A Design of a PBX to ATM User to Network Interface Over DS1/T1 Lines

by

Edmund G. Chen

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements
for the degrees of
BACHELOR OF SCIENCE
and
MASTER OF SCIENCE
in Electrical Engineering at the
Massachusetts Institute of Technology
April 13, 1995

Author _____

Department of Electrical Engineering and Computer Science, April 13, 1995

Certified by _____

Steven Burns, Academic Thesis Supervisor

Certified by _____

Ron Tanizawa, Cooperating Company Supervisor

Accepted by _____

F. R. Morgenthaler, Chairman, Department Committee on Graduate Students

Acknowledgments

I would like to take this opportunity to thank those people and organizations which helped get me to this stage.

Thanks to Siemens Rolm Communications Inc. for taking the chance on me my sophomore year and for providing me with a great thesis topic in the area of ATM networking. Thanks to Jay Thomas, Ron Tanizawa, Bruce Huang, David Weiss, Esmail Kalami and everyone else at Rolm for providing me with the support and opportunity to work on this project.

Thanks to Professor Burns for allocating the time in his schedule to be my thesis advisor when others were "too busy."

Thanks to my parents and brothers, who provided me with all the support and understanding I could possibly ask for, which allowed me to stay focused on my true goals.

Most of all, I would like to thank Michelle Chin and Mark Hansen. Two great friends who have always been there to lend an ear or anything else and made the whole thing a bit more interesting.

Edmund G. Chen

May 5, 1995

A Design of a PBX to ATM User to Network Interface Over DS1/T1 Lines

by

Edmund G. Chen

Submitted to the Department of Electrical Engineering and Computer Science
on May 5, 1995 in partial fulfillment of the requirements for the
Degree of Master of Science in Electrical Engineering.

ABSTRACT

The asynchronous transfer mode (ATM) protocol has been heralded as the future of networking. Its fixed length data cells, which allows for high speed switching, gives ATM the characteristics of possibly being the unifying data protocol for all formats of data. An ATM network should theoretically be able to handle all types of data ranging from delay sensitive constant bit rate (CBR) data, such as voice and video, to delay insensitive variable bit rate (VBR) data such as file transfers.

While the current industry drive is to develop ATM for high speed line rates, such as T3/DS3 or OC-3, and delay insensitive data, the focus of this thesis is in the less explored area of CBR data. The thesis involves the design of an ATM user-to-network interface (UNI) over DS1/T1 lines for a private branch exchange (PBX) switch.

The DS1 line rate was chosen because the development of a global ATM network also hinges on the ability to make use of existing physical networks. Currently there is billions of dollars invested in DS1/T1 lines and switching equipment. To make use of this existing network, a novel design for an interface to a PBX for the purpose of transferring CBR data over DS1/T1 lines is proposed. The design will be for the transmitter and receiver of an UNI which provides the data segmentation and reassembly (SAR) between time division multiplexed data and the ATM protocol.

The goal of the thesis will be the development and design of a PBX to ATM UNI over DS1/T1 lines. The design will be simulated for design and functional verification. If there is additional time, a prototype board will be built.

Thesis Supervisor:	Dr. Steven Burns Technical Director of Biomedical Engineering Center
Company Supervisor:	Ron Tanizawa Manager Digital Lines and Trunks Department

Table Of Contents

List Of Figures

List Of Tables

Chapter 1 Introduction

1.1 Thesis Overview

The goal of this thesis will be to develop a PBX to ATM UNI over DS1/T1 lines. This design with be for the Transmitter and Receiver interfaces necessary to perform the segmentation and reassembly (SAR) of data for the interface between PBX CBR data and the ATM protocol. The ATM board will interface to a Siemens Rolm product, the Trunk Module Digital 24 channel (TMD24) card. The TMD24 card will provide the signaling and data interface to the PBX switch. Section 3 provides more detail on the TMD24 interface.

The ATM Transmitter section will load in the CBR data contained in the Pulse Code Modulated (PCM) data highway, which is received from the TMD24. The Transmitter will then reformat the PCM highway data to satisfy the ATM over DS1 requirements and transmit the ATM cells. The Receiver will load in ATM cells and reformat the ATM payloads to satisfy the PCM highway timing requirements of the PCM highways on the TMD24 board.

This project will not involve the system signaling to either the PBX switch or the ATM network. The signaling required to establish a call or maintain the ATM network connection will be left for higher layer management. The project will focus on the SAR data management required for the transfer of CBR data over DS1 lines.

A large part of this project is to prove ATM technology for the transfer of CBR data, but other project goals are to:

1. Determine requirements for developing an ATM interface for PCM speech highways.
2. Define one architecture that can interact with multiple line speeds.
3. Investigate crucial ATM issues on a small scale project, such as:
   - How ATM cells fit within the T1 span.
   - Effects of different frequencies and amplitudes of delay, caused by late arrival of ATM cells.

The extent of the investigation of the effects of delay will depend on the available time remaining in the assignment.

1.2 Overall System

With the ATM over DS1 UNI being developed, a point-to-point network as shown in Figure 1, "Overall System Setup," could be installed. The interface to an ATM network is ideal so that each connection contained within an ATM cell can be routed independently from all other connections. Other possible system configurations such as connections through a central office (CO) over leased T1 lines, would

essentially establish a permanent connection requiring all connections to have the same source and destination.

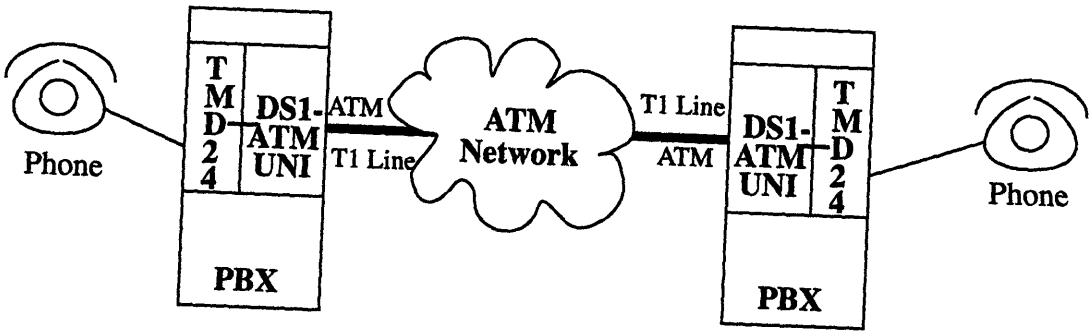


FIGURE 1. Overall System Setup

Chapter 2 Background

2.1 Time Division Multiplexing

Time division multiplexing (TDM) is the telecommunication standard for carrying CBR data. The TDM format is based on the concept that each connection or channel occupies a specific time slot with reference to global system clock. For a DS1/T1 line, twenty-four channels and a framing bit are grouped together into frames, see Figure 2, "DS1 TDM frame." Each channel contains an eight bit value obtained from sampling CBR data at 8KHz. The framing bit is used primarily for synchronization.

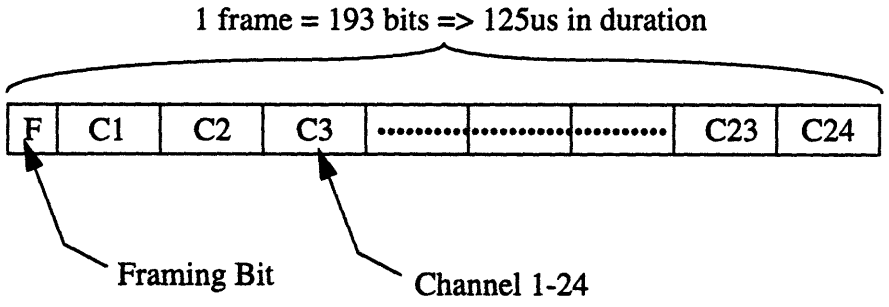


FIGURE 2. DS1 TDM frame

The 24 channels plus 1 framing bit, a total of 193 bits, must fit within a 125μs time frame in order to satisfy the 8KHz sampling rate. This resulting transmission rate of 1.544Mbits-per-second (Mbps) and the 193 bit data format is the standard known as DS1/T1.

2.2 Asynchronous Transfer Mode (ATM)[1]

The ATM cell is composed of 53 bytes, of which the first 5 bytes are reserved for addressing, and are referred to as header bytes. The usage of the remaining 48 bytes vary depending on the application, but mainly are used for carrying data, either CBR or VBR, and are referred to as the payload, see Section 2.3 for more detail. Figure 3, "Basic ATM Cell Structure," shows how the 53 bytes are partitioned into header and payload.

1. For more information about ATM, see the bibliography or any of the many published texts.

2.2.1 Header

The exact header bits differ depending on whether the ATM is being generated from a User to Network Interface (UNI) or a Network to Network Interface (NNI). For the UNI, there are six fields within the five byte header. The order of occurrence in the header, field name, abbreviation and bit size are listed in Table 1, "UNI Header Fields."

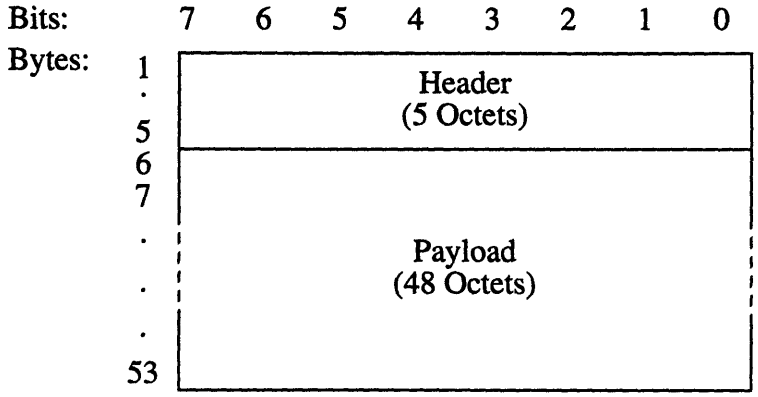


FIGURE 3. Basic ATM Cell Structure

The NNI header fields are essentially the same as the UNI header fields except for NNI header fields, the VPI is extended to 12 bits, replacing the GFC field, see Table 1, "UNI Header Fields." The header structure for both UNI and NNI ATM cells is shown in Figure 4, "Header Structure (UNI/NNI),"

TABLE 1. UNI Header Fields

Field Name	Abbreviation	Size
1.) Generic Flow Control	GFC	4 bits
2.) Virtual Path Identifier	VPI	8 bits
3.) Virtual Channel Identifier	VCI	12 bits
4.) Payload Type	PT	3 bits
5.) Cell Loss Priority	CLP	1 bit
6.) Header Error Control	HEC	8 bits

TABLE 2. NNI Header Fields

Field Name	Abbreviation	Size
1.) Virtual Path Identifier	VPI	12 bits
2.) Virtual Channel Identifier	VCI	12 bits
3.) Payload Type	PT	3 bits

TABLE 2. NNI Header Fields

Field Name	Abbreviation	Size
4.) Cell Loss Priority	CLP	1 bit
5.) Header Error Control	HEC	8 bits

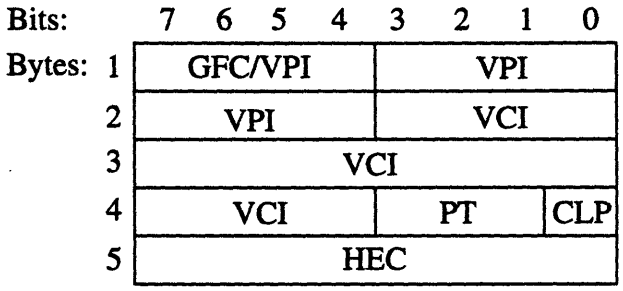


FIGURE 4. Header Structure (UNI/NNI)

The main fields of concern are VPI, VCI, and HEC. The VPI and VCI fields are used for addressing and can be considered as a single address field. The HEC field is actually a CRC-32 value, generated over the first four bytes of the header and is used for cell delineation and error correction of the header.

2.3 ATM Adaptation Layer (AAL)

ATM adaptation layers (AALs) have been defined for the mapping of data between the higher layer protocol and the ATM payload. For the transfer of CBR data, AAL1 is used. AAL1 requires that first byte of the 48 byte payload be used to carry a sequence number (SN) field, which can be used for higher layer error correction such as out of order cell correction. The presence of the SN byte reduces the available ATM cell payload to 47 bytes.

Chapter 3 TMD24 Interface

The short time span over which this project must be completed, requires the need for an interface with the existing TMD24 card. The TMD24 card is normally used to provide the interface between the PBX and a DS1/T1 line. The ATM board interface with the TMD24 board is required to establish and maintain connections and provide system signaling with the switch. This alleviates the need for the ATM board to handle the system level signaling between the ATM board and the PBX switch required to setup and maintain a connection. The TMD24 board will also be used to provide the actual physical interface with the DS1/T1 line.

3.1 Software Impact

Because of the lack of software support, it is important that the design have no F/W, L/W or S/W effects on the TMD24 card or the switch. This requires that the ATM board operates transparently to the TMD24 card. This way, there will be no changes to existing code on the TMD24 or switch. However, in an actual product, there will be modifications and additions, ATM related, to existing L/W, F/W and perhaps S/W.

3.2 Configuration Requirements

The TMD24 must be configured to operate in Extended Superframe Format (ESF) with MOS signaling, see Section 3.4. Because of bandwidth limitations, three time slots must be masked off, see Section 4.1. For the project implementation, the masked off time slots are arbitrarily chosen to be time slots 1, 2, and 3.[2, 3]

3.3 Interface to the TMD24

The ATM UNI project will focus on the SAR data management for the mapping of the PCM speech highways to the ATM protocol requirements. The ATM board will interface between the Memory Time Switch (MTS) chip and the Frame Alignment Unit (FAU) chip on the TMD24 board, see Figure 5, "TMD24 Interface."

---

2. Any three B channels can be masked off, leaving 20 B channels and the D channel to be supported in ATM over DS1. For now, all 21 supported channels will be assumed to be in use. B channels indicate time slots carrying data information and D channels indicate time slots carrying signaling information.

3. Time slots are the physical numbering implied from the system timing. The Channel Number is a logical number applied to each time slot.

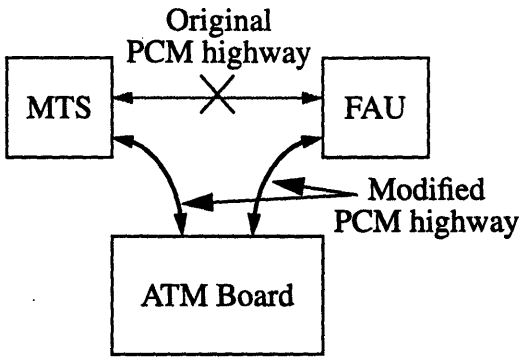


FIGURE 5. TMD24 Interface

The signals required for this point of interface are listed in Table 3, "Input Signals Required By ATM Board," and Table 4, "Output Signals Produced By ATM Board." This point of interface removes the requirement of having to manage the insertion and removal of the framing bit (FB) and allows the ATM board to deal strictly with the PCM highway data. One draw back of this point of interface is the fact that we must deal with 32 time slots, 8 of which are inactive[4], at a data rate of 2.048Mbps instead of 24 time slots at a data rate of 1.544Mpbs.

TABLE 3. Input Signals Required By ATM Board

Signal	Input From:
FMB	backplane
CLKA	backplane
FAU Data In	output of TMD24's MTS
FAU Data Out	output of TMD24's FAU
Power	backplane
Ground	backplane

TABLE 4. Output Signals Produced By ATM Board

Signal	Output To:
Modified FAU Data In	input to TMD24's FAU24
Modified FAU Data Out	input to TMD24's MTS

In order for the ATM board to be transparent to the TMD24 card, the serial data timing must be identical to those required by the TMD24 card. The FMB signal is used to indicate the start of time slot 0 and

4. Dead time slots are time slots 0, 4, 8, 12, 16, 20, 24, an 28, which are filled with an all 1's value.

only occurs once every 64 time slots, see Figure 6, "FMB and Time Slot Alignment Timing." A detailed timing interface is shown in Figure 7, "TMD24 Interface Timing."

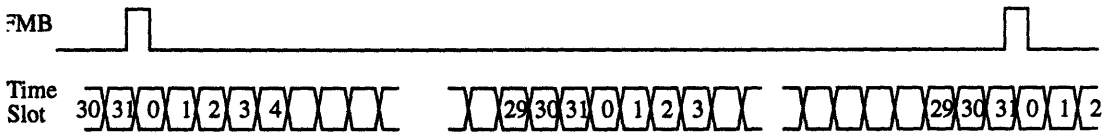


FIGURE 6. FMB and Time Slot Alignment Timing

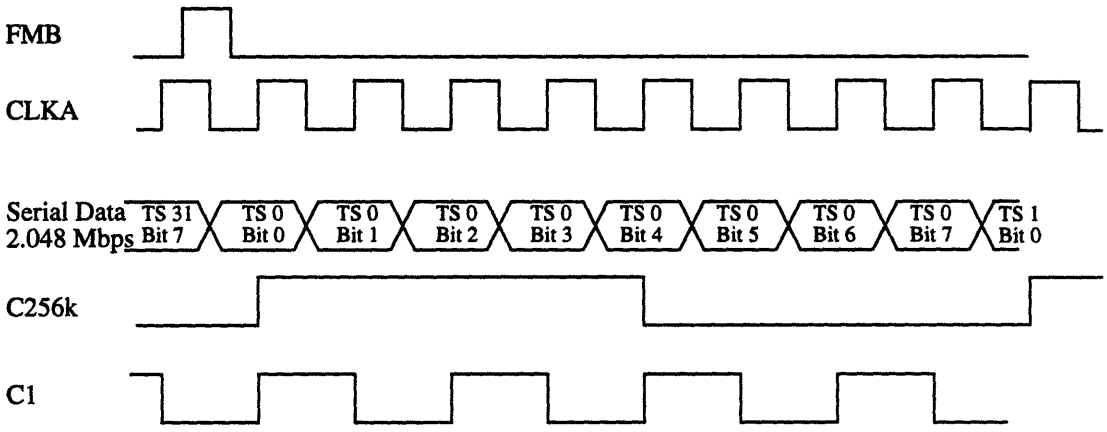


FIGURE 7. TMD24 Interface Timing

CLKA, C256k, and C1 are 2.048MHz, 256KHz and 1.024MHz clocks respectively. Both the FMB and the CLKA signals are generated by the PBX switch. The 256k and the C1 clocks are generated internal to the ATM board.

3.4 TMD24 Signaling Requirements

With the ATM board interfaces between the MTS and FAU chips, the signaling modes supported by the TMD24 are limited, see Section 3.3.

The Channel Associated Signaling (CAS) and Bit Oriented Signaling (BOS) signaling modes will not be supported by the ATM board. The problem occurs because both CAS and BOS signaling modes are inserted by the FAU. CAS is a "robbed bit"[5] signaling mode, which causes problems when bits of the ATM header are replaced by signaling bits. BOS signaling causes a similar problem.[6] Another factor is

5. "Robbed bit" signaling is a signaling mode where every 6 frames, the least significant bit of each channel (1-24) is replaced with a signaling bit. Therefore "robbed bit" signaling will cause data to be distorted. For example, for standard framing, robbed bit occurs in frames 6 and 12, and for ESF, robbed bit occurs in frames 6, 12, 18, and 24.

6. BOS signaling uses the same type of signaling as CAS, except all signaling bits are multiplexed, typically, into channel 24. The problem with BOS is that it is inserted by the FAU, which would also distort our modified data that the ATM board is sending to the FAU.

that CAS and BOS are old signaling modes and it does not make sense to try to support them with modern ATM equipment.

The only signaling mode that will be supported by the ATM board is Message Oriented Signaling (MOS). MOS can be supported since it is inserted, normally into time slot 32, by the MTS.[7]

7. If the MOS signaling channel is assigned to a different time slot, refer to Section 3.2 for restrictions.

Chapter 4 Principles of Operation

4.1 Bandwidth

The bit rate for DS1/T1 is 1.544Mbits/sec. The actual PCM highway data transfer rate over DS1/T1 is 1.536Mbits/sec. This is because only 192 bits, of the 193 bits making up a T1 frame, are available for PCM data.

Since ATM over DS1/T1 must also carry the ATM header bytes, as shown in Figure 8, "ATM Cell Mapping for DS1/T1," then it is not possible to support all 24 T1 channels. Specifically, the ATM protocol requires an additional 5 byte header, plus an extra byte for AAL1, the SN byte. This leaves a 47 byte payload and thus, only 21 time slots can be supported for ATM over DS1/T1.[8] This requires 3 time slots to be masked off as stated in Section 3.2.

8.

$$\left(\frac{53bytes}{ATMcell}\right)\left(\frac{8bits}{byte}\right) = \frac{424bits}{ATMcell} \quad \text{(Number of bits in an ATM cell)} \qquad \text{(EQ 1)}$$

$$\left(\frac{1.536Mbits}{s}\right)\left(\frac{1ATMcell}{424bits}\right) = \frac{3622.6415ATMcells}{s} \quad \text{(ATM cells transmitted per second)} \qquad \text{(EQ 2)}$$

$$\left(\frac{47bytepayload}{ATMcell}\right)\left(\frac{8bits}{byte}\right) = \frac{376bitpayload}{ATMcell} \quad \text{(Available bits per ATM payload)} \qquad \text{(EQ 3)}$$

$$\left(\frac{3622.6415ATMcells}{s}\right)\left(\frac{376bitpayload}{ATMcell}\right) = \frac{1.3621132Mbitpayload}{s} \quad \text{(Availablebitstransmittedpersecond)} \qquad \text{(EQ 4)}$$

$$\left(\frac{1.3621132Mbitpayload}{s}\right)/\left(\frac{1.536Mbit}{s}\right) = 0.88679 \quad \text{(Percent of bit rate supportable)} \qquad \text{(EQ 5)}$$

$$(0.88679)\ (24channels) = 21.283channels \Rightarrow 21channels \quad \text{(Number of channels supported)} \qquad \text{(EQ 6)}$$

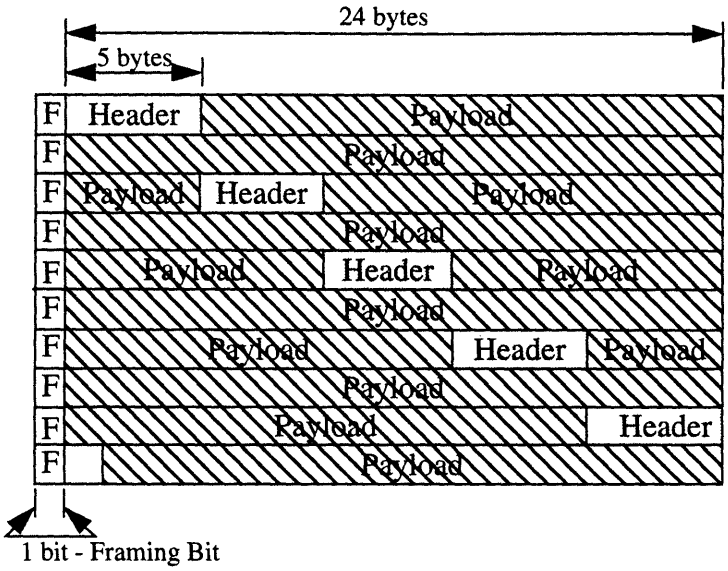


FIGURE 8. ATM Cell Mapping for DS1/T1

Chapter 5 Idle Cell Insertion

Because ATM over DS1 does not fit exactly, there is a need to insert idle ATM cells in order to match incoming PCM highways, outputted ATM cells and T1 rates, see Section 4.1.

5.1 Data Rate Interface

The mismatch of the input and output data rates is evident from the timing calculations. The time to accumulate 47 bytes, from the PCM highway, generates a 5.875ms delay.[9] Therefore, after 5.875ms, there are 21 ATM payloads[10], of 47 bytes each, ready to be transmitted. On the Receiver side, it takes 276.04μs[11] to receive 1 ATM cell and 5.769ms[12] to receiver 21 ATM cells.

Since it only takes 5.769ms to receive 21 ATM cells, but requires 5.875ms to accumulate the data for the 21 ATM cells, then there is obviously a data rate mismatch. Therefore, at specific times, the Transmitter will be required to transmit idle ATM cells in order to match the input and output data rates.

5.2 Repeatable Pattern

The easiest way of determining when idle ATM cells must be inserted is to determine the smallest repeatable pattern of valid and idle ATM cells for which the input and output data rates match exactly. In order to determine this pattern, two equations were used. Equation 1 requires that the ending of the

---

9.

$$\left(\frac{193bits}{T1frame}\right)\left(\frac{1}{1.544Mbps}\right) = \frac{125\mu s}{T1frame}$$

$$(47T1frames)\left(\frac{125\mu s}{T1frame}\right) = 5.875ms$$

10. This time is assuming that all 21 supported channels are in use.

11.

$$\frac{\left(\frac{125\mu s}{T1frame}\right)\left(\frac{53bytes}{ATMcell}\right)}{\left(\frac{24bytes}{T1frame}\right)} = \frac{276.04167\mu s}{ATMcell}$$

12.

$$\left(\frac{276.04167\mu s}{ATMcell}\right)(21ATMcells) = 5.76875ms$$

total number of valid ATM cells and idle ATM cells corresponds exactly with the end of a T1 frame.[13] Equation 2 requires that the number of inputted and outputted T1 frames are equal.[14]

$$21n + i = 24a \qquad \text{(EQ 1)}$$

$$(((24a)\,53)/24) = \left(\frac{47 T1 frames}{ATMcell}\right)(n) \qquad \text{(EQ 2)}$$

n = # of groups of 21 ATM cells
i = # of Idle ATM cells
a = # of groups of 24 ATM cells

Solving Equation 1 and Equation 2 for the minimal values of a, n and i produces the values $a = 47$, $n = 53$ and $i = 15$. This gives the minimal number of ATM cells, valid and idle, such that the data rates, for the 21 supported channels, of the PCM highway, ATM cells and T1 rate are satisfied.

Therefore, the minimal repeatable pattern contains (53 * 21) + 15 = 1128 ATM cells. The actual idle ATM cell insertions should be made in order to keep the Receiver buffer size to a minimal. This is achieved by making the most uniform delay possible. It turns out that the delay and buffer size are minimized by having 3 groups of 75 and 12 groups of 74, with idle cells inserted between each group. This grouping gives a total of 15 idle cells, and a total cell count of 1128, see Figure 9, "Idle Cell Insertions for Smallest Repeating Pattern."

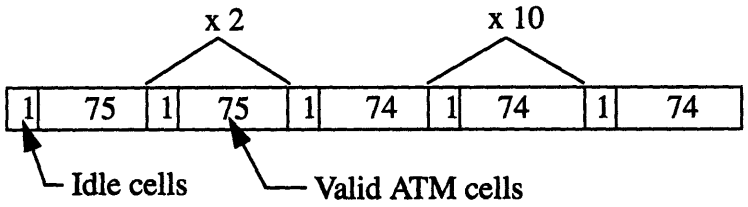


FIGURE 9. Idle Cell Insertions for Smallest Repeating Pattern

13. Every 24 ATM cells fits exactly into 53 T1 frames.

14. The right side of the Equation 2, (47 * n), is the number of T1 frames that have been accumulated. The left side of the Equation 2, (24 * a), is the number of ATM cells in the pattern. Then (24 * a * 53) is the number of bytes, and ((24 * a * 53)/24), is the number of T1 frames needed to carry all the ATM cells in the pattern.

Chapter 6 VPI/VCI Values

6.1 ATM VPI/VCI Addressing

Currently the design will implement VPI/VCI addressing through the use of an EPROM look-up table. The time slot number on the Transmitter side will be used to select the 5 byte header. On the Receiver side, the 5 byte header will be used to decode the appropriate T1 time slot for reconstruction of the T1 frame. By allowing each channel to have its own unique VPI/VCI address, each channel can be sent to a unique destination. This is in contrast to having all 21 channels mapped into the same ATM cell, which would require that all 21 channels have the same destination.

In an actual product, the EPROM would be replaced with a RAM, so that VPI/VCI addressing could be programmable. This would require minor F/W and L/W modifications.

Only a portion of the cell delineation procedure specified in ITU I.432 will be implemented. The basic principle of HEC detection and cell delineation will be used. Specifically, the ATM board cell delineation operation will only cover the HUNT and PRE-SYNC states of cell delineation, as defined by ITU I.432. One point that should be noted is that for a DS1 interface, received data is byte aligned in reference to the framing bit and FMB signal.

Error detection will be limited to only the detection of an error in the HEC value. No error correction features will be supported. Any ATM cell found with an error will be dropped.

The cell delineation function may eventually be replaced by an off the shelf component.[15]

6.2 Cell Delineation

Two requirements on the VPI/VCI values are necessary in order to simplify the cell delineation block.

1. The VPI/VCI addresses are within the values of 32 to 63.
2. The VPI/VCI address minus 32 is the time slot value in which the payload data should be deposited.

Requirement 1 is necessary so that our valid ATM cells are outside of the VPI/VCI range designated for idle and Operations And Management (OAM) cells and simplifies the cell delineation procedure. This also allows for a 23 fixed bit pattern of 00000000000000000000001 as an indicator for the start of an ATM cell header. The second requirement is to simplify our control logic. In theory there could be a RAM or a EPROM which decodes the VPI/VCI address to its specific time slot. However, with the proper VPI/VCI assignments, the second requirement allows the use of the 5 LSBs of the VPI/VCI value for the ATM cell channel number. See Section 11.2 for more details.

---

15. One such possible component is the TranSwitch TXC-05150. A limitation on the use of such a device is the lack of a processor or L/W to initialize the component.

Chapter 7 Physical Construction

7.1 Component Technology

Because of the high chip count, FPGAs will be used to implement the ATM board design. This will allow us to keep both the Transmitter and Receiver on a single board. There will also be some components external to the FPGAs, such as SRAMs and EPROMs. The Transmitter and Receiver portions of the ATM board are each contained in separate FPGAs. The Transmitter is described in Section 8 and Section 9. The Receiver is described in Section and Section 11.

The FPGAs used are Xilinx 3090PGA175-100. The Transmitter FPGA is configured in Master Parallel Mode and the Receiver FPGA is configured in Serial Slave Mode.

7.1.1 Design Considerations

Because Xilinx FPGAs were used, special design considerations were necessary that differ from standards designs in discrete logic. The FPGAs are composed of units called Cell Logic Blocks (CLBs). Each CLB is composed of a look up table, multiplexers and registers, see Figure 10, "Block Diagram of a Cell Logic Block (CLB)." Because of the use of a 4 to 5 input look up table, it is inexpensive to implement very complex combinational logic. However, since each CLB only has two registers, the use of registers and CLBs are directly related.

An example of such a design consideration is parity generation. One method is simply to use an eight input XOR gate. This would take up approximately 2 CLBs. However, if parity was generated serially using a register and feedback, then only 1 CLB is required. While this may seem like an insignificant savings, it is a very important savings. This is because in the initial stages of a design, the number of CLBs required to implement the design in unknown. Therefore, it is important to try at every step of the design to use the fewest possible CLBs to implement the desired functionality.

Another consideration is routing resources within the FPGA. There is actually very little that the designer can do in this area. Some designs naturally lend themselves to structures that fit well within FPGAs, others do not. Also most of the placement and routing within the FPGA is synthesized by software. Only in the extreme cases where routing resources become scarce is it necessary for manual placement and routing of CLBs and wires.

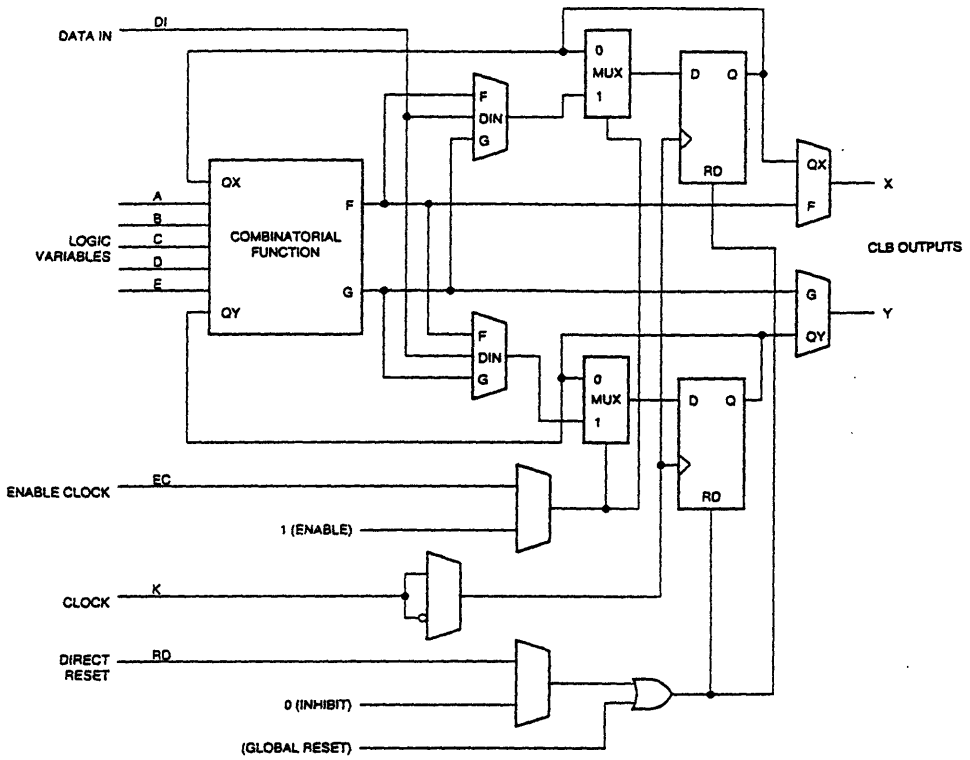


FIGURE 10. Block Diagram of a Cell Logic Block (CLB)[16]

7.2 Prototype Board Construction

7.2.1 Reset

The Xilinx FPGAs (3090PG175) will be reset by an external RESET signal. Assertion of the RESET signal will cause the FPGA's registers to be cleared. The RESET signal is generated by a debounced switch and a multivibrator/one shot. The RESET signal will be asserted for 100ns.

7.2.2 Reprogram

There is also a reprogram switch used to reprogram the Xilinx FPGAs. The RESET and REPROGRAM signals are implemented as suggested by the Xilinx data book.

7.3 Layout

The parts used and the layout for the construction of the ATM board is shown in Appendix C.

16. *The Programmable Logic Data Book*, Xilinx, Inc., 1994, p. 2-109.

Chapter 8 Transmitter Architecture

8.1 Principle of Operation

The functionality of the Transmitter can be divided into two sides, Load and Unload.[15] The basic block diagram of the Transmitter is shown in Figure 11, "Transmitter Block Diagram."

The Transmitter Load side will map and store all 32 time slots from the TMD24 into memory, regardless of whether the channels are actually in use. The active and inactive channels are separated by a mapping of the time slots. Because this is a prototype, the 21 possibly active time slots are selected and mapped in a fixed manner, see Section 9.6.2.1.2 Table 12, "FAU Channel Mapping of Speech Highways." The active time slots are mapped into SRAM channel numbers 0 to 20 and the inactive time slots are mapped into channel locations 21 to 30, see Section 9.6.2.1.2. ATM cells are generated by collecting 47 bytes, of voice or data, all of which are accumulated from a specific time slot.

The Unload side will be responsible for unloading complete ATM cells to send to the TMD24's FAU chip. All 21 possibly active channels will be unloaded. Whether active ATM cells or idle ATM cells are sent is determined by the channel status information.

Initialization data, channel status and header bytes will be obtained from the EPROM. For the Unload side, a channel number and a VPI/VCI value are conceptually equivalent since each channel has its own unique VPI/VCI address. The Unload side is also responsible for inserting the 8 inactive time slots in order to match the PCM timing specifications, see Figure 7, "TMD24 Interface Timing." The insertion of the framing bit and the actual DS1/T1 line interface is left to the TMD24.

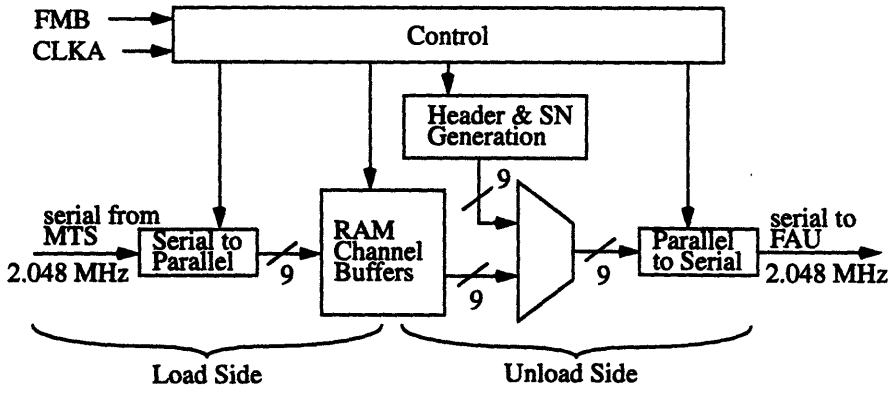


FIGURE 11. Transmitter Block Diagram

15. Generically, the Load side refers to the portion of the Transmitter or Receiver, which deals with the receiving of data. The Unload side is the portion that deals with the transmission of data.

8.2 System Architecture

The serial time slot data is internally converted into bytes on the ATM board. This allows the use of the 256KHz, C256k, clock as the reference for all the operations occurring on the byte level, see Section 9.4.1. By using C256, a significant portion of the control signals required for incrementing address counters and latching of data registers can now be hardwired to operate independently of the Control logic. The basic functions that must occur in the Transmitter for each byte loaded and unloaded can be simplified down to three types of control functions as shown in Figure 12, "Transmitter Functional Control Timing."

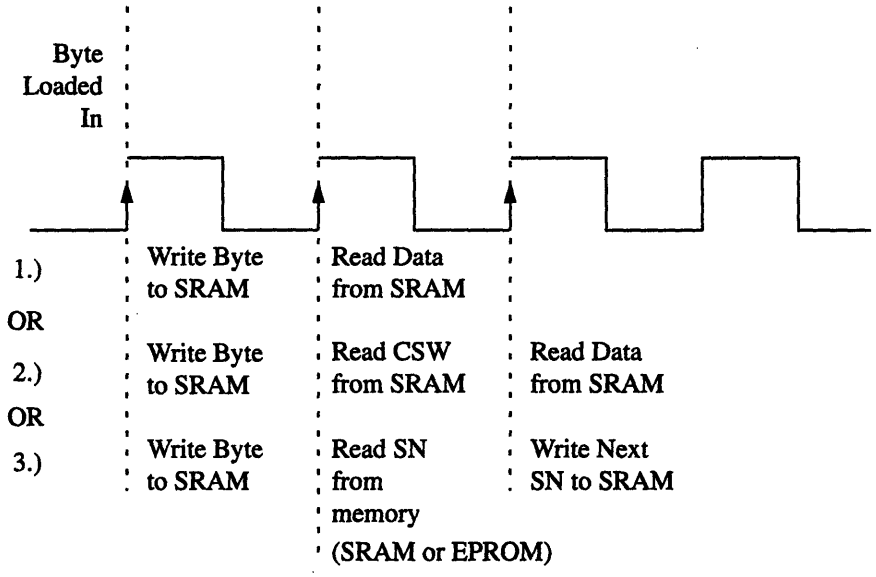


FIGURE 12. Transmitter Functional Control Timing[16]

All three options take care of the required storing of the time slot data. Option 1 is used the unloading of payload data, except for the SN byte. Option 2 is used when a new channel is started to be unloaded and option 3 is used for unloading the current SN value and generating the next SN byte. From these basic requirements, the data paths required are shown in Figure 14, "Transmitter Data Path Block Diagram."

The basic operation is as follows:

1. Serial data from TMD24 MTS chip is shifted in and an even parity bit is generated for the data receiver from the TMD24.
2. The 9 bits are then registered in the DataIn register.
3. Data from the DataIn register is written into SRAM.
4. Data is read from either the SRAM or the EPROM and registered in the TUL_Data_Out register.
5. Data from the TUL_Data_Out register is loaded into the FAU Parallel to Serial shift register.
6. Data is shifted out serially to the MTS FAU chip.

16. CSW is the Channel Status Word, see Section 8.3.4.

It should be noted that the pictorial representation of the memory address counters represent how the counters are logically incremented and does not represent how the counters are physically connected.

8.3 Transmitter Memory

The Transmitter memory is composed of a SRAM and an EPROM.

8.3.1 Data Buffer

The Transmitter data buffer is maintained in an 8k x 9 SRAM. The SRAM functions as the buffer for the time slot data. A 9 bit word SRAM is used in order to store a parity bit. The chip enable is connected to the NSRAMCS signal. The SRAM read/write input is connected to the NSRAMR/W signal. The output enable is connected to the inverse of the NSRAMR/W signal. All of these signals are generated by the Transmitter portion of the ATM board.

8.3.2 Initialization and Status Data

The Transmitter initialization and status data is maintained in an 8k x 8 EPROM. The EPROM is used to store cell header information and initial start-up data. The EPROM is used to simulate the presence of a processor interface to the switch, which would normally program initial values and status information directly into the SRAM. The EPROM therefore shares the same address and data buses as the SRAM. The EPROM chip select and output enable are both connected to the NEPROMCS signal generated from the Transmitter side of the ATM board. The programmed content of the Transmitter EPROM is shown in Appendix D.

8.3.3 Memory Partitioning

The memory addressing bits are broken down into fields as shown in Figure 13, "Transmitter Memory Address Bits."

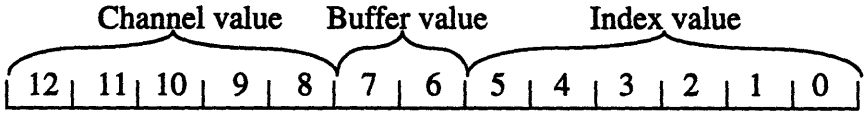


FIGURE 13. Transmitter Memory Address Bits

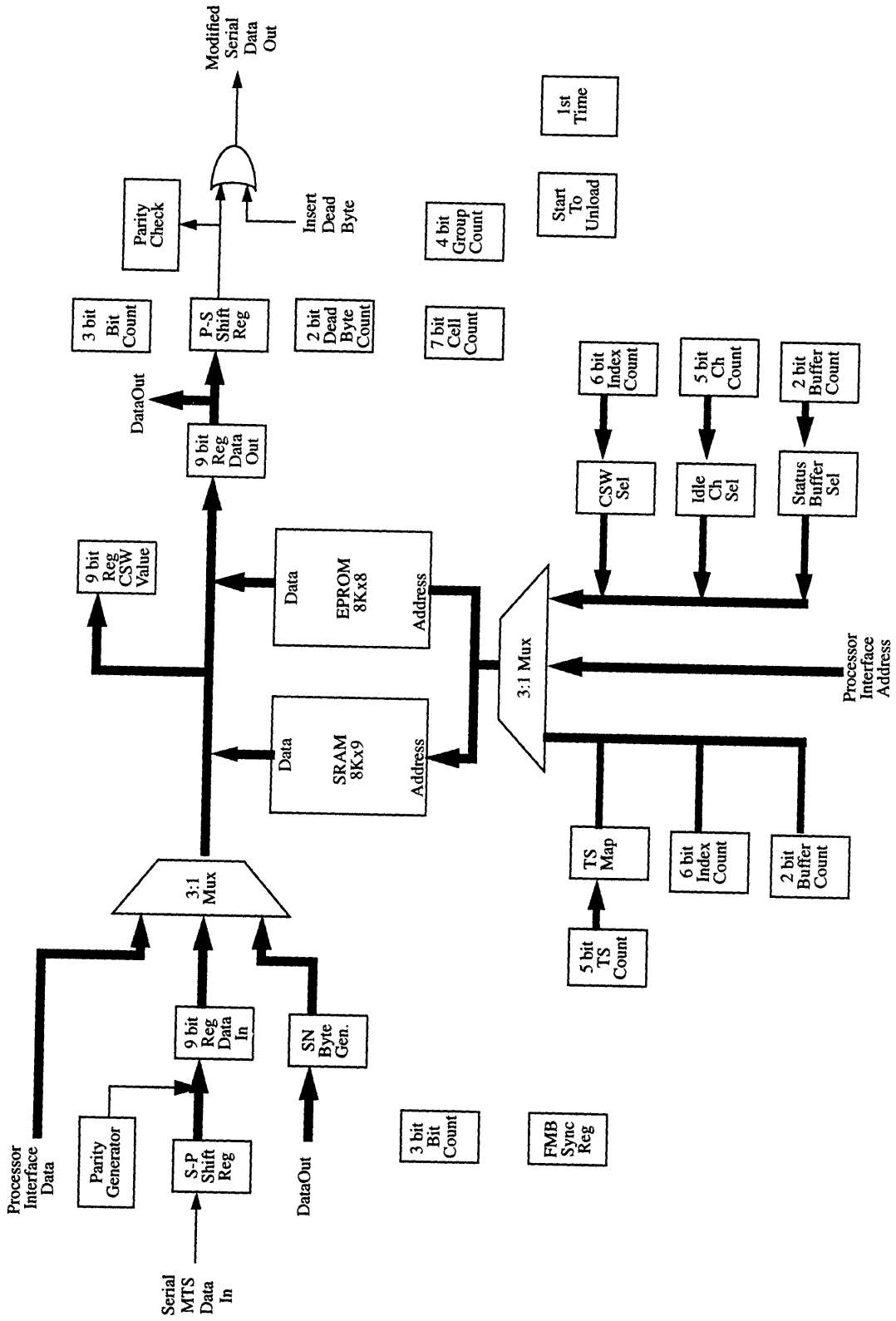


FIGURE 14. Transmitter Data Path Block Diagram

Both the Load and Unload side memory addressing are maintained in three types of fields, a Channel value, a Buffer value and an Index value. The Channel value is a five bit value used to keep track of the 32 channels loaded and the 21 unloaded channels. The Buffer value is a 2 bit value used to keep track of the three data buffers and one status buffer. The Index value is a 6 bit value used to indicate the byte location within an ATM cell.

The memory is therefore partitioned in the following manner, see Figure 15, "Transmitter Memory Partitioning." The memory consists of two types of buffers. Buffers 0, 1, and 2 are used to store PCM highway data and buffer 3 is used for status information. Address locations 0h01FC0 to 0h01FF4, which are located in channel 31 of buffer 3, are reserved for the byte values representing an idle ATM cell.

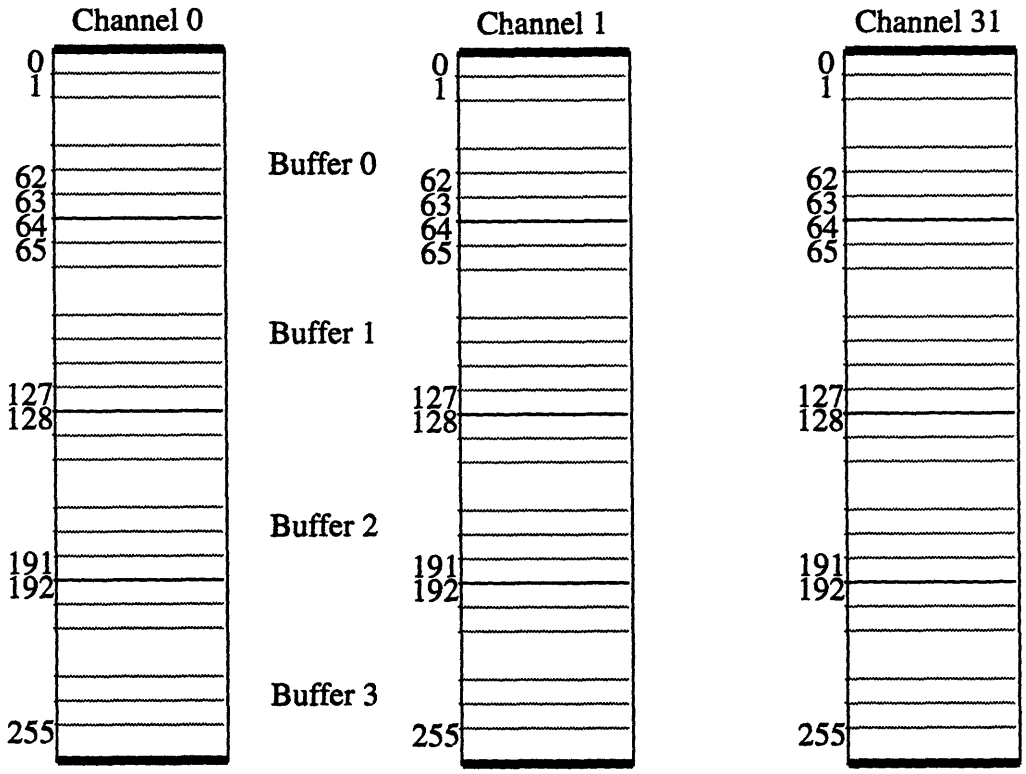


FIGURE 15. Transmitter Memory Partitioning

8.3.4 Memory Buffer Description

The data and status buffer format is shown in Figure 16, "Transmitter Buffer Partitioning." Initial SN values, channel status values and ATM header values are stored in the status buffer portion of the EPROM. Consequent SN values are stored in the status buffer portion of the SRAM. The ATM cell payloads are stored in the data buffers in the SRAM. The buffers are organized in this fashion to facilitate the loading and unloading of ATM cells as well as allowing for future modifications.

The buffers are defined as 64 byte blocks because of the simplicity of a 2^k boundary. The data buffers start at location 6, this is the first empty ATM payload location after the header and SN bytes. This therefore allows the Unload side to continuously count from 0 to 52 during the process of unloading an ATM cell. This also allows for future modifications where the header and SN bytes are directly written into the SRAM.

The Channel Status Word (CSW) byte is the status byte for each channel, see Table 5, "Channel Status Word (CSW) Byte Composition." This status value is stored in buffer 3 and is a value which is read from the EPROM. This is because for the current board, there is no high level management which can be used to set the status for active and inactive time slots.

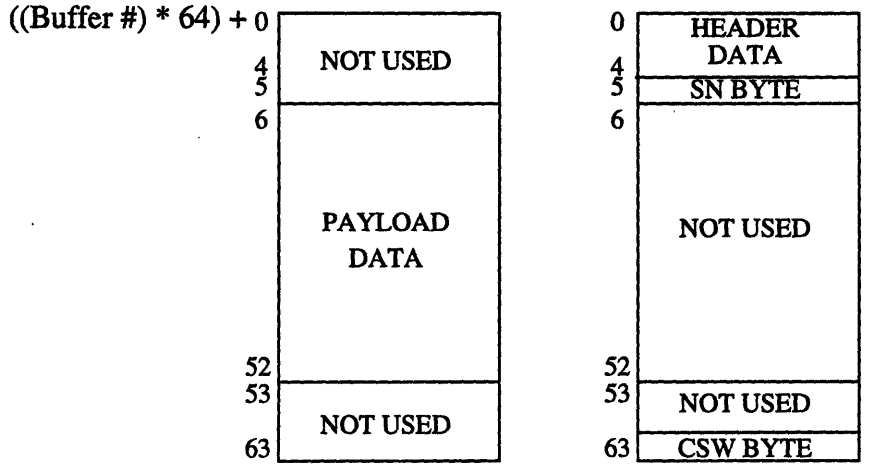


FIGURE 16. Transmitter Buffer Partitioning

TABLE 5. Channel Status Word (CSW) Byte Composition

Bit	Description
7	Active Channel Flag - set high indicates active channel
6-0	Reserved for future use.

8.3.5 Memory Selection

Ideally, if there was an interface with higher layer management, both the data and status buffers could be combined into the SRAM. This would allow the higher layer management to program channel status, initialization data and ATM header values directly into the SRAM. However, since there is no interface to higher layer management, the status buffer, except for the SN field, is mapped into the EPROM. Only the initial value for the SN is stored in the EPROM, later SN values are stored in the SRAM status buffer.

The mapping between the SRAM and EPROM is transparent to the Transmitter control FSM and is handled by lower level logic. This design architecture allows for the simulation of a higher layer management interface and in the future is higher layer management is added, the architecture allows for straight forward modifications without having to modify the Transmitter control FSM. Whether data is actually read from the SRAM or EPROM is determined by the flow chart shown in Figure 17, "Memory Access Decision Flow Diagram."

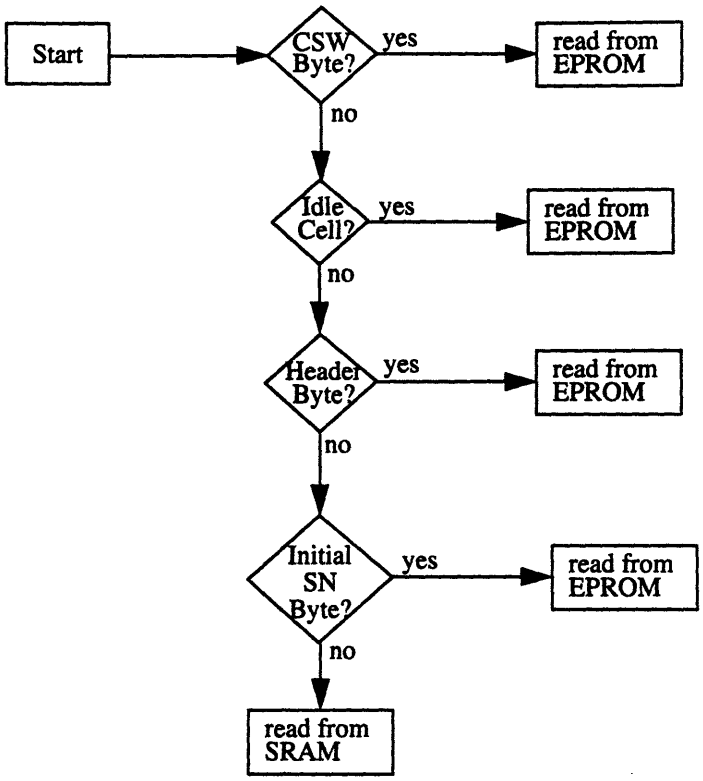


FIGURE 17. Memory Access Decision Flow Diagram

The logic equation supporting the Memory Access Decision Flow Diagram follows:

$$\text{EPROM\_Select} = \overline{\text{T\_Ack}} * \text{NSRAMR\_W\_OUT} * (\text{InsertIdle} + \text{CSW\_Sel} + \text{TUL\_IndexLT5} + (\text{SN\_Byte} * \overline{\text{NFirstTime}}))$$

If EPROM_Select is high, then the Transmitter NSRAMCS is forced high and the EPROM becomes the active memory. Otherwise if EPROM_Select is low, then the SRAM is the active memory and NEPROMCS_OUT is forced high.

The generation of an EPROM_Data signal is needed for the Parity Check block, see Section 9.7.3. This is so that parity is not checked for data read from the EPROM. The EPROM_Data signal is generated from the cascade of two registers. The first register is clocked by C1 and has its CE is connected to the T_DataOutCE signal. The input to the first register is the EPROM_Select signal. This is to store the occurrence of data being read from the EPROM and stored in the output data register, TUL_Data_Out. The second register is to provide the one time slot delay during which the serial parity check occurs. The second register is always chip enabled and is clocked by C256k. Therefore the two registers are needed to provide for the delay introduced by the latching of data in the TUL_Data_Out register and the delay of generating and checking the parity bit.

Chapter 9 Transmitter Block Descriptions

9.1 System Synchronization

9.1.1 Time Slot Synchronization

The synchronization of the ATM board to the TMD24 interface can be implemented using the Framing Marker Bit (FMB) and the 2.048MHz clock (CLKA), both of which are generated from the PBX switch. As shown in Figure 7, "TMD24 Interface Timing," the FMB is used to indicate the start of time slot 0 and therefore can be used as a system synchronization signal for the ATM board to the incoming PCM time slot data.

9.1.2 FMB Synchronization

Because the FMB signal is high during the falling edge of CLKA or C256k, it is necessary to generate a FMB_Sync signal for logic clocked by either CLKA or C256k, that require system synchronization. The FMB_Sync signal is generated by registering the FMB signal using CLKA.

9.2 Control Finite State Machine (FSM)

The data paths required are based upon the requirements of the control finite state machine (FSM). Because both the Load and Unload sides of the Transmitter are synchronous with respect to CLKA, the data management for both the Load and Unload sides of the Transmitter can be implemented in a single FSM.

Control signals for all the address counters and data registers can be implemented in the FSM. However, because of the system synchronization to CLKA and FMB, by generating a 256KHz byte clock, C256k, many of the control signals needed for memory addressing can be hardwired to increment based on the byte clock.

It is evident from Figure 12, that the clock rate for the Transmitter FSM is limited by the number of memory accesses. Therefore, in order to minimize the number of excess NOP states, CLKA is divided down by 2 to obtain a 1.024MHz clock rate (C1). The bubble flow diagram for the Transmitter FSM is shown in Figure 18, "Transmitter FSM Bubble Diagram ."

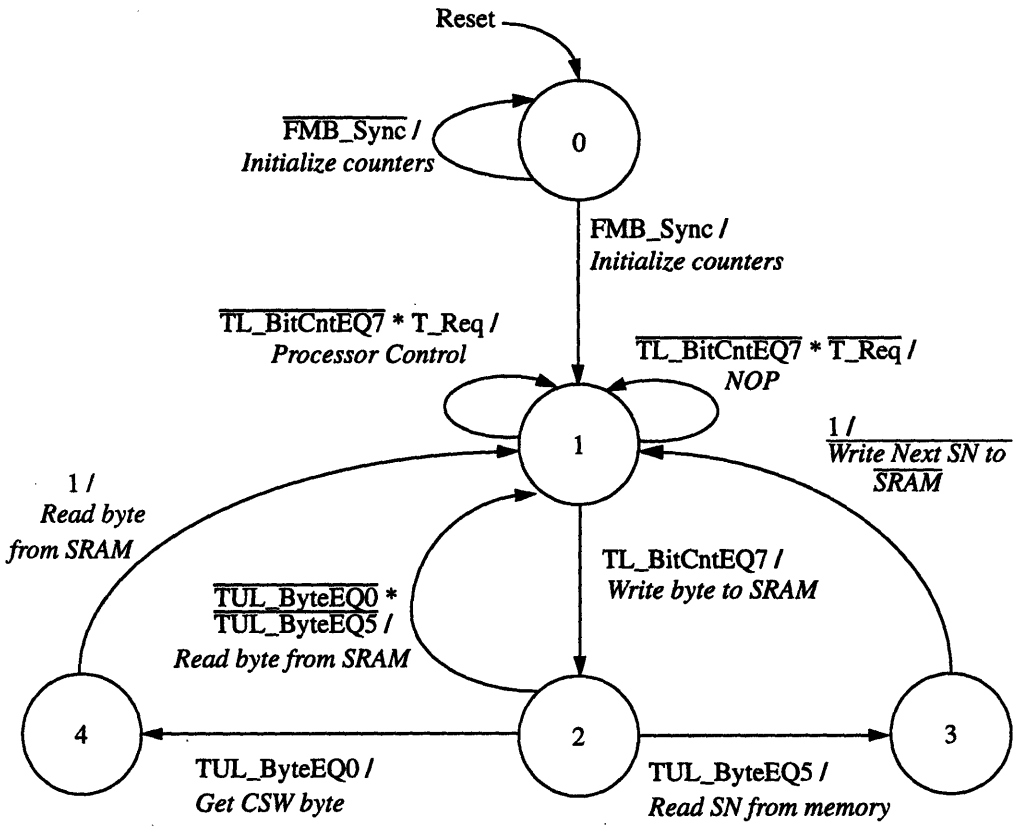


FIGURE 18. Transmitter FSM Bubble Diagram

The actual control signals asserted for each of the tasks are shown in Table 6, "Signal Assertions for Transmitter FSM Functions." The control signal descriptions are listed in Table 7, "Transmitter FSM Input Signal Description." The transition from state 1 to state 2 is the Load side function and all other states are used to maintain the Unload side functions.

TABLE 6. Signal Assertions for Transmitter FSM Functions

Task	Signals Asserted
Initialize counters	TL_IndexLD
Processor Control	T_Ack
Write byte to SRAM	MuxSel0 NSRAMCS[a] SRAM write[a]
Read byte from SRAM	MuxSel1 NSRAMCS[a] SRAM read[a] T_DataOutCE

TABLE 6. Signal Assertions for Transmitter FSM Functions

Task	Signals Asserted
Get CSW byte	CSWSel MuxSel1 NSRAMCS[a] SRAM read[a]
Read SN from memory	MuxSel1 NSRAMCS[a] SRAM read[a] T_DataOutCE
Write Next SN to SRAM	MuxSel1 NSRAMCS[a] SRAM write[a]
NOP	(No OPeration)

a. The SRAM chip select and the SRAM read/ write control signals are both negatively asserted signals.

TABLE 7. Transmitter FSM Input Signal Description

Signal Name	Signal Function
FMB_Sync	CLKA synchronized frame marker bit, see Section 9.1.2.
TL_BitCntEQ7	Status signal used to indicate that a byte of data has been shifted in, see Section 9.4.1.
T_Req	Signal used to indicate a processor request for memory access, see Section 9.5.4.
TUL_ByteEQ0	Status signal used to indicate the byte to be unloaded is the start of a new ATM cell, see Section 9.6.3.1.
TUL_ByteEQ5	Status signal used to indicate that byte to be unloaded is the SN byte of the ATM cell, see Section 9.6.3.1.

TABLE 8. Transmitter Control Signal Description

Signal Name	Signal Function
TL_IndexLD	Initial load signal for Transmitter Load side Index counter, see Section 9.6.2.2.
T_Ack	Acknowledge signal indicating that the processor interface has control of memory data and address buses, see Section 9.5.4.

TABLE 8. Transmitter Control Signal Description

Signal Name	Signal Function
MuxSel0, MuxSel1	Input source selection signals for data and address multiplexers, see Section 9.5.1 and Section 9.6.1.
NSRAMCS	Negative asserted Transmitter SRAM chip select, see Section 8.3.1.
NSRAMR_W	Negative asserted Transmitter SRAM read/ write select, see Section 8.3.1.
T_DataOutCE	TUL_Data_Out register chip enable, see Section 9.5.6.
CSWSel	Forces memory addressing to CSW status and chip enable for CSW register, see Section 9.5.5.

9.3 Initialization

9.3.1 First Time Register

Because there is no interface to higher layer management, all the necessary initialization information must be contained within the EPROM.

The NFirstTime is a negatively asserted status signal used to indicate that initialization data for the SN byte must be read from the EPROM. While NFirstTime is low, the initial value for a ATM cell's SN is read from the EPROM. The NFirstTime signal is set high after one cycle of all 21 channels has been transmitted. This is done by using feedback from the output of the First Time register and ORing it with the TULBufferCntCE signal. The TULBufferCntCE signal is used to increment the Transmitter Unload side Buffer count value and therefore can be used to indicate when all 21 channels have been transmitted. The output of the OR gate is then the input to the First Time register. The NFirstTime signal is clocked by CLKA and can only be cleared by a RESET.

In a final product, with an interface to high layer management, the initial SN values can be directly programmed into the SRAM. In this case, the NFirstTime status signal should be tied high.

9.3.2 StartToUnload

Before the Transmitter can start unloading ATM cells, it is first necessary for the Transmitter to accumulate 47 bytes of at least one channel. In order to make things conceptually simpler, we will wait until 47 bytes of all 21 channels have been accumulated. The StartToUnload signal allows the Unload side of the Transmitter to begin unloading ATM cells containing the channel data.

The StartToUnload signal is set when the Load side buffer counter is incremented. This is implemented by using the TLBufferCntCE signal and then ORing it with the feedback value from the output of the StartToUnload signal register, see Section 9.6.2.3. Therefore, the StartToUnload signal can only be cleared by a RESET.

The StartToUnload signal is a status signal that while currently is implemented in hardware, should become a status bit in the Channel Status Word (CSW) byte for each channel, see Section 8.3.4. The use

of such a status bit would require software changes to the switch along with loadware and firmware changes to the TMD24.

9.3.3 Finite State Machine Started

The FSM_Started signal is a status signal to indicate when the Transmitter control FSM has properly started and is ready to load and unload data. The FSM_Started signal is required because many of the memory address counters are hardwired to start counting as soon as they receive a clock. Therefore, the FSM_Started signal is used to prevent counters from being enabled until the Transmitter FSM is ready.

The FSM_Started signal is asserted when the Transmitter FSM reaches state 2, and remains asserted until the ATM board is reset. This is implemented by using a register and a feedback loop to the register input.

9.4 MTS to Transmitter Load Side Interface

9.4.1 Load Bit Counter

A 3 bit counter, TL_BitCnt, is used to delineate the byte boundaries of the serial data and to generate internal clocks. The TL_BitCnt counter is clocked by CLKA and reset by FMB. A 1.024 MHz clock, C1, is generated by inverting bit 0 of the counter and is used for clocking of the Transmitter control FSM. A 256 KHz clock, C256k, is generated by inverting bit 3 of the counter and is used as a byte clock. Figure 19, "TL_BitCnt and Clock Timing," shows the timing of these signals. The carry output from the counter is used to generate the status signal TL_BitCntEQ7. The TL_BitCntEQ7 status signal is used to indicate that a new byte has been received and latched into the input register.

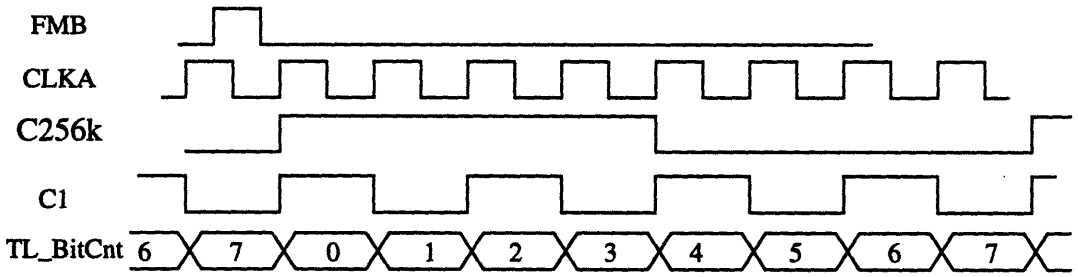


FIGURE 19. TL_BitCnt and Clock Timing

9.4.2 MTS Serial to Parallel Shift Register

This block consists of an 8 bit serial to parallel shift register which provides the conversion of serial data from the TMD24 MTS chip to 8 bit parallel data for usage on the ATM board. The MTS serial input timing is a known pattern with reference to the system FMB and CLKA, therefore only CLKA and FMB are necessary to provide synchronization with the incoming MTS data, see Figure 7, "TMD24 Interface Timing." The MTS serial to parallel shift register is hardwired to latch a bit on each rising edge of CLKA, see Figure 20, "Serial MTS Interface Timing."

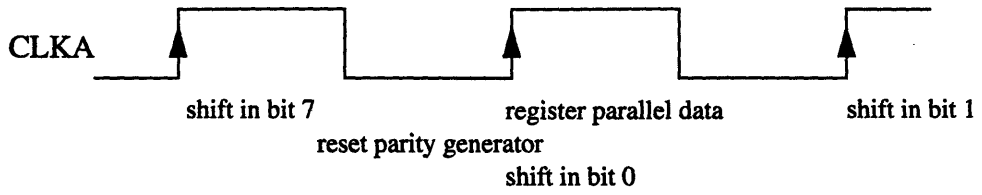


FIGURE 20. Serial MTS Interface Timing

9.4.3 Even Parity Generator

An even parity bit is generated for all the incoming data. The parity value is determined serially as the MTS serial to parallel converter shifts in each new bit. The TL_BitCntEQ7 status signal is used to clear the parity feedback path at the appropriate byte boundaries, see Section 9.4.1. The parity generator circuit is shown in Figure 21, "Even Parity Generator Block."

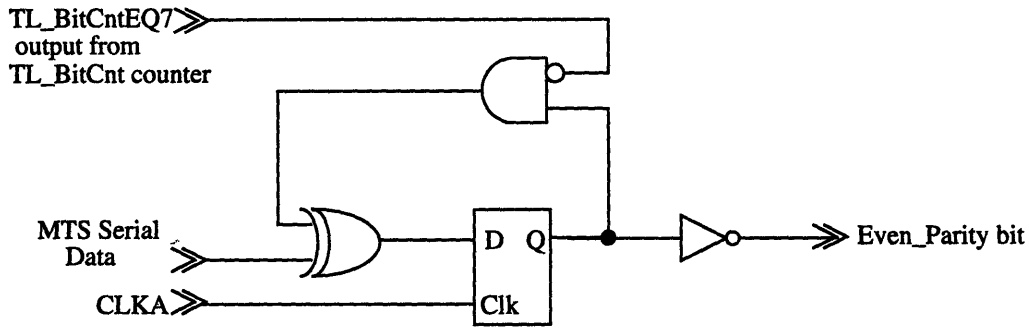


FIGURE 21. Even Parity Generator Block

9.5 Transmitter Memory Data Bus

9.5.1 Data Input Multiplexer

The data input to the SRAM can come from three sources, the DataIn register, the SN generation block, refer to Section 9.5.3, or from a processor interface. The data selection is done using a 9 bit 3 to 1 multiplexer, DataInMux, which is controlled by the selection signals MuxSel0 and MuxSel1, see Section 9.2.

TABLE 9. Data_In Multiplexer Selection Table

Input		Output
MuxSel1	MuxSel0	Data Source
0	0	Processor Interface
0	1	DataIn Register
1	0	SN Generator Block
1	1	Not Used

9.5.2 DataIn Register

The DataIn register is a 9 bit register that latches the 8 bit output from the MTS serial to parallel shift register and the 1 bit output from the even parity generator. The DataIn register is clocked by CLKA and the CE of the DataIn register is connected to the TL_BitCntEQ7 output from the Transmitter Load Bit counter block. Therefore data will be latched at the end of each time slot. This combination of control signals is used instead of the C256k clock with the CE tied high because the timing between the MTS serial to parallel shift register and DataIn register is critical. Since routing delays may vary in a FPGA and also because C256k is derived from CLKA, the setup and hold times for the DataIn register may not be met if C256k was used instead of CLKA. The output of the DataIn register is one of the inputs to the DataInMux multiplexer. Figure 20, "Serial MTS Interface Timing," shows the timing for latching in data by the DataIn register.

9.5.3 Sequence Number (SN) Generation Block

The Unload section is also responsible for the generation of the next sequence number byte for the current channel being processed. The steps for the SN byte generation are as follows, also see Figure 12, "Transmitter Functional Control Timing," option 3:

1. Read SN byte from memory and latch data in TUL_Data_Out register.
2. Generate next SN based on value in TUL_Data_Out register.
3. Write next SN in the status buffer location in the SRAM.

The generation of the next SN is implemented in combinational logic. The input to the SN generator block is from the TUL_Data_Out register, see Section 9.5.6, and the output of the SN generator block is to the DataInMux. The format for the SN byte is shown in Figure 22, "SN Byte Format," and the truth table implemented by the SN generator block is shown in Table 10, "Truth Table for SN Generations."

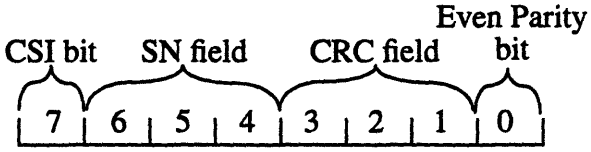


FIGURE 22. SN Byte Format

TABLE 10. Truth Table for SN Generations

INPUTS				OUTPUTS							
CSI	SN2	SN1	SN0	CSI	SN2	SN1	SN0	CRC2	CRC1	CRC0	Parity
0	0	0	0	0	0	0	1	0	1	1	0
0	0	0	1	0	0	1	0	1	1	0	0
0	0	1	0	0	0	1	1	1	0	1	1
0	0	1	1	0	1	0	0	1	1	1	1
0	1	0	0	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0	1	0
0	1	1	0	0	1	1	1	0	1	0	1
0	1	1	1	0	0	0	0	0	0	0	0

TABLE 10. Truth Table for SN Generations

INPUTS				OUTPUTS							
CSI	SN2	SN1	SN0	CSI	SN2	SN1	SN0	CRC2	CRC1	CRC0	Parity
1	0	0	0	1	0	0	1	1	1	0	1
1	0	0	1	1	0	1	0	0	1	1	1
1	0	1	0	1	0	1	1	0	0	0	0
1	0	1	1	1	1	0	0	0	1	0	0
1	1	0	0	1	1	0	1	0	0	1	1
1	1	0	1	1	1	1	0	1	0	0	1
1	1	1	0	1	1	1	1	1	1	1	0
1	1	1	1	1	0	0	0	1	0	1	0

The logic equations representing Table 10, "Truth Table for SN Generations," are shown below. The left side of the equations represent the output values and the right side of the equations represent the input values:

$$CSI = CSI$$
$$SN2 = SN2\,\overline{SN1} + SN1*(SN2 \oplus SN0)$$
$$SN1 = SN1 \oplus SN0$$
$$SN0 = \overline{SN0}$$
$$CRC2 = \overline{SN1}*\overline{SN0}*(CSI \oplus SN2) + \overline{CSI}*\overline{SN2}*(SN2 + SN1) + CSI*SN2*(SN1 + SN0)$$
$$CRC1 = \overline{SN2}*(\overline{SN1} + SN0) + SN2*SN1*\overline{SN0}$$
$$CRC0 = \overline{CSI}*\overline{SN2}*(SN1+\overline{SN0}) + CSI*SN2*(SN1 + \overline{SN0}) + \overline{SN1}*SN0*(CSI \oplus SN2)$$
$$Parity = CSI*\overline{SN1} + \overline{CSI}*SN1*(\overline{SN2} + \overline{SN0})$$

9.5.4 Processor Interface

The processor interface is a feature which is not implemented in the current version of the Transmitter. The original purpose for a processor interface was for direct and simple access to the contents of the SRAM. The hooks in the Transmitter control FSM are in place for the addition of a processor interface, refer to Section 9.2, however, currently there is no actual support for the processor addressing and data registers.

The Transmitter FSM is designed such that when there is a free clock cycle and a processor request, then the SRAM data and addressing paths are allocated for use by the processor. The processor request signal for usage of the Transmitter side data paths is the T_Req signal. During the clock cycle which the processor has access to the memory data and address buses, the Transmitter control FSM asserts the signal T_Ack. Any additional hardware or state machines required for a specific processor interface is not provided.

9.5.5 Channel Status Word (CSW) Register

The CSW register is an 8 bit register, CSW_Value, which is clocked by C1 and used to hold the status byte for the current channel being unloaded. The CE for the CSW_Value register is connected to the CSWSel signal from the Transmitter FSM. The input to the CSW_Value register is the memory data bus. Currently only the MSB of the CSW value is used, see Section 8.3.4.

9.5.6 DataOut Register

The TUL_Data_Out register is used to buffer data before it is sent to the FAU parallel to serial shift register. The TUL_Data_Out register is a 9 bit register which is clocked by C1. The CE is connected to the T_DataOutCE signal from the Transmitter FSM. The input to the TUL_Data_Out register is the output from the memory data bus.

9.6 Transmitter Memory Addressing

9.6.1 Memory Address Multiplexer

The Transmitter memory addressing can come from three sources, the processor address interface, see Section 9.5.4, the Load side addressing, see Section 9.6.2, and the Unload side addressing, see Section 9.6.3. The Transmitter memory address multiplexer is a 3 to 1 13 bit multiplexer. The input selection is controlled by the MuxSel0 and the MuxSel1 signals, see Table 11, "Transmitter Address Multiplexer Selection Table." The output of this multiplexer is used as the memory addressing bits.

TABLE 11. Transmitter Address Multiplexer Selection Table

Input Select		Output
MuxSel1	MuxSel0	Address Source
0	0	Processor Address Interface
0	1	Load Side Addressing
1	0	Unload Side Addressing
1	1	NA

9.6.2 Memory Load Side Addressing

The memory addressing counters for the Load side are logically organized as shown in Figure 23, "Load Side Logical Organization of Memory Addressing Bits." The Channel Counter value is used as the LSBs, followed by the Index Counter value and finally the Buffer Counter value as the MSBs. The actual physical bit organization is shown in Figure 13, "Transmitter Memory Address Bits."

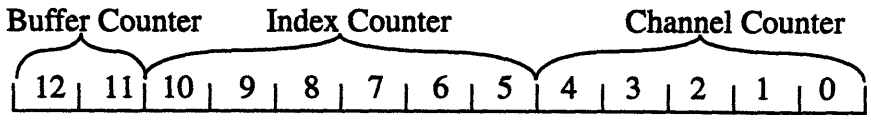


FIGURE 23. Load Side Logical Organization of Memory Addressing Bits

The Channel Counter value ranges from 0 to 31, the Index Counter value ranges from 6 to 52 and the Buffer Counter value ranges from 0 to 2.

9.6.2.1 Channel Counter

The Channel counter field is generated by the Time Slot Counter and the Time Slot Mapping blocks.

9.6.2.1.1 Time Slot Counter

The time slot counter, TL_TimeSlot_Count, is a 5 bit counter used to keep track of the time slot of the data being loaded by the MTS serial to parallel shift register, see Section 9.5.2. The TL_TimeSlot_Count counter is clocked by C256k and uses FMB_Sync for the synchronous reset signal. Because TL_TimeSlot_Count is clocked by C256k, the CE can be tied high, eliminating the need for external control signals. The output of the TL_TimeSlot_Count counter is used as the input to the Time Slot Mapping block, see Section 9.6.2.1.2.

9.6.2.1.2 Time Slot Mapping

The time slot mapping block, TS_Mapping_Logic, takes the time slot value from the TL_TimeSlot_Count counter and generates the final Channel count value used for memory addressing. The time slot to channel mapping in an actual product should be implemented in programmable memory which is programmed by higher layer management during either system configuration or board initialization. However, for the prototype, the mapping is fixed as shown in Table 12, "FAU Channel Mapping of Speech Highways," and therefore is implemented in combinational logic.

It should be noted that since the data is delayed by one time slot because of the Data_In register. This one time slot delay is incorporated into the Time Slot Mapping block as a minus 1 function. For example, the logical mapping desired for time slot 0 is channel 21. However, due to the one time slot delay, when the data is actually written into the SRAM, the TL_TimeSlot_Count counter value will be equal to 1. The mapping logic accommodates for this delay so that the physical SRAM channel address, to which the time slot is mapped, is channel 21, as shown in Table 12. It should also be noted that because channel 31 is reserved for the idle ATM cell, time slots 24 and 28 are both mapped into channel 30.

Column 1 of Table 12 shows the actual time slot values from the TL_TimeSlot_Count counter. Column 2 shows the standard time slot assignments as implemented by the TMD24. Column 3 contains the logical channel mappings for the time slots. Finally, column 4 shows the actual output from the TS_Mapping_Logic block, taking the 1 time slot delay into account, which is used as the value for the Channel count portion of the memory addressing.

TABLE 12. FAU Channel Mapping of Speech Highways

PCM Time Slot	Standard PBX Channel Numbering	Logical Channel Mapping	Physical Channel Address Mapping[a]
0	-	21	20
1[b]	1	22	21
2[b]	2	23	22
3[b]	3	24	23
4	-	25	24
5	4	0	25
6	5	1	0

TABLE 12. FAU Channel Mapping of Speech Highways

PCM Time Slot	Standard PBX Channel Numbering	Logical Channel Mapping	Physical Channel Address Mapping[a]
7	6	2	1
8	-	26	2
9	7	3	26
10	8	4	3
11	9	5	4
12	-	27	5
13	10	6	27
14	11	7	6
15	12	8	7
16	-	28	8
17	13	9	28
18	14	10	9
19	15	11	10
20	-	29	11
21	16	12	29
22	17	13	12
23	18	14	13
24	-	30	14
25	19	15	30
26	20	16	15
27	21	17	16
28	-	30	17
29	22	18	30
30	23	19	18
31	24	20	19

a. Only SRAM address values 0-20 will actually be transmitted.

b. Time slot will be masked off during configuration.

The logic equations used for the mapping are as follows:

$$Q4 = \overline{D4}\,\overline{D3}\,\overline{D2} + \overline{D2}\,\overline{D1}\,D0 + D4\,D3\,D0 + D4\,D3\,D2\,\overline{D0} + D2\,\overline{D1}\,D0 + \overline{D4}\,\overline{D3}\,\overline{D2}\,\overline{D1}$$

$$Q3 = D4\,\overline{D2}\,\overline{D0} + D4\,\overline{D3} + D3\,D2\,\overline{D1}\,D0 + \overline{D4}\,\overline{D3}\,D2\,\overline{D1} + D3\,\overline{D2}\,\overline{D1}\,D0$$

$$Q2 = \overline{D4}\,\overline{D3}\,\overline{D2} + D4\,D3\,\overline{D2}\,\overline{D0} + D4\,\overline{D2}\,\overline{D1}\,D0 + \overline{D4}\,D3\,D1\,D0 + \overline{D4}\,D3\,D2\,\overline{D0} + D4\,\overline{D3}\,D2\,D1 + D4\,D2\,\overline{D1}\,D0$$

$$Q1 = D3\,\overline{D2}\,\overline{D0} + D3\,D1\,\overline{D0} + \overline{D4}\,\overline{D3}\,\overline{D2}\,D1 + D3\,\overline{D1}\,D0 + D3\,D2\,D0 + D4\,\overline{D3}\,\overline{D2}\,D1\,D0 + D4\,\overline{D3}\,D2\,\overline{D1}\,\overline{D0}$$

$$Q0 = D3\,\overline{D2}\,D1\,\overline{D0} + D4\,\overline{D2}\,D1\,\overline{D0} + \overline{D4}\,\overline{D3}\,D0 + \overline{D4}\,D2\,D0 + D2\,D1\,D0 +$$

$D3\ D2\ \overline{D1}\ \overline{D0} + D4\ \overline{D3}\ D2\ \overline{D1}$

D4 - D0: are the time slot values (0-31). D4 is the MSB and D0 is the LSB.

Q4 - Q0: are the channel values (0-31). Q4 is the MSB and Q0 is the LSB.

The mapped channels fill the SRAM as shown in Figure 24, "SRAM Time Slot Mapping." The Time Slot Mapping block also generates the signal EQ20. This signal is based on the TS_Mapping_Logic output and is asserted when the output value of the TS_Mapping_Logic is equal to 20. This is used to indicate an end of a T1 frame.

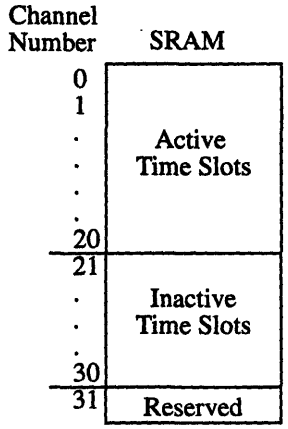


FIGURE 24. SRAM Time Slot Mapping

9.6.2.2 Index Counter

The TL_Index_Count counter is a 6 bit loadable counter used to keep track of which byte of the ATM cell payload is currently being written to. The Index Counter block is shown in Figure 25, "Index Counter Block."

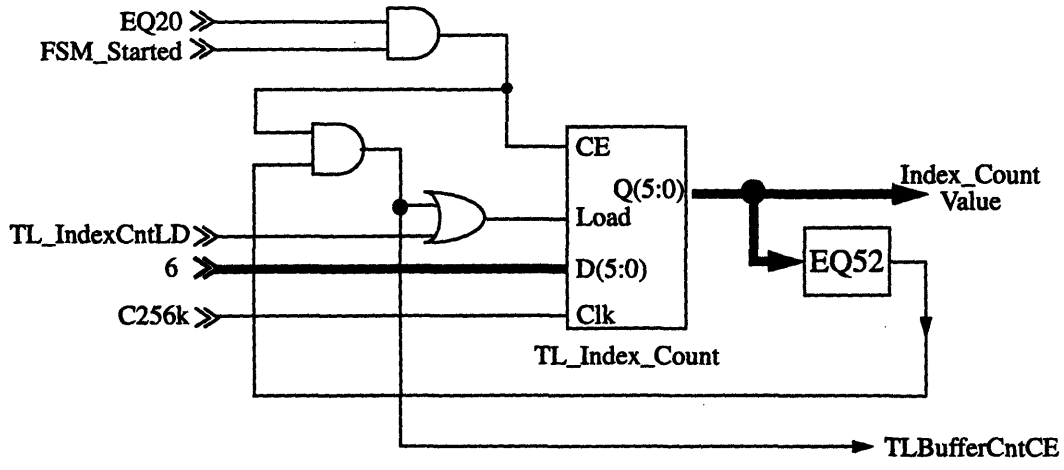


FIGURE 25. Index Counter Block

The EQ20 signal is the status signal generated from the Time Slot Mapping Logic block and is the basis for the CE for the TL_Index_Count counter. The EQ20 is ANDed with the FSM_Started signal to generate the final CE for the Index_Count counter. The FSM_Started signal is used to prevent the Index_Count counter from incrementing before the Transmitter FSM is ready. Thus, once the FSM has started, the Index value is incremented after each T1 frame. The TL_IndexCntLD signal is a control signal generated by the Transmitter control FSM and is used for the initial loading of the counter. Normal loading of the TL_Index_Count counter occurs when the TL_Index_Count counter value equals 52 and a CE occurs, this signal, TLBUfferCntCE, is also used in the Transmitter Load side Buffer Counter block, see Section 9.6.2.3. The TL_Index_Count counter is hardwired to always load in the value of 6, see Section 8.3.4. Because data is byte aligned internally on the ATM board, the TL_Index_Count counter is clocked by C256k and is reset only by a RESET signal.

9.6.2.3 Buffer Counter Block

The Buffer Counter block consists of a 2 bit counter, TL_Buffer_Count, and is used to keep track of the active data buffer. The TL_Buffer_Count counter is clocked by C256k and uses the TLBufferCntCE signal, generated as shown in Figure 25, "Index Counter Block," for the chip enable. Therefore, after the TL_IndexCntLD signal assertion for initialization, the TL_Buffer_Count counter increments each time the TL_Index_Count counter is loaded. Since buffer 3 is reserved for status information, the Buffer_Count counter is reset when the TL_Buffer_Count value equals 2 and a TLBufferCntCE occurs.

9.6.3 Unload Side Addressing

The memory addressing counters for the Unload side are logically organized as shown in Figure 23, "Load Side Logical Organization of Memory Addressing Bits." Logically the counters increment with the Index Counter value as the LSBs, followed by the Channel Counter value and the Buffer Counter value as the MSBs. The actual physical bit organization is shown in Figure 13, "Transmitter Memory Address Bits."

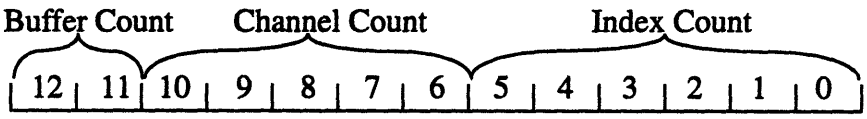


FIGURE 26. Unload Side Logical Organization of Memory Addressing Bits

The Index Counter value ranges from 0 to 52, the Channel Counter value ranges from 0 to 20 and the Buffer Counter value ranges from 0 to 2.

9.6.3.1 Index Counter

The Index Counter block consists of a 6 bit counter, TUL_Index_Cnt, an Index_Value_Decoder logic block and a CSW_Status_Force logic block. Figure 27, "Transmitter Unload Side Index Counter Block," shows the interconnection of the 3 blocks.

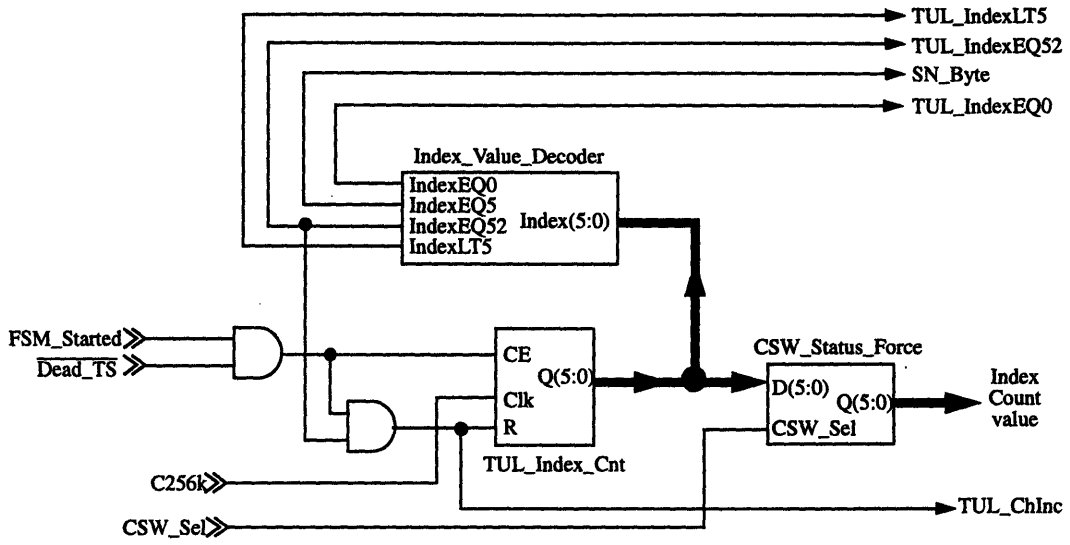


FIGURE 27. Transmitter Unload Side Index Counter Block

The TUL_Index_Cnt counter is clocked by C256k and has its CE connected to the AND of FSM_Started and Dead_TS status signals. The FSM_Started signal is needed so that the TUL_Index_Cnt counter value does not start incrementing until the Transmitter control FSM has been initialized and is ready to start unloading ATM cells. The Dead_TS signal is needed to prevent the TUL_Index_Cnt counter from incrementing during inactive time slots, see Section 9.7.4. The synchronous reset signal to the TUL_Index_Cnt counter occurs when the TUL_Index_Cnt value is equal to 52 and a CE for the TUL_Index_Cnt occurs, this signal TUL_ChInc, is also used in the Channel Count block, see Section 9.6.3.2.

The output of the TUL_Index_Cnt counter is used as the input to the CSW_Status_Force block. When the CSW_Sel control signal is asserted, the CSW_Status_Force block forces the Index Count value to 63, otherwise the TUL_Index_Cnt value is passed through. An Index Count value of 63 is the address location for the CSW status for a channel, see Section 8.3.4. The output from the CSW_Status_Force block is then the value used for the Index Count field for memory addressing.

The Index_Value_Decoder is composed of combinational logic used to generate status signals based on the TUL_Index_Cnt counter value. The four status signals generated are TUL_IndexEQ0, SN_Byte, TUL_IndexEQ52 and IndexLT5. The TUL_IndexEQ0 signal is generated from the IndexEQ0 output of the Index_Value_Decoder block and is used to indicate the start of a new ATM cell. The SN_Byte signal is the IndexEQ5 output from the Index_Value_Decoder block and is used to indicate that the current byte being unloaded is the SN byte. The TUL_IndexEQ52 signal is generated from the IndexEQ52 output of the Index_Value_Decoder block and is used to indicate the end of an ATM cell. The TUL_IndexLT5 is equal to the IndexLT5 output of the Index_Value_Decoder block and is used to indicate that the current byte is part of the 5 byte ATM cell header.

9.6.3.2 Channel Counter

The Channel Counter block consists of a 5 bit counter, TUL_Channel_Cnt, and an IdleChSel block, see Figure 28, "Transmitter Unload Side Channel Count Block." The Channel Counter block is used to keep track of the current channel being unloaded. Since the 21 active time slots have been mapped to channel numbers 0 to 20, TUL_Channel_Cnt counter needs only to count from 0 to 20. The TUL_Channel_Cnt counter is clocked by C256k and its CE is the logical AND of TUL_ChInc,

StartToUnload and InsertIdle. The TUL_ChInc signal is generated by the Transmitter Unload Side Index Counter block, see Section 9.6.3.1. Therefore, each time the TUL_Index_Cnt counter is reset, indicating that an entire ATM cell has been transmitted, the TUL_Channel_Cnt counter is incremented. The StartToUnload signal is used to prevent the TUL_Channel_Cnt value from incrementing until the Transmitter has accumulated enough data to begin transmitting ATM cells. The InsertIdle signal is required so that when there is a need to insert an idle ATM cell to maintain the data rate, the TUL_Channel_Cnt counter is disabled. The synchronous reset signal for the Transmitter Unload side Channel counter occurs when the TUL_Channel_Cnt value equals 20 and a CE occurs, resetting the TUL_Channel_Cnt counter value to zero. The TUL_Channel_Cnt counter reset signal is also used as the CE signal for the TUL_Buffer_Count counter, refer to Section 9.6.3.3.

The output of the TUL_Channel_Cnt counter is used as the input to the IdleChSel block. The IdleChSel block is used to force the Channel Count value to 31 when the IdleCh_Sel signal is asserted. Otherwise the IdleChSel block will pass on the unaltered TUL_Channel_Cnt counter value. A channel value of 31 is the channel number allocated for the idle ATM cell structure, see Section 8.3.3. The IdleCh_Sel is logically equal to (CSW_Sel * (StartToUnload + InsertIdle + CSW_Value(7)). Therefore the IdleCh_Sel signal is asserted when the Transmitter is not ready to start unloading ATM cells, when it is necessary for the Transmitter to insert an idle ATM cell and when a particular channel is inactive. However, when the CSW_Sel signal is asserted, the IdleCh_Sel signal is deasserted, this allows a new CSW status byte to be obtained from the appropriate channel.

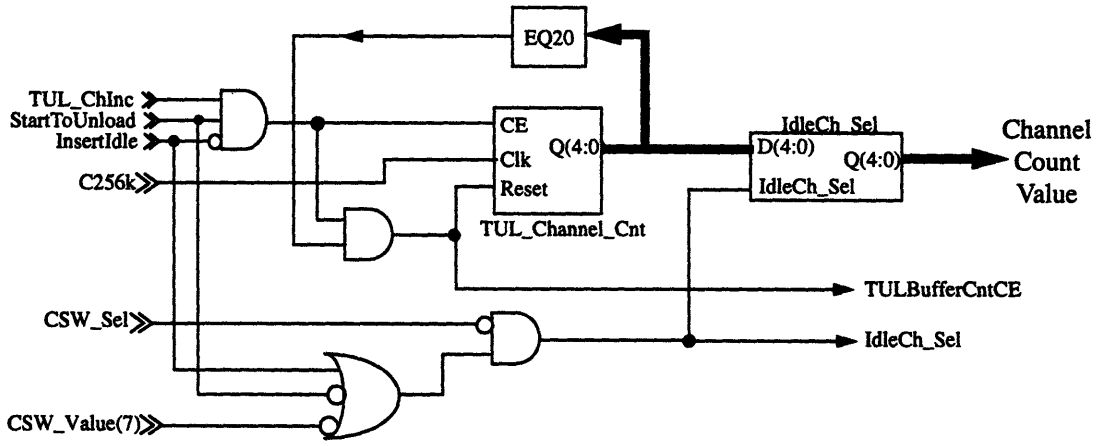


FIGURE 28. Transmitter Unload Side Channel Count Block

9.6.3.3 Buffer Count

The Transmitter Unload side Buffer Count block consists of a 2 bit counter, TUL_Buffer_Count, and a TUL_Status_Buffer_Force logic block.

The TUL_Buffer_Count counter is used to keep track of the current active buffer number. The TUL_Buffer_Count counter is clocked by CLKA and its CE is connected to the signal TULBufferCntCE, generated in the TUL_Channel_Cnt counter block, see Section 9.6.3.2. Therefore each time the TUL_Channel_Cnt counter is reset, the TUL_Buffer_Count counter is incremented. The TUL_Buffer_Count counter is reset when the TUL_Buffer_Count counter value is equal to 2 and a TULBufferCntCE occurs. This is because only buffer values 0, 1 and 2 are used for ATM cell payload storage.

The input to the TUL_Status_Buffer_Force logic block is from the TUL_Buffer_Count counter and the output from the TUL_Status_Buffer_Force logic block is used as the Buffer Count field for memory

addressing. The TUL_Status_Buffer_Force logic block forces the buffer value to 3, which is the buffer number reserved for channel status and header values. The TUL_Status_Buffer_Force logic block is enabled by the output of the logical OR of the SN_Byte status signal, the TUL_IndexLT5 status signal, the CSW_Sel signal and the IdleCh_Sel signal from the Transmitter Unload Channel Count block. The assertion of any of these status signals indicates that data needs to be read from the status buffer, see Section 8.3.5.

9.7 Transmitter to FAU Unload Side Interface

9.7.1 Unload Bit Counter

A 3 bit counter, TUL_BitCnt, is used for byte delineation of the unloaded data. The TUL_BitCnt counter is clocked by CLKA and reset by FMB. The terminal count (TC)[17] output of the TUL_BitCnt counter is used as the LOAD signal for the FAU parallel to serial shift register, see Section 9.7.2, and the CE for the Dead_Byte_Counter counter, see Section 9.7.4. There is also a NOR gate used to generate a status signal for indicating when the TUL_BitCnt value is equal to zero. This signal is used for clearing the parity check in by the Parity Check block, see Section 9.7.3.

9.7.2 FAU Parallel to Serial Shift Register

In order to match the TMD24 timing requirements for the FAU chip, the FAU parallel to serial interface is clocked by CLKA, see Figure 7, "TMD24 Interface Timing." The TC output from the TUL_BitCnt counter is used as the LOAD signal to the FAU parallel to serial converter, see Figure 29, "Parallel to Serial Conversion Timing." The input to the FAU parallel to serial shift register is from the TUL_Data_Out register. The serial output is to the Dead Byte Insertion block, see Section 9.7.4.

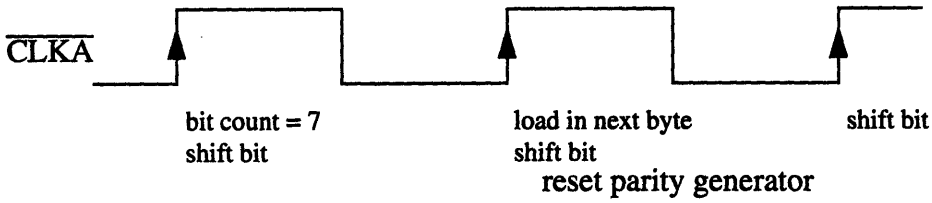


FIGURE 29. Parallel to Serial Conversion Timing

9.7.3 Parity Check

An even parity check is performed serially on data loaded into the FAU parallel to serial shift register. The circuit used is shown in Figure 30, "Parity Check."

Note that while the FAU parallel to serial converter is clocked by CLKA, the parity check logic is clocked using CLKA. This is done in order to provide adequate setup and hold timing for the parity generation register. Because of the half clock cycle delay between CLKA and CLKA, Byte_Parity and Parity_Error value registers are clocked in by C256k. The EPROM_Data signal is required to zero out

17. The logic equation for TC is $(Q_0 * ... * Q_{n-2} * Q_{n-1} * Q_n)$ for a n bit counter, where Q_k represents the k^{th} bit of the counter.

the parity check when the data being shifted out is from the EPROM. This is because the EPROM data bus is only 8 bits, and therefore lacks a parity bit, see Section 8.3.2.

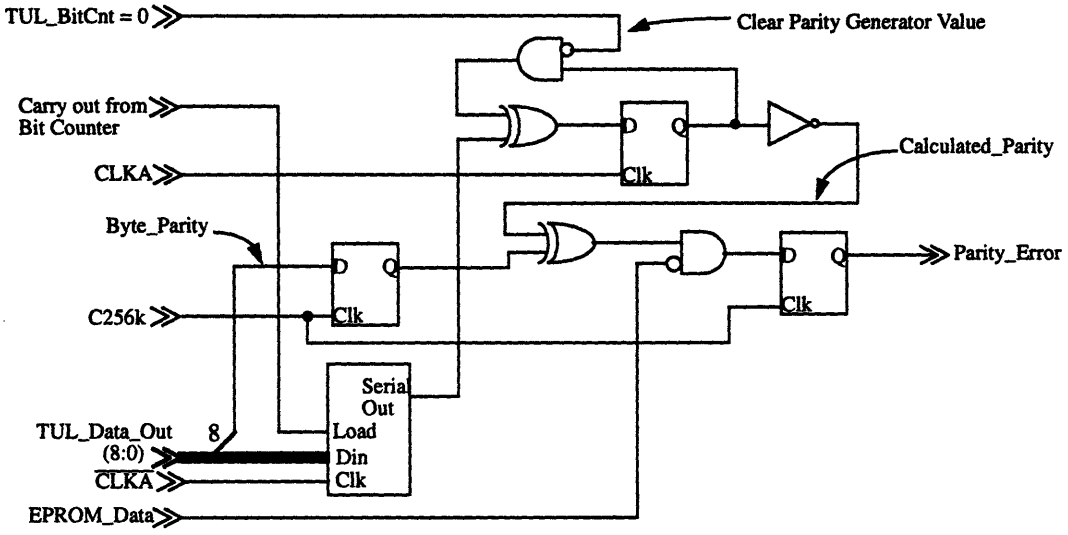


FIGURE 30. Parity Check

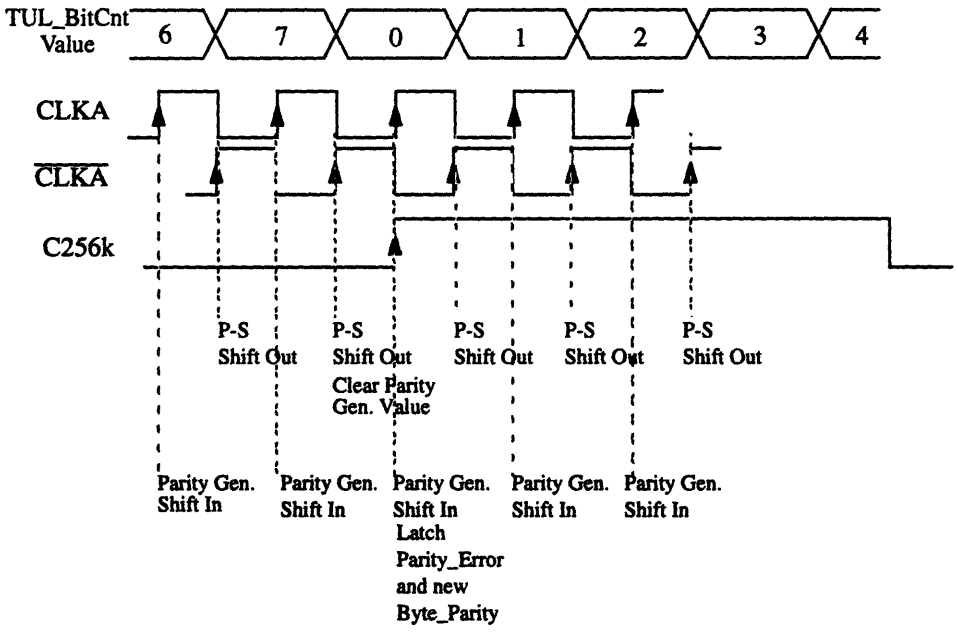


FIGURE 31. Parity Generator Timing

9.7.4 Dead Byte Insertion

In order to properly match the time slots to the timing expected by the FAU, 8 inactive time slots must be inserted. This is because while the actual T1 data rate is for 24 channels, the internal rate on the

TMD24 board is for 32 channels. The 8 inactive time slots are 0, 4, 8, 12, 16, 20, 24 and 28, see column 2 of Table 12, "FAU Channel Mapping of Speech Highways." A 2 bit counter, Dead_Byte_Counter, is used to count when a particular time slot should be replaced with the all 1's pattern used to indicate an inactive time slot. When the Dead_Byte_Counter value is equal to zero, then the serial data out is forced high for eight cycles of CLKA, see Figure 32, "Dead Byte Data Insertion." Since data is clocked out with reference to CLKA, then in order to obtain the appropriate timing, Dead_Byte_Counter must also be clocked by CLKA and can therefore be reset by FMB, see Figure 33, "Dead Byte Data Insertion Timing." The Dead_Byte_Counter counter is chip enabled by the TC output from the TUL_BitCnt counter. The Dead_TS signal generated by the TC output from the Dead_Byte_Counter counter is used to indicate that the next time slot is one of the 8 inactive time slots.

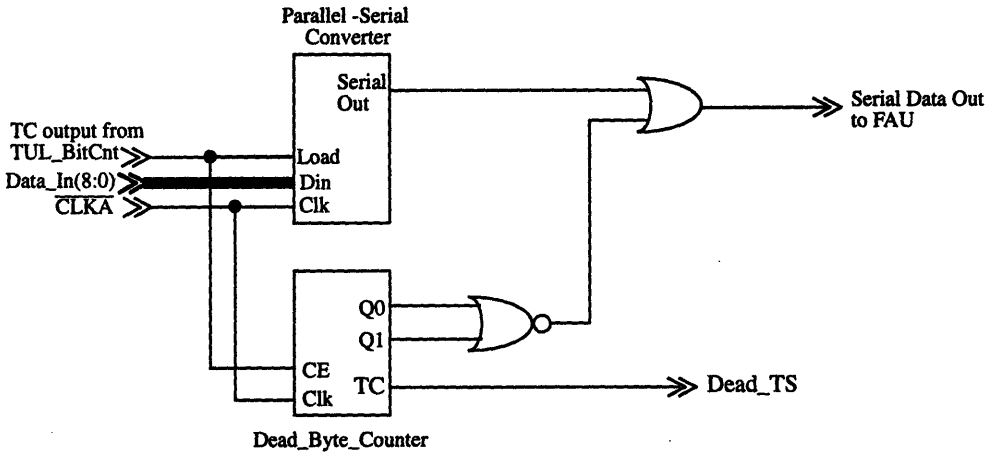


FIGURE 32. Dead Byte Data Insertion

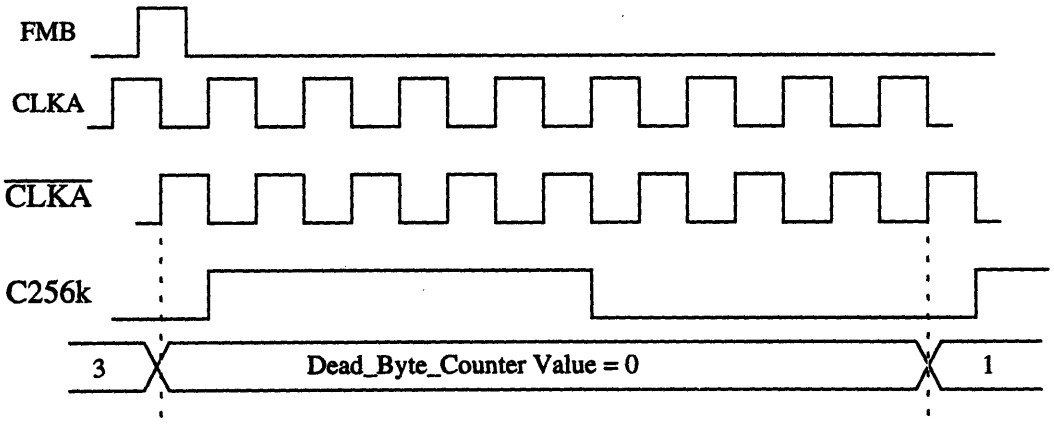


FIGURE 33. Dead Byte Data Insertion Timing

9.8 Idle Pattern Insertion Counters

There are many methods for organizing the insertion of idle ATM cells to match the input and output data rate timings. The only requirement for the smallest repeatable pattern is that there are 1113 ATM cells filled with PCM data from 21 channels, and 15 idle ATM cells. In order to minimize the delay and buffer size, the cells are grouped in 3 groups of 75 and 12 groups of 74 ATM cells containing PCM data.

The 15 idle cells will be inserted between each group of 75 or 74, bringing the total cell count to the required 1128. Refer to Section 4.1 and Section 5.2 for calculations and further discussion.

This idle cell insertion section of logic is used to generate the status signal InsertIdle, which forces the Unload side of the Transmitter to unload an idle ATM cell. This is accomplished by forcing the channel and buffer bits, of the Unload side memory addressing, high, see Section 8.3.3 and Section 9.6.3.

In order to keep track of the position in this idle ATM cell insertion pattern, two loadable counters are required. A loadable bidirectional 7 bit counter, Cell_Count, is used to keep track of the 74 or 75 ATM cells and a loadable bidirectional 4 bit counter, Group_Count, is used to keep track of the 15 groups of 74 or 75 ATM cells.

9.8.1 Cell Count Block

The Cell Count block is made up of a loadable bidirectional 7 bit counter, Cell_Count, which is clocked by C256k and hardwired to down count. The TC from the Cell_Count counter is used as the InsertIdle status signal.[18] When the InsertIdle status signal is asserted then the next unloaded ATM cell is an idle cell. The chip enable to the Cell_Count counter is controlled by the AND of StartToUnload and TUL_ChInc. The StartToUnload signal prevents the pattern counter section from starting until the Transmitter has buffered enough data to start unloading ATM cells. The TUL_ChInc signal is used because each increment of the TUL_Channel_Cnt counter indicates that a complete ATM cell has been unloaded.

The load signal for the Cell_Count counter is the logical AND of the Cell_Count counter TC output and CE input. Therefore the Cell_Count counter is loaded when the Cell_Count value is equal to zero and a CE occurs. The D(6:1) load inputs of the Cell_Count counter are hardwired to the value 100101. The D(0) value is determined by the state of the Group_Count counter, see Section 9.8.2. If the Group_Count counter value equals 0, 13 or 14, the D(0) bit is set high, loading the Cell_Count counter with the value of 75. Otherwise the D(0) bit is set low, loading the Cell_Count counter with the value of 74.

9.8.2 Group Count Block

The Group Count block is made up of a loadable bidirectional 4 bit counter, Group_Count, clocked by C256k and hardwired to down count. The count enable out (CEO) of the Cell_Count counter is used for the chip enable of the Group_Count counter, see Section 9.8.2.[19] The LOAD signal for the Group_Count counter is its own CEO output. The Group_Count counter is hardwired to load the value 14. Therefore, Group_Count down counts whenever the Cell_Count counter reaches zero, and the Group_Count counter is reloaded whenever the Group_Count value equals zero and a CE occurs. The output of the Group_Count counter is used to load the value of 75 into the Cell_Count counter for Group_Count values of 0, 13 and 14 and 74 for all other values of Group_Count. The value of 0 is chosen to take care of the initial start-up condition, and 14 and 13 are the next two following Group_Count values.[20] See Section 9.8.1 for more details.

---

18. The TC output for an UP/$\overline{\text{DOWN}}$ counter is logically equal to: $U(Q_0 * Q_1 * \ldots * Q_n) + \overline{U}(\overline{Q}_0 * \overline{Q}_1 * \ldots * \overline{Q}_n)$

19. The logic equation for CEO is (TC * CE).

20. Actually any three Group_Count values can be chosen, the value 0, 13 and 14 were chosen for conceptual simplicity.

9.9 Test Logic

In order to simplify bench testing of the ATM board, additional logic was added for the sole purpose for generating clocks and known test patterns.

9.9.1 CLKA and FMB Generator

The input to the Clock_Gen block is a 4.086 MHz clock, C4M. The Clock_Gen block then generates the CLKA and FMB system timing signals. The CLKA signal is generated by dividing the C4M signal by 2. This is implemented in a toggle flip flop. The FMB signal is generated using C4M and a 10 bit counter. The CEO output of the 10 bit counter is used as the FMB signal. The 10 bit counter is needed because there are 1024 cycles of C4M between each FMB assertion.[21] The Clock_Gen block is reset by the global RESET signal.

9.9.2 Pattern Generator

In order to simplify bench testing, a pattern generator block, Pattern_Gen, is used to generate a known input pattern to the Transmitter. This is useful because it provides a known pattern which can be compared to those obtained from simulation. The Pattern_Gen block circuit is shown in Figure 34, "Pattern Generator Block." This circuit generates 2 values for each time slot. The time slot values are shown in Table 13, "Pattern Generator Time Slot Values." The time slot values start with the First value and then alternate between the Second and the First values.

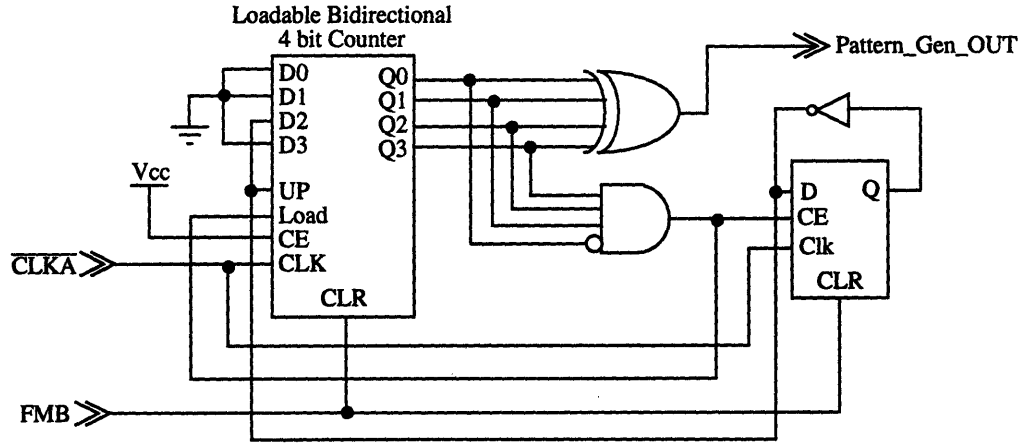


FIGURE 34. Pattern Generator Block

21. The number 1024 is obtained because there is 1 FMB every 2 T1 frames. Each T1 frame has 32 time slots. Each time slot is made up of 8 cycles of CLKA and each CLKA cycle is 2 $\overline{C4M}$ cycles. Therefore $2*32*8*2 = 1024 = 2^{10}$.

TABLE 13. Pattern Generator Time Slot Values

PCM Time Slot	First Value	Second Value
0	69	D9
1	97	69
2	65	97
3	A6	65
4	5D	A6
5	96	5D
6	99	96
7	76	99
8	5A	76
9	65	5A
10	D9	65
11	69	D9
12	97	69
13	65	97
14	A6	65
15	5D	A6
16	96	5D
17	99	96
18	76	99
19	5A	76
20	65	5A
21	D9	65
22	69	D9
23	97	69
24	65	97
25	A6	65
26	5D	A6
27	96	5D
28	99	96
29	76	99
30	5A	76
31	65	5A

Chapter 10 Receiver Architecture

10.1 Principle of Operation

The Receiver Load side performs the SAR functionality for interfacing between the ATM data protocol and the TDM data protocol. This interface includes the cell delineation for ATM cells, the mapping of ATM payloads into the appropriate SRAM buffer and synchronization to the PCM highways for unloading data from the SRAM into the appropriate time slots. All 32 time slots in the PCM highway will be filled with the appropriate payload data or in the case of inactive time slots, will be replaced by an all 1's pattern.

The basic architecture for the Receiver is shown in Figure 35, "Receiver Architecture Diagram." Management of the Receiver buffers is based on a simple rule:

> For a particular channel, the most recent received valid ATM cell will be used.

This means that if an ATM cell arrives late or is missing, then data will be unloaded based on buffered data. If all the data buffers have been used, then the latest received valid ATM cell's data will be reused as the replacement for the late or missing ATM cell's payload.[21]

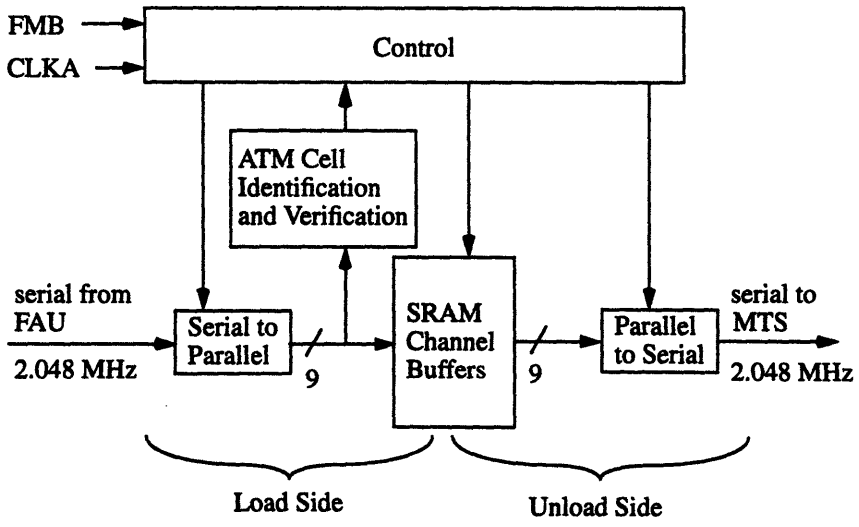


FIGURE 35. Receiver Architecture Diagram

21. While idle ATM cells are technically valid ATM cells, from now on, the term "valid ATM cells" will specifically refer only to ATM cells carrying channel data.

The SN byte of an ATM cell will be stored as part of the status. However, no error correction, such as out of order cell detection, will be implemented based on the ATM cell SN.

While it is possible to load in an ATM cell and concurrently begin unloading the same cell's payload, in order to simplify timing issues, ATM cells and SRAM buffers will be treated as an entire unit. This means that data cannot be unloaded from a buffer until the ATM cell has been completely loaded into the buffer.

10.2 System Architecture

The Receiver also uses a 256KHz clock (C256k) based on the same reasoning as the Transmitter, see Section 8.2. The SRAM buffer will contain 16 buffers for each time slot, which are organized as a linked list structure. This requires a 32kx9 SRAM. Of the 16 buffers, 15 are used for ATM cell data and 1 is reserved for channel status, see Section 10.3.4. In order to maintain each channel independently to the others, several status values are required for each time slot. These status values will keep track of the head and tail buffer values of the link list and the index of the current value being addressed within a buffer.

Therefore, the basic Receiver operations required by the Receiver are shown in Table 14, "Basic Receiver Functions."

TABLE 14. Basic Receiver Functions

Load Side	Unload Side
Get Unload Status	Get Unload Status
Get Load Status	Get Load Status
Write Data	Get Index Status
Write Load Status	Get Data
	Write Index Status
	Write Unload Status

The functions listed in Table 14 show a maximum of 10 possible memory accesses for loading and unloading data for a time slot. Because not all operations occur at every cycle, the maximum number of memory operations is actually 8. These 8 are composed of 2 of the memory operations for the Load side and all 6 memory operations from the Unload side, see Section 11.3. The data paths used to support the Receiver functions are shown in Figure 36, "Receiver Load Side Data Path Block Diagram," and Figure 37, "Receiver Unload Side Data Path Block Diagram." The SRAM, EPROM and multiplexers partitioned off by the dotted line are components that are shared by both the Load and Unload sides, and are repeated in Figure 36 and Figure 37.

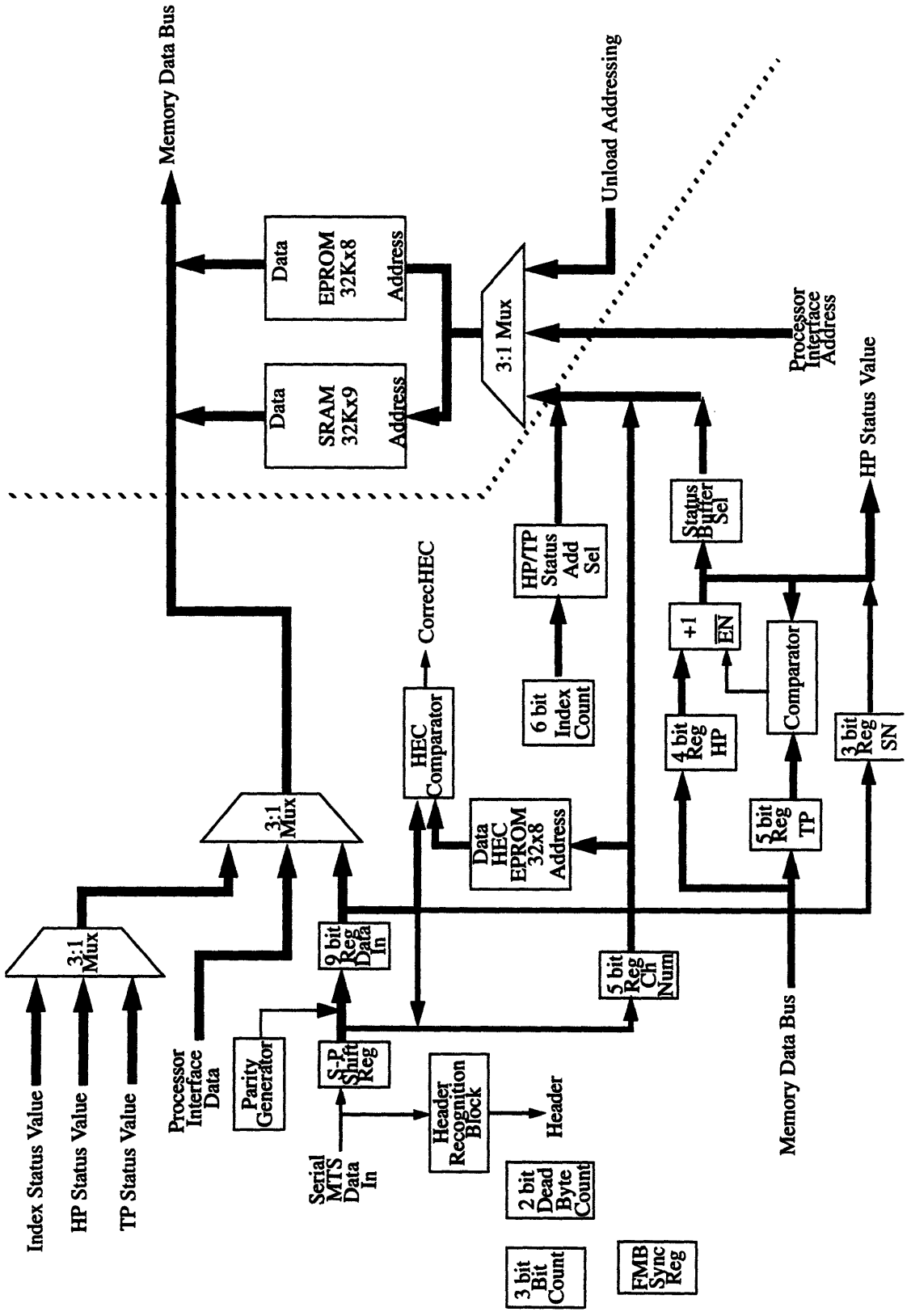


FIGURE 36. Receiver Load Side Data Path Block Diagram

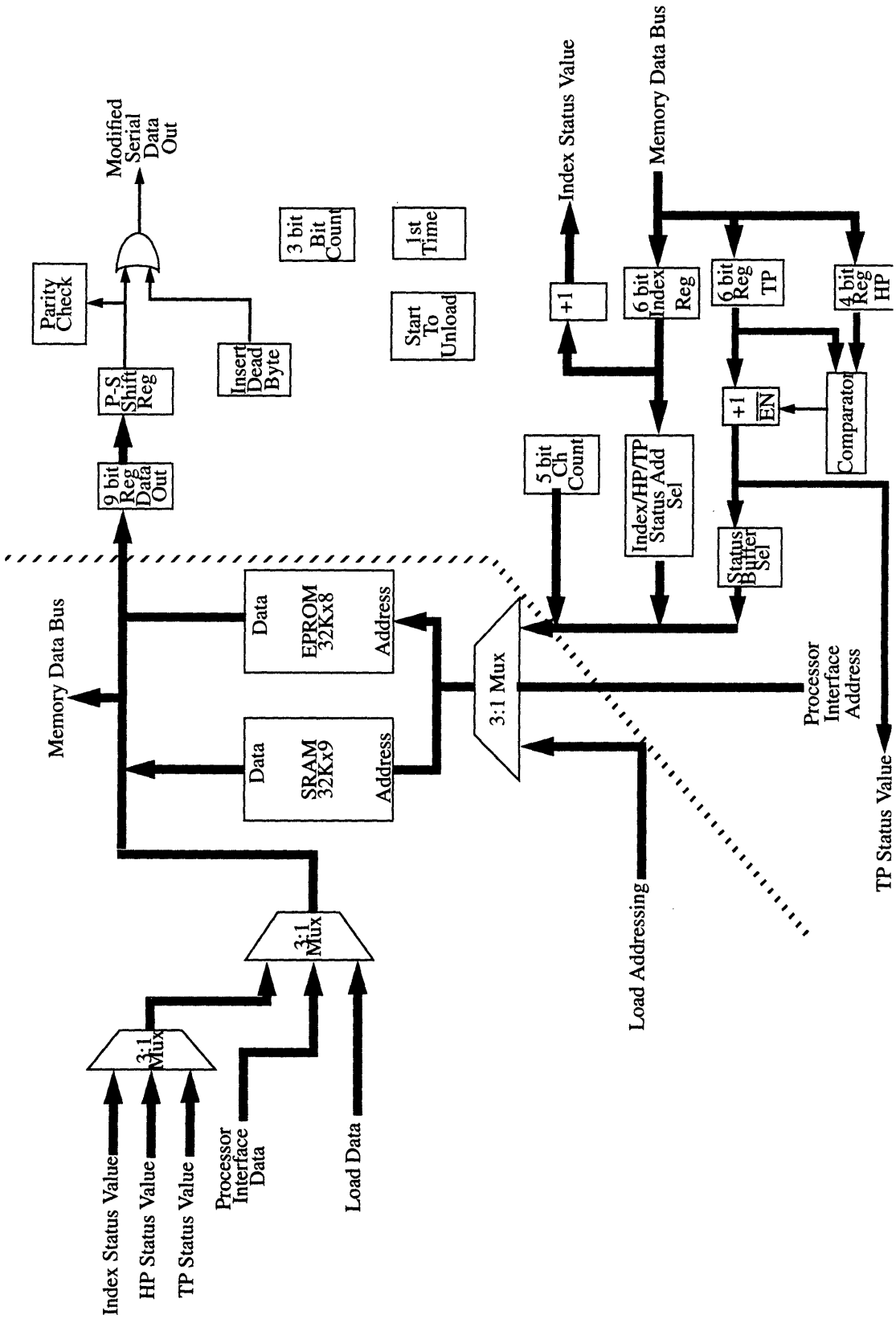


FIGURE 37. Receiver Unload Side Data Path Block Diagram

10.3 Receiver Memory

The Receiver memory is composed of a SRAM and an EPROM.

10.3.1 Data Buffer

The Receiver data buffer is maintained in a 32k x 9 SRAM. The SRAM is used to store the ATM payload and the status information after the initialization process. The extra bit in the data word is used for parity. The chip enable is connected to the Receiver NSRAMCS control signal and the read/write input is connected to the Receiver NSRAMR/W control signal. The output enable of the SRAM is connected to the inverted NSRAMR/W signal.

10.3.2 Initialization Data

The Receiver initialization data is maintained in a 32k x 8 EPROM. The EPROM specifically is used to store initialization information for the status values. The chip and output enables of the EPROM are tied together and connected to the control line NEPROMCS.

The programmed content of the Receiver EPROM is shown in Appendix E.

10.3.3 HEC Values

The HEC values for the 32 allowable VPI/VCI addresses are stored within another EPROM. The addressing for the HEC EPROM is taken from the Receiver Load side Channel register, see Section 11.7.2.2. The output of the HEC EPROM is used as the correct HEC value input to the Header Comparison block, see Section 11.2.2.3. The HEC EPROM is always chip and output enabled.

The programmed content of the Receiver HEC EPROM is shown in Appendix F.

10.3.4 Memory Partitioning

The memory addressing bits are broken down into fields as shown in Figure 38, "Receiver Memory Address Bits."

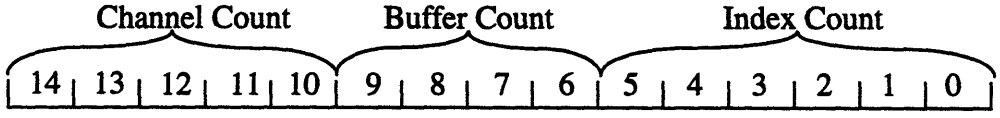


FIGURE 38. Receiver Memory Address Bits

Both the Load and Unload side memory addressing are maintained in three types of values, a Channel Counter value, a Buffer Counter value and an Index Counter value. The Channel Counter value is a 5 bit value used to keep track of the channels. The Buffer Counter value is a 4 bit value used to keep track of the 16 buffer locations. The Index Counter value is a 6 bit value used to indicate the byte location within an ATM payload.

The memory is therefore partitioned in the manner depicted in Figure 39, "Receiver Memory Partitioning." The memory consists of two types of buffers. Buffers 0 through 14 are data buffers and buffer 15 is used for status information.

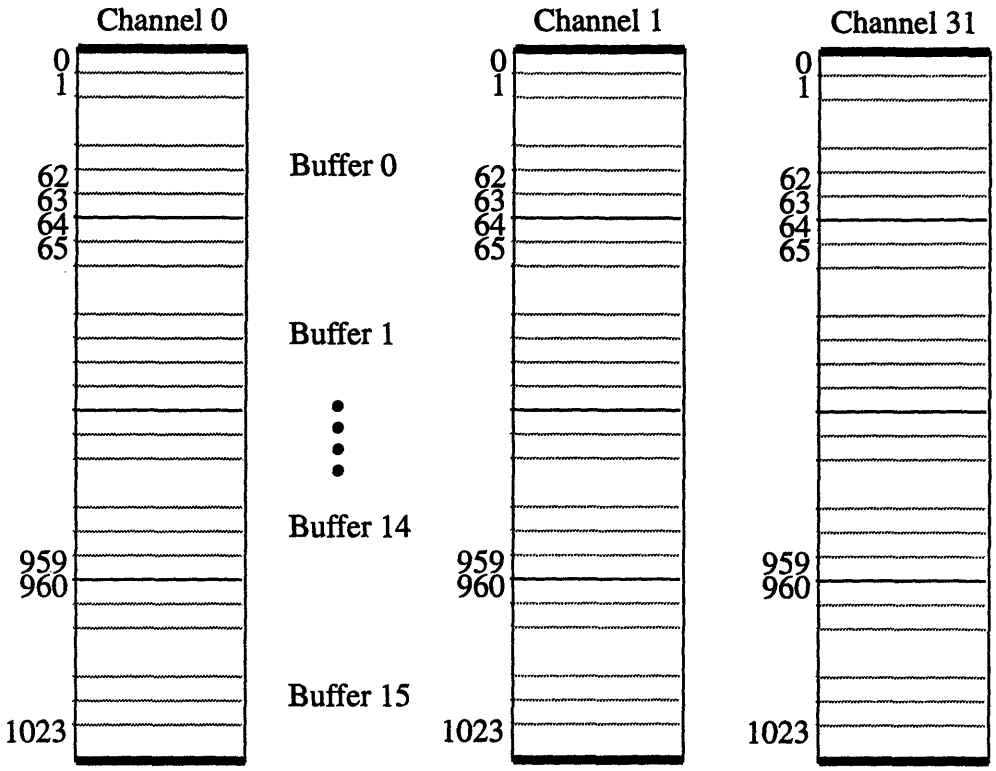


FIGURE 39. Receiver Memory Partitioning

10.3.5 Memory Buffer Description

The data and status buffer formats are shown in Figure 40, "Receiver Buffer Partitioning." The buffers are organized in this fashion to facilitate the loading and unloading of ATM cells as well as allowing for future modifications.

The buffers are defined as 64 byte blocks because of the simplicity of a 2^k boundary. The data buffers are used to contain the ATM cell payload, excluding the SN, and therefore only locations 0 to 46 are used. The status buffer is used to contain channel status information. The status buffer locations 61, 62 and 63 are reserved for the Index, Unload and Load status fields respectively, see Section 10.3.6 and Section 10.3.7

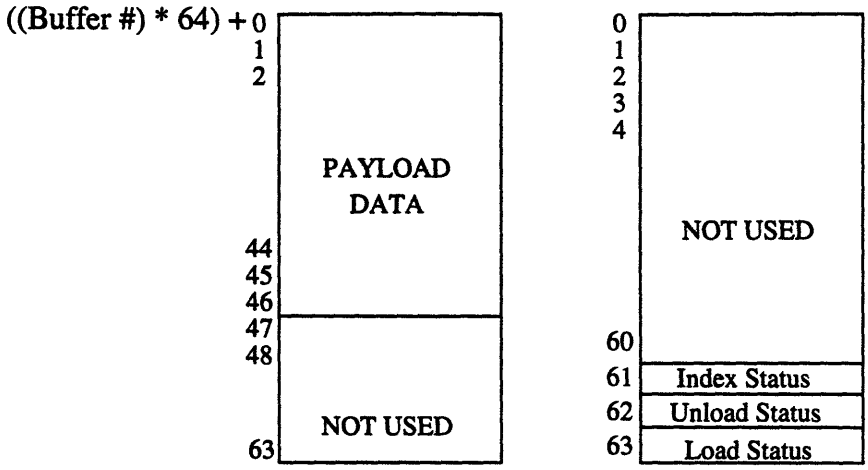


FIGURE 40. Receiver Buffer Partitioning

10.3.6 Buffer Pointers

A channel's buffers are managed as a linked list, independent of other channel buffers. Each channel has its own header pointer (HP), tail pointer (TP), Index value and channel status information.

The HP status is a 4 bit value that points to the last buffer which has had a complete ATM payload loaded into it. The TP status is a 4 bit value that points to the buffer that is currently being used to unload data. While both the Load and Unload side require access to the HP and TP status values, only the Load side of the Receiver updates the HP value and only the Unload side of the Receiver updates the TP value. This eliminates any problems concerning data validity or the need for a token ring type of status data access. Initial values for the HP and TP values are stored in the EPROM. The initial value for the both the HP and the TP is zero.

The Index status is used by the Unload side of the Receiver to keep track of the byte location within a buffer from which data is being unloaded. The Index status is a 6 bit number and is initialized to zero upon connection establishment. The Index status value can range from 0 to 46.

10.3.7 Status Values

Other channel status fields are Active_Channel, UnderRun, OverRun, Init and SN. The Active_Channel status is set high to indicate that the channel is active, if Active_Channel is low, then an all 1's pattern will be outputted as data during the channel's time slot.

The UnderRun and the OverRun status values are both initialized to zero upon connection establishment. UnderRun is set high when no new data is available to be unloaded, this condition is also known as starvation. In the case of an UnderRun situation, the last valid data buffer is reused. The UnderRun signal is generated in the Receiver Load side Buffer Addressing block, see Section 11.7.2.3. OverRun is set high when there are no empty data buffers and a new ATM cell arrives and its payload needs to be written to the SRAM buffer. In the case of an OverRun occurrence, the new ATM payload is lost. The OverRun signal is generated in the Receiver Unload Buffer Addressing block, see Section 11.7.3.3.

The Init field is actually the registered StartToUnload signal, see Section 11.4.1. This signal is used to indicate when initialization has been completed and data is ready to be unloaded. This bit should eventually become a field which is set by higher layer management.

The SN field is used to contain the sequence number of the ATM cell that has just been received. The organization of these status bits and the TP and HP status values is shown in Figure 41, "Channel Status Information Addressing and Organization." The status information is organized to allow for the fewest possible memory operations for accessing and updating status values.

By adjusting the delta between the HP and TP values, the amount of buffering can be varied. This delta value can be user set by dip switches connected to the Buffer_Delta_Value(3:0) inputs. While larger delta values reduce the possibility of UnderRun, each buffer adds approximately 6ms of delay.[22] See Section 11.4.1 for further details.

Addressing			Bit Number								
Channel Value	Buffer Value	Index Value	7	6	5	4	3	2	1	0	
xxxxx	1111	111101	Reserved/ Unused		Index Status Value						Index Status
xxxxx	1111	111110	Act_ Ch	R/U	Under Run	Init	Tail Pointer Status Value				Unload Status
xxxxx	1111	111111	Over Run	Sequence Number (SN)			Head Pointer Status Value				Load Status

R/U = Reserved/Unused
xxxxx = values 0 to 31

FIGURE 41. Channel Status Information Addressing and Organization

10.3.8 Memory Selection

This is the supporting hardware that is used to mask the selection of either the data buffer SRAM or initialization EPROM as the active memory component. This logic is implemented external to the control FSMs in order to simplify future modifications that may remove the need for the EPROM. The memory selection logic is required to provide the initial start up information for the status values. The initialization time is different for the Load and Unload sides. This is because the Load side must finish it's initialization and fill it's buffers before the Unload side can start unloading data. The determination for SRAM or EPROM selection is based on the Receiver FSM states and the states of RL_NFirstTime and RUL_NFirstTime.

Only the FSM states that access status values during initialization need to access the EPROM, therefore the basic SRAM and EPROM selection logic is shown in Figure 42, "EPROM Selection Logic."

22. End-to-end transmission and reception delay times become an issue around 25ms and echo cancellation circuitry becomes necessary.

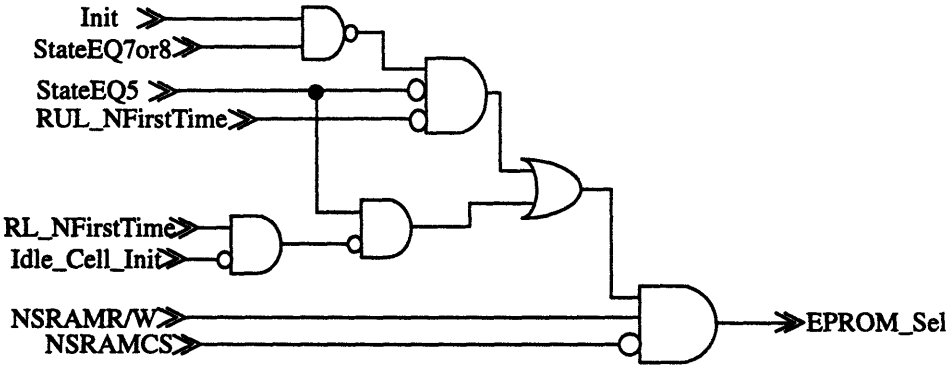


FIGURE 42. EPROM Selection Logic

The logic equation for EPROM_Sel is as follows:

$$\text{EPROM\_Sel} = \overline{\text{NSRAMCS}} * \text{NSRAMR\_W} * ((\text{StateEQ5} * (\overline{\text{RL\_NFirstTime}} + \text{Idle\_Cell\_Init})) + (\overline{\text{RUL\_NFirstTime}} * \overline{\text{StateEQ5}} * \overline{(\text{StateEQ7or8} * \text{Init})}))$$

When EPROM_Sel is high, then the EPROM is selected as the active memory, otherwise, the SRAM is the active memory device.

Chapter 11 Receiver Block Descriptions

11.1 System Synchronization

The Receiver side system synchronization with external signals is implemented identically as the Transmitter side system synchronization, see Section 9.1.

11.2 Cell Delineation

The Cell Delineation portion of the Receiver is based on the VPI/VCI values specified in Section 6. The Cell_Delineation FSM is used to perform the ATM cell delineation and provide the necessary status signals to indicate the start of an ATM payload.

11.2.1 Cell Delineation FSM

The ATM header format is shown in Figure 43, "ATM Cell Header Structure for UNI."

The basic procedure used in the cell delineation is as follows:

1. Search for 00000000000000000000 pattern.
2. Since everything is byte aligned, the value actually found is 0000000000000000000000ab. The a bit must be 0 for idle ATM cells and 1 for active ATM cells. The b bit is actually the MSB of the header value which contains the time slot information.
3. At the reception of the next valid byte, the 4 MSBs, c, d, e and f, are latched and used as to generate a possible valid VPI/VCI value. The bcdef value is also the time slot value.
4. The 5 bits, b, c, d, e and f, are then used to address a HEC EPROM, see Section 10.3.3, which contains the HEC values for the 000000000000000000000000abcdef VPI/VCI value.
5. The HEC value found from the HEC EPROM is then compared with the next valid received byte which is suppose to be the ATM cell's HEC byte. If the two values are identical and the a bit is the appropriate value, then a valid header has been found. If they differ, then a header error is flagged and the cell is dropped. No header error correction procedure is implemented.

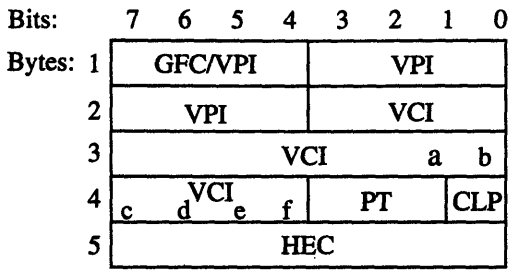


FIGURE 43. ATM Cell Header Structure for UNI

The Cell_Delineation FSM takes in the following status signals RL_DeadByteEQ0, Header, CorrectHEC and RL_IndexEQ46 and generates the ChNumRegCE, the SNRegCE and the Payload control signals. The FSM bubble diagram is shown in Figure 44, "Cell Delineation Control FSM Bubble Diagram."

The RL_DeadByteEQ0 status signal indicates that the current time slot is inactive, see Section 11.5.4. The assertion of the Header signal is used to indicate that the pattern of 22 consecutive zeros have been located. The last 2 bits and the next byte are then the possible channel number that should be used to address the memory on the Load side of the Receiver, see Section 11.7.2.2. The CorrectHEC signal is used to indicate that a correct HEC match has been accomplished, see Section 11.2.2.3.

The ChNumRegCE signal is used as the CE for the ChNumReg register which is used to hold the channel number, bits b, c, d, e, and f, see Section 11.7.2.2. Because of timing issues, the ChNumRegCE control signal is generated as an unregistered output. This is based on the fact that the FSM is using C256k as the clock. The other reason is to be able to handle a long series of consecutive zeros that occur before an ATM cell. A possible scenario is the following:

1. Initially the FSM is in State 0.
2. A series of 8 zeros occur, that are unrelated to the VPI/VCI header bits, followed by an ATM cell. The Header signal is asserted after the second byte of the ATM cell's header. This causes the FSM to transition to state 1. Having guessed an incorrect starting point for cell delineation, an incorrect channel number will be latched into the ChNumReg register.
3. The FSM then transitions to state 2 after the assumed HEC value is latched into the RL_Data_In register. However, because the bytes contained in the ChNumReg register and the data from the FAU serial to parallel convertor are neither the correct channel value nor the correct HEC value, then the CorrectHEC signal will not be asserted. At this point, if the FSM returns to state 0, then the correct channel number would be missed and the entire ATM cell would be lost.

   Therefore, to avoid this situation, state 2 has a conditional statement for the case where there is an incorrect HEC, but the Header signal is still asserted. This indicates that the initial header guess was incorrect and the current byte being shifted in may actually contain the correct channel number value.

   If the ChNumRegCE signal was registered, then there would be a byte delay between the detection of this condition and the actual latching of the channel number. Similar situations occur for other numbers of zeros occurring before the actual header of an ATM cell. It is for this reason, that the ChNumRegCE signal is based on combinational logic and the HEC comparison is based on the data output of the FAU serial to parallel convertor and not based on the data from the RL_Data_In register.

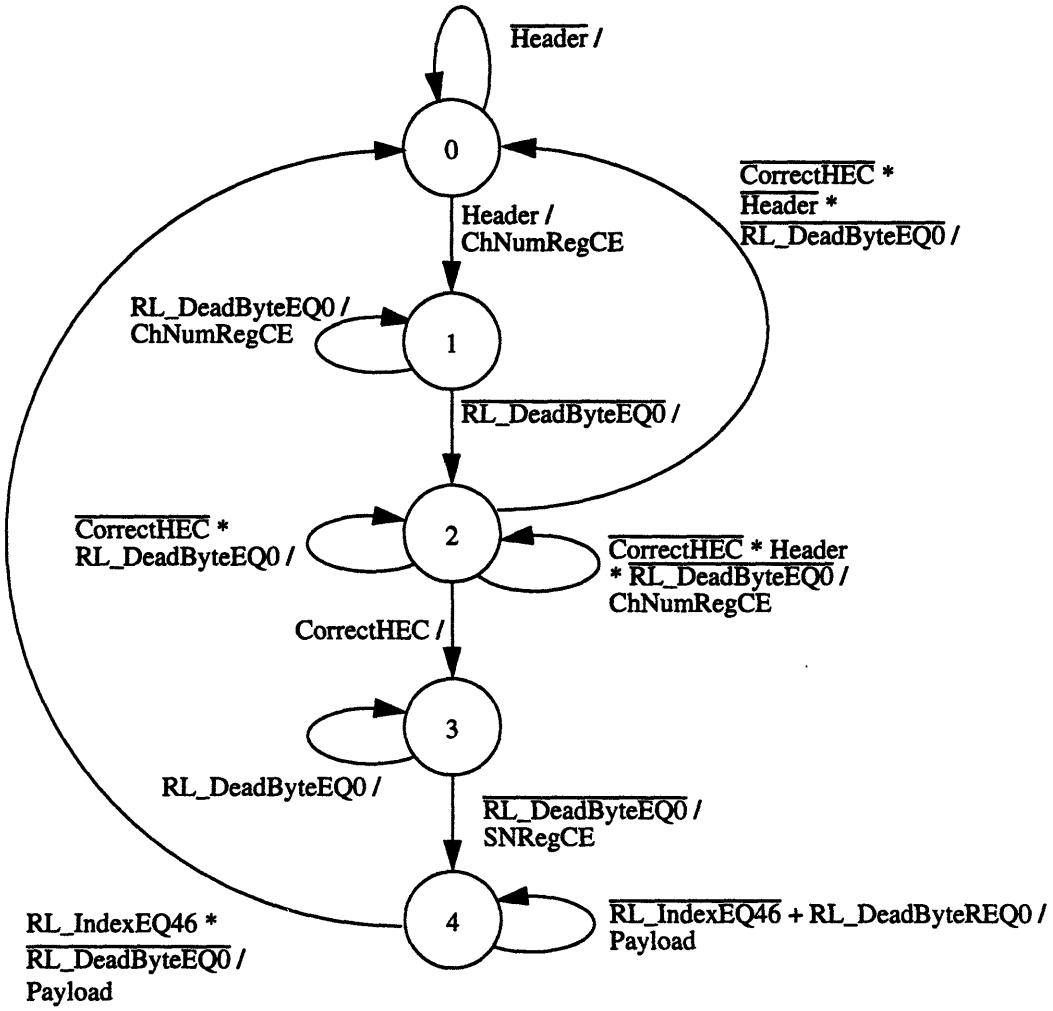


FIGURE 44. Cell Delineation Control FSM Bubble Diagram

Table 15, "Cell Delineation FSM Signal Description," list and described the signals associated with the Cell Delineation FSM.

TABLE 15. Cell Delineation FSM Signal Description

Signal Name	Signal Function
Header	Used to indicate the reception of the 20 zeroes pattern, refer to Section 11.2.2.1.
ChNumRegCE	Chip enable signal for the Channel Number register, refer to Section 11.2.2.2.
RL_Dead_ByteEQ0	Status signal used to indicate that the current time slot data stored in the RL_Data_In register is an inactive time slot and should not be stored, refer to Section 11.5.4.

TABLE 15. Cell Delineation FSM Signal Description

Signal Name	Signal Function
CorrectHEC	Status signal used to indicate that a correct HEC value was located, refer to Section 11.2.2.3.
SNRegCE	Chip enable signal for the SN register, refer to Section 11.6.2.3.1.
Payload	Status signal used to indicate that the data in the RL_Data_In register is part of an ATM payload, refer to Section 11.3.
RL_IndexEQ46	Status signal used to indicate the end of an ATM cell, refer to Section 11.7.2.1.

The assertion of SNRegCE and Payload are delayed by one clock cycle because these signals are used in blocks that use data from the RL_Data_In register and not data directly from the FAU serial to parallel converter. The SNRegCE signal is used to latch the SN bits, see Section 11.6.2.3.1, and the Payload signal is used to indicate that the data presented by the RL_Data_In register is part of the ATM cell's payload.

The Cell_Delineation FSM only looks at the Header signal after an entire ATM cell has been collected. This is due to the use of the RL_IndexEQ46 status signal. The signal RL_IndexEQ46 is generated from the Receiver Load Index Addressing Counter block, and is used to indicate the end of an ATM cell, after 47 bytes of payload data have been stored, see Section 11.7.2.1. This is not a necessary function, but is useful in preventing the Cell_Delineation FSM from indicating a new VPI/VCI occurrence that may appear within the payload of the ATM cell.

11.2.2 Cell Delineation Data Paths

11.2.2.1 Header Recognition Block

The Header Recognition block is used as part of the cell delineation function. Because of the assignment of VPI/VCI values, the Header Recognition block searches for the pattern 00000000000000000000. This pattern of 20 zeros asserts the status signal Header indicating the possible start of an ATM header. After the first pattern of 20 zeros is detected, the Header signal will stay asserted for any data satisfying the xxxx00xx pattern, where "x" represents a "don't care" value. The xxxx00xx pattern is necessary in cases where the Header signal is already asserted, but the start of an ATM cell has not been located. In this case the Cell_Delineation FSM is in state 2. The use of the xxxx00xx pattern prevents the 4th byte of the ATM header from deasserting the Header signal before the channel number can be latched.

The 20 zeros essentially span the first three header bytes. Each of these header bytes are sectioned into bits 0 to 5 and bits 6 and 7. This partitioning of the bytes is due to how the time slot value is encoded into the last 6 bits of the VPI/VCI value and the relationship of these last 6 bits with the physical byte boundaries. The Zero Detection block data paths are shown in Figure 45, "Zero Detection Portion of Header Recognition Block."

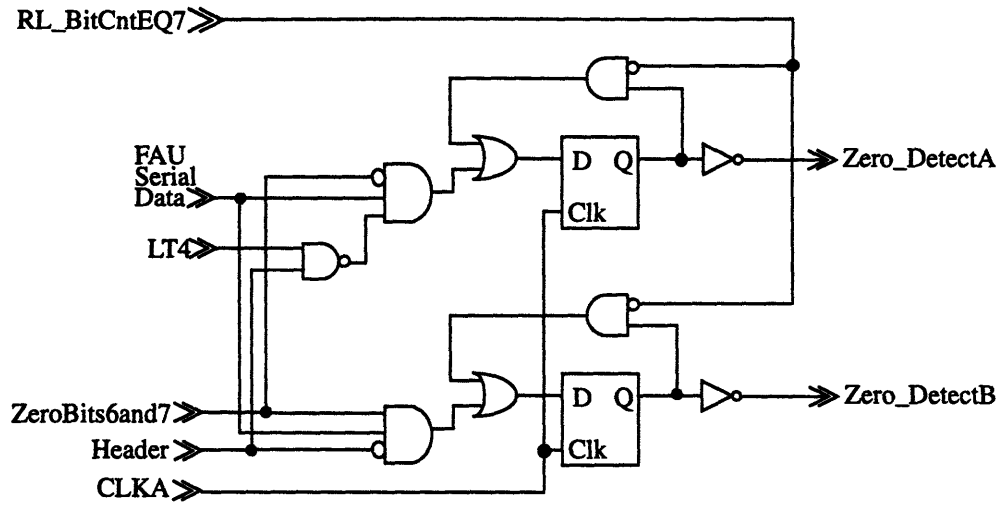


FIGURE 45. Zero Detection Portion of Header Recognition Block

The Zero Detection block serially shifts in the PCM highway data from the FAU and accumulates the data through the use of an OR gate and feedback from the register's output. The two signals generated, Zero_DetectA and Zero_DetectB, are respectively set high when the data input satisfies the required data pattern for bits 0 to 5 and bits 6 and 7.

The AND gate in the feedback path is used to clear the old Zero_Detect value when a new byte is started. The combinational logic in front of the OR gate is used to force certain bits to zero. The AND gate in the Zero_DetectA portion is used to force bits 6 and 7 to zero, so that they do not effect the zero detection of bits 0 through 5. The NAND gate is used so that once the Header signal is asserted, then bits 0 through 3 are all forced low. This is necessary to obtain the xxxx00 portion of the xxxx00xx pattern. The AND gate in the Zero_DetectB portion is used to force bits 0 through 5 to zero, so that the zero detection of bits 6 and 7 are not effected. The use of the Header input is to obtain the final xx portion of the xxxx00xx pattern. The timing is shown in Figure 46, "Zero Detection Timing."

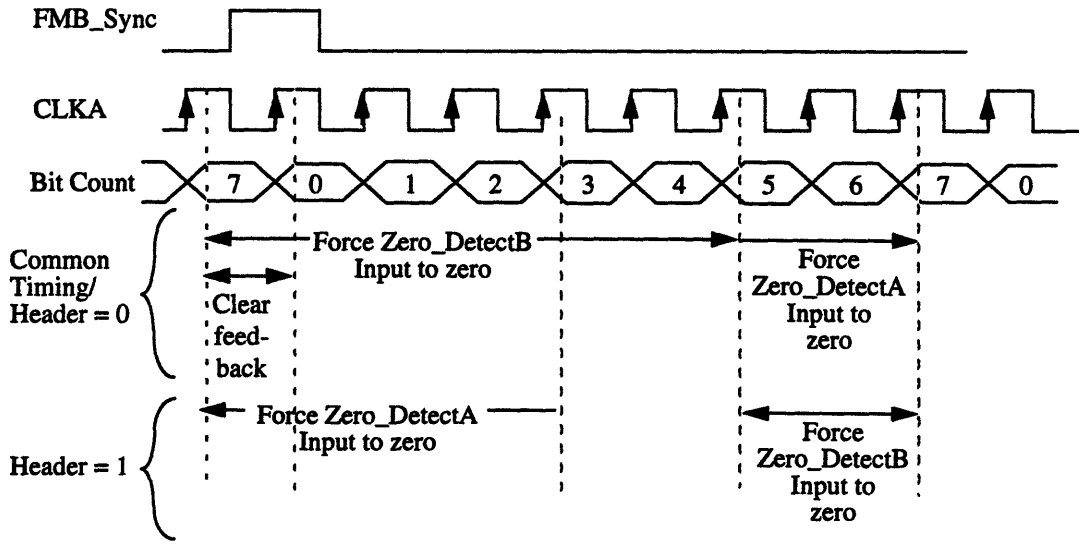


FIGURE 46. Zero Detection Timing

Because the zeros in the VPI/VCI value spans 2 1/2 bytes, or equivalently 20 bits, it is necessary to keep a history of the previous zero occurrences. This is done through the use of two registers. The data paths are shown in Figure 47, "Header Signal Generation."

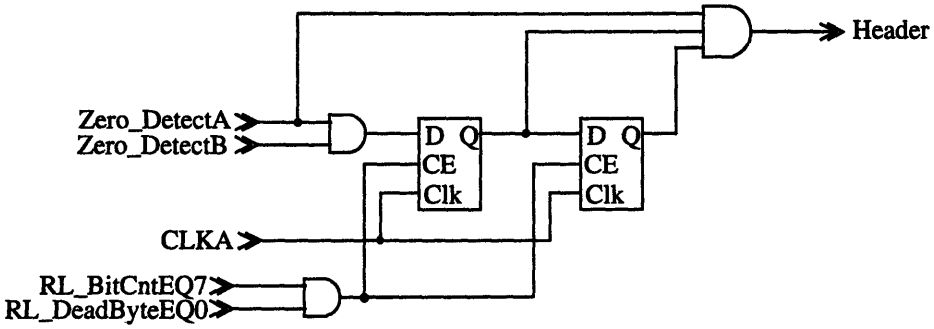


FIGURE 47. Header Signal Generation

The zero status stored in the registers are the AND of Zero_DetectA and Zero_DetectB. This determines whether the overall byte fit the required pattern. The status registers are clocked using CLKA, and are chip enabled using the AND of RL_BitCntEQ7 and RL_DeadByteEQ0. The AND of the RL_BitCntEQ7 and RL_DeadByteEQ0 allows data to be register at the byte boundary of an active time slot, see Section 11.5.1 and Section 11.5.4 for more information on the respective signals. Therefore the registers are enabled at each byte boundary of the active time slots. The Header signal is then generated by ANDing Zero_DetectA and the outputs from the two status registers, which provides the coverage of the 20 bit pattern.

11.2.2.2 Channel Number Register

This register is used to store the relevant portion of the VPI/VCI address which contains the channel number in which the ATM cell is to be stored. The channel number register, ChNumReg, is a 6 bit register clocked by CLKA and is chip enabled by the AND of RL_BitCntEQ7 and the control signal ChNumRegCE. The ChNumRegCE signal is generated from the Cell_Delineation FSM, see Section 11.2.1. The data path is shown in Figure 48, "Channel Number Register Block Data Paths."

The channel number value is made up of bits 1 and 0 of byte 3 of the ATM header and bits 7 through 4 of byte 4 of the ATM header. Since these cross a byte boundary, it is necessary to obtain the byte 3 bits from the output of the RL_Data_In register and the bits from byte 4 from the output of the FAU serial to parallel shift register.The ChNumReg register bits are NORed together to generate the Idle_Cell signal and ANDed together to generate the RL_ChNumEQ31 signal.[23] The Idle_Cell signal and MSB of the channel number are then ORed together to produce the OK_Header signal. The MSB of the ChNumReg register should be high for a correct active ATLM cell's VPI/VCI value. The lower 5 bits are then used for the channel value. The OK_Header signal is used in by the Header Error Check Comparator, see Section 11.2.2.3.

See Section 6.2 and Section 11.2.1 for information on VPI/VCI values and for more details.

23. An idle ATM cell has a VPI/VCI value of 0. Channel 31 is the signaling channel for the PBX.

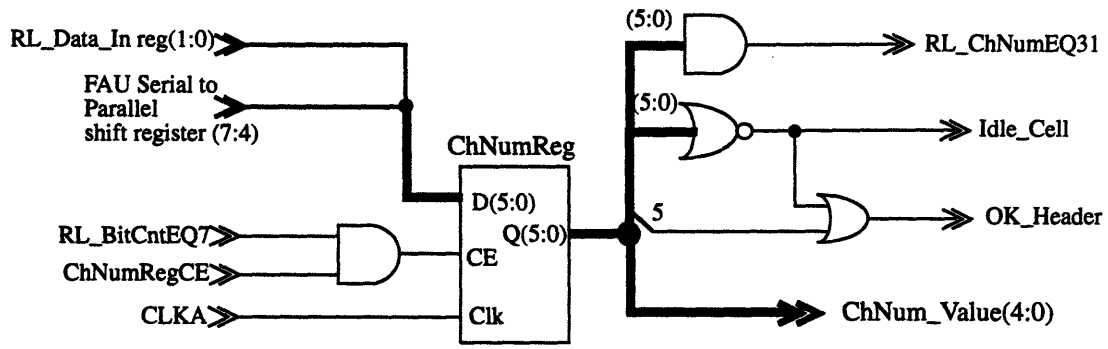


FIGURE 48. Channel Number Register Block Data Paths

11.2.2.3 Header Error Check (HEC) Comparator

An eight bit comparator, HEC_Comp, is used to determine whether a correct HEC is found. The comparator takes the HEC value from the HEC EPROM using the VPI/VCI value stored in the ChNumReg register and compares it with the value shifted in by the FAU serial to parallel shift register. The equal output of HEC_Comp is then ANDed with the OK_Header signal to generate the final CorrectHEC status signal. The OK_Header signal is used as a final check to make sure that the correct VPI/VCI value has been stored, see Section 11.2.2.2. The timing for the assertion of CorrectHEC is shown in Figure 49, "Signal Timing for Assertion of CorrectHEC."

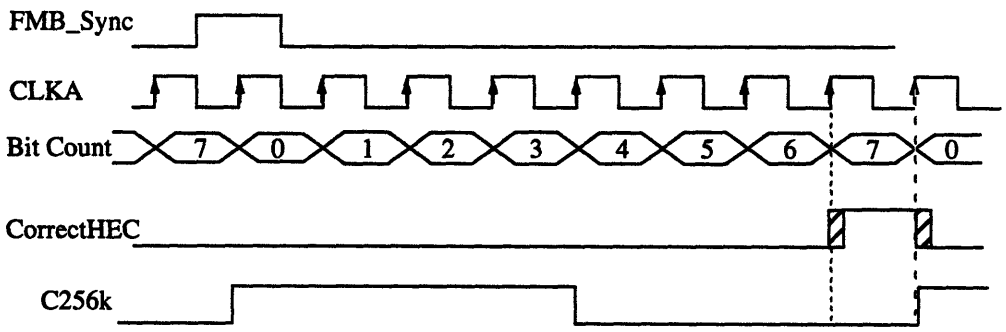


FIGURE 49. Signal Timing for Assertion of CorrectHEC

11.3 Control Finite State Machine (FSM)

The Receiver Side FSM controls all the data and addressing pathways associated with the SRAM and EPROM memory elements. Because the load and unload timing is completely synchronous and in order to simplify the control FSM, the Load side and Unload side controls are combined into a single Receiver Side Control FSM. The clock rate used for the Receiver Side FSM is dependant on the maximum number of memory operations required for the loading and unloading of a single time slot. The number of memory access can be reduced to a minimum of 8. This is done by appropriately grouping the status values, see Figure 41, "Channel Status Information Addressing and Organization."

Because the Load and Unload sides are essentially operating independently to each other, to avoid data validity questions, both sides must have access of their own copies of the Load and Unload status

values. However, only the Load side is allowed to update the Load status value and only the Unload side is allowed to update the Unload and Index status values. The bubble flow diagram for the Receiver Side FSM is shown in Figure 50, "Receiver Control FSM Bubble Flow Diagram." State 0 is the initial start-up state, states 1, 2, 3, 7, 8 and 9 are Unload related states and states 4, 5 and 6 are Load related states.

Figure 50 shows the various conditional branches and the type of action to be taken without specifying the signals asserted to perform each task. The signals asserted for each task differs for the Load and Unload sides and are shown in Table 16, "Load Side Functions," and Table 17, "Unload Side Functions." The control signal descriptions are listed in Table 18, "Receiver Control Signal Description."

It should be noted that all of the control signals used are active high, except the NSRAMCS and the NSRAMR_W signal which are active low. All signals are registered outputs.

The Unload states are split by the Load states because of timing considerations. Both the Unload status value and the Index status value contain information required by state 5. If the state flow was to go from state 2 directly to state 5, then data would not be ready by state 5. By placing the Load states between states 2 and 5, any unnecessary memory operations are removed.

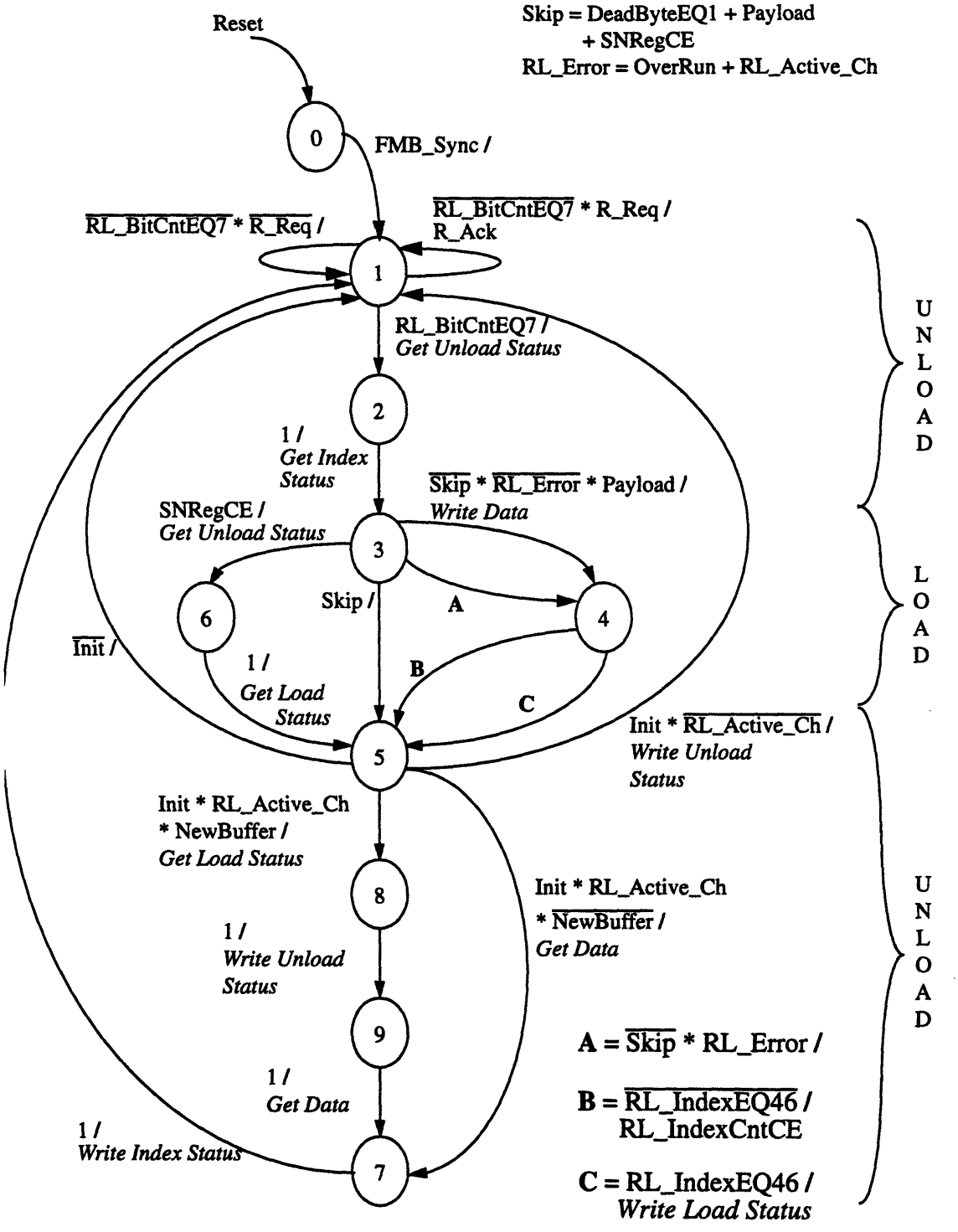


FIGURE 50. Receiver Control FSM Bubble Flow Diagram

TABLE 16. Load Side Functions

Task	Signals Asserted
Get Unload Status	R_IndexSel_S0 RL_TPRegCE SRAM_AddSel_S0 NSRAMCS SRAM Read
Get Load Status	R_IndexSel_S1 RL_HPRegCE SRAM_AddSel_S0 NSRAMCS SRAM Read
Write Data	SRAM_AddSel_S0 SRAM_Data_In_S1 NSRAMCS SRAM Write
Write Load Status	R_IndexSel_S1 SRAM_AddSel_S0 SRAM_Data_In_S0 NSRAMCS SRAM Write

TABLE 17. Unload Side Functions

Task	Signals Asserted
Get Unload Status	R_IndexSel_S0 RUL_TPRegCE SRAM_AddSel_S1 NSRAMCS SRAM Read
Get Load Status	R_IndexSel_S1 RUL_HPRegCE SRAM_AddSel_S1 NSRAMCS SRAM Read
Get Index Status	RUL_IndexRegCE SRAM_AddSel_S1 NSRAMCS SRAM Read
Get Data	R_IndexSel_S0 R_IndexSel_S1 R_DataRegOutCE SRAM_AddSel_S1 NSRAMCS SRAM Read
Write Index Status	SRAM_AddSel_S1 SRAM_Data_In_S0 NSRAMCS SRAM Write

TABLE 17. Unload Side Functions

Task	Signals Asserted
Write Unload Status	R_IndexSel_S0 SRAM_Add_Sel_S1 SRAM_Data_In_S0 NSRAMCS SRAM Write

TABLE 18. Receiver Control Signal Description

Signal Name	Signal Function
R_Ack	Acknowledge signal from the Receiver, used to indicate that memory data and address buses are allocated for processor interface usage, see Section 11.6.2.6.
R_IndexSel_S0, R_IndexSel_S1	Used in selecting Index values for memory addressing, see Section 11.3.1, Section 11.6.1.1, Section 11.7.2.1 and Section 11.7.3.2.
RL_TPRegCE	Receiver Load side TP register chip enable, see Section 11.7.2.3.
SRAM_AddSel_S0, SRAM_AddSel_S1	Input source selection signals for Receiver Addressing Multiplexer, see Section 11.7.1.
NSRAMCS	Negative asserted Receiver SRAM chip select signal, see Section 10.3.1.
NSRAMR_W	Negative asserted Receiver SRAM read/write signal, see Section 10.3.1.
RL_HPRegCE	Receiver Load side HP register chip enable, see Section 11.7.2.3.
SRAM_Data_In_S0, SRAM_Data_In_S1	Input source selection signals for Receiver SRAM Memory Data Bus Multiplexer, see Section 11.6.1.2.
RUL_TPRegCE	Receiver Unload side TP register chip enable, see Section 11.7.3.3.
RUL_IndexRegCE	Receiver Unload side Index register chip enable, see Section 11.7.3.2.
R_DataRegOutCE	Receiver Unload Data register chip enable, see Section 11.6.2.5.

11.3.1 Status Buffer Address Force Logic

The two control lines RL_StatusBuffSel and RUL_StatusBuffSel, see Section 11.7.2.3 and Section 11.7.3.3 respectively, are used to force the buffer value to all 1's in order to access status values, see Section 10.3.6. The two signals are generated by the decoding the R_IndexSel_S0 and the R_IndexSel_S1 control signals based on the logic equations shown below:

$$RL\_StatusBuffSel = (R\_IndexSel\_S0 \oplus R\_IndexSel\_S1)$$

$$RUL\_StatusBuffSel = (RL\_StatusBuffSel + (\overline{R\_IndexSelS0} * \overline{R\_IndexSelS1}))$$

Therefore when the R_IndexSel_S0 and R_IndexSel_S1 select the Unload status or the Load status Index address values, the RL_StatusBuffSel signal is asserted so that the Load addressing status Buffer value is selected. When the R_IndexSel_S0 and R_IndexSel_S1 select the Index status, the Unload status or the Load status Index values, the RUL_StatusBuffSel signal is asserted so that the Unload addressing status Buffer value is selected. Use of the RL_StatusBuffSel and the RUL_StatusBuffSel signals are shown in Section 11.7.2.3 and Section 11.7.3.3 respectively.

11.4 Initialization Blocks

11.4.1 StartToUnload

Before the Receiver can start to unload the ATM payload data, certain conditions must first occur. The StartToUnload signal is used to indicate that at least one buffer is completely full and data can start to be unloaded. By varying the state that triggers the assertion of StartToUnload, the amount of buffering can be varied. The larger the amount of buffering, then the less sensitive the Receiver will be to situations where an ATM cell arrives late or is lost. However, the usage of each buffer adds on a delay of 5.79ms. For voice, when the end-to-end delay approaches 25ms, echo cancellation circuitry becomes necessary.

The timing of the StartToUnload signal dictates what the initial delta between the header and tail pointers, i.e. the amount of buffering used, see Section 10.3.6. The StartToUnload signal is based on the comparison of the buffer number used by the Load side, RL_Buffer_Value(3:0), and the user selectable value, Buffer_Delta_Value(3:0). To ensure that StartToUnload is asserted at the beginning of an unload frame, the chip enable, of the register used for the StartToUnload signal, is connected to the status signal RUL_ChCntEQ31. The register is clocked by C256k and uses feedback from the register output so that once set, StartToUnload can only be cleared by a RESET.

The StartToUnload signal is equivalent to the Init field in the Unload status field, see Section 10.3.7, Section 11.6.2.4 and Section 11.7.3.3.

11.4.2 Receiver Load First Time

The RL_NFirstTime signal is a negative asserted signal used to indicate that it is the first time the Load side is accessing the memory. Therefore the RL_NFirstTime signal is used for initialization purposes so that status data can be properly read from the EPROM, see Section 10.3.2 and Section 10.3.8.

The RL_NFirstTime signal is set high at the end of the first reception of an ATM cell for channel 31. This is accomplished by using the AND of the RL_ChNumEQ31 and RL_IndexEQ46 signals which are discussed in Section 11.7.2.2 and Section 11.7.2.1 respectively. The RL_NFirstTime signal is implemented in this fashion for the prototype because there is no way to selectively activate and deactivate channels. Therefore, initially all supported channels will be activated and the reception of channel 31 can be used to indicate the state where all the active channels have been received once. Channel 31 is also chosen because it represents the signaling channel and must always be present regardless of which of the data channels are active.

11.4.3 Receiver Unload First Time

The RUL_NFirstTime signal is a negative asserted signal used to indicate that it is the first time the Unload side is accessing the memory. Therefore the RUL_NFirstTime signal is used for initialization

purposes so that status data can be properly read from the EPROM, see Section 10.3.2 and Section 10.3.8.

The RUL_NFirstTime signal is set high at the first complete T1 frame after the assertion of the StartToUnload signal. This one frame delay, from the start of the assertion of the StartToUnload signal, is to make sure that all 32 time slots have obtained the proper initialization status information from the EPROM.

11.5 MTS to Receiver Load Side Interface

11.5.1 Load Bit Count

The Receiver Load bit counter, RL_BitCnt is a 3 bit counter used to determine the byte boundaries of the serial FAU data. The RL_BitCnt counter is clocked by CLKA, and synchronously reset by FMB_Sync. Supporting combinational logic is used to generate LT4, ZeroBits6and7 and RL_BitCntEQ7 status signals and a 256 KHz clock, C256k. The LT4 signal is high for RL_BitCnt values of 0, 1, 2 and 7. The LT4 signal is used in the Header Recognition block to zero out bits 0, 1, 2 and 3, see Section 11.2.2.1. The ZeroBits6and7 is another status signal used by the Cell Delineation Block. The ZeroBits6and7 signal is used to zero out the MTS serial input during bits 6 and 7. The RL_BitCntEQ7 is generated using the TC output of the RL_BitCnt counter. The inverse of the MSB of the RL_BitCnt counter is used to generate the 256 KHz clock, C256k. Figure 51, "Receiver Load Bit Count Block," shows the data paths and Figure 52, "RL_BitCnt Timing," shows the timing for these signals.

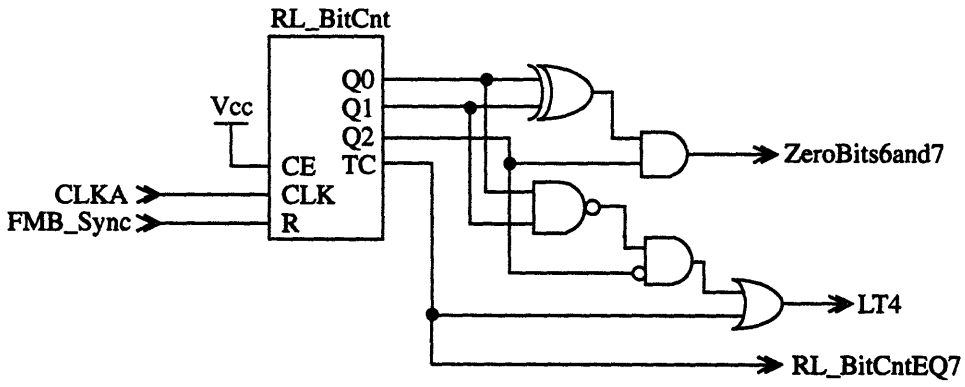


FIGURE 51. Receiver Load Bit Count Block

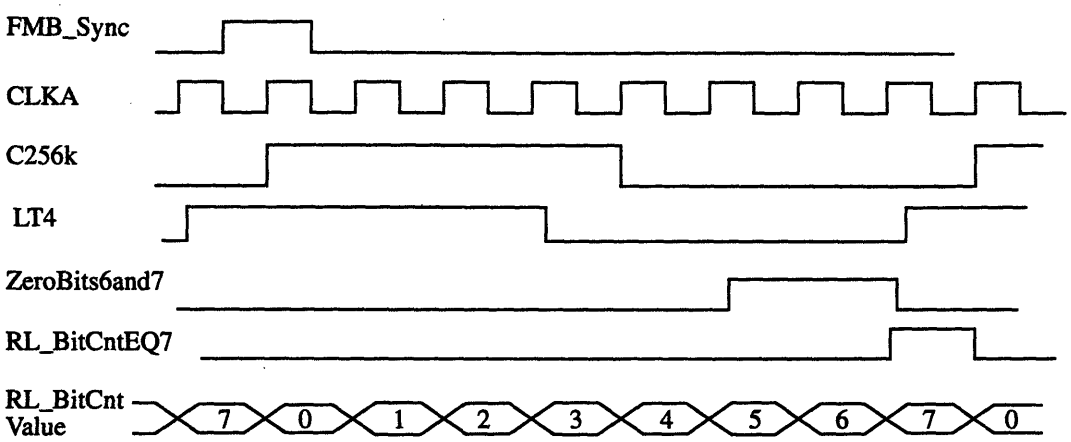


FIGURE 52. RL_BitCnt Timing

11.5.2 FAU Serial to Parallel Interface

The interface to the FAU serial data is synchronized through the use of the FMB and CLKA signals. The timing and components are the same as that used for the MTS serial to parallel interface on the Transmitter Load side, see Section 9.4.2.

11.5.3 Even Parity Generator

The Even Parity Generator block on the Receiver is identical to that used for the Transmitter, see Section 9.4.3.

11.5.4 Load Side Dead Byte Count Block

The TMD24 board data rate is 2.048MHz, which contains 8 inactive time slots. It is necessary to ignore the inactive time slot data from the data bytes sent to the ATM cell delineation block and the ATM payload storage blocks.

A 2 bit counter, RL_Dead_ByteCnt, is used to keep track of which time slots are inactive. The Dead Byte Count block generates the two status signals RL_DeadByteEQ0 and RL_DeadByteEQ1, which are asserted when the RL_Dead_ByteCnt value is equal to 0 and 1 respectively. RL_DeadByteEQ0 is used to indicate when the current byte in the FAU serial to parallel shift register is from one of the 8 inactive time slot.

11.6 Receiver Memory Data Bus

11.6.1 Data Multiplexers

11.6.1.1 Status Data Multiplexer

The Status Data multiplexer, Status_Mux, is a 3 to 1 8 bit multiplexer used to select status values. The status values to select from are the Index, Unload, and the Load status values. A XNOR gate is used at the output of the Status_Mux multiplexer to generate a parity bit for the status data. Table 19, "Status

Multiplexer Selection Table," shows the outputs selections for the Status_Mux multiplexer. The multiplexer inputs S0 and S1 are connected to the control signals R_IndexSel_S0 and R_IndexSel_S1 respectively, which are described in Section 11.3. The output of the Status_Mux is connected to one of the inputs of the Data_In_Mux, refer to Section 11.6.1.2.

TABLE 19. Status Multiplexer Selection Table

Input		Output
S1	S0	Status Value
0	0	Index Status
0	1	Unload Status
1	0	Load Status
1	1	Not Used

11.6.1.2 Memory Data Bus Input Multiplexer

A 3 to 1 9 bit wide data multiplexer, Data_In_Mux, is used to make the final source selection that will be passed on to the memory data bus. The Data_In_Mux selects from three data sources, the Status_Mux, the RL_Data_In register and the Processor data interface. The S0 and S1 inputs of Data_In multiplexer are connected to the control signals SRAM_Data_In_S0 and SRAM_Data_In_S1 respectively. The selection of the input source is shown in Table 19, "Status Multiplexer Selection Table."

TABLE 20. Data_In_Mux Selection Table

Input		Output
S1	S0	Status Value
0	0	Processor Data
0	1	Status_Mux output
1	0	Data_In register output
1	1	Not Used

11.6.2 Data Multiplexer Input Sources

11.6.2.1 Data In Register

Once the serial data has been converted and formed into an 8 bit data value plus 1 bit parity, these 9 bits are latched into the RL_Data_In register. This allows the Receiver Control FSM to write the loaded data into the appropriate SRAM buffer anytime before the next byte is shifted in by the FAU serial to parallel shift register. The RL_Data_In register is hardwired to latch in all bytes except those corresponding to the eight inactive time slots. This is done by using CLKA as the clock and the logical AND of the RL_BitCntEQ7 output of RL_BitCnt counter and the inverse of RL_DeadByteCntEQ0 as the chip enable. Thus, the RL_Data_In register is enabled for all active time slots.

11.6.2.2 Index Status Value

The Index status value is generated from the Receiver Unload Index block, see Section 11.7.3.2.

11.6.2.3 Load Status Value

The Load status value is obtained from the Receiver Load Buffer Addressing block, see Section 11.7.2.3. The Load status value also contains the SN from the received ATM cell.

11.6.2.3.1 Load Sequence Number Register

A 3 bit register, Act_SN_Value, is used to store the received ATM cell's SN value. The data input to the SN register is from the RL_Data_In register. The Act_SN_Value register is clocked by CLKA and uses the AND of SNRegCE and RL_BitCntEQ7 as the CE. The SNRegCE control signal is produced from the Cell_Delineation FSM. The use of CLKA and RL_BitCntEQ7 eliminates any timing issues that might occur due to routing timing delays if C256k was used. The output of the Act_SN_Value register goes to the Data_In_Mux multiplexer.

11.6.2.4 Unload Status Generation

The Unload status value is generated from the Receiver Unload Buffer Addressing block, see Section 11.7.3.3.

11.6.2.5 Unload Data Register

A 9 bit register, RUL_Data_Out, is used to register data read from either the SRAM or the EPROM. The RUL_Data_Out register is clocked by CLKA and is chip enabled by the R_DataRegOutCE signal, which is generated by the Receiver FSM. The output of RUL_Data_Out register is used as the input to the MTS parallel to serial converter and parity check blocks, see Section 11.8.1 and Section 11.8.3 respectively. The RUL_Data_Out register allows data to be read out of memory at any point in time prior to the time that data is required by the MTS parallel to serial shift register.

11.6.2.6 Processor Interface

In the event of future modifications and to allow access to the SRAM for testing, a processor interface to the SRAM is provided. This interface allows the processor to have access to the SRAM data input and the SRAM addressing.

The Receiver FSM has a conditional statement for the processor access from the home state, state 1. This is based off of the signals R_Req and R_Ack. If the processor requests access to the memory then it asserts the R_Req signal. If there is excess time between byte cycles, then the Receiver FSM asserts the R_Ack signal and relinquishes the memory data and address paths to the processor.

11.7 Receiver Memory Addressing

11.7.1 Memory Address Multiplexer

The Receiver memory addressing can come from three sources, the processor address interface, see Section 11.6.2.6, the Load side addressing, see Section 11.7.2, and the Unload side addressing, see Section 11.7.3. The Receiver memory address multiplexer is a 3 to 1 15 bit multiplexer. The input

selection is controlled by the SRAM_AddSel_S0 and the SRAM_AddSelS1 signals, see Table 21, "Receiver Address Multiplexer Selection Table." The output of this multiplexer is used as the memory addressing bits.

TABLE 21. Receiver Address Multiplexer Selection Table

Input Select		Output
SRAM_AddSel_S1	SRAM_AddSel_S0	Address Source
0	0	Processor Address Interface
0	1	Load Side Addressing
1	0	Unload Side Addressing
1	1	NA

11.7.2 Load Side Addressing

The Load side addressing is composed of three parts, the Index Counter value, the Channel Counter value, and the Buffer Counter value. The logical arrangement of these fields is shown in Figure 53, "Receiver Load Side Logical Organization of Memory Addressing Bits."

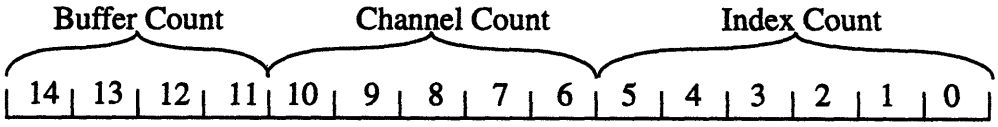


FIGURE 53. Receiver Load Side Logical Organization of Memory Addressing Bits

11.7.2.1 Index Addressing Block

The index addressing consists of a 6 bit counter, RL_Index_Cnt, which is clocked by CLKA. The RL_Index_Cnt counter is chip enabled by the control signal RL_IndexCntCE and reset by the inverse of the Payload status signal. The RL_IndexCntCE is generated from the Receiver Control FSM and the Payload status signal is generated by the Cell_Delineation FSM.

The Index Count goes from 0 to 46, which covers the 47 bytes of payload data.[24] A RL_IndexEQ46 status signal is generated to indicate when the index count equals 46, and is used to indicate the end of a payload. There is also combinational logic which can be selected to force the index value to 111110 or 11111, which represent the index addresses for the Unload and Load status values. The address forcing logic is selected by the control signals R_IndexSel_S0 and R_IndexSel_S1 and is shown in Table 19, "Status Multiplexer Selection Table."

---

24. The value of 47 is used because while the SN is technically part of the payload, it is treated as status information, and therefore there are only 47 bytes of data.

TABLE 22. Index Selection Table

Input		Output
R_IndexSel_S0	R_IndexSel_S1	Status Value
0	0	RL_Index_Cnt value
0	1	Unload Status Address
1	0	Load Status Address
1	1	NA

11.7.2.2 Channel Number Register Block

This block contains the ChNumReg register used to hold the channel value of the received ATM cell and is described in Section 11.2.2.2.

11.7.2.3 Buffer Addressing Block

The buffer value used for addressing is determined through the comparison of the HP and TP values. The comparison is implemented through the use of the data paths depicted in Figure 54, "Receiver Load Buffer Comparison Data Paths."

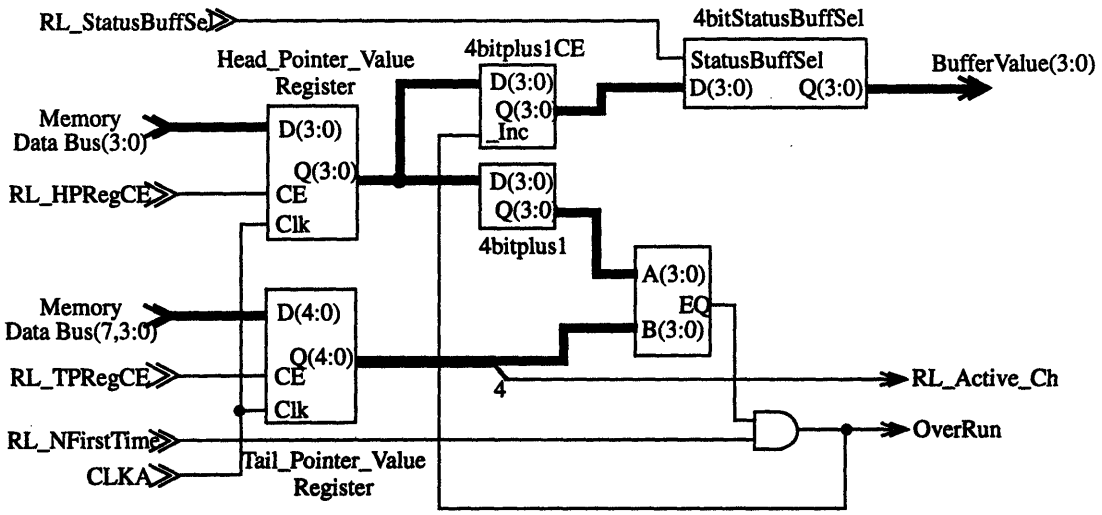


FIGURE 54. Receiver Load Buffer Comparison Data Paths

The Head_Pointer_Value register is used to hold the 4 bit HP value. The Tail_Pointer_Value register is used to hold the 4 bit TP value and the Active_Channel status bit.

The Buffer Comparison block function can be summed up in the follow pseudo C code:

```
if (((HP_Value + 1) == (TP_Value)) & RL_NFirstTime) then
        OverRun = 1;
else OverRun = 0;
```

```
if (RL_StatusBuffSel == 1) then
        BufferValue:= 0b1111;
else    {
        if (OverRun == 0) then
                BufferValue:= HP_Value + 1;
        else BufferValue:= HP_Value;
        }
```

It should be noted that the BufferValue(3:0) is always based on the HP value, which is responsible for keeping track of the buffer position for loading in new ATM cells. The RL_Active_Ch value is not modified because there is no higher layer management which can update this status field.

11.7.3 Receiver Unload Addressing

The Unload side addressing is composed of three parts, the Index Counter value, the Channel Counter value, and the Buffer Counter value. The logical arrangement of these fields is shown in Figure 53, "Receiver Load Side Logical Organization of Memory Addressing Bits."

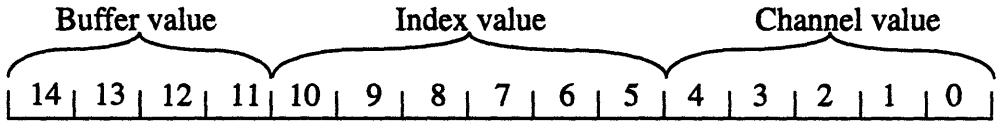


FIGURE 55. Receiver Unload Side Logical Organization of Memory Addressing Bits

11.7.3.1 Channel Count Addressing Block

The Channel Count block is composed of a 5 bit counter, RUL_Channel_Cnt, and an adder. The RUL_Channel_Cnt counter is clocked by C256k and is reset by FMB_Sync. The RUL_Channel_Cnt counter is used to keep track of the time slot alignment between the ATM board and the data from the TMD24 board. However, a 1 time slot delay is introduced by the RUL_Data_Out register. Thus an adder for the RUL_Channel_Cnt counter is required so that data sent to the RUL_Data_Out register will arrive 1 time slot before it is needed by the MTS parallel to serial shift register. The adder is therefore a simple incrementor which outputs the result as a mod 32 value. The output of the incrementor is the Channel Count field which is used in the memory Unload side addressing.

11.7.3.2 Index Addressing Block

The Index Counter value for the unloading of data is stored in a 6 bit register, Index_Value. The input of the Index_Value register is the memory data bus. The output of the Index_Value register goes to a NOR gate, an adder and a forcing block as shown in Figure 56, "Receiver Unload Index Value Data Paths."

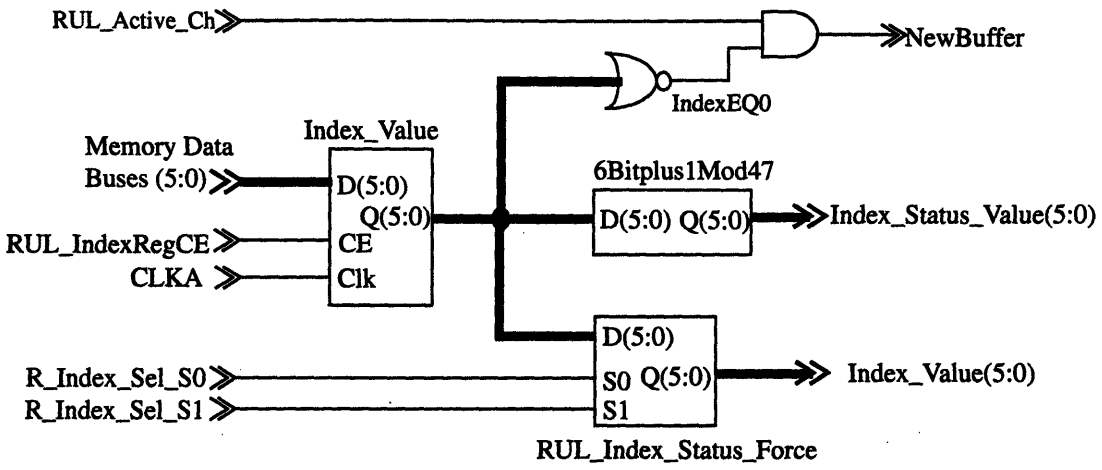


FIGURE 56. Receiver Unload Index Value Data Paths

The 6 input NOR gate is used to signal the start of a new buffer, i.e. when the Index value is equal to 0. The output of the NOR gate is then ANDed with the RUL_Active_Ch status bit for the channel. This produces the final status signal NewBuffer.

The adder is actually a 6 bit incrementor which outputs the result as a mod 47 value. This value is the next Index Counter value, and is the Index_Status_Value used by the Status_Mux multiplexer, see Section 11.6.1.1. The RUL_Index_Status_Force block is essentially a 4 to 1 multiplexer, but because 3 of the 4 inputs are fixed values, it is more efficient to implement the RUL_Index_Status_Force block using combinational logic. The RUL_Index_Status_Force block selects between the values shown in Table 19, "Status Multiplexer Selection Table." The output of the RUL_Index_Status_Force block is used as the Index Count field for the Receiver Unload side memory addressing.

TABLE 23. RUL_Index_Status_Force Block Selection Table

Input		Output
R_IndexSel_S1	R_IndexSel_S0	Status Value
0	0	Index Status Address
0	1	Unload Status Address
1	0	Load Status Address
1	1	RUL_IndexCnt value

11.7.3.3 Buffer Addressing Block

The Buffer Count value used for addressing is determined through the comparison of the TP and the HP values. The comparison is implemented through the use of the data paths depicted in Figure 54, "Receiver Load Buffer Comparison Data Paths."

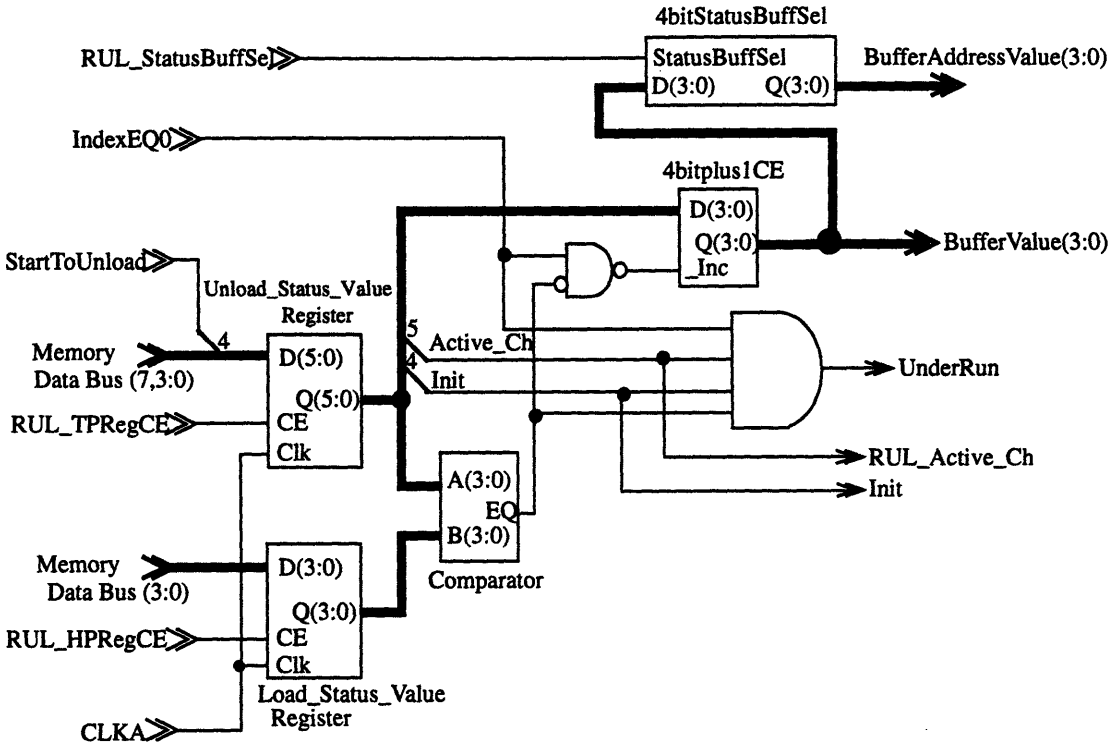


FIGURE 57. Receiver Unload Buffer Comparison Data Paths

The Tail_Pointer_Value register is a 6 bit register used to hold the 4 bit TP value, the RUL_Active_Ch status bit and the Init status bit. The Head_Pointer_Value register is a 4 bit register used to hold the 4 bit HP value. The Buffer Comparison block function can be summed up in the follow pseudo C code:

```
if (RUL_StatusBuffSel == 0) then
        if ((TP_Value!= HP_Value) & (IndexEQ0)) then
                BufferValue:= TP_Value + 1;
        else BufferValue:= TP_Value;
else BufferValue:= 0b1111;
```

The BufferAddressValue(3:0) is used for the Buffer Count field for the Unload memory addressing. The BufferValue(3:0) is the value used for the TP Unload status value. The RUL_Active_Ch status value remains unchanged because there is no higher layer management to update this status field. The Init value is the registered value of the StartToUnload signal. The UnderRun status value is generated by the AND of the IndexEQ0, the RUL_Active_Ch, the Init and the EQ output from the comparator. The IndexEQ0 signal is used so that the UnderRun value is only generated at the start of a new buffer.

11.8 Receiver to MTS Unload Side Interface

11.8.1 MTS Parallel to Serial Converter

The components and timing for the MTS parallel to serial converter are identical to those of the FAU parallel to serial converter in the Transmitter Unload side, see Section 9.7.2.

11.8.2 Inactive Time Slot Data Insertion

The all 1's pattern for inactive time slots is inserted based on the status signals RUL_Active_Ch and Init. RUL_Active_Ch is the Active_Channel status bit for the time slot being unloaded and the Init status signal is as described in Section 10.3.7. The data paths for the forcing of the serial output is shown in Figure 58, "Inactive Time Slot Data Insertion Data Paths." The status signal Inactive_Ch is generated from the NAND of the RUL_Active_Ch and the Init signals.

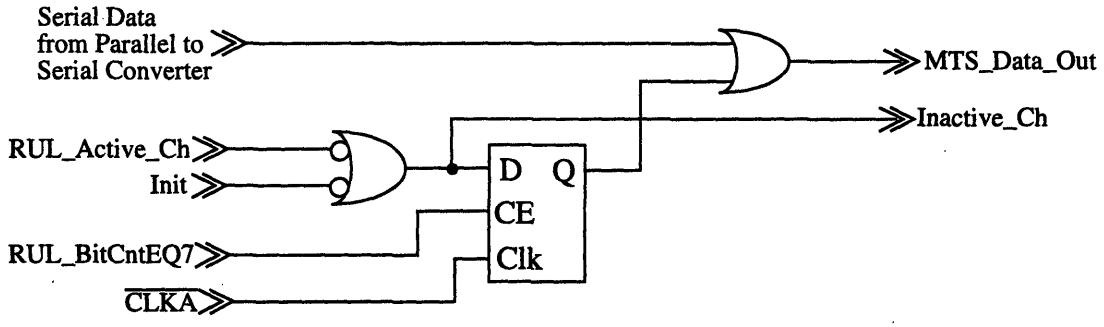


FIGURE 58. Inactive Time Slot Data Insertion Data Paths

11.8.3 Parity Check

The components and timing for the Receiver Unload serial parity check are based on those of the parity check on the Transmitter Unload side, see Section 9.7.3. The difference is that the final parity error register is chip enabled by the Inactive_Ch status signal, see Section 11.8.2, so that parity is not checked for inactive time slots and there is no need to check for EPROM data.

Chapter 12 Conclusion

12.1 Project Status

Initially there were several goals for this project. Some of the primary goals are listed below:

- learn and gather specifications on ATM specifications
- investigate the various chip sets available
- centralize this information and distribute it to interested engineers
- develop the design for a PBX to ATM UNI
- satisfy the Master's thesis requirements

All of the above goals were met. With the primary goals met, an attempt to build and test the board in the remaining time was also attempted. However, the returned wire wrapped board was not finished until the last week of the assignment. At this point debugging was begun. Unfortunately, because of difficulties with the chip set used, the Xilinx FPGAs could not be configured properly. Therefore, the current status of the project is that there is a physical board built and final debugging of the board will be continued at Rolm as time allows.

12.2 Design Verification

Design verification was done through the use of end-to-end simulation techniques. The simulation incorporated both the Xilinx FPGA designs, SRAMs and EPROMs programmed with the appropriate values. The results of key simulations is shown in Appendixes G-J. Appendix G shows the timing simulation results for the Transmitter. Appendix H contains the timing simulation results for the cell delineation blocks on the receiver. Appendix I and Appendix J both are timing simulations for the Receiver. Appendix I depicts the state where the Receiver is still buffering ATM cells and has yet to start unloading valid data. Appendix J shows the state where the Receiver has buffered the required number of ATM cells and has begun unloading valid data from the ATM cell payloads.

These simulations are based on a loop back test configuration, where the input to the Transmitter is the TDM data with the appropriate FMB and CLKA signals generated by the test logic described in Section 9.9. The ATM cell output of the Transmitter is then connected to the input of the Receiver. The output of the Receiver was then checked against the input pattern and the required PCM highway timing to verify the design.

12.3 Closing Remarks

12.3.1 Why Implement a PBX to ATM UNI

The first question is why would one want to implement this type of interface. The first reason is while current telephony and data networks operate as separate entities, eventually the two will be unified into

a single ATM network. As shown in Section 4.1, only 21 of the normal 24 T1 channels can be supported in ATM over DS1, which is a loss in data bandwidth. However, the benefit from ATM is obtained because for ATM, the user is only paying for the actual bandwidth used. This is in contrast with standard time division multiplexing (TDM) over a leased T1 line where the user is paying for the entire DS1 bandwidth whether or not they are actually using all or a fraction of the bandwidth. With ATM over DS1, in place of the unused channels/bandwidth, idle ATM cells are sent. These idle cells are necessary to maintain the connection data rate, but are filtered out at the first UNI to NNI interface and therefore do not occupy bandwidth in the ATM network.

Many people may ask why implement the ATM protocol, which is suppose to be very fast, running in the hundreds of megabits, over DS1 lines? There are two major reasons for implementing ATM over DS1 lines. The first is because of the enormous investment already made in DS1 lines. In order of ATM to be commercially accepted, it must be able to make use of the already existing physical network. The second reason is centered more on the educational and implementation benefits. Educationally, the basic underlying design and conceptual understanding of ATM is the same whether the line rate is 1.544Mbps or 155Mbps. Also the actual construction of a board operating at 1.544Mbps is simpler than one running at 155Mbps, because of the routing and layout constraints due to cross talk and transmission line effects at higher speeds. Therefore, operating within the time constraints, ATM over DS1 lines is the perfect balance of educational and practical benefits.

12.3.2 The Future

Overall, I believe that true ATM to the desktop and homes is still some time away. What I allude to when I say "true" ATM can be shown by referring back to Figure 1, "Overall System Setup." Mentioned are two methods of using the ATM UNI developed.

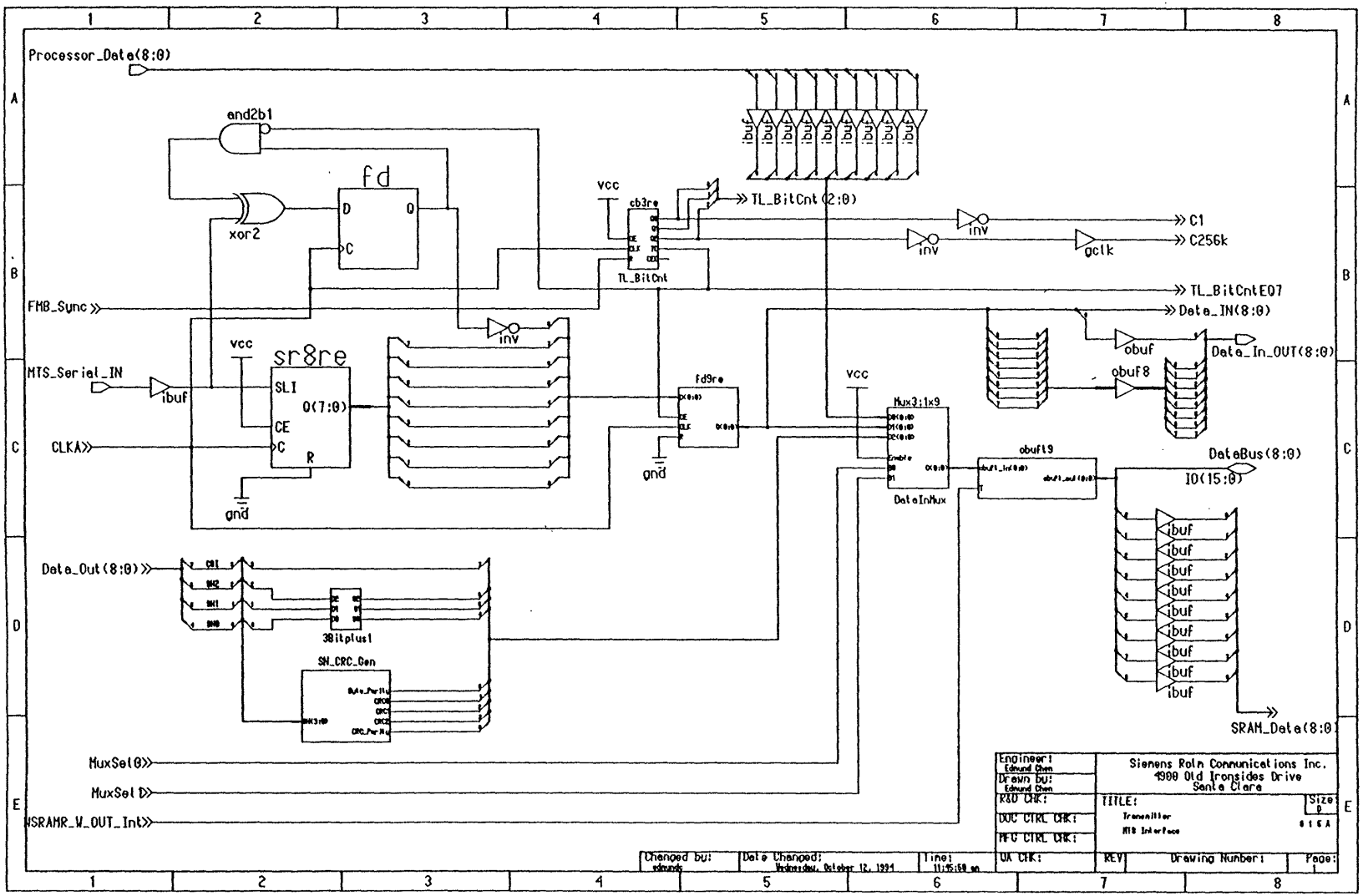
One type of network is simply to use leased T1 lines and the standard Central Office (CO) switching. While this method has the benefit of not requiring the purchasing of new switching equipment required for ATM cell switching, this network also does not use any of the advantages of ATM. Actually, in this method, line bandwidth is actually lost due to the presence of the ATM header bytes. If ATM was to be used in any one part of the network, it should start at the network backbone and then migrated to the LAN and user interfaces.

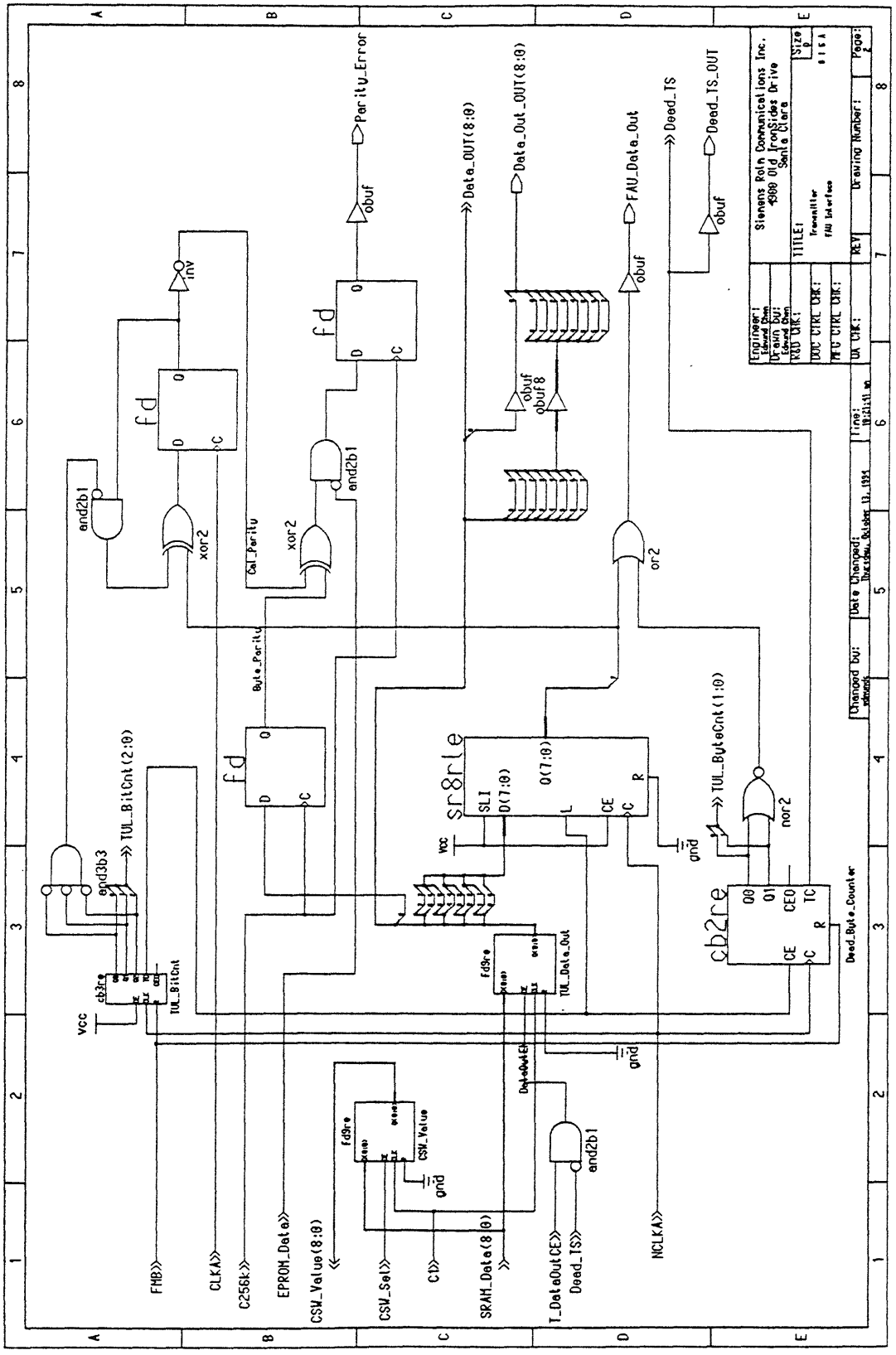
The second type of network is what is considered as true ATM switching. In this network format, the ATM UNI board developed is actually interfacing with an ATM network. The advantage of this network is now the ATM cells can be separately routed to their individual address locations. Another advantage is that only the bandwidth required is actually used. That is idle ATM cells are inserted to maintain the physical line data rate and can be removed by the ATM network. The freed bandwidth can then of course be reallocated. The one disadvantage of this network is the investment required to purchase and install the necessary ATM switching components.

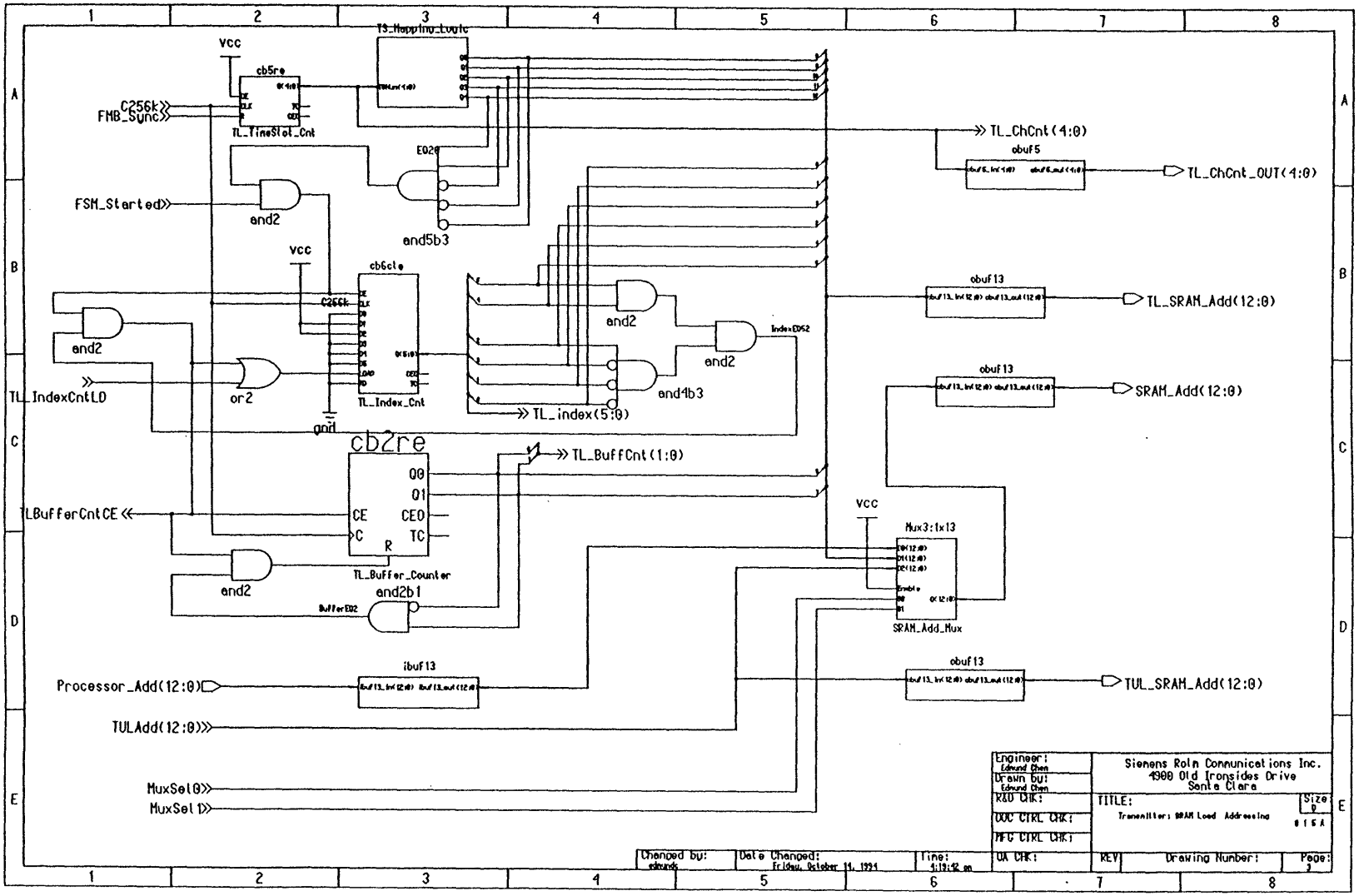
While ATM is heralded as the unifying protocol for CBR and VBR data, I believe that these two data types will mostly remain sperate for sometime. Specifically, the telecommunications and data networks will remain independent entities until the ATM network has been well established. Many ATM issues such as quality of service, dynamic bandwidth allocation and latency must still be addressed before ATM networks will be able to successfully handle both CBR and VBR data. Even with these issues solved, before ATM can be properly supported in a public network, the issue of billing must be resolved. However, the migration to ATM is inevitable. While technologies such as FDDI and 100Mbs switched ethernet maybe network stepping stones, the demand for even higher network speeds capable of handling voice, video and data indicates the underlying need for an ATM network.

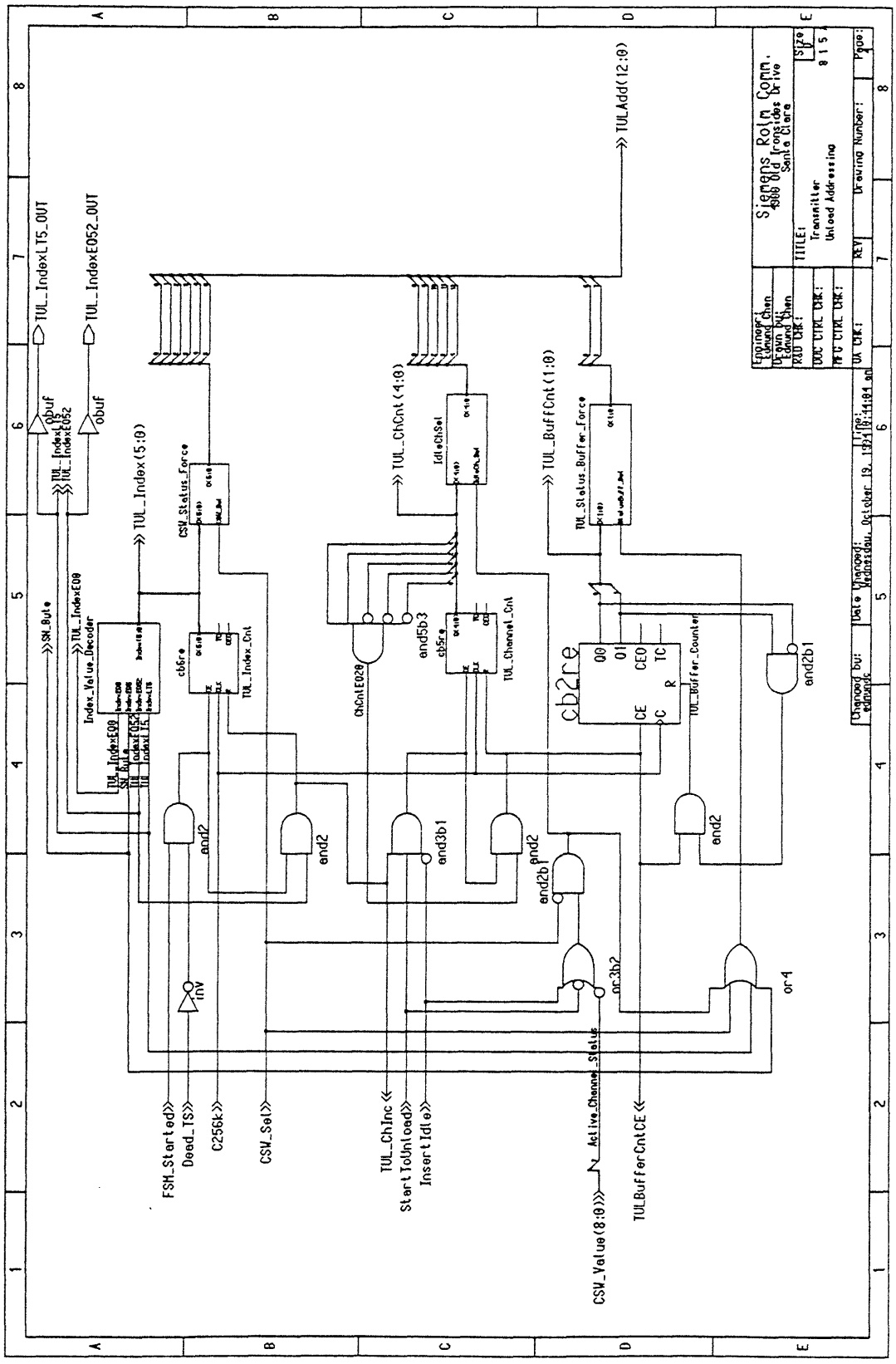
Appendix A Transmitter Schematics

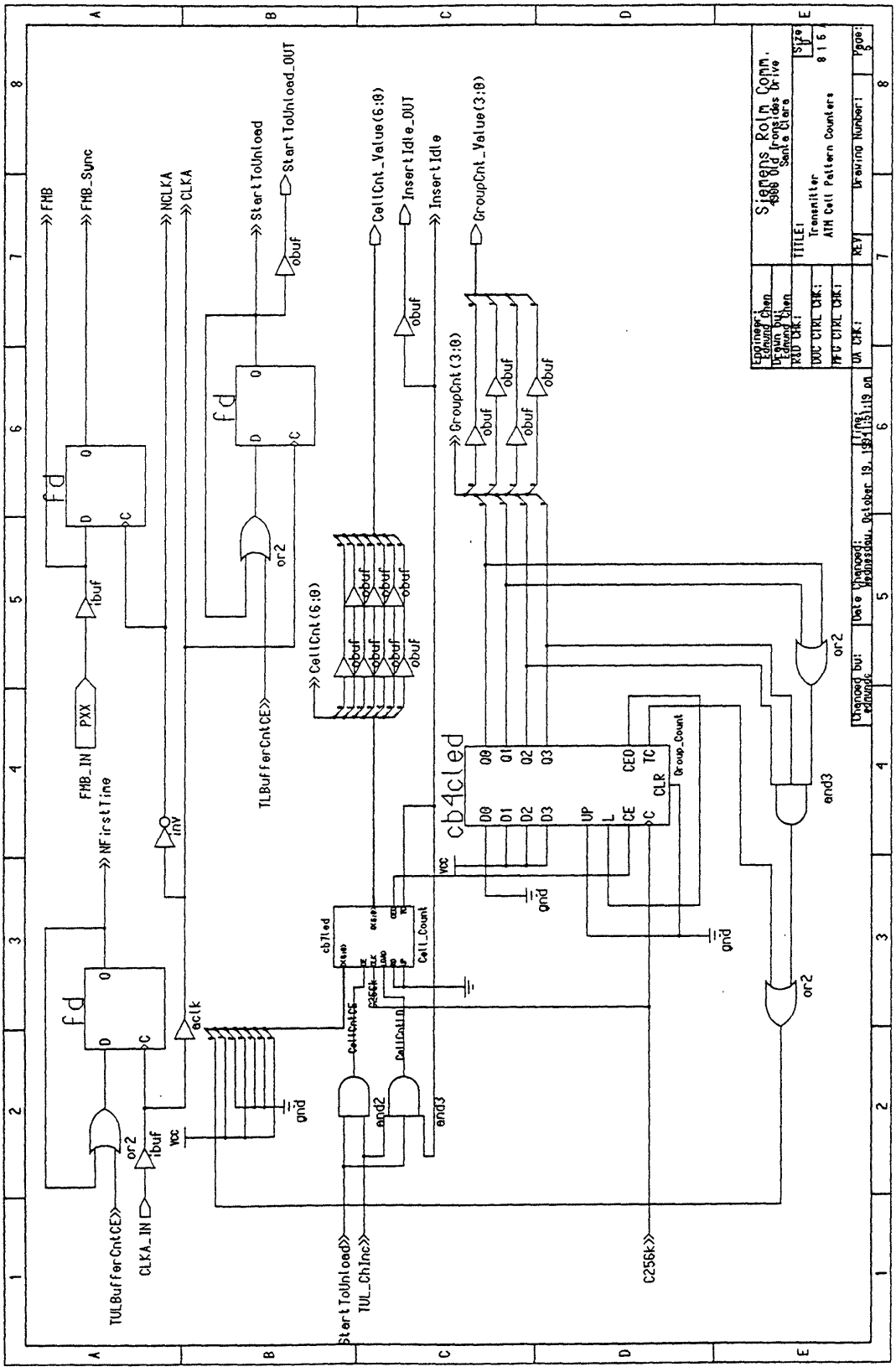
Schematics for Transmitter portion of PBX to ATM UNI.

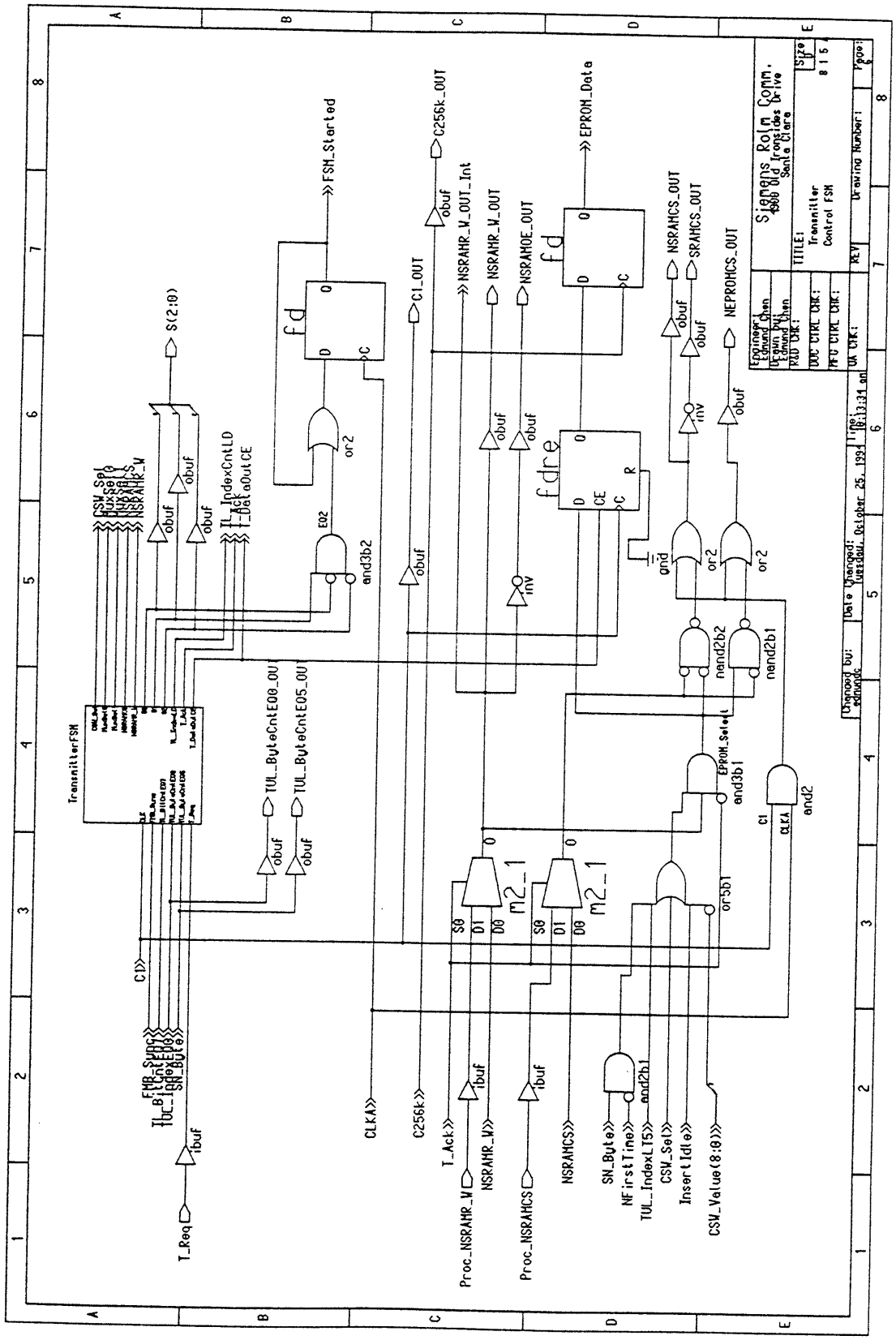

Processor_Data(8:0)
and2b1
fd
xor2
D
Q
C
vcc
cb3re
TL_BitCnt(2:0)
TL_BitCnt
inv
C1
C256k
gclk
TL_BitCnt[0:7]
FMB_Sync
Data_IN(8:0)
obuf
Data_In_OUT(8:0)
obuf8
sr8re
SLI
Q(7:0)
CE
C
R
MTS_Serial_IN
ibuf
CLKA
gnd
fd9re
Mux3:1x9
DataInMux
obuf9
DataBus(8:0)
IO(15:0)
Data_Out(8:0)
3Bitplus1
SN_CRC_Gen
SRAM_Data(8:0)
MuxSel(0)
MuxSel(1)
NSRAMR_W_OUT_Int
Engineer: Edmund Chen
Drawn by: Edmund Chen
R&D CHK:
DOC CTRL CHK:
MFG CTRL CHK:
QA CHK:
Siemens Rolm Communications Inc.
4900 Old Ironsides Drive
Santa Clara
TITLE:
Transmitter
MTB Interface
Size: D
B 1 6 A
Changed by: edmunds
Date Changed: Wednesday, October 12, 1994
Time: 11:15:58 am
REV
Drawing Number:
Page: 1

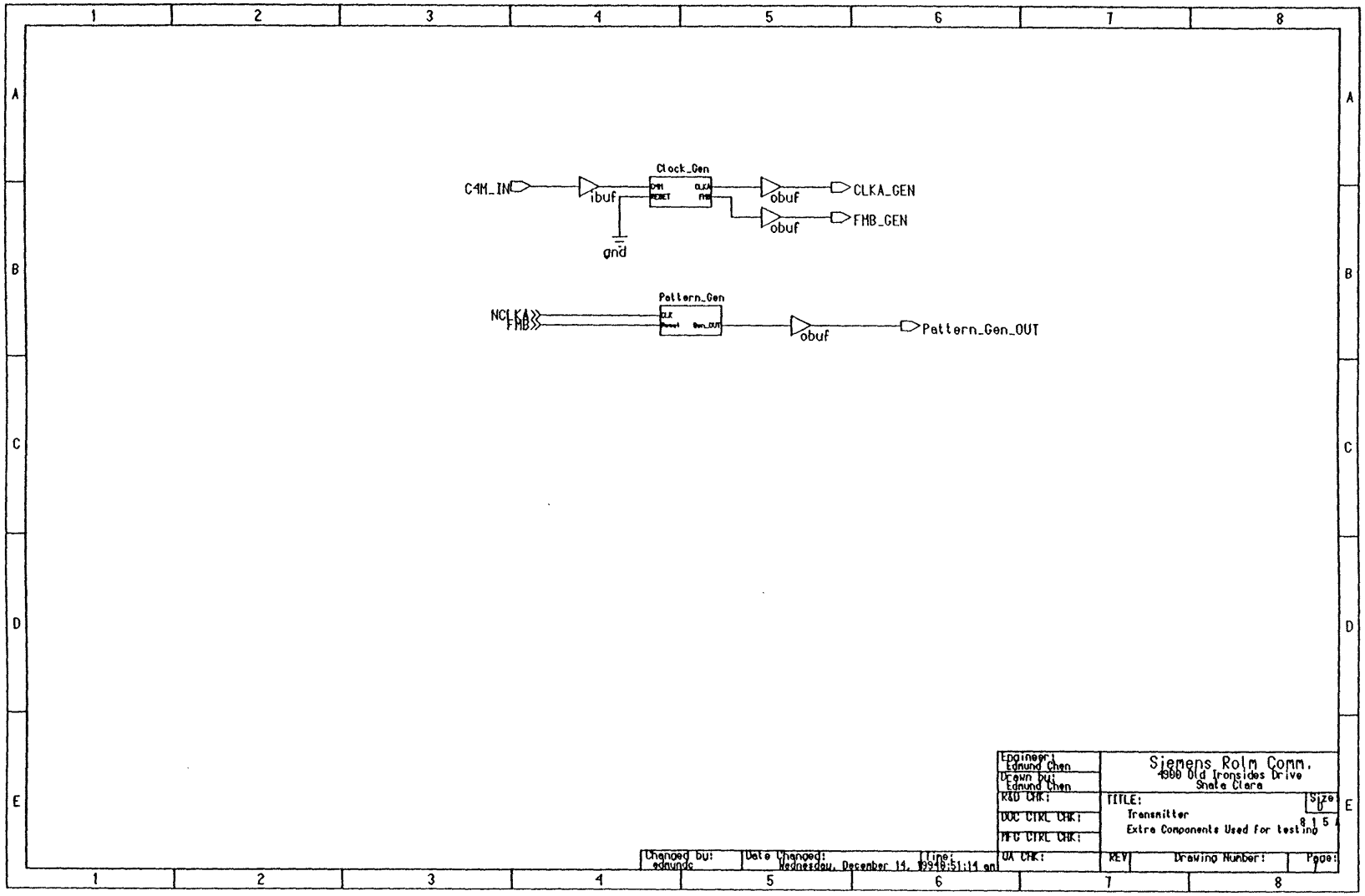

Parity_Error
Data_OUT(8:0)
Data_Out_OUT(8:0)
FAU_Data_Out
Dead_TS
Dead_TS_OUT
inv
fd
and2b1
xor2
Cal_Parity
obuf
obuf8
and3b3
TUL_BitCnt(2:0)
Byte_Parity
sr8rle
SLI
D(7:0)
Q(7:0)
L
CE
C
R
vcc
gnd
or2
nor2
TUL_ByteCnt(1:0)
cb2re
Q0
Q1
CEO
TC
cb3re
TUL_BitCnt
fd9re
TUL_Data_Out
Dead_Byte_Counter
CSW_Value
CSW_Value(8:0)
EPROM_Data
CLKA
C256k
FMB
CSW_Sel
C1
SRAM_Data(8:0)
T_DataOutCE
Dead_TS
NCLKA
Siemens Rolm Communications Inc.
4900 Old Ironsides Drive
Santa Clara
TITLE:
Transmitter
FAU Interface
Size
D
KEY
Drawing Number
Page
2


TS_Mapping_Logic
vcc
cb5re
C256k
FMB_Sync
TL_TimeSlot_Cnt
E02e
TL_ChCnt(4:0)
obuf5
TL_ChCnt_OUT(4:0)
FSM_Started
and2
and5b3
vcc
cb6cle
C256k
obuf13
TL_SRAM_Add(12:0)
and2
and2
and4b3
IndexED52
TL_IndexCntLD
or2
gnd
TL_Index_Cnt
TL_index(5:0)
obuf13
SRAM_Add(12:0)
cb2re
TL_BuffCnt(1:0)
Q0
Q1
CE
CEO
C
TC
R
TLBufferCntCE
TL_Buffer_Counter
and2
and2b1
BufferED2
vcc
Mux3:1x13
SRAM_Add_Mux
ibuf13
obuf13
Processor_Add(12:0)
TUL_SRAM_Add(12:0)
TULAdd(12:0)
MuxSel0
MuxSel1
Engineer: Edmund Chen
Drawn by: Edmund Chen
R&D CHK:
DOC CTRL CHK:
MFG CTRL CHK:
QA CHK:
Siemens Rolm Communications Inc.
4900 Old Ironsides Drive
Santa Clara
TITLE:
Transmitter: SRAM Load Addressing
Size: D
8 1 5 A
Changed by: edmunds
Date Changed: Friday, October 14, 1994
Time: 4:13:52 pm
REV
Drawing Number:
Page: 7


TUL_IndexLT5_OUT
TUL_IndexEQ52_OUT
TUL_IndexEQ00_OUT
TUL_Index(5:0)
TUL_ChCnt(4:0)
TUL_BuffCnt(1:0)
TULAdd(12:0)
SW_Byte
Index_Value_Decoder
CSW_Status_Force
IdleChSel
TUL_Status_Buffer_Force
TUL_Index_Cnt
TUL_Channel_Cnt
TUL_Buffer_Counter
ChCntEQ20
and5b3
and2
and3b1
and2b1
or3b2
or4
inv
cb5re
cb2re
FSM_Started
Dead_TS
C256k
CSW_Sel
TUL_ChInc
StartToUnload
InsertIdle
CSW_Value(8:0)
Active_Channel_Status
TULBufferCntCE
Siemens Rolm Comm.
Old Ironsides Drive
Santa Clara
TITLE:
Transmitter
Unload Addressing
Drawing Number:

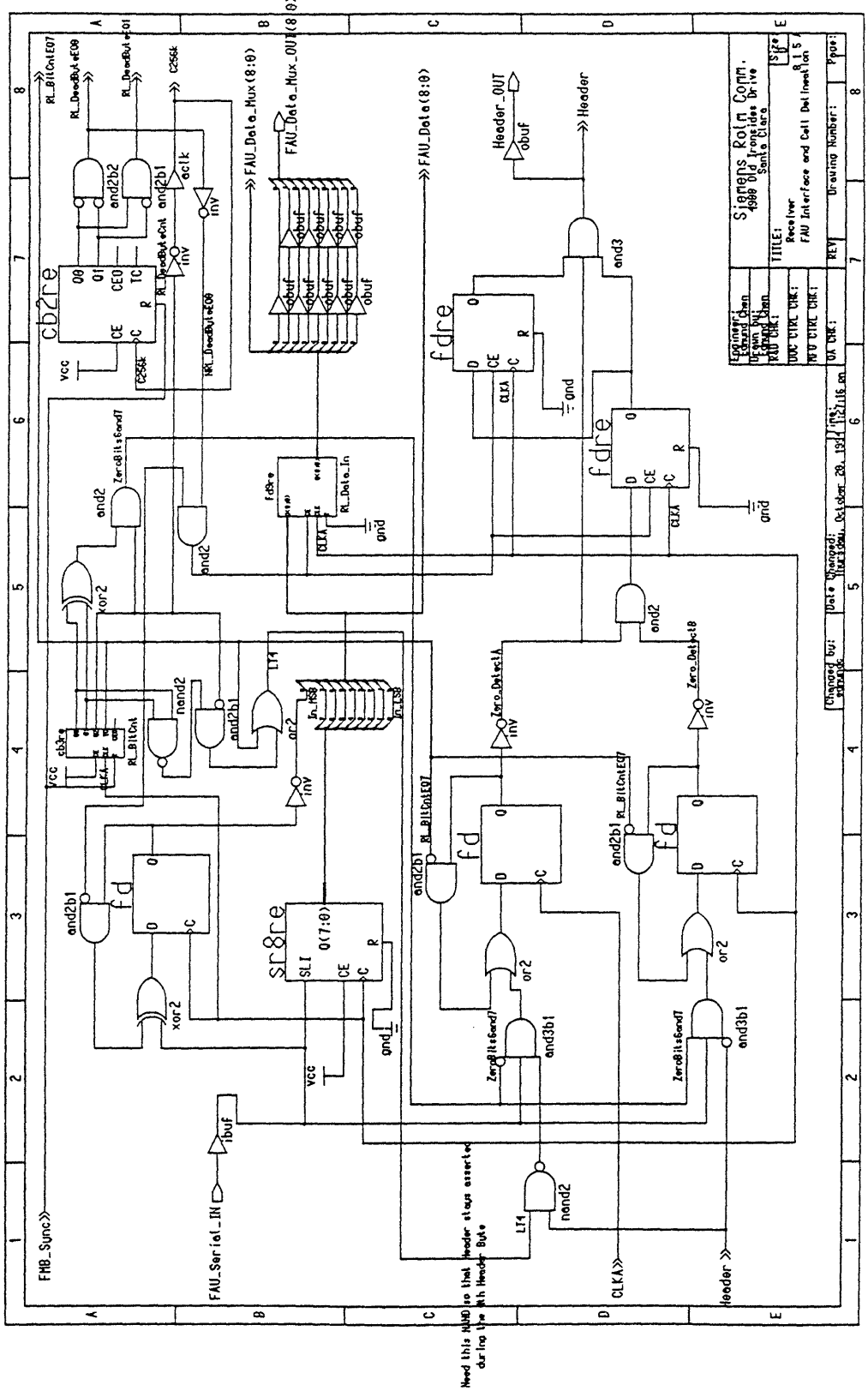

FMB
FMB_Sync
NCLKA
CLKA
StartToUnload
StartToUnload_OUT
CellCnt_Value(6:0)
InsertIdle_OUT
InsertIdle
GroupCnt_Value(3:0)
fd
or2
ibuf
obuf
PXX
FMB_IN
NFirstTime
inv
TLBufferCntCE>>
CellCnt(6:0)
GroupCnt(3:0)
cb4cled
Q0
Q1
Q2
Q3
CEO
TC
CLR
UP
L
CE
C
D0
D1
D2
D3
Group_Count
Cell_Count
vcc
gnd
aclk
and2
and3
TULBufferCntCE>>
CLKA_IN
StartToUnload>>
TUL_ChInc>>
C256k>>
Siemens Rolm Comm.
4900 Old Ironsides Drive
Santa Clara
TITLE:
Transmitter
ATM Cell Pattern Counters
Engineer:
Edmund Chen
Drawn by:
Edmund Chen
Size
B
8164
Page:
8
Drawing Number:
Date Changed:
Wednesday, October 19, 1994 11:19 am
Changed by:
edmundc

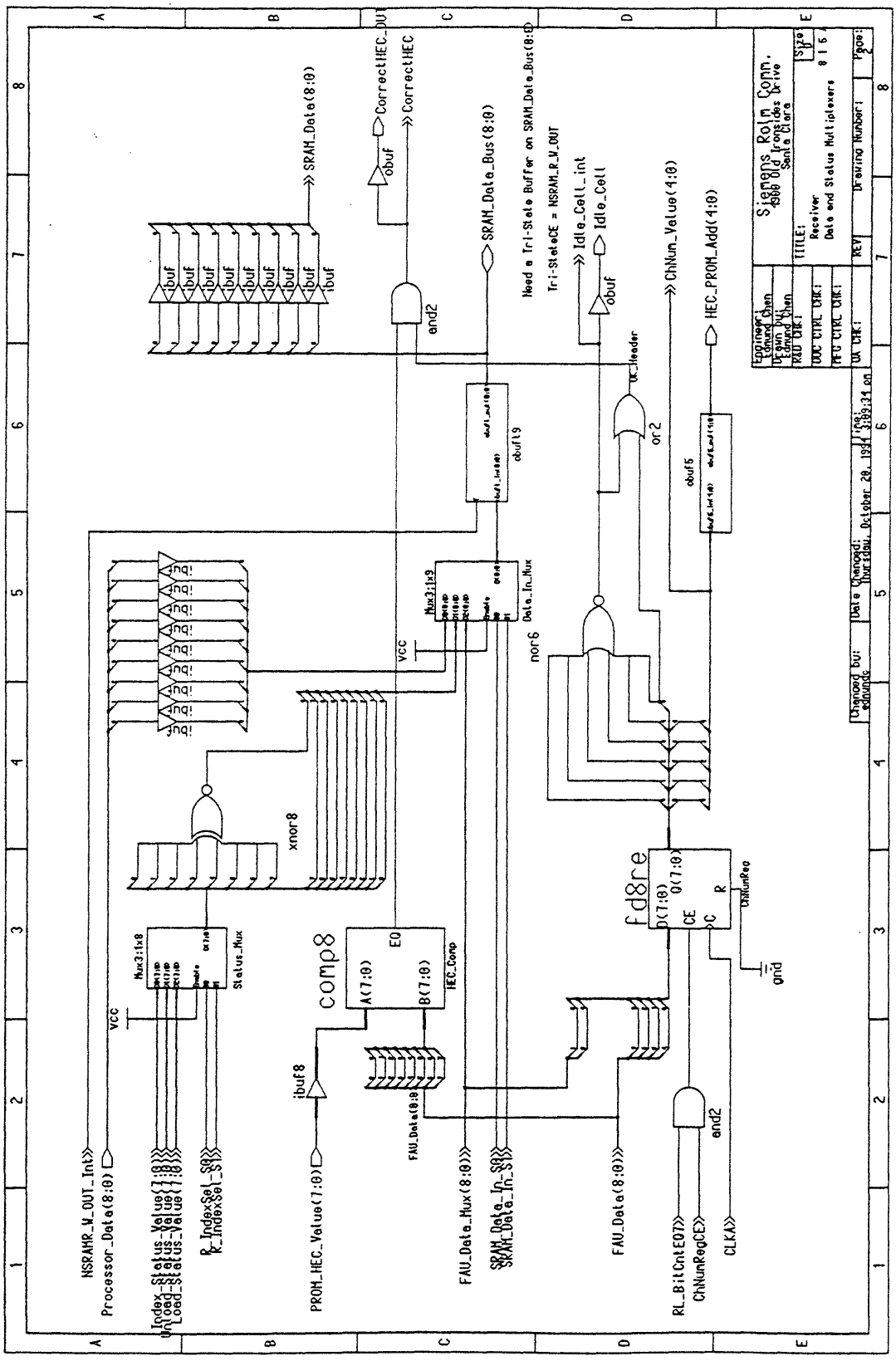

TransmitterFSM
T_Req
ibuf
FMB_Sync
TUL_ByteOrder
TUL_IndexEn0
SN_Byte
C1>>
S(2:0)
CSW_Sel
NSRAMCS_W
TUL_IndexCntLD
T_DataOutCE
TUL_ByteCntE00_OU
TUL_ByteCntE05_OU
obuf
fd
or2
and3b2
FSM_Started
C1_OUT
C256k_OUT
NSRAHR_W_OUT_Int
NSRAHR_W_OUT
NSRAMOE_OUT
EPROM_Data
fdre
inv
gnd
NSRAMCS_OUT
SRAMCS_OUT
NEPROMCS_OUT
nand2b2
nand2b1
or2
m2_1
S0
D1
D0
or5b1
EPROM_Select
and3b1
and2
C1
CLKA
CLKA>>
C256k>>
T_Ack>>
Proc_NSRAHR_W>>
NSRAHR_W>>
Proc_NSRAHCS>>
NSRAMCS>>
SN_Byte>>
NFirstTime>>
TUL_IndexLT5>>
and2b1
CSW_Sel>>
InsertIdle>>
CSW_Value(8:0)>>
Siemens Rolm Comm.
4900 Old Ironsides Drive
Santa Clara
TITLE:
Transmitter
Control FSM
Engineer:
Edmund Chen
Drawn by:
Edmund Chen
Changed by:
edmundc
Date Changed:
Tuesday, October 25, 1994
18:13:31 am
REV
Drawing Number:
Page:
Size
B

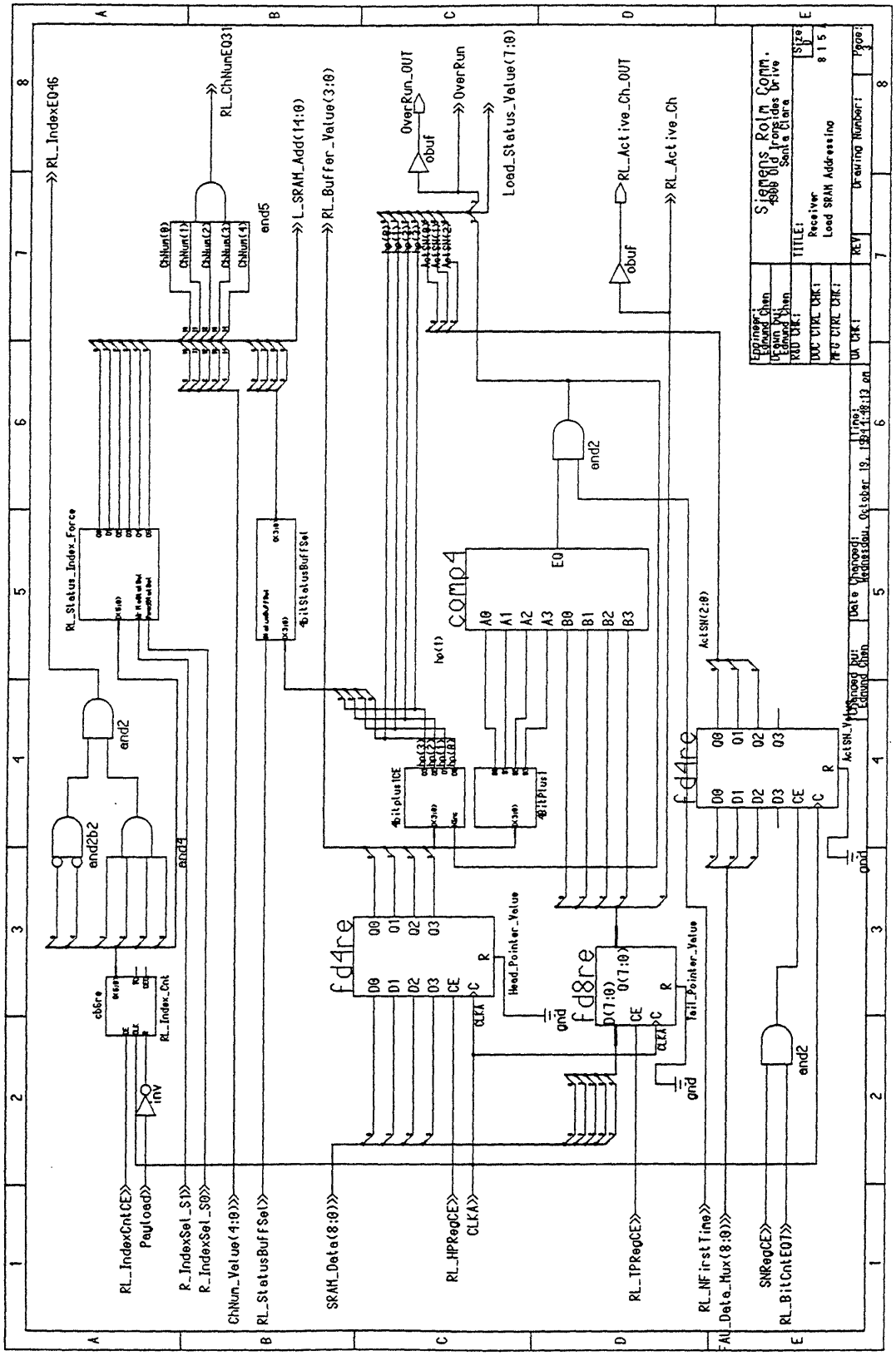

Clock_Gen
C4M_IN
ibuf
gnd
obuf
obuf
CLKA_GEN
FMB_GEN
Pattern_Gen
NCLKA
FMB
obuf
Pattern_Gen_OUT
Changed by: edmundc
Date Changed: Wednesday, December 11, 1996 8:51:11 am
Engineer: Edmund Chen
Drawn by: Edmund Chen
R&D CHK:
DOC CTRL CHK:
MFG CTRL CHK:
QA CHK:
Siemens Rolm Comm.
4900 Old Ironsides Drive
Shata Clara
TITLE:
Transmitter
Extra Components Used for testing
Size
D
8 1 5 4
REV
Drawing Number:
Page:

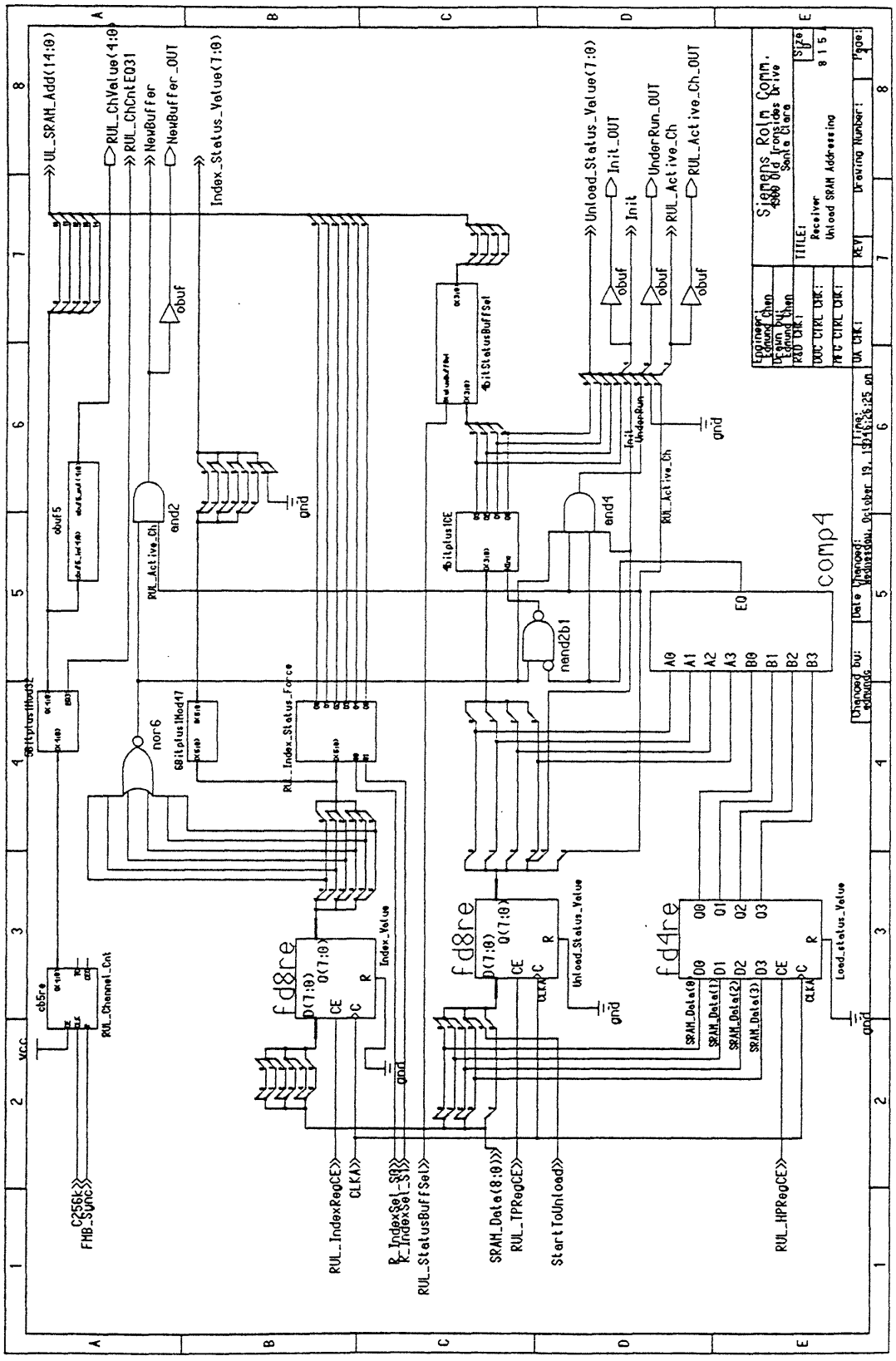
Appendix B Receiver Schematics

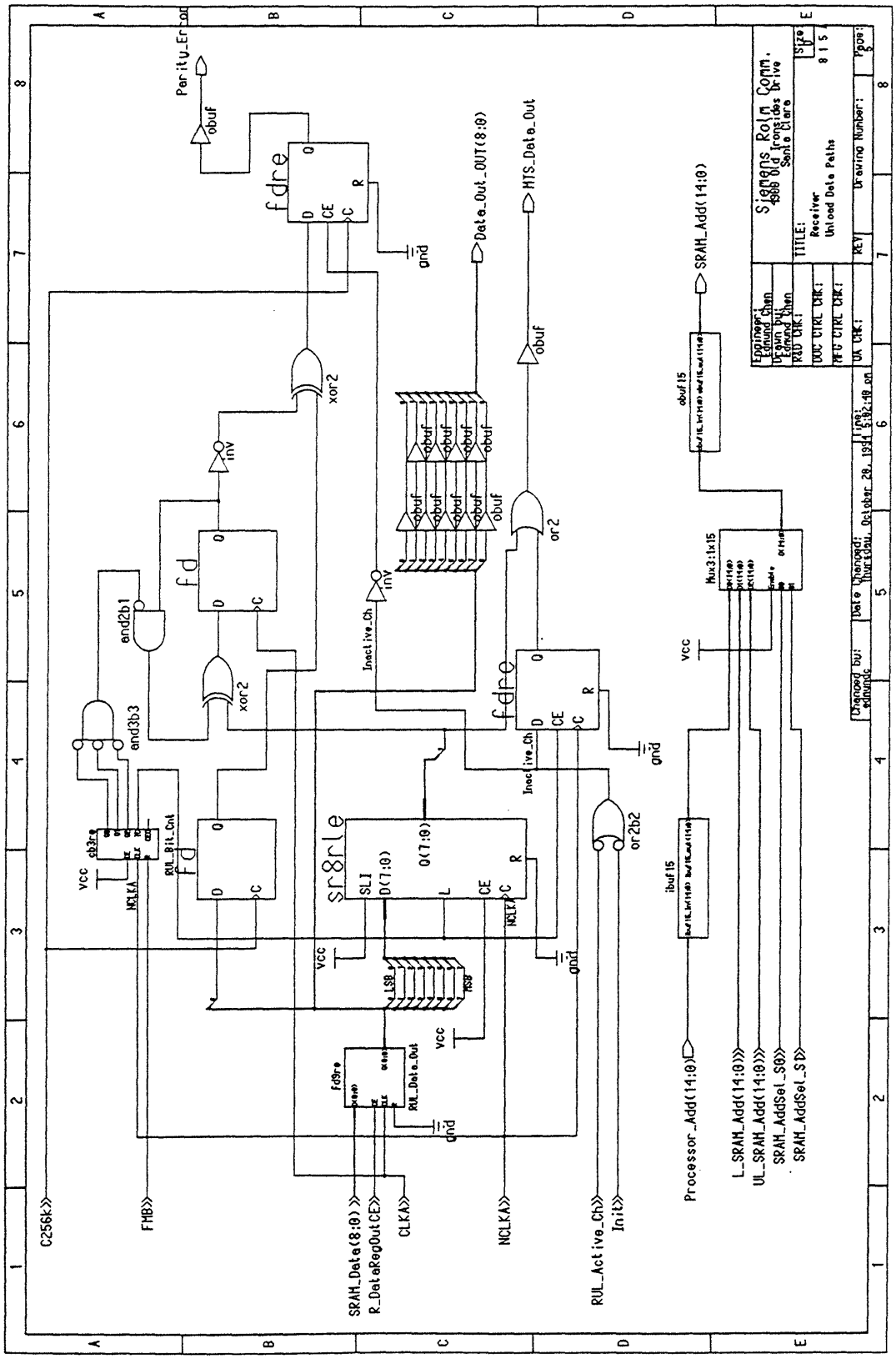
Schematics for Receiver portion of PBX to ATM UNI.

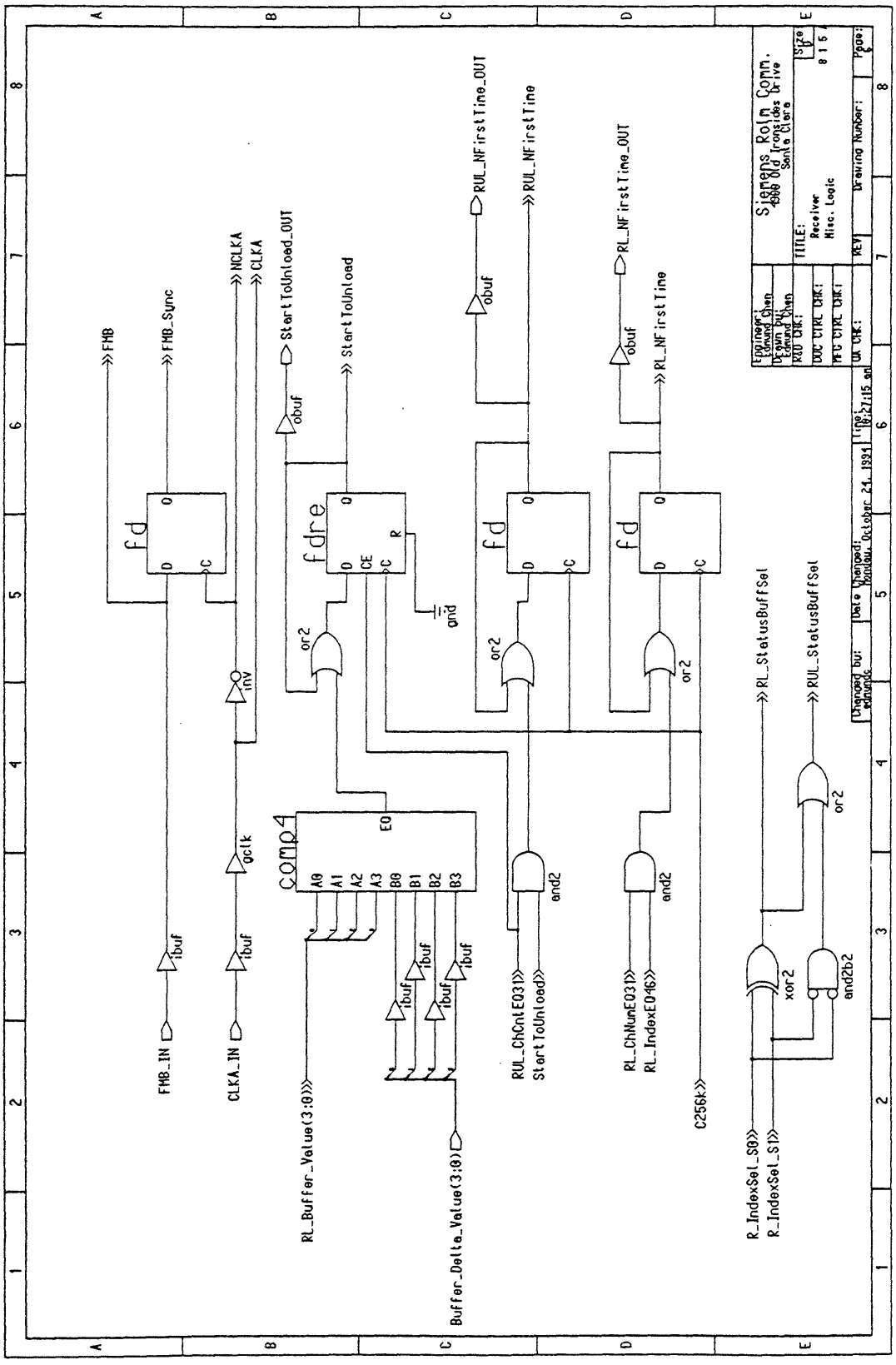

Siemens Rolm Comm.
4900 Old Ironsides Drive
Santa Clara
TITLE:
Receiver
FAU Interface and Cell Delineation
FMB_Sync
FAU_Serial_IN
FAU_Data_Mux(8:0)
FAU_Data(8:0)
Header_OUT
Header
CLKA
cb2re
sr8re
fdre
fd
and2
and2b1
and3
and3b1
or2
xor2
inv
ibuf
obuf
nand2
vcc
gnd

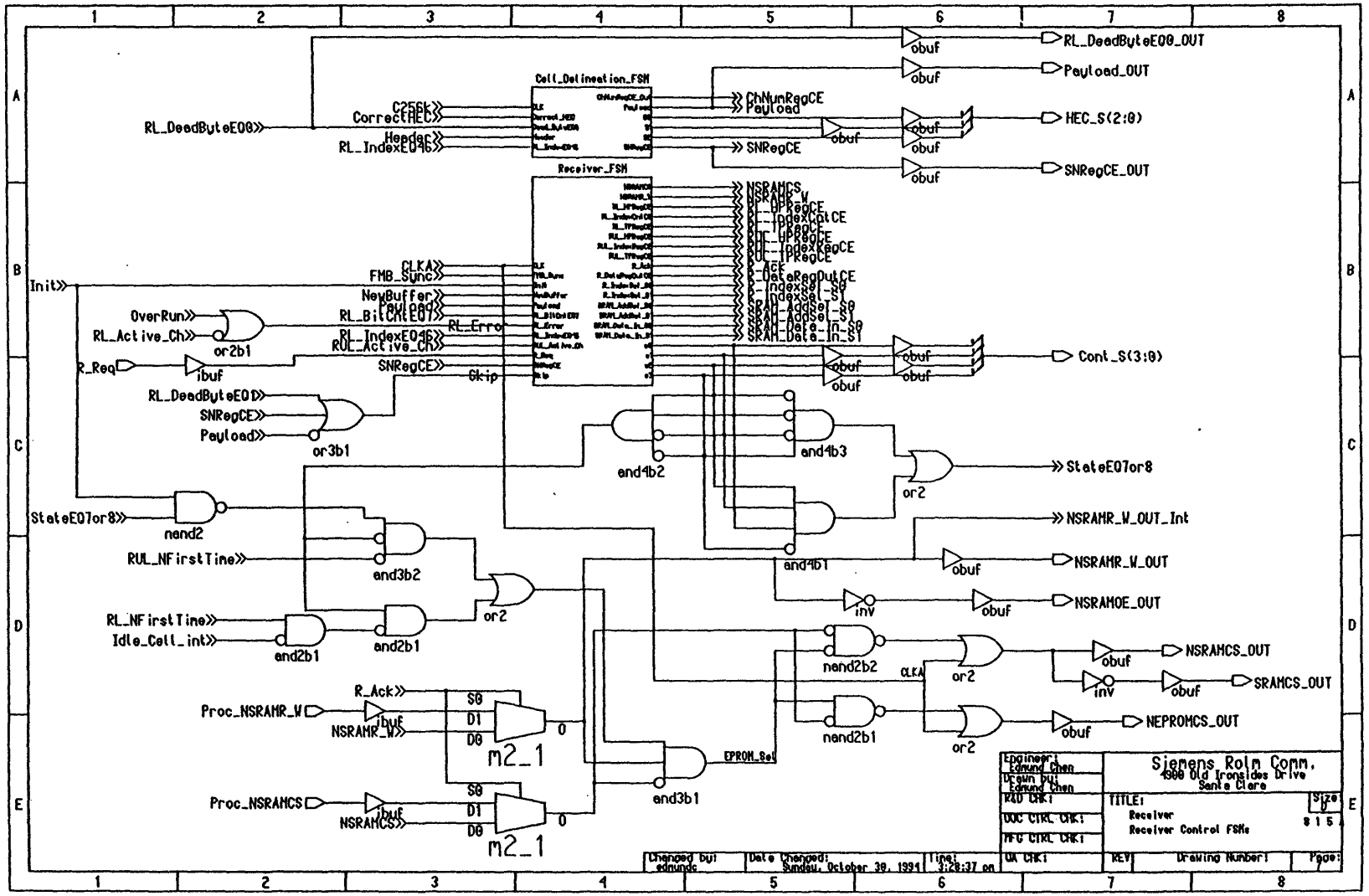

SRAM_Data(8:0)
CorrectHEC_DUT
CorrectHEC
SRAM_Data_Bus(8:0)
Idle_Cell_int
Idle_Cell
ChNum_Value(4:0)
HEC_PROM_Add(4:0)
Head a Tri-State Buffer on SRAM_Data_Bus(8:0)
Tri-StateCE = NSRAM_R_W_OUT
buf
obuf
and2
or2
obuf19
obuf5
nor6
xnor8
vcc
Mux3:1x9
Data_In_Mux
Mux3:1x8
Status_Mux
comp8
A(7:0)
B(7:0)
EQ
HEC_Comp
ibuf8
fd8re
D(7:0)
Q(7:0)
CE
C
R
ChNumReg
gnd
NSRAMR_W_OUT_Int
Processor_Data(8:0)
Index_Status_Value(7:0)
Unload_Status_Value(7:0)
Load_Status_Value(7:0)
R_IndexSel_SR
R_IndexSel_S1
PROM_HEC_Value(7:0)
FAU_Data_Mux(8:0)
SRAM_Data_In_SR
SRAM_Data_In_S1
FAU_Data(8:0)
RL_BitCntEQ7
ChNumRegCE
CLKA
Siemens Rolm Comm.
Old Ironsides Drive
Santa Clara
TITLE:
Receiver
Data and Status Multiplexers
Size
D
REV
Drawing Number
Page:
Engineer:
Edward Chan
Drawn by:
Edward Chan
RAU CHK:
DOC CTRL CHK:
REV CTRL CHK:
QA CHK:
Changed by:
Comments:
Date Changed:
Time Stamp: October 28, 1994 13:39:34 on


RL_IndexEQ16
RL_ChNumEQ31
OverRun_OUT
OverRun
Load_Status_Value(7:0)
RL_Active_Ch_OUT
RL_Active_Ch
L_SRAM_Add(14:0)
RL_Buffer_Value(3:0)
RL_Status_Index_Force
ChNum(0)
ChNum(1)
ChNum(2)
ChNum(3)
ChNum(4)
and5
and2
and2b2
and4
inv
cb5re
RL_Index_Cnt
4bitStatusBuffSel
4bitPlus1CE
4bitPlus1
comp4
fd4re
fd8re
Head_Pointer_Value
Tail_Pointer_Value
gnd
RL_IndexCntCE
Payload
R_IndexSel_S1
R_IndexSel_S0
ChNum_Value(4:0)
RL_StatusBuffSel
SRAM_Data(8:0)
RL_HPRegCE
CLKA
RL_TPRegCE
RL_NFirstTime
RL_Data_Mux(8:0)
SNRegCE
RL_BitCntEQ1
Siemens Rolm Comm.
8300 Old Ironsides Drive
Santa Clara
Receiver
Load SRAM Addressing
Drawing Number:
Page:


UL_SRAM_Add(14:0)
RUL_ChValue(4:0)
RUL_ChCntE031
NewBuffer
NewBuffer_OUT
Index_Status_Value(7:0)
Unload_Status_Value(7:0)
Init_OUT
Init
UnderRun_OUT
RUL_Active_Ch_OUT
RUL_Active_Ch
obuf
obuf5
and2
and4
nand2b1
nor6
gnd
vcc
fd8re
fd4re
comp4
cb5re
RUL_Channel_Cnt
C256k
FMB_Sync
RUL_IndexRegCE
CLKA
R_IndexSel_SR
R_IndexSel_SI
RUL_StatusBuffSel
SRAM_Data(8:0)
RUL_TPRegCE
StartToUnload
RUL_HPRegCE
Index_Value
Unload_Status_Value
Load_status_Value
SRAM_Data(0)
SRAM_Data(1)
SRAM_Data(2)
SRAM_Data(3)
RUL_Index_Status_Force
RUL_Active_Ch
Init
UnderRun
Siemens Rolm Comm.
TITLE:
Receiver
Unload SRAM Addressing
Drawing Number:

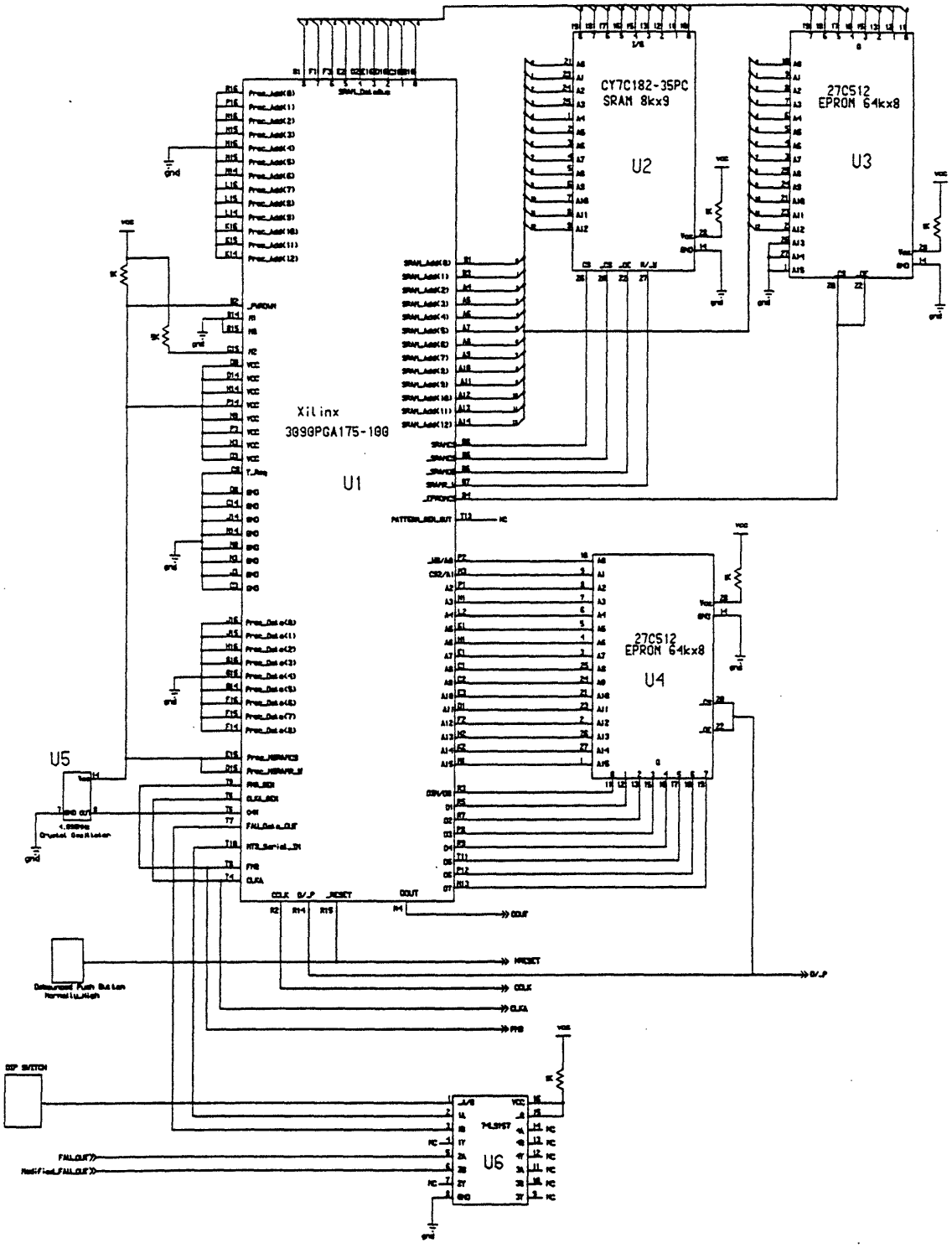

Parity_Error
obuf
fdre
gnd
Data_Out_OUT(8:0)
MIS_Data_Out
SRAM_Add(14:0)
xor2
inv
fd
and2b1
and3b3
or2
or2b2
sr8rle
Q(7:0)
SLI
D(7:0)
RUL_Bit_Cnt
Inactive_Ch
vcc
NCLKA
LSB
MSB
RUL_Data_Out
fdSre
ibuf15
obuf15
mux3:1x15
Processor_Add(14:0)
L_SRAM_Add(14:0)
UL_SRAM_Add(14:0)
SRAM_AddSel_S0
SRAM_AddSel_S1
C256k
FMB
SRAM_Data(8:0)
R_DataRegOutCE
CLKA
NCLKA
RUL_Active_Ch
Init
Siemens Rolm Comm.
4900 Old Ironsides Drive
Santa Clara
TITLE:
Receiver
Unload Data Paths
Engineer:
Edmund Chen
Design Ph:
Edmund Chen
R&D Chk:
DOC CTRL Chk:
MFG CTRL Chk:
QA Chk:
Changed by:
edmundc
Date Changed:
Thursday, October 28, 1993 5:02:48 pm
Drawing Number:
Size
D
REV
Page:
8 1 5

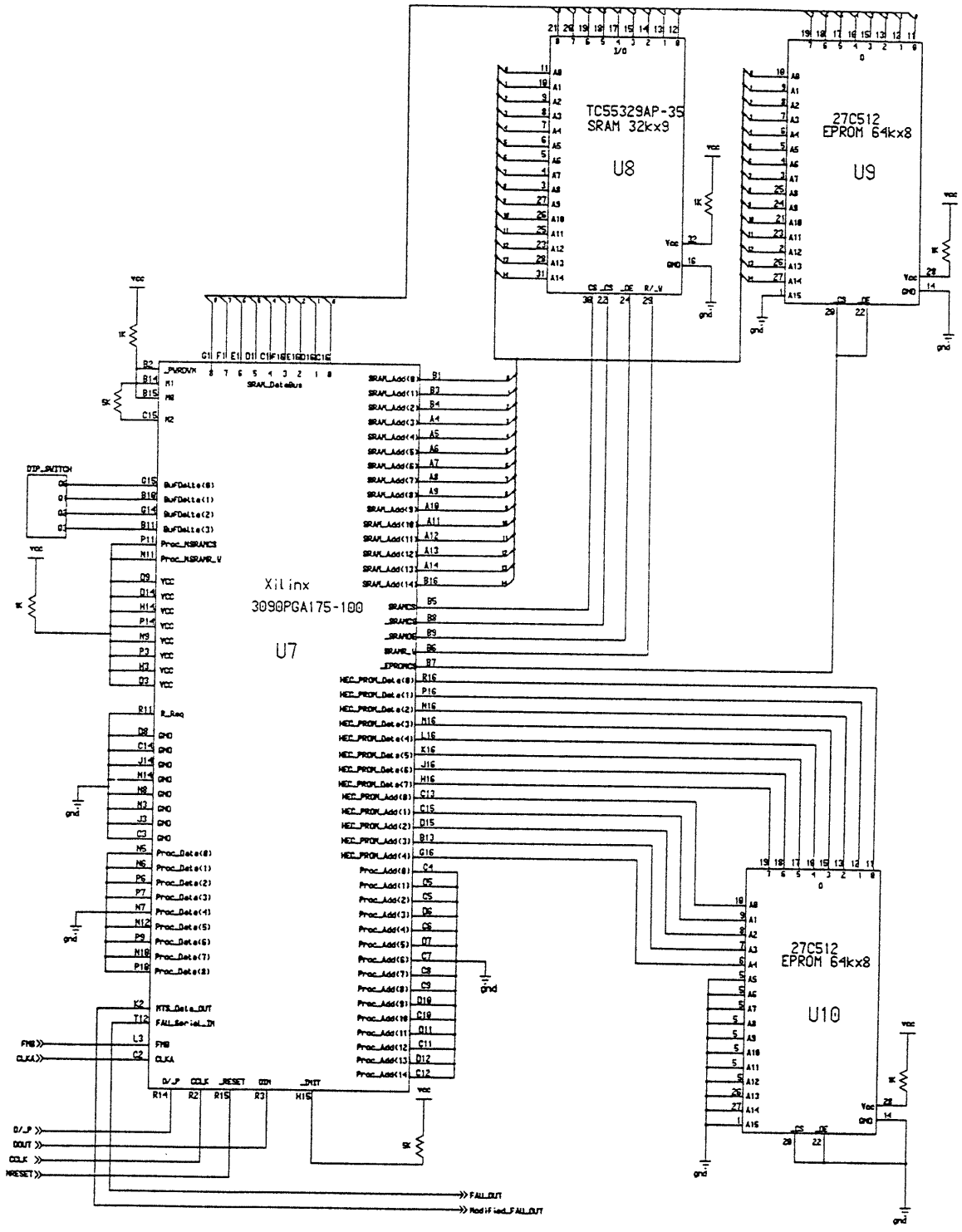

FMB
FMB_Sync
NCLKA
CLKA
StartToUnload_OUT
StartToUnload
RUL_NFirstTime_OUT
RUL_NFirstTime
RL_NFirstTime_OUT
RL_NFirstTime
fd
fdre
obuf
inv
or2
gnd
comp4
A0
A1
A2
A3
B0
B1
B2
B3
EQ
ibuf
gclk
and2
xor2
and2b2
FMB_IN
CLKA_IN
RL_Buffer_Value(3:0)
Buffer_Delta_Value(3:0)
RUL_ChCntE03
StartToUnload
RL_ChNumE03
RL_IndexE04
C256k
R_IndexSel_S0
R_IndexSel_S1
RL_StatusBuffSel
RUL_StatusBuffSel
Siemens Rolm Comm.
4900 Old Ironsides Drive
Santa Clara
TITLE:
Receiver
Misc. Logic
Drawing Number:
Size
D
8 1 5
Page:
8
Changed by:
Date Changed:
Monday, October 24, 1994
Time:
8:27:15 am

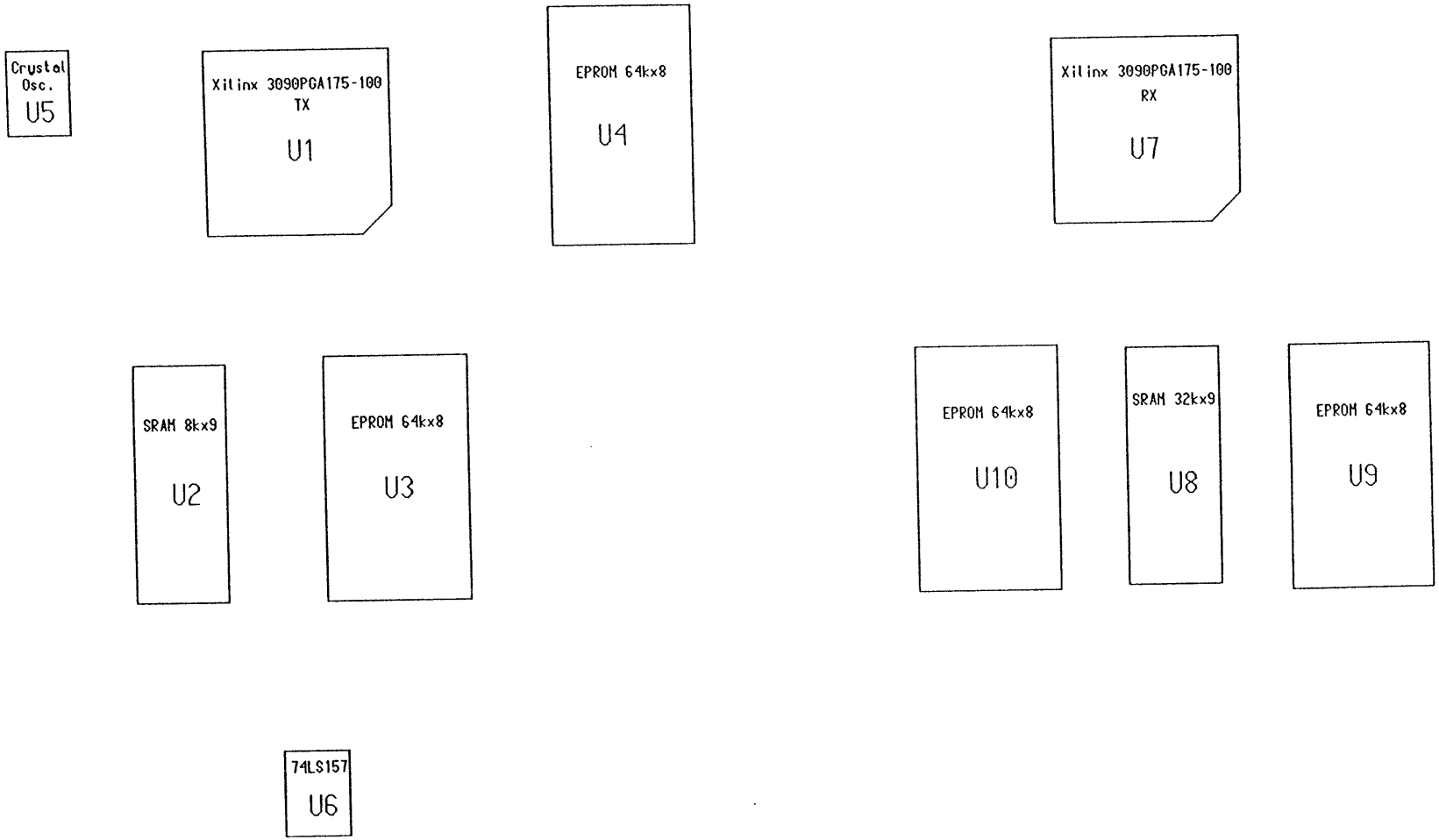

Cell_Delineation_FSM
Receiver_FSM
RL_DeadByteEQ0
Init
CLKA
FMB_Sync
NewBuffer
Payload
RL_BitCntEQ1
RL_IndexEQ46
RUL_Active_Ch
SNRegCE
Ckip
RL_Error
OverRun
RL_Active_Ch
or2b1
R_Req
ibuf
RL_DeadByteEQ1
SNRegCE
Payload
or3b1
StateEQ7or8
nand2
RUL_NFirstTime
and3b2
RL_NFirstTime
Idle_Cell_int
and2b1
and2b1
or2
and4b2
and4b3
or2
and4b1
inv
R_Ack
Proc_NSRAMR_W
NSRAMR_W
Proc_NSRAMCS
NSRAMCS
S0
D1
D0
O
m2_1
m2_1
and3b1
EPROM_Sel
nand2b2
nand2b1
CLKA
or2
or2
inv
ChNumRegCE
Payload
SNRegCE
HEC_S(2:0)
SNRegCE_OUT
Cont_S(3:0)
RL_DeadByteEQ0_OUT
Payload_OUT
StateEQ7or8
NSRAMR_W_OUT_Int
NSRAMR_W_OUT
NSRAMOE_OUT
NSRAMCS_OUT
SRAMCS_OUT
NEPROMCS_OUT
obuf
inv
Siemens Rolm Comm.
4900 Old Ironsides Drive
Santa Clara
Engineer: Edmund Chen
Drawn by: Edmund Chen
R&D CHK:
DOC CTRL CHK:
MFG CTRL CHK:
QA CHK:
TITLE: Receiver
Receiver Control FSMs
Size: D
8 1 5
Changed by: edmundc
Date Changed: Sunday, October 30, 1994
Time: 3:28:37 pm
REV
Drawing Number:
Page:

Appendix C Board Layout

Physical connections and layout for the PBX to ATM UNI board.


CY7C182-35PC
SRAM 8kx9
U2
27C512
EPROM 64kx8
U3
Xilinx
3090PGA175-100
U1
27C512
EPROM 64kx8
U4
U5
U6
74LS157
DIP SWITCH


TC5329AP-35
SRAM 32kx9
U8
27C512
EPROM 64kx8
U9
XiLinx
3090PGA175-100
U7
27C512
EPROM 64kx8
U10
vcc
gnd
1K
5K
DIP_SWITCH
SRAM_DataBus
SRAM_Add(0)
SRAM_Add(1)
SRAM_Add(2)
SRAM_Add(3)
SRAM_Add(4)
SRAM_Add(5)
SRAM_Add(6)
SRAM_Add(7)
SRAM_Add(8)
SRAM_Add(9)
SRAM_Add(10)
SRAM_Add(11)
SRAM_Add(12)
SRAM_Add(13)
SRAM_Add(14)
_SRAMCS
_SRAMOE
SRAM_W
_EPROMCS
BufData(0)
BufData(1)
BufData(2)
BufData(3)
Proc_NSRAMCS
Proc_NSRAMR_W
R_Req
Proc_Data(0)
Proc_Data(1)
Proc_Data(2)
Proc_Data(3)
Proc_Data(4)
Proc_Data(5)
Proc_Data(6)
Proc_Data(7)
Proc_Data(8)
MTSL_Data_OUT
FAIL_Serial_IN
FMB
CLKA
D/_P
DOUT
CCLK
_RESET
_INIT
FAIL_OUT
Modified_FAIL_OUT


Crystal Osc.
U5
Xilinx 3090PGA175-100
TX
U1
EPROM 64kx8
U4
Xilinx 3090PGA175-100
RX
U7
SRAM 8kx9
U2
EPROM 64kx8
U3
EPROM 64kx8
U10
SRAM 32kx9
U8
EPROM 64kx8
U9
74LS157
U6

Appendix D Transmitter Initialize EPROM Contents

This is the data programmed into Transmitter Initialize EPROM:

Key:

xxC0 to XXC5 are the ATM cell header bytes and the initial SN byte.
xxFF is the CSW byte value. A CSW = 00 indicates an inactive channel and
a CSW = 80 indicates an active channel.
aaaa/dd; is the address/data format. Both values are in hex.

C0/00;
C1/00;
C2/02;
C3/50;
C4/FB;
C5/01;
FF/80;
1C0/00;
1C1/00;
1C2/02;
1C3/60;
1C4/F2;
1C5/01;
1FF/80;
2C0/00;
2C1/00;
2C2/02;
2C3/70;
2C4/F5;
2C5/01;
2FF/80;
3C0/00;
3C1/00;
3C2/02;
3C3/90;
3C4/DF;
3C5/01;
3FF/80;
4C0/00;
4C1/00;
4C2/02;
4C3/A0;
4C4/D6;
4C5/01;
4FF/80;
5C0/00;
5C1/00;
5C2/02;
5C3/B0;
5C4/D1;
5C5/01;
5FF/80;
6C0/00;
6C1/00;
6C2/02;
6C3/D0;
6C4/C3;
6C5/01;
6FF/80;
7C0/00;
7C1/00;
7C2/02;
7C3/E0;
7C4/65;
7C5/01;
7FF/80;
8C0/00;
8C1/00;
8C2/02;
8C3/F0;
8C4/C5;
8C5/01;
8FF/80;
9C0/00;
9C1/00;
9C2/03;
9C3/10;
9C4/97;
9C5/01;
9FF/80;
AC0/00;
AC1/00;
AC2/03;
AC3/20;
AC4/9E;

AC5/01;
AFF/80;
BC0/00;
BC1/00;
BC2/03;
BC3/30;
BC4/99;
BC5/01;
BFF/80;
CC0/00;
CC1/00;
CC2/03;
CC3/50;
CC4/8B;
CC5/01;
CFF/80;
DC0/00;
DC1/00;
DC2/03;
DC3/60;
DC4/82;
DC5/01;
DFF/80;
EC0/00;
EC1/00;
EC2/03;
EC3/70;
EC4/85;
EC5/01;
EFF/80;
FC0/00;
FC1/00;
FC2/03;
FC3/90;
FC4/AF;
FC5/01;
FFF/80;
10C0/00;
10C1/00;
10C2/03;
10C3/A0;
10C4/A6;
10C5/01;
10FF/80;
11C0/00;
11C1/00;
11C2/03;
11C3/B0;
11C4/A1;
11C5/01;
11FF/80;
12C0/00;
12C1/00;
12C2/03;
12C3/D0;
12C4/B3;
12C5/01;
12FF/80;
13C0/00;
13C1/00;
13C2/03;
13C3/E0;
13C4/BA;
13C5/01;
13FF/80;
14C0/00;
14C1/00;
14C2/03;
14C3/F0;
14C4/B5;
14C5/01;
14FF/80;
1FC0/00;
1FC1/00;
1FC2/00;
1FC3/01;
1FC4/52;
1FC5/6A;
1FC6/6A;
1FC7/6A;
1FC8/6A;
1FC9/6A;
1FCA/6A;
1FCB/6A;
1FCC/6A;
1FCD/6A;
1FCE/6A;
1FCF/6A;
1FD0/6A;
1FD1/6A;
1FD2/6A;
1FD3/6A;
1FD4/6A;
1FD5/6A;
1FD6/6A;
1FD7/6A;
1FD8/6A;
1FD9/6A;
1FDA/6A;
1FDB/6A;
1FDC/6A;
1FDD/6A;
1FDE/6A;
1FDF/6A;
1FE0/6A;
1FE1/6A;
1FE2/6A;
1FE3/6A;
1FE4/6A;
1FE5/6A;
1FE6/6A;
1FE7/6A;
1FE8/6A;
1FE9/6A;
1FEA/6A;
1FEB/6A;
1FEC/6A;
1FED/6A;
1FEE/6A;
1FEF/6A;
1FF0/6A;
1FF1/6A;
1FF2/6A;
1FF3/6A;
1FF4/6A;
1FFF/80;

Appendix E Receiver Initialize EPROM Contents

This is the data programmed into the Receiver Initialize EPROM:

Key:

xxFD is the address location for the initial Index Status Value.
xxFE is the address location for the initial Tail Pointer Value.
xxFF is the address location for the initial Head Pointer Value.
aaaa/dd; is the address/data format. Both values are in hex.

3FD/00;	27FD/00;	4BFD/00;
03FE/00;	27FE/80;	4BFE/80;
03FF/70;	27FF/70;	4BFF/70;
07FD/00;	2BFD/00;	4FFD/00;
07FE/00;	2BFE/80;	4FFE/80;
07FF/70;	2BFF/70;	4FFF/70;
0BFD/00;	2FFD/00;	53FD/00;
0BFE/00;	2FFE/80;	53FE/00;
0BFF/70;	2FFF/70;	53FF/70;
0FFD/00;	33FD/00;	57FD/00;
0FFE/00;	33FE/00;	57FE/80;
0FFF/70;	33FF/70;	57FF/70;
13FD/00;	37FD/00;	5BFD/00;
13FE/00;	37FE/80;	5BFE/80;
13FF/70;	37FF/70;	5BFF/70;
17FD/00;	3BFD/00;	5FFD/00;
17FE/80;	3BFE/80;	5FFE/80;
17FF/70;	3BFF/70;	5FFF/70;
1BFD/00;	3FFD/00;	63FD/00;
1BFE/80;	3FFE/80;	63FE/00;
1BFF/70;	3FFF/70;	63FF/70;
1FFD/00;	43FD/00;	67FD/00;
1FFE/80;	43FE/00;	67FE/80;
1FFF/70;	43FF/70;	67FF/70;
23FD/00;	47FD/00;	6BFD/00;
23FE/00;	47FE/80;	6BFE/80;
23FF/70;	47FF/70;	6BFF/70;

6FFD/00;
6FFE/80;
6FFF/70;
73FD/00;
73FE/00;
73FF/70;
77FD/00;
77FE/80;
77FF/70;
7BFD/00;
7BFE/80;
7BFF/70;
7FFD/00;
7FFE/80;
7FFF/70;

Appendix F Receiver HEC EPROM Contents

This is the data programmed into the Receiver HEC EPROM:

Key:

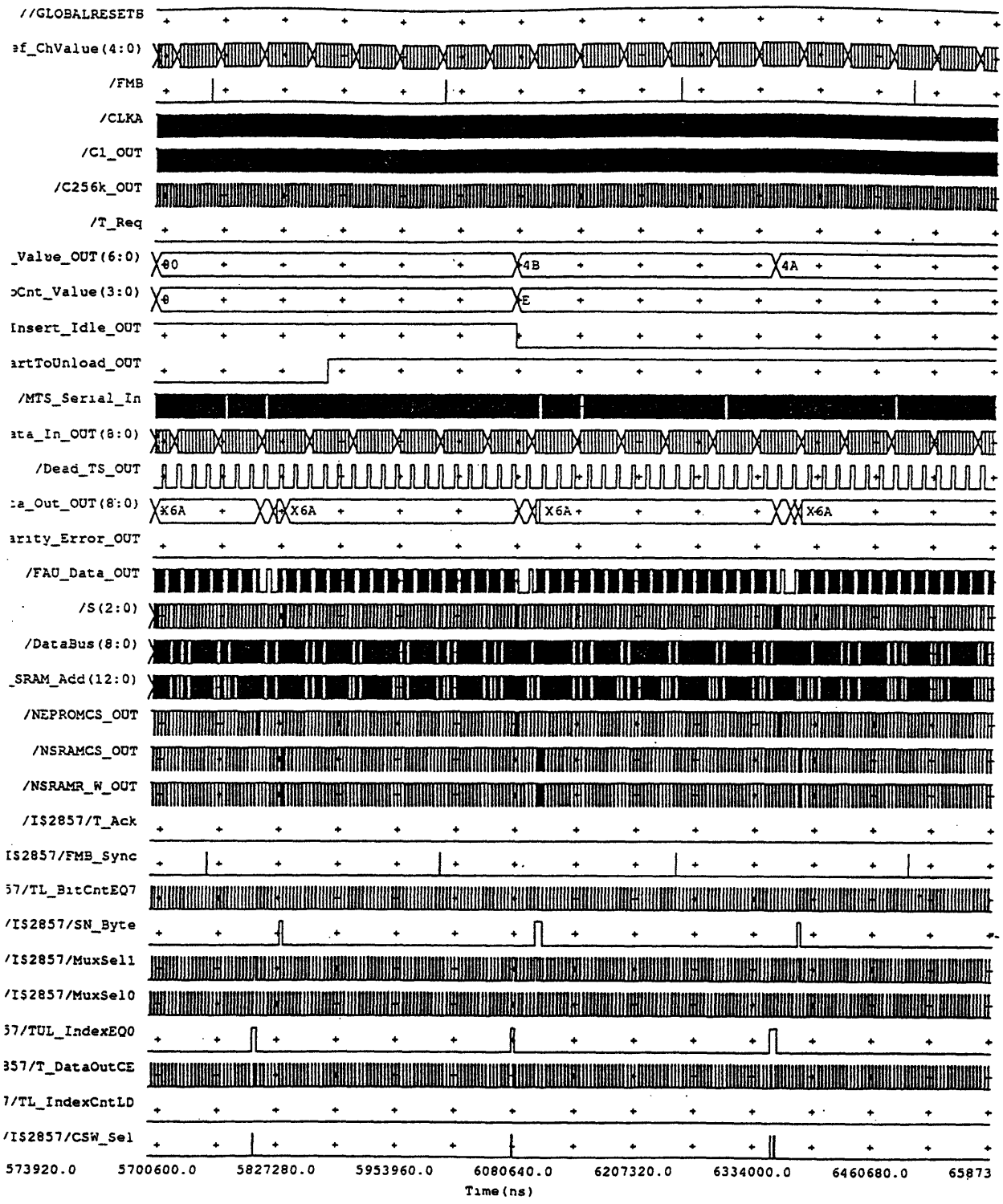
aaaa/dd; is the address/data format. Both values are in hex.

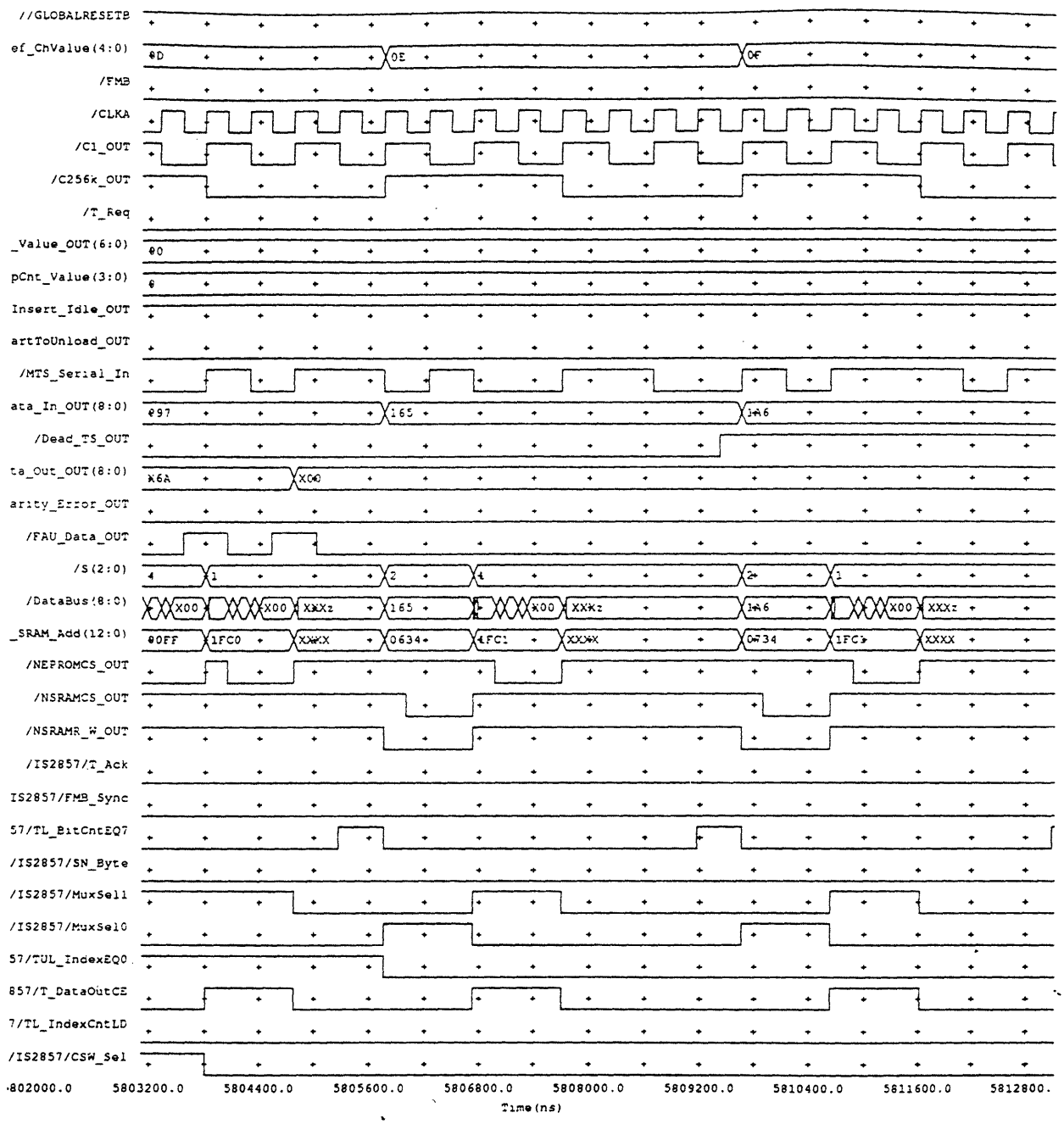
00/52;
01/E7;
02/EE;
03/E9;
04/FC;
05/FB;
06/F2;
07/F5;
08/D8;
09/DF;
0A/D6;
0B/D1;
0C/C4;
0D/C3;
0E/65;
0F/C5;
10/90;
11/97;
12/9E;
13/99;
14/8C;
15/8B;
16/80;
17/85;
18/A8;
19/AF;
1A/A6;
1B/A1;
1C/B4;
1D/B3;
1E/BA;
1F/B5;

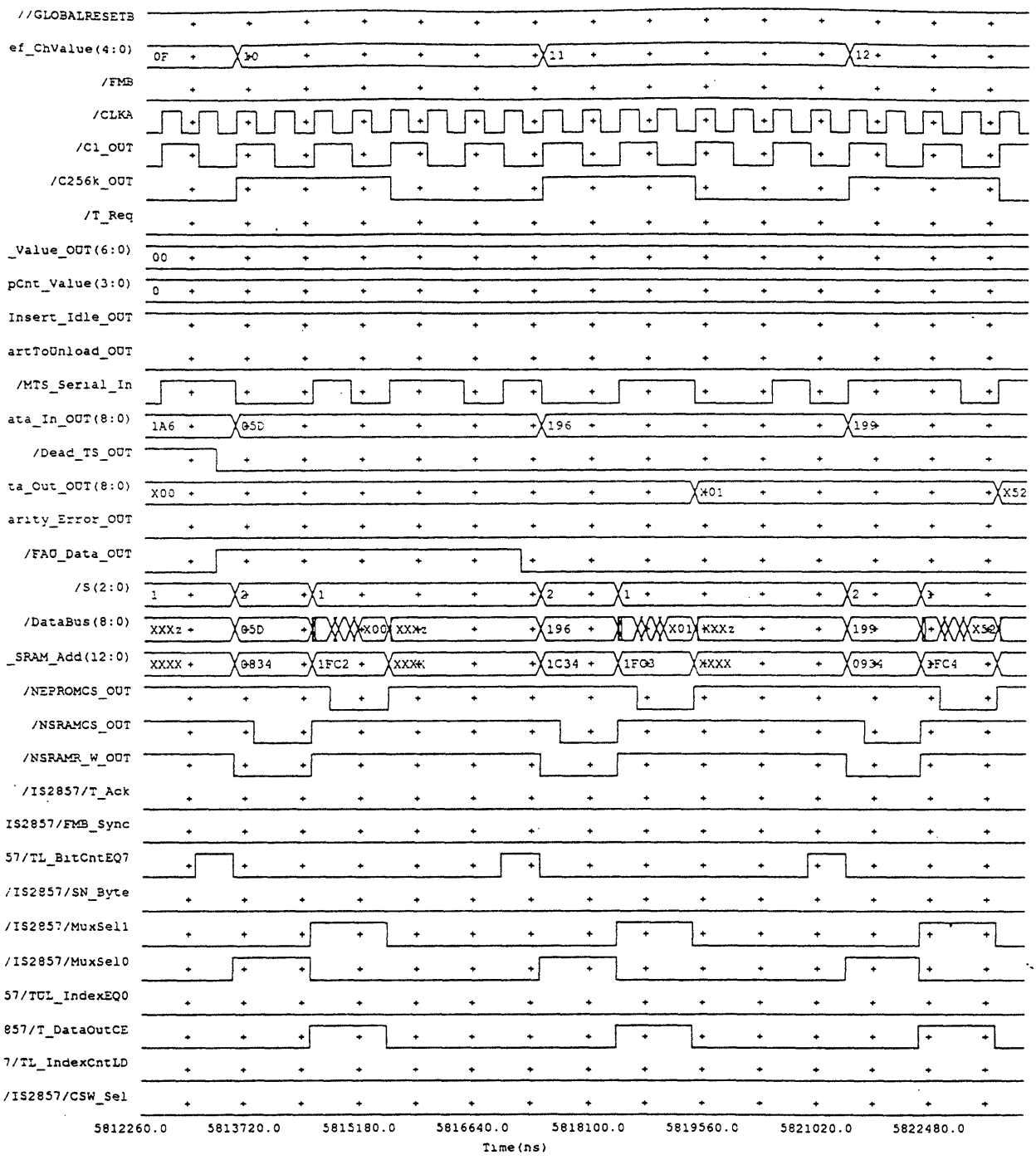
Appendix G Transmitter Simulation

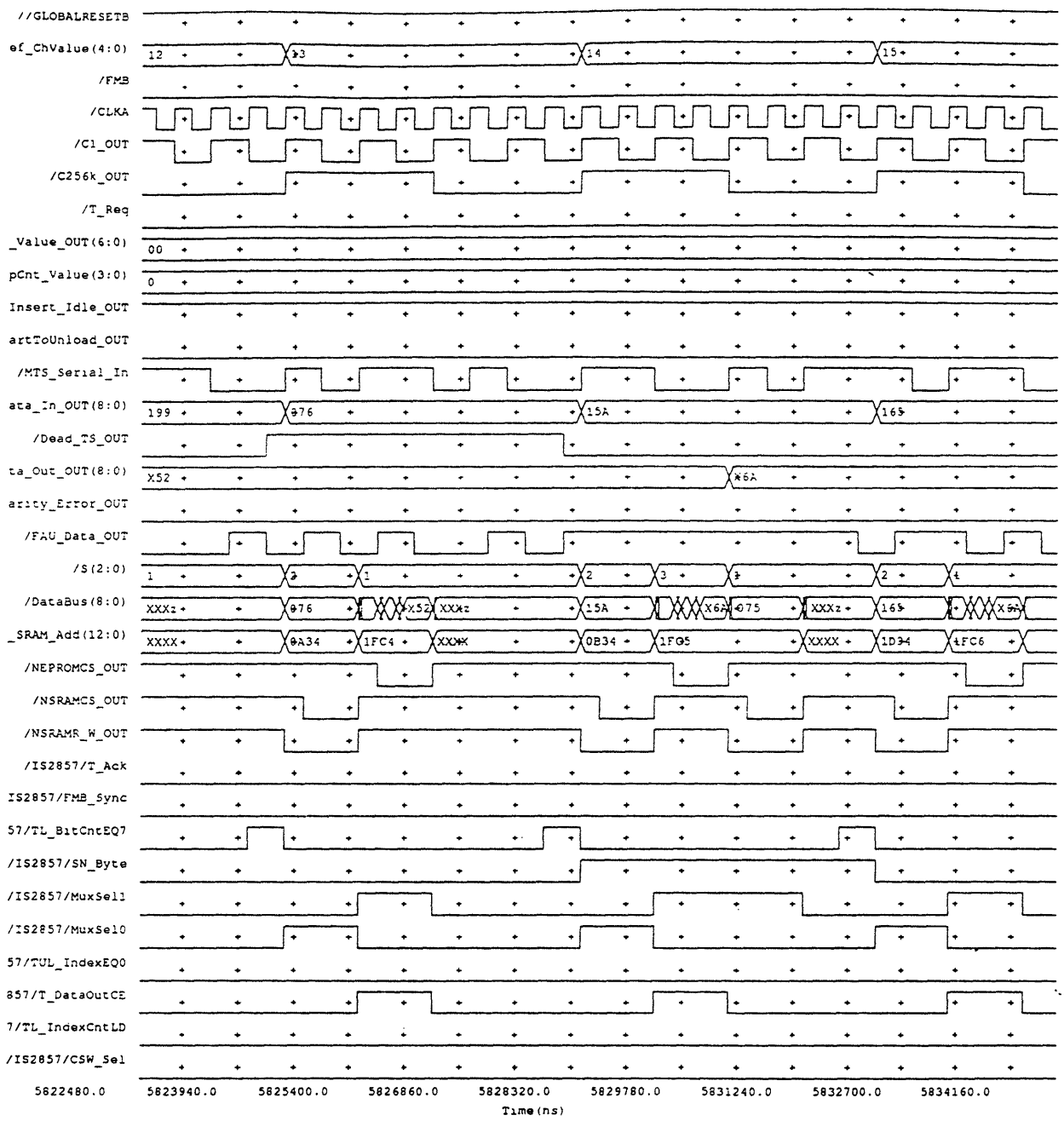
The timing simulation shown in this Appendix is for the Transmitter. Only time slots 27 and 31 are active.

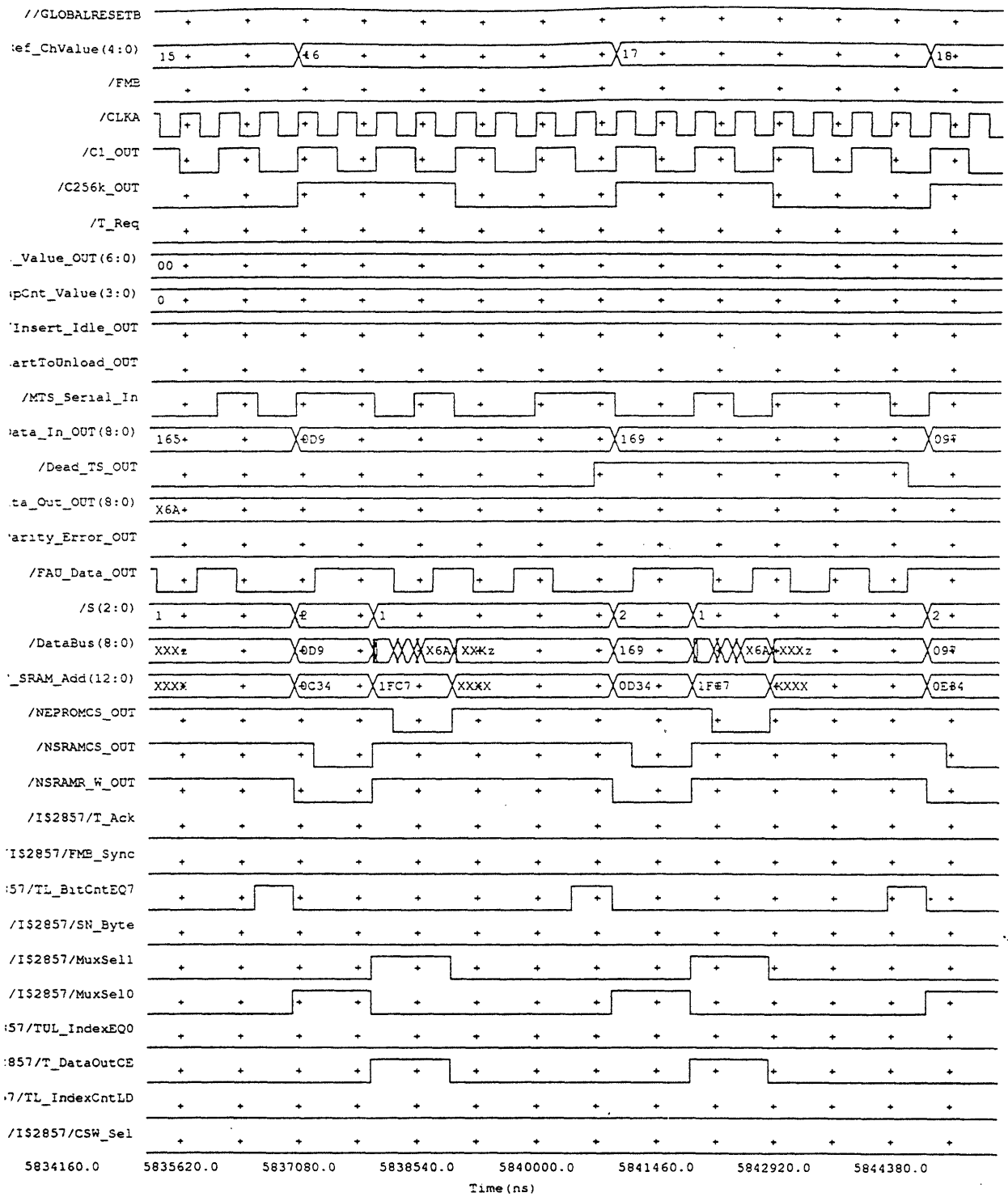
Simulation Time Period	Page(s)	Simulation Description
5700000 - 6600000	124	Transition of StartToUnload signal
5800000 - 5860000	125-129	Transmission of Idle ATM Cells, StartToUnload = 0 and InsertIdle = 1
10760000 - 11080000	130-132	Overview of transmission of an ATM cell for channel 27
10780000 - 1083000	133-137	Beginning of the unloading for an ATM cell for channel 27
26770000 - 26830000	138-142	Insertion of an Idle ATM Cell, StartToUnload = 1 and InsertIdle = 1

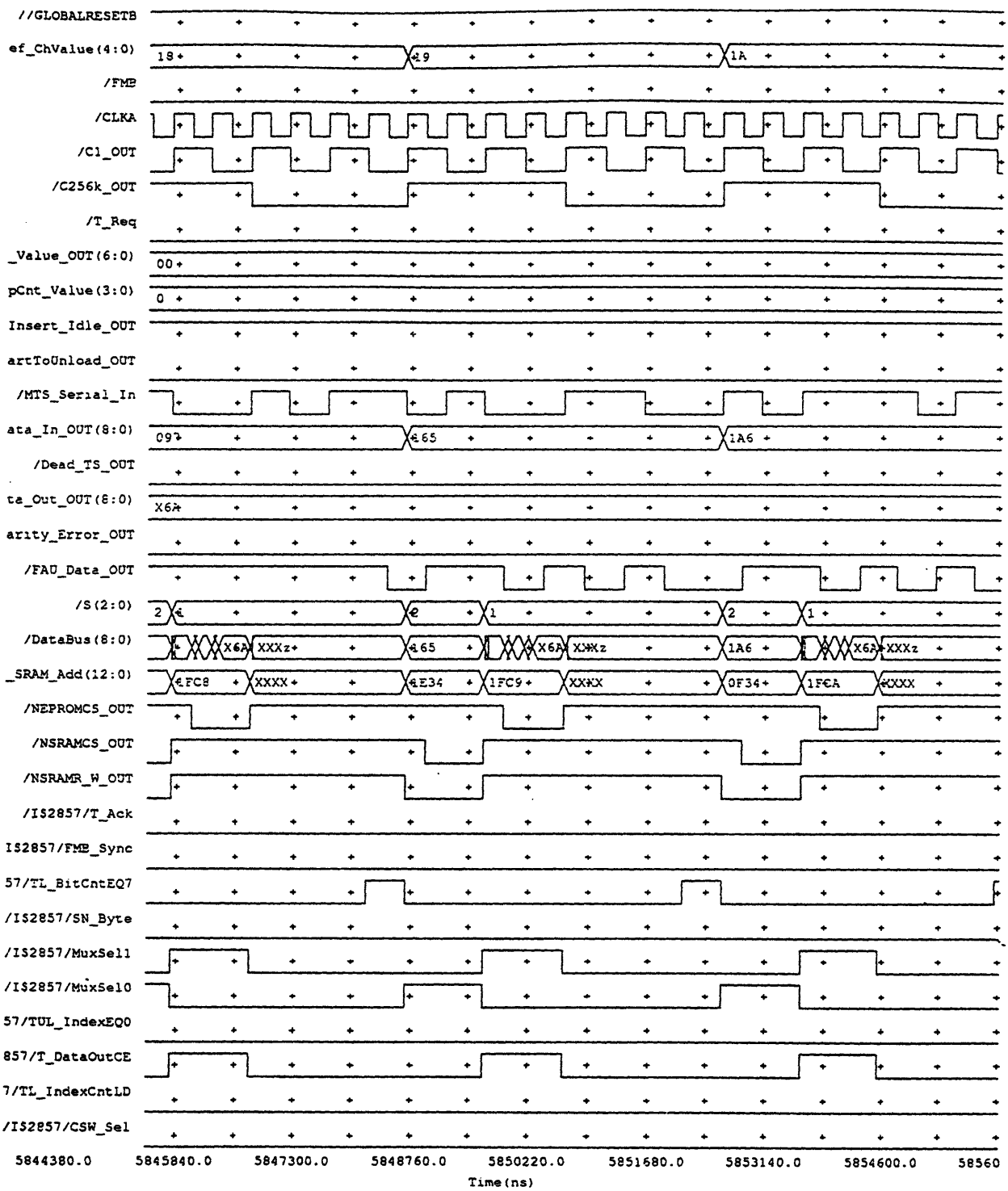

//GLOBALRESETB
af_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
0B
4B
4A
oCnt_Value(3:0)
0
E
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
:a_Out_OUT(8:0)
X6A
X6A
X6A
X6A
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
357/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
573920.0
5700600.0
5827280.0
5953960.0
6080640.0
6207320.0
6334000.0
6460680.0
65873
Time(ns)

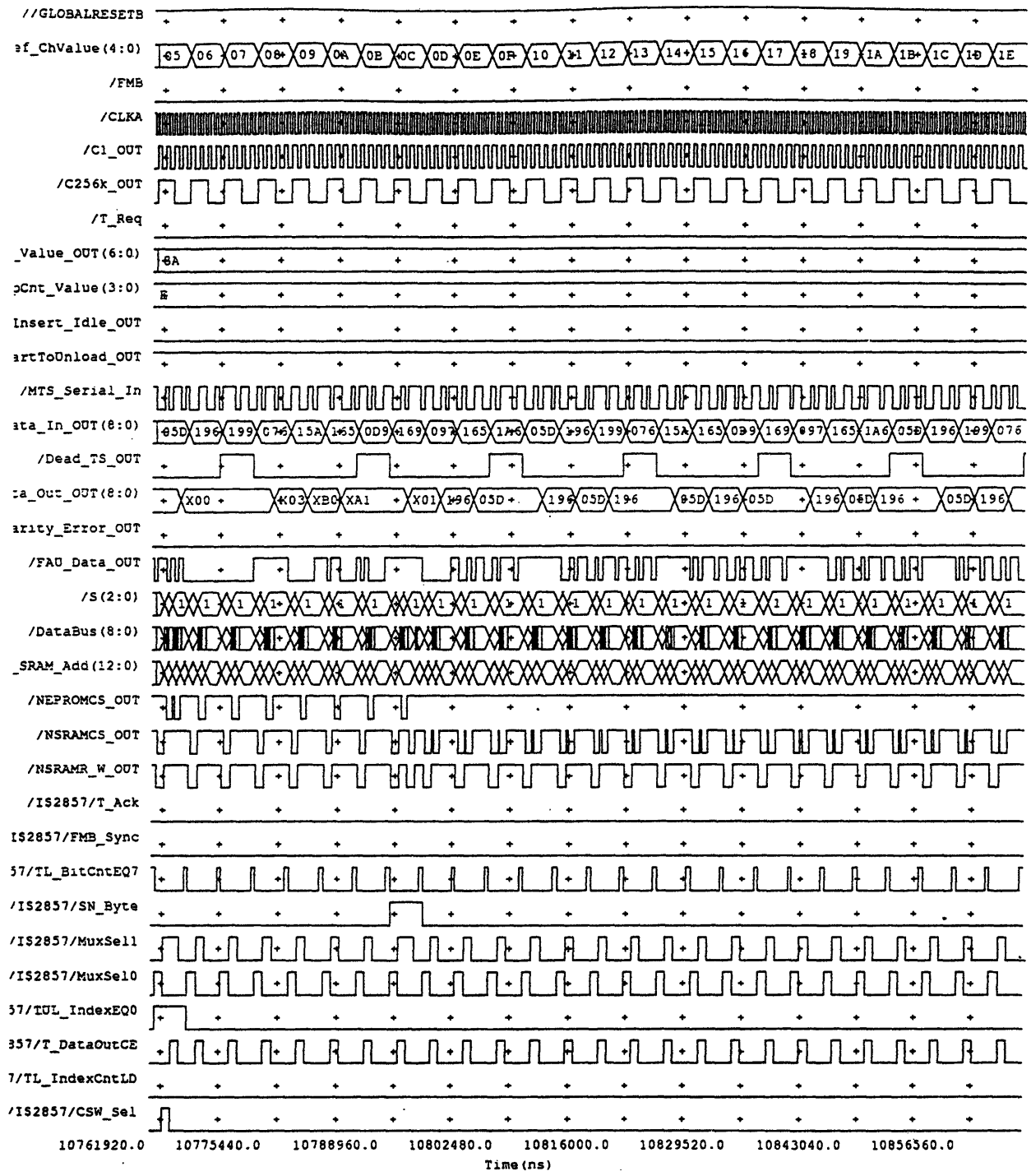

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
5802000.0
5803200.0
5804400.0
5805600.0
5806800.0
5808000.0
5809200.0
5810400.0
5811600.0
5812800.
Time(ns)

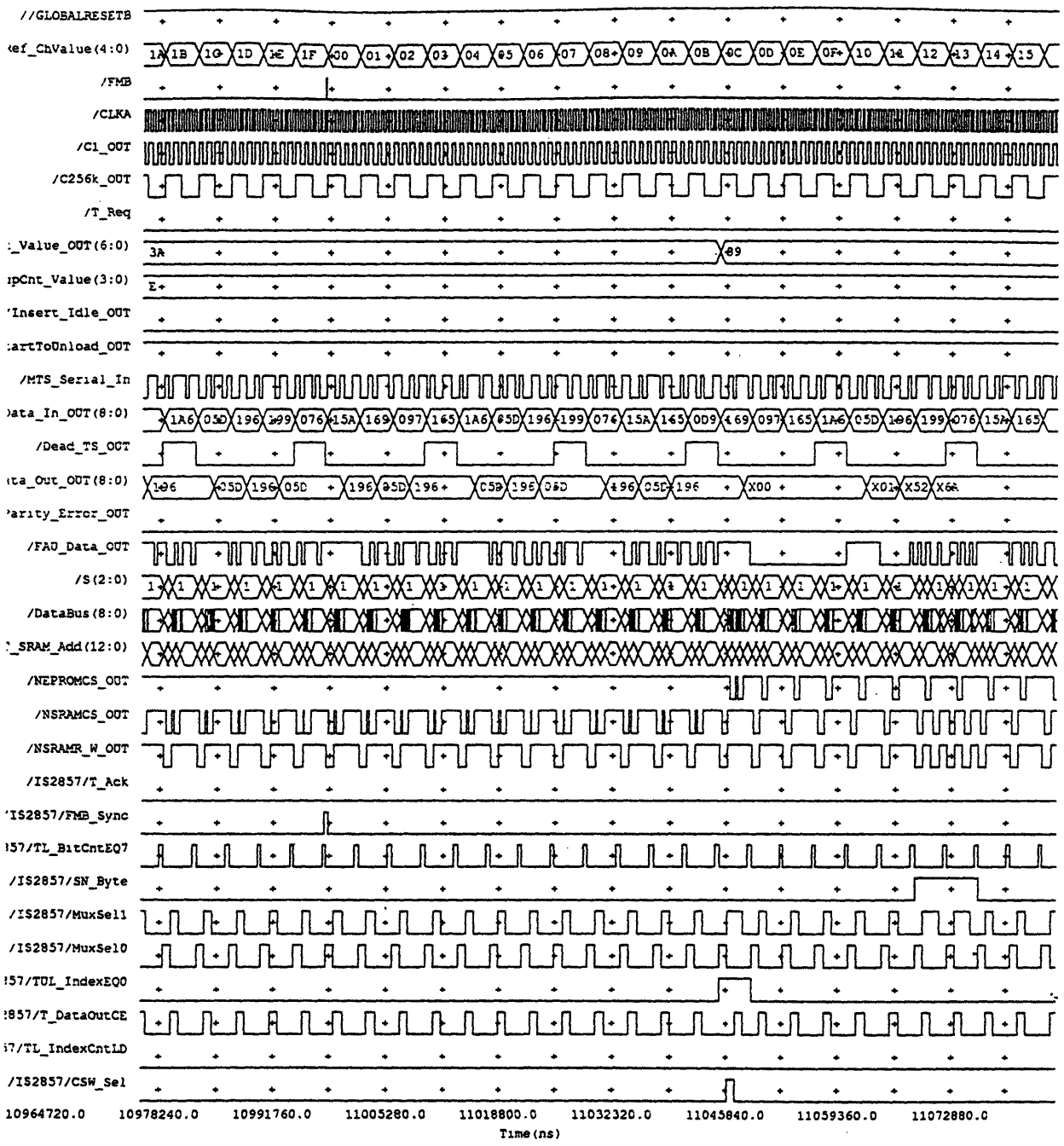

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
5812260.0
5813720.0
5815180.0
5816640.0
5818100.0
5819560.0
5821020.0
5822480.0
Time(ns)

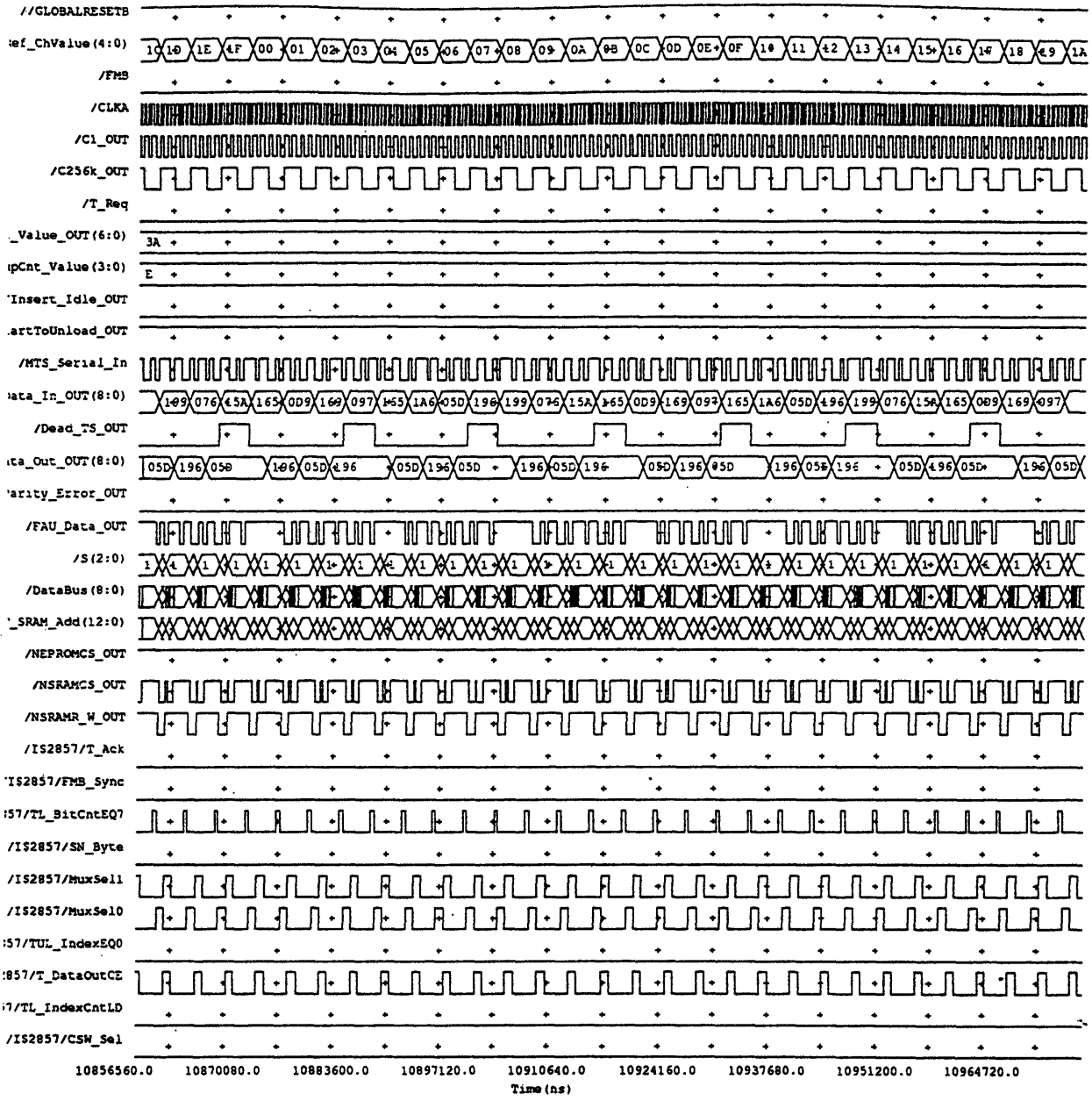

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
5822480.0
5823940.0
5825400.0
5826860.0
5828320.0
5829780.0
5831240.0
5832700.0
5834160.0
Time(ns)

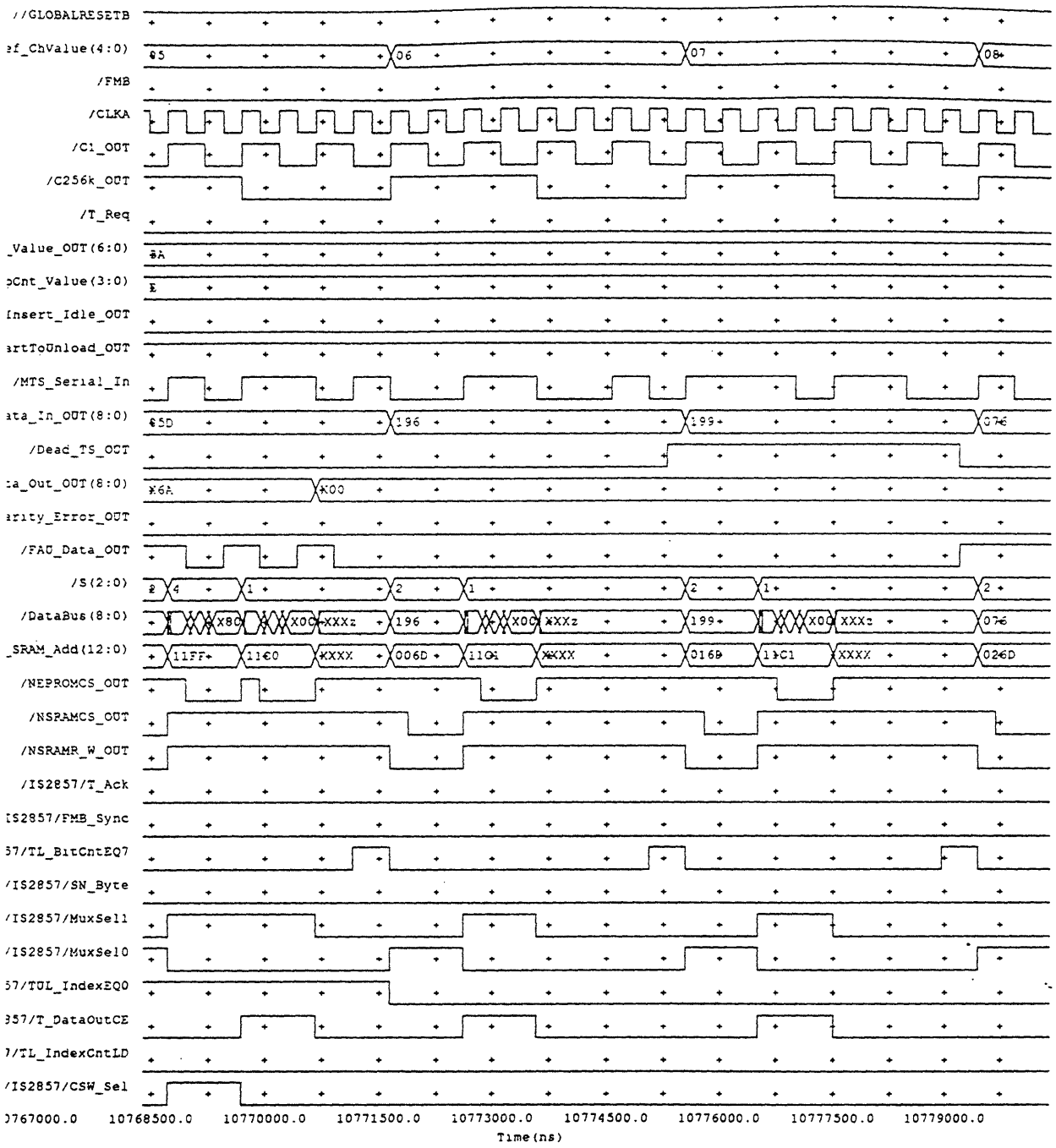

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/I$2857/T_Ack
I$2857/FMB_Sync
57/TL_BitCntEQ7
/I$2857/SN_Byte
/I$2857/MuxSel1
/I$2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/I$2857/CSW_Sel
5834160.0
5835620.0
5837080.0
5838540.0
5840000.0
5841460.0
5842920.0
5844380.0
Time(ns)

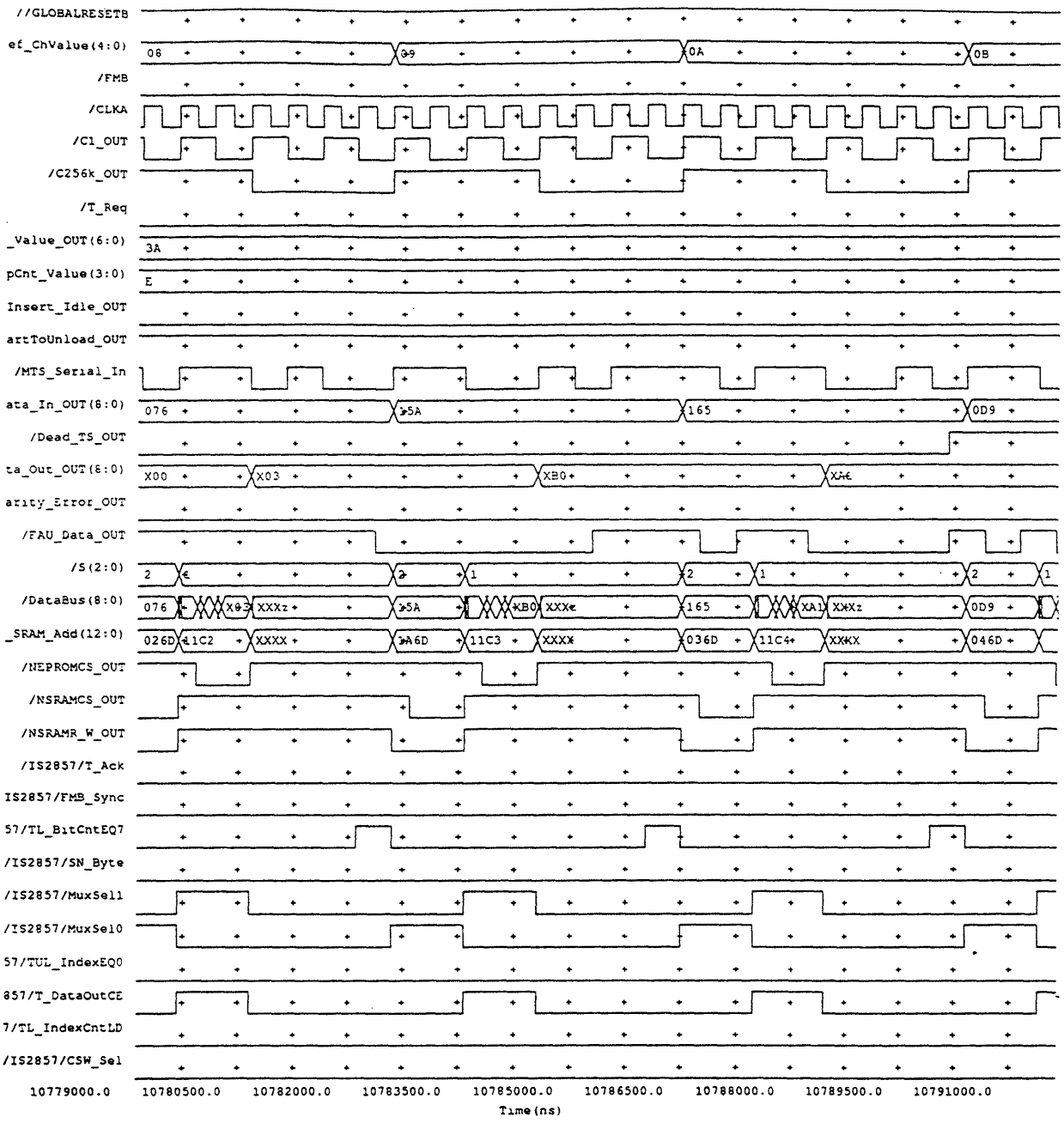

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
5844380.0
5845840.0
5847300.0
5848760.0
5850220.0
5851680.0
5853140.0
5854600.0
58560
Time(ns)

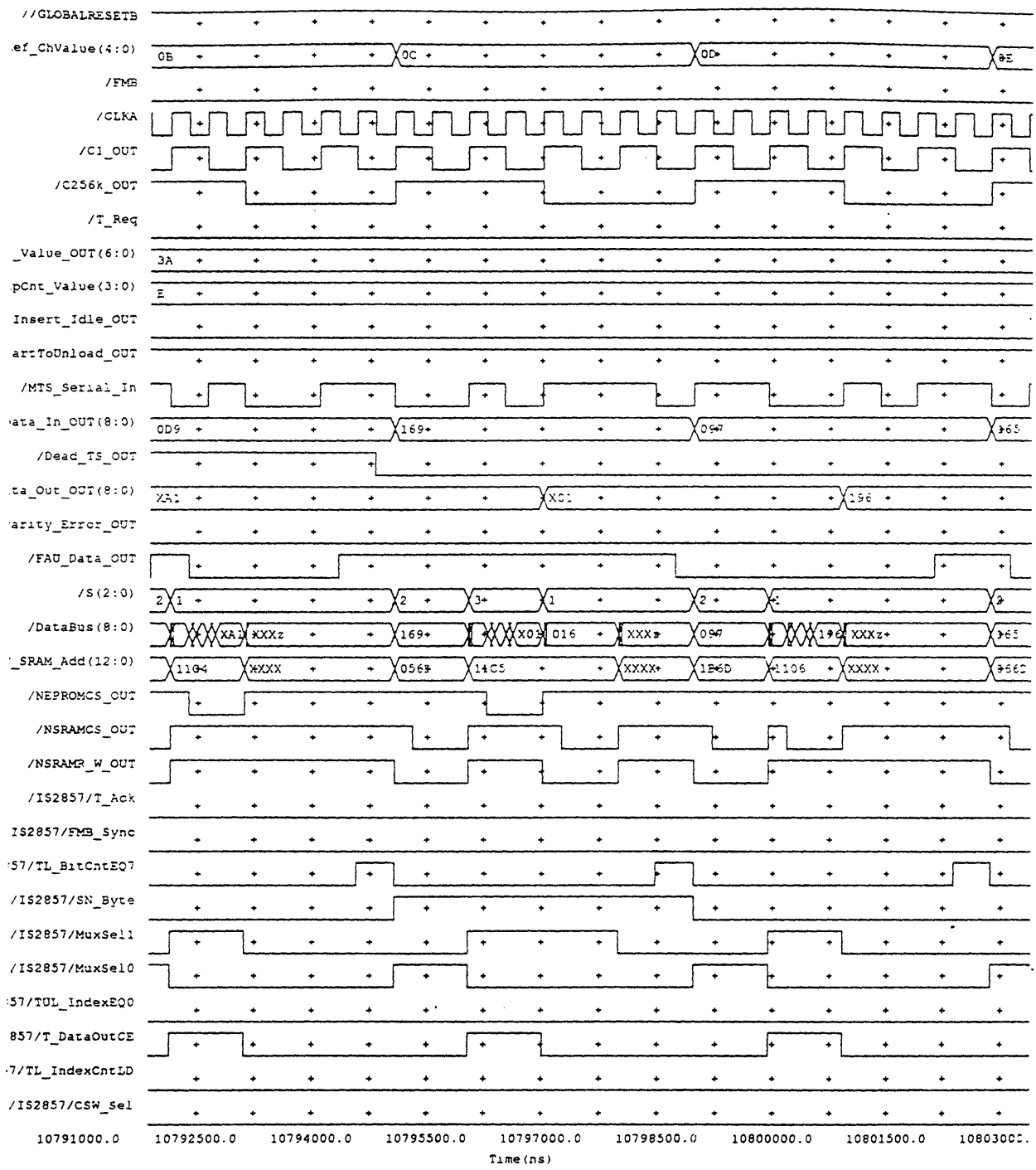

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
357/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
10761920.0
10775440.0
10788960.0
10802480.0
10816000.0
10829520.0
10843040.0
10856560.0
Time(ns)

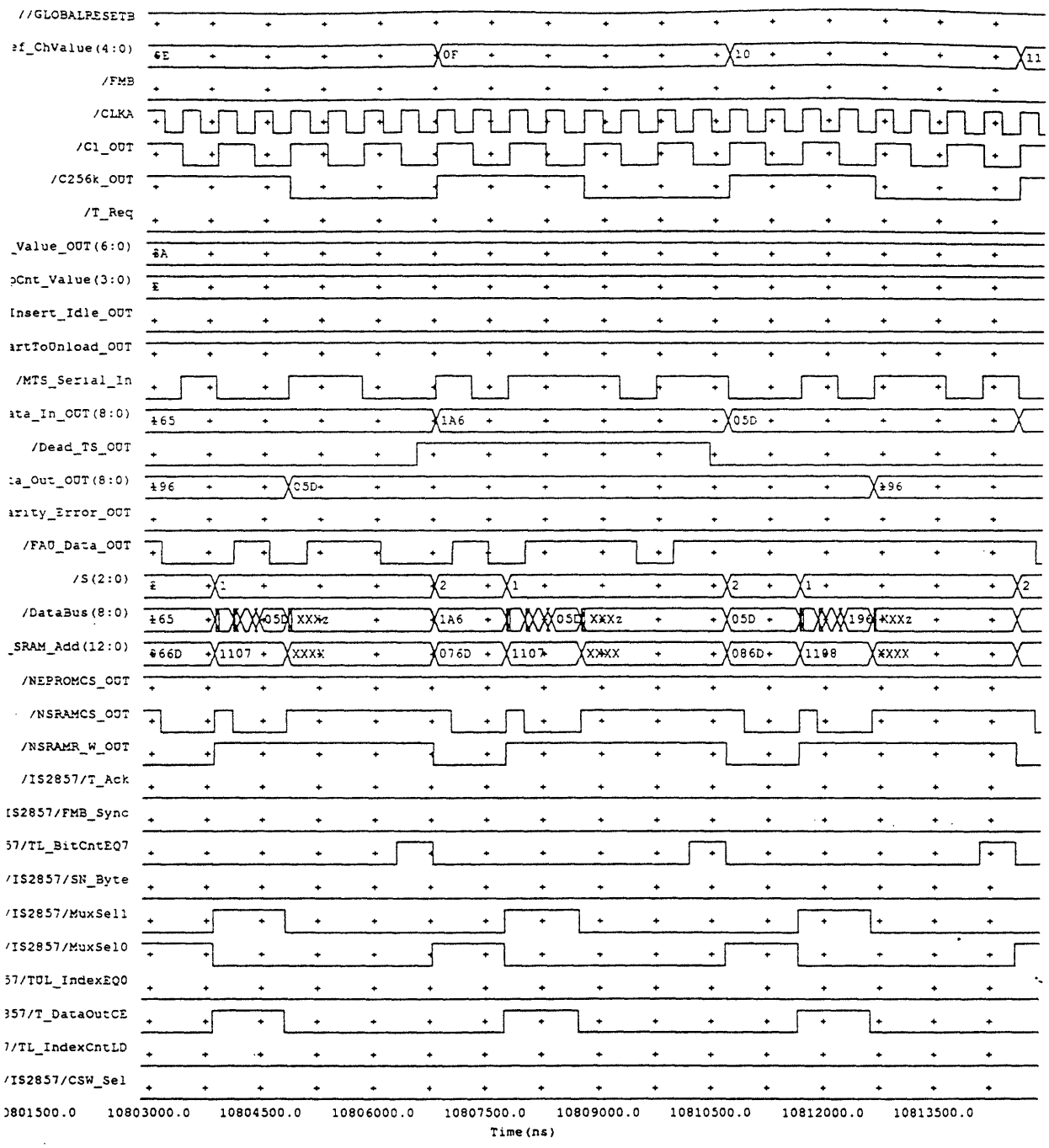

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
10964720.0
10978240.0
10991760.0
11005280.0
11018800.0
11032320.0
11045840.0
11059360.0
11072880.0
Time(ns)

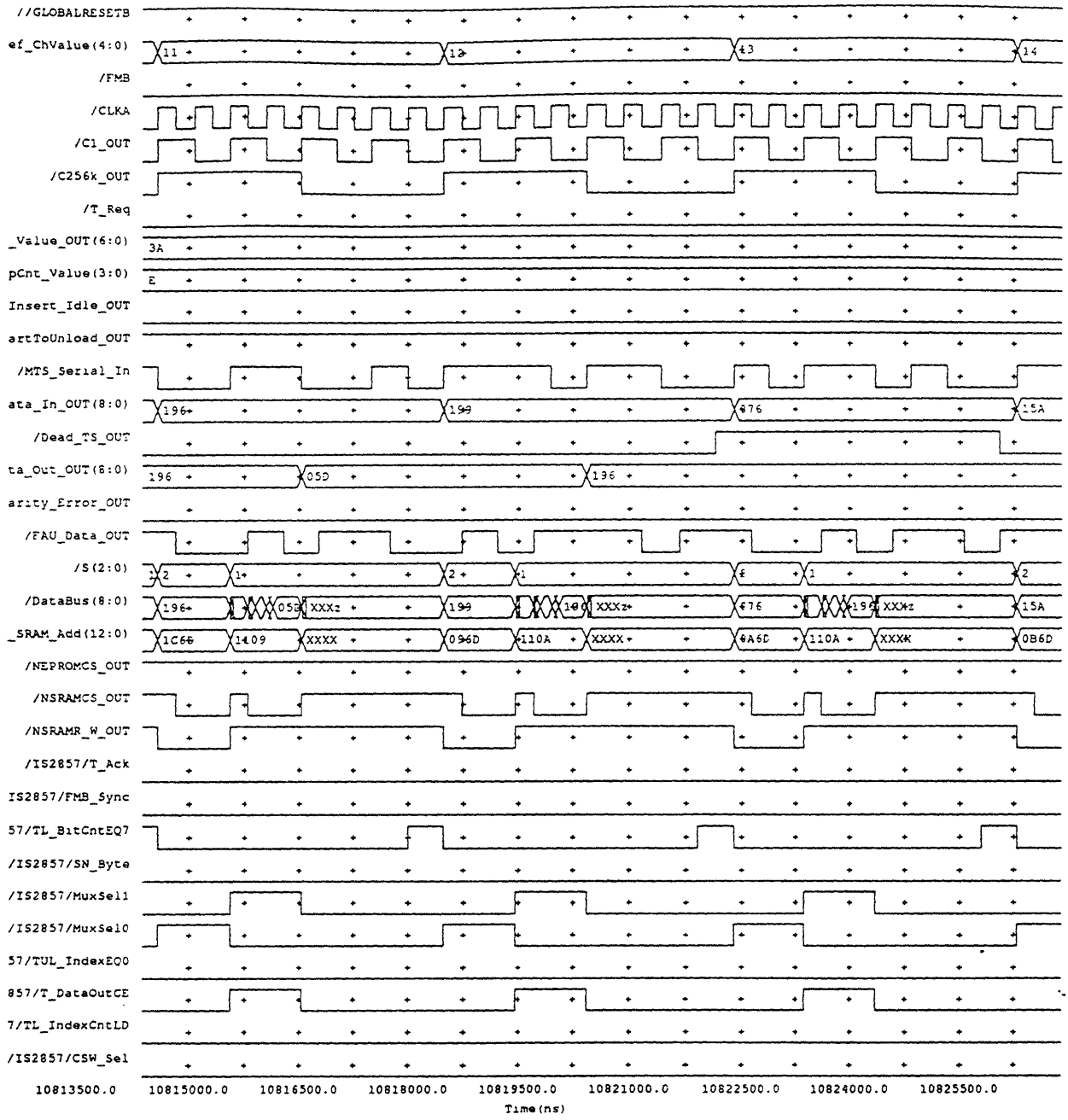

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
:57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
:57/TUL_IndexEQ0
:857/T_DataOutCE
:7/TL_IndexCntLD
/IS2857/CSW_Sel
10856560.0
10870080.0
10883600.0
10897120.0
10910640.0
10924160.0
10937680.0
10951200.0
10964720.0
Time(ns)

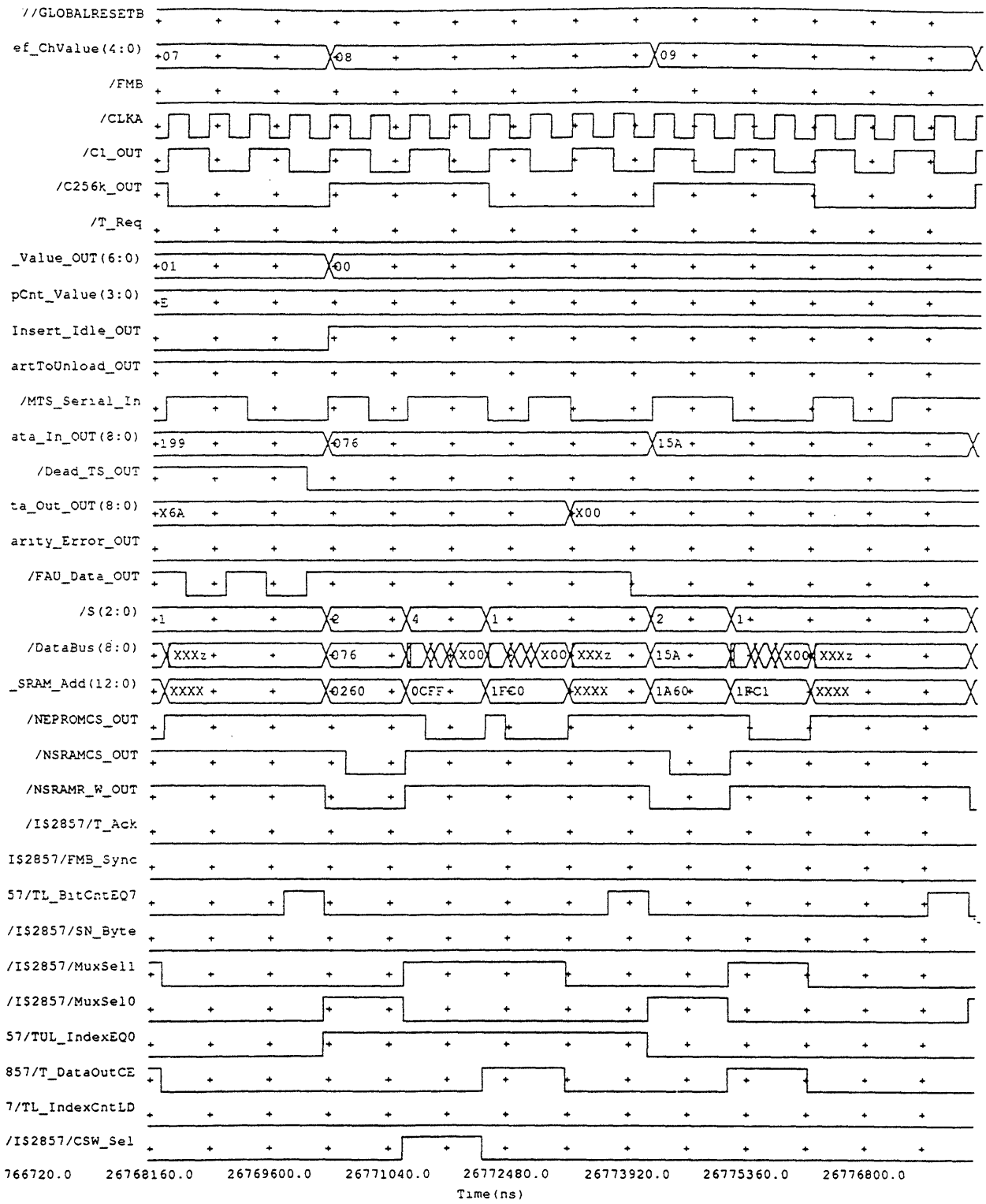

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAC_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSPAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
0767000.0
10768500.0
10770000.0
10771500.0
10773000.0
10774500.0
10776000.0
10777500.0
10779000.0
Time(ns)

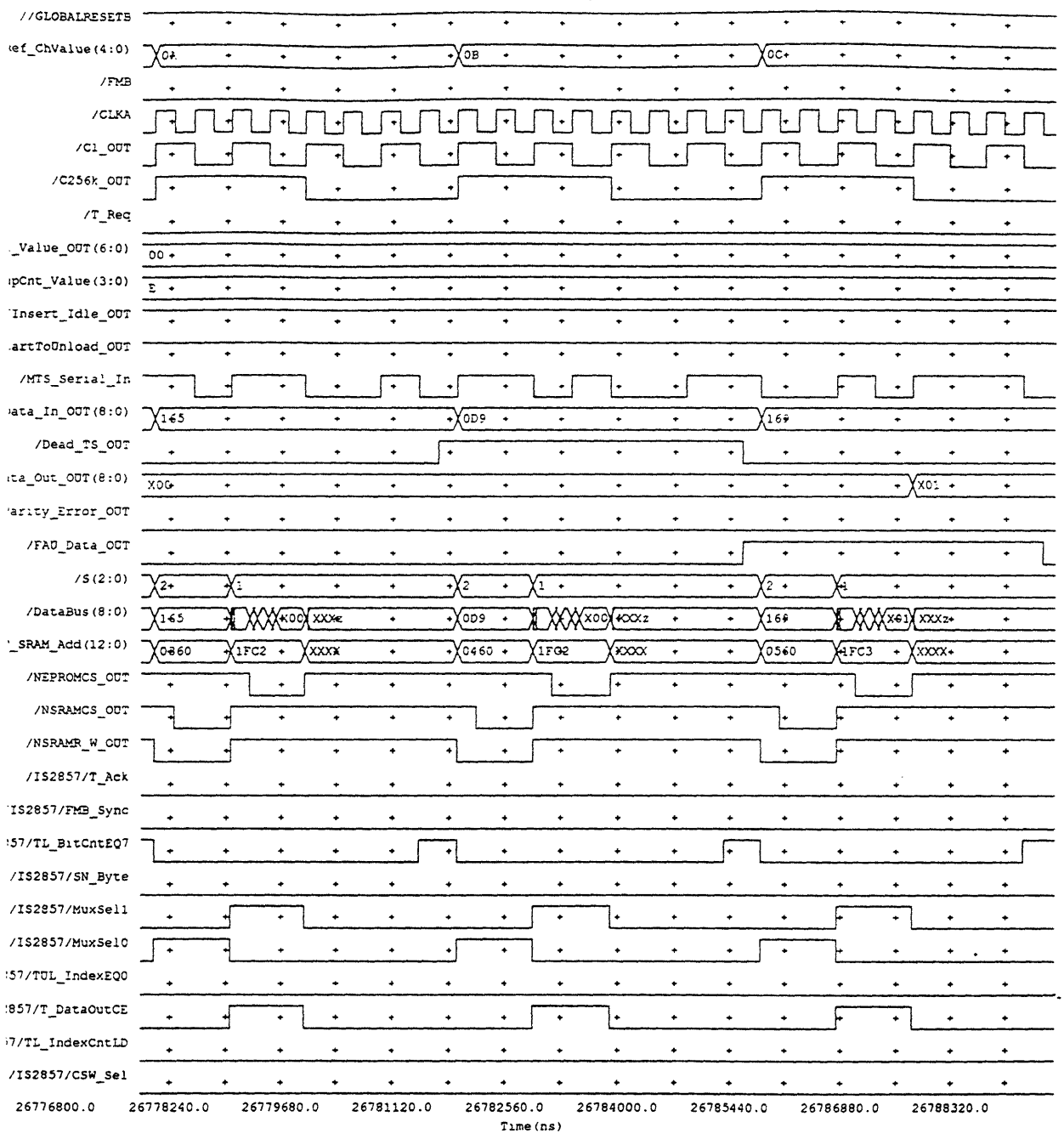

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
10779000.0
10780500.0
10782000.0
10783500.0
10785000.0
10786500.0
10788000.0
10789500.0
10791000.0
Time(ns)

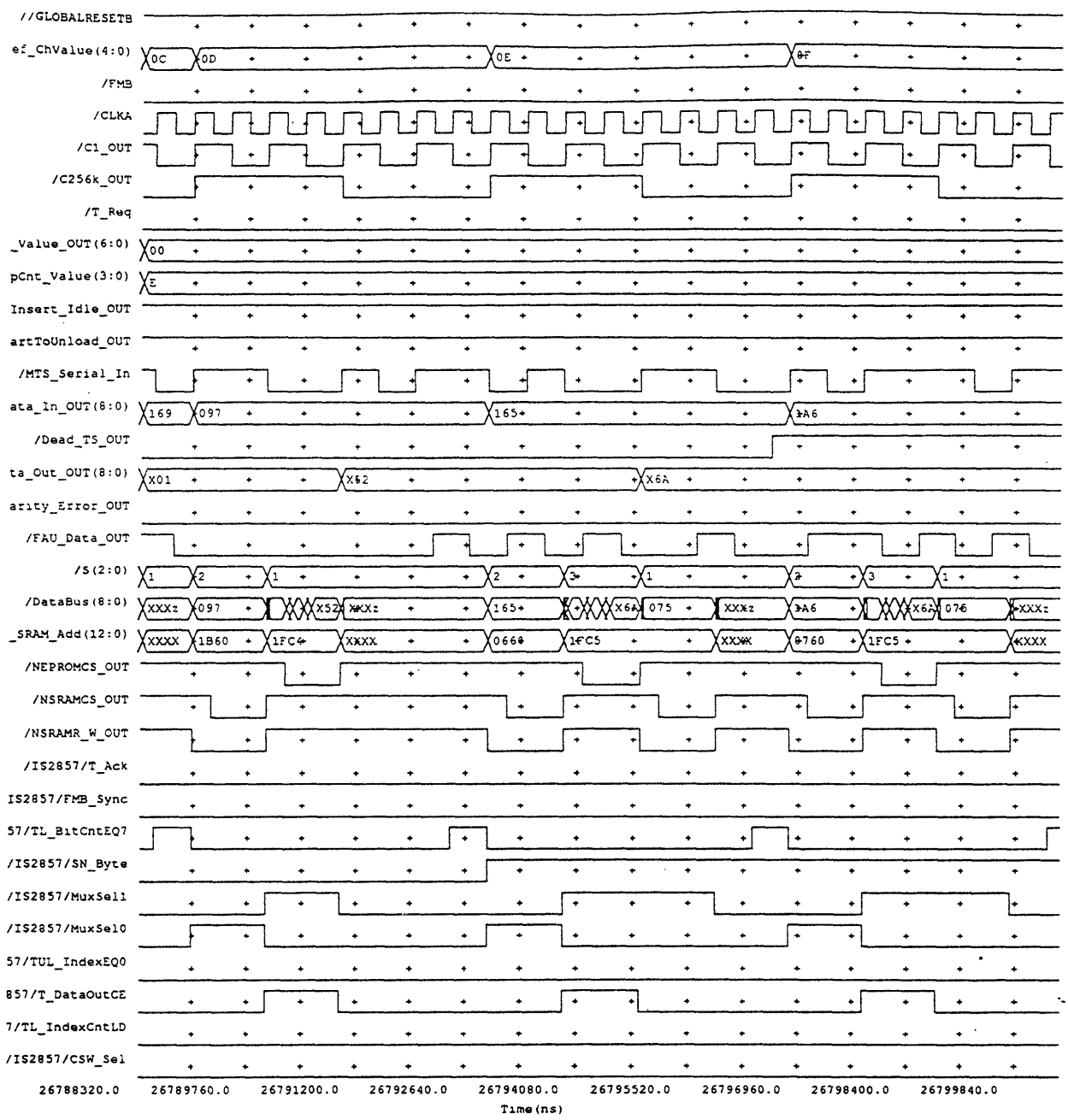

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
10791000.0
10792500.0
10794000.0
10795500.0
10797000.0
10798500.0
10800000.0
10801500.0
Time(ns)

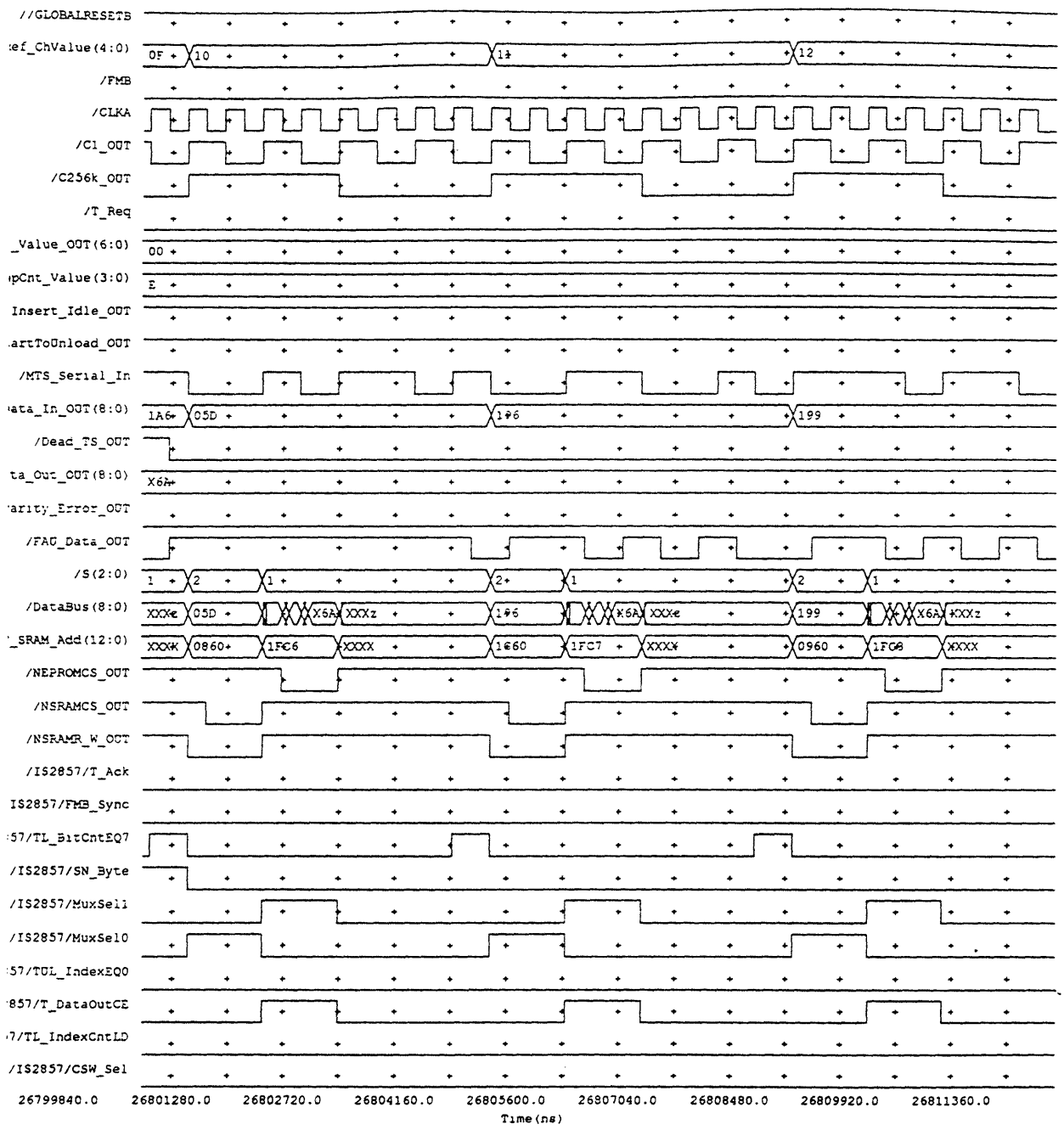

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FA0_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
357/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
0801500.0
10803000.0
10804500.0
10806000.0
10807500.0
10809000.0
10810500.0
10812000.0
10813500.0
Time(ns)

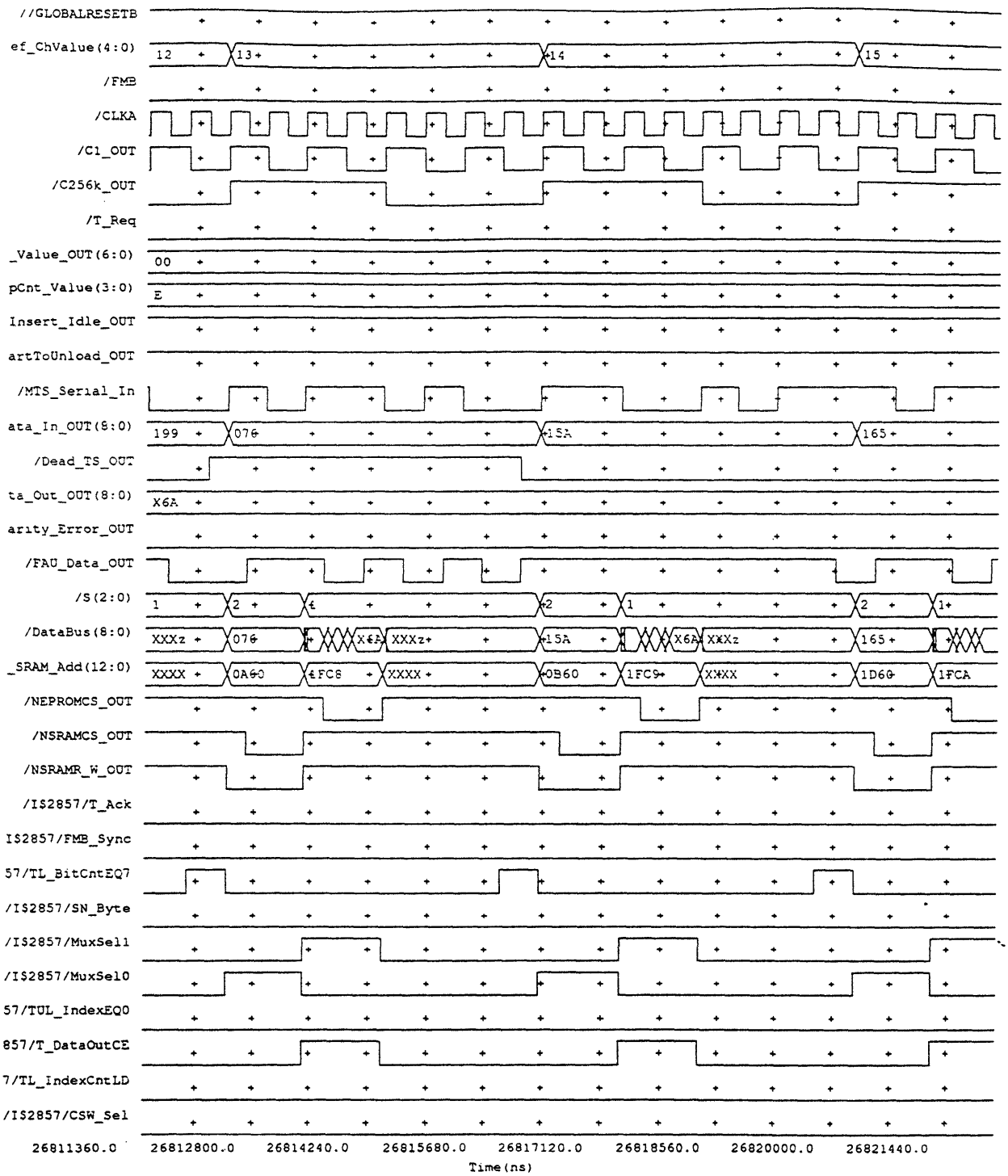

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
10813500.0
10815000.0
10816500.0
10818000.0
10819500.0
10821000.0
10822500.0
10824000.0
10825500.0
Time(ns)


//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
I$2857/FMB_Sync
57/TL_BitCntEQ7
/I$2857/SN_Byte
/I$2857/MuxSel1
/I$2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
+07
08
09
+01
00
+E
+199
076
15A
+X6A
X00
+1
2
4
1
2
1
XXXz
076
X00
X00
XXXz
15A
X00
XXXz
XXXX
0260
0CFF
1FE0
XXXX
1A60
1FC1
XXXX
766720.0
26768160.0
26769600.0
26771040.0
26772480.0
26773920.0
26775360.0
26776800.0
Time(ns)


//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
26776800.0
26778240.0
26779680.0
26781120.0
26782560.0
26784000.0
26785440.0
26786880.0
26788320.0
Time(ns)


//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
26788320.0
26789760.0
26791200.0
26792640.0
26794080.0
26795520.0
26796960.0
26798400.0
26799840.0
Time(ns)

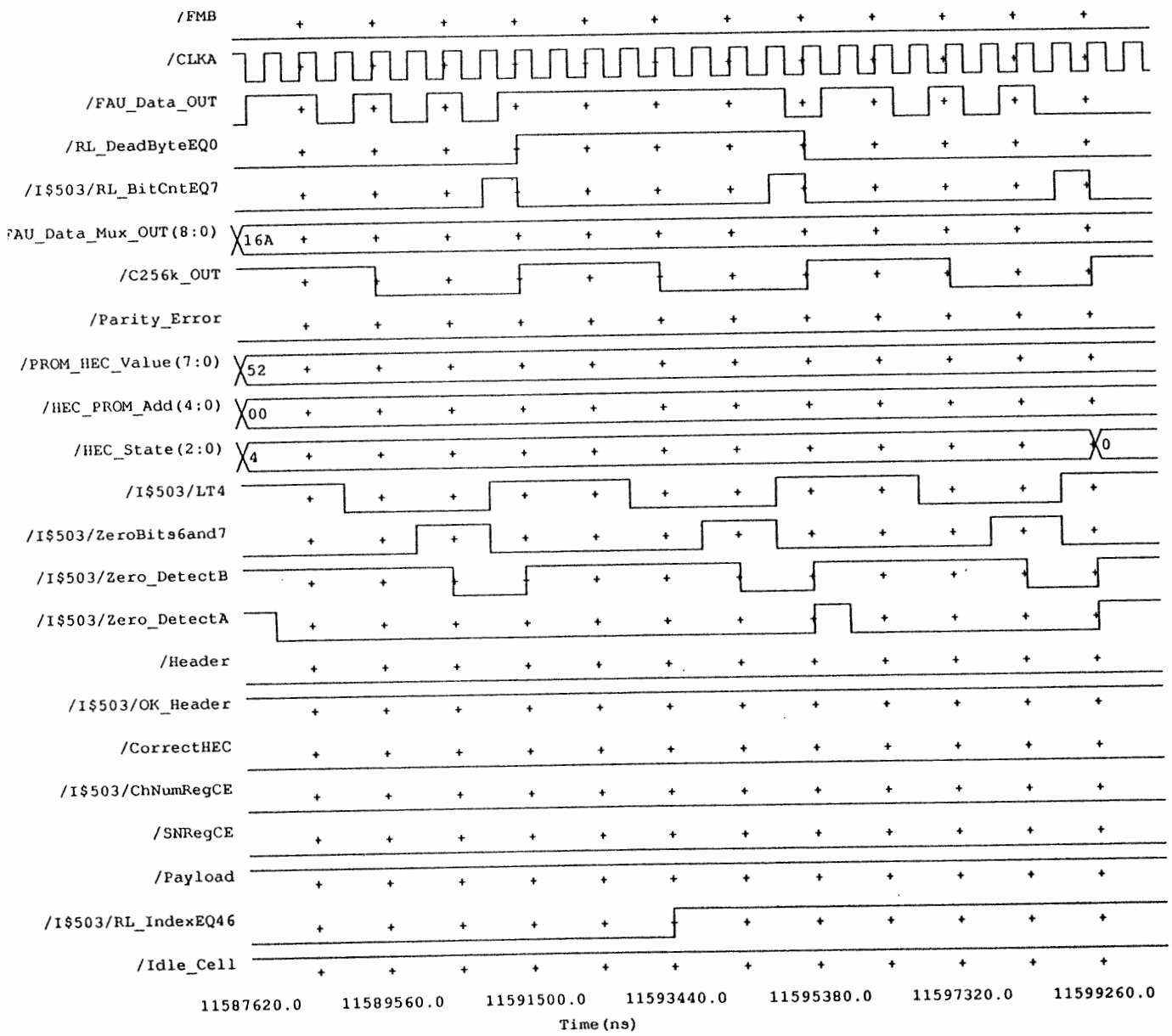

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/IS2857/T_Ack
IS2857/FMB_Sync
57/TL_BitCntEQ7
/IS2857/SN_Byte
/IS2857/MuxSel1
/IS2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/IS2857/CSW_Sel
26799840.0
26801280.0
26802720.0
26804160.0
26805600.0
26807040.0
26808480.0
26809920.0
26811360.0
Time(ns)

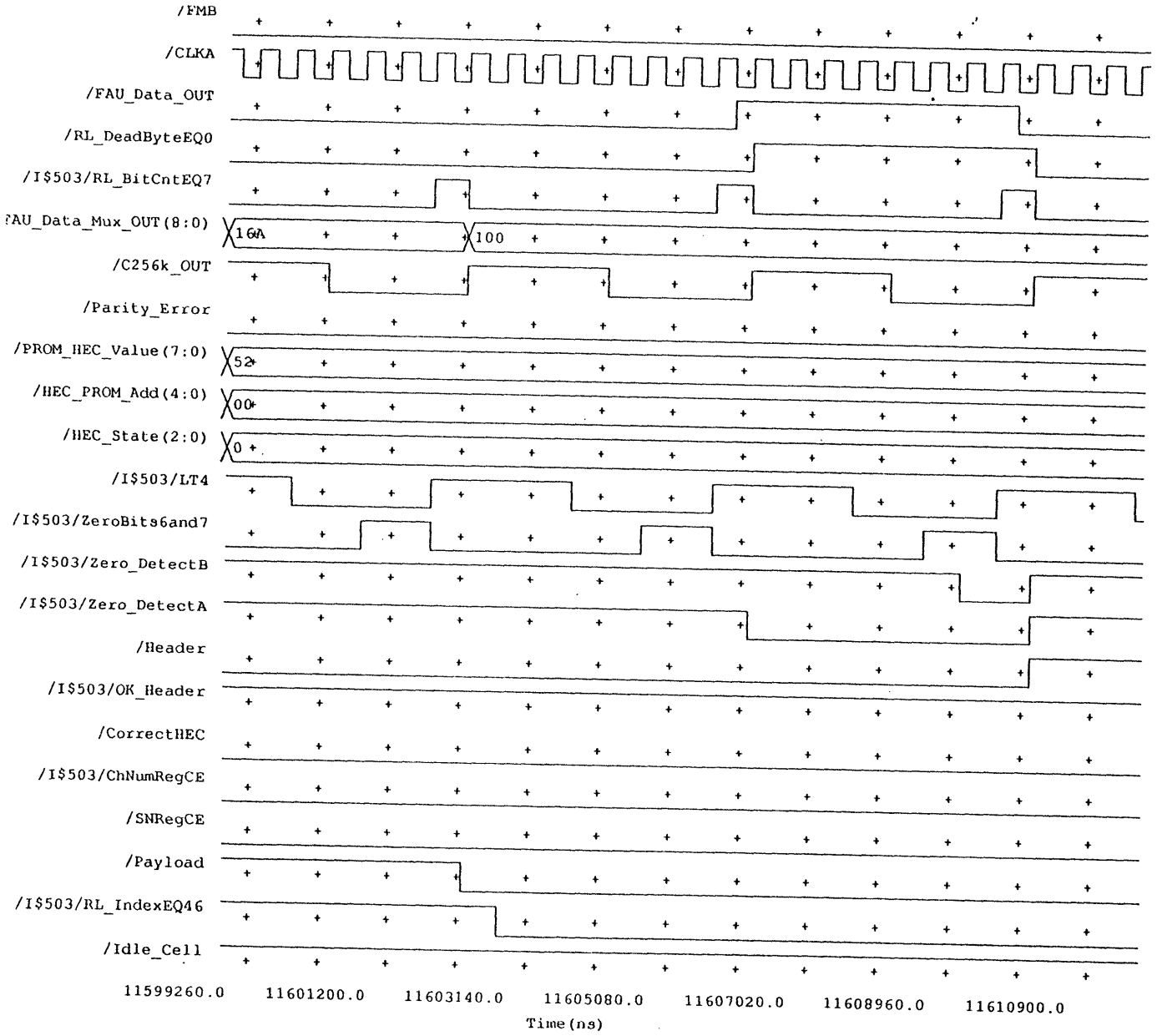

//GLOBALRESETB
ef_ChValue(4:0)
/FMB
/CLKA
/C1_OUT
/C256k_OUT
/T_Req
_Value_OUT(6:0)
pCnt_Value(3:0)
Insert_Idle_OUT
artToUnload_OUT
/MTS_Serial_In
ata_In_OUT(8:0)
/Dead_TS_OUT
ta_Out_OUT(8:0)
arity_Error_OUT
/FAU_Data_OUT
/S(2:0)
/DataBus(8:0)
_SRAM_Add(12:0)
/NEPROMCS_OUT
/NSRAMCS_OUT
/NSRAMR_W_OUT
/I$2857/T_Ack
I$2857/FMB_Sync
57/TL_BitCntEQ7
/I$2857/SN_Byte
/I$2857/MuxSel1
/I$2857/MuxSel0
57/TUL_IndexEQ0
857/T_DataOutCE
7/TL_IndexCntLD
/I$2857/CSW_Sel
26811360.0
26812800.0
26814240.0
26815680.0
26817120.0
26818560.0
26820000.0
26821440.0
Time(ns)

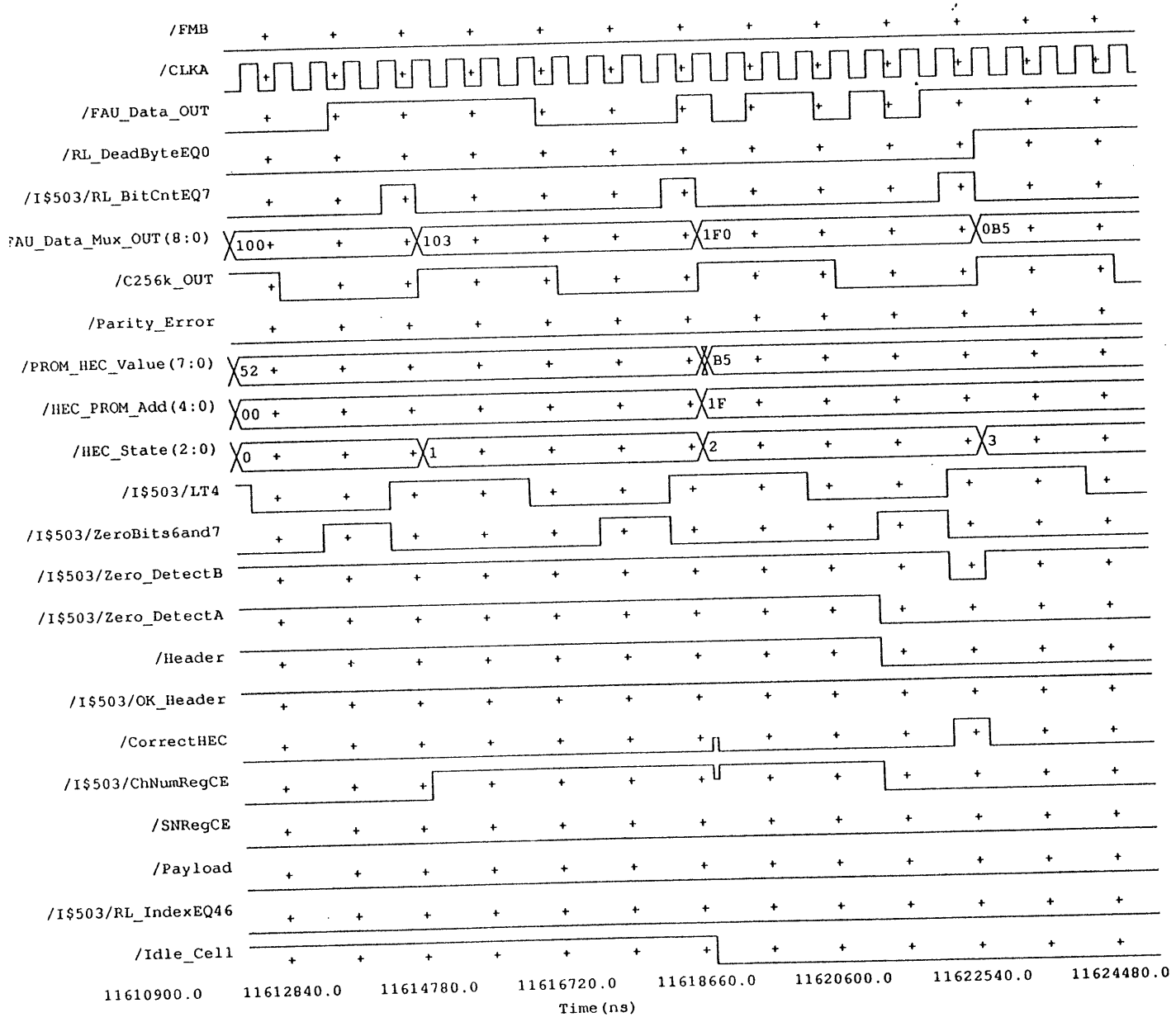
Appendix H Receiver Cell Delineation Simulation

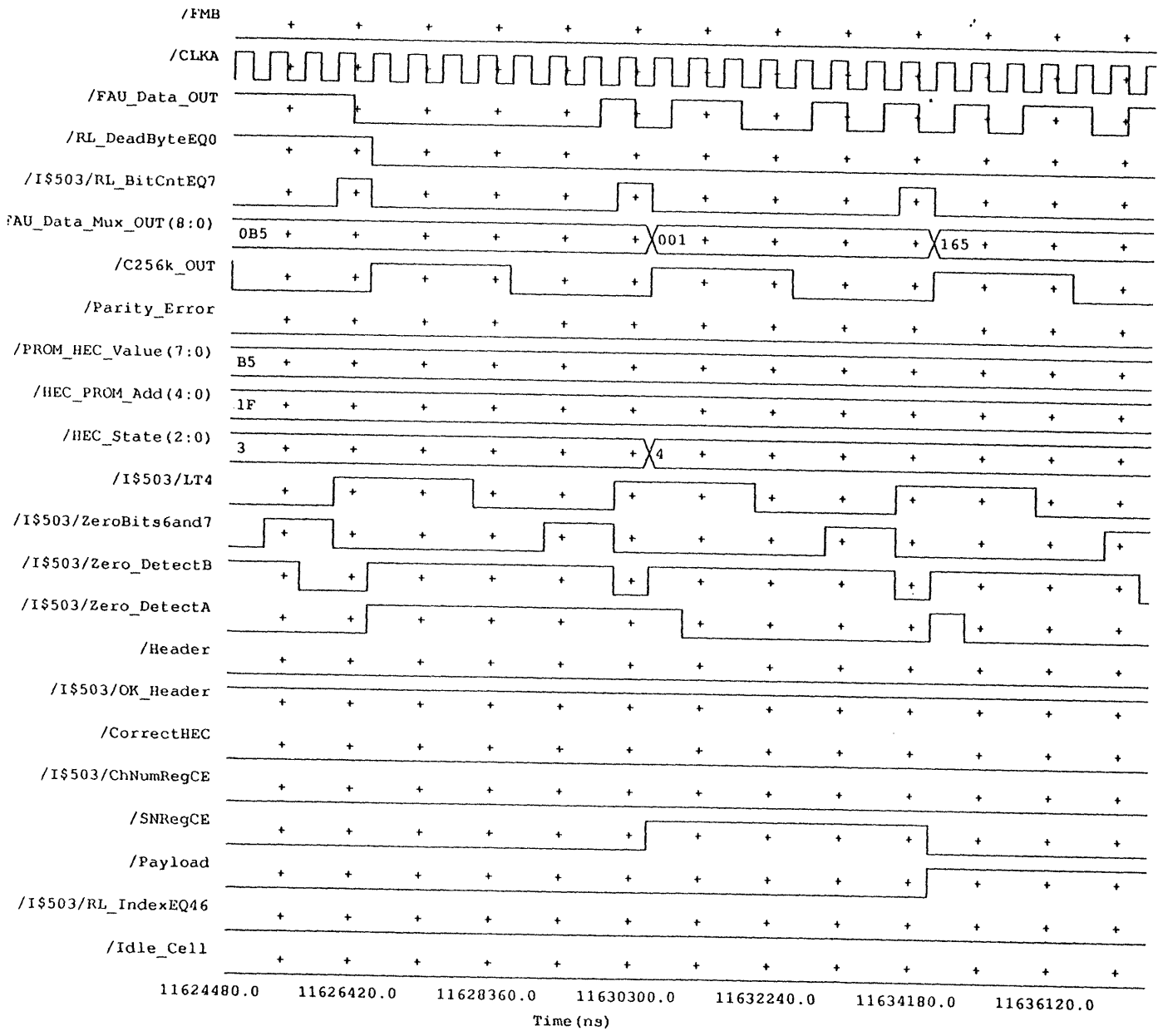
The timing simulation shown in this Appendix is for the Cell Delineation Function of the Receiver. The input pattern for the Cell Delineation is the output pattern generated from the Transmitter simulation. Only time slots 27 and 31 are active.

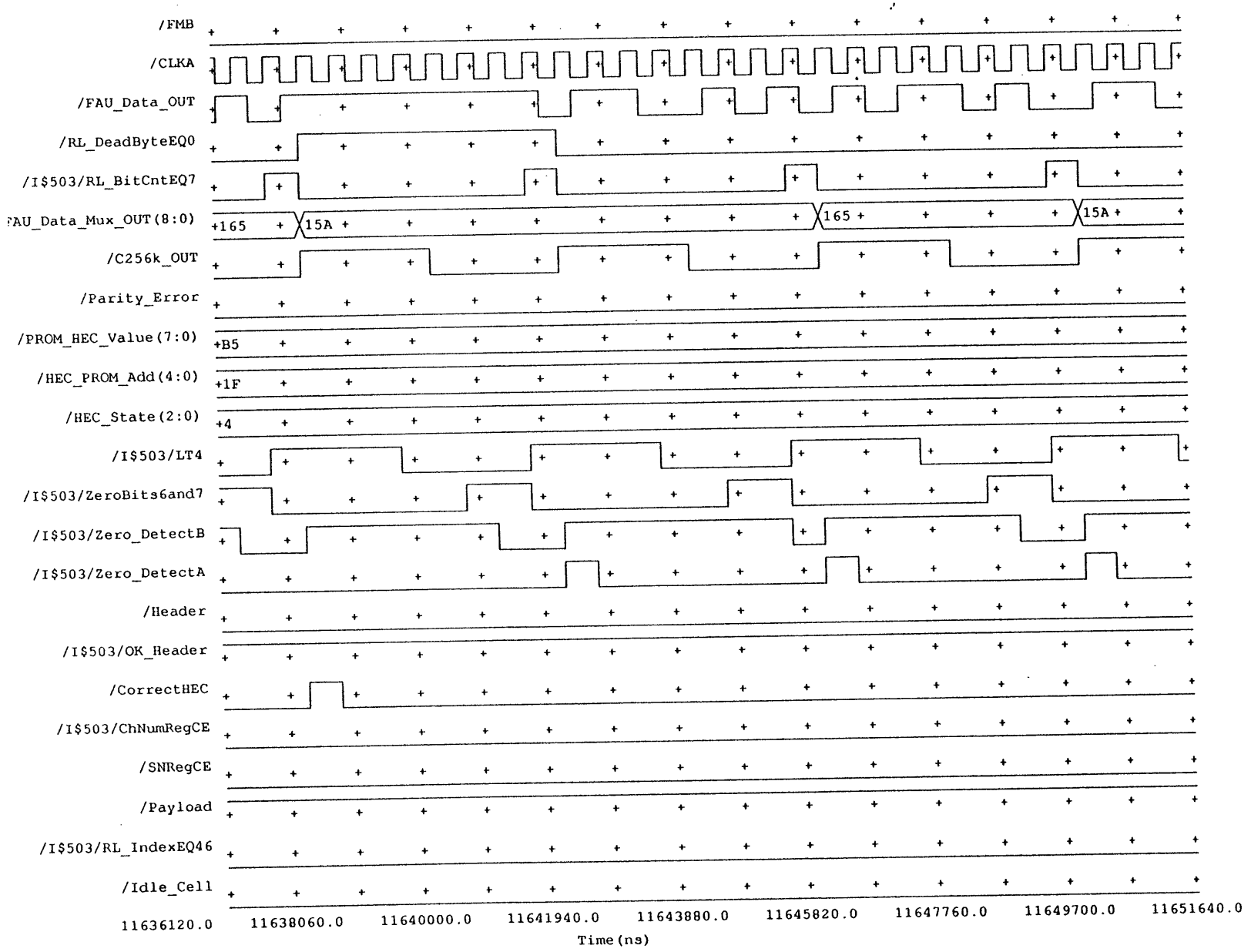
Simulation Time Period	Pages	Simulation Description
11560000 - 11660000	144-148	Portion of the beginning of a normal cell delineation of an ATM cell for channel 31
11850000 - 11920000	149-152	End of the cell delineation for the above ATM cell for channel 31
11560000 - 11660000	153-157	Portion of the beginning of a cell delineation for a channel 31 ATM cell with leading zeros

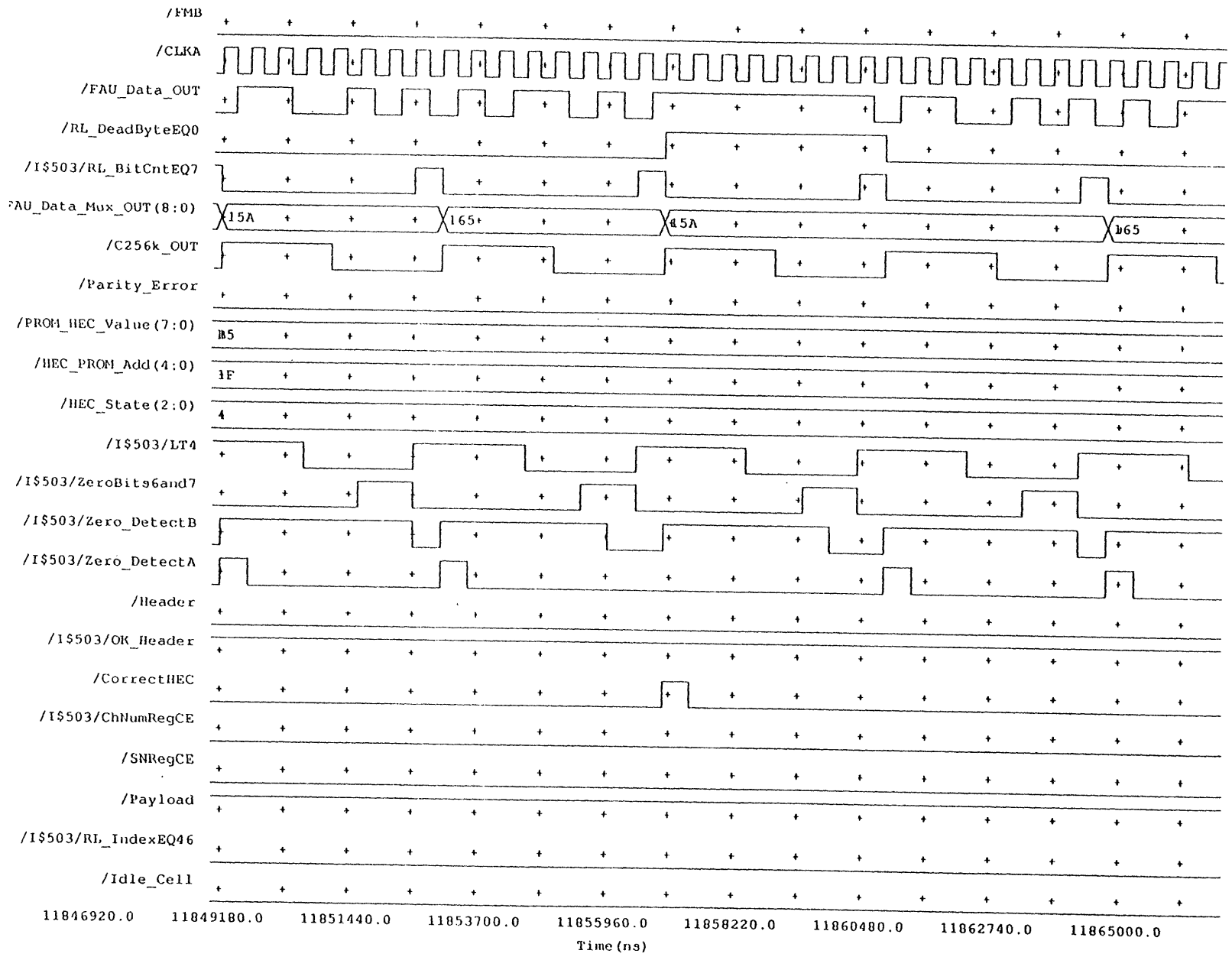

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
16A
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
52
/HEC_PROM_Add(4:0)
00
/HEC_State(2:0)
4
0
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11587620.0
11589560.0
11591500.0
11593440.0
11595380.0
11597320.0
11599260.0
Time (ns)

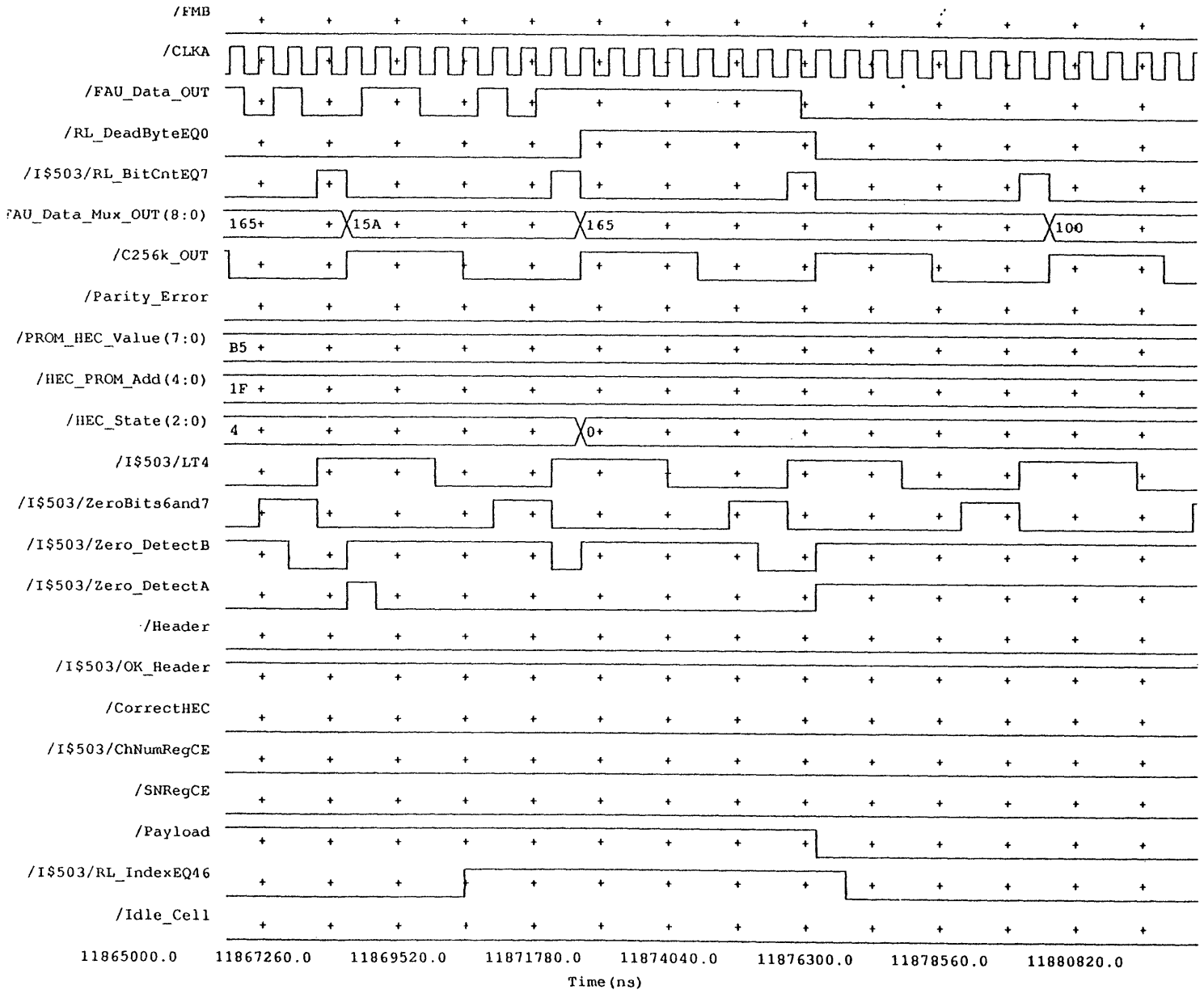

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
16A
100
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
52
/HEC_PROM_Add(4:0)
00
/HEC_State(2:0)
0
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11599260.0
11601200.0
11603140.0
11605080.0
11607020.0
11608960.0
11610900.0
Time(ns)

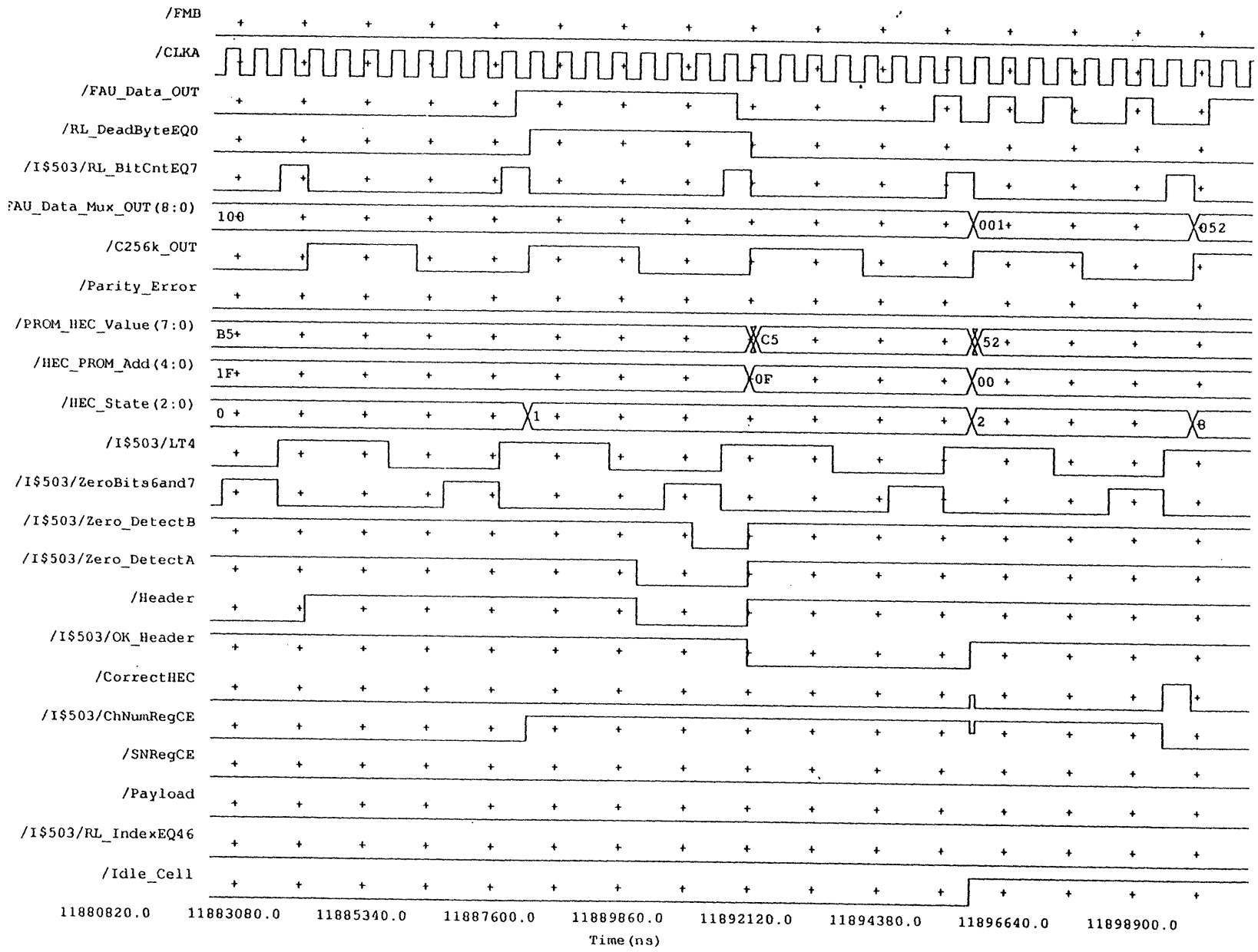

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
100+
103
1F0
0B5
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
52
B5
/HEC_PROM_Add(4:0)
00
1F
/HEC_State(2:0)
0
1
2
3
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11610900.0
11612840.0
11614780.0
11616720.0
11618660.0
11620600.0
11622540.0
11624480.0
Time(ns)

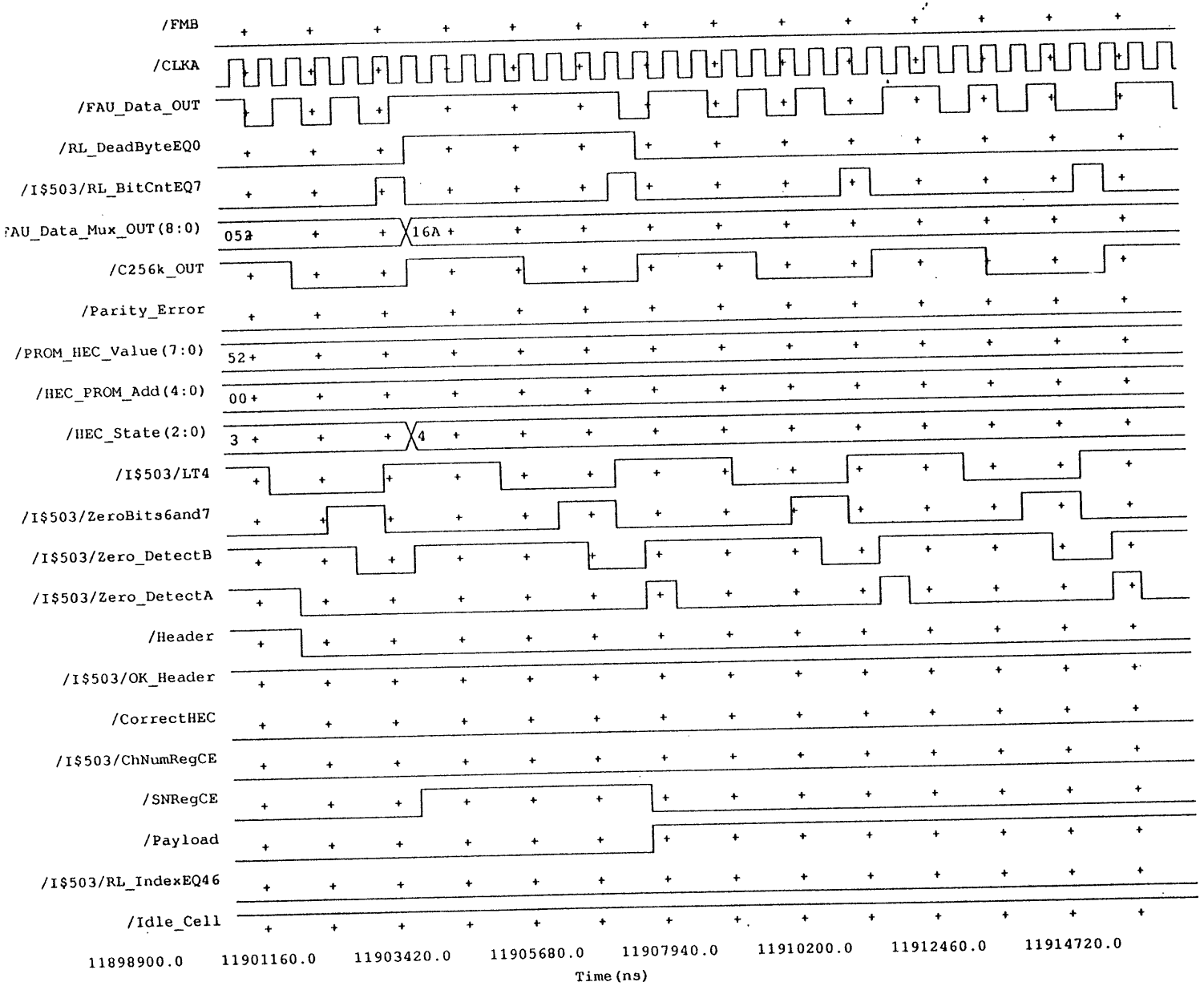

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
0B5
001
165
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
B5
/HEC_PROM_Add(4:0)
1F
/HEC_State(2:0)
3
4
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11624480.0
11626420.0
11628360.0
11630300.0
11632240.0
11634180.0
11636120.0
Time(ns)

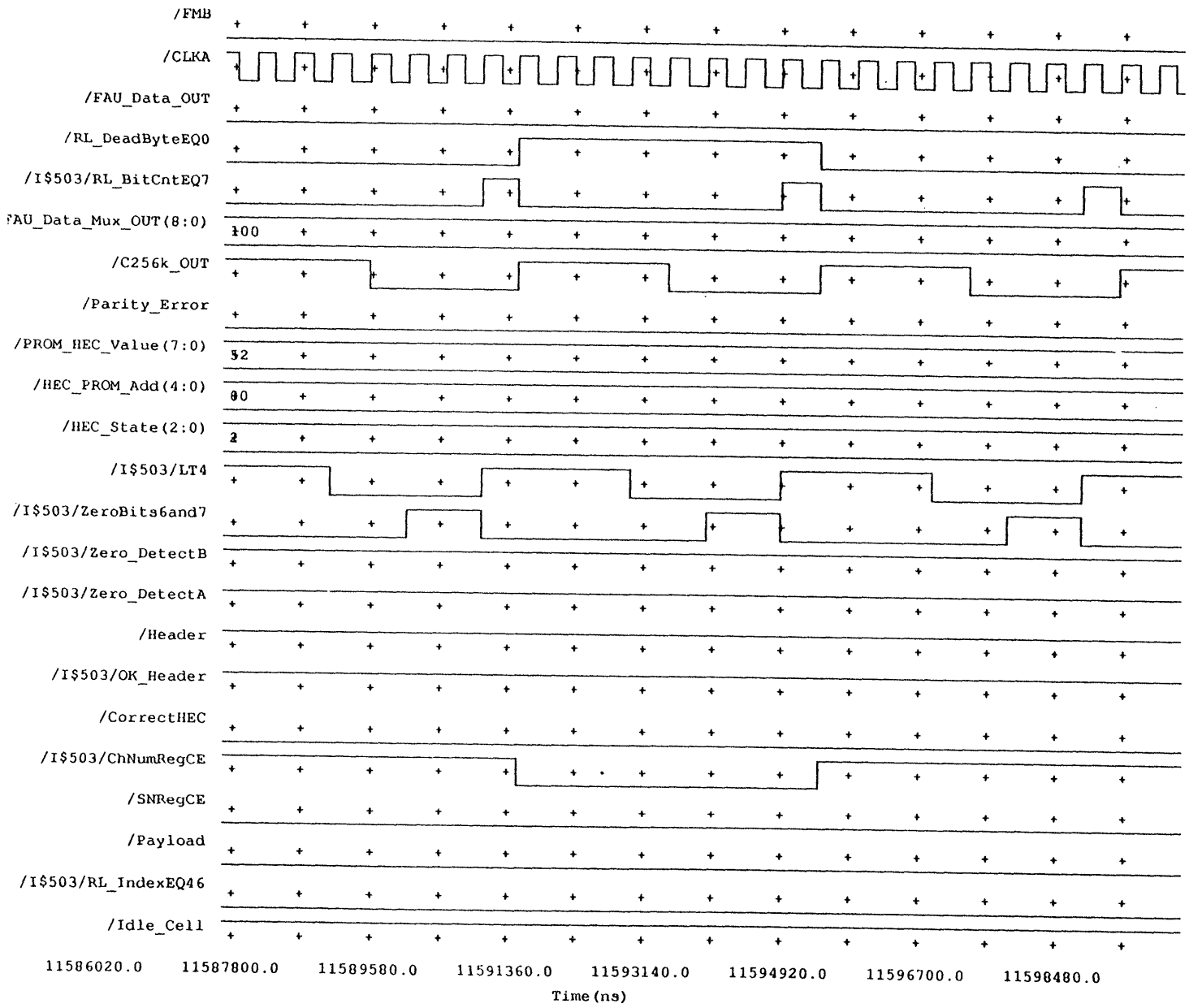

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
+165
15A
165
15A
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
+B5
/HEC_PROM_Add(4:0)
+1F
/HEC_State(2:0)
+4
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11636120.0
11638060.0
11640000.0
11641940.0
11643880.0
11645820.0
11647760.0
11649700.0
11651640.0
Time(ns)

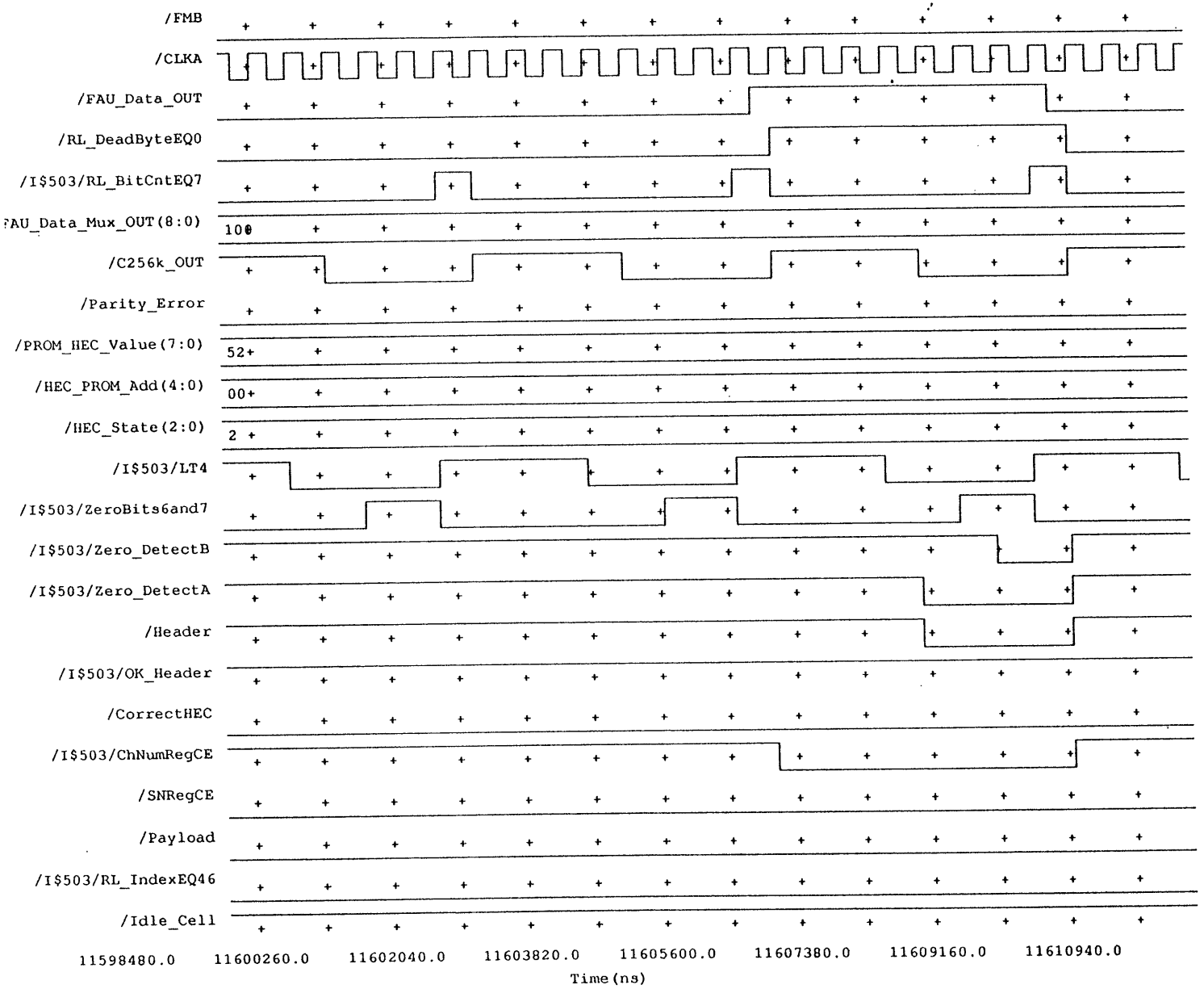

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
15A
165
65
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
/HEC_PROM_Add(4:0)
1F
/HEC_State(2:0)
4
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11846920.0
11849180.0
11851440.0
11853700.0
11855960.0
11858220.0
11860480.0
11862740.0
11865000.0
Time(ns)

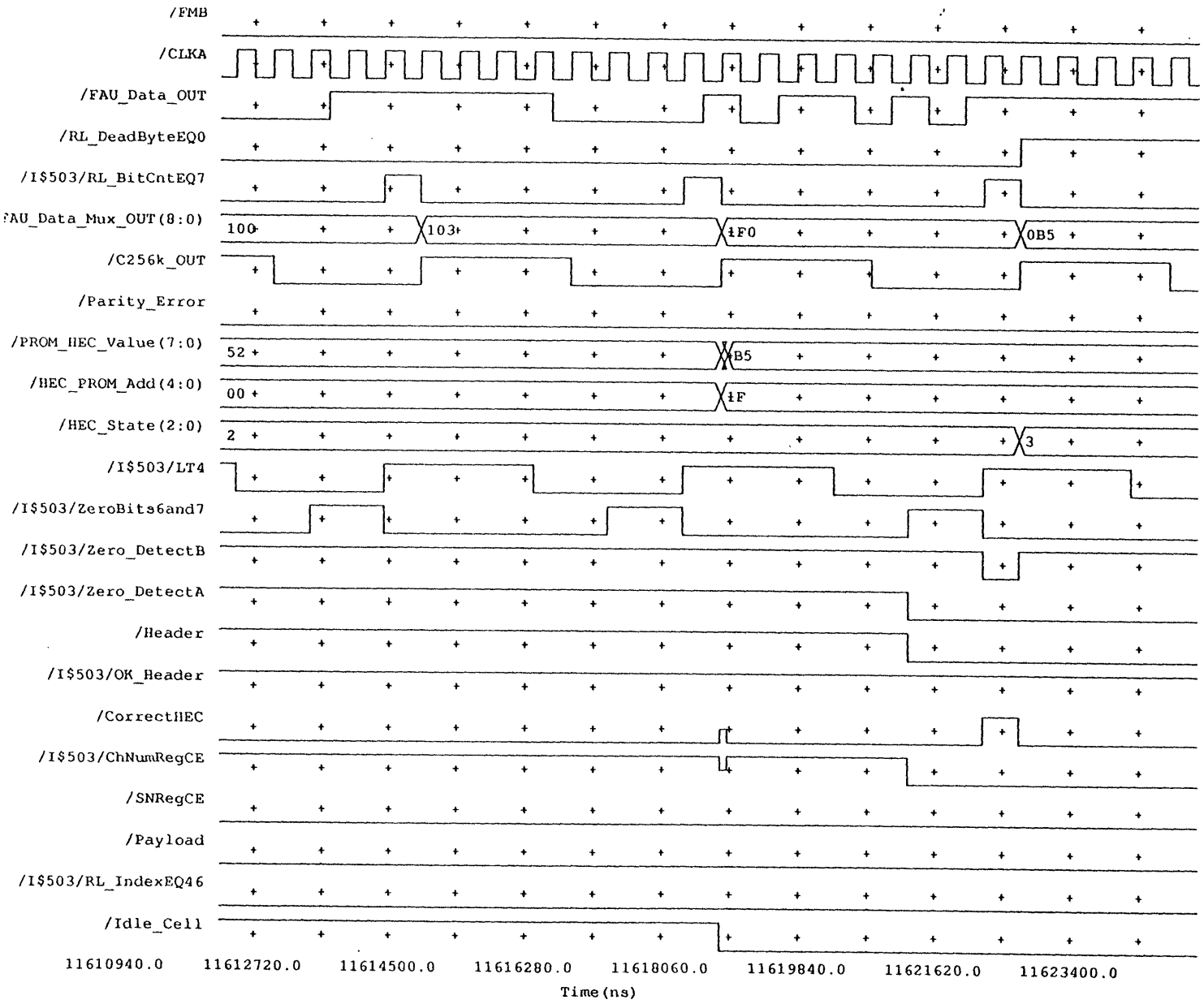

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
165
15A
165
100
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
B5
/HEC_PROM_Add(4:0)
1F
/HEC_State(2:0)
4
0
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11865000.0
11867260.0
11869520.0
11871780.0
11874040.0
11876300.0
11878560.0
11880820.0
Time(ns)

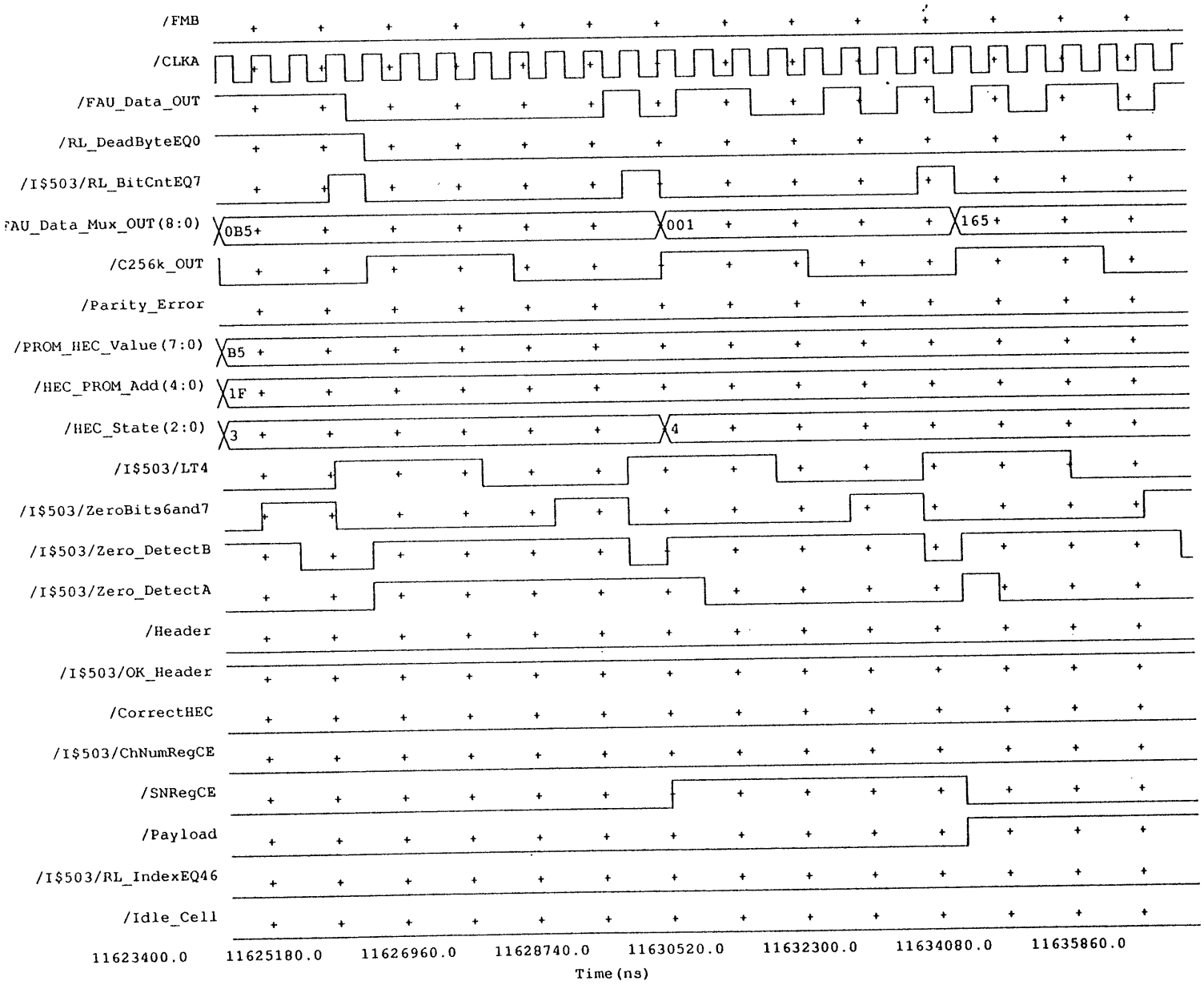

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
100
001
052
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
B5
C5
52
/HEC_PROM_Add(4:0)
1F
0F
00
/HEC_State(2:0)
0
1
2
0
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11880820.0
11883080.0
11885340.0
11887600.0
11889860.0
11892120.0
11894380.0
11896640.0
11898900.0
Time(ns)

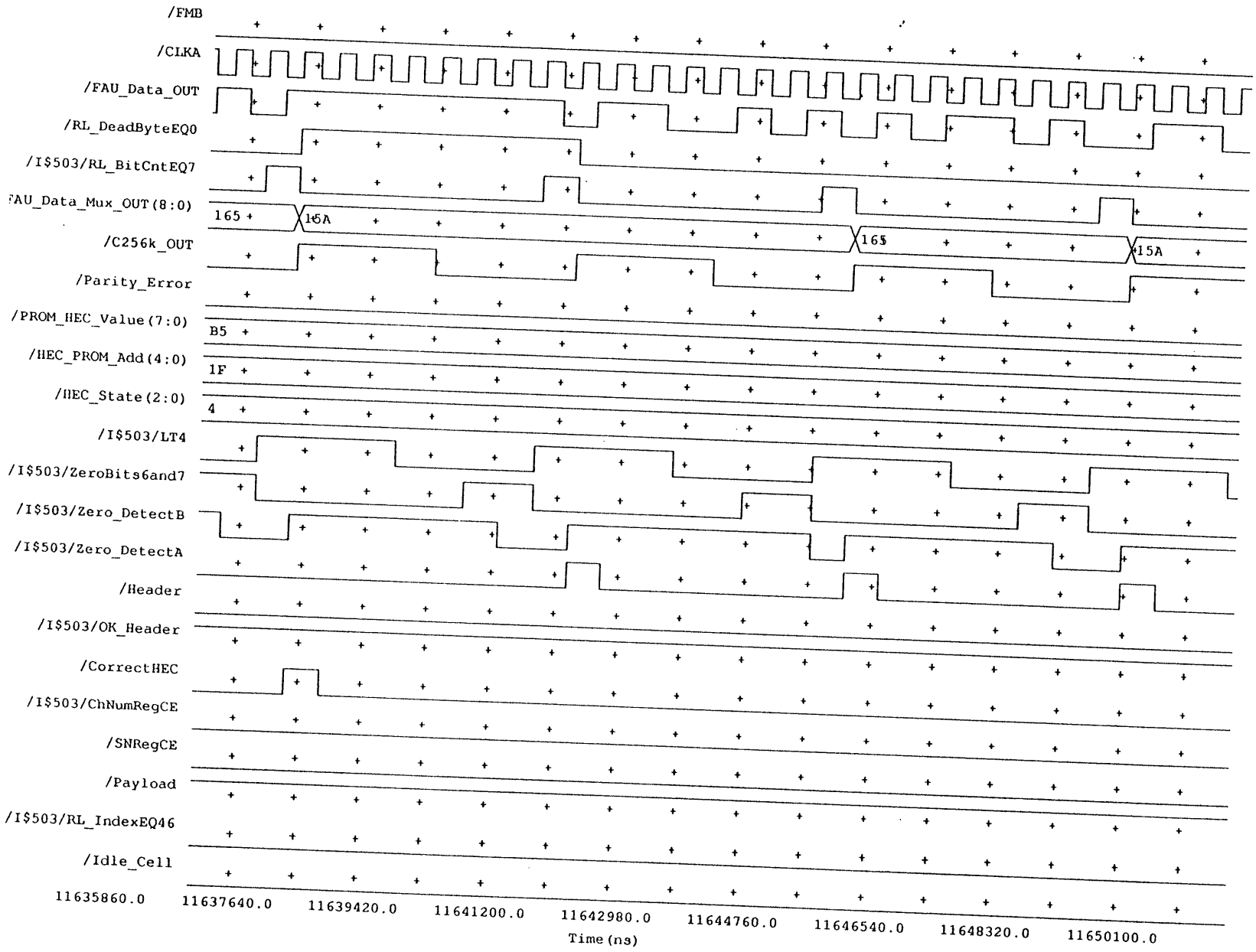

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
052
16A
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
52
/HEC_PROM_Add(4:0)
00
/HEC_State(2:0)
3
4
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11898900.0
11901160.0
11903420.0
11905680.0
11907940.0
11910200.0
11912460.0
11914720.0
Time (ns)


/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
100
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
52
/HEC_PROM_Add(4:0)
00
/HEC_State(2:0)
2
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11586020.0
11587800.0
11589580.0
11591360.0
11593140.0
11594920.0
11596700.0
11598480.0
Time (ns)


/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
100
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
52+
/HEC_PROM_Add(4:0)
00+
/HEC_State(2:0)
2 +
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11598480.0
11600260.0
11602040.0
11603820.0
11605600.0
11607380.0
11609160.0
11610940.0
Time (ns)


/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
100
103
1F0
0B5
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
52
B5
/HEC_PROM_Add(4:0)
00
1F
/HEC_State(2:0)
2
3
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11610940.0
11612720.0
11614500.0
11616280.0
11618060.0
11619840.0
11621620.0
11623400.0
Time(ns)

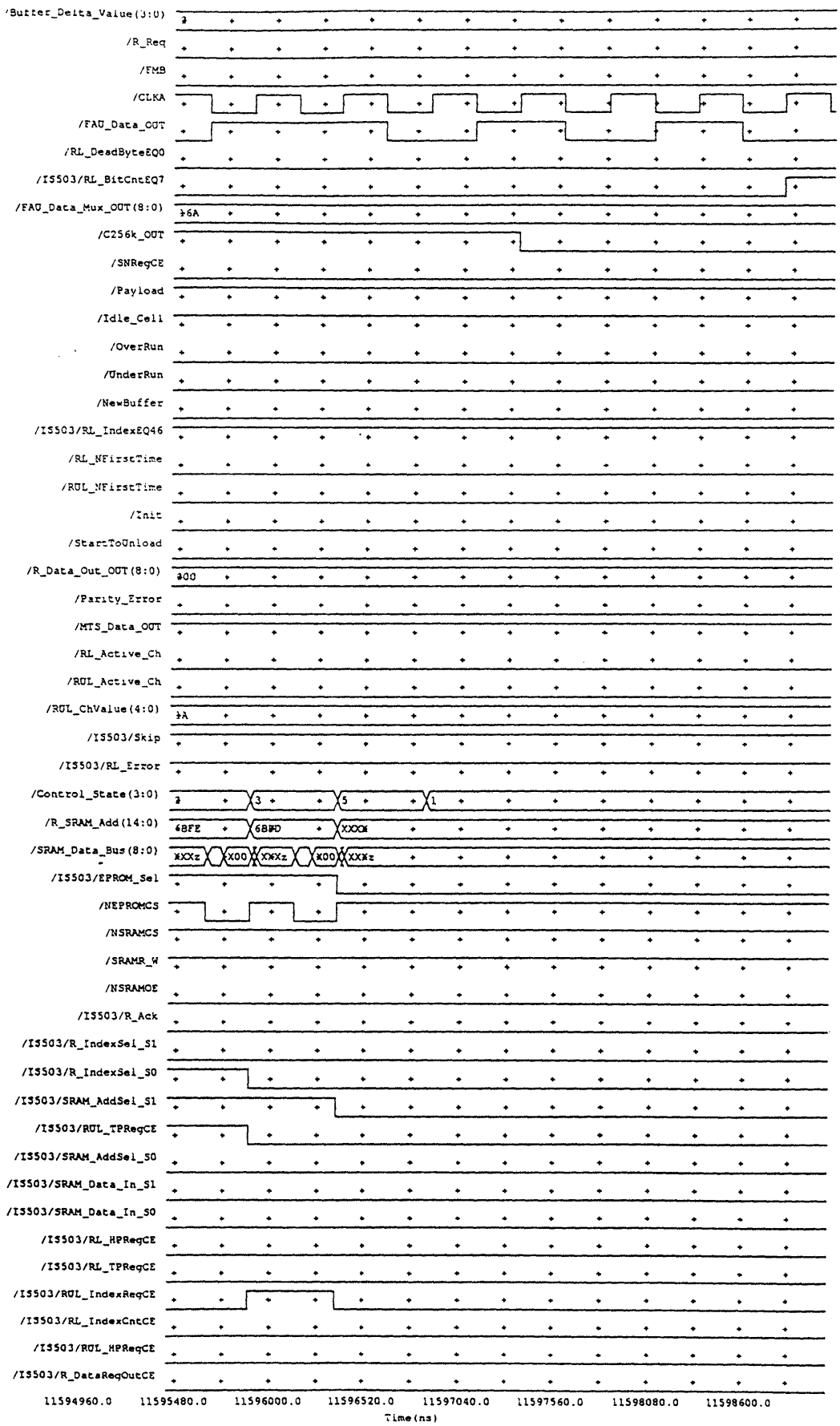

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
0B5
001
165
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
B5
/HEC_PROM_Add(4:0)
1F
/HEC_State(2:0)
3
4
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11623400.0
11625180.0
11626960.0
11628740.0
11630520.0
11632300.0
11634080.0
11635860.0
Time(ns)

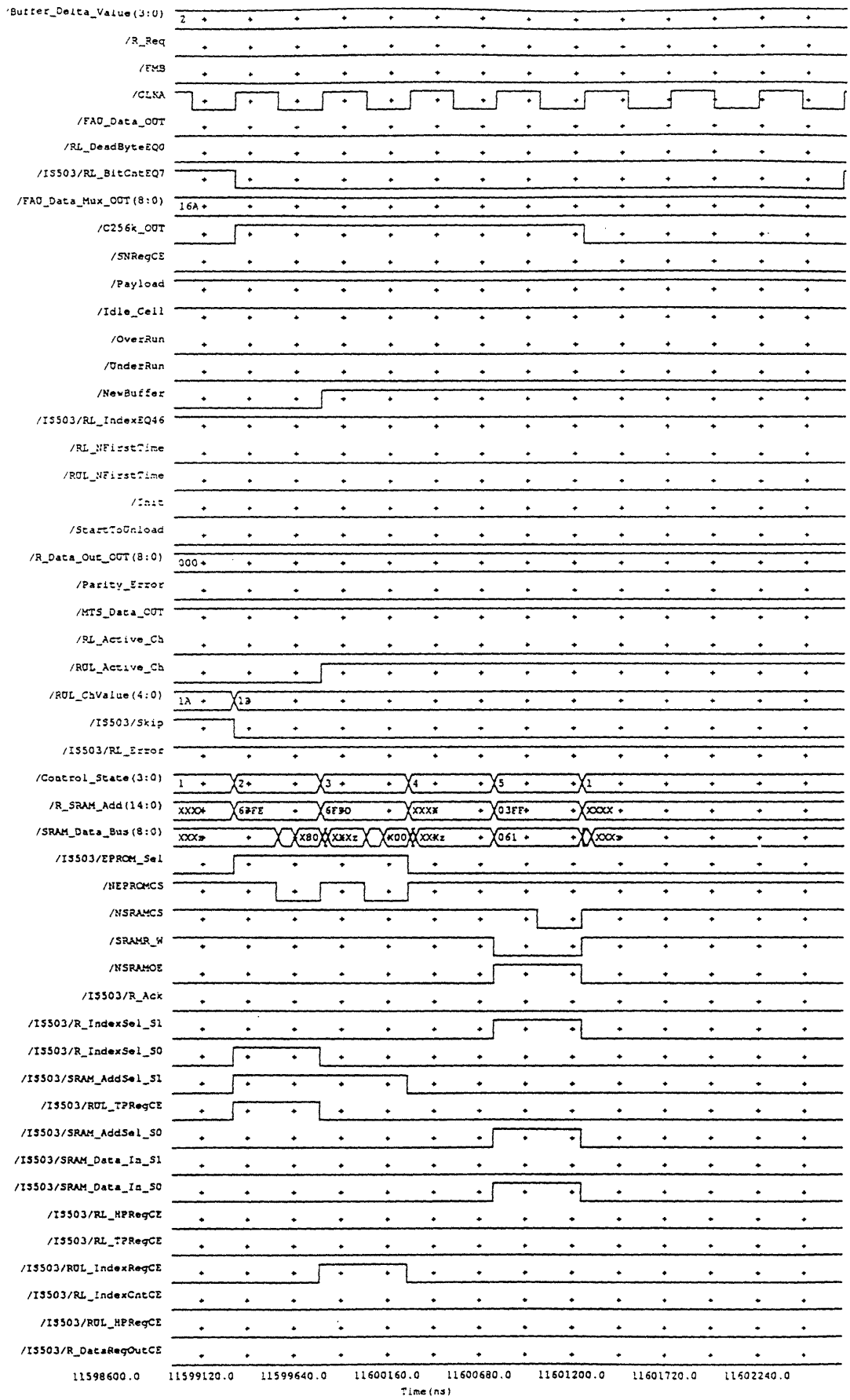

/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I$503/RL_BitCntEQ7
FAU_Data_Mux_OUT(8:0)
165
15A
16$
15A
/C256k_OUT
/Parity_Error
/PROM_HEC_Value(7:0)
B5
/HEC_PROM_Add(4:0)
1F
/HEC_State(2:0)
4
/I$503/LT4
/I$503/ZeroBits6and7
/I$503/Zero_DetectB
/I$503/Zero_DetectA
/Header
/I$503/OK_Header
/CorrectHEC
/I$503/ChNumRegCE
/SNRegCE
/Payload
/I$503/RL_IndexEQ46
/Idle_Cell
11635860.0
11637640.0
11639420.0
11641200.0
11642980.0
11644760.0
11646540.0
11648320.0
11650100.0
Time (ns)

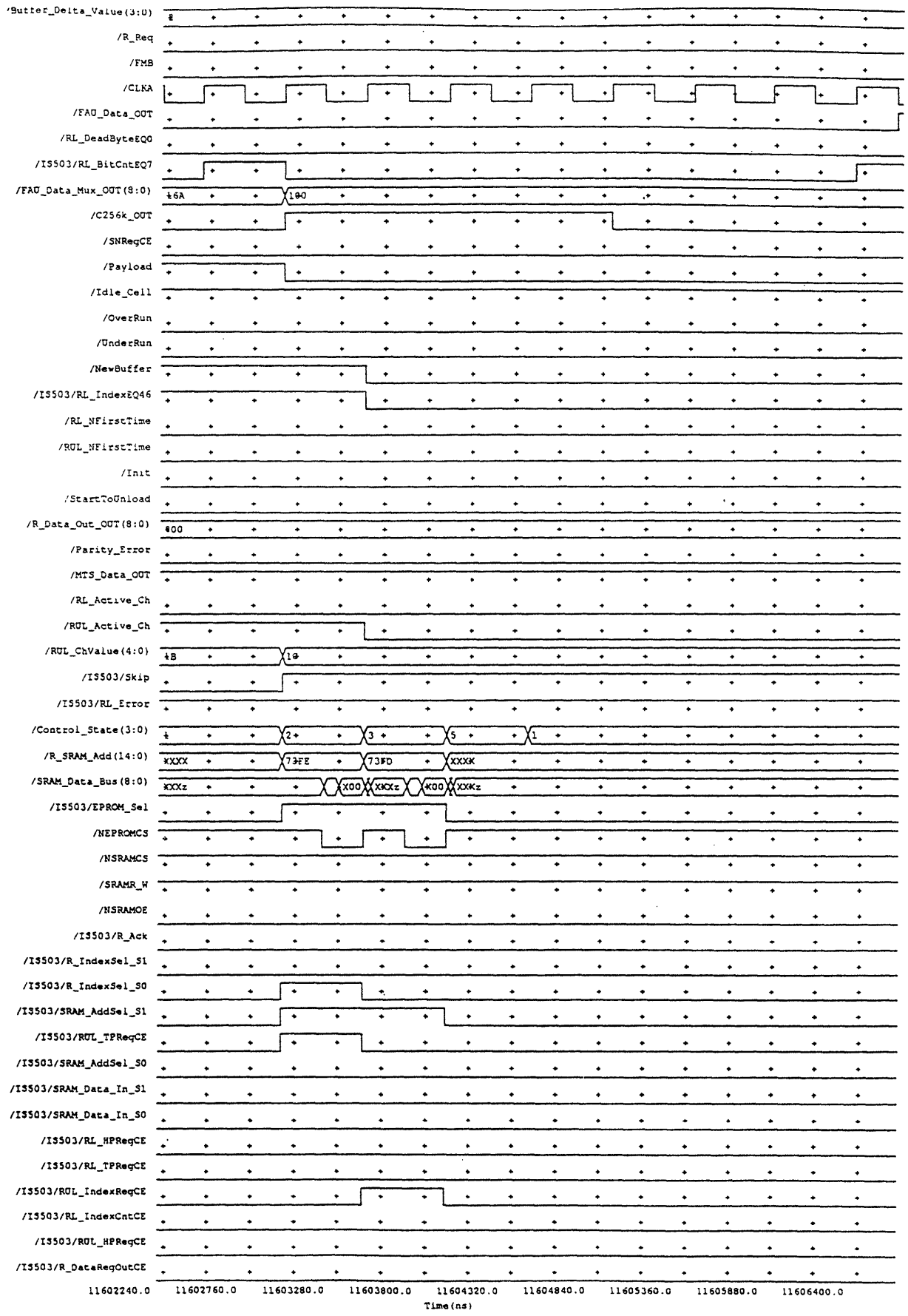
Appendix I Receiver Simulation: Loading Data

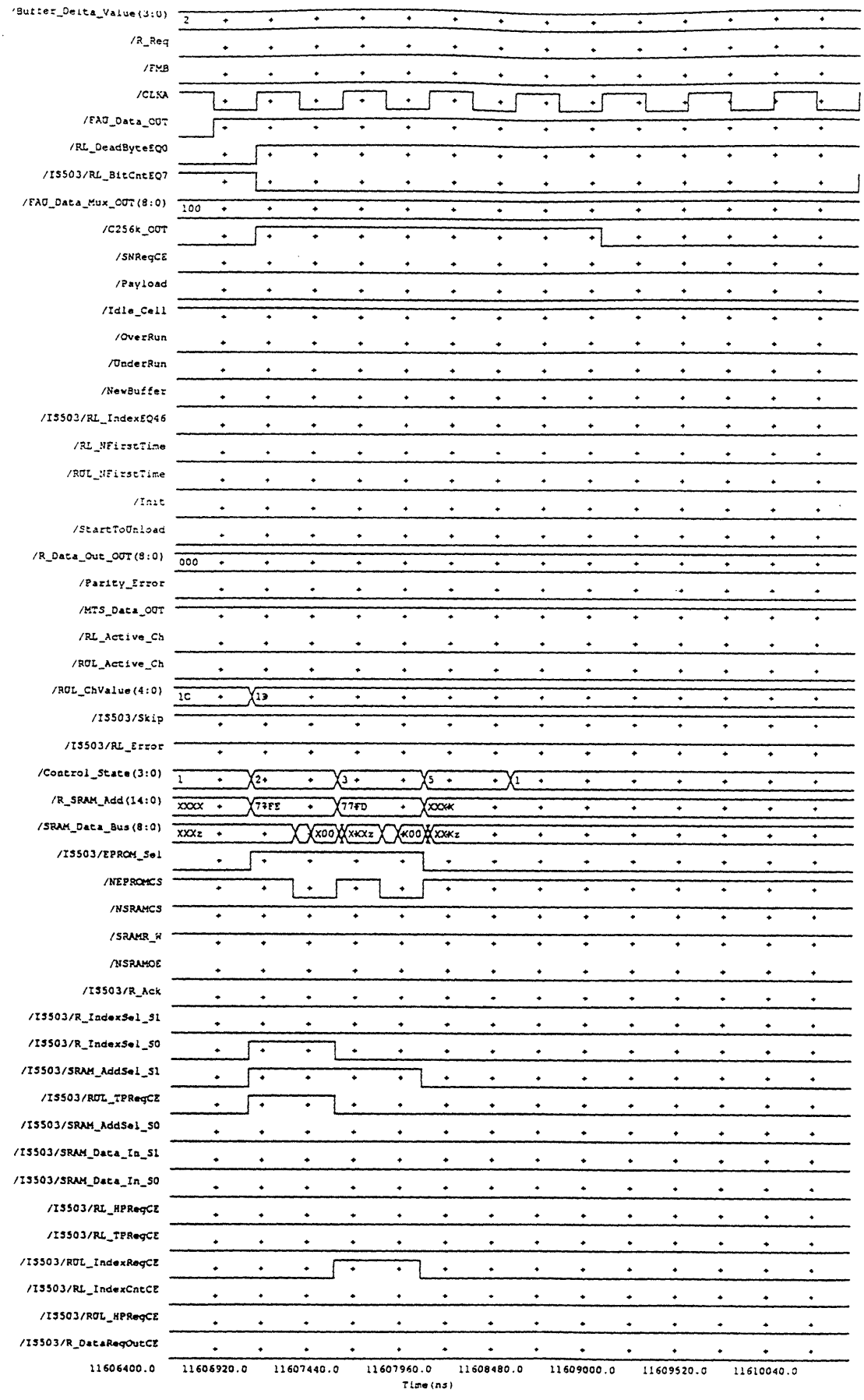
The timing simulation shown in this Appendix is for the Receiver. Only time slots 27 and 21 are active.

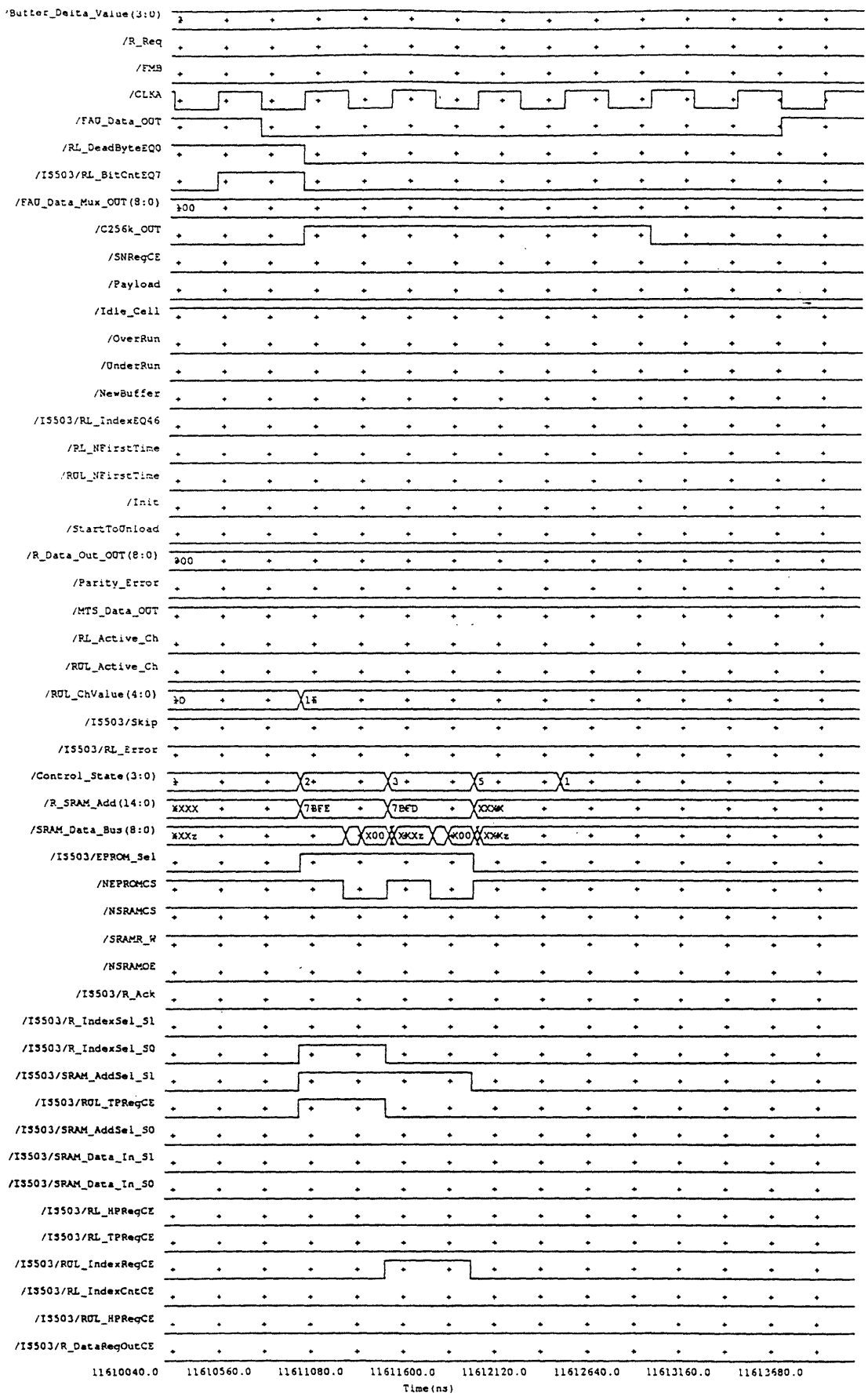
Simulation Time Period	Pages	Simulation Description
11590000 - 11650000	160-174	Beginning of a reception of an ATM cell for channel 31 Unloading of data has not started
11870000 - 11890000	175-178	End of the ATM cell shown above

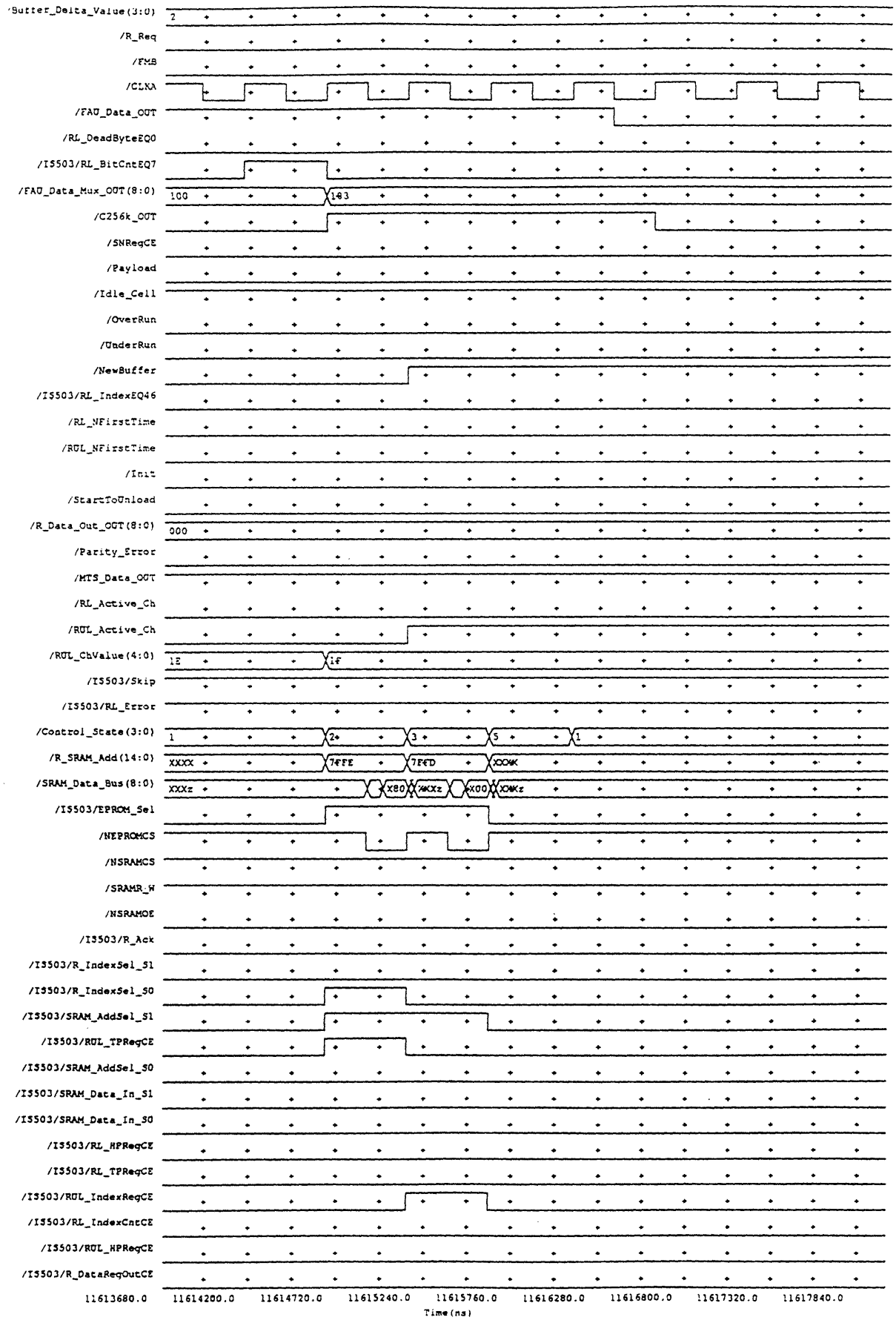

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RUL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPRegCE
/IS503/R_DataReqOutCE
11594960.0
11595480.0
11596000.0
11596520.0
11597040.0
11597560.0
11598080.0
11598600.0
Time(ns)

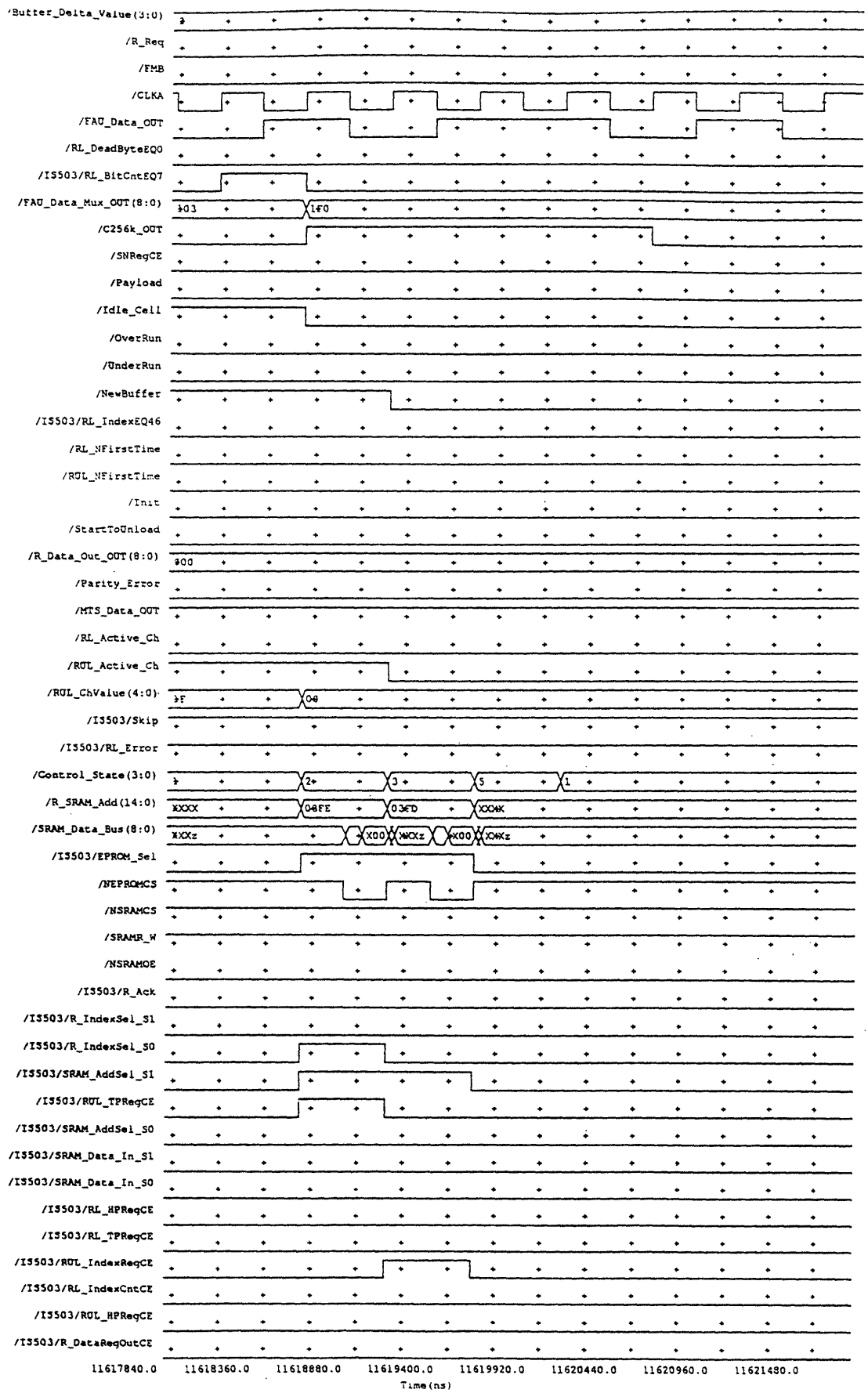

/Buffer_Delta_Value(3:0)
/R_Req
/EMB
/CLKA
/FA0_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FA0_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RUL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPRegCE
/IS503/R_DataRegOutCE
11598600.0
11599120.0
11599640.0
11600160.0
11600680.0
11601200.0
11601720.0
11602240.0
Time(ns)

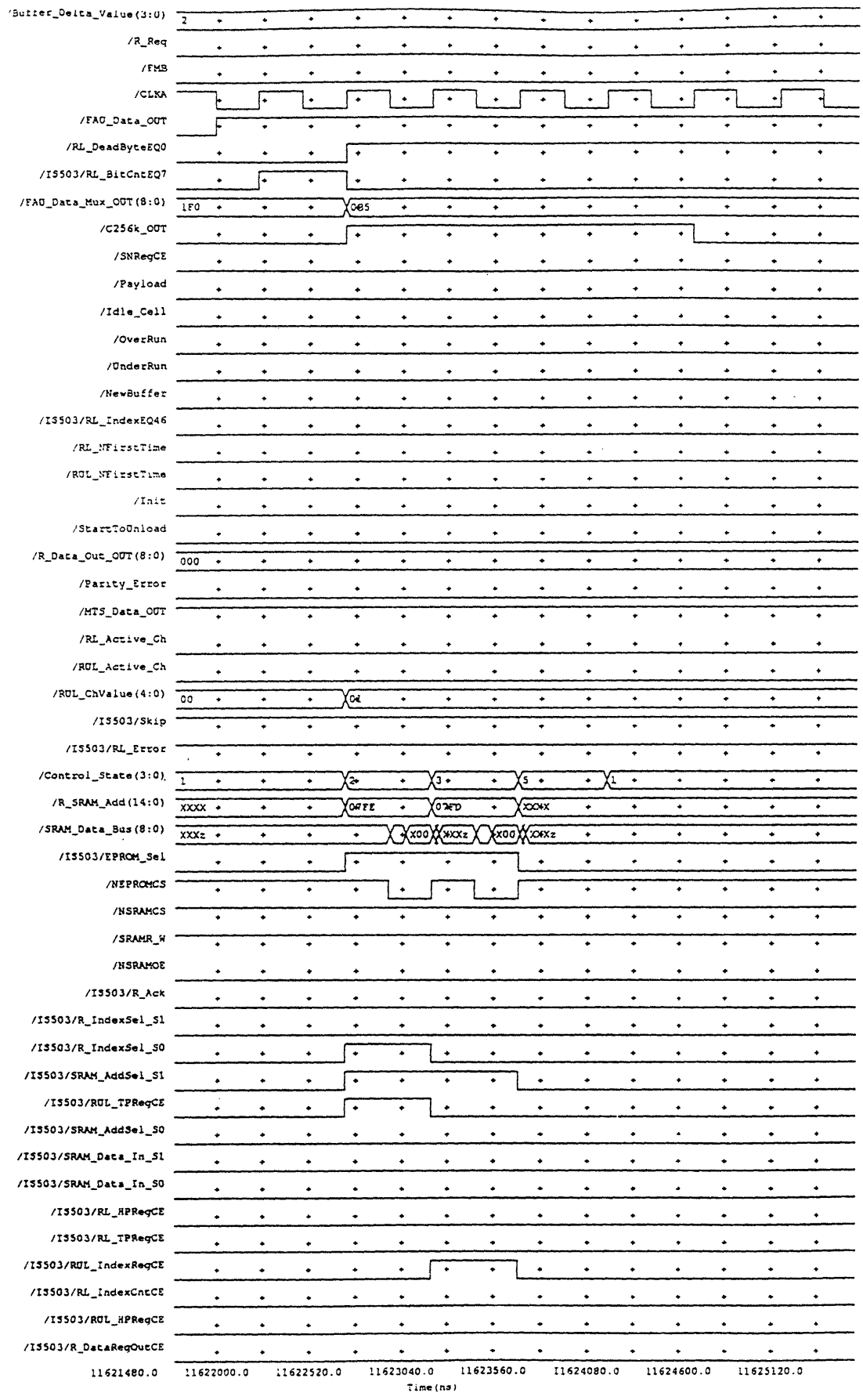

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPReqCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPReqCE
/IS503/RL_TPReqCE
/IS503/RUL_IndexReqCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPReqCE
/IS503/R_DataReqOutCE
11602240.0
11602760.0
11603280.0
11603800.0
11604320.0
11604840.0
11605360.0
11605880.0
11606400.0
Time(ns)

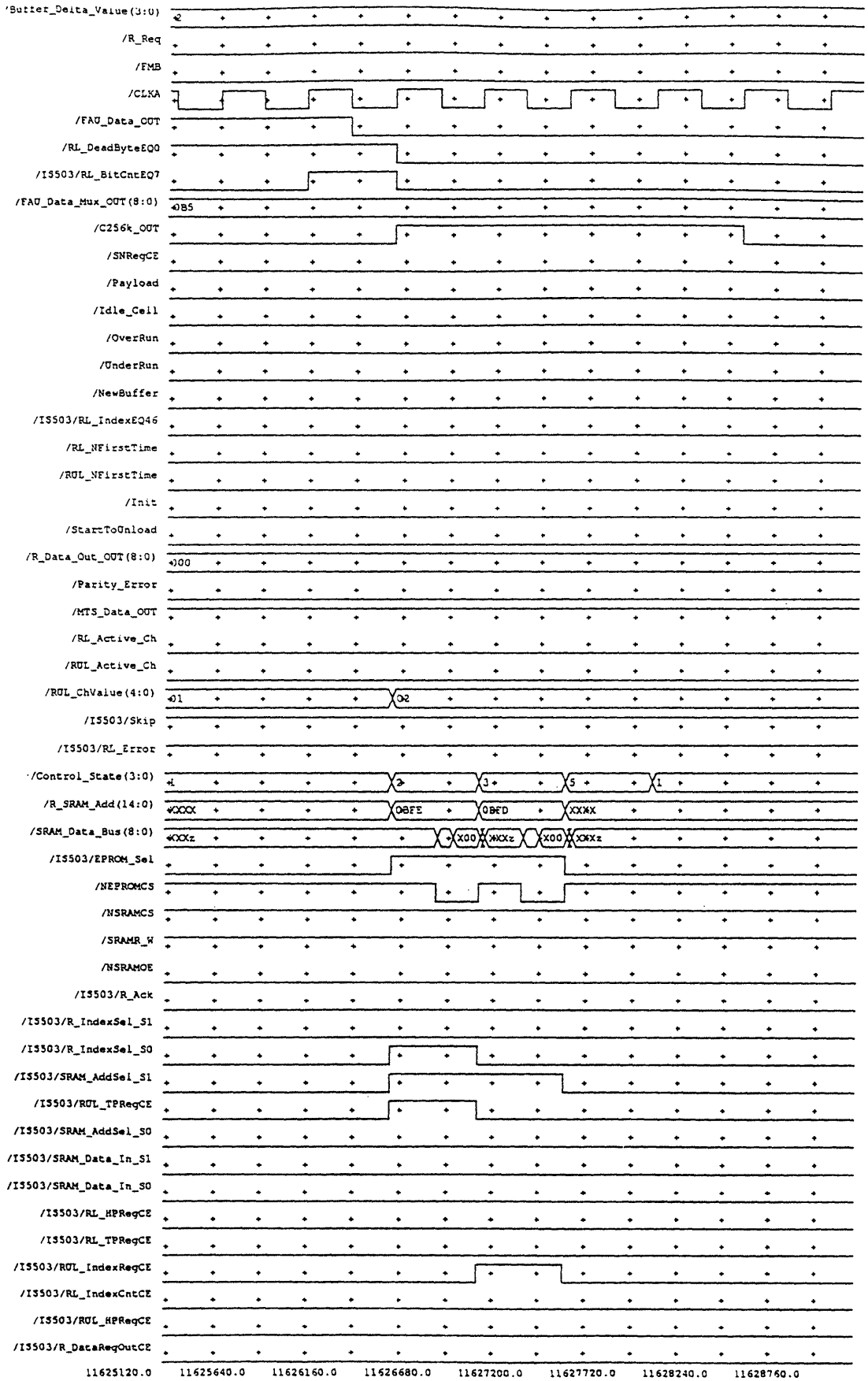

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPReqCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPReqCE
/IS503/RL_TPReqCE
/IS503/RUL_IndexReqCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPReqCE
/IS503/R_DataReqOutCE
11606400.0
11606920.0
11607440.0
11607960.0
11608480.0
11609000.0
11609520.0
11610040.0
Time(ns)

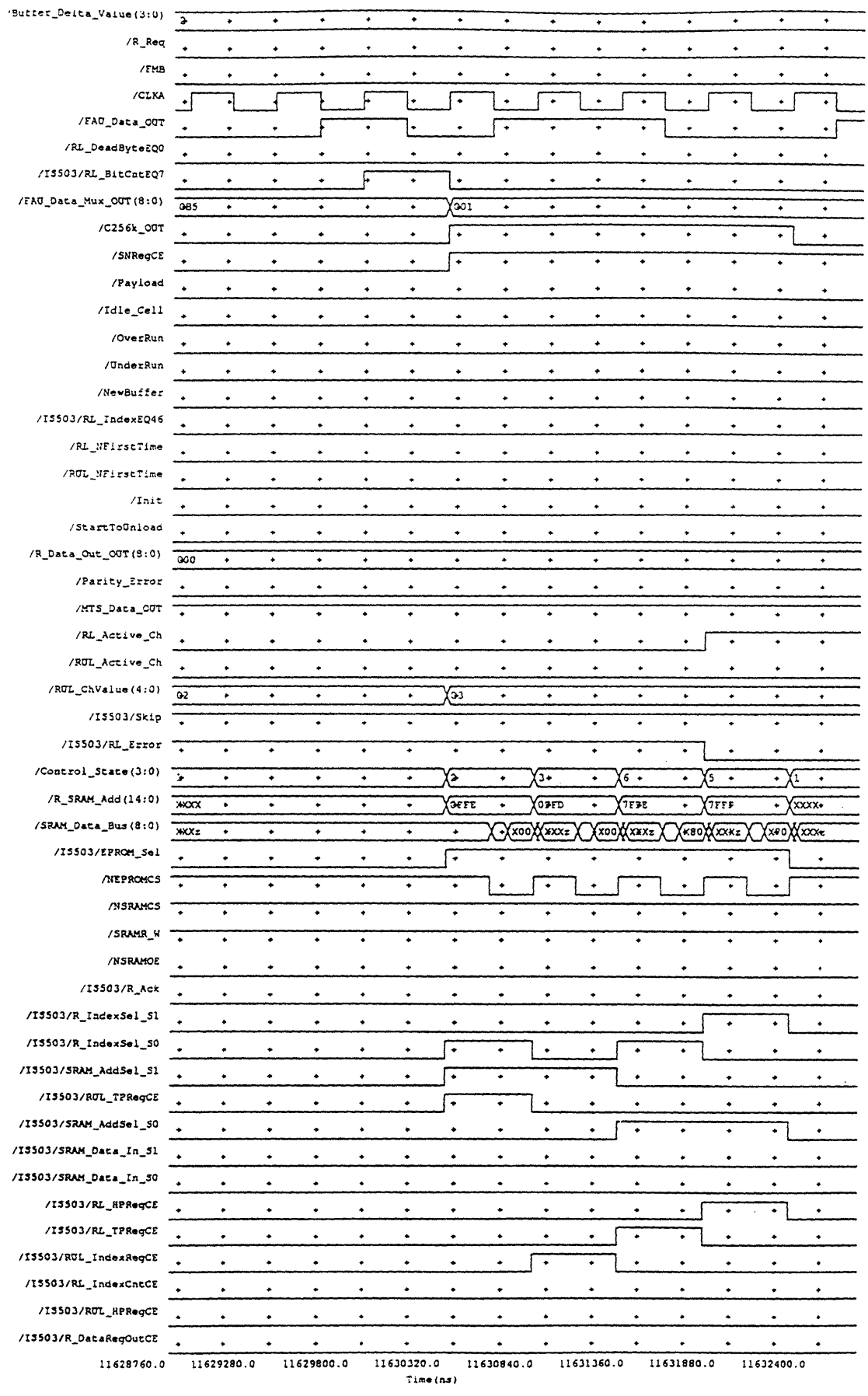

'Butter_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPReqCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPReqCE
/IS503/RL_TPReqCE
/IS503/RUL_IndexReqCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPReqCE
/IS503/R_DataReqOutCE
11610040.0
11610560.0
11611080.0
11611600.0
11612120.0
11612640.0
11613160.0
11613680.0
Time(ns)

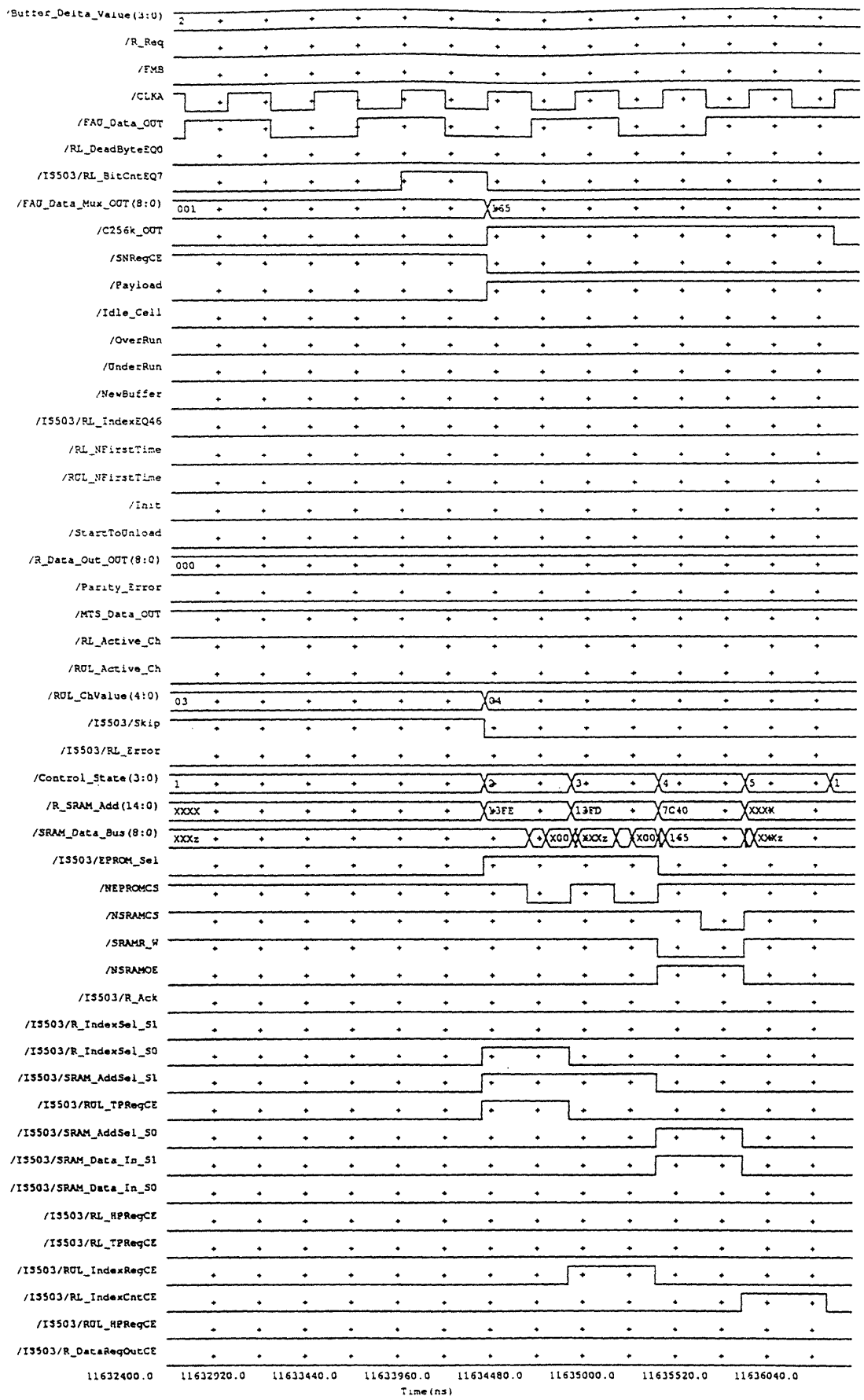

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RUL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPRegCE
/IS503/R_DataReqOutCE
11613680.0
11614200.0
11614720.0
11615240.0
11615760.0
11616280.0
11616800.0
11617320.0
11617840.0
Time(ns)

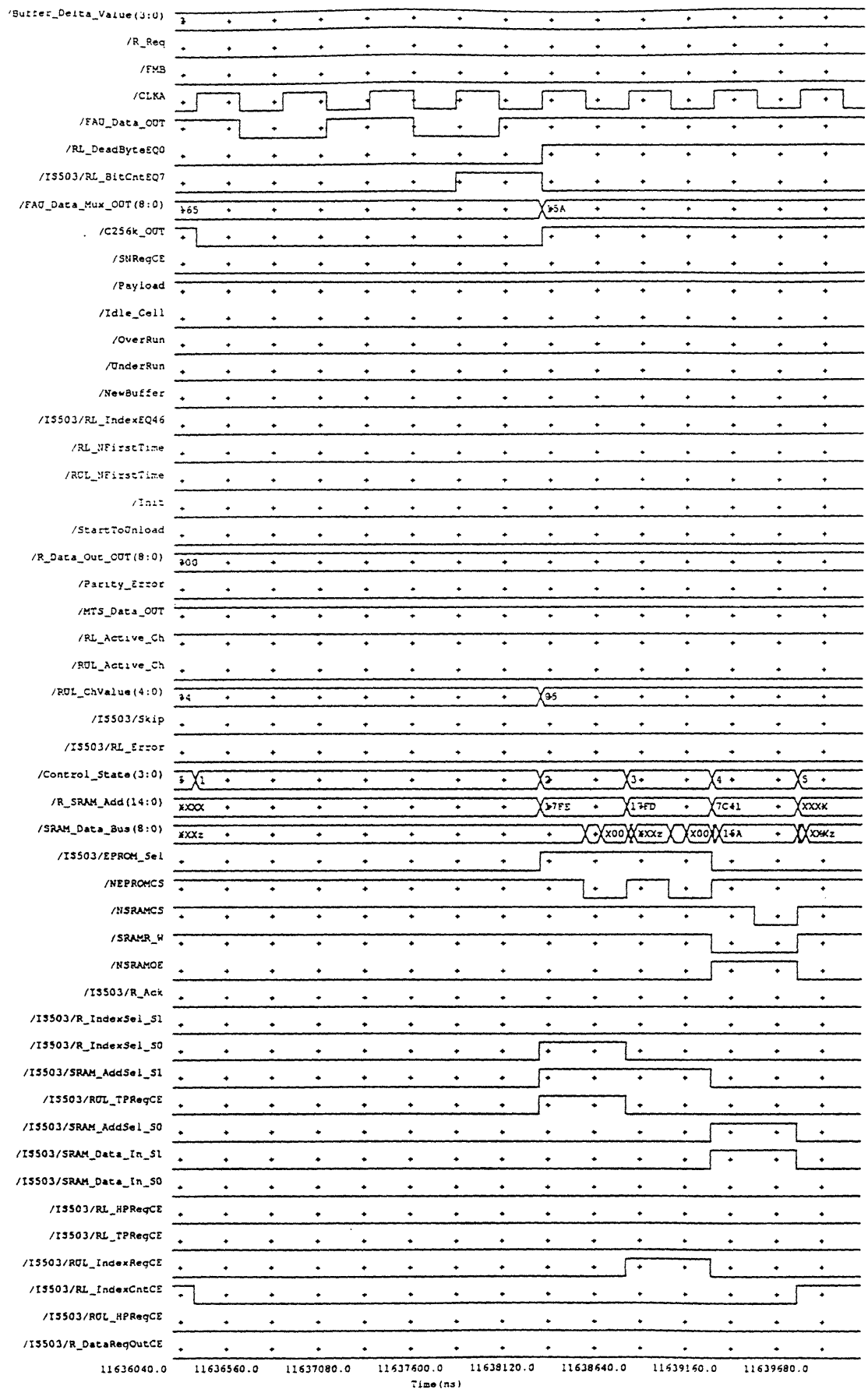

/Butter_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPReqCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPReqCE
/IS503/RL_TPReqCE
/IS503/RUL_IndexReqCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPReqCE
/IS503/R_DataReqOutCE
11617840.0
11618360.0
11618880.0
11619400.0
11619920.0
11620440.0
11620960.0
11621480.0
Time(ns)

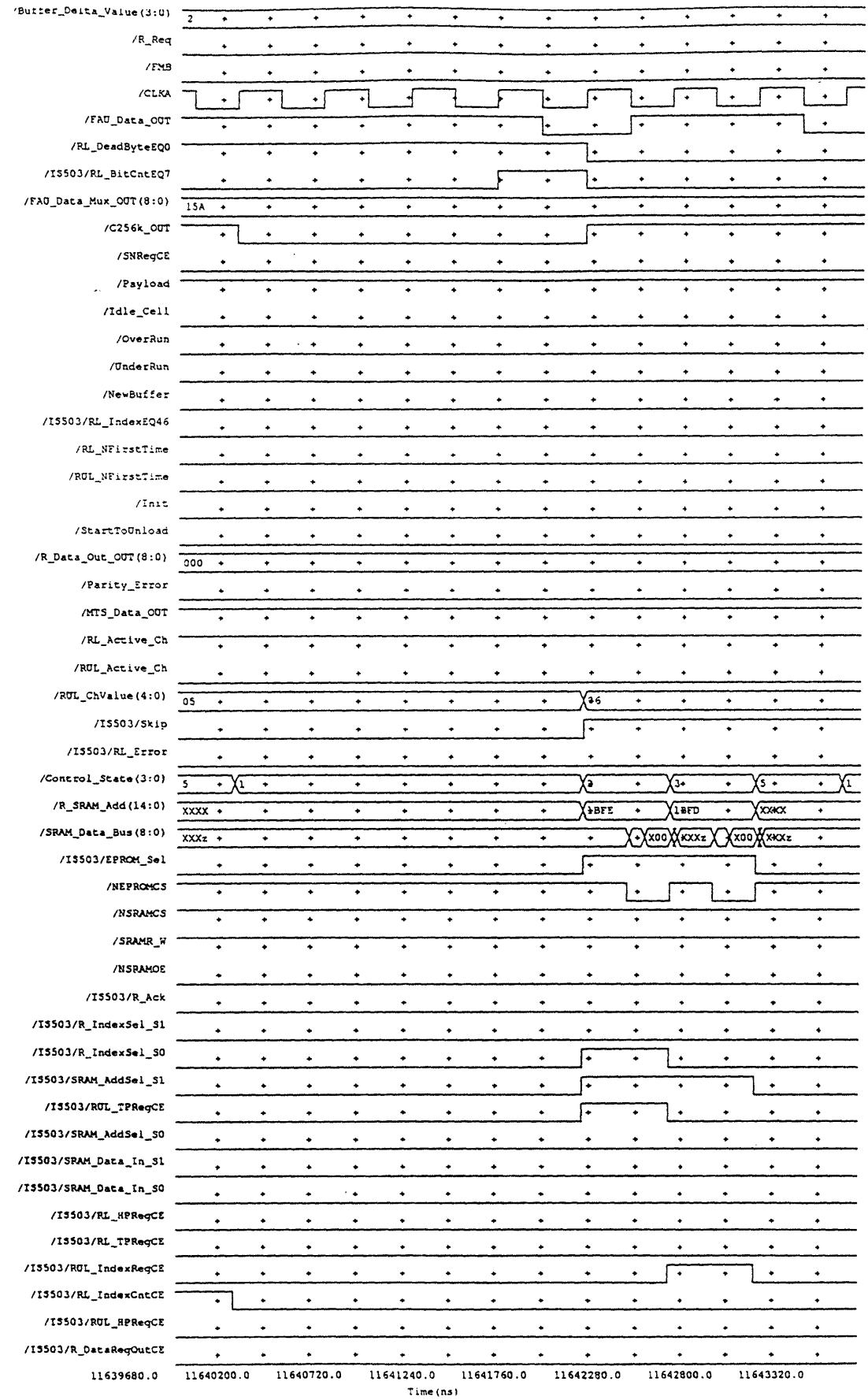

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RUL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPRegCE
/IS503/R_DataRegOutCE
11621480.0
11622000.0
11622520.0
11623040.0
11623560.0
11624080.0
11624600.0
11625120.0
Time(ns)

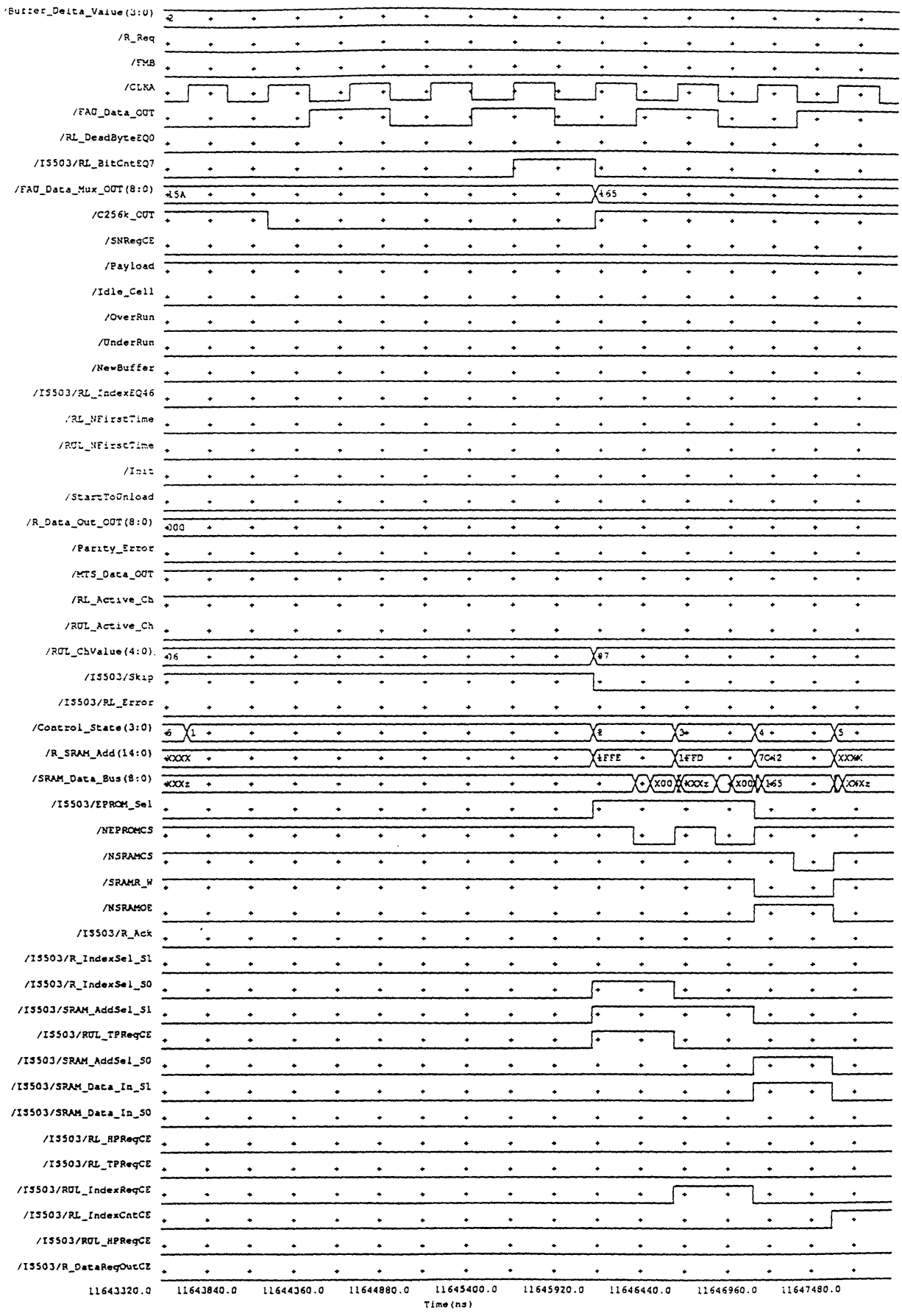

'Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RUL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPRegCE
/IS503/R_DataReqOutCE
0B5
00
01
02
1
3
5
1
0BFE
0BFD
XXXX
X00
X00
11625120.0
11625640.0
11626160.0
11626680.0
11627200.0
11627720.0
11628240.0
11628760.0
Time(ns)


'Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/RUL_TPRegCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPRegCE
/I5503/RL_TPRegCE
/I5503/RUL_IndexRegCE
/I5503/RL_IndexCntCE
/I5503/RUL_HPRegCE
/I5503/R_DataRegOutCE
11628760.0
11629280.0
11629800.0
11630320.0
11630840.0
11631360.0
11631880.0
11632400.0
Time(ns)

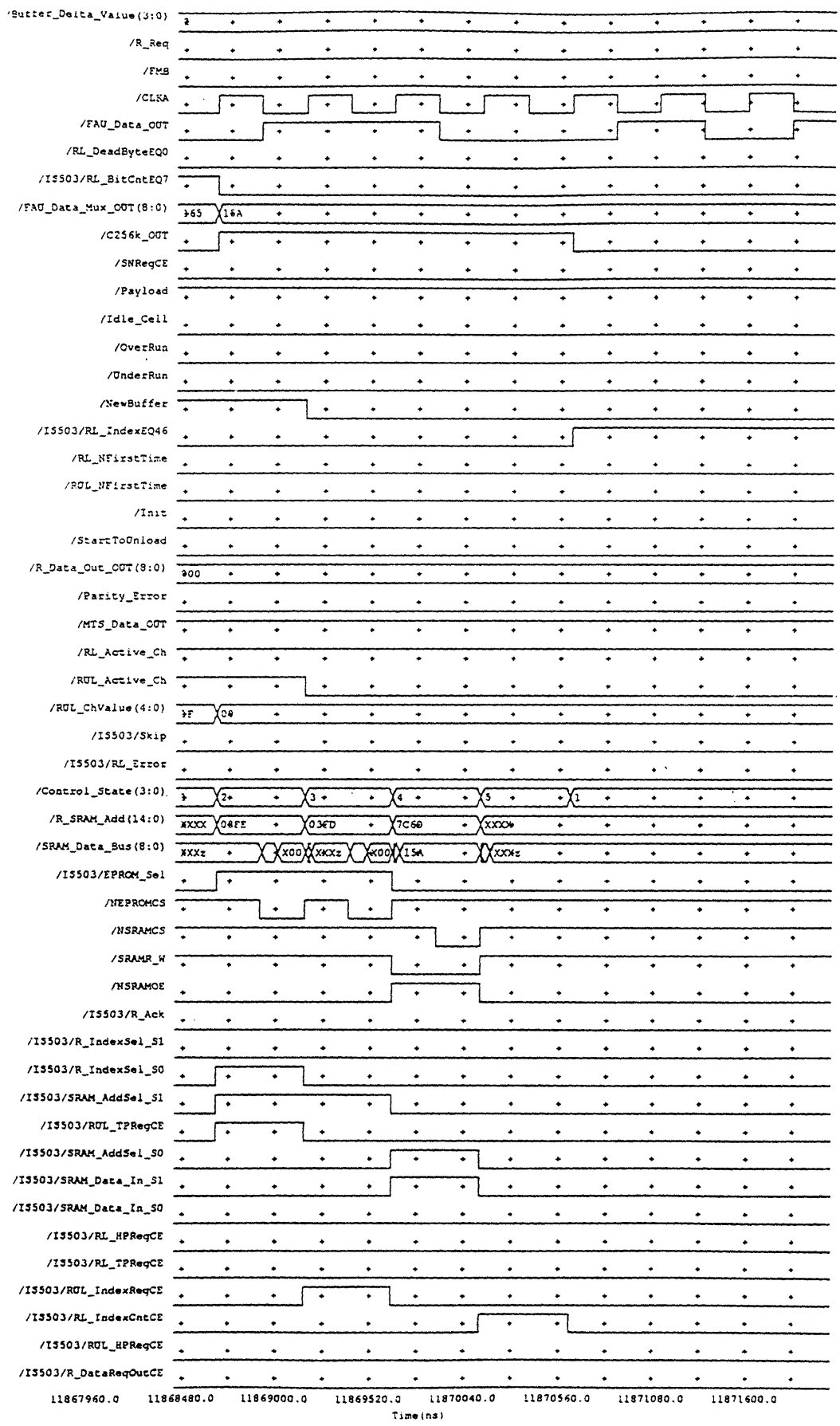

/Butter_Delta_Value(3:0)
/R_Req
/FMS
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RCL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RCL_Active_Ch
/RCL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RCL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RCL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RCL_HPRegCE
/IS503/R_DataReqOutCE
11632400.0
11632920.0
11633440.0
11633960.0
11634480.0
11635000.0
11635520.0
11636040.0
Time(ns)

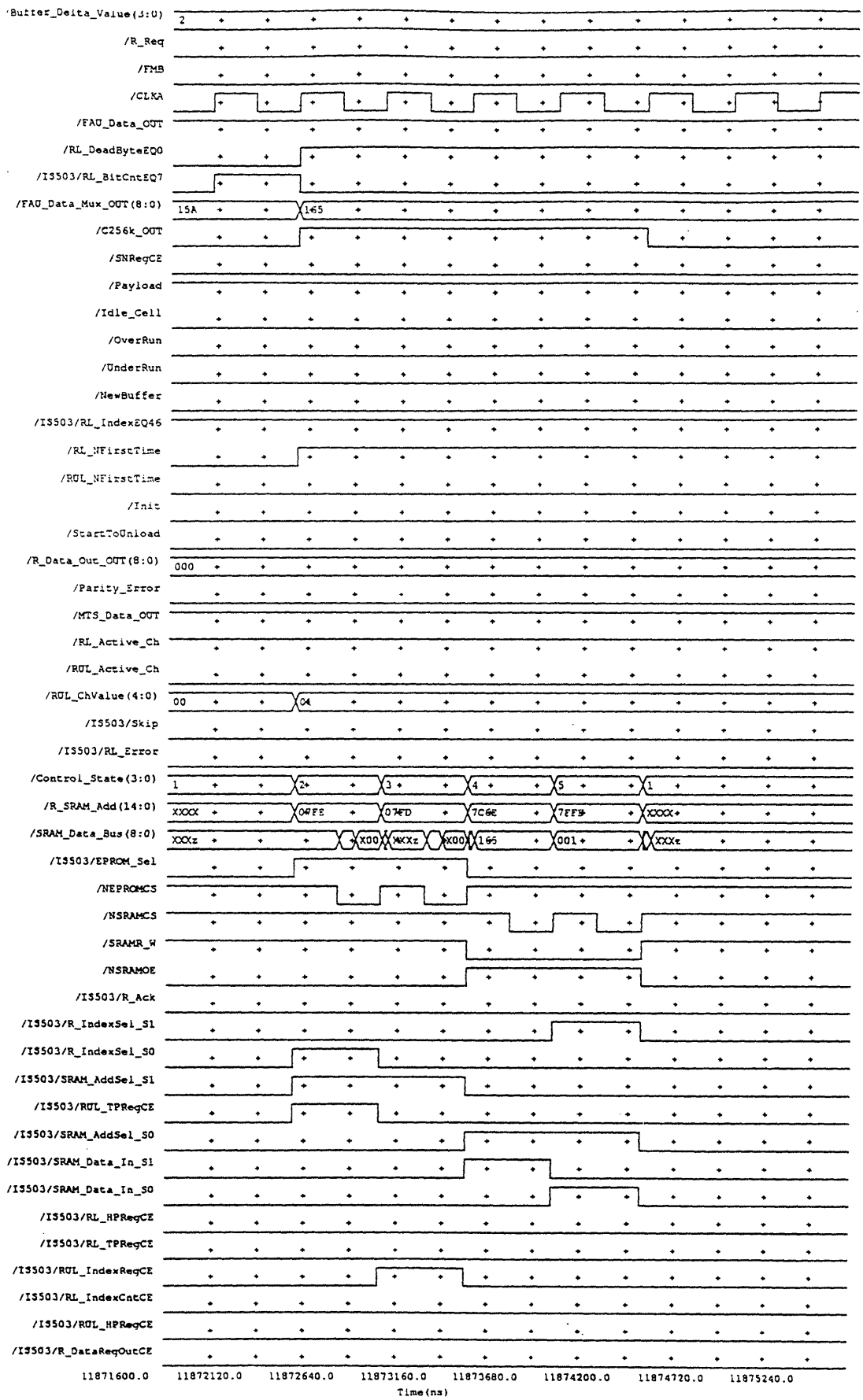

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPReqCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPReqCE
/IS503/RL_TPReqCE
/IS503/RUL_IndexReqCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPReqCE
/IS503/R_DataReqOutCE
11636040.0
11636560.0
11637080.0
11637600.0
11638120.0
11638640.0
11639160.0
11639680.0
Time(ns)

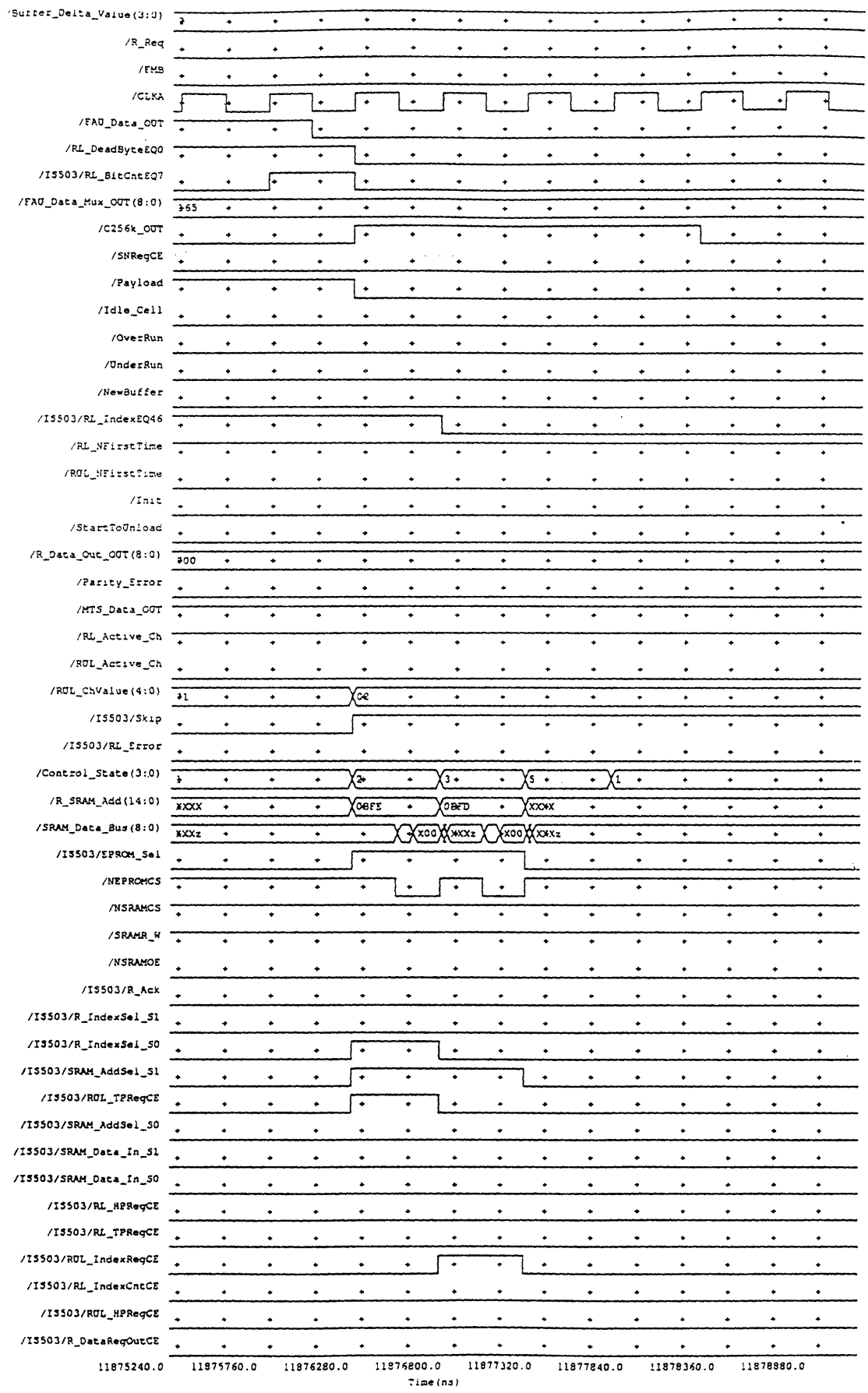

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPReqCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPReqCE
/IS503/RL_TPReqCE
/IS503/RUL_IndexReqCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPReqCE
/IS503/R_DataReqOutCE
11639680.0
11640200.0
11640720.0
11641240.0
11641760.0
11642280.0
11642800.0
11643320.0
Time(ns)

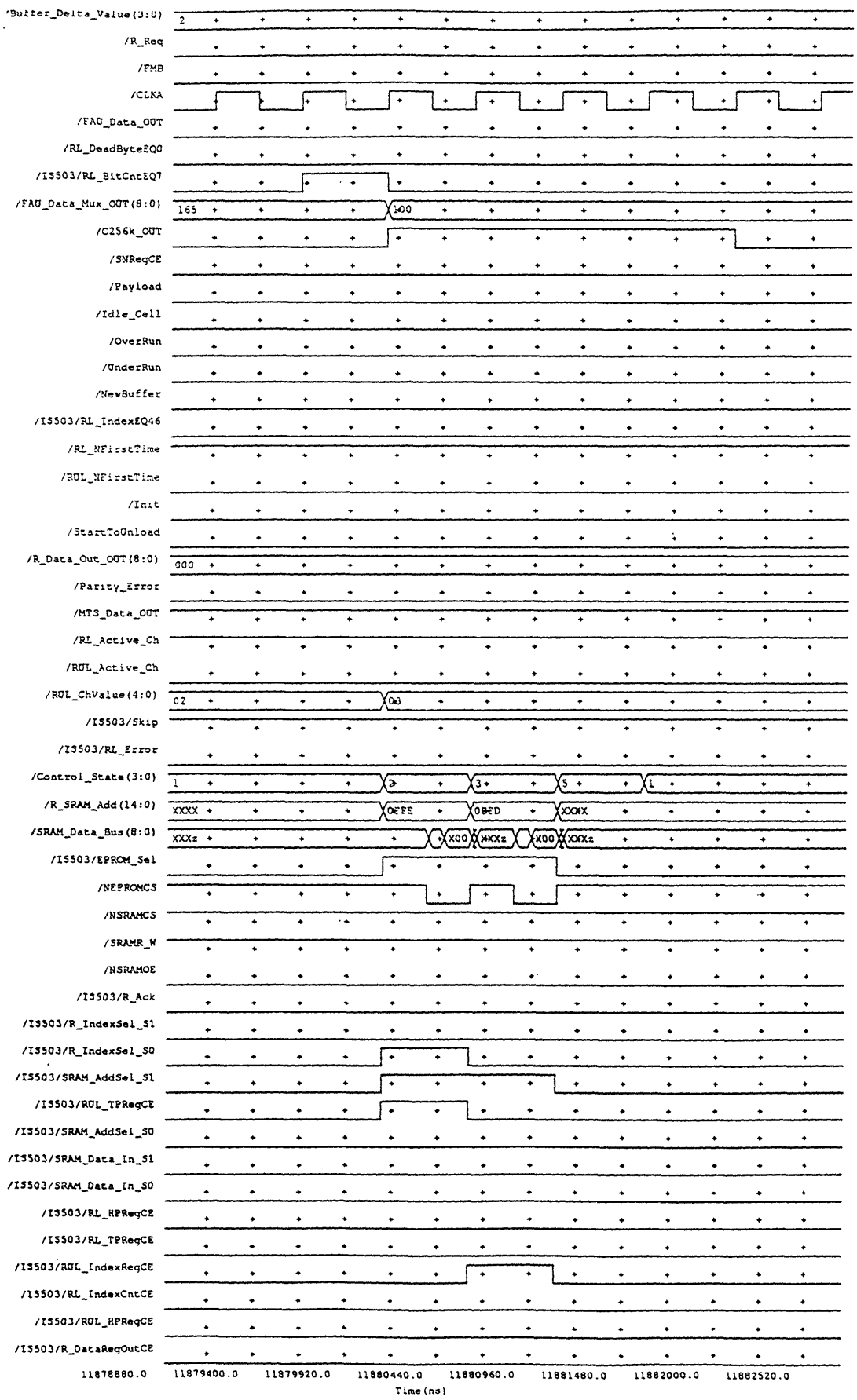

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RUL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPRegCE
/IS503/R_DataRegOutCE
11643320.0
11643840.0
11644360.0
11644880.0
11645400.0
11645920.0
11646440.0
11646960.0
11647480.0
Time(ns)


/Butter_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RUL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPRegCE
/IS503/R_DataRegOutCE
11647480.0
11648000.0
11648520.0
11649040.0
11649560.0
11650080.0
11650600.0
11651120.0
Time(ns)


/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/RUL_TPRegCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPRegCE
/I5503/RL_TPRegCE
/I5503/RUL_IndexRegCE
/I5503/RL_IndexCntCE
/I5503/RUL_HPRegCE
/I5503/R_DataReqOutCE
11867960.0
11868480.0
11869000.0
11869520.0
11870040.0
11870560.0
11871080.0
11871600.0
Time(ns)


'Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPRegCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPRegCE
/IS503/RL_TPRegCE
/IS503/RUL_IndexRegCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPRegCE
/IS503/R_DataReqOutCE
11871600.0
11872120.0
11872640.0
11873160.0
11873680.0
11874200.0
11874720.0
11875240.0
Time(ns)

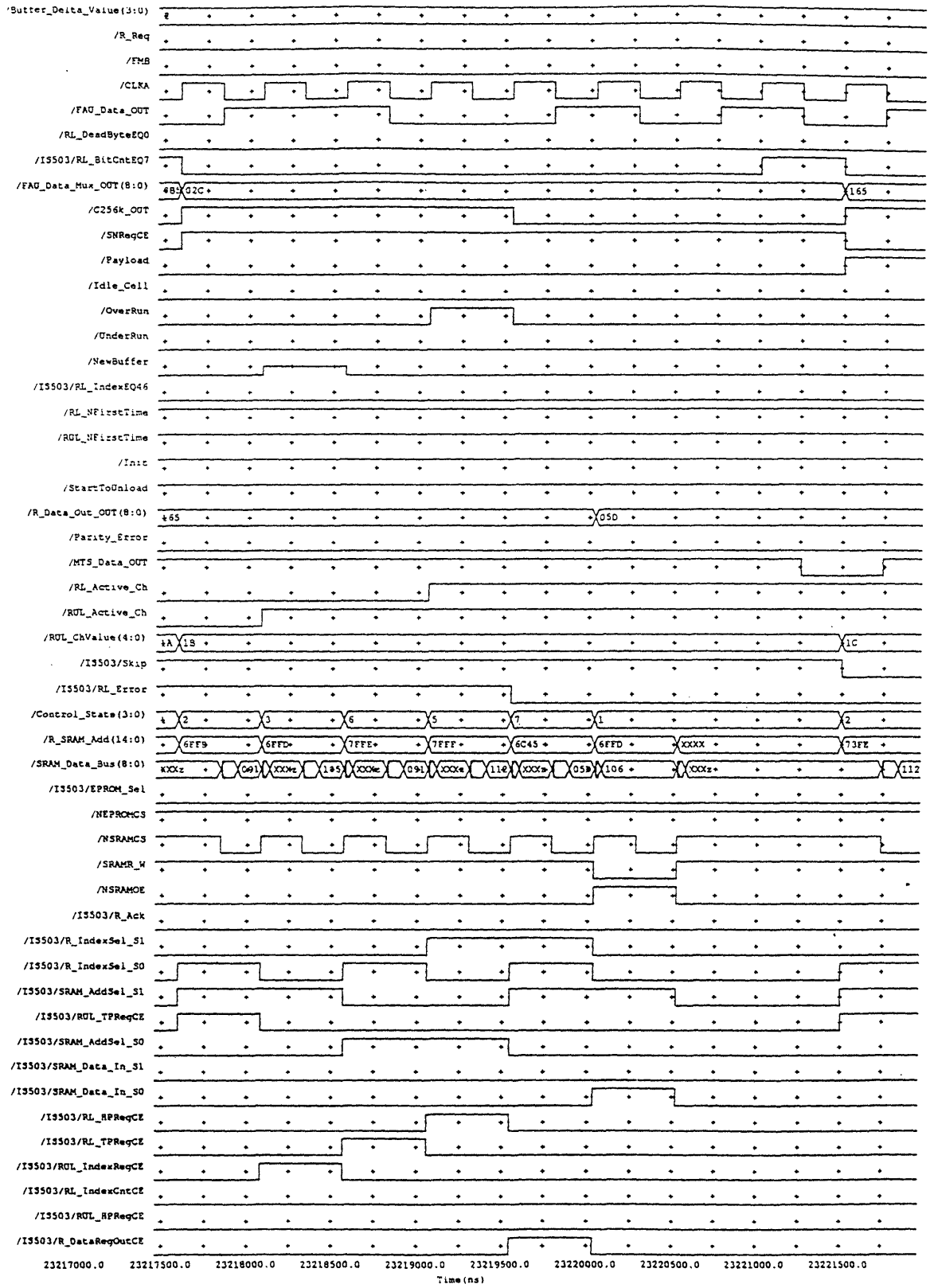

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/RUL_TPRegCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPRegCE
/I5503/RL_TPRegCE
/I5503/RUL_IndexRegCE
/I5503/RL_IndexCntCE
/I5503/RUL_HPRegCE
/I5503/R_DataRegOutCE
11875240.0
11875760.0
11876280.0
11876800.0
11877320.0
11877840.0
11878360.0
11878880.0
Time(ns)

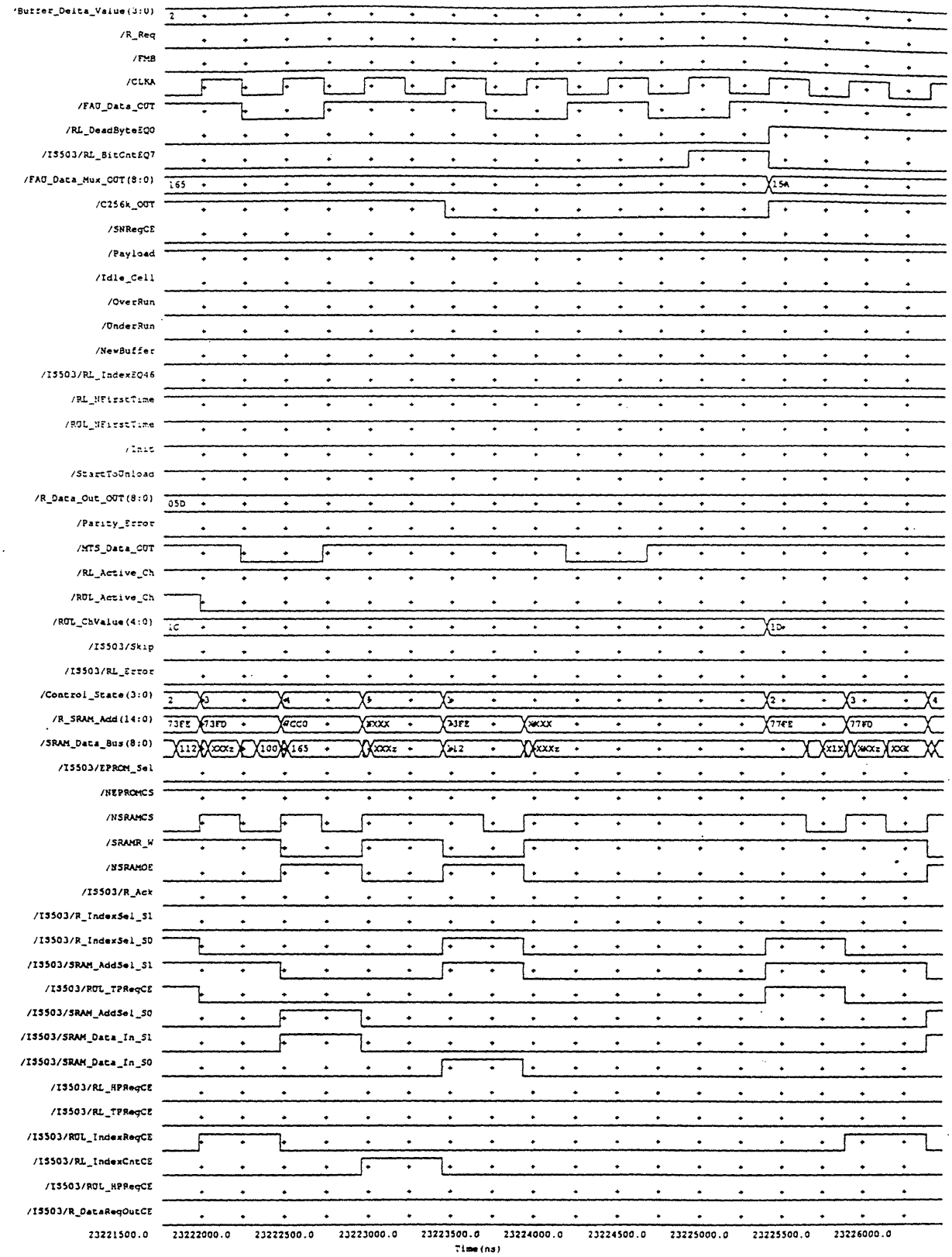

'Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/IS503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/IS503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/IS503/Skip
/IS503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/IS503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/IS503/R_Ack
/IS503/R_IndexSel_S1
/IS503/R_IndexSel_S0
/IS503/SRAM_AddSel_S1
/IS503/RUL_TPReqCE
/IS503/SRAM_AddSel_S0
/IS503/SRAM_Data_In_S1
/IS503/SRAM_Data_In_S0
/IS503/RL_HPReqCE
/IS503/RL_TPReqCE
/IS503/RUL_IndexReqCE
/IS503/RL_IndexCntCE
/IS503/RUL_HPReqCE
/IS503/R_DataReqOutCE
11878880.0
11879400.0
11879920.0
11880440.0
11880960.0
11881480.0
11882000.0
11882520.0
Time(ns)

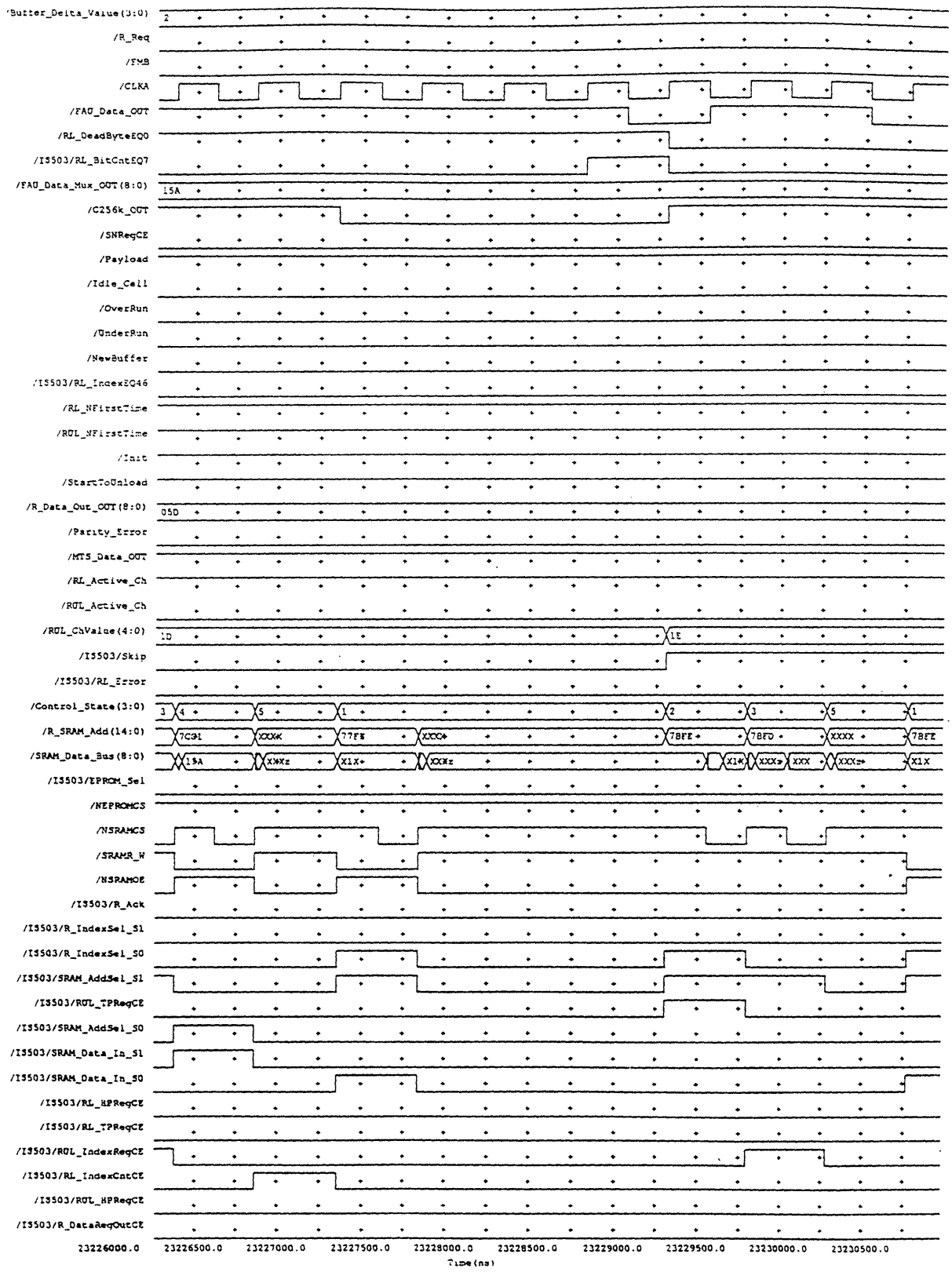
Appendix J Receiver Simulation: Loading and Unloading Data

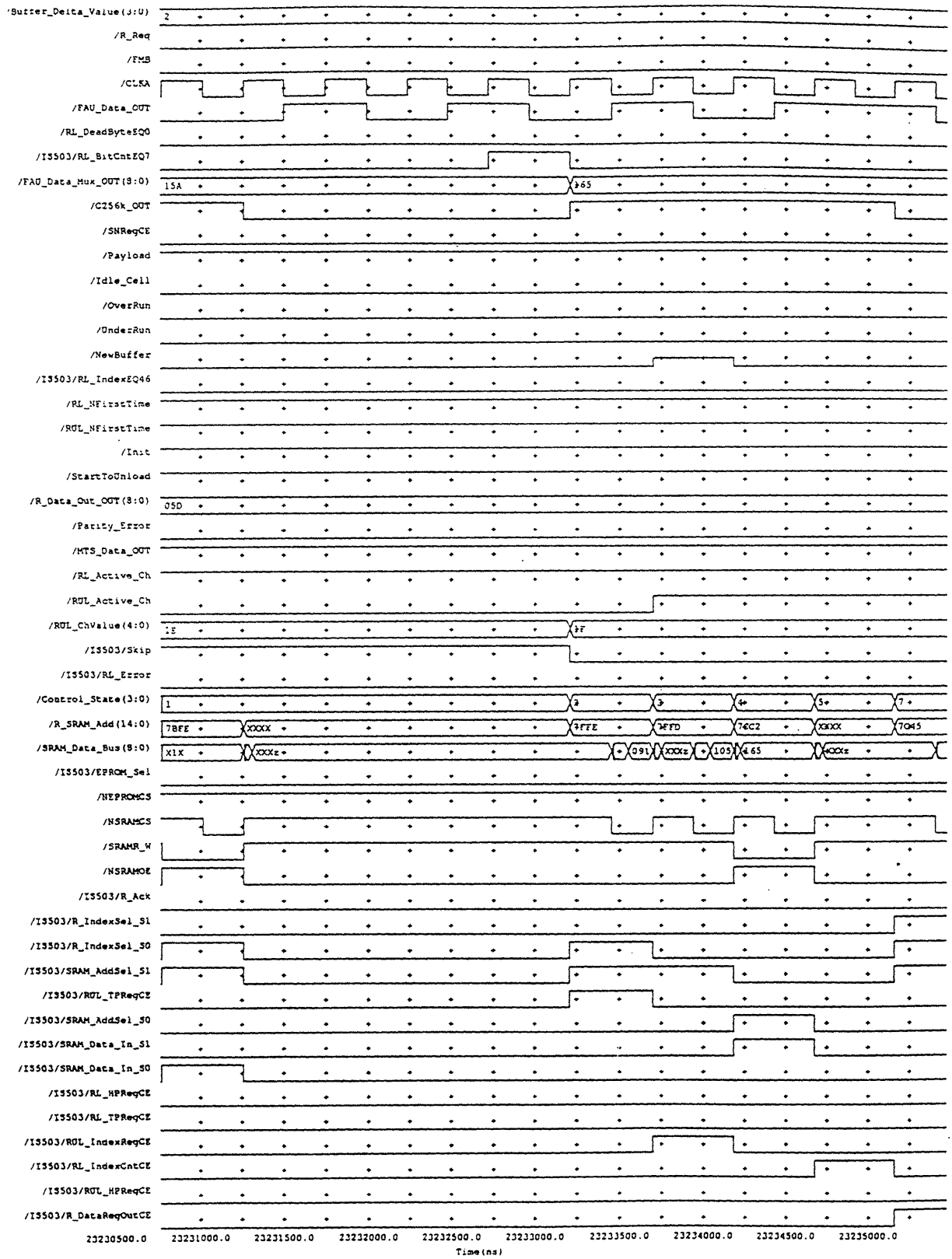
The timing simulation shown in this Appendix is for the Receiver. Only time slots 27 and 31 are active.

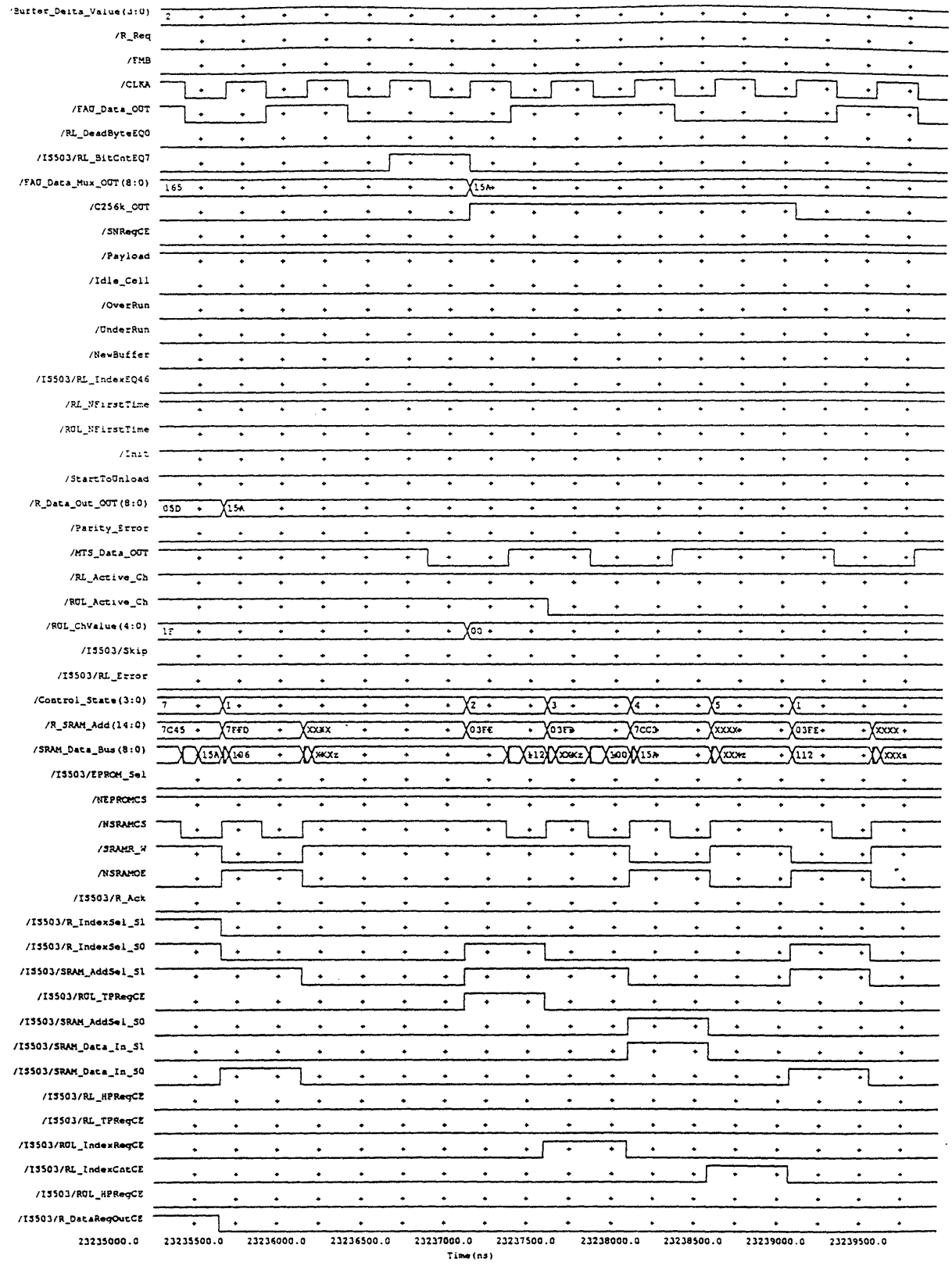
Simulation Time Period	Pages	Simulation Description
23220000 - 2326000	180-187	Receiving ATM cells while also unloading data. StartToUnload = 1.

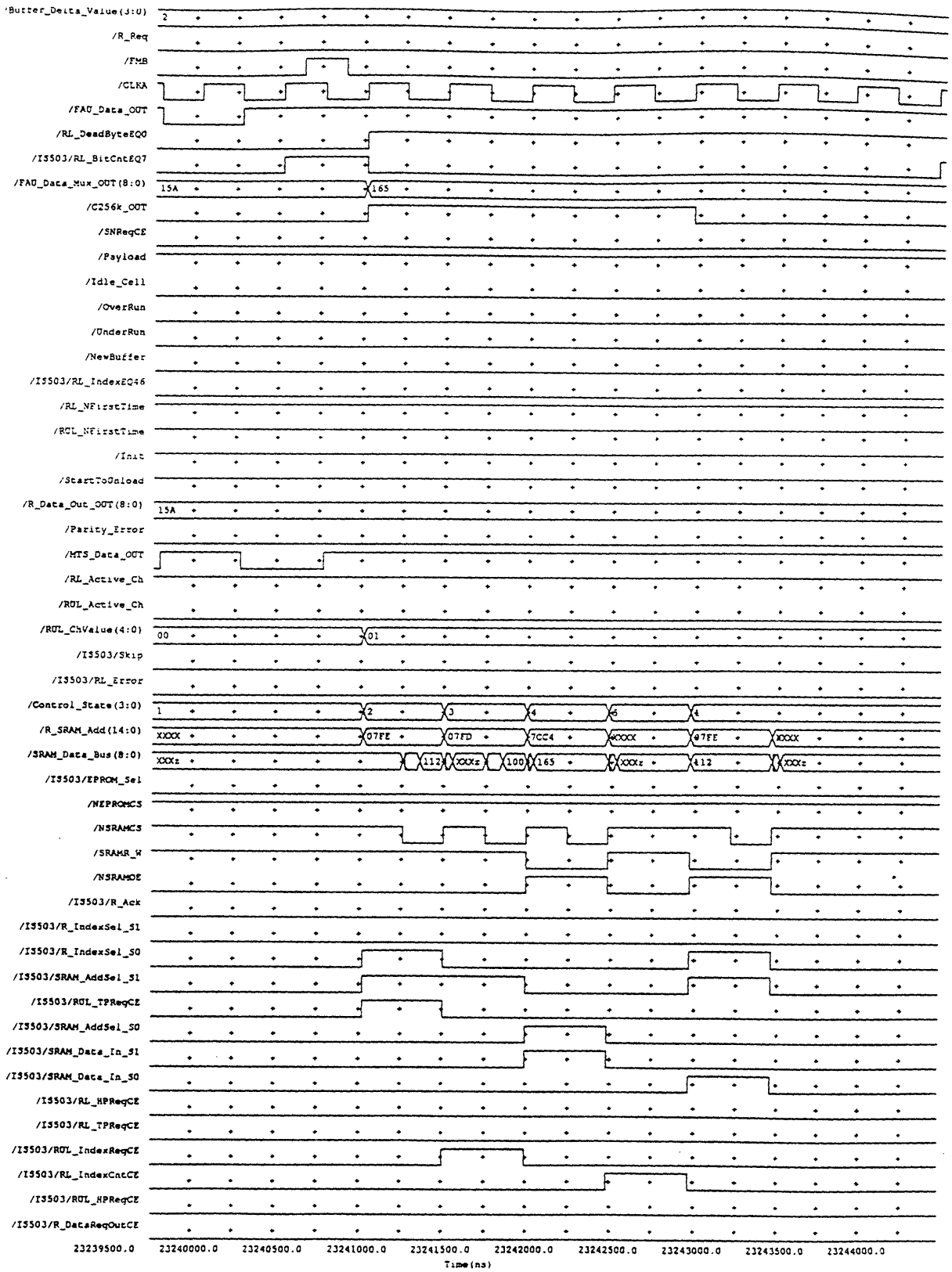

'Butter_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I3503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I3503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/I3503/Skip
/I3503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I3503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I3503/R_Ack
/I3503/R_IndexSel_S1
/I3503/R_IndexSel_S0
/I3503/SRAM_AddSel_S1
/I3503/RUL_TPReqCE
/I3503/SRAM_AddSel_S0
/I3503/SRAM_Data_In_S1
/I3503/SRAM_Data_In_S0
/I3503/RL_HPReqCE
/I3503/RL_TPReqCE
/I3503/RUL_IndexReqCE
/I3503/RL_IndexCntCE
/I3503/RUL_HPReqCE
/I3503/R_DataReqOutCE
23217000.0
23217500.0
23218000.0
23218500.0
23219000.0
23219500.0
23220000.0
23220500.0
23221000.0
23221500.0
Time(ns)

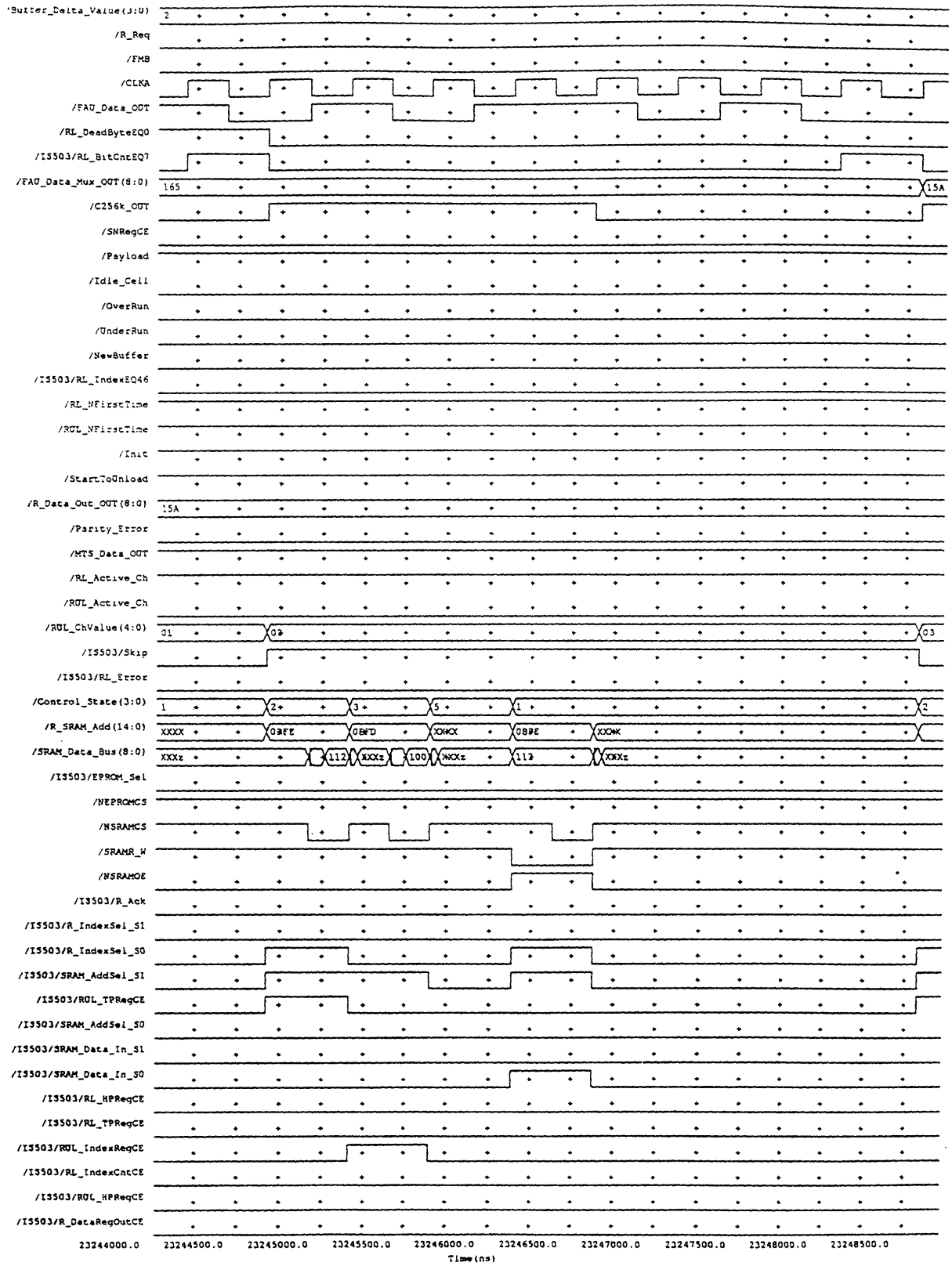

/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/ROL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/ROL_Active_Ch
/ROL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/ROL_TPReqCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPReqCE
/I5503/RL_TPReqCE
/I5503/ROL_IndexReqCE
/I5503/RL_IndexCntCE
/I5503/ROL_HPReqCE
/I5503/R_DataReqOutCE
23221500.0
23222000.0
23222500.0
23223000.0
23223500.0
23224000.0
23224500.0
23225000.0
23225500.0
23226000.0
Time(ns)

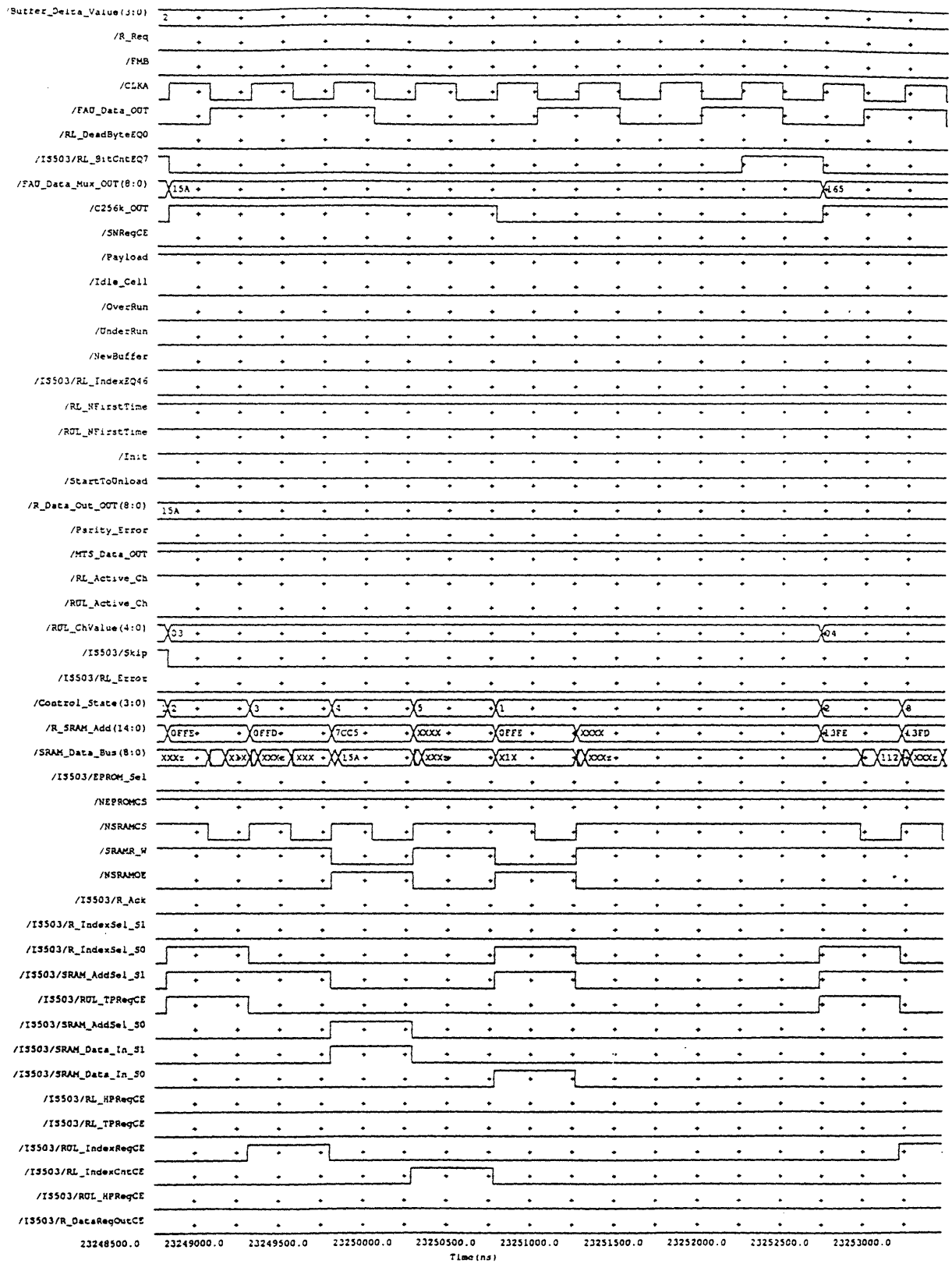

/Butter_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/RGL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RGL_Active_Ch
/RGL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/RGL_TPRegCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPRegCE
/I5503/RL_TPRegCE
/I5503/RGL_IndexRegCE
/I5503/RL_IndexCntCE
/I5503/RGL_HPRegCE
/I5503/R_DataReqOutCE
23226000.0
23226500.0
23227000.0
23227500.0
23228000.0
23228500.0
23229000.0
23229500.0
23230000.0
23230500.0
Time(ns)


/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I3503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(3:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I3503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(3:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/I3503/Skip
/I3503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(3:0)
/I3503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I3503/R_Ack
/I3503/R_IndexSel_S1
/I3503/R_IndexSel_S0
/I3503/SRAM_AddSel_S1
/I3503/RUL_TPReqCE
/I3503/SRAM_AddSel_S0
/I3503/SRAM_Data_In_S1
/I3503/SRAM_Data_In_S0
/I3503/RL_HPReqCE
/I3503/RL_TPReqCE
/I3503/RUL_IndexReqCE
/I3503/RL_IndexCntCE
/I3503/RUL_HPReqCE
/I3503/R_DataReqOutCE
23230500.0
23231000.0
23231500.0
23232000.0
23232500.0
23233000.0
23233500.0
23234000.0
23234500.0
23235000.0
Time(ns)


Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/RUL_TPRegCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPRegCE
/I5503/RL_TPRegCE
/I5503/RUL_IndexRegCE
/I5503/RL_IndexCntCE
/I5503/RUL_HPRegCE
/I5503/R_DataRegOutCE
23235000.0
23235500.0
23236000.0
23236500.0
23237000.0
23237500.0
23238000.0
23238500.0
23239000.0
23239500.0
Time(ns)


'Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/RCL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RCL_Active_Ch
/RCL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/RCL_TPReqCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPReqCE
/I5503/RL_TPReqCE
/I5503/RCL_IndexReqCE
/I5503/RL_IndexCntCE
/I5503/RCL_HPReqCE
/I5503/R_DataReqOutCE
23239500.0
23240000.0
23240500.0
23241000.0
23241500.0
23242000.0
23242500.0
23243000.0
23243500.0
23244000.0
Time(ns)


'Butter_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNRegCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/RUL_TPReqCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPReqCE
/I5503/RL_TPReqCE
/I5503/RUL_IndexReqCE
/I5503/RL_IndexCntCE
/I5503/RUL_HPReqCE
/I5503/R_DataReqOutCE
23244000.0
23244500.0
23245000.0
23245500.0
23246000.0
23246500.0
23247000.0
23247500.0
23248000.0
23248500.0
Time(ns)


/Buffer_Delta_Value(3:0)
/R_Req
/FMB
/CLKA
/FAU_Data_OUT
/RL_DeadByteEQ0
/I5503/RL_BitCntEQ7
/FAU_Data_Mux_OUT(8:0)
/C256k_OUT
/SNReqCE
/Payload
/Idle_Cell
/OverRun
/UnderRun
/NewBuffer
/I5503/RL_IndexEQ46
/RL_NFirstTime
/RUL_NFirstTime
/Init
/StartToUnload
/R_Data_Out_OUT(8:0)
/Parity_Error
/MTS_Data_OUT
/RL_Active_Ch
/RUL_Active_Ch
/RUL_ChValue(4:0)
/I5503/Skip
/I5503/RL_Error
/Control_State(3:0)
/R_SRAM_Add(14:0)
/SRAM_Data_Bus(8:0)
/I5503/EPROM_Sel
/NEPROMCS
/NSRAMCS
/SRAMR_W
/NSRAMOE
/I5503/R_Ack
/I5503/R_IndexSel_S1
/I5503/R_IndexSel_S0
/I5503/SRAM_AddSel_S1
/I5503/RUL_TPReqCE
/I5503/SRAM_AddSel_S0
/I5503/SRAM_Data_In_S1
/I5503/SRAM_Data_In_S0
/I5503/RL_HPReqCE
/I5503/RL_TPReqCE
/I5503/RUL_IndexReqCE
/I5503/RL_IndexCntCE
/I5503/RUL_HPReqCE
/I5503/R_DataReqOutCE
23248500.0
23249000.0
23249500.0
23250000.0
23250500.0
23251000.0
23251500.0
23252000.0
23252500.0
23253000.0
Time(ns)

Bibliography

ITU-TSS Recommendation I.361 "B-ISDN ATM Layer Specification," March 1993.

ITU-TSS Recommendation I.362 "B-ISDN ATM Adaptation Layer (AAL) Functional Description," March 1993.

ITU-TSS Recommendation I.363 "B-ISDN ATM Adaptation Layer (AAL) Specification," March 1993.

ITU-TSS Recommendation I.432 "B-ISDN User-Network Interface Physical Layer Specification," March 1993.

ITU-TSS Recommendation G.804 "ISDN Experts of Study Group 13," March 1993.

GR-113-CORE "Asynchronous Transfer Mode (ATM) and ATM Adaptation Layer (AAL) Protocols," Release 1, Bellcore, July 1994.

T1S1.5/94-184 "Proposed AAL for CBR voice and voiceband service," AT&T, July 1994.

A Description of The T-1 Carrier, R. L. Ponners, April 1992.

ATM User-Network Interface Specification (UNI 3.1), Version 3.1, The ATM Forum, May 1994.

Asynchronous Transfer Mode (ATM) User's Guide, W.A. Flanagan, Flatiron Publishing, Inc., New York, 1994.

The Programmable Logic Data Book, Xilinx, 1994.

A30808-X5182-X-1-59 "System Specification Clock and Data Transmission (HICOM 600C)," Siemens Rolm Communications Inc., May 29, 1990.

"HICOM Digital Multiplexed Interface Design Specification," Siemens Rolm Communications Inc., October 5, 1988.

P30252-X9000-A020-01-7659 "U.S. HICOM TMD24 Interface Specification," Siemens Rolm Communications Inc., July 21, 1989.