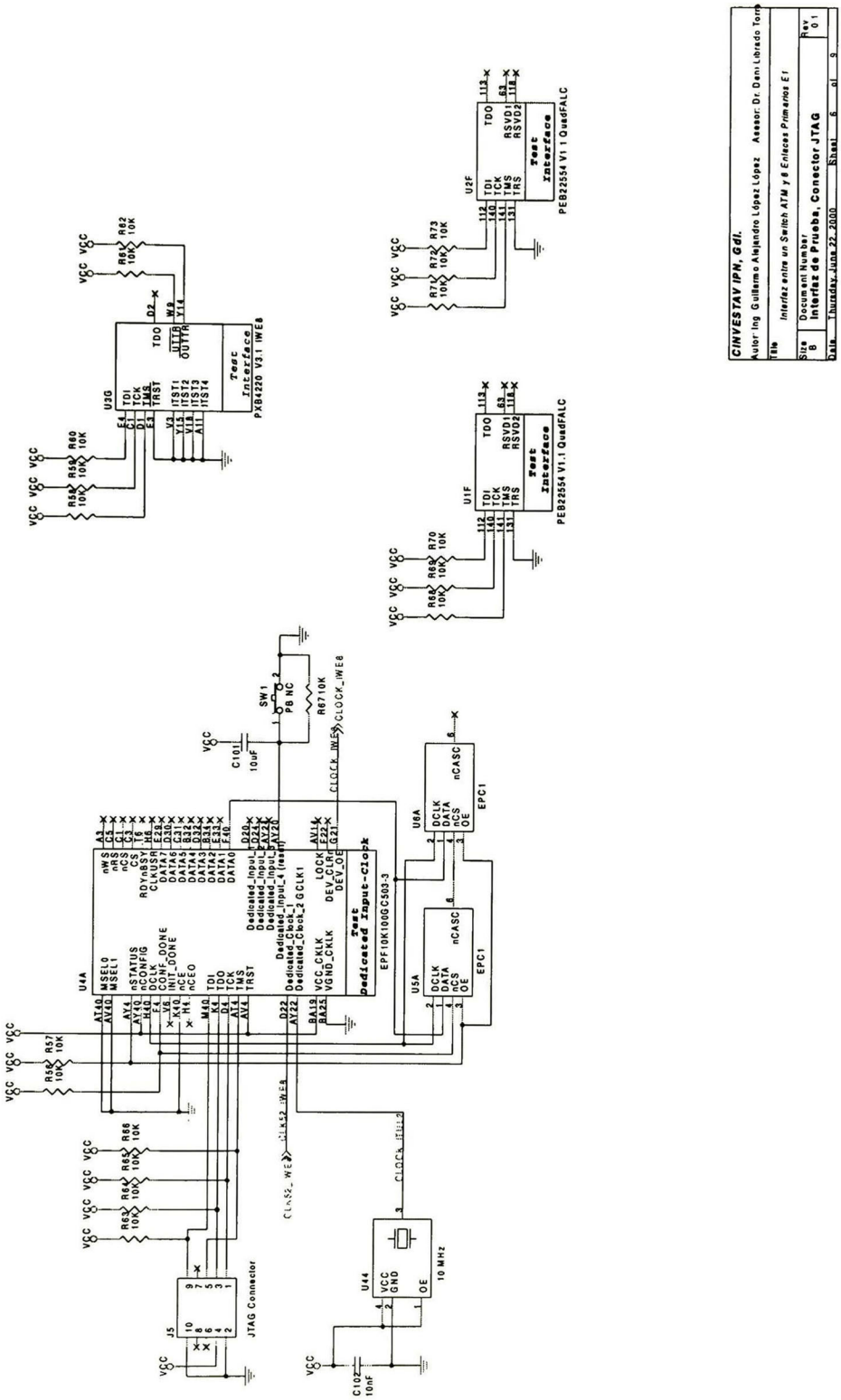

CAPACITORES BULK
CINVESTAV IPN, Gdl.
Autor: Ing. Guillermo Alejandro López López Asesor: Dr. Dani Librado Torres
Title
Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
ALIMENTACIÓN, CAPACITORES DE DESACOPLO
Date: Thursday, June 22, 2000
Sheet 1 of 8
PEB22554 V1.1 QuadFALC
EPF10K100GC503-3
PXB4220 V3.1 IWE8
KM736V687
Chasis
Puente

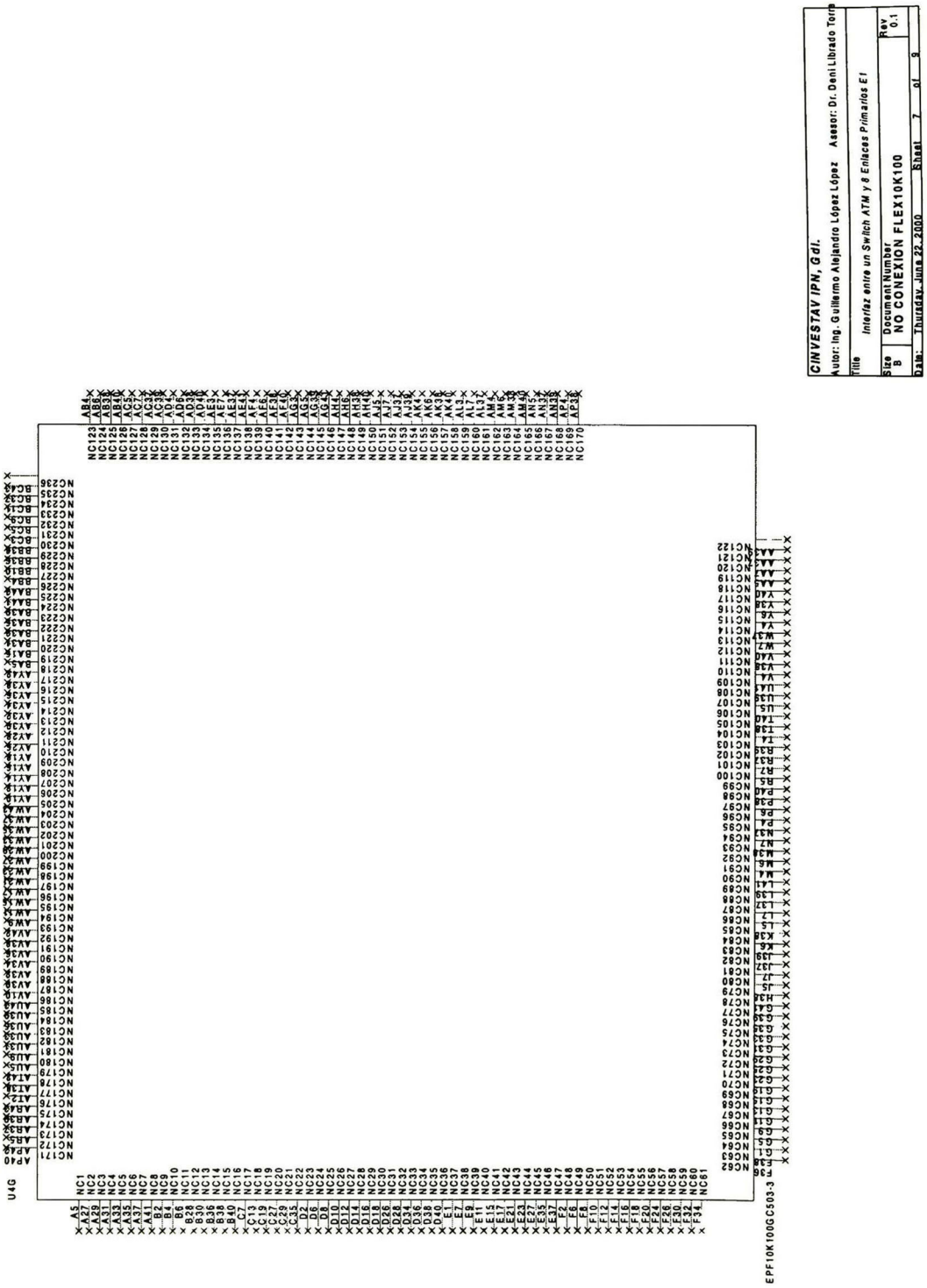
CINVESTAV IPN, Gdl.
Autor: Ing. Guillermo Alejandro López López Asesor: Dr. Dani Librado Torres
Title: Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
Size: B
Document Number: CONECTORES
Rev: 0 1
Date: Thursday, June 22, 2000
Sheet: 2 of 9
CON50A
CON10A
CON3
HEADER 3
JP1
JP2
MAX3232
CONNECTOR DB9
SPI Serial Interface
E1/T1 SEL
+5V
VCC
MOSI
MISO
SCK
RES
RXD
TXD
MAX_VCC
U10A
U10B
U11A
U11B
U12A
U12B
U13A
U13B
J12
J13
J14
J15
J16
J17
J18
J19
J1
J2
J3
J4
P1
U19

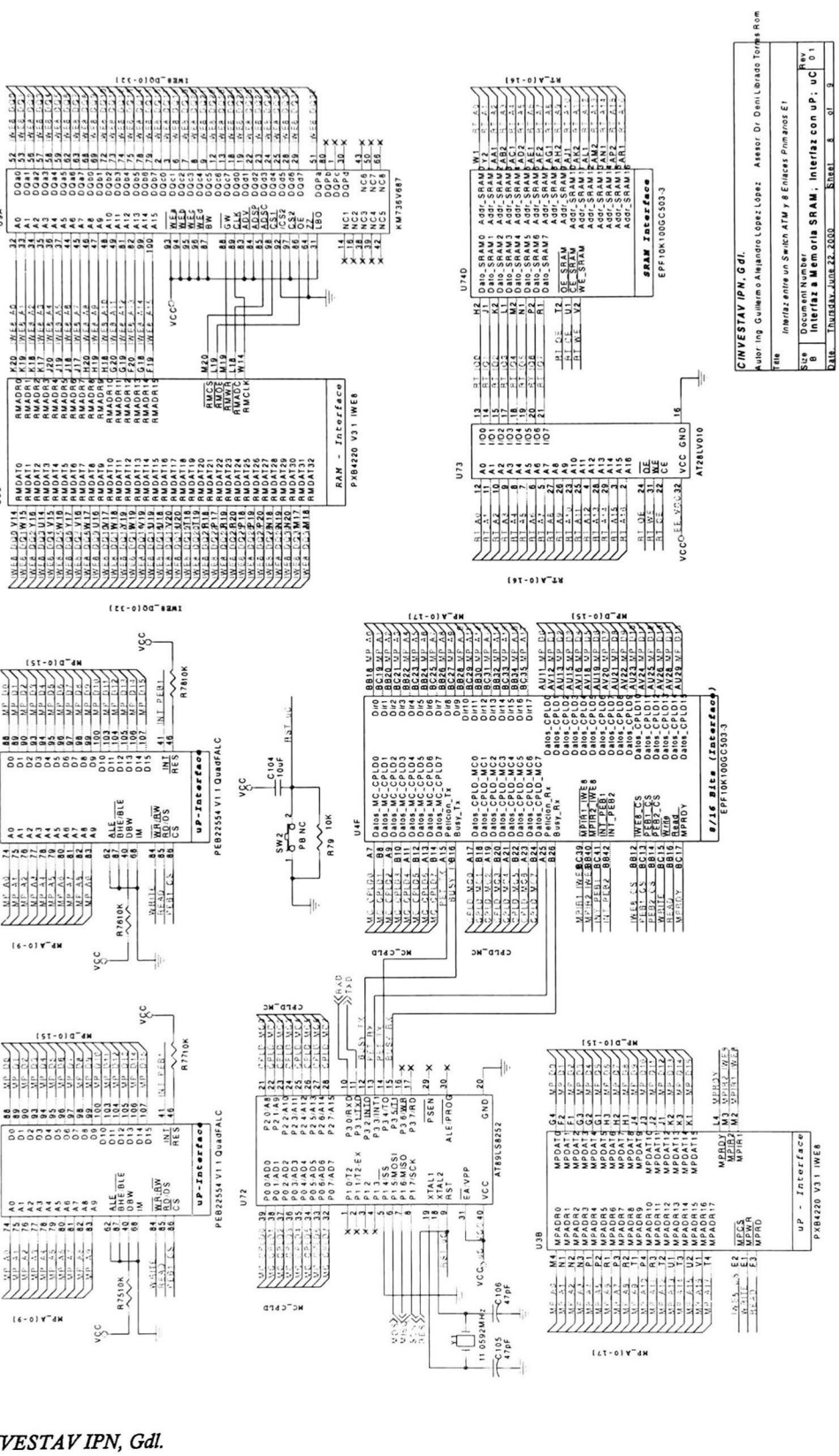

U1C
U1E
U2C
U2E
U1D
U2D
U3D
U1G
U2G
U15
Line Interface
TransmitSystem Interface
ReceiveSystem Interface
Clock Interface
Framer Interface
PEB22554 V1.1 QuadFALC
PXB4220 V3.1 IWE8
2.048 MHz
CINVESTAV IPN, Gdl.
Autor: Ing. Guillermo Alejandro López López Asesor: Dr. Dani Librado Terra
File: Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
Size: B
Document Number: FRAMER INTERFACE
Rev: 0.1
Date: Thursday, June 22, 2000
Sheet 3 of 9

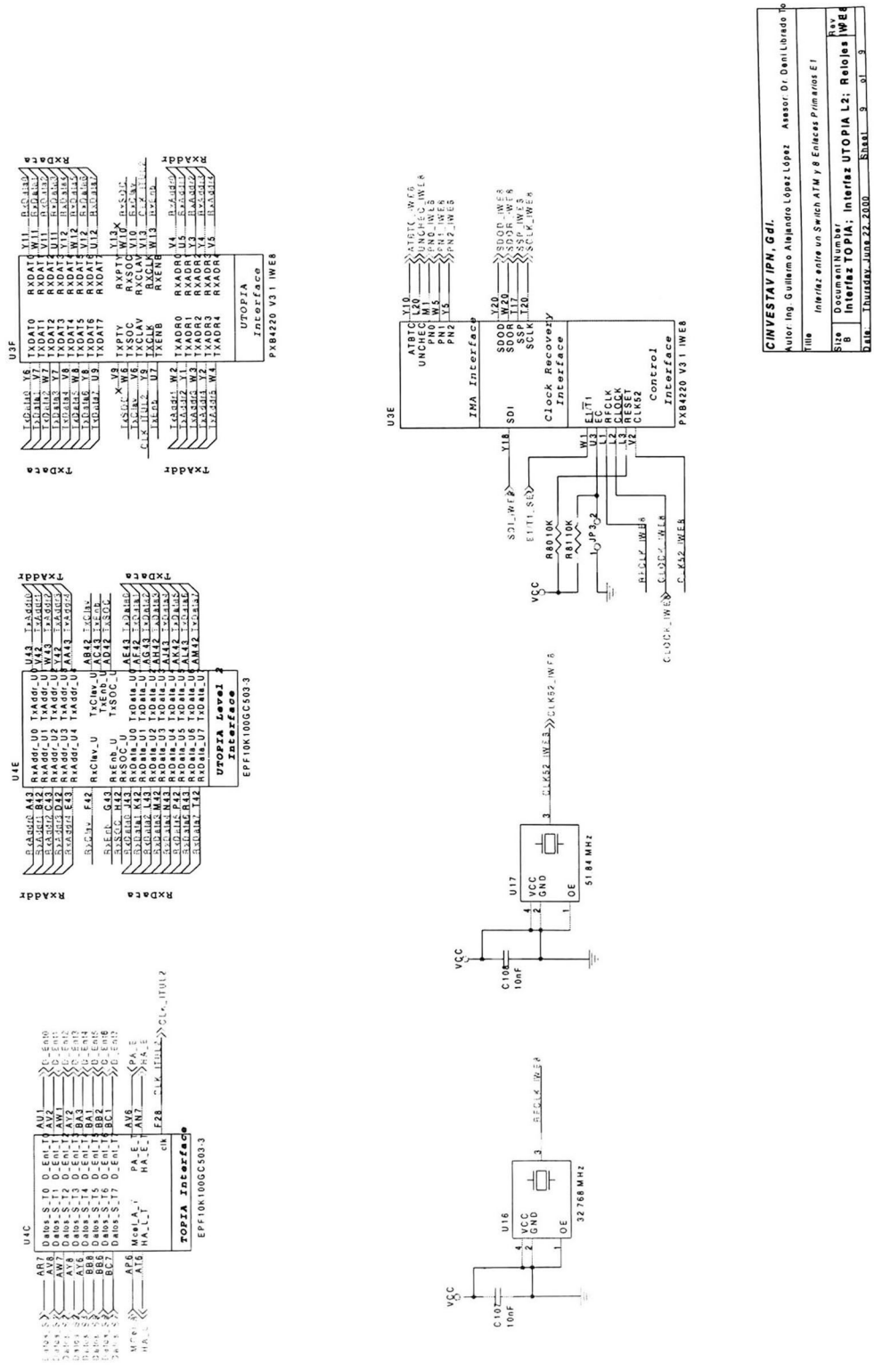

CINVESTAV IPN, Gdl.
Autor Ing Guillermo Alejandro López López Asesor Dr Demitrio Torres
Title Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
Size B Document Number INTERFAZ DE LINEA, PROTECCIÓN (PEB22354)
Date Thursday, June 22, 2000 Sheet 4 of 9
SIDACTOR P3203AB
TG42:1505NX
TG42:1605NX
ALFR-2.5 6
4 x ROP1D
200R
7R5
10R
VCC
U71
NC5 NC21
NC10 NC26
NC15 NC31
NC20 NC36


CINVESTAV IPN, Gdl.
Autor: Ing. Guillermo Alejandro López López Asesor: Dr. Dani Liberato Torre
Title: Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
Size: B
Document Number: INTERFAZ DE LINEA, PROTECCIÓN (PEB22544)
Rev:
Date: Thursday, June 22, 2000
Sheet 5 of 9
4 x RGP1D
SIDACTOR P3203AB
TG42-1505NX
ALFR-2.5 6
7R5
10R
150R
VCC


CINVESTAV IPN, Gdl.
Autor: Ing Guillermo Alejandro López López Asesor: Dr. Dani Lobado Torres
Title
Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
Size B
Document Number
Interfaz de Prueba, Conector JTAG
Rev 0.1
Date Thursday, June 22, 2000 Sheet 6 of 9
JTAG Connector
PXB4220 V3.1 IWE8
PEB22554 V1.1 QuadFALC
EPF10K100GC503-3
EPC1
10 MHz
CLOCK_IWE8


CINVESTAV IPN, Gdl.
Autor: Ing. Guillermo Alejandro López López Asesor: Dr. Daniel Librado Torres
Title
Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
Size B
Document Number
NO CONEXION FLEX10K100
Rev 0.1
Date: Thursday, June 22, 2000 Sheet 7 of 9
U4G
EPF10K100GC503-3


CINVESTAV IPN, Gdl.
Autor Ing Guillermo Alejandro Lopez Lopez Asesor Dr Deni Librado Torres Roman
Title
Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
Size B
Document Number
Interfaz a Memoria SRAM, Interfaz con uP; uC
Rev 0 1
Date: Thursday, June 22, 2000
Sheet 8 of 9
SRAM Interface
EPF10K100GC503-3
RAM - Interface
PXB4220 V3 1 IWE8
uP - Interface
PXB4220 V3 1 IWE8
8/16 Bits (Interface)
EPF10K100GC503-3
uP-Interface
PEB22554 V1 1 QuadFALC
MC_CPLD
CPLD_MC
AT89LS8252
AT28LV010
KM736V687


CINVESTAV IPN, Gdl.
Autor: Ing. Guillermo Alejandro López López
Asesor: Dr. Dani Librado To
Título
Interfaz entre un Switch ATM y 8 Enlaces Primarios E1
Size: B
Document Number: Interfaz TOPIA: Interfaz UTOPIA L2; Reloj
Rev: WE8
Date: Thursday, June 22, 2000
Sheet: 9 of 9
U4C
TOPIA Interface
EPF10K100GC503-3
U4E
UTOPIA Level 2 Interface
EPF10K100GC503-3
U3F
UTOPIA Interface
PXB4220 V3 1 IWE8
U3E
IMA Interface
Clock Recovery Interface
Control Interface
PXB4220 V3 1 IWE8
U16
32.768 MHz
U17
51.84 MHz
VCC
GND
OE
C108 10nF
R8010K
JP3
TxAddr
TxData
RxAddr
RxData

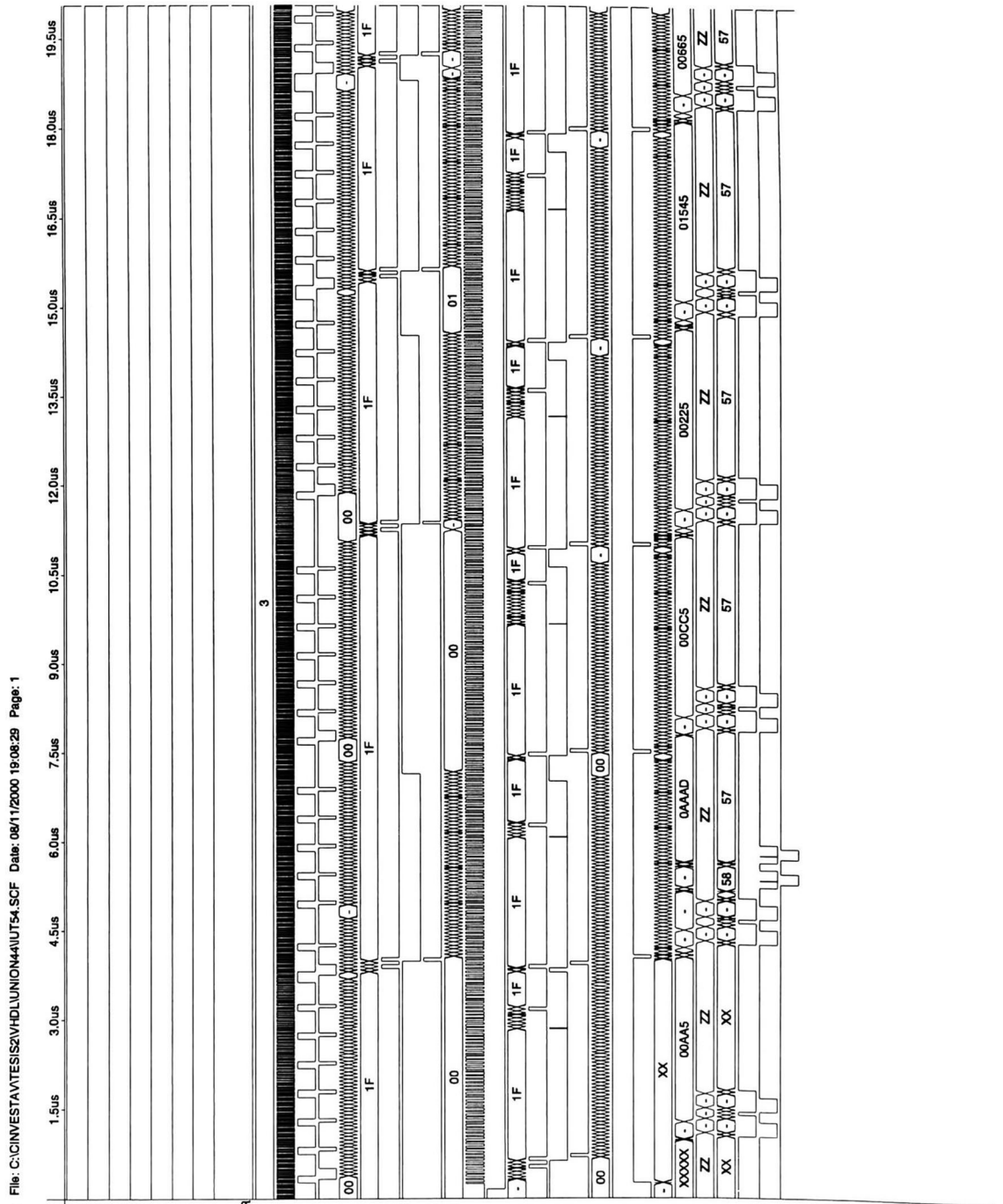
Apéndice C Simulaciones de la *IT/UL2* y *8/16b*

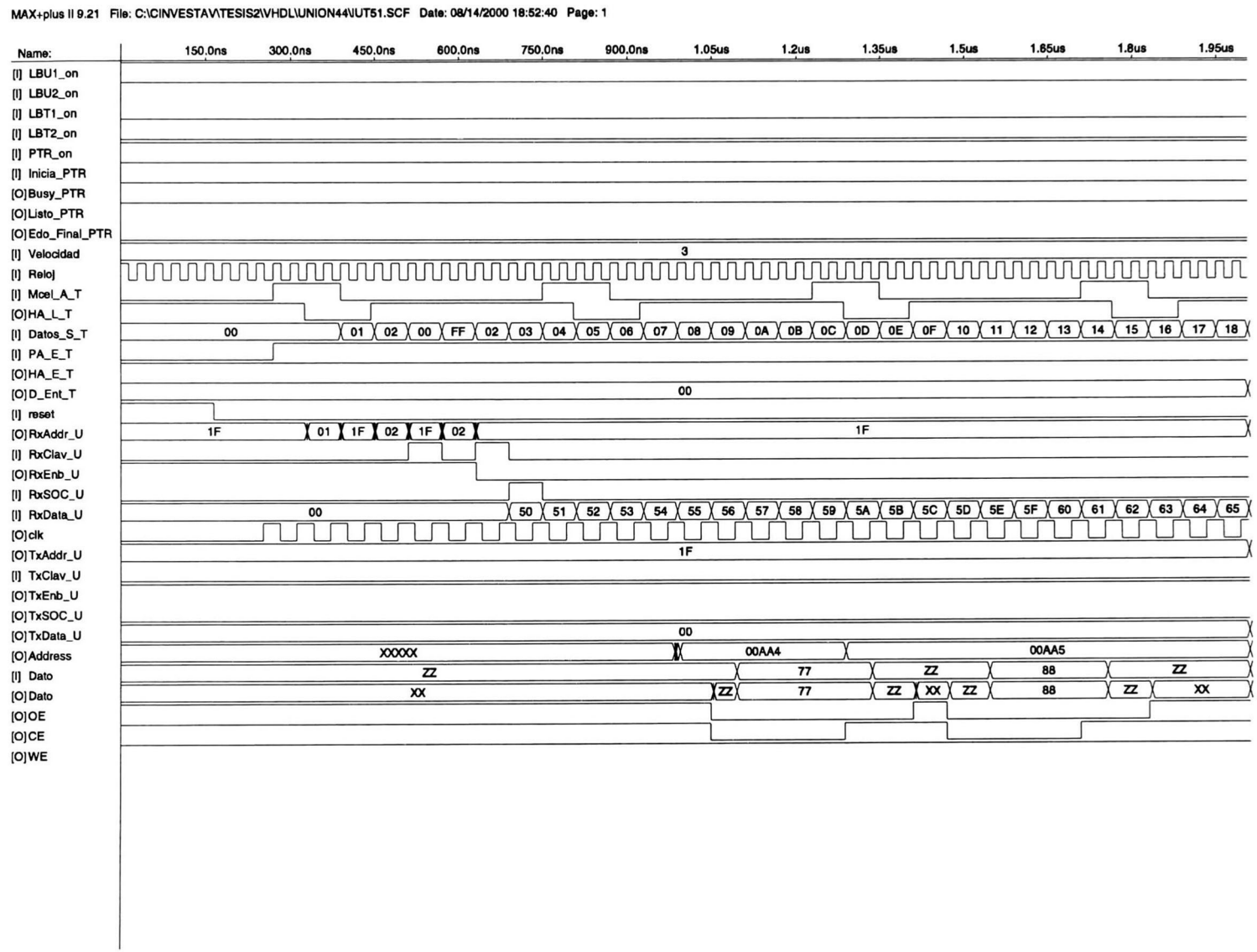
En este apéndice se muestran algunas simulaciones temporales de la *IT/UL2* y de la *I8/16b*. Estas fueron realizadas con la herramienta *Max2Win* de la compañía *ALTERA* para el *FPGA EPF10k100GC503-3* de la familia *Flex10K100.*

Los simulaciones están divididas como sigue:

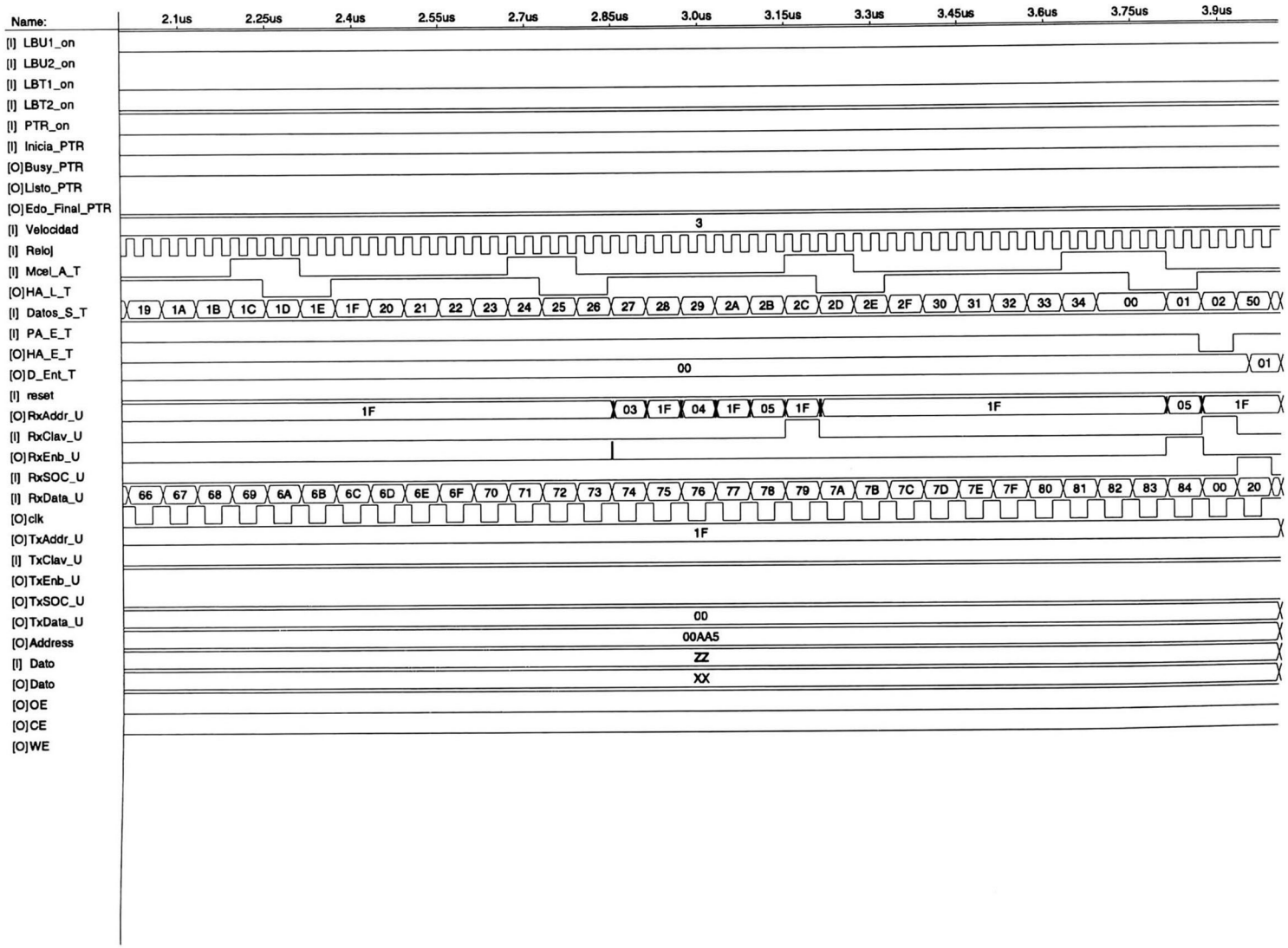
1. La simulación de la *IT/UL2* en modo normal de operación. En este modo la *IT/UL2* convierte la información proveniente de la *MC* (protocolo *TOPIA*) al protocolo *UTOPIA Level 2* para ser entregada a alguna de las capas *PHY* (al circuito *SAR*). También, la *IT/UL2* convierte la información proveniente del circuito *SAR* (protocolo *UTOPIA Level 2*) al protocolo *TOPIA* para entregarla a la *MC*. En este modo los bloques *LBU* y *LBT* se comportan de una forma transparente, esto es, no realizan ningún *loopback*.
2. La *IT/UL2* en modo de pruebas por el lado del protocolo *TOPIA*. En este modo la *IT/UL2* recibe información de la *MC* (en protocolo *TOPIA*) y esta es inmediatamente regresada a la *MC*. Por otro lado, se recibe información del circuito *SAR* (en protocolo *UTOPIA Level 2*), esta información es convertida al protocolo *TOPIA* por la *I_Rx*. Después esta información es entregada al bloque *LBT* pero este no la entrega a la *MC*, sino que la entrega a la *I_Tx* para después ser enviada al circuito *SAR*. Entonces, esta información se vuelve al protocolo *UTOPIA Level 2* y es entregada al circuito *SAR* que fue el que originalmente envió esta celda a la *IT/UL2*.
3. La *IT/UL2* en modo de pruebas por el lado del protocolo *UTOPIA Level 2*. En este modo la *IT/UL2* recibe información de una de las capas *PHY* (circuito *SAR*) en protocolo *UTOPIA Level 2* y esta es inmediatamente regresada al circuito *SAR*. Por otro lado, se recibe información de la *MC* (en protocolo *TOPIA*), esta información es convertida al protocolo *UTOPIA Level 2* por la *I_Tx*. Después, esta información es entregada al bloque *LBU* pero no es entregada al circuito *SAR*, sino que la entrega a la *I_Rx* para ser regresada a la *MC*. Entonces, esta información se vuelve al protocolo *TOPIA* y es entregada a la *MC* que fue la que originalmente envió esta celda a la *IT/UL2*.
4. La simulación de la *I8/16b*. En esta simulación se muestran 4 casos:
   - Una escritura a la *IT/UL2* (0us a 37us aprox.).
   - Una lectura al circuito *SAR* (*IWE8*) (37us a 94us aprox.).

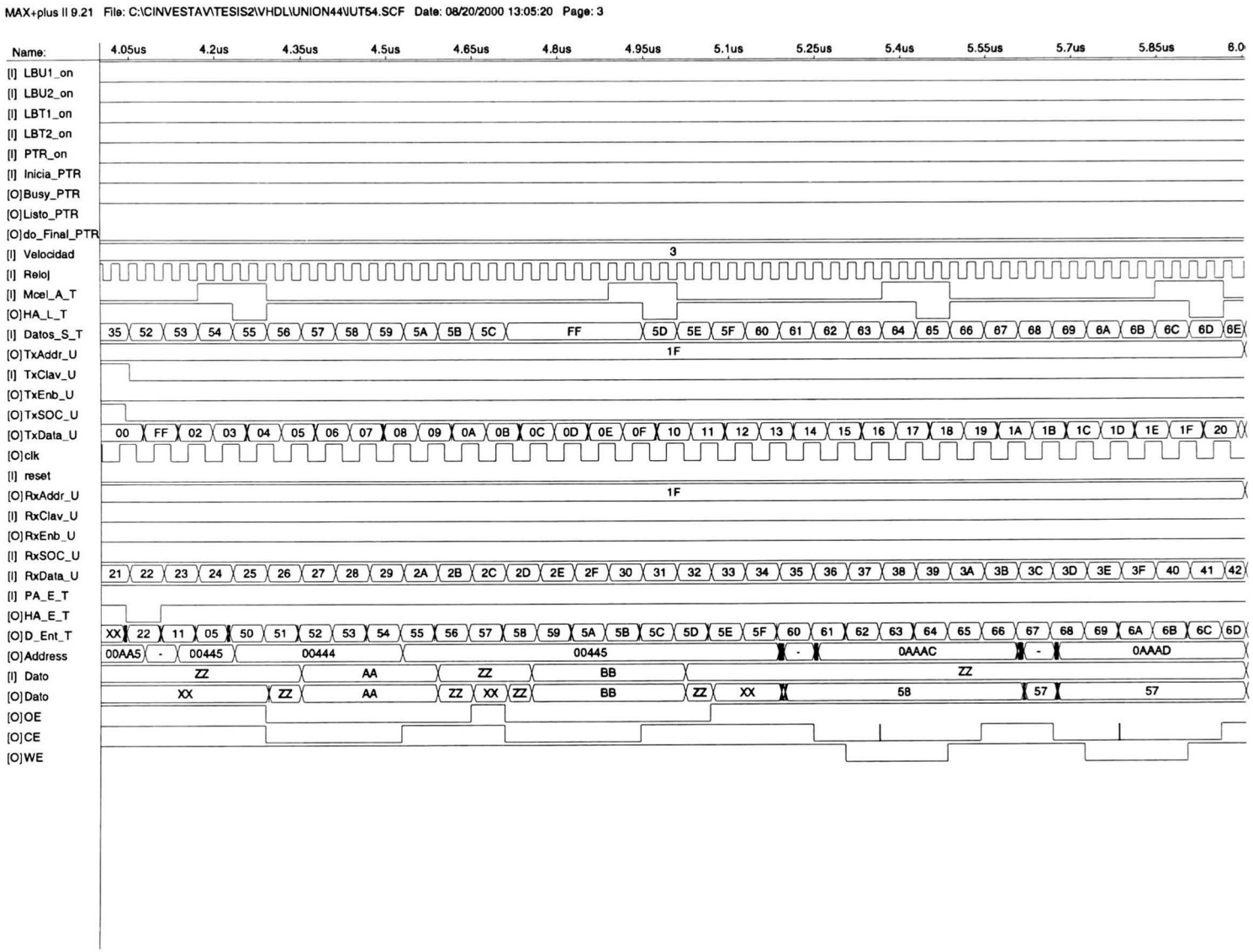
     Una interrupción por parte del circuito *framer* 1 (*PEB1*) (37us a 123us aprox.).
   - Una interrupción por parte de la *IT/UL2* (123us a 150us aprox.).
5. La simulación de la *cama de pruebas* de la *IT/UL2*.

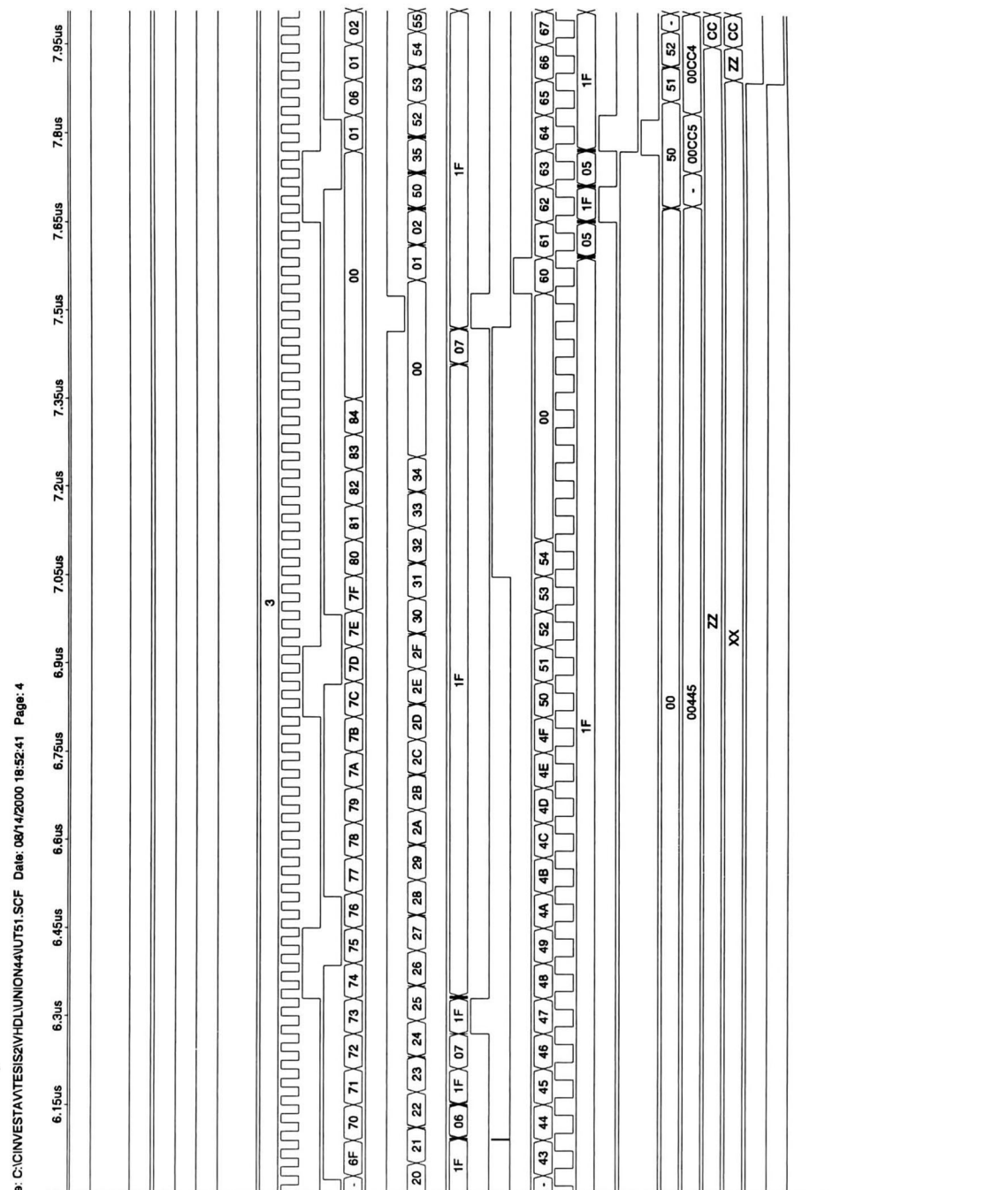

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/11/2000 19:08:29 Page: 1
Name:
1.5us
3.0us
4.5us
6.0us
7.5us
9.0us
10.5us
12.0us
13.5us
15.0us
16.5us
18.0us
19.5us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

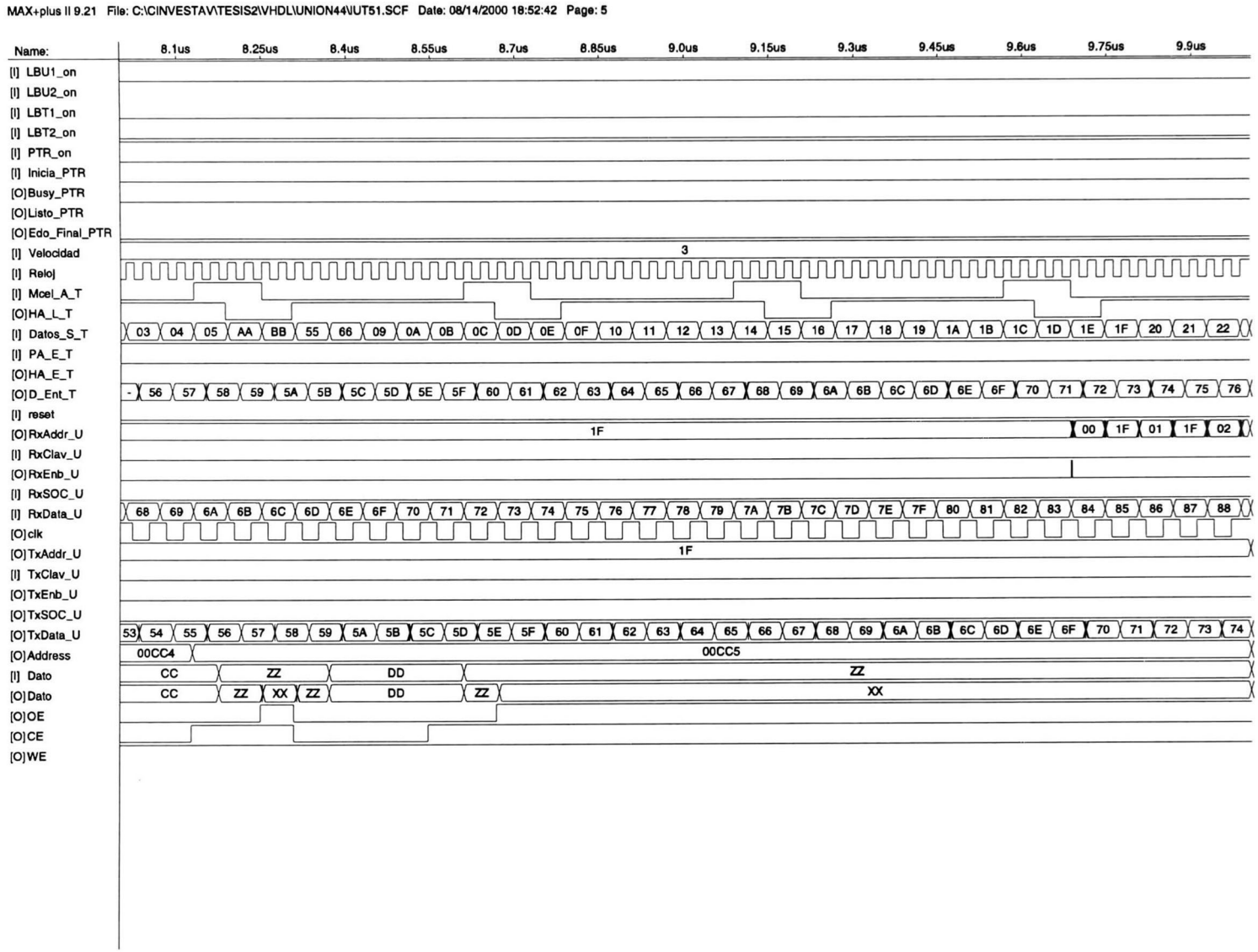

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT51.SCF Date: 08/14/2000 18:52:40 Page: 1
Name:
150.0ns
300.0ns
450.0ns
600.0ns
750.0ns
900.0ns
1.05us
1.2us
1.35us
1.5us
1.65us
1.8us
1.95us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
[I] reset
[O] RxAddr_U
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O] clk
[O] TxAddr_U
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
[O] Address
[I] Dato
[O] Dato
[O] OE
[O] CE
[O] WE
3
00
01 02 00 FF 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18
00
1F
01 1F 02 1F 02
1F
00
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65
1F
00
XXXXX
00AA4
00AA5
ZZ
77
ZZ
88
ZZ
XX
ZZ
77
ZZ
XX
ZZ
88
ZZ
XX

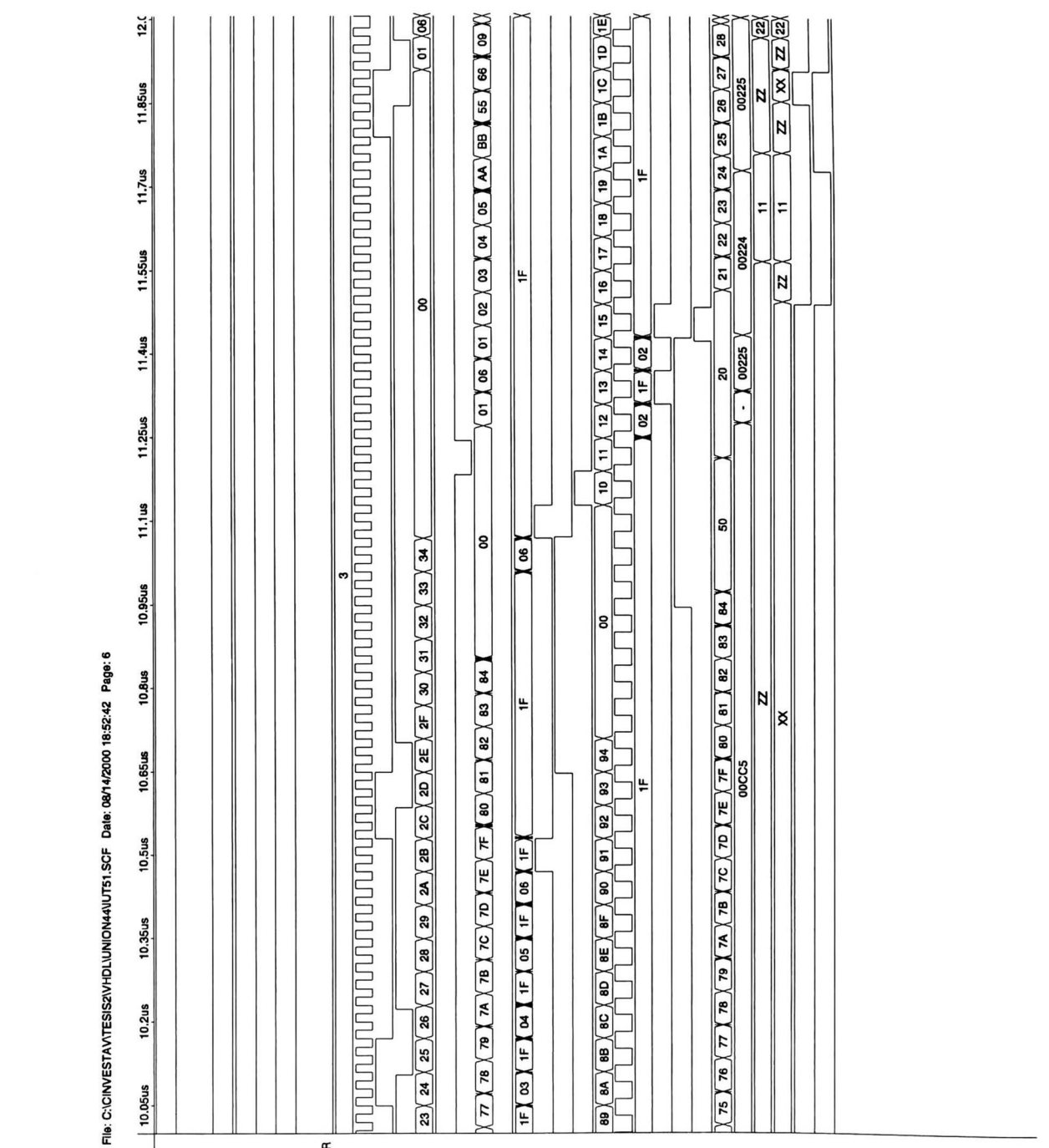
MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\IUT51.SCF Date: 08/14/2000 18:52:41 Page: 2

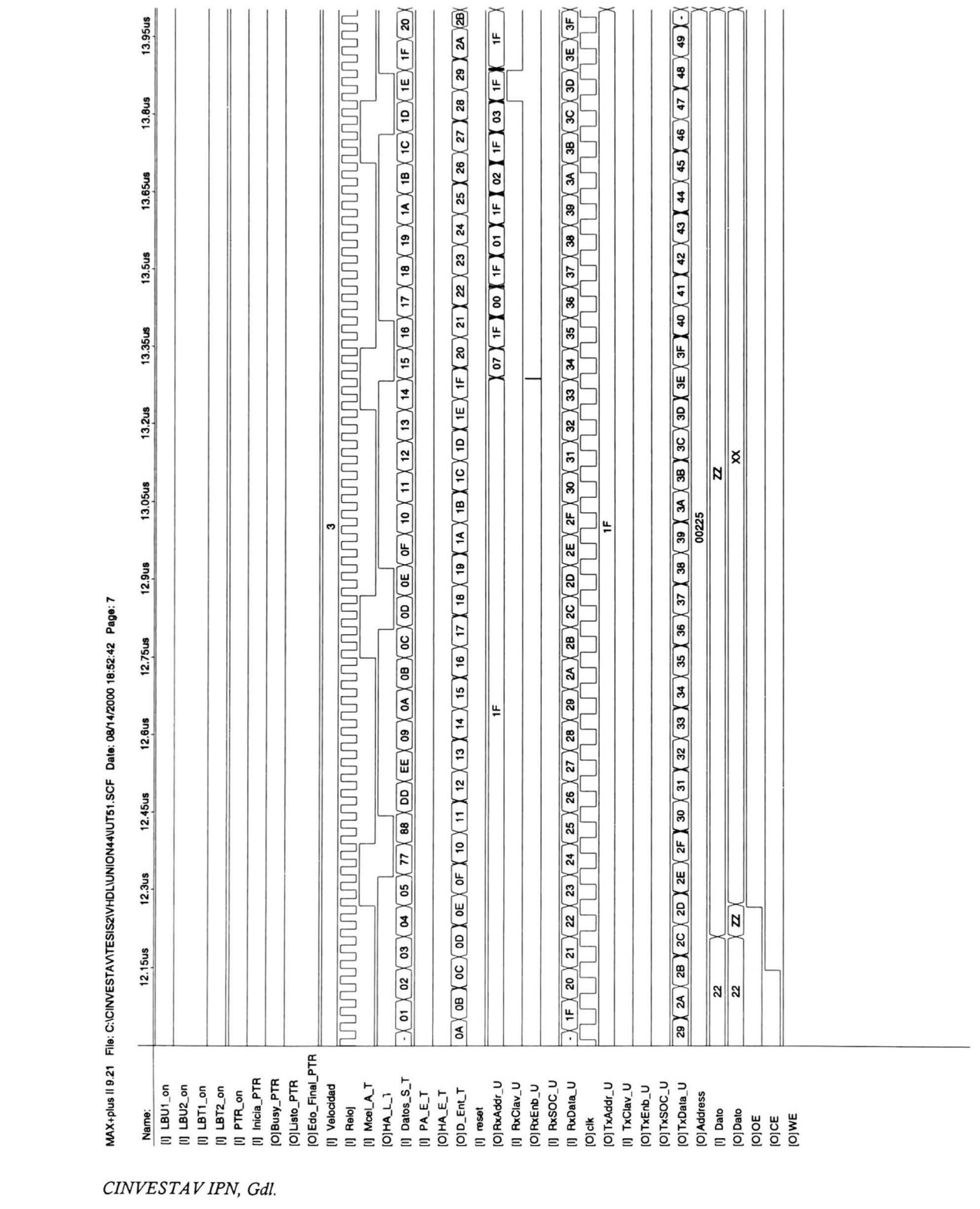


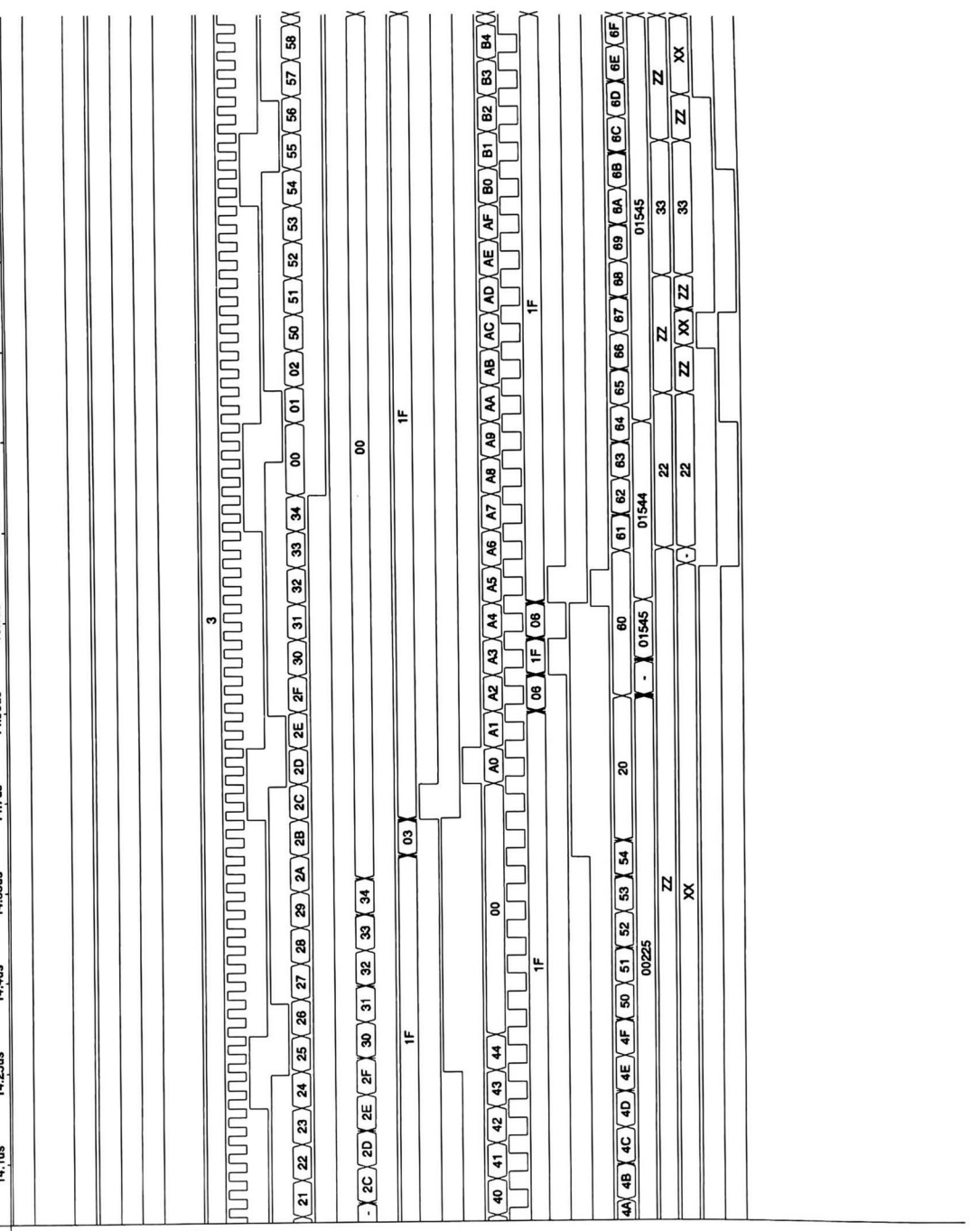

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:20 Page: 3
Name:
4.05us
4.2us
4.35us
4.5us
4.65us
4.8us
4.95us
5.1us
5.25us
5.4us
5.55us
5.7us
5.85us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT51.SCF Date: 08/14/2000 18:52:41 Page: 4
Name:
6.15us
6.3us
6.45us
6.6us
6.75us
6.9us
7.05us
7.2us
7.35us
7.5us
7.65us
7.8us
7.95us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] Moel_A_T
[O]HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O]clk
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

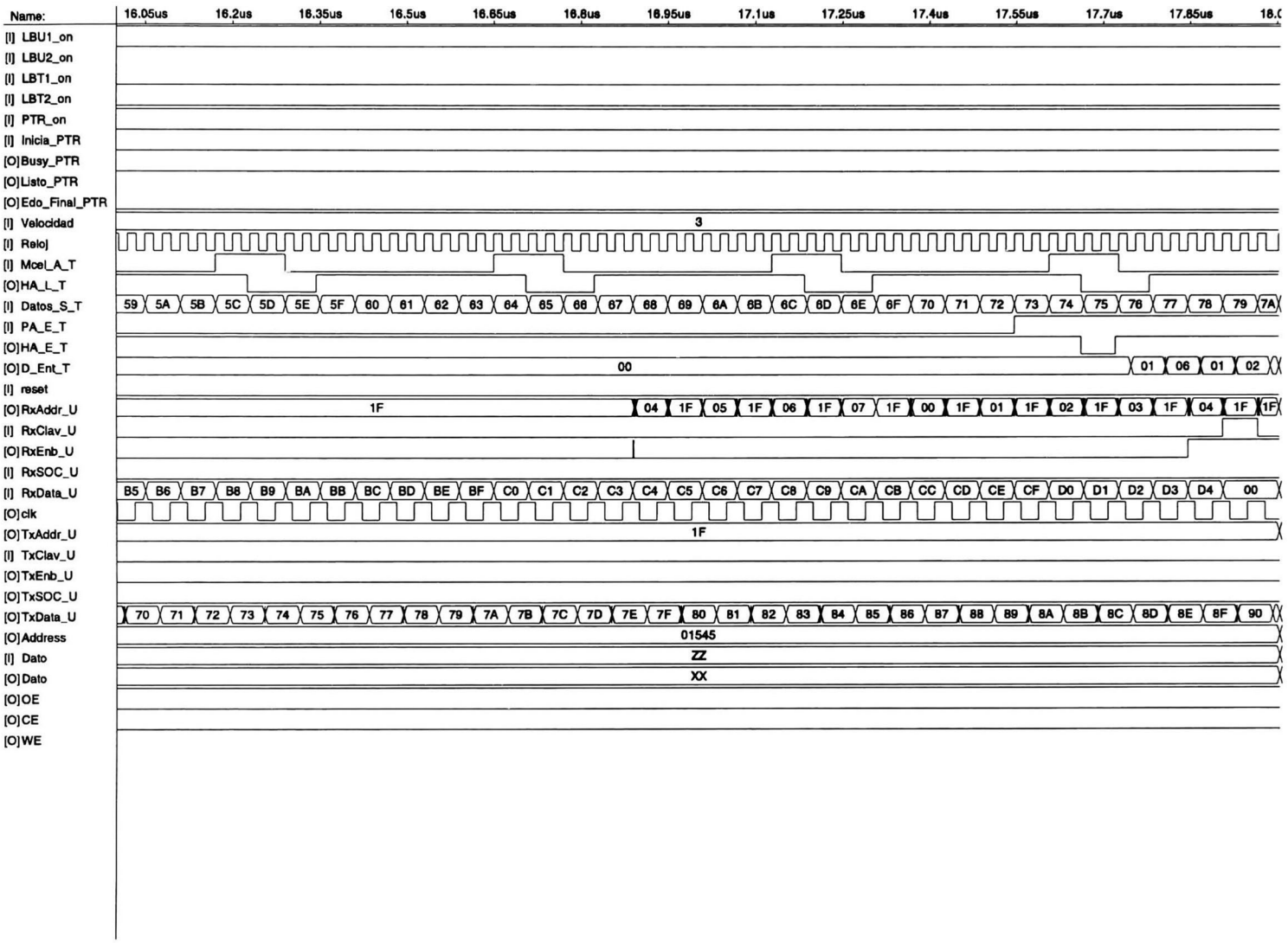

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\IUT51.SCF Date: 08/14/2000 18:52:42 Page: 5
Name:
8.1us
8.25us
8.4us
8.55us
8.7us
8.85us
9.0us
9.15us
9.3us
9.45us
9.6us
9.75us
9.9us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O]clk
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE
3
03 04 05 AA BB 55 66 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22
- 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76
1F
00 1F 01 1F 02
68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81 82 83 84 85 86 87 88
1F
53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74
00CC4
00CC5
CC
ZZ
DD
ZZ
CC
ZZ
XX
ZZ
DD
ZZ
XX


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT51.SCF Date: 08/14/2000 18:52:42 Page: 6
Name:
10.05us
10.2us
10.35us
10.5us
10.65us
10.8us
10.95us
11.1us
11.25us
11.4us
11.55us
11.7us
11.85us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O]clk
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

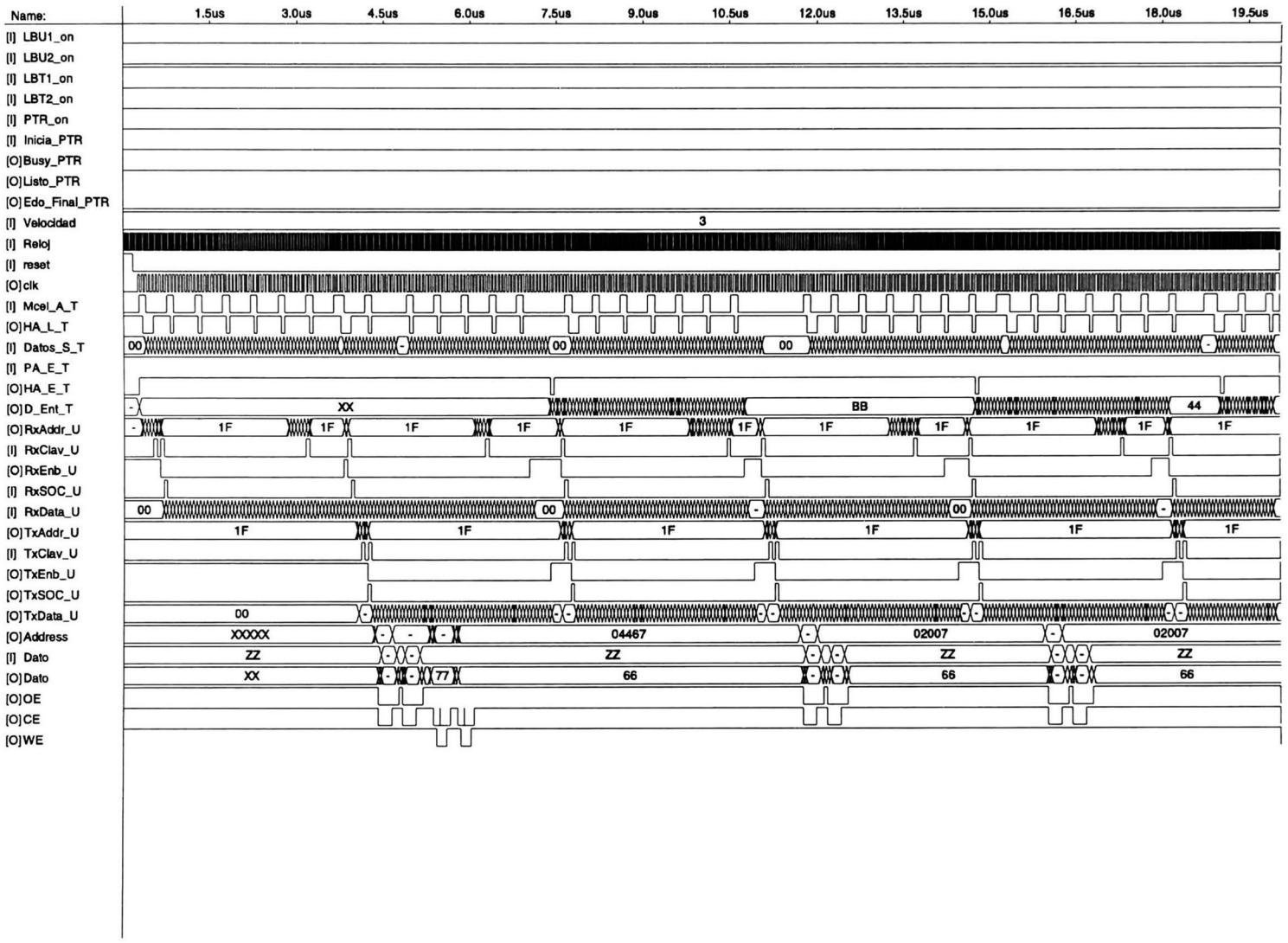

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS\2\VHDL\UNION44\UUT51.SCF Date: 08/14/2000 18:52:42 Page: 7
Name:
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O] HA_L_1
[I] Datos_S_T
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
[I] reset
[O] RxAddr_U
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O] clk
[O] TxAddr_U
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
[O] Address
[I] Dato
[O] Dato
[O] OE
[O] CE
[O] WE
12.15us
12.3us
12.45us
12.6us
12.75us
12.9us
13.05us
13.2us
13.35us
13.5us
13.65us
13.8us
13.95us
3
1F
00225
ZZ
XX
ZZ


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT51.SCF Date: 08/14/2000 18:52:43 Page: 8
Name:
14.1us
14.25us
14.4us
14.55us
14.7us
14.85us
15.0us
15.15us
15.3us
15.45us
15.6us
15.75us
15.9us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mosi_A_T
[O]HA_L_i
[I] Datos_S_T
[I] PA_E_T
[O]HA_E_T
[O] D_Ent_T
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O]clk
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

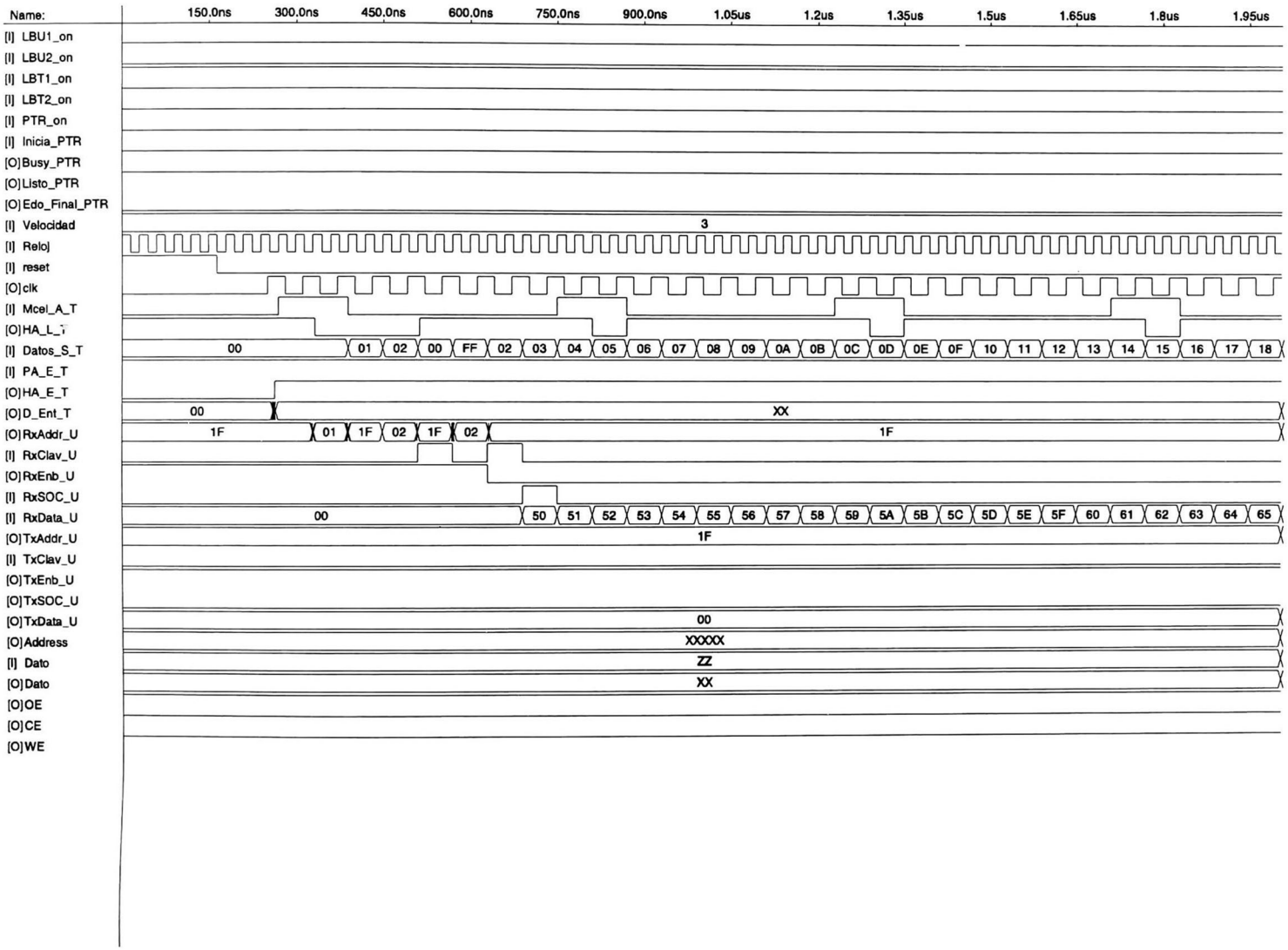
MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT51.SCF Date: 08/14/2000 18:52:43 Page: 9

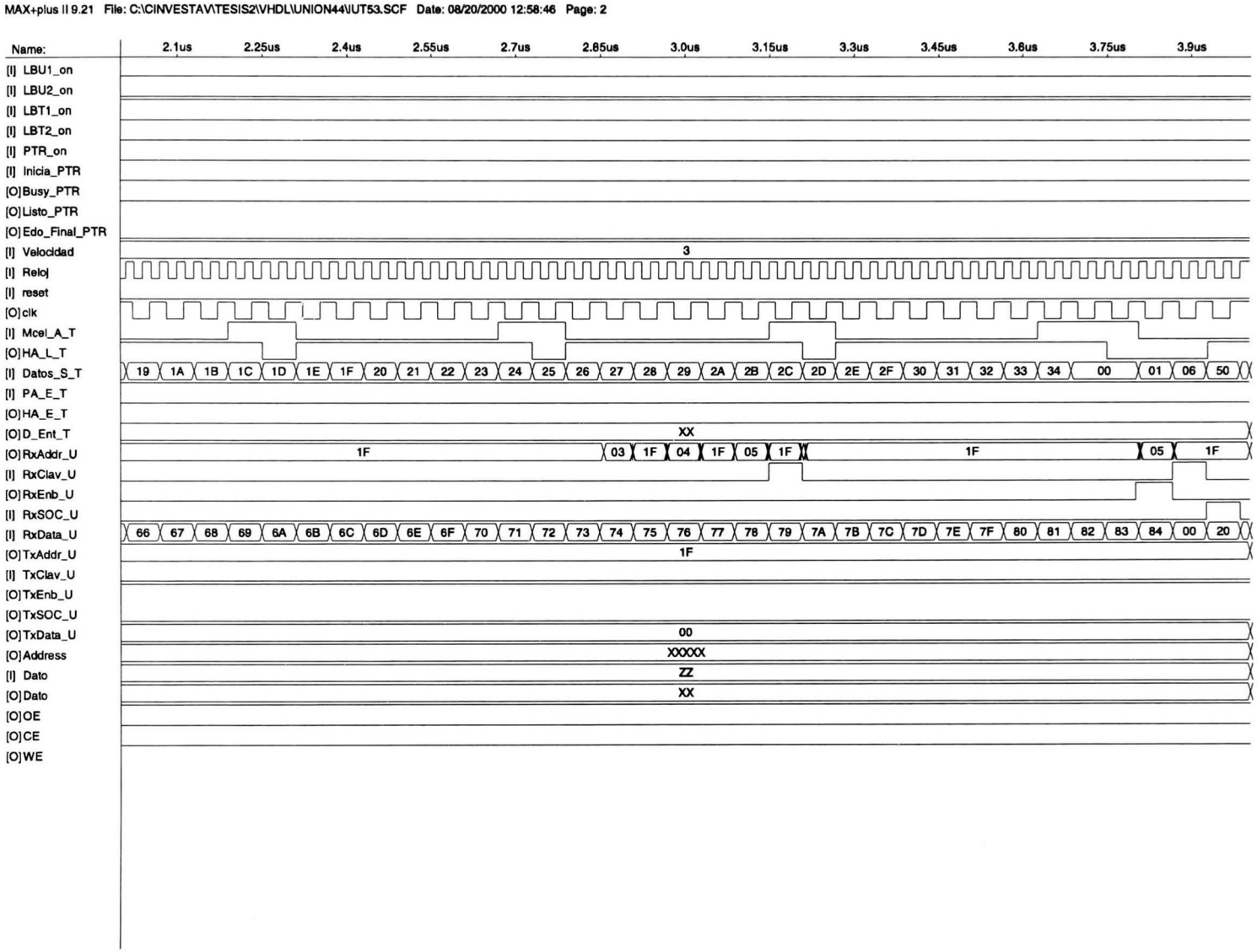


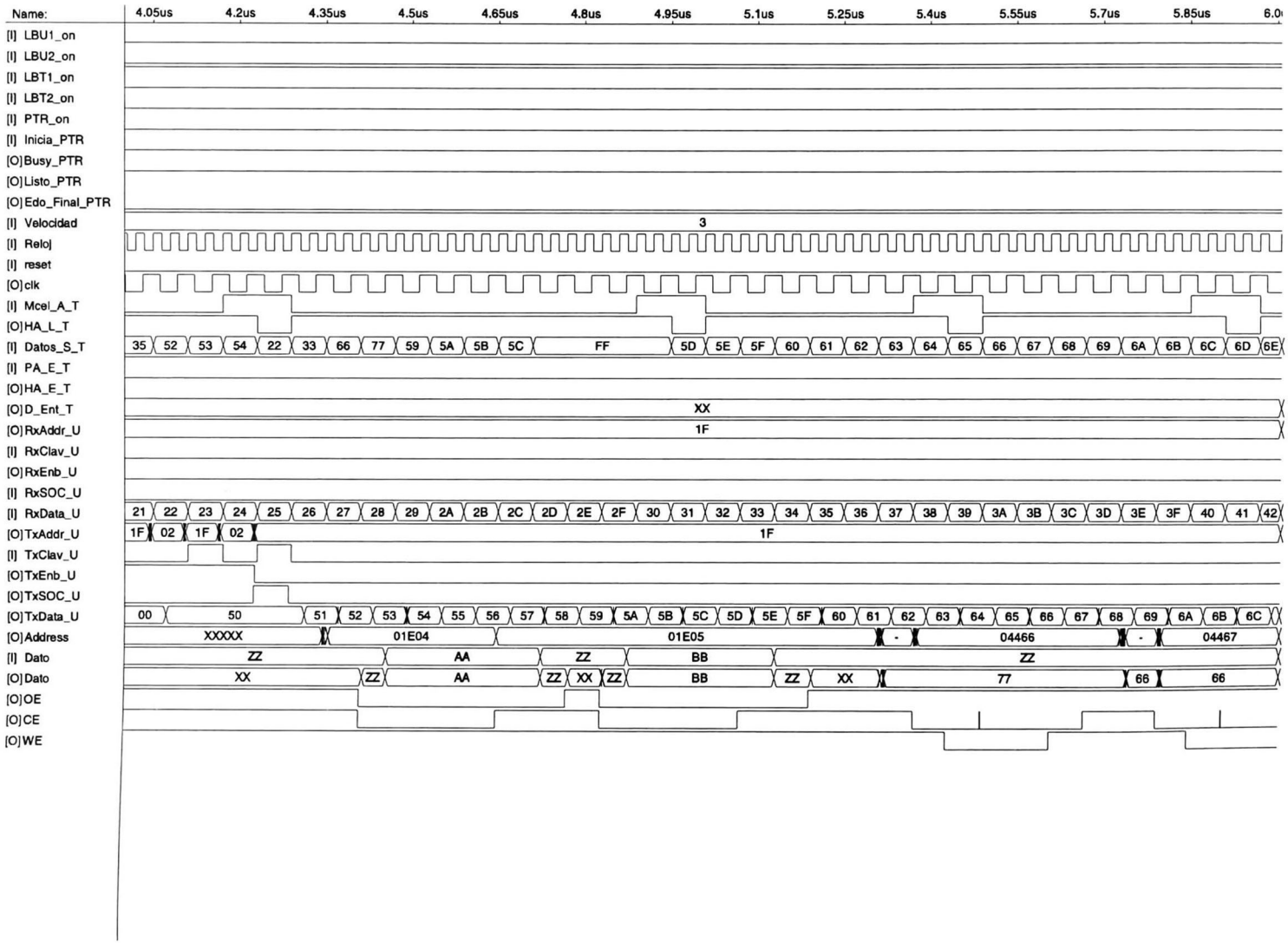
MAX+plus II 9.21 File: C:\CINVESTAVTESIS2\VHDLUNION44\UT51.SCF Date: 08/14/2000 18:52:44 Page: 10
Name:
18.15us
18.3us
18.45us
18.6us
18.75us
18.9us
19.05us
19.2us
19.35us
19.5us
19.65us
19.8us
19.95us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O]HA_E_T
[O]D_EnL_T
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O]clk
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

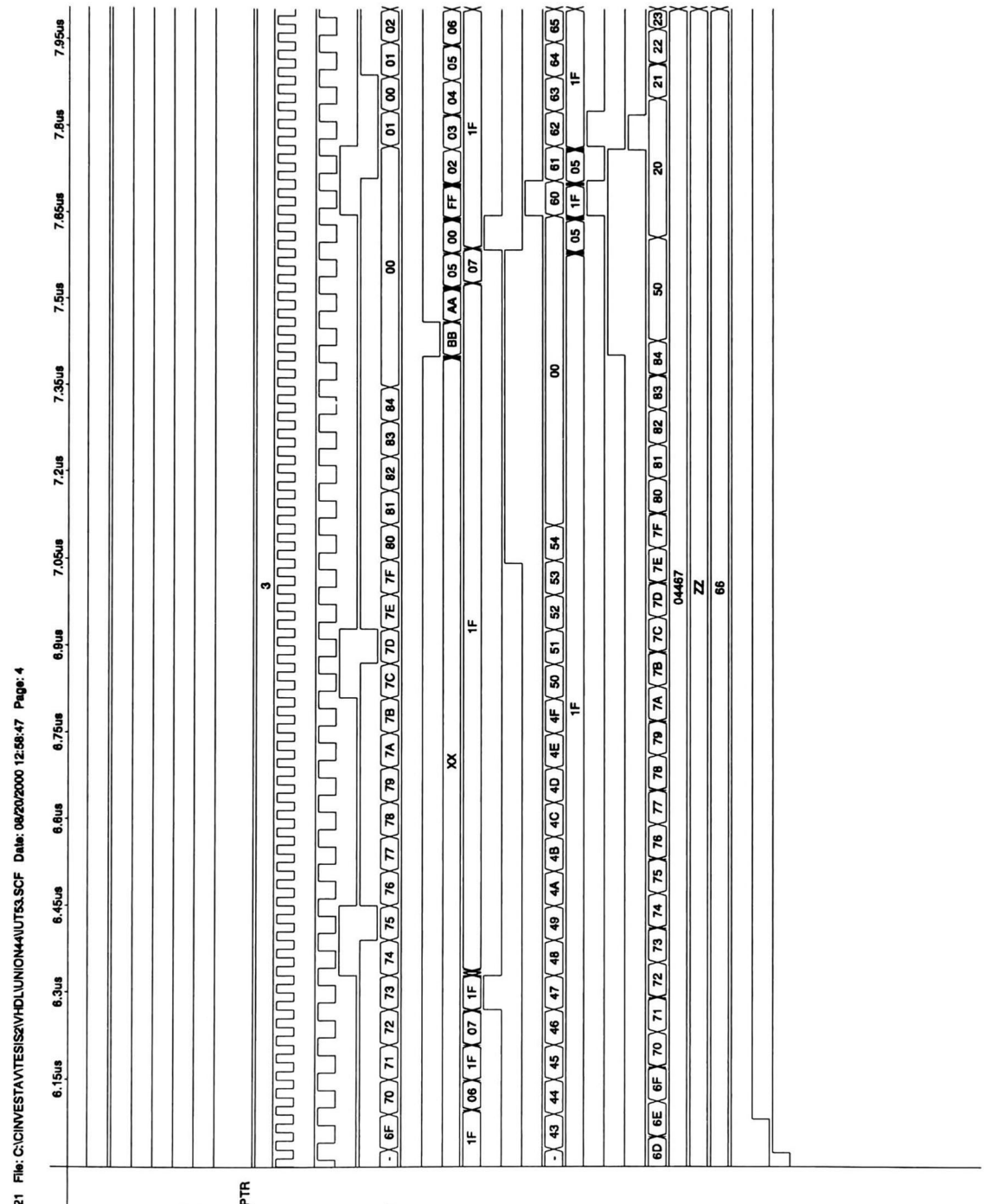

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT53.SCF Date: 08/11/2000 18:50:47 Page: 1
Name:
1.5us
3.0us
4.5us
6.0us
7.5us
9.0us
10.5us
12.0us
13.5us
15.0us
16.5us
18.0us
19.5us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
3
[I] Reloj
[I] reset
[O] clk
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
[O] RxAddr_U
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O] TxAddr_U
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
[O] Address
[I] Dato
[O] Dato
[O] OE
[O] CE
[O] WE
XX
BB
44
1F
00
XXXXX
04467
02007
ZZ
77
66

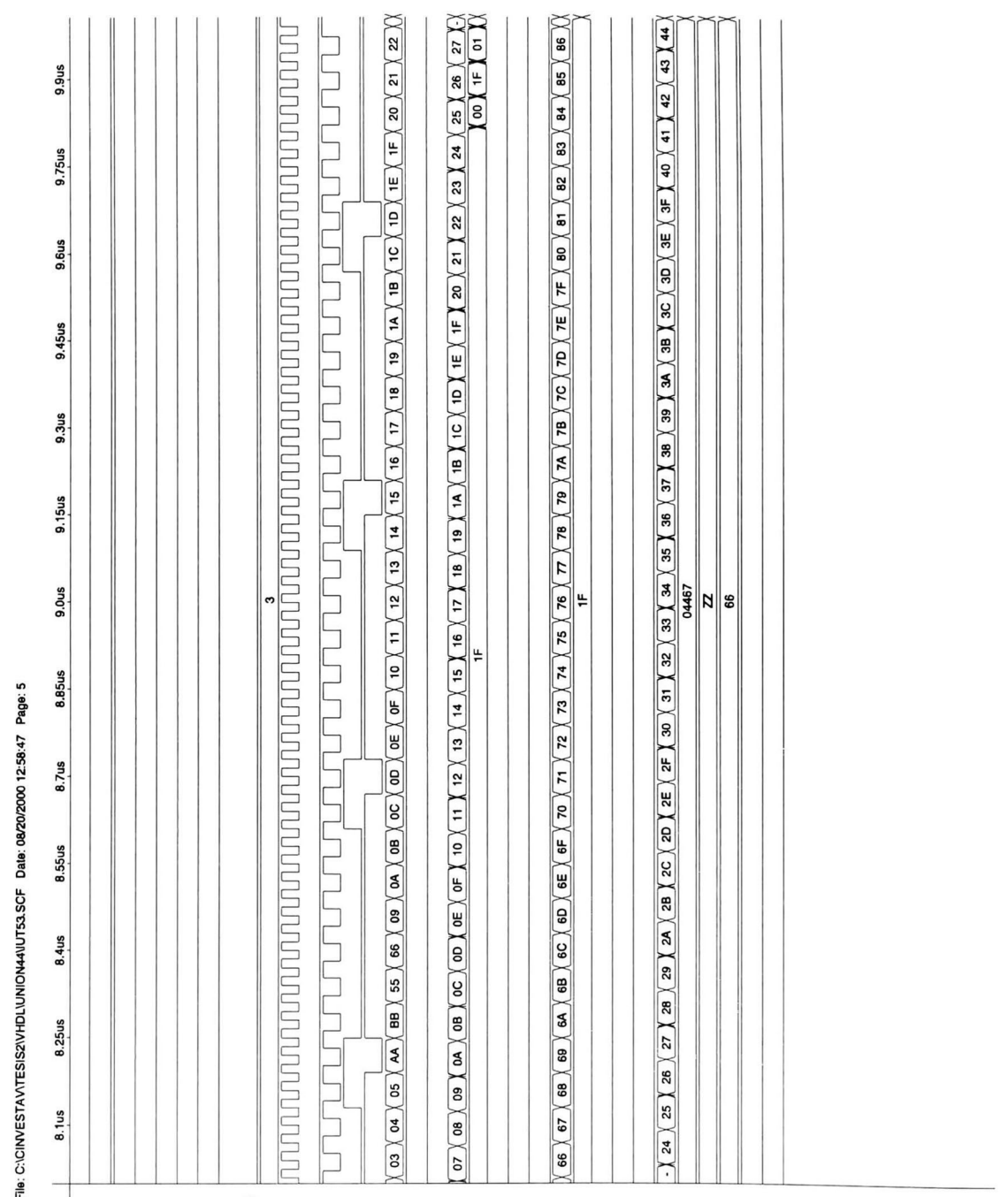
MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT53.SCF Date: 08/20/2000 12:58:46 Page: 1

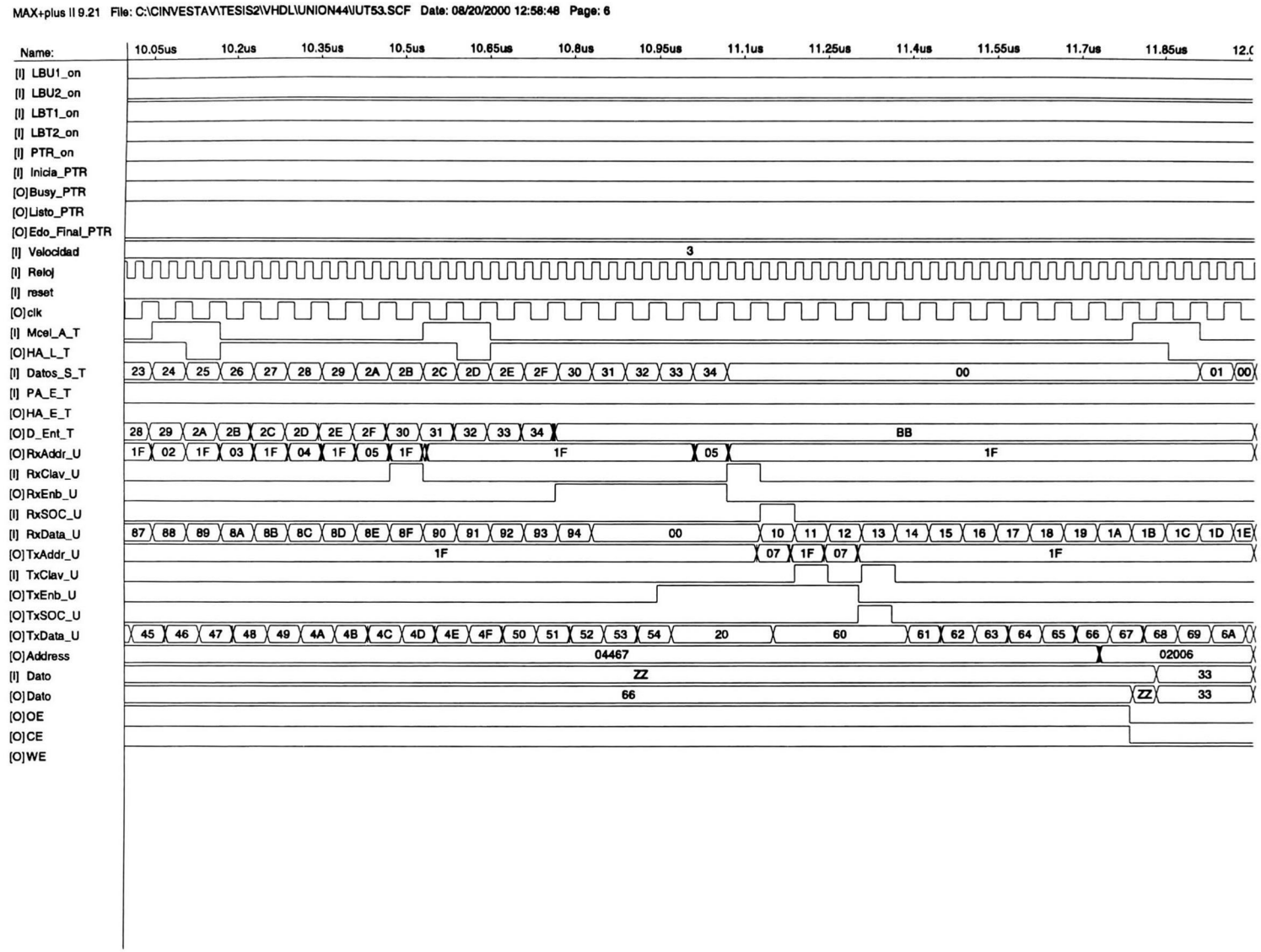


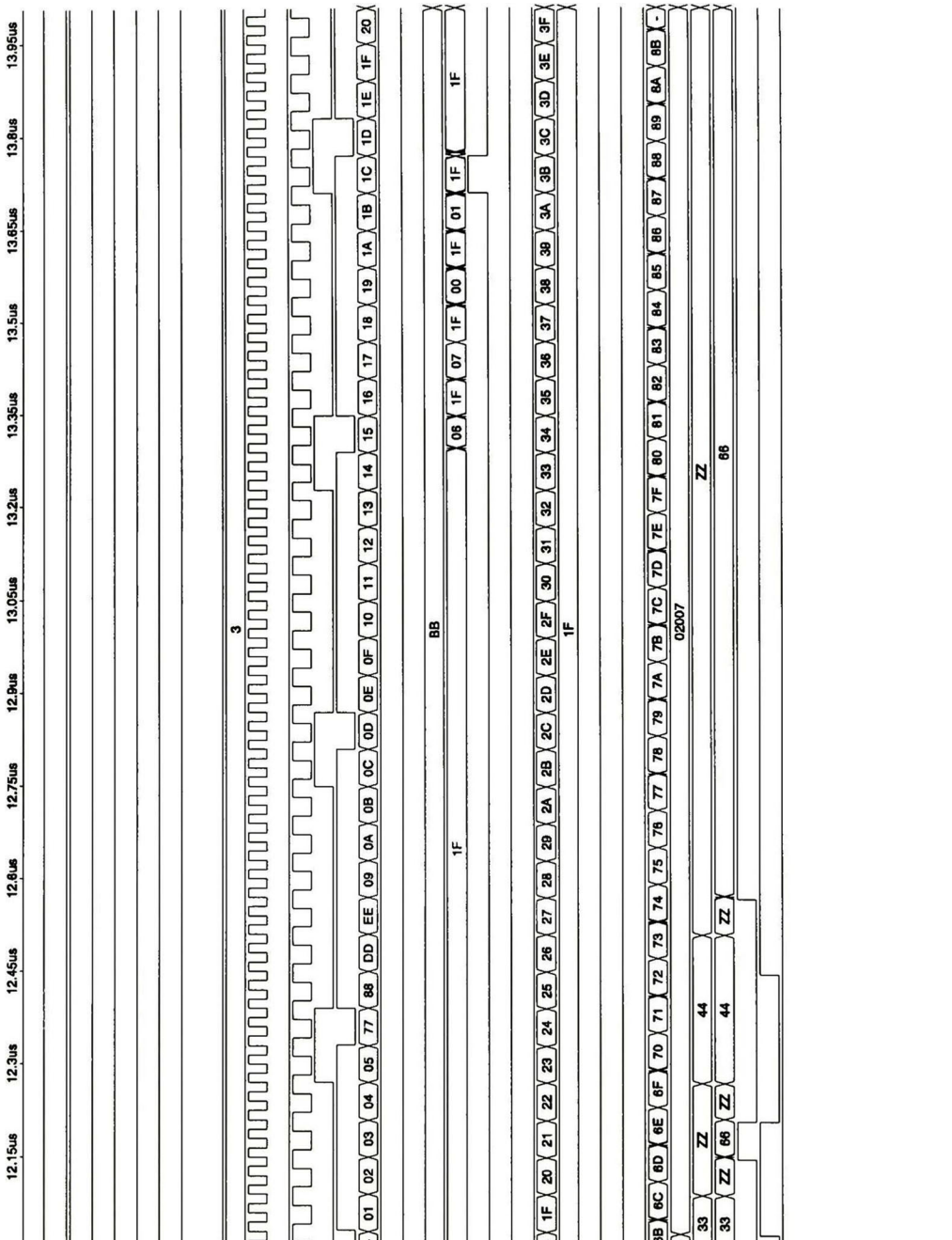

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\IUT53.SCF Date: 08/20/2000 12:58:46 Page: 2
Name:
2.1us
2.25us
2.4us
2.55us
2.7us
2.85us
3.0us
3.15us
3.3us
3.45us
3.6us
3.75us
3.9us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] reset
[O] clk
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
[O] RxAddr_U
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O] TxAddr_U
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
[O] Address
[I] Dato
[O] Dato
[O] OE
[O] CE
[O] WE
3
19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 00 01 06 50
XX
1F
03 1F 04 1F 05 1F
1F
05 1F
66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81 82 83 84 00 20
1F
00
XXXXX
ZZ
XX

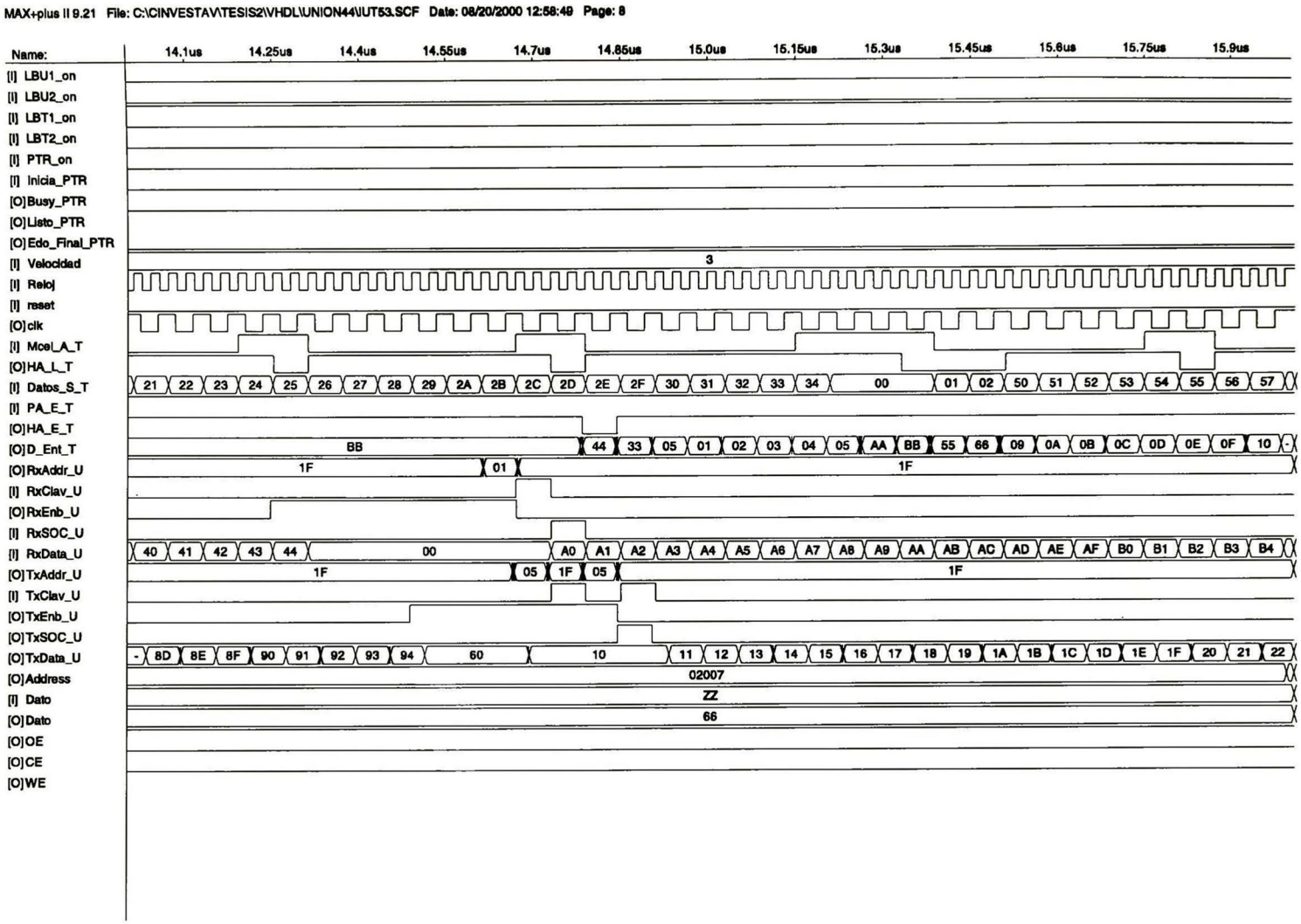

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT53.SCF Date: 08/20/2000 12:58:47 Page: 3
Name:
4.05us
4.2us
4.35us
4.5us
4.65us
4.8us
4.95us
5.1us
5.25us
5.4us
5.55us
5.7us
5.85us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] reset
[O] clk
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
[O] RxAddr_U
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O] TxAddr_U
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
[O] Address
[I] Dato
[O] Dato
[O] OE
[O] CE
[O] WE
3
XX
1F
XXXXX
01E04
01E05
04466
04467
ZZ
AA
BB
XX
77
66

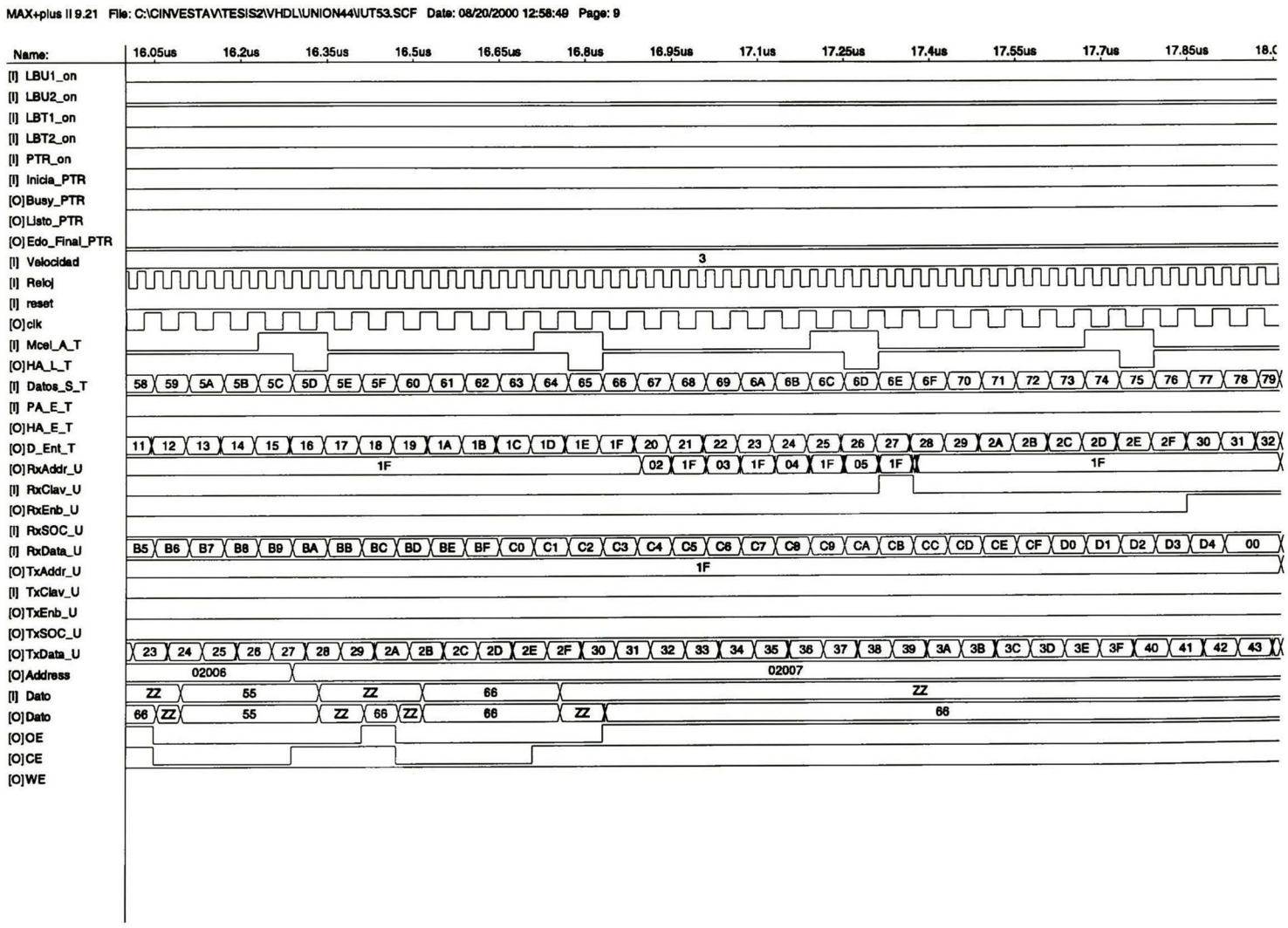

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT53.SCF Data: 08/20/2000 12:58:47 Page: 4
Name:
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] reset
[O] clk
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
[O] RxAddr_U
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O] TxAddr_U
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
[O] Address
[I] Dato
[O] Dato
[O] OE
[O] CE
[O] WE
6.15us
6.3us
6.45us
6.6us
6.75us
6.9us
7.05us
7.2us
7.35us
7.5us
7.65us
7.8us
7.95us

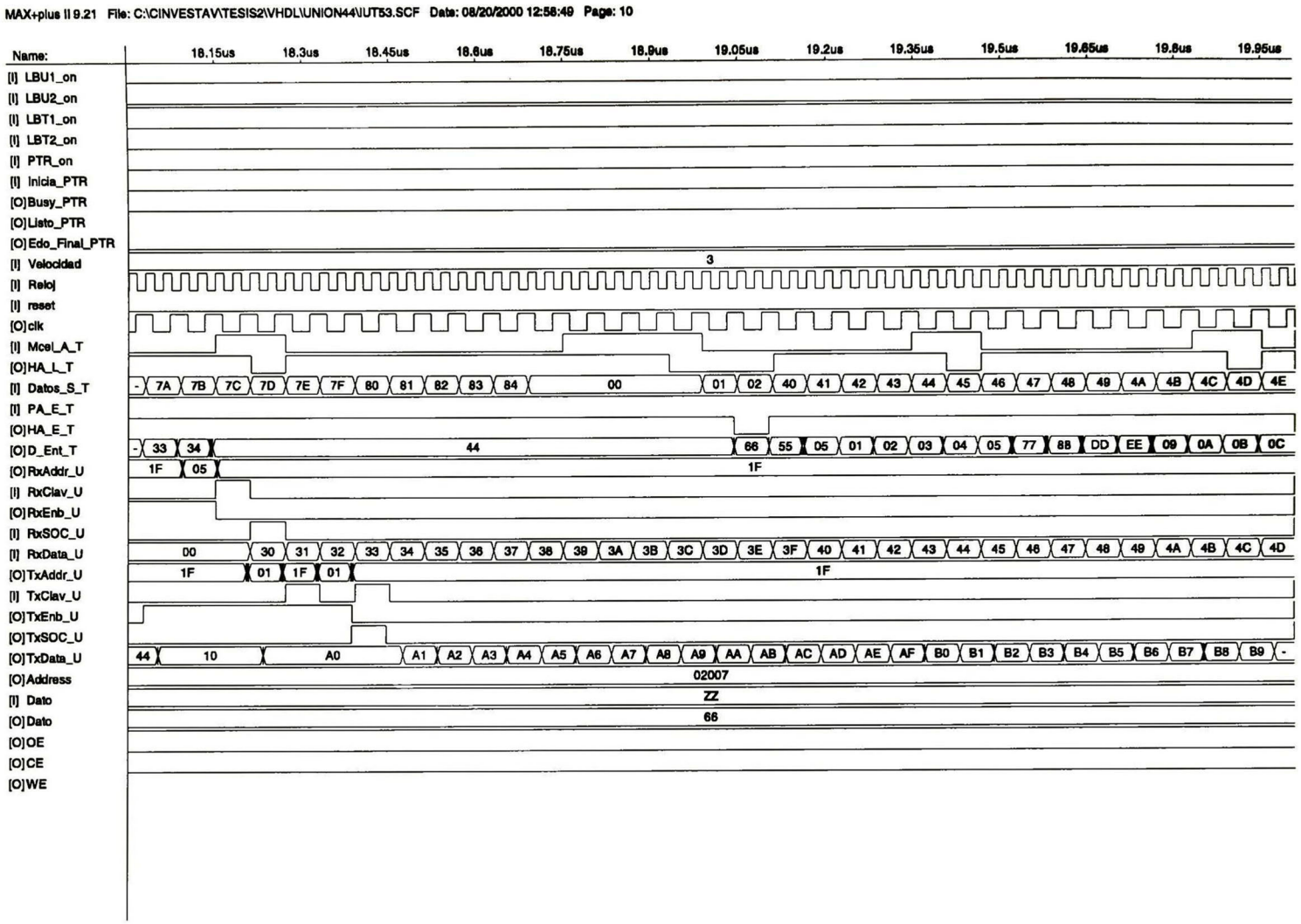

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT53.SCF Date: 08/20/2000 12:58:47 Page: 5
Name:
8.1us
8.25us
8.4us
8.55us
8.7us
8.85us
9.0us
9.15us
9.3us
9.45us
9.6us
9.75us
9.9us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] reset
[O] clk
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
[O] RxAddr_U
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O] TxAddr_U
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
[O] Address
[I] Dato
[O] Dato
[O] OE
[O] CE
[O] WE
3
1F
04467
ZZ
66

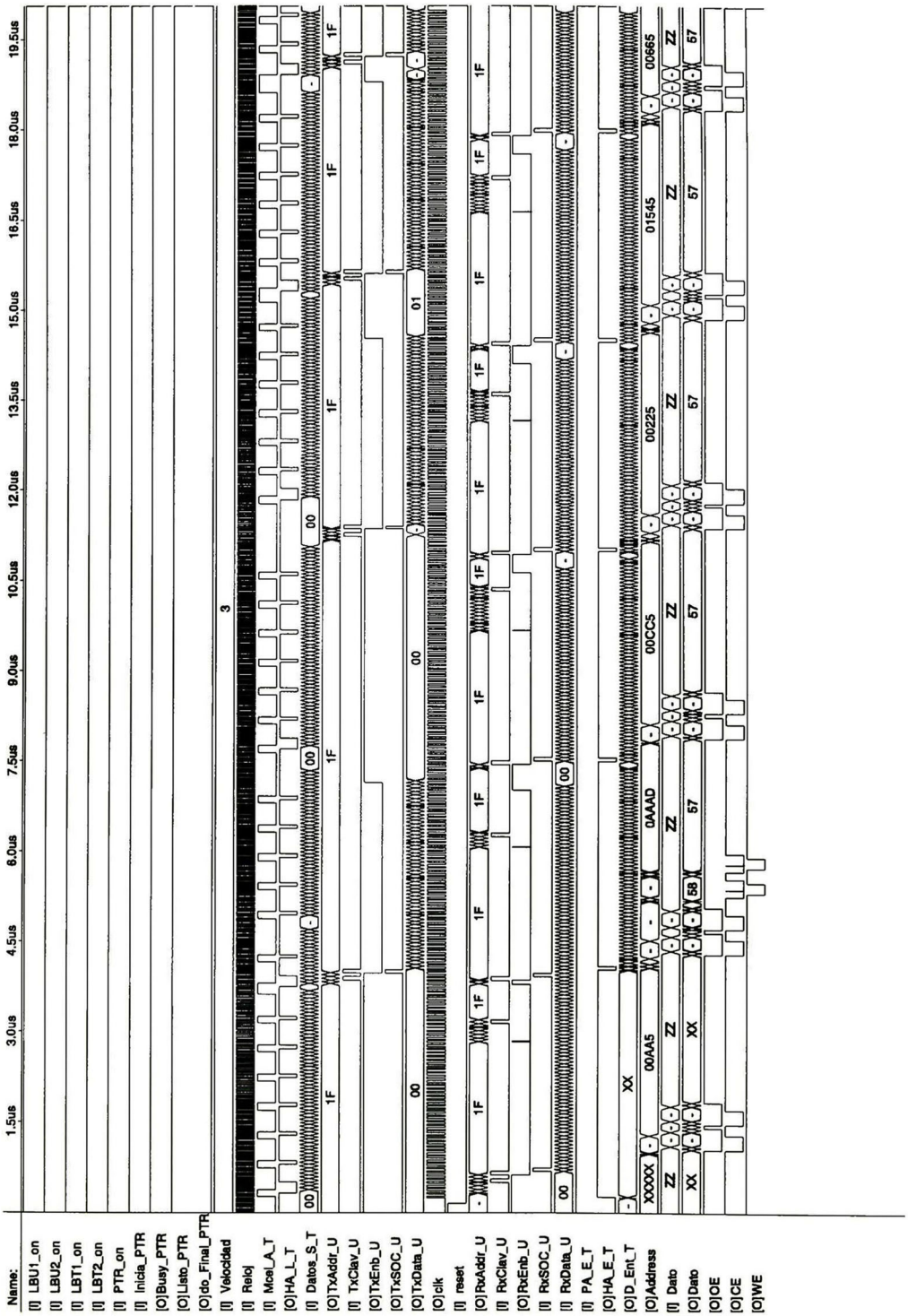

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT53.SCF Date: 08/20/2000 12:58:48 Page: 6
Name:
10.05us
10.2us
10.35us
10.5us
10.65us
10.8us
10.95us
11.1us
11.25us
11.4us
11.55us
11.7us
11.85us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
3
[I] Reloj
[I] reset
[O] clk
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 00 01 00
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 BB
[O] RxAddr_U
1F 02 1F 03 1F 04 1F 05 1F 1F 05 1F
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 00 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E
[O] TxAddr_U
1F 07 1F 07 1F
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 20 60 61 62 63 64 65 66 67 68 69 6A
[O] Address
04467 02006
[I] Dato
ZZ 33
[O] Dato
66 ZZ 33
[O] OE
[O] CE
[O] WE

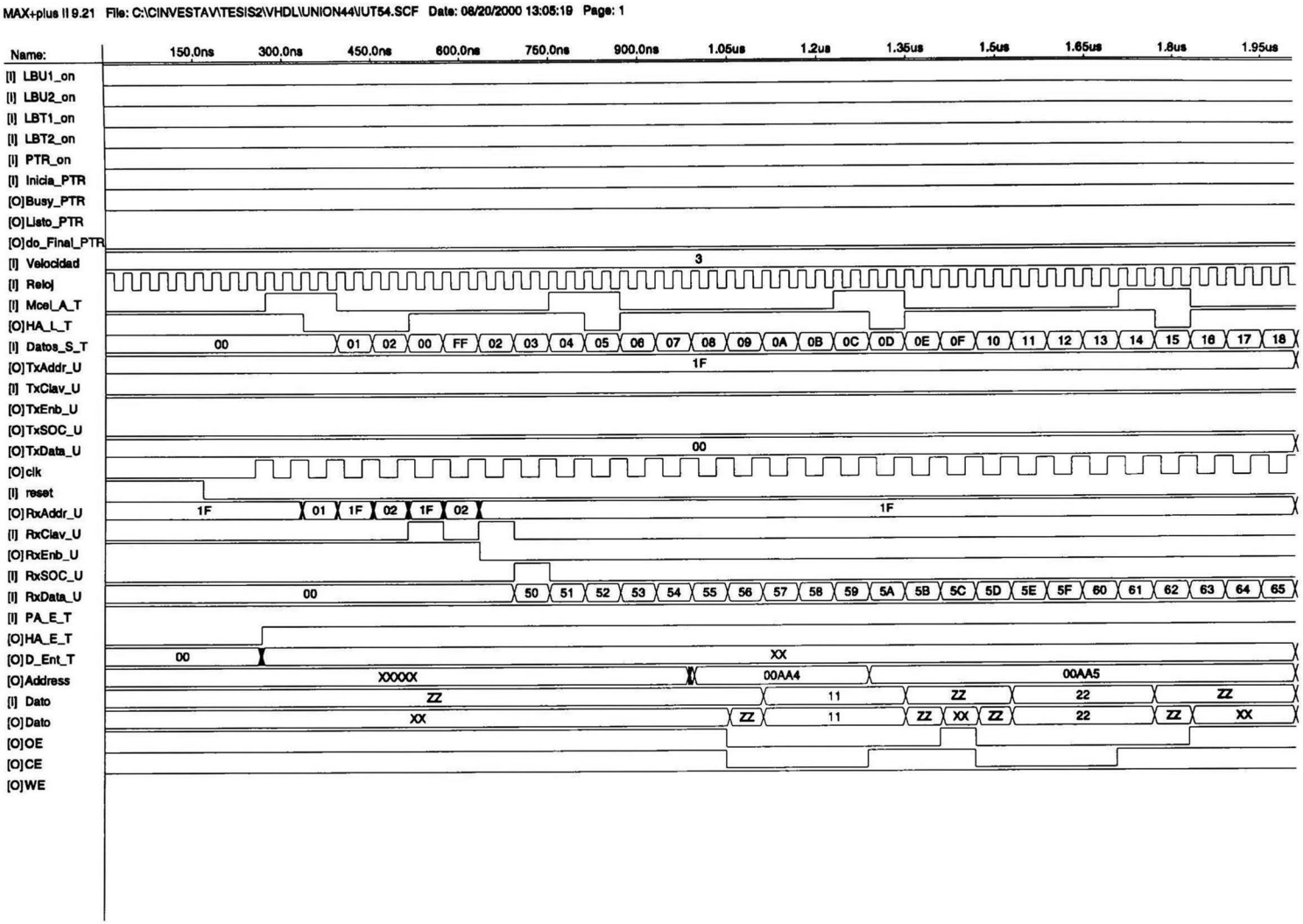

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS\2\VHDL\UNION44\UT53.SCF Data: 06/20/2000 12:58:48 Page: 7
Name:
12.15us 12.3us 12.45us 12.6us 12.75us 12.9us 13.05us 13.2us 13.35us 13.5us 13.65us 13.8us 13.95us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] reset
[O]clk
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

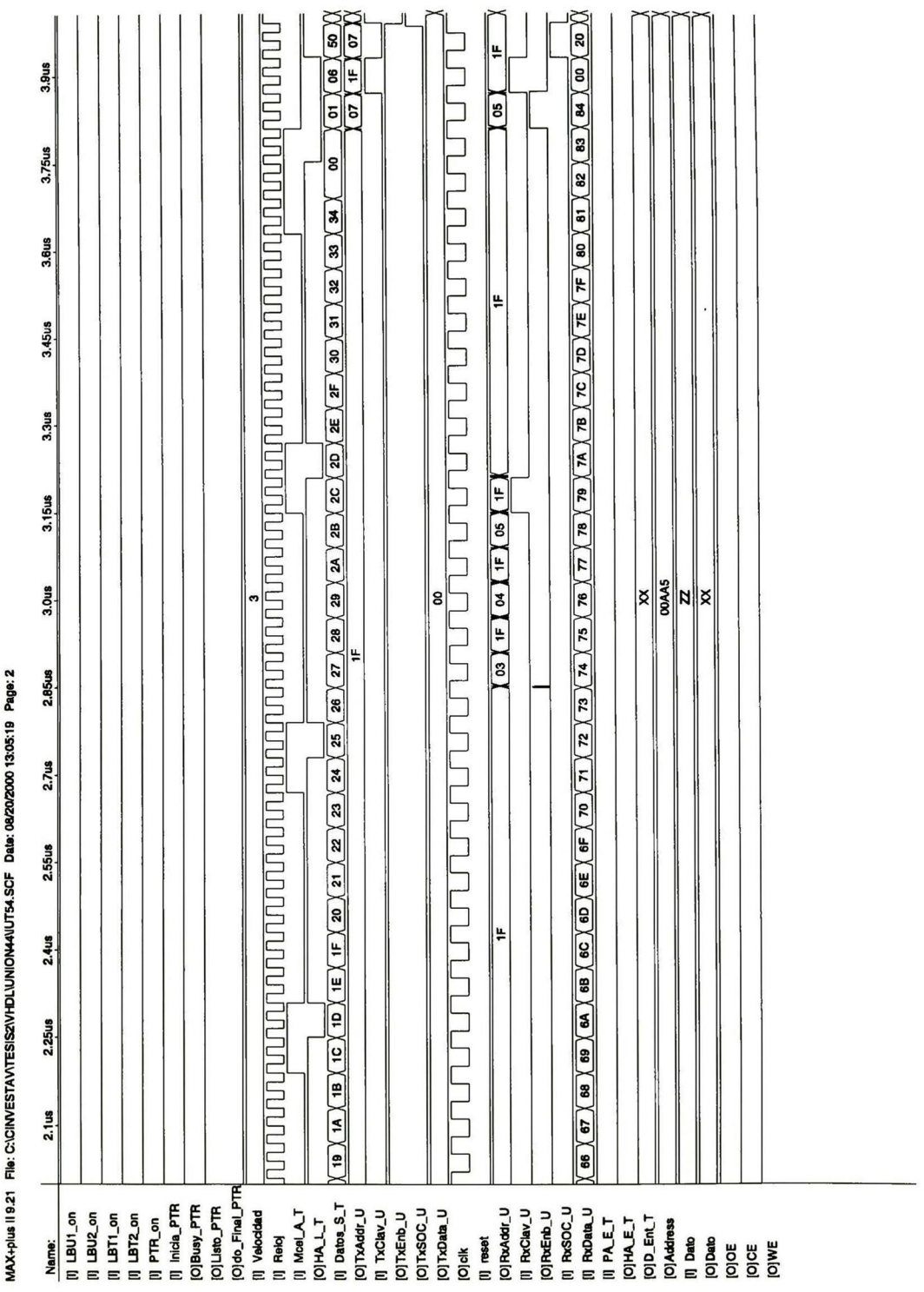

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\IUT53.SCF Date: 08/20/2000 12:58:49 Page: 8
Name:
14.1us
14.25us
14.4us
14.55us
14.7us
14.85us
15.0us
15.15us
15.3us
15.45us
15.6us
15.75us
15.9us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
3
[I] Reloj
[I] reset
[O] clk
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 00 01 02 50 51 52 53 54 55 56 57
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
BB 44 33 05 01 02 03 04 05 AA BB 55 66 09 0A 0B 0C 0D 0E 0F 10
[O] RxAddr_U
1F 01 1F
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
40 41 42 43 44 00 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4
[O] TxAddr_U
1F 05 1F 05 1F
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
8D 8E 8F 90 91 92 93 94 60 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22
[O] Address
02007
[I] Dato
ZZ
[O] Dato
66
[O] OE
[O] CE
[O] WE

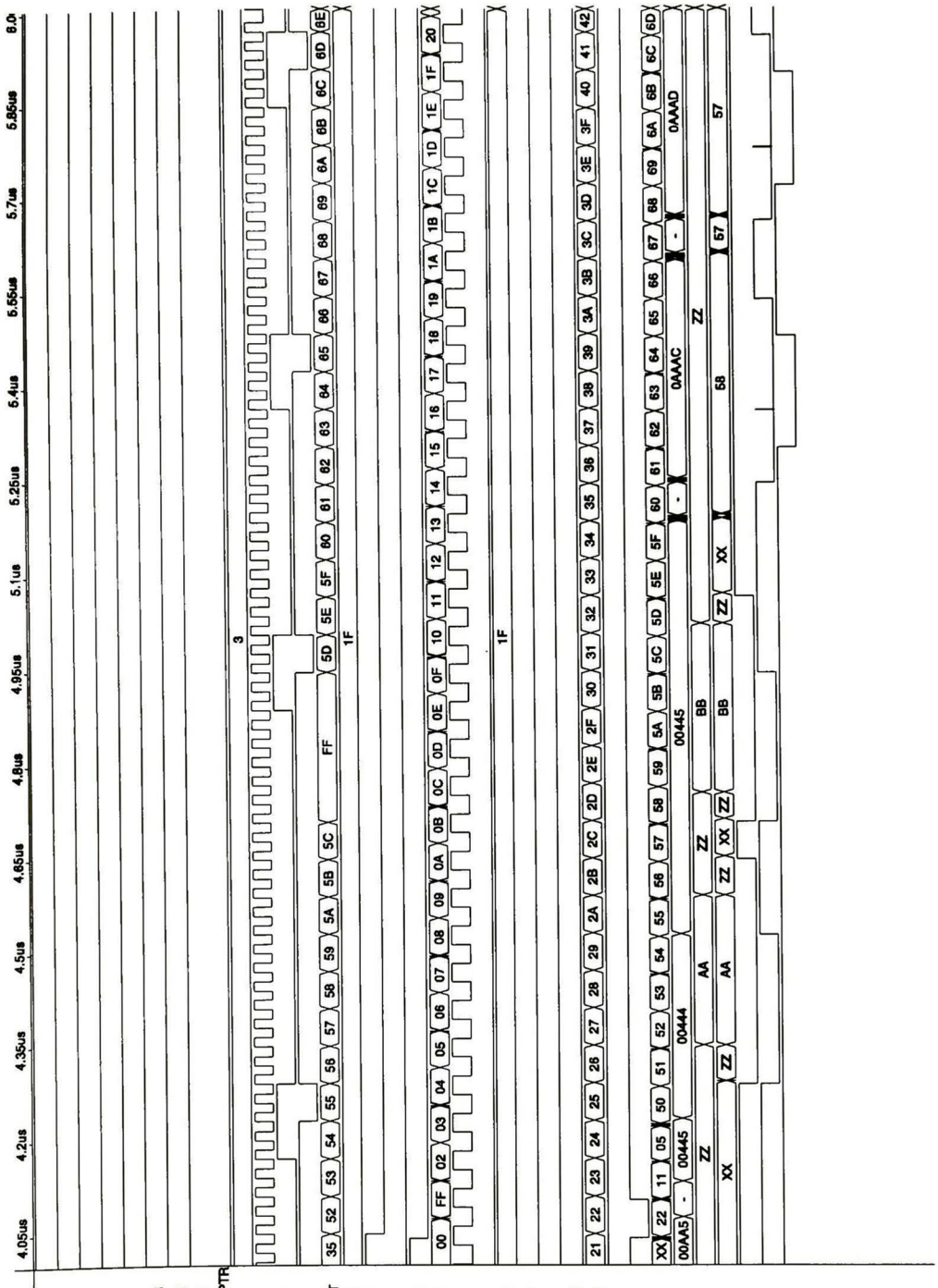

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT53.SCF Date: 08/20/2000 12:58:49 Page: 9
Name:
16.05us
16.2us
16.35us
16.5us
16.65us
16.8us
16.95us
17.1us
17.25us
17.4us
17.55us
17.7us
17.85us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O] Busy_PTR
[O] Listo_PTR
[O] Edo_Final_PTR
[I] Velocidad
[I] Reloj
[I] reset
[O] clk
[I] Mcel_A_T
[O] HA_L_T
[I] Datos_S_T
[I] PA_E_T
[O] HA_E_T
[O] D_Ent_T
[O] RxAddr_U
[I] RxClav_U
[O] RxEnb_U
[I] RxSOC_U
[I] RxData_U
[O] TxAddr_U
[I] TxClav_U
[O] TxEnb_U
[O] TxSOC_U
[O] TxData_U
[O] Address
[I] Dato
[O] Dato
[O] OE
[O] CE
[O] WE
3
1F
02
1F
03
1F
04
1F
05
1F
1F
02006
02007
55
66
ZZ


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\IUT53.SCF Date: 08/20/2000 12:58:49 Page: 10
Name:
18.15us
18.3us
18.45us
18.6us
18.75us
18.9us
19.05us
19.2us
19.35us
19.5us
19.65us
19.8us
19.95us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]Edo_Final_PTR
[I] Velocidad
3
[I] Reloj
[I] reset
[O]clk
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
7A 7B 7C 7D 7E 7F 80 81 82 83 84 00 01 02 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
33 34 44 66 55 05 01 02 03 04 05 77 88 DD EE 09 0A 0B 0C
[O]RxAddr_U
1F 05 1F
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
00 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D
[O]TxAddr_U
1F 01 1F 01 1F
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
44 10 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9
[O]Address
02007
[I] Dato
ZZ
[O]Dato
66
[O]OE
[O]CE
[O]WE


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/11/2000 19:08:29 Page: 1
Name:
1.5us
3.0us
4.5us
6.0us
7.5us
9.0us
10.5us
12.0us
13.5us
15.0us
16.5us
18.0us
19.5us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

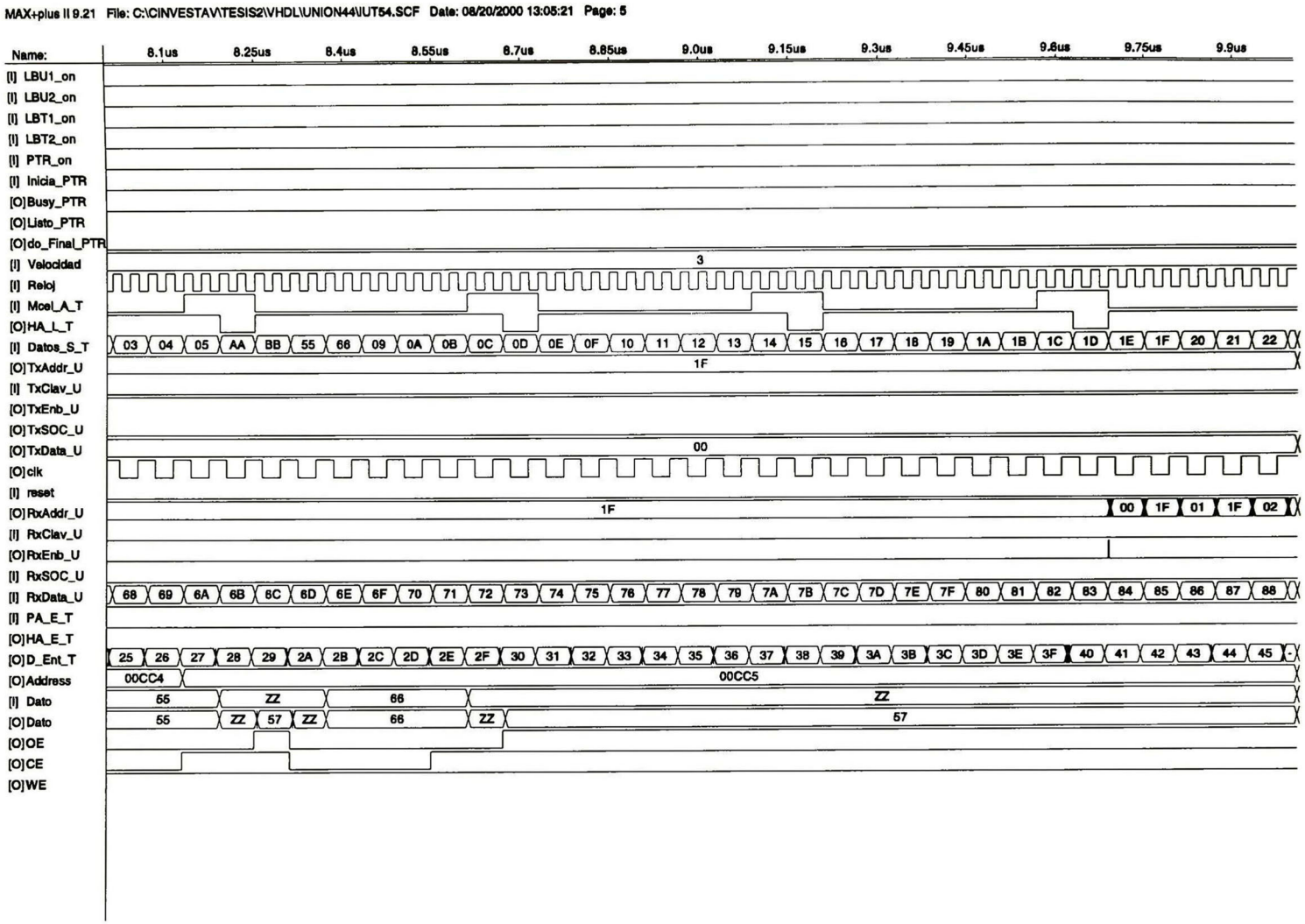

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:19 Page: 1
Name:
150.0ns
300.0ns
450.0ns
600.0ns
750.0ns
900.0ns
1.05us
1.2us
1.35us
1.5us
1.65us
1.8us
1.95us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE
3
00
01
02
00
FF
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
14
15
16
17
18
1F
00
1F
01
1F
02
1F
02
1F
00
50
51
52
53
54
55
56
57
58
59
5A
5B
5C
5D
5E
5F
60
61
62
63
64
65
00
XX
XXXXX
00AA4
00AA5
ZZ
11
ZZ
22
ZZ
XX
ZZ
11
ZZ
XX
ZZ
22
ZZ
XX

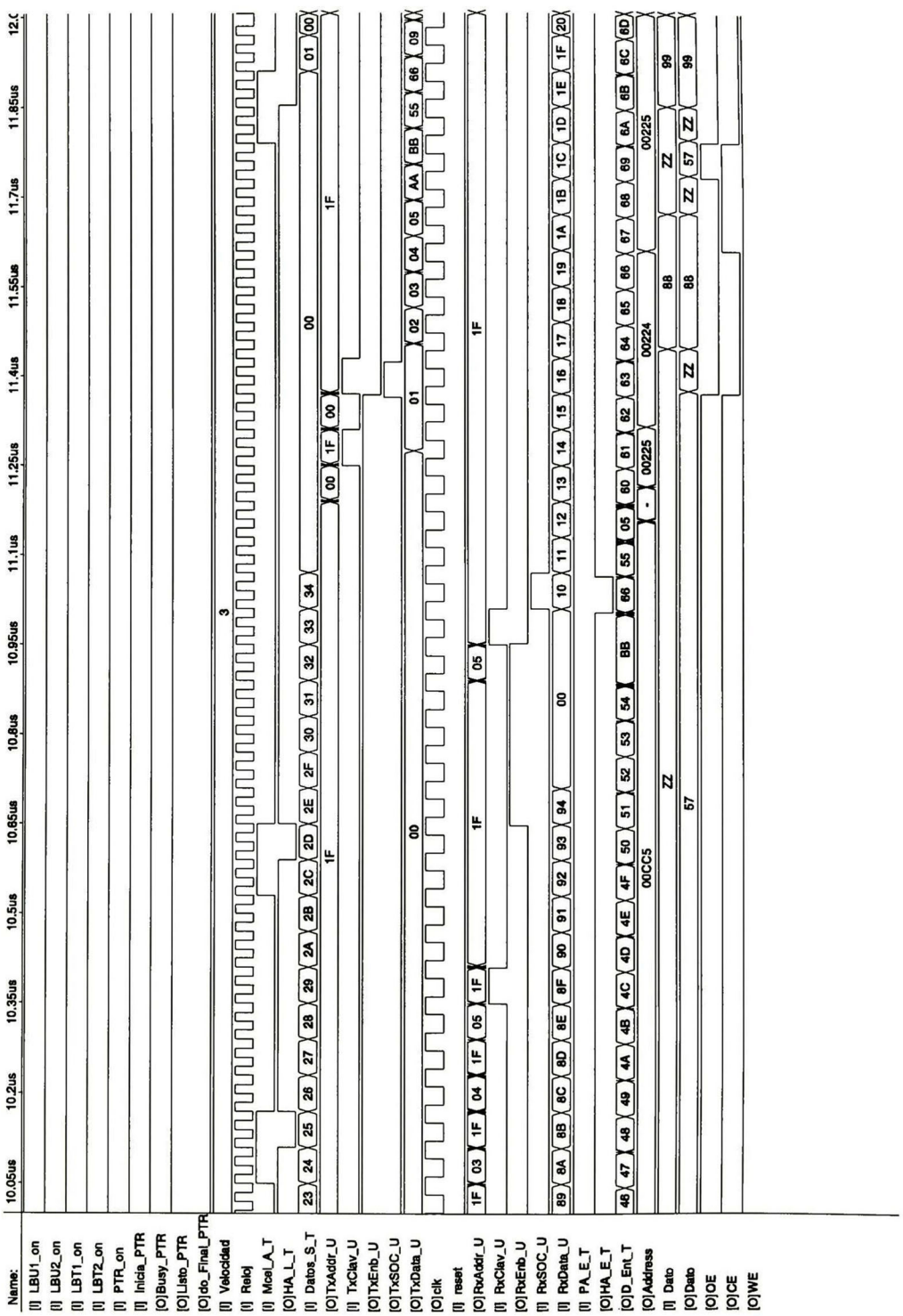

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS\2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:19 Page: 2
Name:
2.1us
2.25us
2.4us
2.55us
2.7us
2.85us
3.0us
3.15us
3.3us
3.45us
3.6us
3.75us
3.9us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_En_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

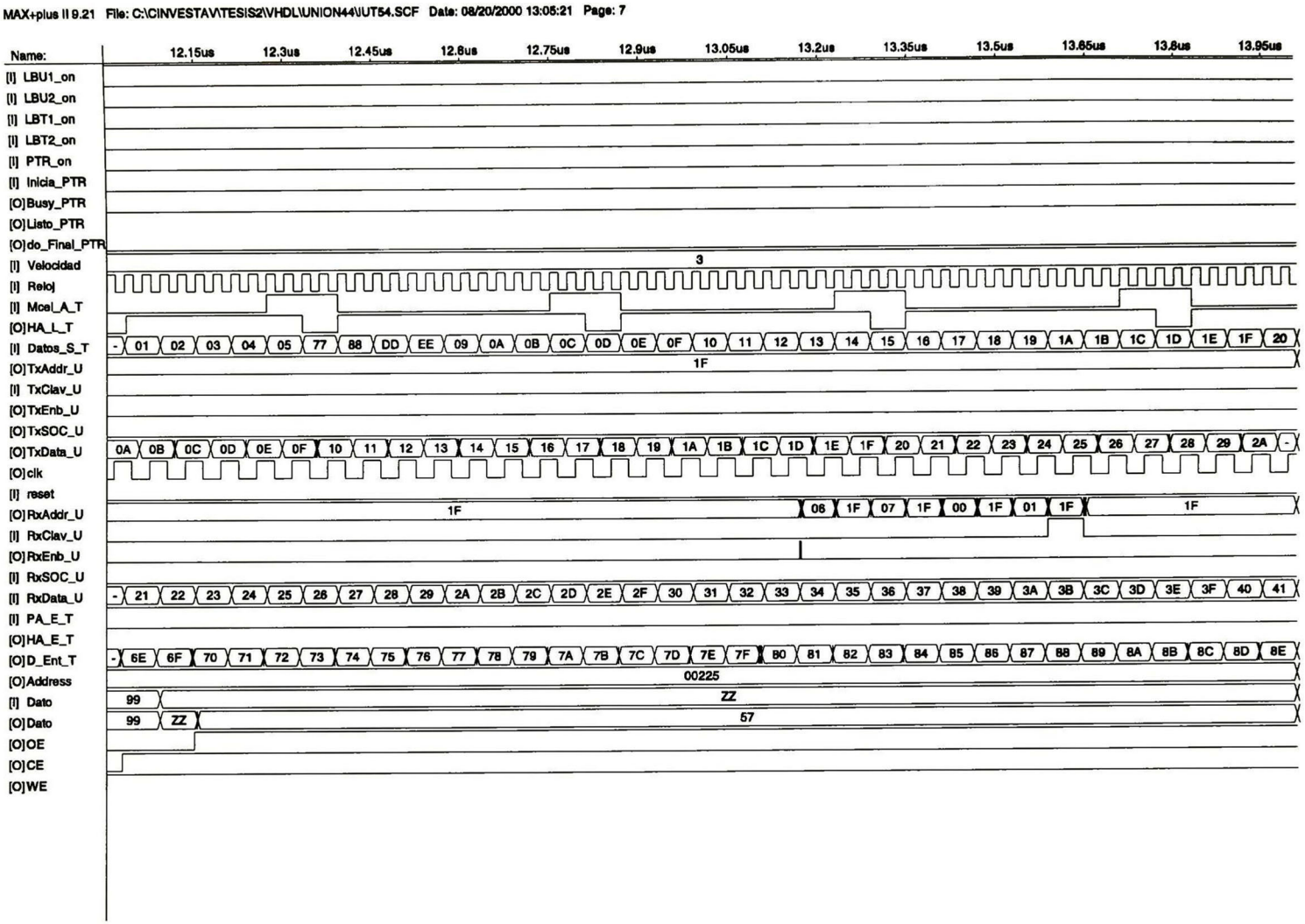


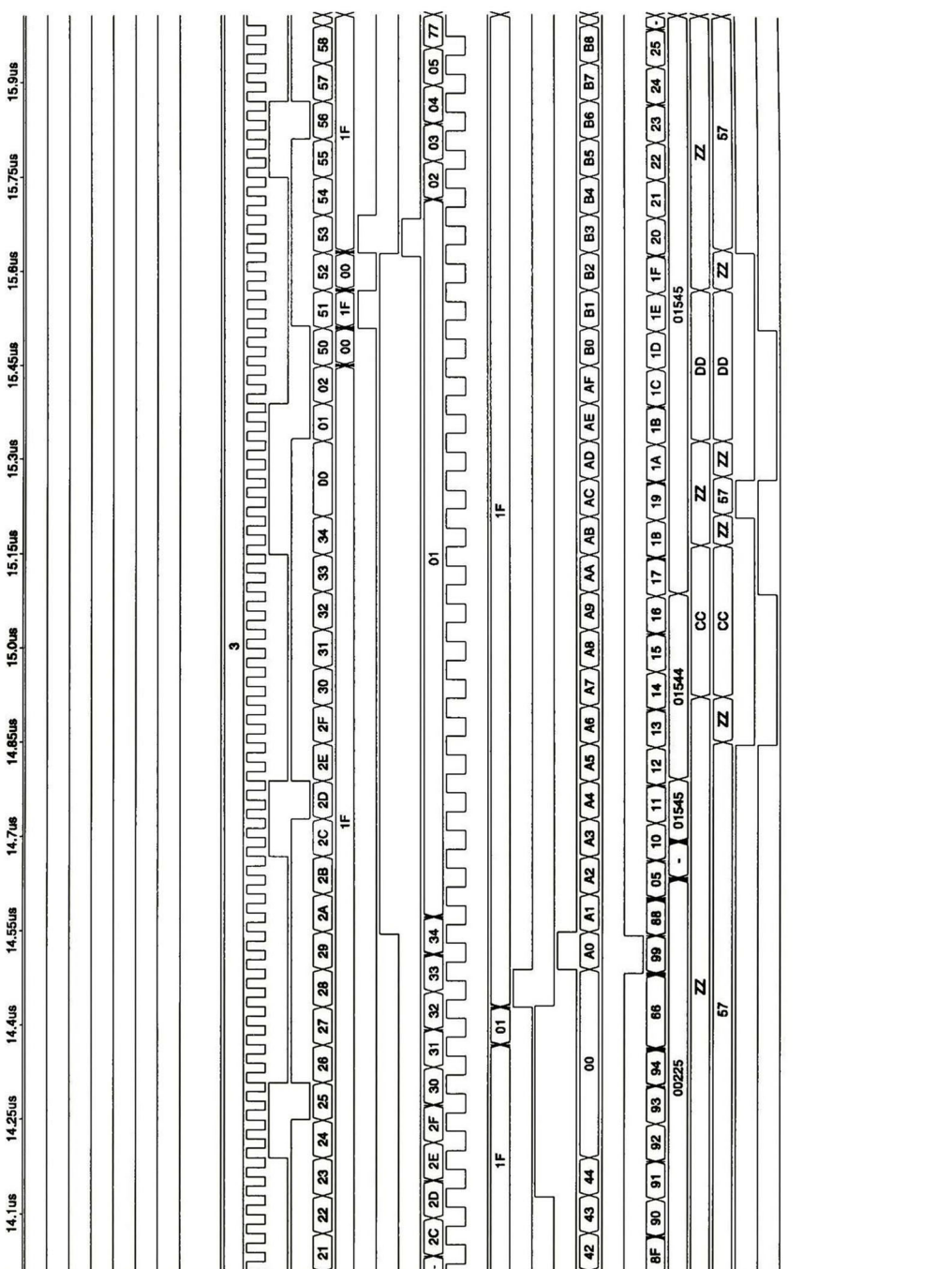
CINVESTAV IPN, Gdl.

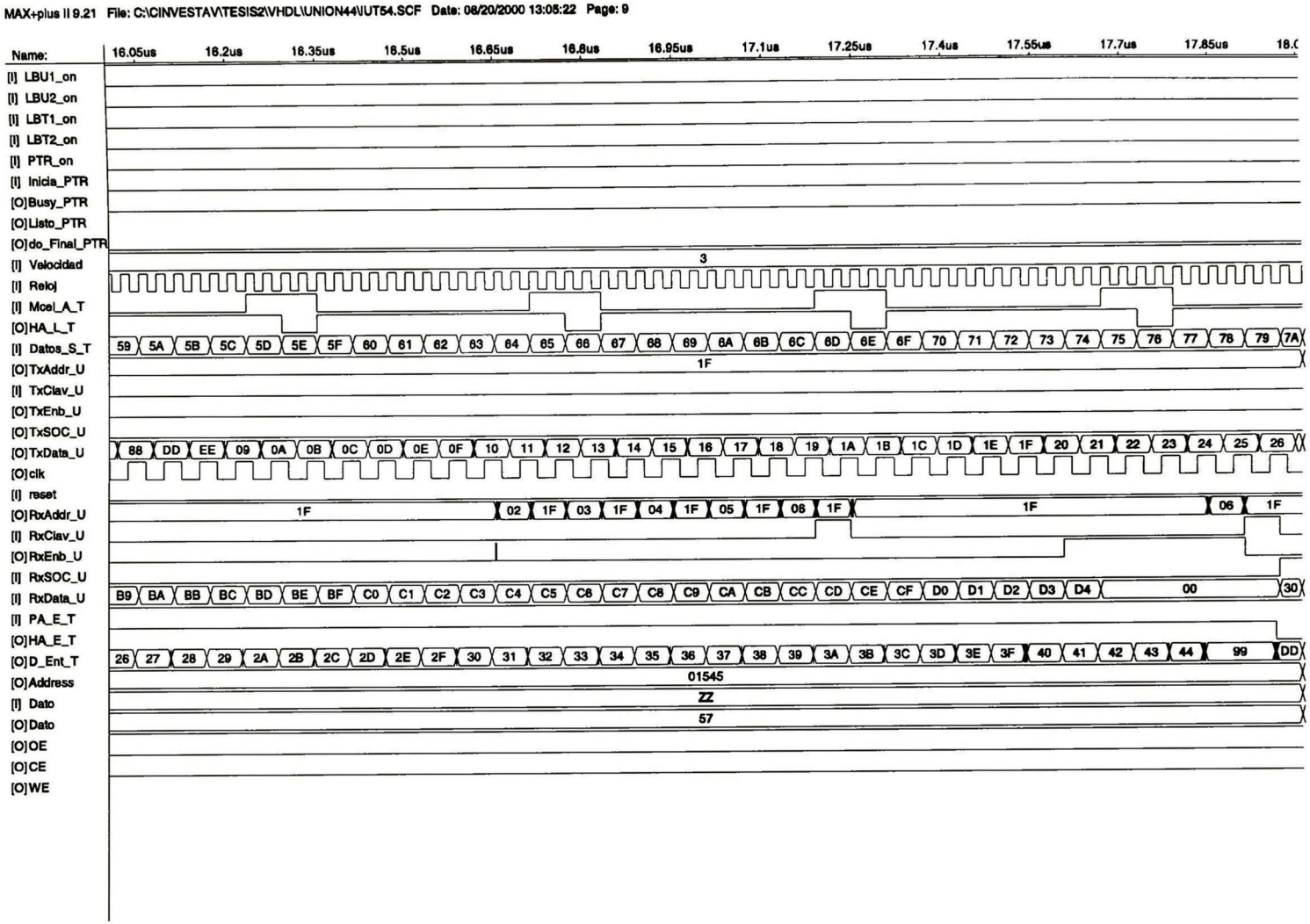
MAX+plus II 9.21 File: C:\CINVESTAV\TESIS\2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:20 Page: 4
Name:
6.15us
6.3us
6.45us
6.6us
6.75us
6.9us
7.05us
7.2us
7.35us
7.5us
7.65us
7.8us
7.95us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] MceI_A_T
[O]HA_I_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_EnI_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

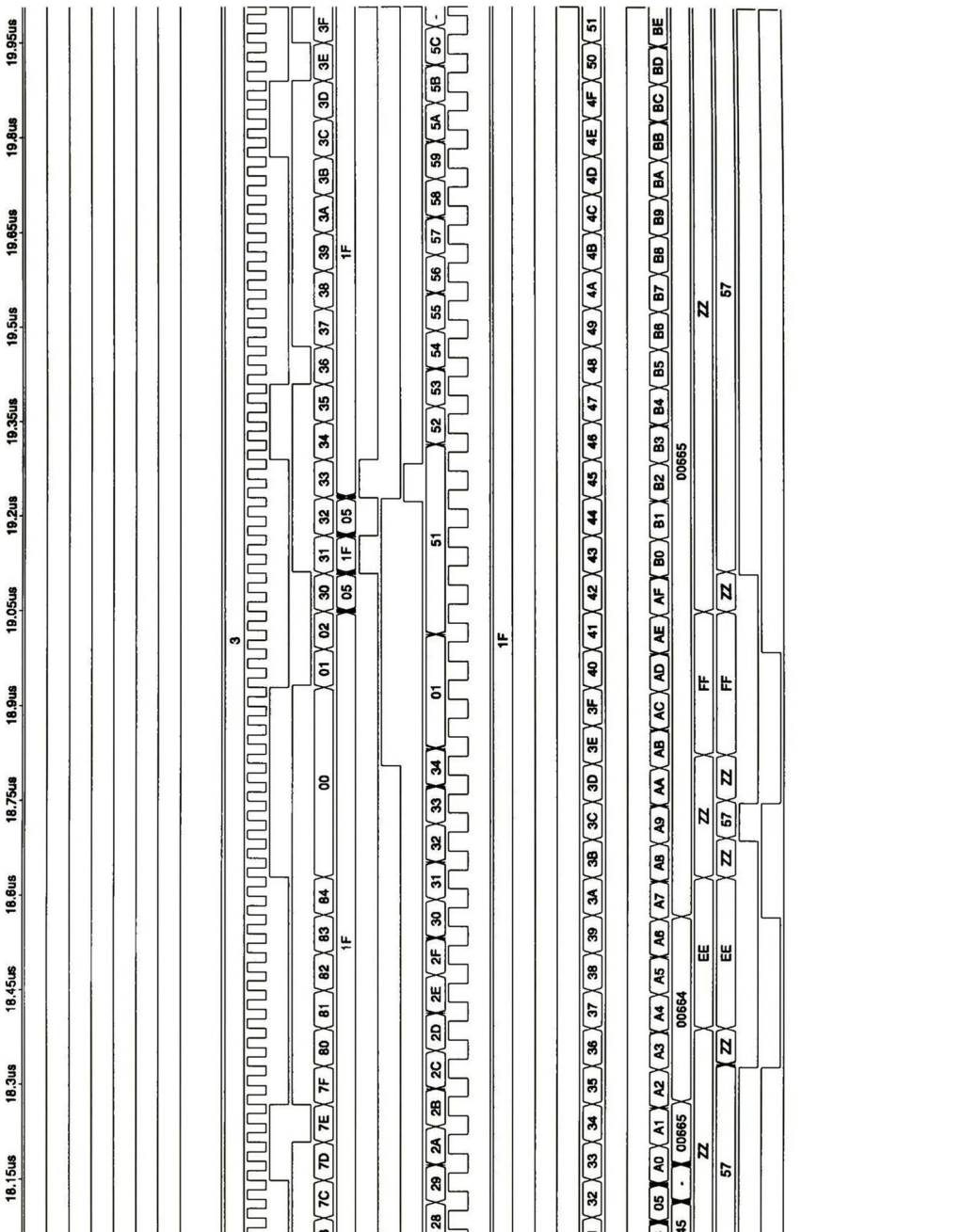

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:21 Page: 5
Name:
8.1us
8.25us
8.4us
8.55us
8.7us
8.85us
9.0us
9.15us
9.3us
9.45us
9.6us
9.75us
9.9us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE
3
1F
00
1F
00CC4
00CC5
55
ZZ
66
ZZ
55
ZZ
57
ZZ
66
ZZ
57

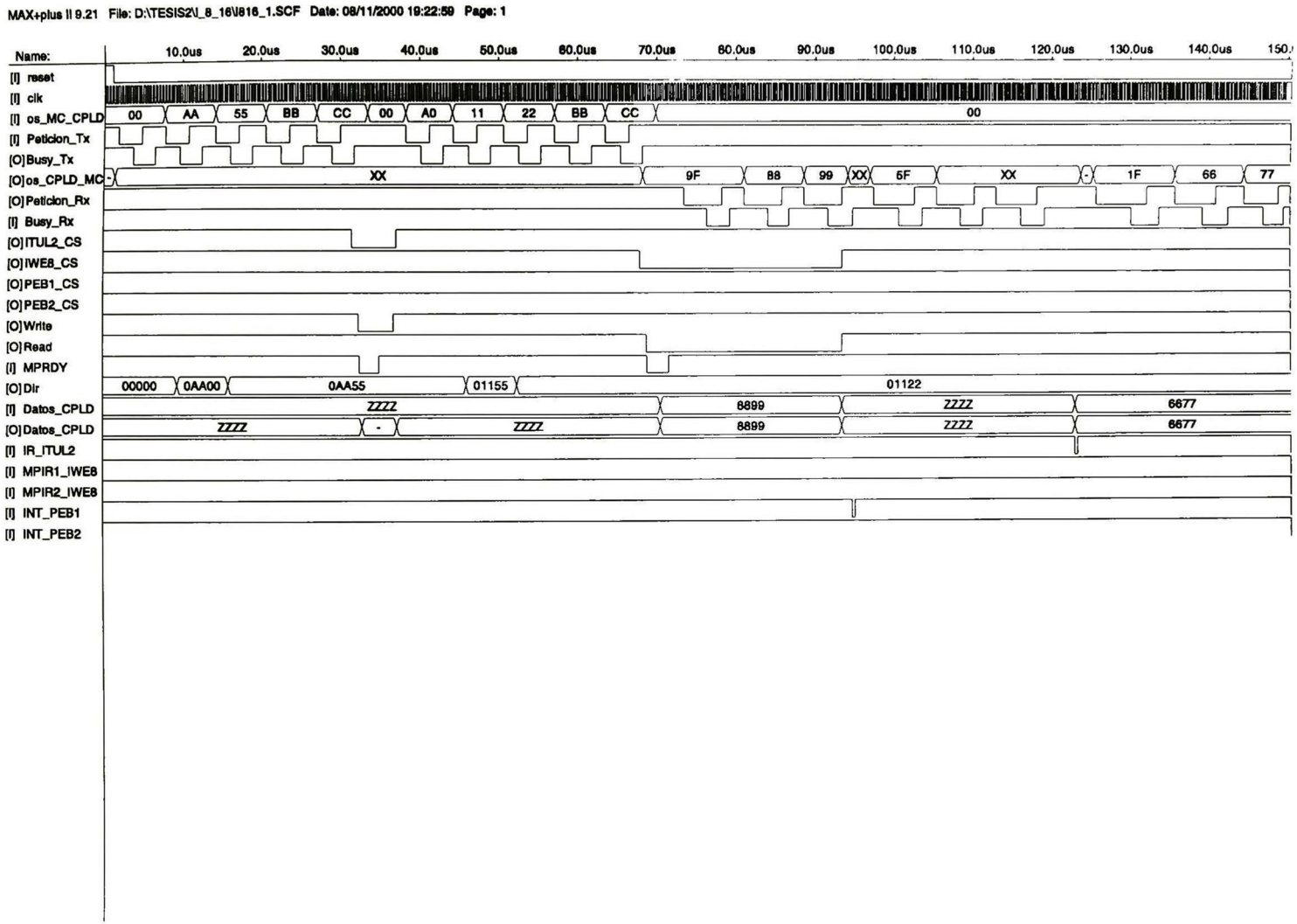

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:21 Page: 6
Name:
10.05us
10.2us
10.35us
10.5us
10.65us
10.8us
10.95us
11.1us
11.25us
11.4us
11.55us
11.7us
11.85us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] MicelA_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_EnL_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

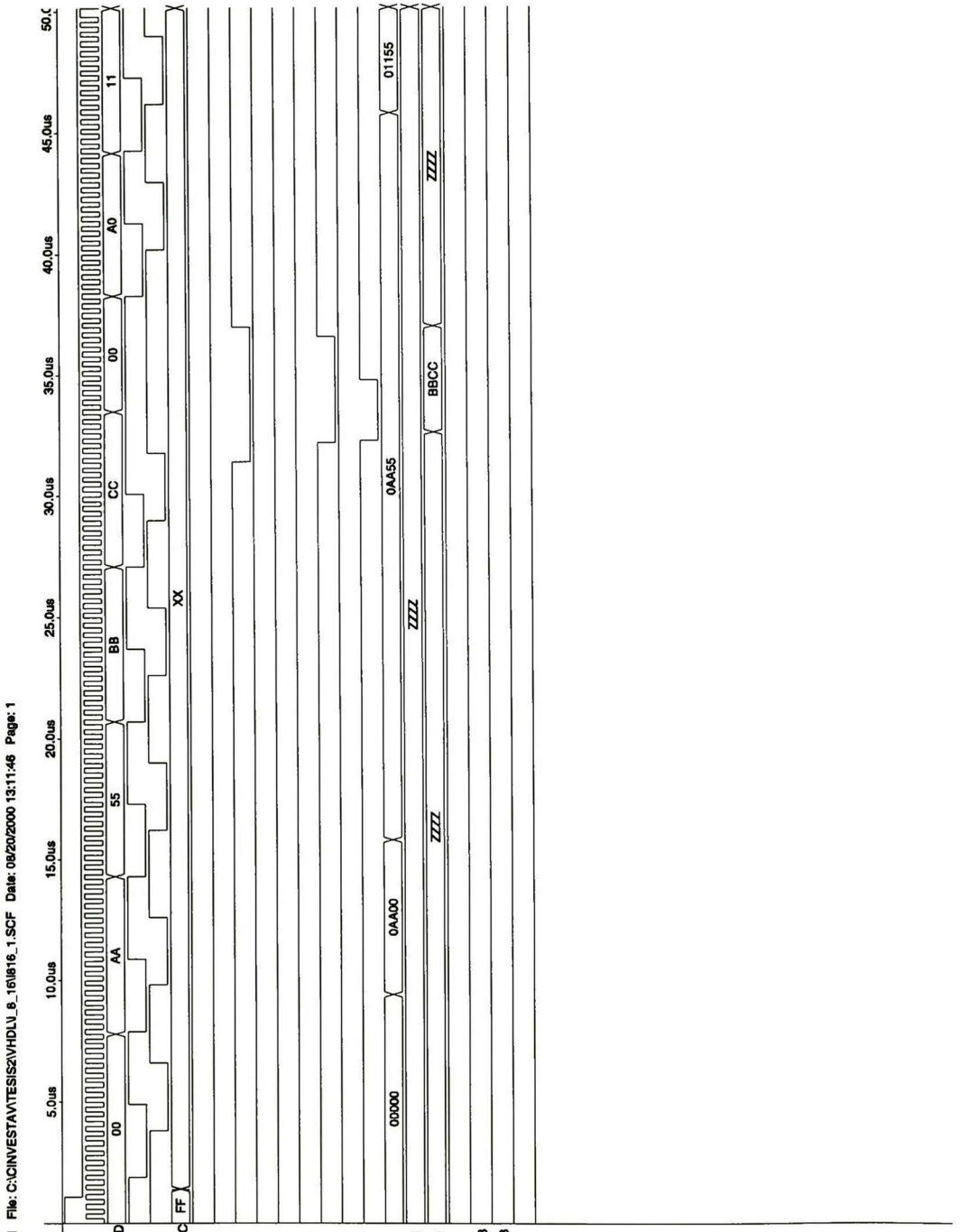

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:21 Page: 7
Name:
12.15us
12.3us
12.45us
12.6us
12.75us
12.9us
13.05us
13.2us
13.35us
13.5us
13.65us
13.8us
13.95us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
3
[I] Reloj
[I] Mcal_A_T
[O]HA_L_T
[I] Datos_S_T
01 02 03 04 05 77 88 DD EE 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20
[O]TxAddr_U
1F
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A
[O]clk
[I] reset
[O]RxAddr_U
1F
06 1F 07 1F 00 1F 01 1F 1F
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E
[O]Address
00225
[I] Dato
99
ZZ
[O]Dato
99
ZZ
57
[O]OE
[O]CE
[O]WE

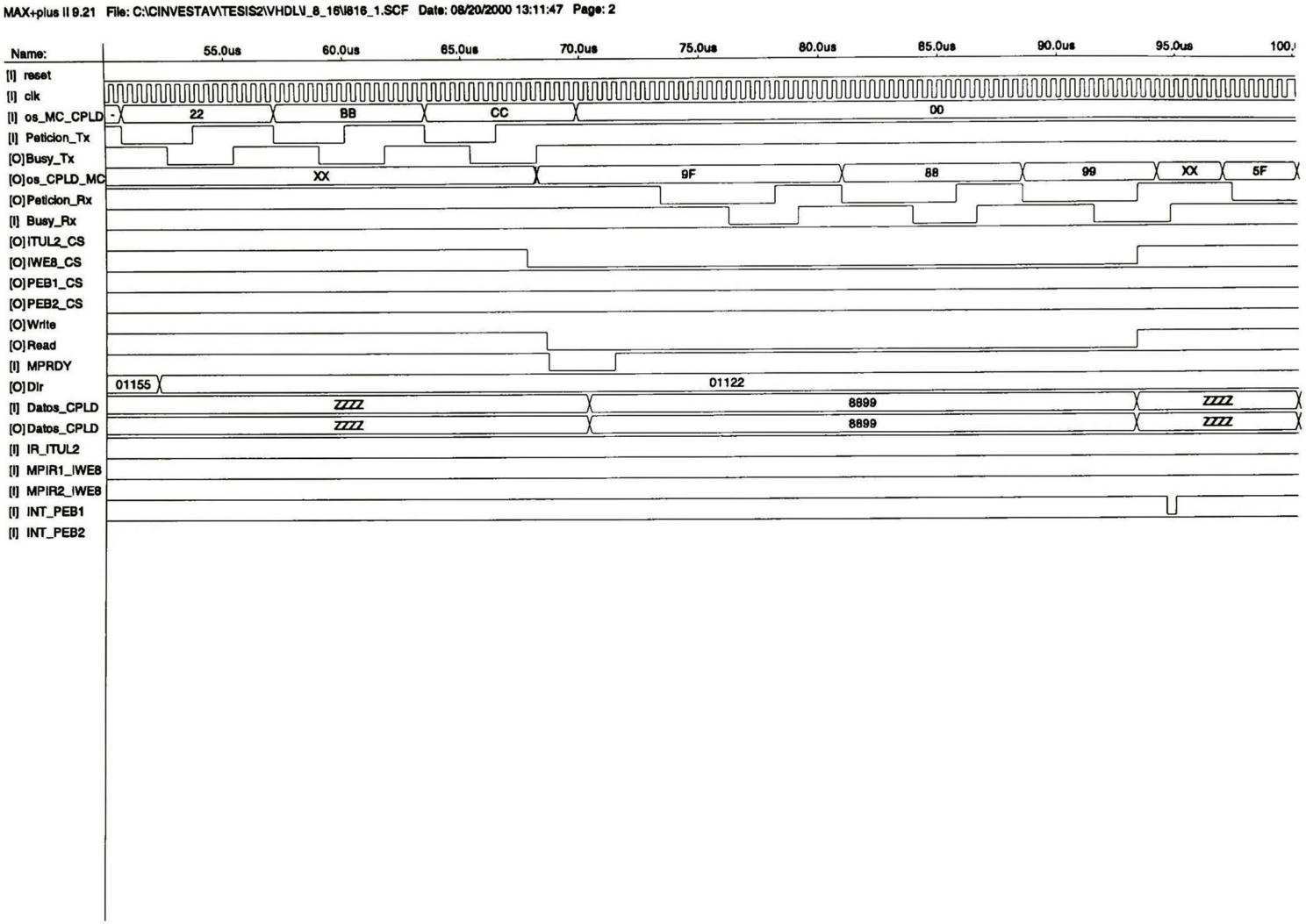

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:22 Page: 8
Name:
14.1us
14.25us
14.4us
14.55us
14.7us
14.85us
15.0us
15.15us
15.3us
15.45us
15.6us
15.75us
15.9us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE

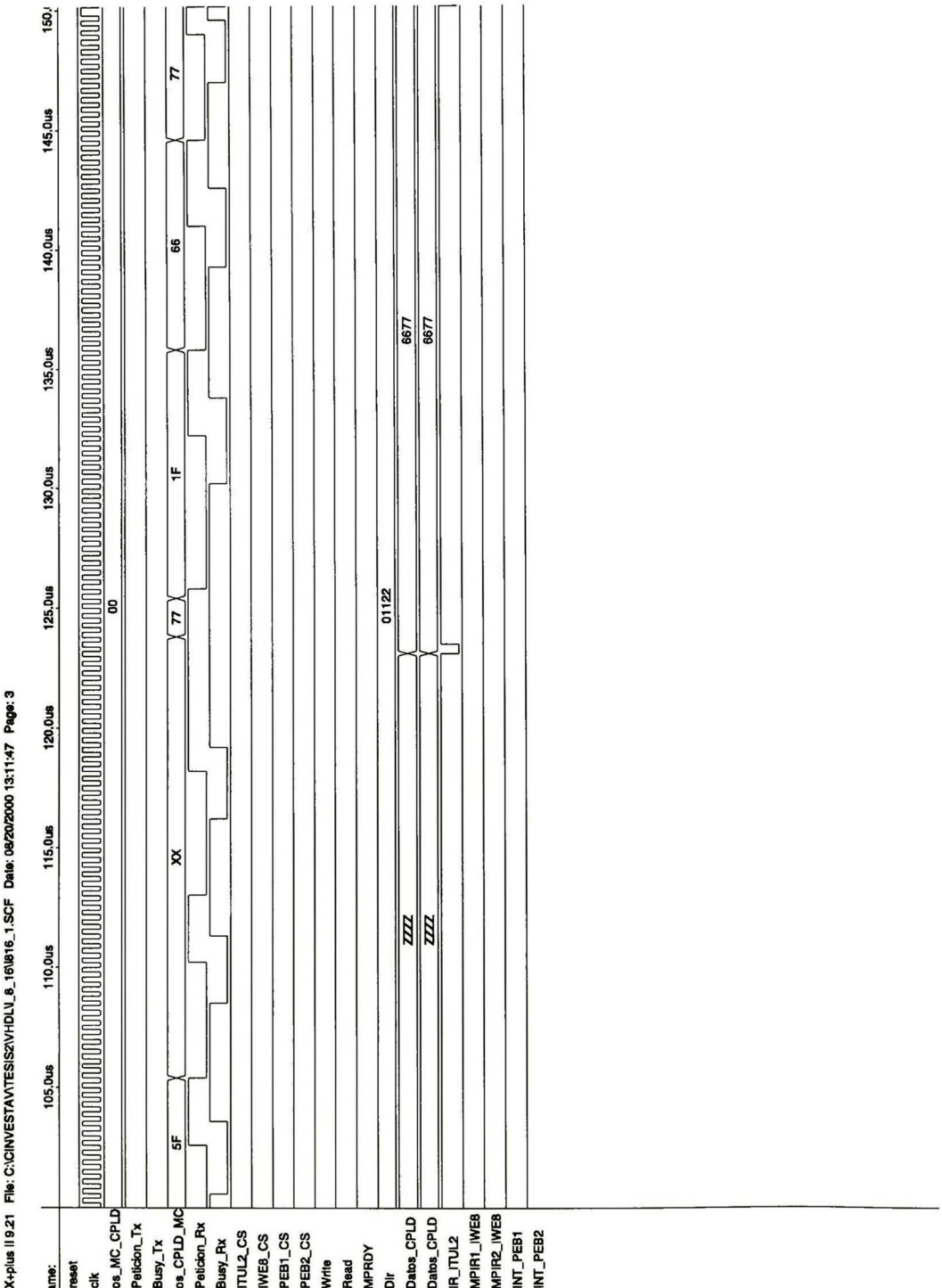

MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Date: 08/20/2000 13:05:22 Page: 9
Name:
16.05us
16.2us
16.35us
16.5us
16.65us
16.8us
16.95us
17.1us
17.25us
17.4us
17.55us
17.7us
17.85us
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] Mcel_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE
3
1F
01545
ZZ
57


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\UNION44\UT54.SCF Data: 08/20/2000 13:05:23 Page: 10
Name:
[I] LBU1_on
[I] LBU2_on
[I] LBT1_on
[I] LBT2_on
[I] PTR_on
[I] Inicia_PTR
[O]Busy_PTR
[O]Listo_PTR
[O]do_Final_PTR
[I] Velocidad
[I] Reloj
[I] MoeL_A_T
[O]HA_L_T
[I] Datos_S_T
[O]TxAddr_U
[I] TxClav_U
[O]TxEnb_U
[O]TxSOC_U
[O]TxData_U
[O]clk
[I] reset
[O]RxAddr_U
[I] RxClav_U
[O]RxEnb_U
[I] RxSOC_U
[I] RxData_U
[I] PA_E_T
[O]HA_E_T
[O]D_Ent_T
[O]Address
[I] Dato
[O]Dato
[O]OE
[O]CE
[O]WE
18.15us
18.3us
18.45us
18.6us
18.75us
18.9us
19.05us
19.2us
19.35us
19.5us
19.65us
19.8us
19.95us

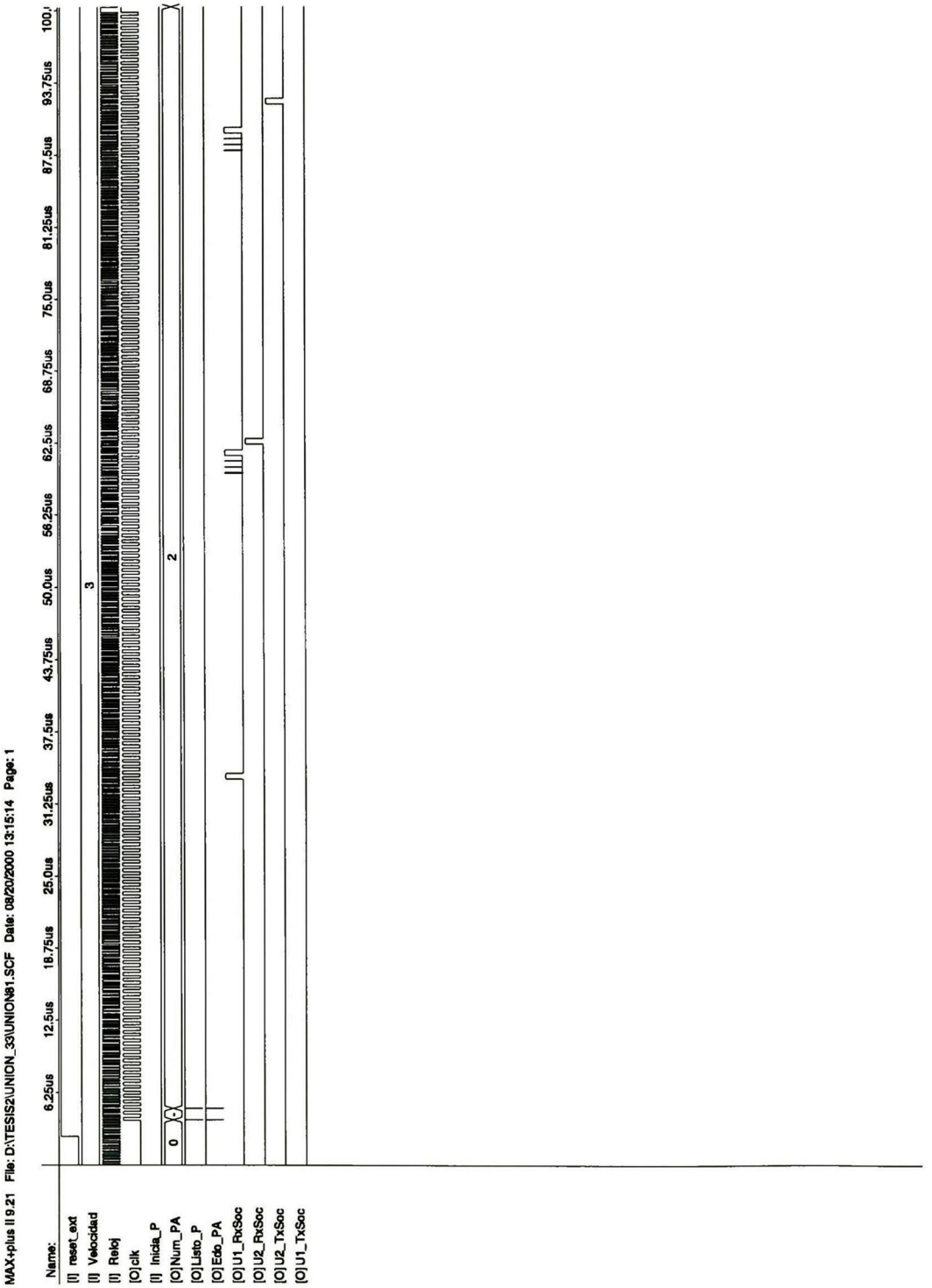

MAX+plus II 9.21 File: D:\TESIS2\I_8_16\I816_1.SCF Date: 08/11/2000 19:22:59 Page: 1
Name:
10.0us
20.0us
30.0us
40.0us
50.0us
60.0us
70.0us
80.0us
90.0us
100.0us
110.0us
120.0us
130.0us
140.0us
[I] reset
[I] clk
[I] os_MC_CPLD
[I] Peticion_Tx
[O] Busy_Tx
[O] os_CPLD_MC
[O] Peticion_Rx
[I] Busy_Rx
[O] ITUL2_CS
[O] IWE8_CS
[O] PEB1_CS
[O] PEB2_CS
[O] Write
[O] Read
[I] MPRDY
[O] Dir
[I] Datos_CPLD
[O] Datos_CPLD
[I] IR_ITUL2
[I] MPIR1_IWE8
[I] MPIR2_IWE8
[I] INT_PEB1
[I] INT_PEB2
00
AA
55
BB
CC
00
A0
11
22
BB
CC
00
XX
9F
88
99
XX
5F
XX
1F
66
77
00000
0AA00
0AA55
01155
01122
ZZZZ
8899
6677


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\U_8_16\8116_1.SCF Date: 08/20/2000 13:11:46 Page: 1
Name:
5.0us
10.0us
15.0us
20.0us
25.0us
30.0us
35.0us
40.0us
45.0us
[I] reset
[I] clk
[I] os_MC_CPLD
[I] Peticion_Tx
[O]Busy_Tx
[O]os_CPLD_MC
[O]Peticion_Rx
[I] Busy_Rx
[O]ITUL2_CS
[O] IWE8_CS
[O] PEB1_CS
[O] PEB2_CS
[O]Write
[O]Read
[I] MPRDY
[O] Dir
[I] Datos_CPLD
[O] Datos_CPLD
[I] IR_ITUL2
[I] MPIR1_IWE8
[I] MPIR2_IWE8
[I] INT_PEB1
[I] INT_PEB2
00
AA
55
BB
CC
00
A0
11
FF
XX
00000
0AA00
0AA55
01155
ZZZZ
ZZZZ
BBCC
ZZZZ


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL\I_8_16\I816_1.SCF Date: 08/20/2000 13:11:47 Page: 2
Name:
55.0us
60.0us
65.0us
70.0us
75.0us
80.0us
85.0us
90.0us
95.0us
[I] reset
[I] clk
[I] os_MC_CPLD
22
BB
CC
00
[I] Peticion_Tx
[O]Busy_Tx
[O]os_CPLD_MC
XX
9F
88
99
XX
5F
[O]Peticion_Rx
[I] Busy_Rx
[O]ITUL2_CS
[O]IWE8_CS
[O]PEB1_CS
[O]PEB2_CS
[O]Write
[O]Read
[I] MPRDY
[O]Dir
01155
01122
[I] Datos_CPLD
ZZZZ
8899
ZZZZ
[O]Datos_CPLD
ZZZZ
8899
ZZZZ
[I] IR_ITUL2
[I] MPIR1_IWE8
[I] MPIR2_IWE8
[I] INT_PEB1
[I] INT_PEB2


MAX+plus II 9.21 File: C:\CINVESTAV\TESIS2\VHDL_6_16\B16_1.SCF Date: 08/20/2000 13:11:47 Page: 3
Name:
105,0us
110,0us
115,0us
120,0us
125,0us
130,0us
135,0us
140,0us
145,0us
[I] reset
[I] clk
[I] os_MC_CPLD
[I] Peticion_Tx
[O]Busy_Tx
[O]os_CPLD_MC
[O]Peticion_Rx
[I] Busy_Rx
[O]ITUL2_CS
[O]IWE8_CS
[O]PEB1_CS
[O]PEB2_CS
[O]Write
[O]Read
[I] MPRDY
[O]Dir
[I] Datos_CPLD
[O]Datos_CPLD
[I] IR_ITUL2
[I] MPIR1_IWE8
[I] MPIR2_IWE8
[I] INT_PEB1
[I] INT_PEB2
5F
XX
77
1F
66
77
00
01122
ZZZZ
ZZZZ
6677
6677

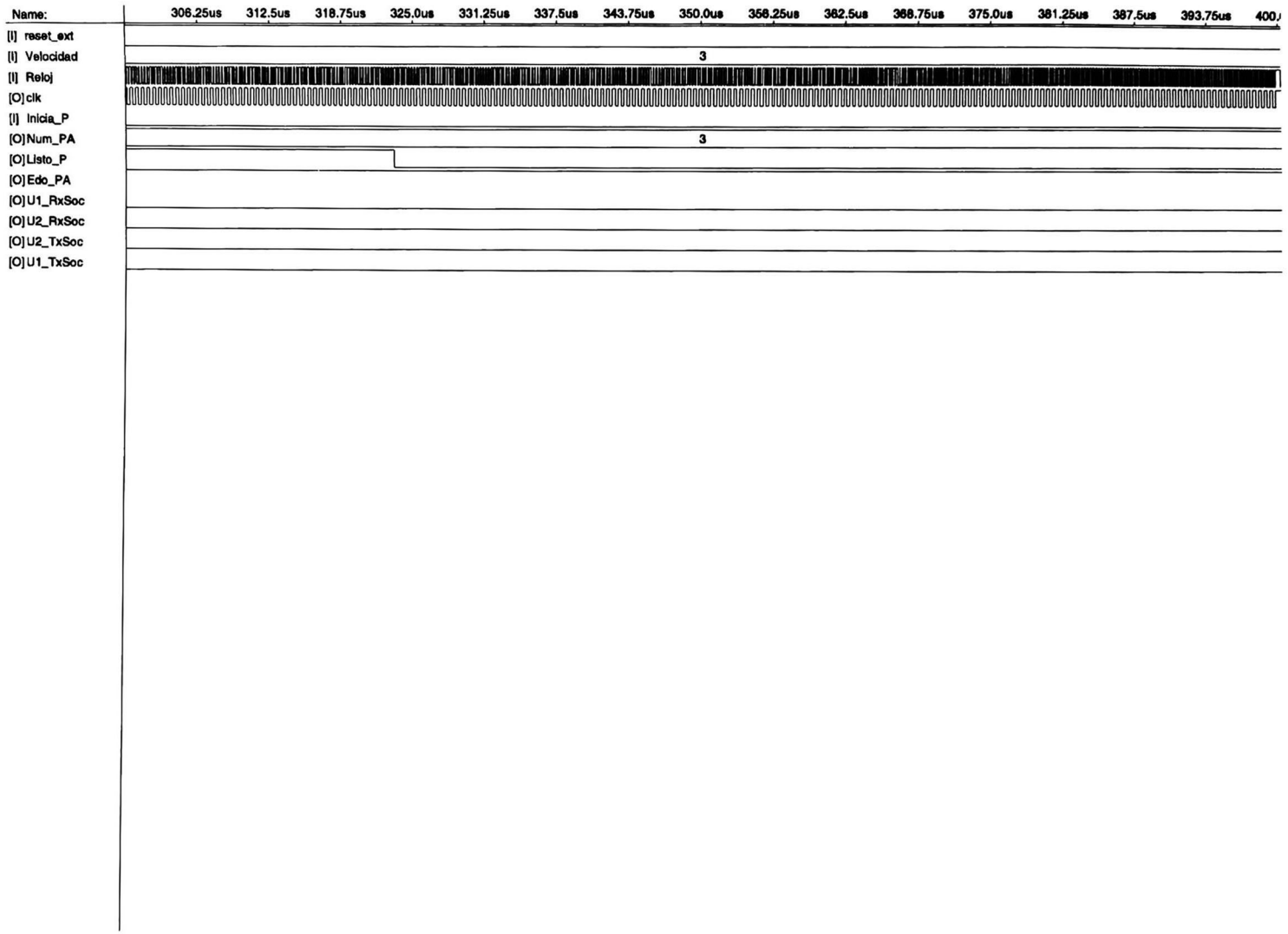
MAX+plus II 9.21 File: D:\TESIS2\UNION_33\UNION81.SCF Date: 08/11/2000 19:33:57 Page: 1
Name:
25.0us
50.0us
75.0us
100.0us
125.0us
150.0us
175.0us
200.0us
225.0us
250.0us
275.0us
300.0us
325.0us
350.0
[I] reset_ext
[I] Velocidad
[I] Reloj
[O] clk
[I] Inicia_P
[O] Num_PA
[O] Listo_P
[O] Edo_PA
[O] U1_RxSoc
[O] U2_RxSoc
[O] U2_TxSoc
[O] U1_TxSoc
3
2
0
3


MAX+plus II 9.21 File: D:\TESIS2\UNION_33\UNION81.SCF Date: 08/20/2000 13:15:14 Page: 1
Name:
6.25us
12.5us
18.75us
25.0us
31.25us
37.5us
43.75us
50.0us
56.25us
62.5us
68.75us
75.0us
81.25us
87.5us
93.75us
[I] reset_ext
[I] Velocidad
[I] Reloj
[O]clk
[I] Inicia_P
[O]Num_PA
[O]Listo_P
[O]Edo_PA
[O]U1_RxSoc
[O]U2_RxSoc
[O]U2_TxSoc
[O]U1_TxSoc
3
2
0

MAX+plus II 9.21 File: D:\TESIS2\UNION_33\UNION81.SCF Date: 08/20/2000 13:15:15 Page: 2
Name:
106.25us
112.5us
118.75us
125.0us
131.25us
137.5us
143.75us
150.0us
156.25us
162.5us
168.75us
175.0us
181.25us
187.5us
193.75us
[I] reset_ext
[I] Velocidad
[I] Reloj
[O] clk
[I] Inicia_P
[O] Num_PA
[O] Listo_P
[O] Edo_PA
[O] U1_RxSoc
[O] U2_RxSoc
[O] U2_TxSoc
[O] U1_TxSoc
3
2
3

MAX+plus II 9.21 File: D:\TESIS2\UNION_33\UNION81.SCF Date: 08/20/2000 13:15:15 Page: 3
Name:
206.25us
212.5us
218.75us
225.0us
231.25us
237.5us
243.75us
250.0us
256.25us
262.5us
268.75us
275.0us
281.25us
287.5us
293.75us
[I] reset_ext
[I] Velocidad
[I] Reloj
[O] clk
[I] Inicia_P
[O] Num_PA
[O] Listo_P
[O] Edo_PA
[O] U1_RxSoc
[O] U2_RxSoc
[O] U2_TxSoc
[O] U1_TxSoc
3
3

MAX+plus II 9.21 File: D:\TESIS2\UNION_33\UNION81.SCF Date: 08/20/2000 13:15:16 Page: 4



MAX+plus II 9.21 File: D:\TESIS2\UNION_33\UNION81.SCF Date: 08/20/2000 13:15:17 Page: 5
Name:
406.25us
412.5us
418.75us
425.0us
431.25us
437.5us
443.75us
450.0us
456.25us
462.5us
468.75us
475.0us
481.25us
487.5us
493.75us
500.
[I] reset_ext
[I] Velocidad
[I] Reloj
[O] clk
[I] Inicia_P
[O] Num_PA
[O] Listo_P
[O] Edo_PA
[O] U1_RxSoc
[O] U2_RxSoc
[O] U2_TxSoc
[O] U1_TxSoc
3
3

Apéndice D ATM

La idea en que se basa *ATM* (*Asynchronous Transfer Mode*, modo de transferencia asíncrona) consiste en transmitir toda la información en paquetes pequeños de tamaño fijo llamados celdas. Las celdas tienen una longitud de 53 bytes, de los cuales 5 son de encabezado y 48 de carga útil. Las redes *ATM* son orientadas a la conexión, esto es, para hacer una llamada primero se debe enviar un mensaje para establecer la conexión. Después, todas las celdas subsecuentes siguen la misma trayectoria al destino. La entrega de celdas no está garantizada, pero sí su orden. Las velocidades de las redes *ATM* son de 155 Mbps y 622 Mbps, aunque ya se empiezan a manejar velocidades del orden de gigabits.

El modelo de referencia *B-ISDN ATM*

La *ISDN* (*Integrated Services Digital Network*, red digital de servicios integrados) de banda ancha, con *ATM* tienen su propio modelo de referencia diferente del modelo *OSI* (*Open Systems Interconnection*, interconexión de sistemas abiertos) y también del modelo *TCP/IP* (*Transmission Control Protocol / Internet Protocol*); y consiste en tres capas: la capa física, la capa *ATM* y la capa de adaptación de *ATM*, más cualquier cosa que los usuarios quieran poner encima.

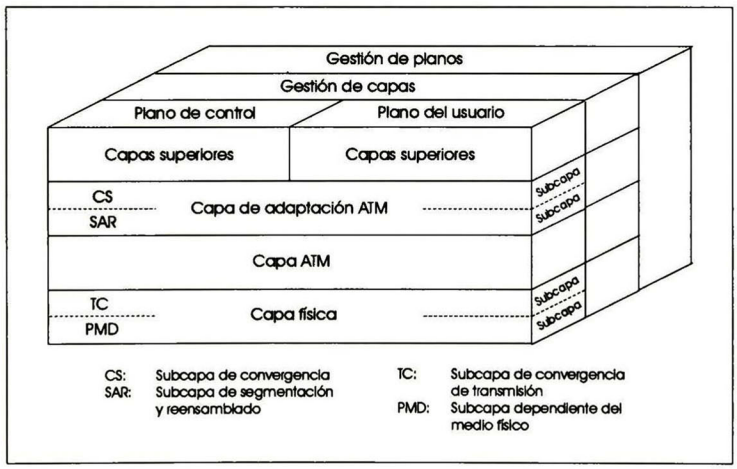


Figura D - 1 El modelo de referencia *B-ISDN ATM*.

La capa física tiene que ver con el medio físico: voltajes, temporización de bits y varias consideraciones más. *ATM* no prescribe un conjunto de reglas en particular, pero en cambio dice que las celdas *ATM* se pueden enviar por sí solas por un cable o fibra, o bien se pueden empacar dentro de la carga útil de otros sistemas portadores. En otras palabras, *ATM* se diseño para que fuera independiente del medio de transmisión.

La capa *ATM* tiene que ver con las celdas y su transporte; define la organización de las celdas y dice lo que significan los campos del encabezado. Esta capa también tiene que ver con el

establecimiento y la liberación de los circuitos virtuales y aquí es donde se localiza el control de la congestión.

Se ha definido una capa sobre la capa *ATM* que permite a los usuarios enviar paquetes mayores que una celda, debido a que la mayor parte de las aplicaciones no quieren trabajar de manera directa con celdas (aunque algunas puedan hacerlo). La interfaz *ATM* segmenta estos paquetes, transmite las celdas en forma individual y las reensambla en el otro extremo. Esta capa es la *AAL* (*ATM Adaptation Layer*, capa de adaptación de *ATM*).

A diferencia de los antiguos modelos de referencia bidimensionales, el modelo *ATM* se define en tres dimensiones, como se muestra en la figura D-1. El plano de usuario se encarga del transporte de los datos, el control de flujo, la corrección de errores y otras funciones de usuario. En contraste, el plano de control tiene que ver con la administración de la conexión. Las funciones de gestión de capas y planos se relacionan con la administración de recursos y la coordinación intercapas.

Las capas física y *AAL* se dividen, cada una, en dos subcapas. Una en el fondo que hace el trabajo y una subcapa de convergencia en la parte superior que proporciona la interfaz adecuada con la capa de arriba. En la Tabla D-1, se indican las funciones de las capas y subcapas.

Capa	Subcapa	Funcionalidad
AAL	CS	Proporciona la interfaz estándar (convergencia).
	SAR	Segmentación y reensamblado.
ATM		Control de flujo. Generación / extracción de encabezados de la celda. Gestión de circuitos / trayectorias virtuales. Multiplexión / Demultiplexión de celdas.
Física	TC	Desacoplamiento de la velocidad de envío de celdas. Generación y comprobación de la suma de verificación del encabezado. Generación de celdas. Empacado / desempacado de celdas de la envoltura que las encierra. Generación de marcos.
	PMD	Temporización de bits. Acceso físico a la red.

Tabla D - 1 Las capas y subcapas de *ATM* y sus funciones

La subcapa *PMD* (*Physical Medium Dependent*, dependiente del medio físico) establece la interfaz con el cable real; transfiere los bits y controla su temporización. Esta subcapa es diferente para diferentes portadoras y cables.

La otra subcapa de la capa física es la subcapa *TC* (*Transmission Convergence*, convergencia de transmisión). Cuando se transmiten las celdas, la subcapa *TC* las envía como una corriente de bits a la capa *PMD*, lo cual es fácil de hacer. En el otro extremo, la subcapa *TC* obtiene una

corriente entrante de puros bits de la subcapa *PMD*; su trabajo es convertir esta corriente de bits en una corriente de celdas para la capa *ATM*. La subcapa *TC* se encarga de todas las consideraciones que se relacionan con determinar donde empiezan y donde terminan las celdas en la corriente de bits.

Como se menciono anteriormente, la capa *ATM* maneja celdas, lo que incluye su generación y transporte. Aquí se localiza la mayor parte de los aspectos interesantes de *ATM*. La capa *ATM* no se divide en subcapas.

La capa *AAL* se divide en subcapa *SAR* (*Segmentation And Reassembly*, segmentación y rensamblado) y la *CS* (*Convergence Sublayer*, subcapa de convergencia). La subcapa inferior divide los paquetes en celdas en el lado de transmisión y los vuelve a armar de nuevo en el destino. La subcapa superior hace posible tener sistemas *ATM* que ofrezcan diferentes clases de servicios a diferentes aplicaciones (por ejemplo, la transferencia de archivos y el vídeo sobre pedido tienen diferentes necesidades en lo concerniente a manejo de errores, temporización, etcétera).

Transmisión de Celdas

Cuando un programa de aplicación produce un mensaje que debe ser enviado, ese mensaje viaja hacia abajo a través de la pila de protocolos *ATM*, agregándosele encabezados y apéndices, y sufriendo segmentación en celdas. En algún momento, las celdas llegan a la subcapa *TC* para su transmisión. El primer paso que hace esta subcapa es la suma de comprobación de encabezado. Cada celda contiene un encabezado de 5 bytes que consiste en 4 bytes de información del circuito virtual y de control seguidos de una suma de comprobación de un byte. Debido a que cubre sólo el encabezado, el campo de suma de comprobación de 8 bits se denomina *HEC* (*Header Error Control*, control de error de encabezado). El esquema *HEC* además de detectar errores, corrige todos los errores de un bit y detecta también muchos errores multibit.

Una vez que se ha generado el *HEC* y se ha introducido en el encabezado de la celda, esta está lista para transmitirse. Los medios de transmisión pertenecen a una de dos categorías: asíncronos y síncronos. Con un medio asíncrono se puede mandar una celda cuando esté lista para irse (no existen restricciones de tiempo).

En un medio síncrono las celdas deben transmitirse de acuerdo con un patrón de temporización predefinido. Si no hay una celda de datos disponible cuando se necesita, la subcapa *TC* debe inventar una. Éstas se llaman celdas de relleno.

Otro tipo de celda que no es de datos es la celda *OAM* (*Operation And Maintenance*, operación y mantenimiento). Las celdas *OAM* también son usadas por los conmutadores *ATM* para intercambiar información de control e información necesaria para mantener funcionando el sistema.

Del lado del receptor, las celdas de relleno se procesan en la subcapa *TC*, pero las *OAM* se entregan a la capa *ATM*. Las celdas *OAM* se distinguen de las de datos por tener ceros en los tres bytes de encabezado, algo no permitido en las celdas de datos. El cuarto byte describe la naturaleza de la celda *OAM*.

Aunque las compañías telefónicas evidentemente piensan usar *SONET* (*Synchronous Optical Network*, red óptica síncrona) como el sistema de transmisión subyacente para *ATM*, también se han definido correlaciones de *ATM* con los campos de carga útil de otros sistemas, y se está trabajando en nuevas. En particular, existen correlaciones para *T1-E1* (nuestro caso), *T3-E3*, *FDDI*, etc.

Recepción de celdas

En lo que respecta a la salida, la tarea de la subcapa *TC* es tomar una secuencia de celdas, agregarle un *HEC* a cada una, convertir el resultado en una corriente de bits, e igualar la corriente de bits con la velocidad de transmisión física subyacente, introduciendo celdas de relleno o celdas *OAM*. En la entrada, la subcapa *TC* hace exactamente lo inverso. Toma una corriente de bits de entrada, localiza los límites de las celdas, verifica los encabezados (descartando las celdas con encabezados no válidos), procesa las celdas *OAM* y pasa las celdas de datos a la capa *ATM*.

La parte más difícil es localizar los límites de las celdas en la corriente de bits de entrada. Para esta función, la subcapa *TC* utiliza un algoritmo de reconocimiento, que consiste de una máquina de estados finito en la que se manejan 3 estados: *Hunt* (busqueda), *Presynch* (pre-sincronizado) y *Synch* (sincronizado). En el estado *Hunt*, la subcapa *TC* recorre bits en los registros de desplazamiento uno a la vez buscando un *HEC* válido. Tan pronto encuentra uno, la máquina de estados finitos se conmuta al estado *Presynch*, lo que quiere decir que ha localizado tentativamente un límite de celda. Ahora recorre los siguientes 424 bits (53 bytes) sin examinarlos. Si su suposición respecto al límite de la celda fue correcta, el registro de desplazamiento ahora contendrá otro encabezado de celda válida, por lo que nuevamente ejecutará el algoritmo *HEC*. Si el *HEC* es incorrecto, la *TC* regresará al estado *Hunt* y continuará buscando bit por bit un encabezado cuyo *HEC* sea correcto.

Por otra parte, si el segundo *HEC* también es correcto, la subcapa *TC* podría haber encontrado algo, por lo que recorre otros 424 bits e intenta de nuevo. Continuará inspeccionando encabezados de esta manera hasta que ha encontrado δ encabezados correctos consecutivos, momento en el cual supone que está sincronizada y pasa al estado *Synch* para comenzar la operación normal.

Además de la resincronización tras perder la sincronía (o al arranque), la subcapa *TC* necesita una heurística para determinar cuándo ha perdido la sincronía, por ejemplo tras haber sido introducido o borrado un bit en la corriente de bits. Sería imprudente darse por vencido si solo un *HEC* fue incorrecto, ya que la mayoría de los errores son inversiones de bit, no inserciones ni eliminaciones. El camino más sensato aquí es descartar simplemente la celda con el encabezado equivocado y esperar que la siguiente sea correcta. Sin embargo, si α *HEC* seguidos están mal, la subcapa *TC* tiene que concluir que ha perdido la sincronía y debe regresar al estado *Hunt*.

La Capa ATM

El elemento básico de la capa *ATM* es el circuito virtual (llamado con frecuencia *canal virtual*). Un circuito virtual normalmente es una conexión de un origen a un destino, aunque también son permitidas conexiones multi-transmisión. Los circuitos virtuales son unidireccionales. La capa *ATM* proporciona la garantía que las celdas que son enviadas por un circuito virtual nunca llegarán fuera de orden. Es permitido descartar celdas si ocurren congestionamientos, pero en ninguna circunstancia se reordenarán las celdas que fueron enviadas por un solo circuito virtual. Un grupo de circuitos virtuales puede agruparse en lo que se llama una *trayectoria virtual.*

Formatos de Celda

En la capa *ATM* se distinguen dos interfaces: la *UNI* (*User-Network Interface*, interfaz usuario-red) y la *NNI (Network-Network Interface*, interfaz red-red). La primera define el límite entre un *host* y una red *ATM* (en muchos casos, entre el cliente y la portadora). La última se aplica a la línea entre dos conmutadores *ATM* (el término *ATM* para los enrutadores). En ambos casos, las celdas consisten de un encabezado de 5 bytes seguida de una carga útil de 48 bytes, pero los dos encabezados son ligeramente diferentes. En la figura D-2, se ilustran estos dos encabezados.

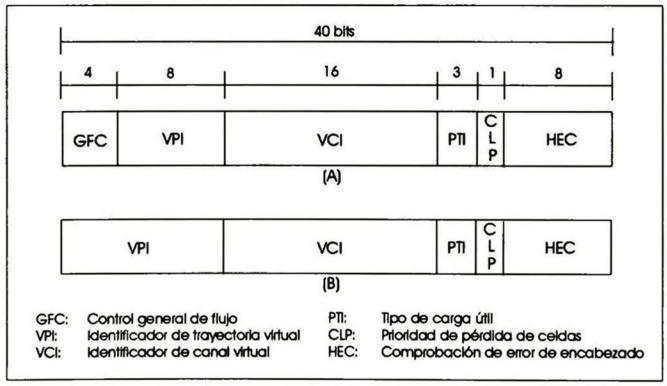


Figura D - 2 (A) Encabezado de la capa *ATM* en la *UNI*. (B) Encabezado de la capa *ATM* en la *NNI*

El campo *GFC* está presente solo en las celdas entre un *host* y la red; es sobrescrito por el primer conmutador al que llega, por lo que no tiene un significado de terminal a terminal, y no se entrega al destino.

El campo *VPI* es un entero que selecciona una trayectoria virtual en particular. De la misma manera, el campo *VCI* selecciona un circuito virtual en particular en la trayectoria virtual seleccionada. Dado que el campo *VPI* tiene 8 bits (en la *UNI*) y el campo *VCI* tiene 16 bits, en

teoría un *host* puede tener hasta 256 haces de trayectorias virtuales, conteniendo cada uno hasta 65,536 circuitos virtuales. En realidad, la cantidad disponible de ambos es un poco menor, pues algunos *VCI* se reservan para funciones de control, como el establecimiento de circuitos virtuales.

El campo *PTI* define el tipo de carga útil que contiene la celda, de acuerdo con los valores dados en la tabla D-2. Aquí los tipos de celda son proporcionados por el usuario, pero la información de congestionamiento es proporcionada por la red.

Tipo de carga	Significado
000	Celda de datos de usuario, sin congestionamientos, celda tipo 0
001	Celda de datos de usuario, sin congestionamientos, celda tipo 1
010	Celda de datos de usuario, hubo congestionamiento, celda tipo 0
011	Celda de datos de usuario, hubo congestionamiento, celda tipo 1
100	Información de mantenimiento entre conmutadores adyacentes
101	Información de mantenimiento entre conmutadores de origen y destino
110	Celda de administración de recursos
111	Reservado para función futura

Tabla D - 2 Tipo de carga útil descrita por el campo *PTI*

El bit *CLP* puede ser establecido por un *host* para distinguir entre el tráfico de alta prioridad y el de baja prioridad. Si ocurre un congestionamiento y deben descartarse celdas, los conmutadores primero intentan descartar las que tienen el *CLP* establecido en 1 antes de descartar cualquiera que lo tenga establecido en 0.

Por último, el campo *HEC* es una suma de comprobación del encabezado; no verifica la carga útil.

A continuación del encabezado vienen 48 bytes de carga útil. Sin embargo, no todos los 48 bytes están disponibles para el usuario, pues algunos de los protocolos *AAL* ponen sus encabezados y sus terminaciones dentro de estos 48 bytes.

El formato *NNI* es igual al formato *UNI*, excepto que el campo *GFC* no está presente y esos 4 bits se usan para hacer que el campo *VPI* sea de 12 bits en lugar de 8.

Categorías de servicios

Las categorías de servicios con las que cuenta *ATM* son:

La clase *CBR* (*Constant Bit Rate*, tasa de bits constante) pretende simular un alambre de cobre o una fibra óptica (sólo que a un costo mucho mayor). Los bits se ponen en un extremo y salen por el otro. No hay comprobación de errores, control de flujo ni ningún otro proceso. No obstante, esta clase es esencial para hacer una transición suave entre el sistema telefónico

actual y los sistemas *B-ISDN*, ya que los canales *PCM* (*Pulse Code Modulation*, modulación por pulsos codificados) de grado de voz, los circuitos *T1-E1* y la mayor parte del resto del sistema telefónico usa la transmisión síncrona de tasa constante. Con la clase *CBR* puede transportarse directamente todo este tráfico a través de un sistema *ATM*. La *CBR* también es adecuada para todas las demás cadenas interactivas (es decir, en tiempo real) de audio y vídeo.

La siguiente clase, la *VBR* (*Variable Bit Rate*, tasa variable de bits), se divide en dos subclases, la de tiempo real (*RT-VBR*) y la de tiempo no real (*NRT-VBR*), respectivamente. La RT-VBR es para servicios que tienen tasas de bits variables en combinación con requisitos muy estrictos de tiempo real, como el vídeo comprimido interactivo (por ejemplo, videoconferencias). La otra subclase *VBR* es para tráfico en el que la entrega a tiempo es importante pero la aplicación puede tolerar una cierta cantidad de fluctuación.

La categoría de servicio *ABR* (*Available Bit Rate*, tasa de datos disponible) se diseño para trabajar en ráfagas cuya gama de ancho de banda se conoce aproximadamente. El uso del servicio *ABR* evita tener que comprometerse con un ancho de banda fijo. La *AVR* es la única categoría de servicio en la que la red proporciona tasa de retroalimentación al transmisor, solicitándole que disminuya la velocidad al ocurrir congestionamientos.

Por último esta la categoría *UBR* (*Unspecified Bit Rate*, tasa de datos no especificada), que no hace promesas y no realimenta información sobre los congestionamientos. Se aceptan todas las celdas *UBR* y, si sobra capacidad, también se entregan. Si ocurren congestionamientos, se descartan las celdas *UBR* sin realimentación al transmisor y sin esperar que el transmisor reduzca su velocidad.

La capa de adaptación de ATM (AAL)

La meta de la capa *AAL* (*ATM Adaptation Layer*, capa de adaptación de *ATM*) es proporcionar servicios útiles a programas de aplicación y protegerlos de la mecánica de dividir datos en celdas en el origen y reorganizarlos en el destino. Para el manejo de las categorías de servicio que maneja *ATM* (*ABR*, *CBR*, *NRT-VBR*, *RT-VBR* y *UBR*), *ATM* tiene definidos 4 protocolos: *AAL1*, *AAL2*, *AAL3/4* y *AAL5*.

Estructura de la capa de adaptación de ATM (AAL)

La capa de adaptación de *ATM* se divide en dos partes principales, una de las cuales se subdivide aún más. La parte superior de la capa de adaptación de *ATM* se llama subcapa de convergencia. Su tarea es establecer la interfaz con la aplicación. Esta subcapa consiste de una subparte común a todas las aplicaciones (para un protocolo *AAL* dado) y una subparte específica para cada aplicación. Las funciones de cada una de estas partes dependen del protocolo pero pueden incluir enmarcado de mensajes y detección de errores.

La parte más baja de la *AAL* se llama subcapa *SAR* (*Segmentation And Reassembly*, segmentación y reensamblado). Esta subcapa puede agregar encabezados y apéndices a las

unidades de datos entregadas a ella por la subcapa de convergencia para formar cargas útiles de celdas. Estas cargas se entregan a la capa *ATM* para su transmisión. En el destino, la subcapa *SAR* básicamente se ocupa de las celdas y la subcapa de convergencia se ocupa de los mensajes. La subcapa *SAR* también tiene algunas funciones adicionales para algunas clases de servicios (pero no todas). En particular, a veces maneja la detección de errores y la multiplexión. La subcapa *SAR* está presente en todas las clases de servicio, aunque el trabajo que realiza depende del protocolo específico.

AAL1

AAL1 es el protocolo usado para transmitir tráfico orientado a conexiones de tiempo real y con tasa de bits constante, como audio o vídeo sin compresión. Los bits son alimentados por la aplicación a una velocidad constante y deben entregarse en el otro lado a la misma velocidad constante, con retardo, fluctuación y carga extra mínimos. La entrada es una corriente de bits, sin límites de mensaje. Para este tráfico no se usan los protocolos de detección de errores, porque los retardos que generan las terminaciones de temporización y las retransmisiones no son aceptables. Sin embargo, las celdas faltantes se informan a la aplicación, que entonces, si lo desea, puede tomar sus propias medidas para recuperarlas.

AAL1 tiene una subcapa de convergencia y una subcapa *SAR*. La primera detecta celdas perdidas y mal introducidas. Esta subcapa también amortigua el tráfico de entrada para proporcionar la entrega de celdas a una tasa constante. Por último, la subcapa de convergencia divide los mensajes o la corriente de entrada en unidades de 46 o 47 bytes que se entregan a la subcapa *SAR* para su transmisión. En el otro extremo se extraen estas unidades y se reconstruye la entrada original. La subcapa de convergencia de *AAL1* no tiene ningún encabezado propio de protocolo.

En contraste la subcapa *SAR* de *AAL1* sí tiene un protocolo. Los formatos de sus celdas se muestran en la figura D-3. Ambos formatos comienzan con un encabezado de un byte que contiene un número de secuencia de celda de 3 bits, *SN*, para detectar celdas perdidas o mal introducidas. A este campo le sigue un número de protección de secuencia (es decir, suma de comprobación) de 3 bits, el *SNP*, basado en el número de secuencia, para permitir la corrección de errores individuales y la detección de errores dobles en el campo de secuencia. Un bit de paridad par que cubre el byte de cabecera, reduciendo aun más la posibilidad de un número de secuencia equivocado.

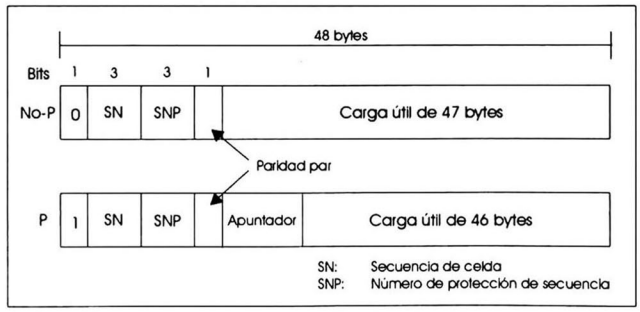


Figura D - 3 Formato de celda del protocolo *AAL1*

Las celdas *P* se usan cuando deben preservarse los límites de los mensajes. El campo de *apuntador* sirve para indicar el desfasamiento del comienzo del siguiente mensaje. Solo las celdas con un número de secuencia par pueden ser celdas *P*, por lo que el apuntador está en el intervalo de 0 a 92, para que apunte dentro de la carga útil de su propia celda o de la que sigue. Este esquema permite que los mensajes tengan una cantidad arbitraria de bytes, por lo que pueden enviarse mensajes continuamente y no necesitan alinearse con los límites de la celda.

El bit de orden mayor del campo *apuntador* está reservado para uso futuro. El bit inicial del encabezado de todas las celdas de número impar forma una corriente de datos usada para la sincronización del reloj.

AAL2

En el audio o vídeo comprimido la tasa puede variar considerablemente con el tiempo. Para estos fines se diseño el *AAL2*. Desafortunadamente este protocolo no fue estandarizado adecuadamente, de tal forma que no es utilizado.

AAL3/4

El *AAL3/4* puede operar de dos maneras: En el modo de mensajes, cada llamada de la aplicación al *AAL3/4* inyecta un mensaje en la red. El mensaje se entrega como tal, es decir, se conservan los límites del mensaje. En el modo de corrientes no se conservan los límites. En cada modo hay disponible transporte confiable y no confiable (es decir, sin garantía).

Una característica del *AAL3/4* no presente en los otros protocolos es la multiplexión. Este aspecto del *AAL3/4* permite que viajen por el mismo circuito virtual múltiples sesiones de un solo *host* y que se separen en el destino.

A diferencia del *AAL1* y el *AAL2*, el *AAL3/4* tiene tanto un protocolo de subcapa de convergencia como uno de subcapa *SAR*.

AAL5

El *AAL5* ofrece varios tipos de servicio a sus aplicaciones. Una posibilidad es el servicio confiable (es decir, entrega garantizada con control de flujo para evitar desbordamientos). Otra posibilidad es el servicio no confiable (es decir, sin entrega garantizada), con opción de descartar o pasar a la aplicación (reportando como malas) las células con errores de suma de comprobación. Se apoyan tanto la unitransmisión como la multitransmisión, pero la multitransmisión no ofrece entrega garantizada.

CENTRO DE INVESTIGACION 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: "Interfaz entre enlaces E1/T1 y una matriz de conmutación ATM de alta velocidad" que presenta el Ing. Guillermo Alejandro López López el día 24 de noviembre del 2000

Dr. Manuel Eduardo Guzmán Rentería
Investigador Cinvestav 3 A
CINVESTAV DEL IPN
Guadalajara

Dr. Deni Librado Torres Román
Investigador Cinvestav 2 C
CINVESTAV DEL IPN
Guadalajara

M.C. Jesús Palomino Echartea
Gerente de Ingeniería
Intel Tecnología de México, S.A. de C.V.
Guadalajara, Jal.

CINVESTAV
BIBLIOTECA CENTRAL
SSIT000003867