A Reconfigurable ATM Transmitter/Receiver Implementation

C.N. Gupta
Joseph B. Evans
Gary J. Minden

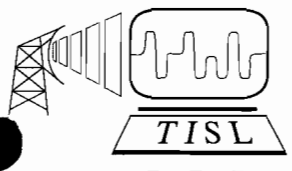
TISL Technical Report TISL-9770-30

Prepared for:

Defense Advanced Research Projects Agency/CSTO

Research on Gigabit Gateways
AARPA Order No. 8634
Issued by EDS/AVS under Contract #F19628-92-C-0080

August 1993



Telecommunications and Information Sciences Laboratory
The University of Kansas Center for Research, Inc.
2291 Irving Hill Drive Lawrence, Kansas 66045

Abstract

An architecture to perform principal receive and transmit functions in a B-ISDN network is presented in this report. This innovative design can be reconfigured to function as a single STS-12c channel or four independent STS-3c channels. The design has been mapped on Field Programmable Gate Arrays (FPGAs). The transmitter chips accept ATM cells from the data source, perform ATM cell scrambling and map the ATM cells into a SONET frame. For the 4xSTS-3c case, the transmitter also byte interleaves the four channels. The receiver performs ATM cell delineation, discards idle cells and descrambles ATM cells. For the 4xSTS-3c case, it de-interleaves the four channels before storing the words into an SRAM. The potential problems in implementing ATM layer functions for higher rate STS signals are analyzed.

Contents

Chapter 1

Introduction

The architecture of the international telecommunications infrastructure is undergoing a change from dedicated transmission facilities toward high-bandwidth virtual channel facilities that are reconfigured. This change is brought about by the introduction of Broadband ISDN (B-ISDN). The emerging B-ISDN networks enable a variety of services, such as video teleconferencing, multimedia communication systems and still-image transmission systems. These new applications have large bandwidth requirements and delay sensitivities that cannot be supported by the existing networks. The emerging demand for multimedia and broadband services and the desire to integrate these future services on a single type of user network interface have prompted a focused effort to define B-ISDN standards. The standards bodies recommend a standardized transport, multiplexing and switching technology called Asynchronous Transfer Mode (ATM) as the basis for B-ISDN. ATM uses small cells, allowing efficient multiplexing of continuous and bursty traffic. ATM provides the flexibility for providing bandwidth on demand. The potential of ATM based networks has prompted all the major public carriers in Japan, Europe and United States to move toward ATM cell based networks [9][12].

Traditionally, computer networks can be divided into Local Area Networks

(LAN) and Wide Area Networks (WAN) . LANs are used to interconnect computers a few miles apart. WANs interconnect computers across wide geographical spans. The use of similar technology in LAN and WAN environments based on B-ISDN provides the opportunity for geographically distributed high performance networks. Fiber has emerged as the transmission technology, since it can provide the required bandwidth. The transmission of digital information over optical fiber is covered in the standards for Synchronous Optical NETwork (SONET) . The standard defines a new hierarchy, that is, a new set of common rates and formats that capitalize on the capabilities of an optical fiber transmission network. The possibility of new services has lead both exchange carriers and computer network providers to embrace technologies such as SONET and particularly ATM for both local and wide area networks. The new formats are more robust compared to the existing protocols and as a result are more complex to implement. For instance, processing SONET STS-12c requires a circuit in which at least a small portion operates at 622 Mb/s. Most parts of the circuit can operate at a much lesser frequency by operating on data in parallel. But, there are limitations on parallelizing the data.

Gateways, which interface LANs with WANs, play a key role in the realization of high performance distributed networks. The work presented here is a part of a gateway that interfaces an AN2 LAN with the MAGIC backbone network. The AN2 is a LAN based on ATM technology, developed by Digital's Systems Research Center. The Multi-Applications and Gigabit Internetwork Consortium, MAGIC is one of the six gigabit wide area testbeds being created for research in high speed networks. MAGIC is a group of industrial, academic, and government organizations participating in gigabit network research. The MAGIC backbone will span approximately 1000 kilometers and will operate at 2.4 Gb/s. The gateway provides a 622 Mb/s connection between the LAN and WAN environments, and supports either a single STS-12c or four STS-3c tributaries. The AN2 LAN

and the gateway are described in Chapter 2. The transmission rate for STS-12c is 622 Mb/s and is 155 Mb/s for each STS-3c stream.

1.1 Organization of the report

This work presents an architecture developed to perform the ATM functions of the B-ISDN protocol reference model in the gateway card. Two circuits are presented, one for a single STS-12c channel and another for four STS-3c channels.

The designs have been implemented using Xilinx Field Programmable Gate Arrays (FPGAs). Field programmable gate arrays have the advantage that they can be reprogrammed easily without removing them from the card. Both of the designs use the same hardware resources and hence the network can be reconfigured. The next chapter gives some background on the B-ISDN protocol reference model. The physical layer, i.e SONET, and the ATM layer of the model are described. Some background material on the AN2 ATM switch and the gateway card are then described. The subsequent chapter describes the STS-12c and the 4 x STS-3c transmitter architectures. Following that, the receiver architectures are described. An analysis of the scalability of ATM and SONET is performed in Chapter 5. The purpose of this analysis is to derive the implementation requirements for higher rate STS signals. Finally, the conclusions are presented.

Chapter 2

Background

2.1 B-ISDN Protocol Reference model

The B-ISDN protocol reference model is analogous to the OSI model of the International Standards Organization (ISO) [14]. The B-ISDN model, shown in Figure 2-1, has been widely used to model a wide variety of communication systems [11].

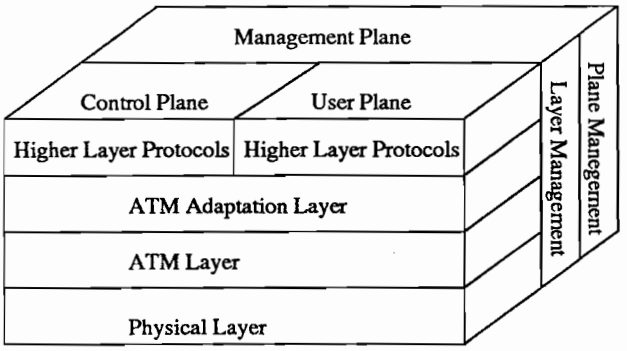


Figure 2-1: B-ISDN Protocol Reference Model

The model uses the concept of separated planes for the segregation of user, control, and management functions [6]. The user plane, with its layered structure, provides for user information flow along with associated controls. The control plane deals with the signaling functions necessary for call and connection setup, supervision and release. The management plane provides plane management and

layer management functions. The user and the control planes have layered structures. The physical layer is based on SONET transmission standards, which is described below. The ATM layer and the AAL layer roughly correspond to layer 2 of the OSI model. The role of the ATM layer is to transport the ATM cells and is common to all services. The ATM layer performs multiplexing and demultiplexing of cells of different connections, translation of VCI and VPI, passing of cell information to and from the ATM Adaptation Layer (AAL), and flow control and policing functions at the UNI. The function of the AAL is to adapt the various applications to the ATM layer.

2.2 A review of SONET

The basic building block of the SONET format is the Synchronous Transport Signal (STS-1) [22]. The STS-1 frame consists of an overhead portion and a payload portion. The overhead is broken into three parts: path, line and section overhead, as shown in Figure 2-2.

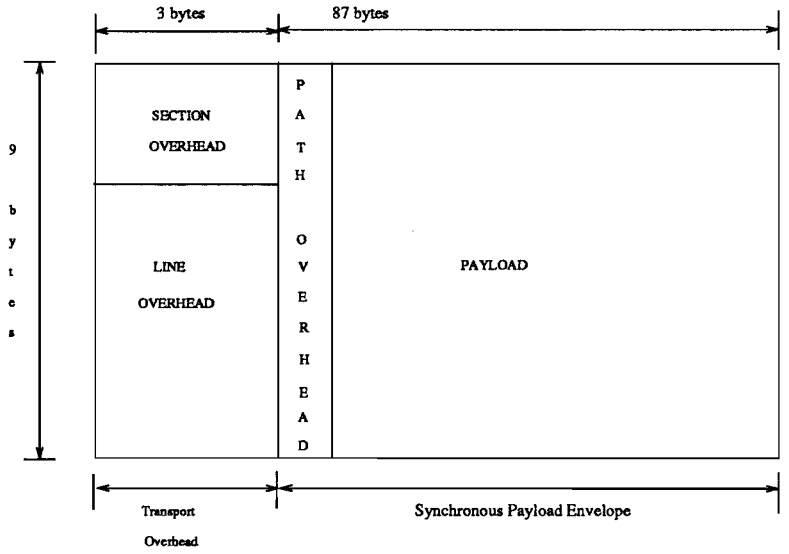


Figure 2-2: SONET STS-1 Frame Format

The Synchronous Payload Envelope (SPE) is the area reserved for transporting the *payload*, and contains 783 bytes per frame. Nine bytes in the SPE are overhead information that the system uses to verify the integrity of the payload and to supply end-to-end signaling. These nine bytes of overhead are referred to as ***Path overhead*** and they stay with the payload until it reaches its final destination. The remainder of the STS-1 frame consists of twenty-seven bytes, used for inter-network signaling and error checking. Nine bytes of the twenty-seven are used for the section portion of the network, that is, between optical repeaters. The remaining eighteen bytes are used for the line portion of the network, that is, between terminals. The first three bytes of every frame contain the frame word (two bytes) and the channel identification byte. The entire STS-1 frame consists of 810 bytes. It repeats at 125 μs intervals, giving a nominal bit rate of 51.84 Mb/s.

Synchronous Transport Signals can be multiplexed into higher level signals denoted by the terminology STS-N, where N denotes the number of STS-1's that have been byte multiplexed together. The equivalent optical signal of STS-N is given the designation OC-N. Higher rate signals can be achieved by concatenation of individual STS-1 frames in a single STS-Nc frame.

SONET networks use a pointer to do frequency and phase aligning of the payload. These pointers are similar to the software pointers. The SONET pointer contains the location of the beginning of the data payload. The payload is allowed to move forward or backward within the frame. Transmission of the STS frames is done row by row, from left to right.

A SONET STS-3c is a concatenated STS-3 and carries a single stream. Figure 2-3 shows the SONET STS-3c frame. It is a 2430 byte pattern that repeats every 125μs. Transmission is row by row, from left to right. Its two main components are the 81 byte Transport Overhead (TOH) and the 2349-byte Synchronous Payload Envelope (SPE). STS tributary multiplexing is a process in which the transport

overhead of the tributary signals are frame aligned, and in which the tributary signals are sequentially byte-interleaved. This results in the the appearance of the byte triplets A1,A2,C1, etc. in the STS-3c frame, even though the STS-3c is not itself created by tributary multiplexing. Bytes A1 and A2 form a framing pattern which identifies the beginning of each frame. Some of the overhead bytes in an STS-3c, such as the path overhead byte, are defined only for the first STS-1 position in the concatenated structure.

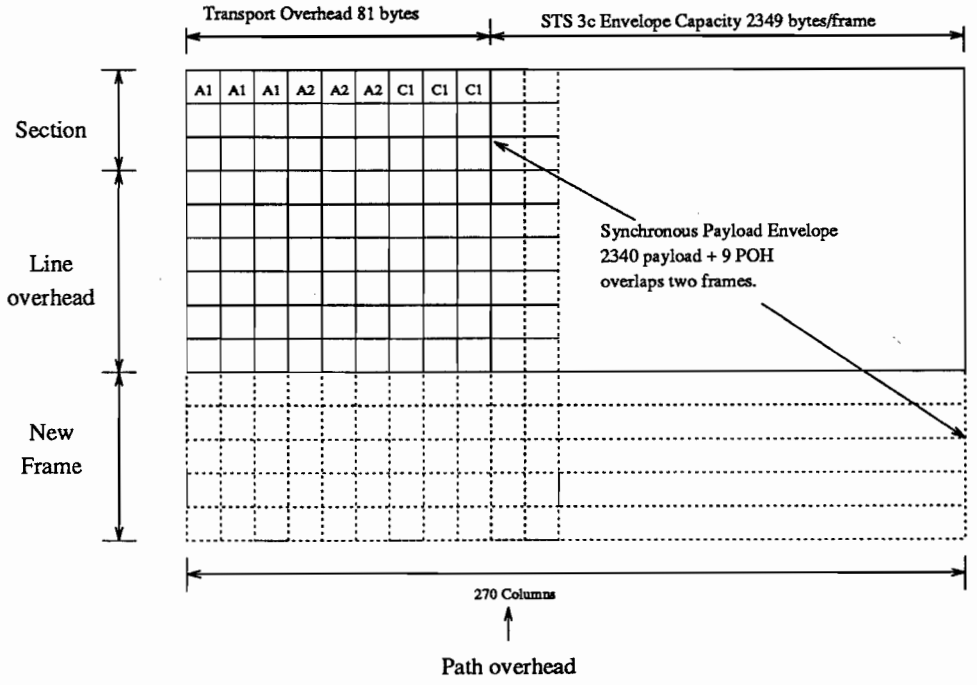


Figure 2-3: STS-3c Frame Structure

An STS-3c stream is at thrice the basic SONET rate, that is at 155.52 Mb/s. STS-12c is a single stream at 622 Mb/s.

2.2.1 An overview of ATM

ATM is the target technology for the implementation of the services offered by Broadband ISDN [3]. ATM cells are fixed-size data units consisting of a header (5 bytes) and body (48 bytes), as shown in Figure 2-4. The header contains a *Virtual*

Circuit Identifier (VCI) and a *Virtual Path Identifier* (VPI) which are used for routing. The PTI field contains the *Payload Type Indicator.* The PTI is used by AALs. The CLP field indicates the *Cell Loss Priority.* The CLP fields are used by the network to decide which cells can be dropped when congestion occurs. A cell with a lower priority is the first candidate for dropping.

The universal transport processing (multiplexing and routing) for B-ISDN is performed by ATM, which enables integrated transport of multirate traffic. ATM cells form the payload portion of the SONET frame in the B-ISDN framework.

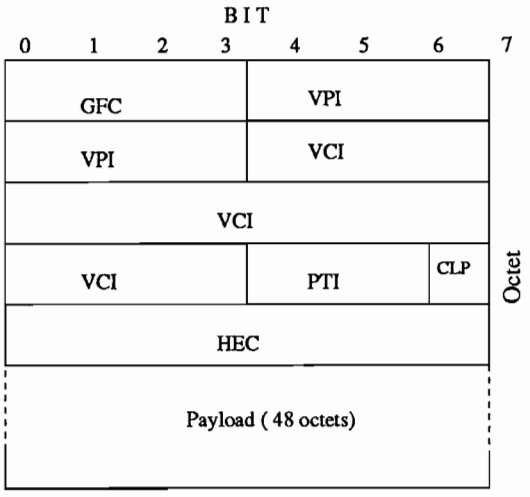


Figure 2-4: ATM Cell Format

2.3 ATM Delineation

If ATM cells form the payload of the SONET frame, the receiver must synchronize with the incoming byte stream. It will then be able to separate the ATM cells and store them for further processing. This is the primary purpose of the ATM receiver, described in this report. Cell delineation is achieved by using the HEC (Header Error Check) field in the ATM cell header to determine if synchronization has been attained [1]. Figure 2-5 shows the state diagram for receiver cell synchronization.

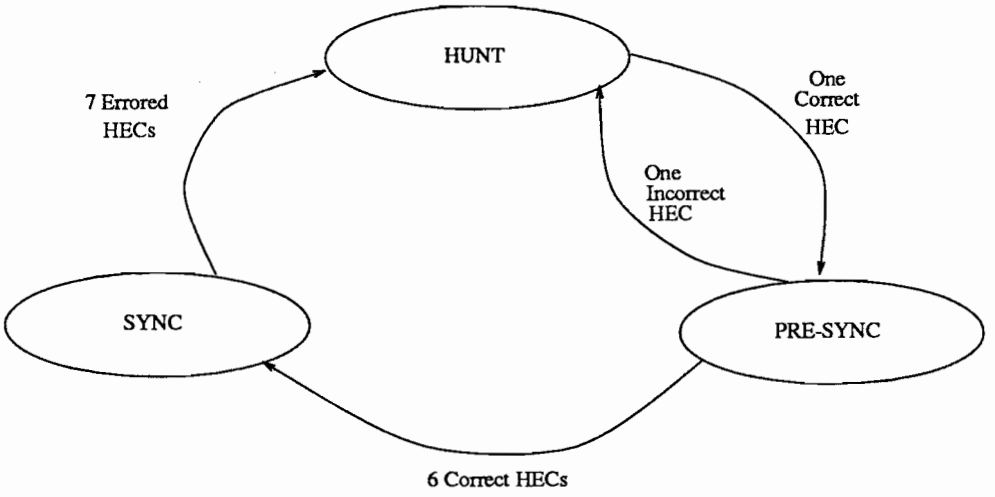


Figure 2-5: ATM Cell Delineation State Diagram

Initially, the delineation unit checks each input stream, byte by byte to see if the HEC of the first four bytes of the assumed header field matches the fifth byte. If it does, the process passes to pre-sync. In the pre-sync phase, the unit checks for valid headers after every 53 bytes. If HEC matches six consecutive times, the unit assumes that it is synchronized with the received stream. The delineation unit constantly checks to see if the input stream is in sync. If it encounters seven consequent cells for which the HECs do not match, it re-initializes itself and begins to hunt for synchronization once again.

2.4 The AN2 ATM Switch

The AN2 LAN is a switch based local area network [2]. Hosts are connected to switches by one or more 155 Mb/s full duplex links, and switches are interconnected by either 155 Mb/s or 622 Mb/s links. Switches are 16 by 16 port crossbars. The AN2 is a virtual circuit based system. The AN2 switches guarantee ordered delivery of cells within a virtual circuit. There is no head of line blocking in the switch. The switches implement a hop-by-hop based flow control scheme. The following two types of line cards plug into switch crossbar:

1. host line cards have four full duplex links operating at 155 Mb/s, and
2. the AN2/SONET Gateway operating at 622 Mb/s using the SONET transmission protocol.

The input side of the line card receives cells over the link, routes those cells, queues them, and sends them through the crossbar. The output side receives cells from the crossbar, buffers them, transmits the cells on the link and manages the flow control mechanism. Queues will never overflow because of the flow control mechanism.

Cells are synchronously switched through the crossbar with a slot period of 520 ns. During arbitration, lasting one slot period, each line card requests access to each output line card for which it has traffic. The arbitration mechanism will result in either no connection or a single connection to an output line card. Cells from the crossbar are immediately transmitted on the output link.

The AN2 implements a strict, window flow control mechanism on a link by link and virtual circuit by virtual circuit basis. During call setup, buffers are allocated on the input side of each line card along the route for the virtual circuit. A line card will not transmit a cell on a link unless it is sure there is a buffer at the receiving end to store that cell. The line card outputs maintain an account balance of the number of buffers available at the receiver for each virtual circuit, which is decremented as each cell is transmitted. When the account balance reaches zero, the output notifies the input line card, through the crossbar, to stop the virtual circuit.

The output side of a line card piggybacks the virtual circuit identifier of a forwarded cell on the cell stream going back to the far end. This serves as an acknowledgement that will be used by the output side of the far end line card to increment its account balance for that virtual circuit.

Each line card has a microprocessor, called the LCP, to monitor, control, and manage the line card. The LCP is involved in call setup, call tear down, route

finding, resource allocation, periodic bandwidth allocation, monitoring the line card and performance measurement.

2.5 The Gateway Card

The AN2/SONET Gateway is a hardware device that connects the Digital Equipment Corporation AN2 Local ATM network to the SONET based B-ISDN ATM network [2]. The purpose of the gateway is to provide a means for data to move between the AN2 and the SONET based wide area network.

The AN2/SONET Gateway is a single card that plugs into an AN2 switch port. The AN2/SONET Gateway supports the following features:

- operation at the SONET STS-12/STS-12c(622.08 Mb/s) capacity on the wide area side via fiber optic connection
- operation within the DEC AN2 Local ATM switch by connecting to the AN2 switch backplane
- experimental techniques for dynamic bandwidth allocation
- experimental techniques for interoperability between connection-oriented and connection less protocols
- experimental signaling protocols for call setup and call parameter negotiations
- measurement of network performance

2.5.1 Transmit Section

The transmit section, connects the AN2 switch crossbar to the transmit SONET Network Termination Equipment (NTE) of the WAN provider. The transmit section will normally use clocks derived from the receive section, but will also have a crystal controlled local clock oscillator. The transmitter receives ATM cells from the AN2 crossbar. The cells are buffered and merged with the SONET overhead information stream. The combined stream forms a SONET frame. The transmit section will maintain status information including the number of buffers available at the remote for each possible virtual circuit. The transmit section supports a single STS-12c stream or four STS-3c streams multiplexed into a single STS-12. A block diagram of this section is given in Figure 2-6.

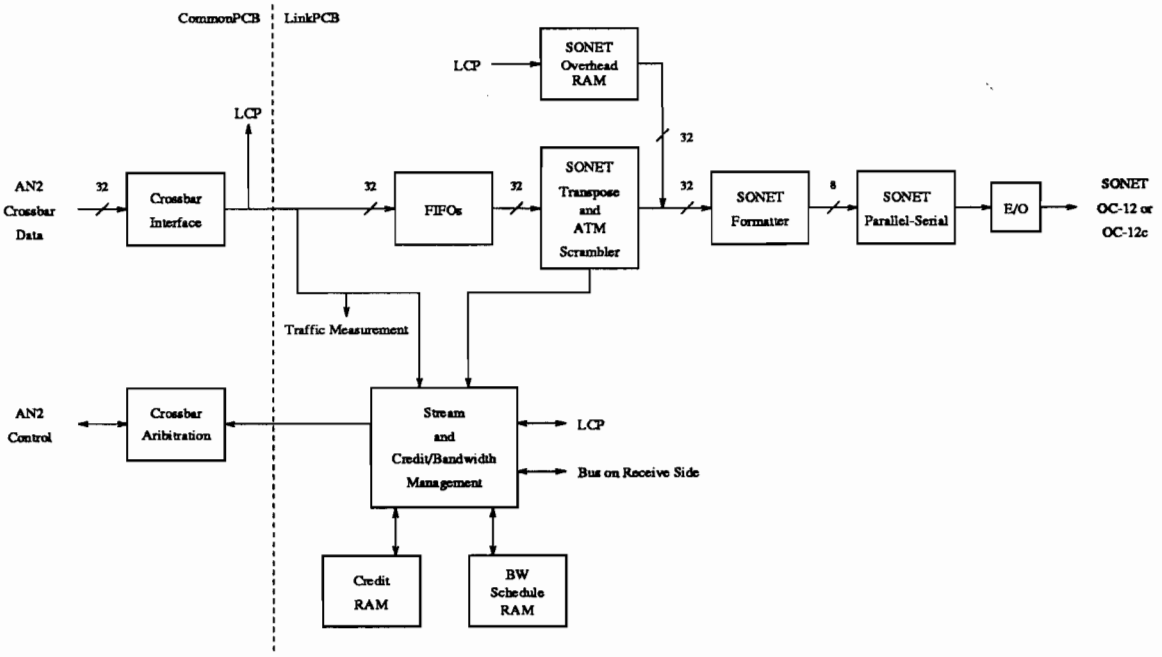


Figure 2-6: Block diagram of the transmit section of AN2/SONET Gateway Card.

ATM cells are written into the transmit FIFO by the AN2 switch. Cells are received at the FIFO in thirteen clock cycles, 32 bits per cycle. The transmit section only accepts cells destined for the gateway.

A complete cell is written into the FIFO on successful arbitration, if there are sufficient buffer resources in the FIFO.

The FIFO is used to temporarily buffer cells arriving from the crossbar. It also serves as a rate adaptation unit. The AN2 operates at a clock rate of 40 ns (25 MHz) per 32 bit word and a cell rate of 520 ns, since 13 words per cell are forwarded across the AN2 crossbar (the HEC byte is not passed through the crossbar). In contrast, the SONET transmission clock is 12.86 ns per byte or 681.6 ns per cell. Cells are written into the FIFO at AN2 rate and read out at the SONET rate. The switch arbitration mechanism insures the FIFOs do not overflow.

The SONET Transpose and ATM Scramble unit operates in one of two modes: a single STS-12c ATM stream, or four STS-3c ATM streams multiplexed into an STS-12. The mode is selected at system configuration time. The functionality and design of this unit is described in the next chapter, *Transmitter Design.*

The SONET overhead RAM is a fast, dual-ported SRAM containing the SONET overhead bytes. The LCP loads the SONET overhead RAM with the proper section, line and path overhead bytes. Sixteen overhead buffers are provided so that the contents can be altered while the system is in operation. The SONET overhead is multiplexed with the byte interleaved ATM cell streams via a tristate bus to generate the input stream to the SONET Formatter.

The SONET Formatter takes payload and overhead data from the preceding stages and performs the combinational operations necessary to fill the parity bytes of the path, section and line overhead, and scrambles the SONET signal according to the standard polynomial [22]. The parallel-serial unit converts the byte-wide parallel stream to a serial stream, which is then fed into the Electrical-to-Optical interface.

The transmit section is implemented using a combination of Xilinx FPGAs, a commercially available SONET parallel to serial conversion integrated circuit,

and commercially available memory chips.

2.5.2 Receive Section

The receive section, shown in Figure 2-7, connects to the received signal from the wide area SONET NTE. Timing information (bit, byte and frame) is extracted from the received SONET signal, and is used in both the receive and transmit sections. The receive section extracts the SONET overhead and the SONET payload from the incoming stream. The SONET payload is processed as ATM cells in accordance with evolving SONET/ATM standards. The ATM header is checked for errors prior to further processing. ATM cells are buffered on a virtual circuit basis. The destination of the received ATM cells is determined by a routing table. The receive section supports a single STS-12c stream or four STS-3c streams multiplexed into a single STS-12.

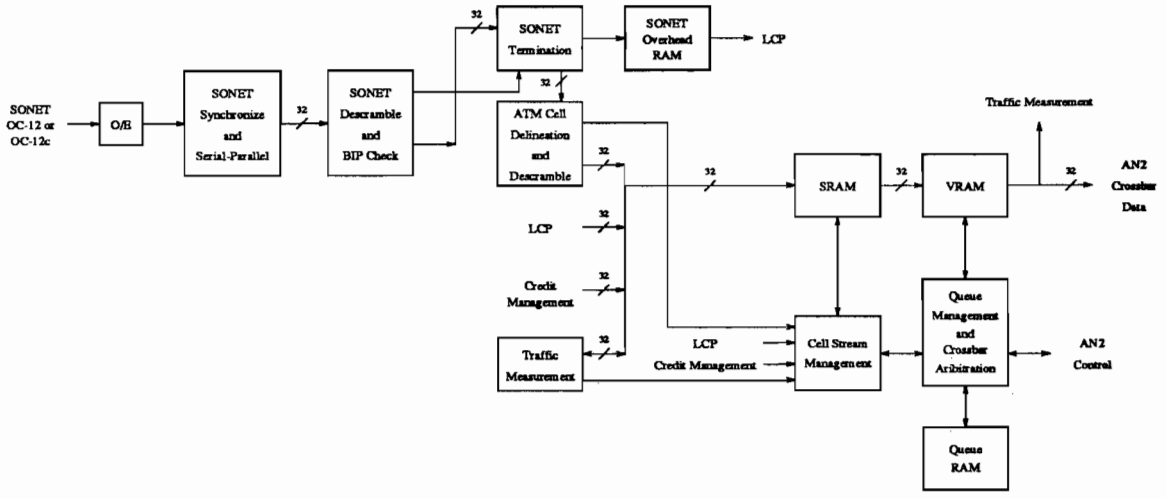


Figure 2-7: Block diagram of the receiver part of AN2/SONET Gateway Card.

The Optical-to-Electrical converter receives and detects the optical serial bit stream and outputs an electrical serial bit stream. A bit clock is recovered from the received signal. The SONET Synchronization and Serial-Parallel unit searches for and synchronizes itself with the SONET frame synchronization pattern. The

byte clock generated by this unit is used throughout the gateway system. The SONET Synchronization and Serial-Parallel unit converts the serial bit stream to a 32-bit wide parallel stream for further processing.

The SONET Descramble and BIP check unit applies the standard descrambling function to the received signal [22]. This unit also checks the SONET section and line bit interleaved parity (BIP) bytes.

The ATM Cell Delineation and Descramble unit searches for valid ATM cell headers in the received payload byte stream, using the standard cell synchronization method. The design of this part is described in the chapter *Receiver Design.*

After the cells are delineated and descrambled, partial cells are written to the SRAM unit for buffering and routing. The SRAM unit also provides the rate adaptation between the SONET and AN2 crossbar clock rates. The SRAM unit performs buffering of the ATM cell streams prior to buffering on a per VCI basis in the VRAM unit. The SRAM unit is managed by the Cell Stream Management unit, which schedules memory accesses for the cell streams and provides indication when cells are ready to be buffered in the VRAM. The Cell stream management unit also controls VC extraction from the received cells.

The VRAM buffer is used to provide sufficient buffering for ATM cells, so that congestion in the AN2 network will not cause cell loss due to buffer overflow. The cells are buffered on a per VCI basis, so that flow control on one VC will not effect other VCs.

The receive section is implemented using a combination of Xilinx FPGAs, a commercially available SONET synchronizer integrated circuit, a three port video dynamic access memories for the VRAM Buffer, and commercially available memory chips.

2.5.3 The Line Card Processor

The Line Card Processor (LCP) manages the resources of the receive section, transmit section, and communications paths [2]. It is responsible for setting up circuits, releasing circuits, monitoring circuits, allocating bandwidth to circuits, and other network management operations both within the AN2 and with the WAN. The LCP communicates with other switch processors via ATM cells. The LCP is composed of a general purpose RISC processor and support chips.

2.6 Xilinx FPGAs

The Xilinx field programmable gate array is a regular array of configurable logic blocks (CLBs) with programmable routing resources. The designs presented here use two different types of Xilinx chips, the Xilinx XC3190, which has 16 columns and 20 rows of CLBs and the Xilinx XC3195, which has 22 columns and 22 rows of CLBs. These chips may be programmed in place by downloading the appropriate program to the device.

Chapter 3

Transmitter Architecture

The transmitter chips form SONET frames from ATM cells and SONET overhead. The transmitter is implemented using one Xilinx XC3195 and one Xilinx XC3190 chip. A block diagram of the transmitter is shown in Figure 3-1.

The first transmitter chip reads its input from the AN2 FIFOs. The input to the second transmitter chip is the data that has been partially processed by the first transmitter. The output of the second transmitter chip is formed of ATM cells arranged contiguously in the payload portion of the SONET frame. A complete SONET frame is formed at the output by multiplexing the payload portion of the SONET frame with the SONET overhead from the overhead SRAMs as shown in Figure 3-1.

The first transmitter chip generates control signals for reading the FIFO. It gets signals from the FIFO indicating whether there are no words in it. Control signals from the second transmitter chip to the first transmitter chip indicate the presence of overhead in the output stream. The signal GenerateIdleCell_H indicates that the chip should generate idle cells and Enable_ATMScramble_H means that the chip should apply the ATM payload self synchronous scrambler to each cell. Enable_ATMScramble_H is to be used for diagnostic purposes. The first chip uses two clocks, Xmit_ByteClk_H, a 12.86 ns clock and Xmit_WordClkP0_H,

a 51.44 ns clock.

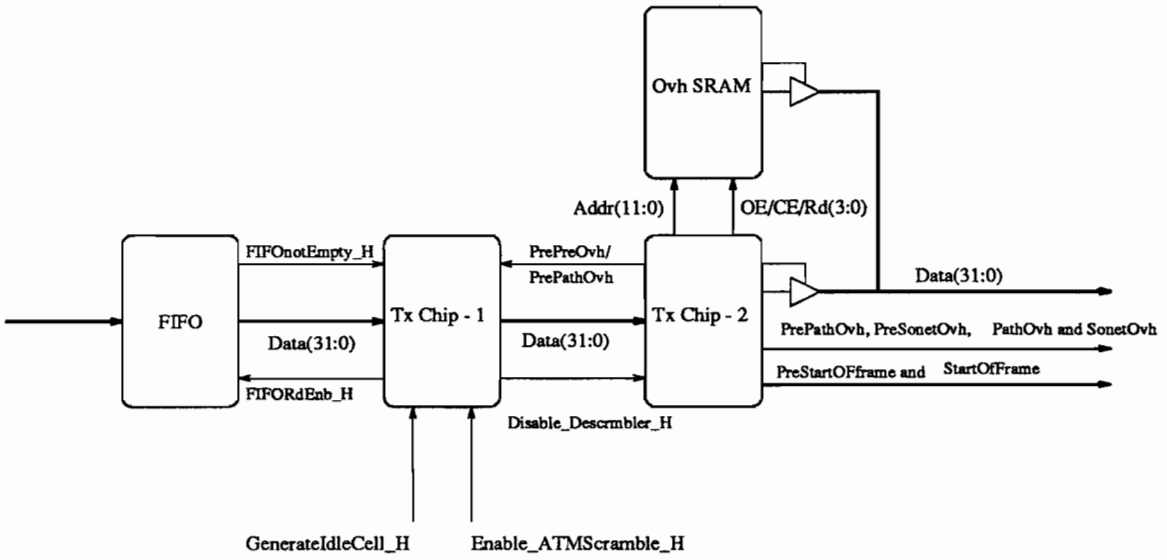


Figure 3-1: System block diagram of the Transmitter.

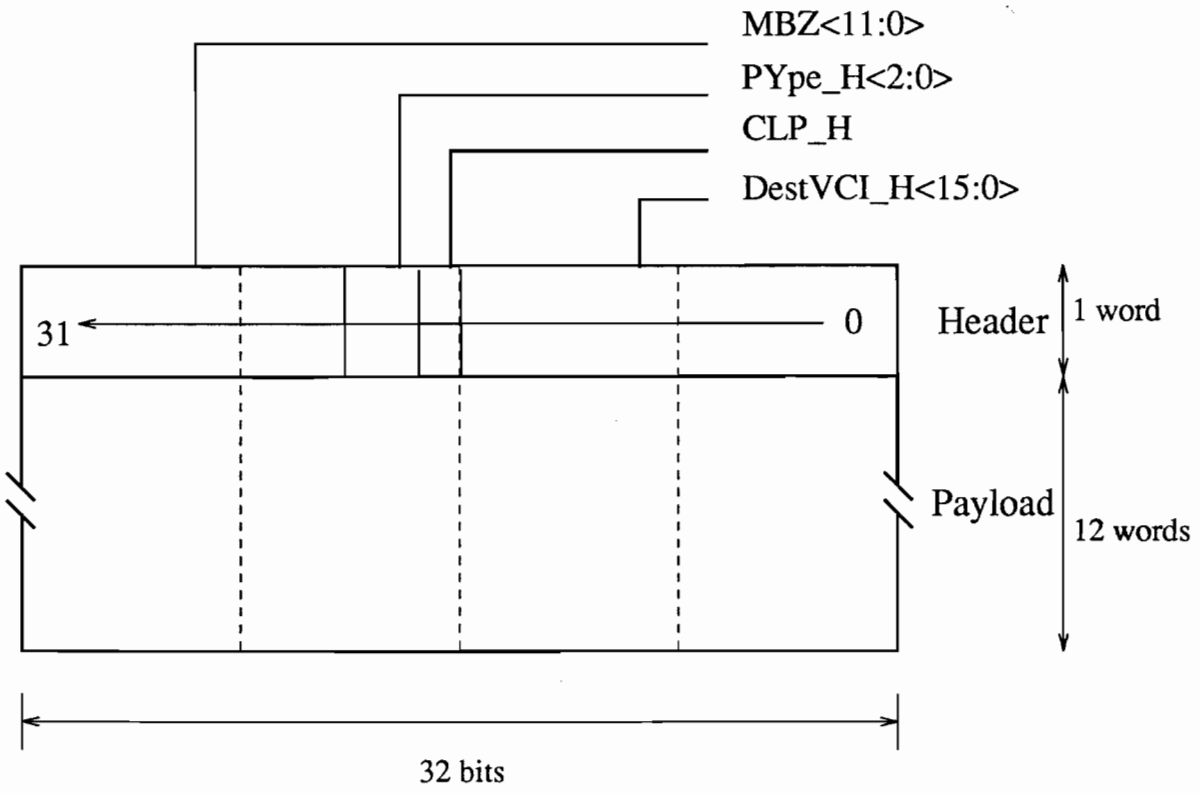


Figure 3-2: Data arrangement in the Transmit FIFOs.

The overhead is read out of the overhead SRAMs by enabling them at the appropriate time. The second transmitter chip also generates signals that indicate the start of frame and SONET overhead in the output data stream. The address for the overhead SRAMs are generated by this chip. This chip uses the Xmit_WordClkP0_H clock, the 51.44ns clock.

Behavioral models, simulation results and FPGA layouts of the transmitter chips are given in Appendix A for the STS-12c mode and in Appendix B for the 4 x STS-3c mode. The receiver and the transmitter were connected together and simulated. These results are given in Appendix C for the STS-12c mode and in Appendix D for the 4 x STS-3c mode.

3.1 Functionality specifications

The functions of the transmitter are summerized below:

- Byte-multiplex ATM cells from a single STS-12c or four STS-3c streams into a single word stream for encapsulation in a SONET frame.
- Insert idle cells into the output data stream when there are no cells available for transmission.
- Calculate and insert the HEC byte into each cell.
- Apply the ATM self-synchronous scrambler operation to each ATM cell payload.
- Generate addresses for the SONET transmit overhead SRAMs.
- Generate control signals to control reading and writing the transmit FIFOs.

3.2 Architecture for the STS-12c mode

3.2.1 Datapath of Transmitter Chip1

This chip reads the 32 bit input data from the FIFO. Four 32 bit FIFOs are present on the board. Each ATM cell is arranged in thirteen words within the FIFO, as shown in Figure 3-2. The bit arrangement of the four byte ATM header in the FIFO is also shown in this figure. Cells are read out of the four FIFOs in a round-robin fashion.

The input is read using a 51.44 ns clock, that is generated within the chip using the Xmit_ByteClk_H. The input is not read continuously. Normally, the input is read at 51.44 ns intervals, using the internally generated clock. The 51.44 ns clock is delayed by 12.86 ns when the HEC byte or the path overhead is inserted. Thus, when either the HEC or the path overhead byte is inserted, the input is read after 64.3 ns. This clock is used for various operations within the chip. It will be referred to as *WdClkShft* from now on.

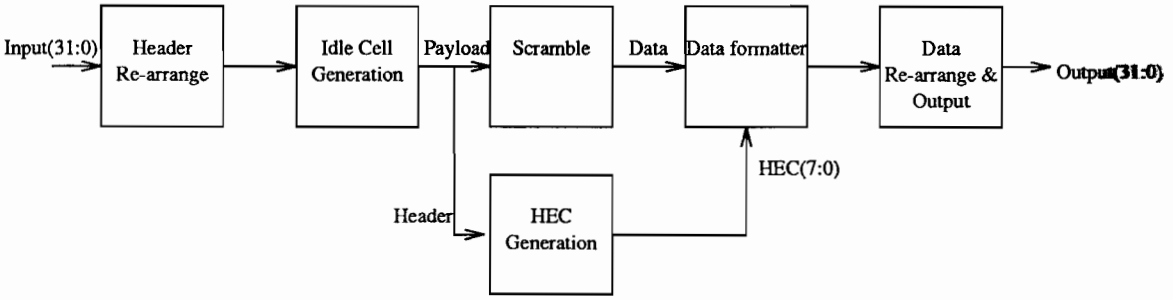


Figure 3-3: Diagram showing the data flow for Chip1 of STS-12c transmitter.

A functional block diagram of the first transmitter chip which shows the data flow through the first transmitter chip is given in Figure 3-3. The data is read from the FIFO and latched at the input IOBs. Then, the header is re-arranged to conform to the first four bytes of a standard ATM cell. Next, an idle cell header is generated. It may be used if actual data is unavailable for transmission. Next, the HEC is calculated over the first four bytes of the header word.

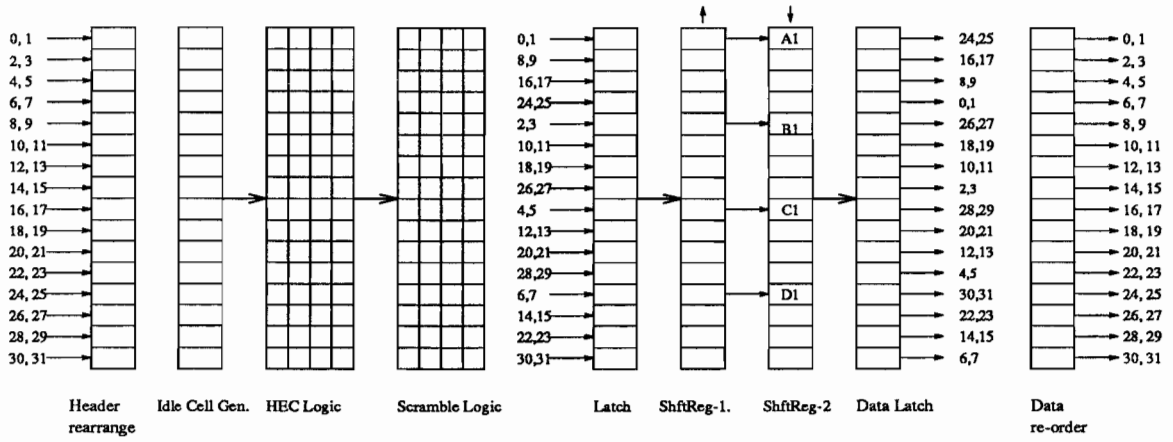


Figure 3-4: Datapath organization for STS-12c transmitter - Chip1.

The next operation is the ATM self synchronous scrambling. The payload portion of the ATM cell is scrambled before transmission. The data is then re-ordered and latched. Finally, the data is formatted to insert the header into the ATM cell and arrange the ATM cells contiguously.

The Xilinx floor plan for the STS-12c mode is as shown in Figure 3-4. The first column in the figure is used for *Header re-ordering.* The order of the bits in the input header word is shown in Figure 3-2. The bits are re-arranged to confirm to the standard ATM cell header [1]. Thus, this column performs a multiplexing operation. When the input data is the header word, it re-arranges the word to form a standard ATM header. When the input is payload data, it just latches the data. The header rearrange column operates using the WdClkShft clock.

The next column is used for *Idle Cell Generation.* Idle cells are required to have the header bit pattern shown in Figure 3-5. This column implements a multiplexer. The output of this column is either the first four bytes of the idle cell header or the input data. The idle cell generation column uses the WdClkShft clock.

The next three columns are used for HEC calculation. HEC is calculated on ATM cell headers including idle cell headers. A signal from the state machine enables HEC calculation when the header is at the input of the HEC calculation

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	a	a	a	0

"a" indicates that the bit is available for the appropriate ATM layer function.

Figure 3-5: Header bit pattern for an idle cell

logic. The HEC calculation is done using the WdClkShft clock.

The polynomial used to calculate the HEC is given below:

$$1 + x + x^2 + x^8 \tag{3.1}$$

After calculating the HEC, the result is added modulo-2 with the following coset polynomial:

$$1 + x^2 + x^4 + x^6 \tag{3.2}$$

Thirty two bit parallel HEC calculating circuit is implemented in these designs. The equations used for this purpose are given below. In these equations, c is the coset polynomial, h is the input 32 bit word on which the HEC is to be calculated and r is the resultant HEC vector. In the input 32 bit header, h_0 is the most significant bit and h_{31} is the least significant bit.

c := "01010101";

$$
\begin{aligned}
r_0 &:= h_0 + h_1 + h_3 + h_8 + h_{10} + h_{12} + \\
&\quad h_{13} + h_{15} + h_{17} + h_{19} + h_{23} + h_{24} + \\
&\quad h_{25} + h_{31}; \\
r_1 &:= h_1 + h_2 + h_3 + h_7 + h_8 + h_9 + h_{10} \\
&\quad + h_{11} + h_{13} + h_{14} + h_{15} + h_{16} + h_{17} \\
&\quad + h_{18} + h_{19} + h_{22} + h_{25} + \\
&\quad h_{30} + h_{31}; \\
r_2 &:= h_2 + h_3 + h_6 + h_7 + h_9 + h_{14} + h_{16} \\
&\quad + h_{18} + h_{19} + h_{21} + h_{23} + h_{25} + h_{29} \\
&\quad + h_{30} + h_{31}; \\
r_3 &:= h_1 + h_2 + h_5 + h_6 + h_8 + h_{13} + h_{15} \\
&\quad + h_{17} + h_{18} + h_{20} + h_{22} + h_{24} + h_{28} \\
&\quad + h_{29} + h_{30}; \\
r_4 &:= h_0 + h_1 + h_4 + h_5 + h_7 + h_{12} \\
&\quad + h_{14} + h_{16} + h_{17} + h_{19} + h_{21} + h_{23} \\
&\quad + h_{27} + h_{28} + h_{29}; \\
r_5 &:= h_0 + h_3 + h_4 + h_6 + h_{11} + h_{13} + h_{15} \\
&\quad + h_{16} + h_{18} + h_{20} + h_{22} + h_{26} + h_{27} \\
&\quad + h_{28}; \\
r_6 &:= h_2 + h_3 + h_5 + h_{10} + h_{12} + h_{14} + \\
&\quad h_{15} + h_{17} + h_{19} + h_{21} + h_{25} + h_{26} + \\
&\quad h_{27}; \\
r_7 &:= h_1 + h_2 + h_4 + h_9 + h_{11} + h_{13} + \\
&\quad h_{14} + h_{16} + h_{18} + h_{20} + h_{24} + h_{25} + \\
&\quad h_{26};
\end{aligned}
$$

$$r := r + c;$$

The ATM self synchronous scrambler is implemented in the next three columns. Only the payload portion of an ATM cell is scrambled. The scrambler uses the WdClkShft clock. Scrambling is performed using the following polynomial [3]:

$$x^{43} + 1 \tag{3.3}$$

A conceptual diagram illustrating the operation of the scrambler is shown in Figure 3-6 .

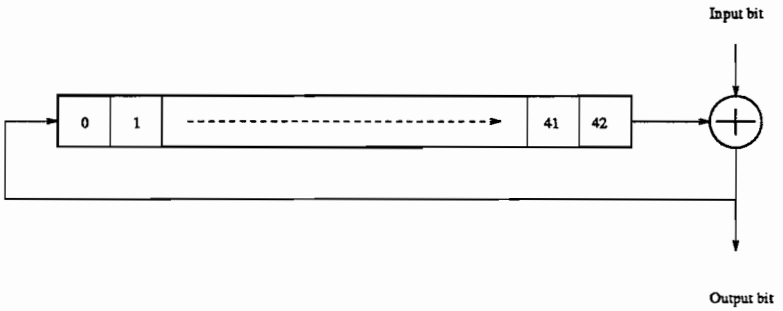


Figure 3-6: Conceptual diagram of the scrambler.

The scrambler used in this design is a 32 bit parallel scrambler. The equations used to implement the 32 bit parallel scrambler are given below.

```
FOR I IN 0 TO 31 LOOP
  Scrambled_Data(I) <= Input_Data(I) XOR Scramble_State(42-I);
END LOOP;

FOR I IN 0 TO 31 LOOP
  Scramble_State(31-I) <= Input_Data(I) XOR Scramble_State(42-I);
END LOOP;

FOR I IN 32 TO 42 LOOP
  Scramble_State(I) <= Scramble_State(I-32);
```

```
END LOOP;
```

The next column is used to latch the data out of the scrambler. The data is re-arranged before latching into this column. The order of the input bits to this column is shown in Figure 3-4. The *Latch* column uses WdClkShft clock.

The input to the *Latch* register is re-arranged as shown in Figure 3-4 to localize the interconnections in *ShiftReg-1* and *ShiftReg-2*. The output of *Latch* is latched and operated upon by *ShiftReg-1* and *ShiftReg-2*. Both of these registers operate using the Xmit_ByteClk_H, which is the 12.86 ns clock.

ShiftReg-1 shifts the data up by two bits, that is one CLB every 12.86 ns. *ShiftReg-2* shifts the data down two bits every 12.86 ns. With reference to Figure 3-4, *ShiftReg-2* gets one byte at CLBs in location A1, B1, C1 and D1 from *ShiftReg-1* after each upshift. Using this design, HEC byte or the path overhead byte can be inserted in the output data stream easily.

The HEC byte is inserted by latching the HEC byte at locations A1, B1, C1 and D1 of Figure 3-4, instead of the data from *ShftReg-1*. When the HEC byte is inserted, the data in *ShftReg-1* is not shifted up. This is done by disabling the clock to the *ShftReg-1* column using the enable clock input in the CLBs. As mentioned earlier, the clock period of WdClkShft is 64.3 ns when the HEC byte is inserted. The clock period of WdClkShft is 51.44 ns normally.

The path overhead byte is externally multiplexed at the output of the second transmitter chip. A byte is left empty in the output stream of this chip to insert the path overhead. To leave an empty byte, *ShftReg-1* is not shifted up for one 12.86 clock period. The clock period of WdClkShft is 64.3 ns when the path overhead byte is inserted.

Before the data is read out, it is re-ordered to the normal data order as shown in Figure 3-4. The output is read out using the Xmit_WordClkP0_H.

All the ATM processing required for the SONET STS-12c architecture is thus

performed on one Xilinx XC-3195 chip.

3.2.2 State machine for STS-12c Chip1

The functions performed by the state machine are listed below.

- Generation of the shifted 51.44 ns clock, WdClkShft, a clock that is shifted by 12.86 ns every time the HEC or path overhead is inserted. This clock is masked when SONET overhead is inserted in the output stream.
- A modulo-13 counter using WdClkShft clock. The states of this state machine are shown in Table 3.1. The counter states are used for the following signals:
  - A signal to disable scrambling the header word.
  - A signal to insert the HEC byte into the data stream.
  - A signal to enable header rearrangement.
- A signal to leave a byte empty to insert the path overhead byte when the word is read out of the second transmitter chip. This signal is generated using the *PrePathOvh* signal that is asserted by the second transmitter chip.
- A signal to disable output when SONET overhead is to be multiplexed in the output data stream. This signal is generated using the *PrePreOvh* signal generated by the second transmitter chip.

Table 3.1 gives the pipeline delay through the chip and the relative timings of the various control signals. Table 3.2 shows the use of the *PrePathOvh* signal to leave a free byte for inserting path overhead. The time ticks in the following tables is not synchronous. Thus, T0 does not refer to the same time tick in all the tables.

Time tick	Action
T0	Idle cell header generated and is at the output of the idle cell column.
	Disable scrambler signal.
T1	Header latched out of the scrambler without scrambling.
	HEC calculation logic stage 1 enable.
T2-t0	HEC calculation logic stage 2 enable. Data is at the output of *Latch* column. Parallel loading of *ShftReg-1* register is enabled.
T2-t1	Data latched out of *ShftReg-1* column.
T2-t2	First byte out of the *ShftReg-2* column.
T2-t3	Second byte out of *ShftReg-2* column.
T3-t0	Third byte out of *ShftReg-2* column.
T3-t1	Fourth byte out of *ShftReg-2* column.
	HEC selection is enabled.
	51.44 ns clock shifted for a 12.86 ns.
	Clock to *ShftReg-1* is disabled.
T12	Idle cell generation is enabled if no cells are available for transmission or if external *GenerateIdleCell_H* signal is asserted.

Table 3.1: Pipeline delay through Chip1 illustrating the generation of control signals.

Time tick	Action
T0-t0	PrePathOvh signal latched into first transmitter chip.
T0-t1	PrePathOvh signal width reduced to 12.86 ns.
T0-t2	PrePathOvh signal delayed by 12.86 ns.
T0-t3	PrePathOvh signal used to disable clock to *ShftReg-1.*
T1-t0	1 byte is left empty for path overhead insertion.

Table 3.2: Relative timing of *PrePathOvh* signal to the actual path overhead byte.

Time tick	Action
T0	*PrePathOvh* signal generated.
T1	*PrePathOvh* signal latched out of second chip.
T2	*PrePathOvh* signal latched into first chip.
T3	First byte of the word left empty for path overhead.
T4	Word with the path overhead byte out of *DataLatch* register in Figure 3-4.
T5	Word with path overhead byte out of first transmitter chip.
T6	Word with path overhead byte into second transmitter chip.
	PrePreOvh signal latched out of second transmitter chip.
T7	Read/output enable of overhead SRAM latched out of chip.
	First SONET overhead word read into the output stream.
T8-T15	SONET overhead words read into the output stream.
T16	Word with path overhead read out of the transmitter.
	Only LSB of overhead SRAM enabled to merge the path overhead byte into the output stream.
T17-T269	Payload of SONET framed is read out of the transmitter.

Table 3.3: Second transmitter chip control signals.

3.2.3 Transmitter Chip2 design for the STS-12c mode

The second transmitter chip generates the SONET overhead SRAM addresses. It generates signals to indicate SONET overhead and path overhead in the output data stream. It generates a signal to indicate the beginning of a new SONET frame. These signals are used by the SONET BIP scrambler chip. The signals *PrePreOvh* and *PrePathOvh* are used by the first transpose chip. The output of this chip consists of complete SONET frames in which the ATM cells are arranged contiguously. Table 3.3 shows the SONET overhead counter states and the pipeline delay through the transmitter.

3.3 Design for the 4 x STS-3c mode

3.3.1 Data path design for the first transmitter chip

This chip reads the 32 bit input data from the FIFO. Four 32 bit FIFOs are present on the board. Each ATM cell is arranged in thirteen words within a FIFO, as shown in Figure 3-2. The bit arrangement of the four byte ATM header in the FIFO is also shown in this figure. The FIFOs are written by the AN2 crossbar. Each FIFO consists of data for a single channel. For broadcast or multicast, all the FIFOs will be simultaneously written with the same data by the AN2 crossbar. The chip reads the FIFOs in a round-robin fashion. Thus, during each clock cycle, a word from a single channel is read by the chip.

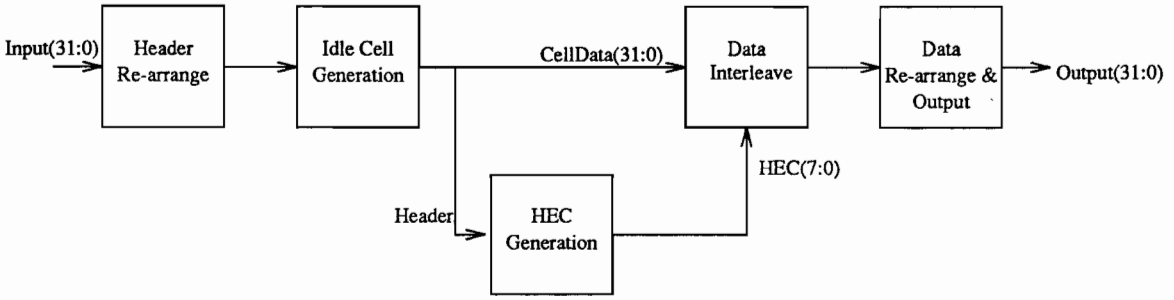


Figure 3-7: Data path of first transmitter chip

A functional block diagram illustrating the data flow through the first transmitter chip is shown in Figure 3-7. Two clocks are used within the chip. A 51.44 ns clock and a 12.86 ns clock. The 51.44 ns clock will be referred to as WdClkShft and the 12.86ns clock will be referred to as ByteClk. The WdClkShft is generated within the chip and will be masked when the SONET overhead or the path overhead is inserted.

The Xilinx floor plan for this chip is as shown in Figure 3-8.

The input data is latched in the input IOBs. Then, the header is re-arranged in the first column. The header re-arrange column is same as the STS-12c mode.

The next column is used for idle cell generation. This column implements a

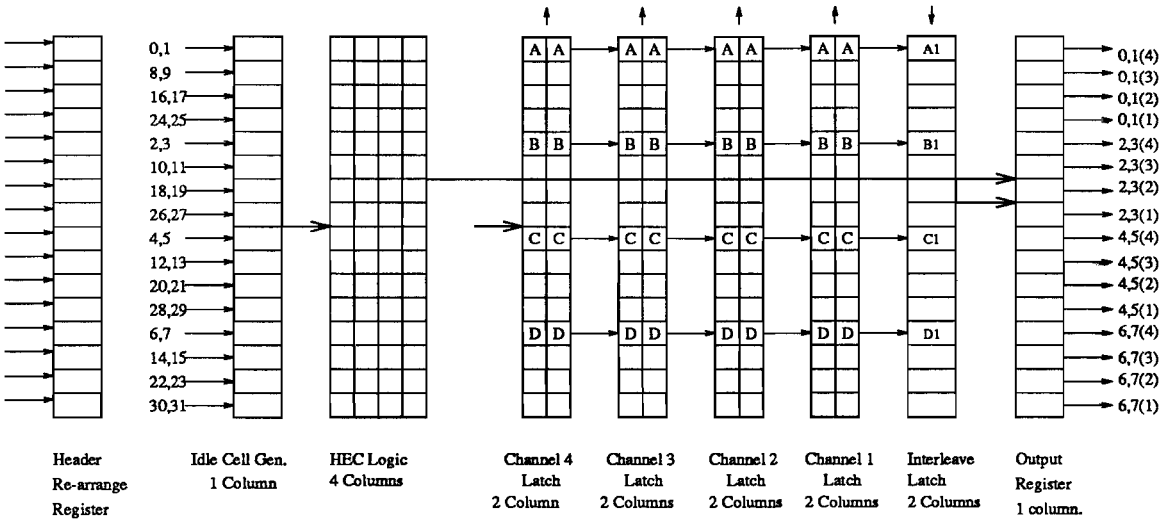


Figure 3-8: Data path of first transmitter chip

multiplexer. The output of this column is either the first four bytes of the idle cell header or the input data. The data out of this column is as shown in Figure 3-8. The idle cell generation column uses the WdClkShft.

The next four columns are used to generate the HEC. The HEC polynomial is the same as the one used for the STS-12c mode. The 32 bit parallel HEC implementation that was used in the STS-12 mode is also used here.

The headers of the four channels are read sequentially. The HECs of all the four channels are calculated one after the other and stored in the next column. The HEC bytes of the four channels are stored in this column in a byte interleaved fashion. Thus, this interleaved word can be directly multiplexed at the output, when required.

The *Channel* registers are used to byte interleave the data. The CLBs A, B, C and D in Figure 3-8 function using the ByteClk. The other CLBs of the *Channel* registers function using the WdClkShft. Two columns are used for each channel. While one column is used to buffer the data, the data in the other column is used to form the interleaved word which is output. Byte interleaving is explained below.

Interleaving is achieved by getting a byte from the *Channel 1* register into locations A1, B1, C1 and D1 of the *Interleave Latch* once every ByteClk. The data in this register is down shifted using the ByteClk. The data is also right shifted from the *Channel 4* register to *Channel 3* to *Channel 2* to the *Channel 1* register using the ByteClk. After four ByteClk periods, which is 51.44 ns, the interleave register contains one complete byte interleaved word.

The *Channel* registers are shifted up using the WdClkShft. By repeating this operation, all the data in one set of the four channel registers are used to form interleaved words at the end of four WdClkShft periods. In the mean time, four new words have been written into the other set of four *Channel* registers.

A new word is written into one of the two *Channel-4* registers at every WdClk-Shft. One set of the *Channel-4*, *Channel-3*, *Channel-2* and *Channel-1* registers are connected in a shift register configuration to read the input data. The input data is right shifted from *Channel-4* to *Channel-3* to *Channel-2* to *Channel-1* register. Thus, after four clock periods, four new words have been written into the channel registers. For the next four WdClkShft periods, these words will be used to form the interleaved words. The other set of four channel registers that were used to form the output words will be used during the next four clock periods to buffer the input.

The output of the interleave register is read in by the output register. This register also multiplexes the HEC word and the data word from the interleave register. When HEC word is to be inserted, the *Sel_Hec* control signal goes high and the HEC is selected instead of the data word from the interleave register. At the same time, the clocks to the interleave register and the channel registers are disabled. Clocks to all the registers using ByteClk are disabled by using the enable clock input in a CLB. The WdClkShft is masked when HEC is inserted.

During SONET overhead or path overhead insertion, all the CLBs within the chip are disabled and the input is also disabled. The ByteClk as well as the WdClkShft are masked to disable the operations within the chip.

3.3.2 State machine for the 4 x STS-3c Chip1

The various signals generated by the state machine and their significance is given below:

- WdClkShft: This clock is the same as the Recv_WordClkP0_H, except that it is masked for one clock cycle when the HEC byte is inserted. It is also masked for the time when the path overhead or the SONET overhead is inserted.

- A six bit counter that counts from 0-51 using WdClkShft. It only counts 52 states because WdClkShft is masked for the state when HEC is inserted. This counter is used to generate the following control signals:

  - Header rearrange control: To re-arrange the header word.

  - Idle cell generation: Idle cells need to be generated when either the *GenerateIdleCells* input signal is high or when there are no cells in the FIFOs. Idle cells will be generated only after the transmission of the present cell.

  - HEC generation controls: The HEC logic is enabled appropriately to latch the headers and generate HECs.

  - Enable clock to the channel registers: CLBs A, B, C and D in Figure 3-8 will be disabled whenever the HEC is inserted. The clock that is used by these CLBs is already masked when path overhead or SONET overhead is to be inserted. This signal also controls the enable to these CLBs when the CLBs are used to buffer the words. These CLBs operate

at the 12.86 ns clock only when they are used to form the interleaved word.

- Parallel load/ right shift control: For the set of channel registers that are used to interleave the data, the CLBs A, B, C and D will get the data from the CLBs directly below them once every four 12.86 ns clock periods. During other times, these CLBs will right shift the data. For the set of channel registers that are used to buffer data, all the CLBs are parallelly loaded every 51.44 ns.
- Select HEC control: This is used to multiplex the data input to *Output* register.

- *PrePreOverhead* and *PrePathOverhead* signals are used to mask the generation of WdClkShft and ByteClk.

The relative timing of the control signals and the pipeline delays for a single channel are illustrated in Table 3.5. The clock period T_x represents a 51.44 ns clock, that is synchronous to Xmit_WordClkP0_H and t_x represents the 12.86 ns clock period.

3.3.3 Design of the second transmitter chip for 4 x STS-3c mode

This chip scrambles the byte interleaved data that it receives from the first chip. An 8 bit self synchronous scrambler is implemented in this design. The scrambler occupies the first seven columns of CLBs in the chip. Equations for the scrambler are given next in VHDL:

Time tick	Action
T0	Generate signal *HeaderCtl* to re-arrange header bits to form Stream-1 header.
	Enable FIFO to read the Stream-1 header word.
T1	Stream-1 header at the input of *Header Rearrange* column.
	HeaderCtl signal at the input of *Header Rearrange* column.
T2	*GenerateIdleCell* signal latched out if there are no cells in stream-1 available for transmission or if the input GenerateIdleCell_H is asserted.
	Header for stream-1 is at the input of idle cell column.
T3	Stream-1 header is at the input of Channel 4 latch.
	Stream-1 header is at the input of stage 1 of HEC calculation logic.
	Clock to first stage of HEC calculation logic is enabled.
T4	Stream-1 header is at the input of Channel 3 latch.
	Stream-1 header is at the input of stage 2 of HEC calculation logic.
	Clock to second stage of HEC calculation logic is enabled.
T5	Stream-1 header is at the input of Channel 2 latch.
	HEC for stream-1 is calculated and is ready to be latched into the HEC interleave register.
	Clock to HEC interleave register enabled.
T6	Stream-1 header is at the input of Channel 1 latch.
T7-t0	Stream-1 header is at the output of Channel 1 latch, i.e at the input of interleave latch.
T7-t1	First header byte latched out of interleave latch.
T7-t2	First byte of stream-2 header latched out of interleave latch.
T7-t3	First byte of stream-3 header latched out of interleave latch.
T8-t0	First interleaved header word latched out of interleave latch.
T8-t1	The interleaved header word is latched out of output register.

Table 3.4: Pipeline delay through Chip1 illustrating the generation of control signals for the 4 x STS-3c mode.

Time tick	Action
T9-t0	The interleaved header word is latched out of Chip1 using Xmit_WordClkP0_H.
T9-t1	Second interleaved header word is latched out of output register.
T10-t0	The second interleaved header word is latched out of Chip1.
T10-t1	Third interleaved header word latched out of output register.
T11-t0	Third interleaved header word latched out of Chip1.
	Generate signal to mask clock to all the logic within the first chip other than the output, which uses Xmit_WordClkP0_H.
T11-t1	Fourth interleaved header word latched out of output register.
T12-t0	Fourth interleaved header word latched out of Chip1.
	Interleaved HEC word latched out of output register.

Table 3.5: Pipeline delay through Chip1 illustrating the generation of control signals for the 4 x STS-3c mode.

```
FOR I IN 0 TO 7 LOOP
  Scrambled_bit(I)           <= Input_bit(I) Xor Scramble_state_bit(42-I);
  Scramble_state_bit(7-I)    <= Input_bit(I) Xor Scrambled_state_bit(42-I);
  Scramble_state_bit(42 - I) <= Scramble_state_bit(34 - I);
  Scramble_state_bit(34 - I) <= Scramble_state_bit(26 - I);
  Scramble_state_bit(26 - I) <= Scramble_state_bit(18 - I);
  Scramble_state_bit(18 - I) <= Scramble_state_bit(10 - I);
END LOOP;
  Scramble_state_bit(10) <= Scramble_state_bit(2);
  Scramble_state_bit(9) <= Scramble_state_bit(1);
  Scramble_state_bit(8) <= Scramble_state_bit(0);
```

Only the ATM cell payload is scrambled. Scrambling is controlled by the *Enable_Scrambler* signal from the first transmitter chip that goes low to disable scrambling. This signal will go low when the header is at the input of the scrambler or when the external control disables scrambling by asserting the external signal *Enable_ATMScramble_H* to low. The organization of the data path for this chip is shown in Figure 3-9.

Chip2 generates the SONET overhead SRAM addresses. It generates signals to indicate SONET overhead and path overhead in the output data stream. It also generates a signal to indicate the beginning of a new SONET frame. These

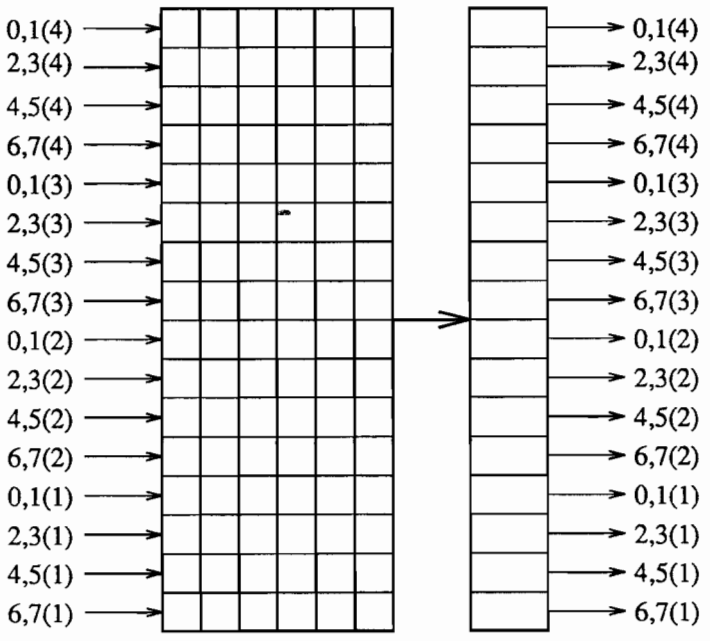


Figure 3-9: Data path of first transmitter chip

Time tick	Action
T0	*PrePathOvh* signal generated.
T1	*PrePathOvh* signal latched out of second chip.
T2	*PrePathOvh* signal latched into first chip.
T3	All the clocks in first chip other than Xmit_WordClkP0_H are disabled.
T4	Path overhead word is latched out of output latch in Figure 3-8.
	PrePreOvh signal latched out of Chip2.
T5	Path overhead word latched out of Chip1.
	PrePreOvh signal latched into Chip1.
T6	Path overhead word ready to be latched out of Chip2 IOBs.
	PrePreOvh signal used to generate disable all functions in Chip1.
T7	Read/output enable of overhead SRAM latched out of Chip2.
	First SONET overhead word read into the output stream.
T8-T15	SONET overhead words read into the output stream.
T16	Path overhead word read into the output stream.
T17-T269	Payload of SONET framed is read out of the transmitter.

Table 3.6: Second transmitter chip control signals.

signals are used by the SONET BIP scrambler chip. The signals *PrePreOvh* and *PrePathOvh* are used by the first transmitter chip. The scrambler is also disabled when SONET overhead is multiplexed in the output data stream. The output of this chip consists of complete SONET frames in which the ATM cells are arranged contiguously. SONET overhead is multiplexed externally by enabling the output of the overhead SRAMs at the appropriate time.

Table 3.6 shows the SONET overhead counter states and the pipeline delay through the transmitter.

3.4 Design for the STS-12c transmitter using 51.44 ns clock

In this section, a way of performing all the transmitter functions using the 51.44 ns clock is presented, keeping the input and output specifications unchanged. In the design that was presented earlier, there are two operations that are performed at 12.86 ns. First, insertion of the HEC byte into the output data stream and next, insertion of the path overhead byte into the output data stream. Due to the odd number of bytes in the ATM cell, the position of the HEC byte in the output data stream could be in one of the four positions shown in Figure 3-10.

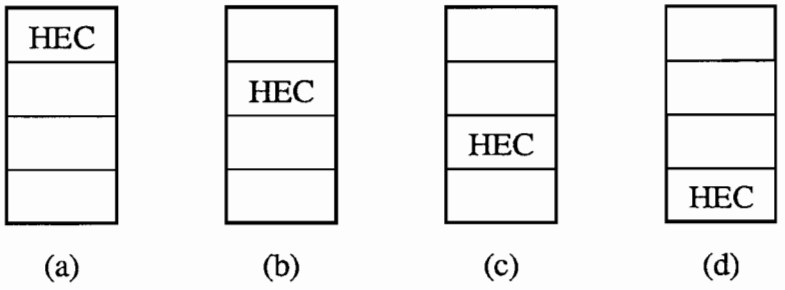


Figure 3-10: Position of the HEC byte in the output data stream.

To insert the HEC byte into the output data stream, the input data has to be shifted down by one, two or three bytes depending upon where the HEC byte is to be inserted. The input bytes that are unused for forming the present output word will be used in the next output word. The path overhead is always inserted in bits (7:0) of the first word following the SONET overhead.

To perform these functions, four sets of barrel shift registers are required. The Xilinx XC3195 floor plan is as shown in Figure 3-11. The first eight columns are used for header rearrange, idle cell generation, HEC logic, and ATM self synchronous scrambling. The data is then re-ordered as shown in Figure 3-11 and latched into the latch column. The re-ordering will reduce the net routing distances. The re-ordered data is latched into the first register of each barrel shift

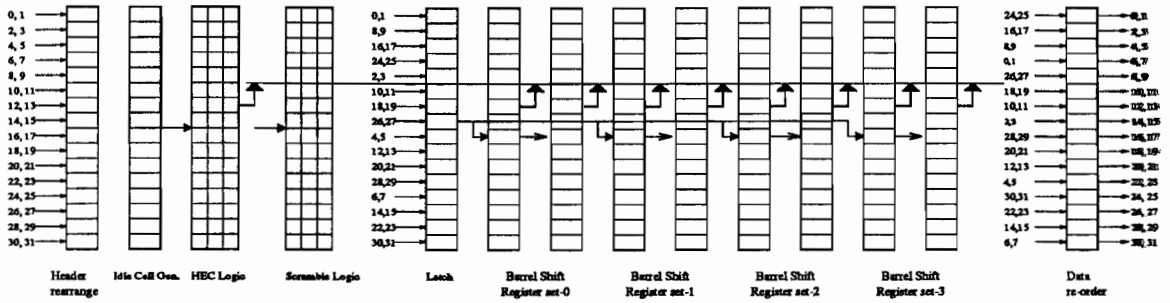


Figure 3-11: Transmitter design for STS-12c mode using 51.44 ns clock.

register set. Figure 3-12(a) shows the byte ordering within the first barrel shift register set. Figures 3-12(b), 3-12(c) and 3-12(c) show the byte ordering within barrel shift register sets 1, 2 and 3 respectively. In this figure, bytes 0(0), 1(0), 2(0) and 3(0) correspond to the first register of the register set. Bytes 0(1), 1(1), 2(1) and 3(1) correspond to the second register of each register set. The bytes 0(1), 1(1), 2(1) and 3(1) are delayed by one 51.44 ns clock with respect to bytes 0(0), 1(0), 2(0) and 3(0). Due the order of the bits in the input bus, the maximum vertical net length to realize the byte order shown in Figure 4-24 is 3 CLBs.

The input data is latched into the first register of each barrel shift register sets during each clock period. The data in the first register of each set is latched into the second register at the same time.

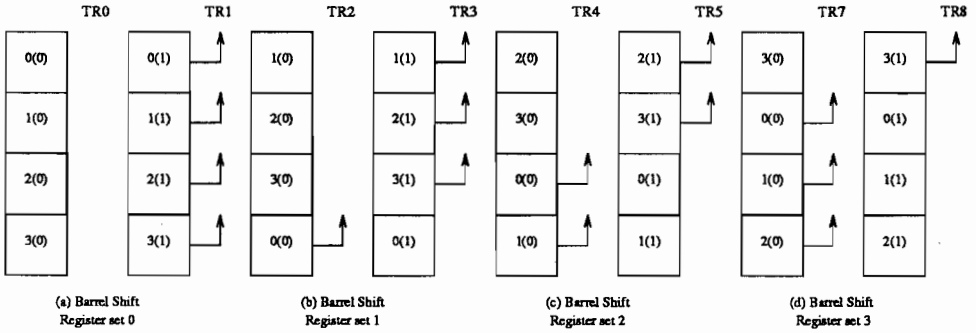


Figure 3-12: Byte configurations within the barrel shifter

The output data word is formed by selectively enabling the output tristate enable signals to latch the data onto horizontal long lines.

This process will be clear with an example. If the previous HEC byte was

inserted in the position shown in Figure 3-10(a), then barrel shift register set 1 will be used to form the output for the next cell. The byte configuration for barrel shift register 1 is shown in Figure 3-12(b). To form an output word out of this register, bytes 1, 2, and 3 are latched out of register 1 and byte 0 is latched out of register 0. Note that the word in register 1 was latched one clock cycle before the word in register 0. Hence, the output word ordering is actually bytes 0, 1, 2 and 3, which is the right order.

Output will continue to be generated from barrel shift register set 1 until the next HEC byte or path overhead byte is to be inserted. After inserting the HEC or path overhead, output will be read from barrel shift register set 2. To form an output word out of this register, bytes 2 and 3 are latched out of register 1 and bytes 0 and 1 are latched out of register 0.

Barrel shift register set 2 will be used to form the output if the HEC byte was inserted as in Figure 3-10(b) and barrel shift register set 3 will be used if the HEC byte was inserted as in Figure 3-10(c). If the HEC byte was inserted as in Figure 3-10(d), then barrel shift register set 0 will be used to output the data for the next cell. After inserting four HEC bytes or three HEC bytes and a path overhead byte, one input read operation will be skipped.

The insertion of path overhead byte is similar to the insertion of HEC byte. The path overhead byte will always be inserted in bits (7:0). If the output was being read from barrel shift register set 0 before the path overhead was inserted, after the insertion, the output will be read from barrel shift register set 1. That is, the output will be read from the next set of barrel shift registers after inserting the path overhead.

The state machine for this scheme would have five master states. In the first state, barrel shift register set 1 is used to output the data. In the second state, barrel shift register set 2 is used to form the output and in the third and fourth states, barrel shift register set 3 and barrel shift register set 0 form the output

words, respectively. During the fifth state, the input read operation is skipped. The output of the latch forms the input for all the six barrel shift registers, as shown in Figure 3-11. The outputs of all the barrel shift registers are connected to the horizontal long lines through the tri-state buffers. These tri-state buffers are enabled selectively by the state machine.

3.5 Design for 4 x STS-3c transmitter using 51.44 ns clock

In the design that was presented earlier, byte interleaving was the only function that was performed using an 77.76 MHz clock. In the design presented here, **all** the functions can be performed using a 19.44 MHz clock.

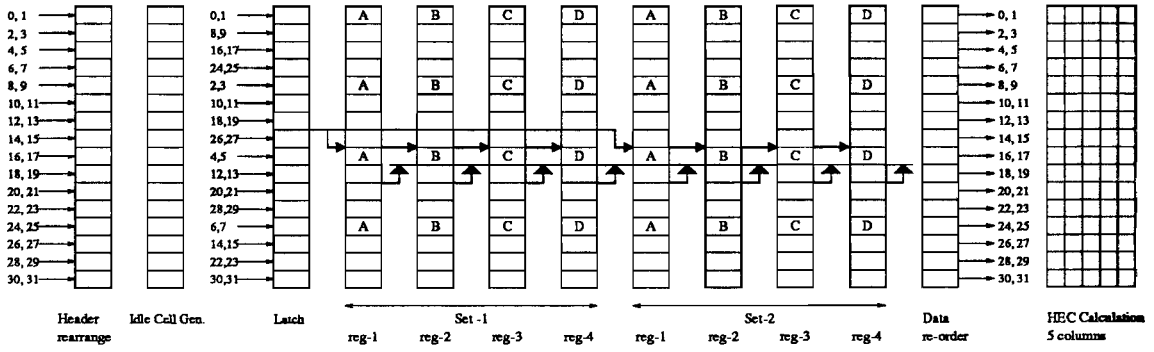


Figure 3-13: Transmitter design for STS-3c mode using 51.44 ns clock.

The floor plan for the design using Xilinx XC3195 is shown in Figure 3-13. After performing header rearrange and idle cell generation functions, the data is re-arranged and latched into the *Latch* column in Figure 3-13. Two sets of registers are used for byte interleaving. The output of *Latch* is latched into either the first register of the first set or the first register of the second set of registers. The two sets of registers are alternated to form the output interleaved word. When one set of registers is used to form the output word, the other set is used to buffer the input words.

To buffer the input words, the four registers within a set can be connected in a shift register configuration, so that the load on the input bus will not be high. The input is latched into register 1 of set 1 or register 1 of set 2. During each clock, the data is shifted right to the next register within a set. Thus, at the end of four clock periods, four new words are present that can be used to generate the output. The word in reg-1 in Figure 3-13 corresponds to the stream 1 input word.

The word in reg-2 corresponds to the stream 2, the word in reg-3 corresponds to stream 3 and the word in reg-4 corresponds to stream 4 input words.

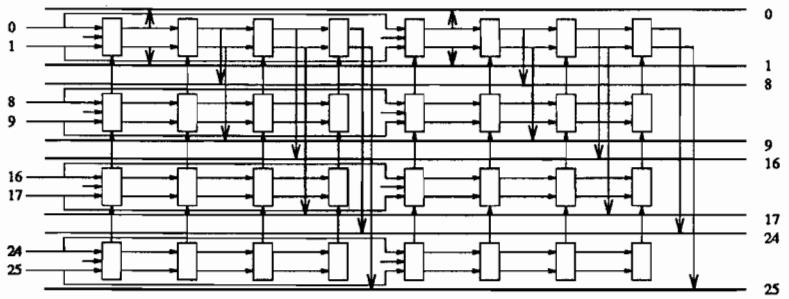


Figure 3-14: CLB output connections to long lines for 4 x STS-3c mode transmitter.

To interleave the bytes at the output the horizontal long lines are used as shown in Figure 3-14. Figure 3-14 shows the top four CLBs of both sets. Assume that the present output word is generated using the data in set 2 registers. With reference to Figure 3-13, only CLBs A, B, C and D of set 2 are used to generate the interleaved word during a given clock period. All the registers of set 2 are up shifted by two bits, so that the next byte can be used to generate the output during the next clock period. Note that because of the data order within the registers, up-shifting by two bits corresponds to up shifting by one byte. The output of CLB A is tied to long lines just above and below it, as shown in Figure 3-14. The outputs of the CLBs B, C and D are tied to the long lines as shown in Figure 3-14. The enables to the long lines are controlled using a state machine. Set 2 registers can be used to form four output words. In the mean time four new words have been written into set 1 registers. So, the next four words use the data from set 1 and set 2 will be filled with input words in the mean time.

The enables for latching the input will ping-pong between set 1 and set 2 registers. The output enables will also ping-pong between set 1 and set 2 registers. When the input is enabled to set 1 registers, the output of set 2 registers will be enabled. The maximum vertical distance for any net is 4 CLBs to connect to the horizontal long lines. Two controls are required to perform byte interleaving. A

parallel load/upshift control, which would enable a set of registers to either buffer input data or use the buffered data to form interleaved data at the output. The second control is used to enable the tri-state buffers of one of the sets.

HEC is calculated on the interleaved word using an 8 bit parallel HEC calculation circuit. This implementation uses 5 columns. To insert the path overhead word or the SONET overhead, the clocks to all the registers are masked.

3.6 Summary

The 12.8 ns clock designs for both the STS-12c and the 4 x STS-3c transmitters have been verified by simulating the RTL VHDL model using Mentor Graphics tools. Modula-3 code was written to map the designs into Xilinx XC3195 chips and generate the Xilinx design files. The maximum delay through any net in the STS-12c transmitter Chip1 was found to be 12.0 ns. The maximum delay through a net in the 4 x STS-3c transmitter Chip1 was also 12.0 ns.

Chapter 4

Receiver Architecture

The ATM delineators use two Xilinx XC-3195 chips. A block diagram of the delineators in the receive section of the gateway card is shown in Figure 4-1. The first delineator gets the input data from the *Path termination* chip. It gets the control signals *Stream_Payload(3:0)* and *Stream_PathOvh(3:0)* from the *Path termination* chip. The signals *Stream_Payload(3:0)* indicate the presence of valid payload in the input data. The *Stream_PathOvh(3:0)* signals indicate that the path overhead byte is present in bits (7:0) of the next input word. For the STS-12c mode, only bits *Stream_Payload(0)* and *Stream_PathOvh(0)* are used.

The first delineator asserts *Stream_ValidCell(3:0)* signals when the cell it processed has a valid HEC and it is not an idle cell. It asserts the *Stream_ValidHdr(3:0)* signals to the second chip when a valid header is transferred. For the STS-12c mode, only signals *Stream_ValidCell(0)* and *Stream_ValidHdr(0)* are used.

The data output of the first chip is the input to the second delineator chip. The data is clocked out of the first chip using the ATM_DataOutClk in the STS-12c mode. The output data from Chip1 is written to the SRAM for further processing. Chip2 sends a request to write to the *Receive Buffer Controller* by asserting one of the *ATM_Request(3:0)* signals when it wants to write to the SRAM.

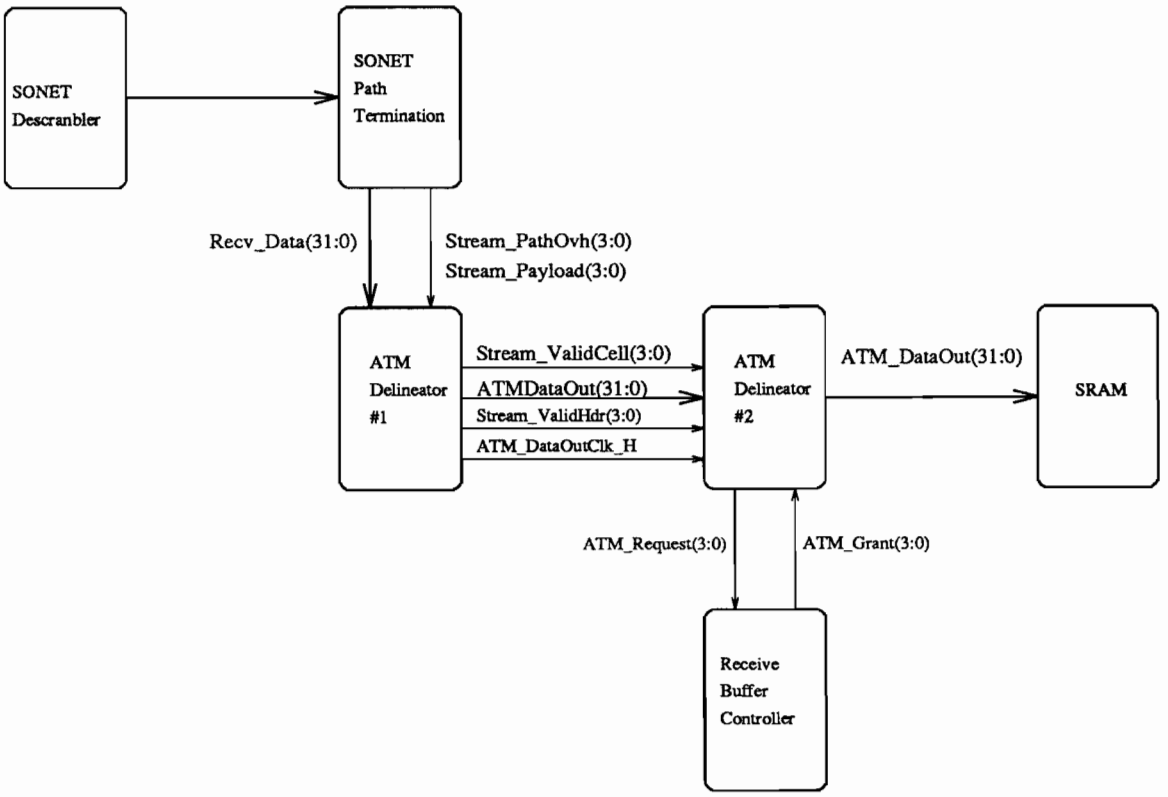


Figure 4-1: System block diagram of the Receiver.

When it gets a grant on the corresponding *ATM_Grant(3:0)* from the *Receive Buffer Controller*, it outputs the data.

Behavioral models, schematics and FPGA layouts of the receiver chips are given in Appendix E for the STS-12c mode and in Appendix F for the 4 x STS-3c mode. The simulation results of the transmitter and receiver connected together are given in Appendix C for the STS-12c mode and in Appendix D for the 4 x STS-3c mode. The simulation results and FPGA layout for the STS-12c design using 51.44 ns clock is given in Appendix G. This design was directly implemented in by writing Modula-3 code.

The STS-12c receiver design using 51.44 ns clock and the 4 x STS-3c designs were simulated using the VHDL code derived from Modula-3 code. All of these designs were simulated by connecting Chip1, Chip2 and the *Receive Buffer Controller*. These results are given in Appendix H.

4.1 Functions of ATM Delineators

The functions of the ATM Delineators are:

- synchronize with the input data to delineate ATM cells,
- compute and compare the HECs of each incoming ATM cell,
- discard cells with incorrect HECs,
- detect and discard idle cells,
- discard the HEC byte and the Path overhead byte so that the output stream can be re-ordered as contiguous 32 bit words,
- re-arrange the ATM header bits as required by the AN2 crossbar,
- de-interleave the input to form a 32 bit word for each of the four channels in STS-3c mode.

4.2 Architecture for the STS-12c mode

4.2.1 Datapath of ATM Delineator Chip1

A parallel architecture is used to implement the STS-12c ATM cell delineation function. Chip1 performs all of the functions listed earlier. An ATM cell in the SRAM occupies thirteen 32 bit words. The organization of the data within the SRAM is similar to the organization of the data within the transmit FIFOs. The organization of data in the transmit FIFOs is shown in Figure 3-2.

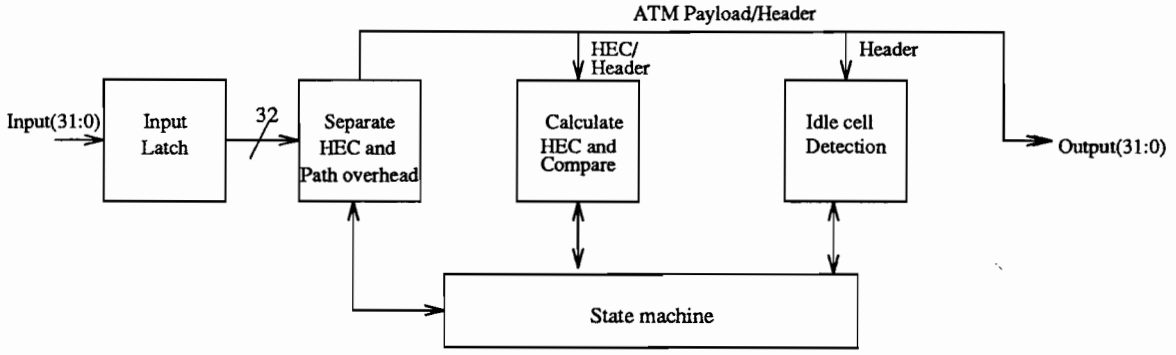


Figure 4-2: Functional block diagram of STS-12c ATM Delineator - Chip1.

A functional block diagram illustrating the data flow through this chip is shown in Figure 4-2. The input data is latched at the IOBs. The input data is read using Recv_WordClkP0_H clock, which is a 51.44 ns clock. Next, the data is formatted to separate the HEC byte or the path overhead byte from the input data stream. HEC is computed on the input header word and compared with the input HEC byte. Next, the ATM cell payload is descrambled. Idle cell detection is done by comparing the input header word with the bit pattern of an idle cell header. Chip1 outputs a complete ATM cell in thirteen 32 bit words. The HEC byte, path overhead byte and the SONET overhead are discarded from the ATM cell before the data is output. The output data clock has a period of 51.44 ns, normally. When the HEC byte or the path overhead byte is discarded, the output clock period is 64.3 ns. The output clock is called ATM_DataOutClk_H from here on.

No data is output when SONET overhead is at the input of the chip.

The Xilinx XC3195 floor plan of the chip's data path is shown in Figure 4-3.

The input, which is 32 bits wide, is latched in the input pads using a 51.44 ns clock. The input data is re-ordered and latched by the *Input Register* column. The data order is as shown in Figure 4-3. The re-ordering is necessary to localize the interconnections in *ShiftReg-1* and *ShiftReg-2*. By using this order, the entire word can be shifted up or down by 8 bits by just moving the bits up or down by one CLB. This is explained next.

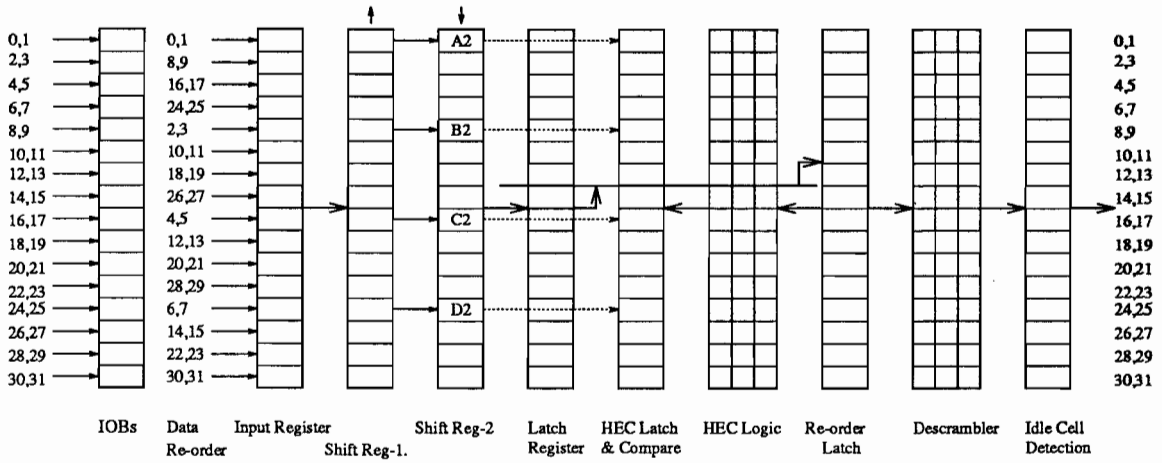


Figure 4-3: STS-12c ATM Delieator - Chip1.

The output of the *Input Register* column is latched by *ShiftReg-1*. The data in *ShiftReg-1* are upshifted by 2 bits every 12.86 ns, which is one fourth of the 51.44 ns rate at which the input is latched. The 12.86 ns clock is called ByteClk. Using the ByteClk, 8 bits are shifted right to the *ShiftReg-2*, as shown in Figure 4-3. In this figure, the eight bits are shifted right from *ShiftReg-1* to CLBs A2, B2, C2 and D2 of *ShiftReg-2*. The data in *ShiftReg-2* is also shifted down using the ByteClk. An illustration of the operation of *ShiftReg-1* and *ShiftReg-2* is shown in Figure 4-4.

When the HEC is present in cells A2, B2, C2 and D2 of Figure 4-3, *ShiftReg-2* is not down shifted. Instead, the HEC is latched into the HEC latch register.

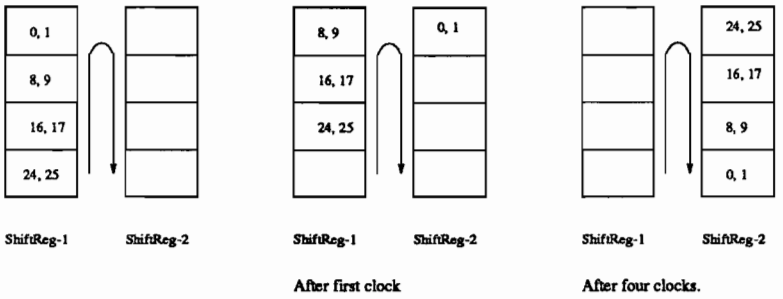


Figure 4-4: Illustration of the shift register operation

Only the payload and the header are latched into the *Latch* register. The next column is used to store the HEC byte and compare it to the calculated value of the HEC. The HEC computation logic is implemented in the next 3 columns, as shown in Figure 4-3. The following polynomial is used to calculate the HEC [1]:

$$1 + x + x^2 + x^8. \tag{4.1}$$

After calculating the HEC, the result is added modulo-2 with the following coset polynomial:

$$1 + x^2 + x^4 + x^6. \tag{4.2}$$

A 32 bit parallel HEC calculation circuit is used. The data is then re-ordered to get back the input order and latched by the *Re-order* latch.

The next three columns implement a self synchronous descrambler. Descrambling is performed using the following polynomial:

$$x^{43} + 1. \tag{4.3}$$

A conceptual diagram illustrating the operation of the descrambler is shown in

Figure 4-5. The descrambler used in this implementation is a 32 bit parallel descrambler. Figure 4-6 shows the descrambler organization in the Xilinx. The equations for the parallel descrambler can be easily derived from the serial descrambler. VHDL code to perform 32 bit parallel descrambling is given below.

```
FOR I IN 0 TO 31 LOOP
  bit(I) <= bit(I) XOR state_bit(42-I)
END LOOP;
FOR I IN 0 TO 31 LOOP
  state_bit(31-I) <= bit(I)
END LOOP;
FOR I IN 32 TO 42 LOOP
  state_bit(I) <= state_bit(I-32)
END LOOP;
```

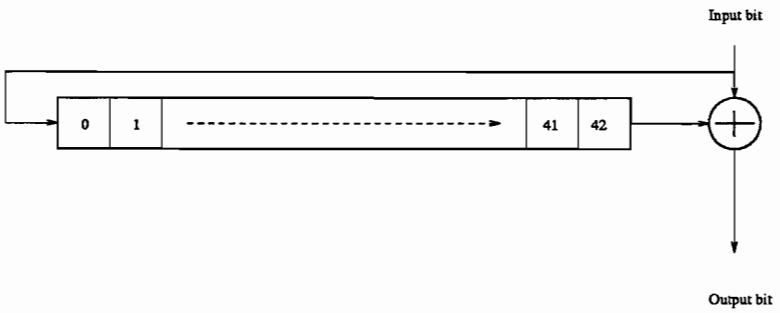


Figure 4-5: ATM self synchronous descrambler

The next column detects idle cells. An idle cell or an unassigned cell is defined in [1] as a cell which has the header as shown in Figure 4-7. The bits labeled *a* in the figure are considered as *don't care* bits for idle cell detection. The HEC of an idle cell is the HEC computed over the first four bytes of the header and it should match the received HEC byte. Hence, this column basically detects this pattern in the header of the ATM cell and sets a signal called *IdleCell* to high if the pattern is detected. This signal is used in generating the *ValidCell* signal which is used to discard cells from being stored in the SRAM.

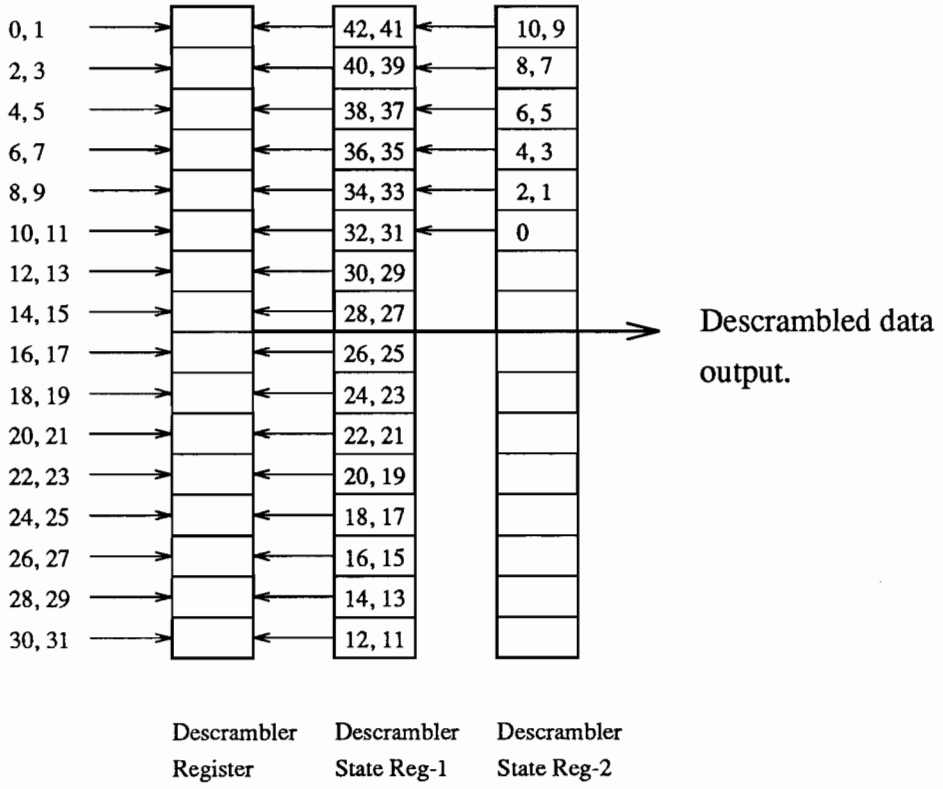


Figure 4-6: 32 bit parallel descrambler organization in the Xilinx

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	a	a	a	0

"a" indicates that the bit is available for the appropriate ATM layer function.

Figure 4-7: Idle cell header bits.

Time tick	Action
T0	Data at the input of Chip1 IOBs.
	Stream_PathOvh signal after latching at the IOB.
T1-t0	Data at the input of latch register.
	Stream_PathOvh signal delayed by 1 clock.
T1-t1	Stream_PathOvh signal reduced to 1 byte-clock wide signal.
T1-t2	Delayed byte clock wide Stream_PathOvh signal.
T1-t3	Stream_PathOvh signal delayed by two byte-clocks.
T2-t0	Data latched into Upshift register. Stream_PathOvh signal delayed by three byte-clocks.
	Disable DownShift signal generated using Stream_PathOvh signal delayed by three byte clocks.(before latching)
T2-t1	Path overhead byte is latched into Down-shift register.
T2-t2	Path overhead byte is overwritten by the next byte.

Table 4.1: Pipeline delay through Chip1 - illustrates the relative timing between path overhead signal and the word containing path overhead.

Table 4.1 shows the relative delay between the *Stream_PathOvh* signal and the path overhead byte in the *ShiftReg-2* where it is discarded. Table 4.2 shows the pipeline delays through the different stages in Chip1.

4.2.2 State Machine for Chip1

The signals generated by the state machine and a brief description of their use are listed below.

- Hunt, Sync and Pre-Sync: These signals indicate the ATM delineation state of the receiver.
- ATM_DataOutClk_H: This is the clock with which the data is read out of the chip. This clock is slipped by 12.86 ns when the path overhead byte or the HEC byte is discarded from the data stream. The clock is masked when the input stream consists of SONET overhead data.
- WdClkMask_L: This masks the ATM_DataOutClk_H.

Time tick	Action
T0	Header latched into LatchPayload register.
	HeaderLatchSt2 signal is valid.
T1	Header re-ordered in Re-Ord register.
	Computed value of HEC will become valid after next rising edge.
	HeaderLatchSt3 signal is valid.
	HECMatchInpSt1 signal generated by HEC Compare circuit.
T2	Header latched out of the de-scrambler(without descrambling).
	Stream_ValidHdr(0) generated.
	Idle cell detected and IdleCellInp signal generated.
	Stream_ValidCell(0) signal generated using IdleCellInp and HECMatchInpSt1.
T3	Header latched out of Chip1.
	Stream_ValidHdr(0) latched out of Chip1.
	Stream_ValidCell(0) signal latched out of Chip1
T4	Stream_ValidHdr(0) latched into Chip2.
	Header latched into Chip2; Header at input of Latch register.
	Stream_ValidCell(0) latched into Chip2; used to control latching the input word at the latch register.
T5	Header word at the input of Header re-arrange register in Chip2.
	Stream_ValidHdr(0) delayed by one clock at the input of header re-arrange register; used to control header multiplexing at the header re-arrange register.

Table 4.2: Pipeline delay through Chip1.

- ParLd_H: When this signal is high, *ShiftReg-1* is parallel loaded with data from the *Latch.* Otherwise, *ShiftReg-1* bits are shifted up by two bits every byte clock period.

- DnShftEnb_H: This signal is used to control the enable to *ShiftReg-2.* When the state is in hunt, this signal goes low once every five 12.86 ns clocks. When the state is in sync or presync, this signal goes low exactly when the HEC byte or the path overhead byte is present in the positions A2, B2, C2 and D2 of *ShiftReg-2* in Figure 4-3.

- HECLatch_H: This signal is used to latch the HEC byte into the HEC latch and compare register. When the state is in hunt, this signal goes high once every five byte clock periods. When the state is in pre-sync or sync, this signal goes high exactly when the HEC byte is present is the A2, B2, C2 and D2 positions of the *ShiftReg-2.*

- Stream_ValidHdr_H: Indicates that the there is a valid header word being read out of the chip.

- Stream_ValidCell_H: Indicates that a valid cell is being read out of the chip.

In order to minimize the state decoding and generating delays, a 53 bit one hot state machine is used. The states are reset whenever the ATM state is in hunt. When the first HEC match is encountered, a byte clock wide pulse is injected into the one hot state machine, which then keeps circulating.

4.2.3 Chip2 Datapath

This chip reads its input from Chip1. It re-arranges the header bits as required by the AN2. It implements a FIFO for rate adjustment to write the data to an SRAM. This chip writes the data using a 40ns clock. When the chip has a valid word to write into the SRAM, it issues a request to the *Receive Buffer Controller.*

A subsequent grant from the *Receive Buffer Controller* will complete the write operation.

A block diagram of the data path for Chip2 is as shown in Figure 4-8. The chip reads the input data using the clock ATM_DataOutClk_H. The input data is first latched in the *Latch* column. This chip re-arranges the header bits into the format shown in Figure 3-2. The second column of CLBs use the *Stream_ValidHdr(0)* signal to control the header re-arrange function. The next eight columns implement a FIFO to buffer the data and read it out at the appropriate time. The input signal *Stream_ValidCell(0)* remains high for the duration of a valid cell input. When the input data is part of a valid cell, data is written into one of the eight FIFO registers. Simultaneously, a request is sent out to *Receive Buffer Controller* by asserting one of the bits of the signal ATM_Request_H. Requests are issued in a round-robin fashion on signals ATM_Request_H(0), ATM_Request_H(1), ATM_Request_H(2) and ATM_Request_H(3), so that similar circuit can be used in the *Receive Buffer Controller* for the STS-12c and the 4 x STS-3c modes. The *Receive Buffer Controller* will subsequently issue a grant corresponding to this request by asserting one of the ATM_Grant_H signals. The grant will enable the output of the FIFO register to be output.

4.2.4 State machine for Chip2

The state transition diagram for the request grant states is shown in Figure 4-9. There are two independent finite state machines, one for the writing into the buffers and another for reading out of the buffers. Data is written into or read from one of the eight buffers depending upon the state of the respective state machine. After a read or write the state of the respective state machine transitions so that the next time the next buffer is used for reading or writing.

For example, if the request state machine is in State0, then the 0th buffer is used to write the next word. After writing, the request state machine would get

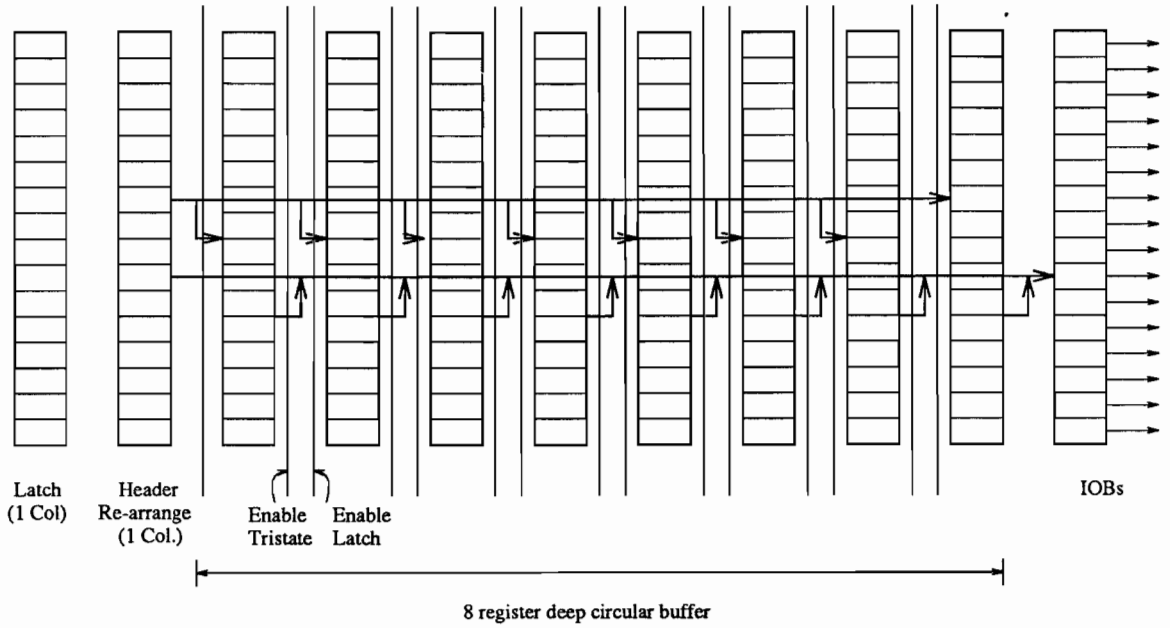


Figure 4-8: STS-12c ATM Receiver - Chip2.

into State1. The grant state machine operates similarly. After receiving the first grant, data is read from the 0th buffer. After reading data, the state of the grant state machine goes to State1.

Hence, during each request state, one of the eight buffers is enabled for writing a 32 bit data word. During each grant state, one of the eight buffers is enabled to write the data on to horizontal long lies. The request state determines which of the eight buffers to write. The grant state determines which of the eight states to read.

The following list contains the various signals that are used or generated by the state machine.

- ATM_Request_H(3:0): This signal is asserted to issue a request to write a word into the SRAM. This signal is synchronized to the AN2_WordClk_H. A request is issued when a valid word has been written into one of the eight FIFO registers. The write to the FIFO registers is synchronous to the ATM_DataOutClk_H. Hence, the ATM_Request_H signals are made syn-

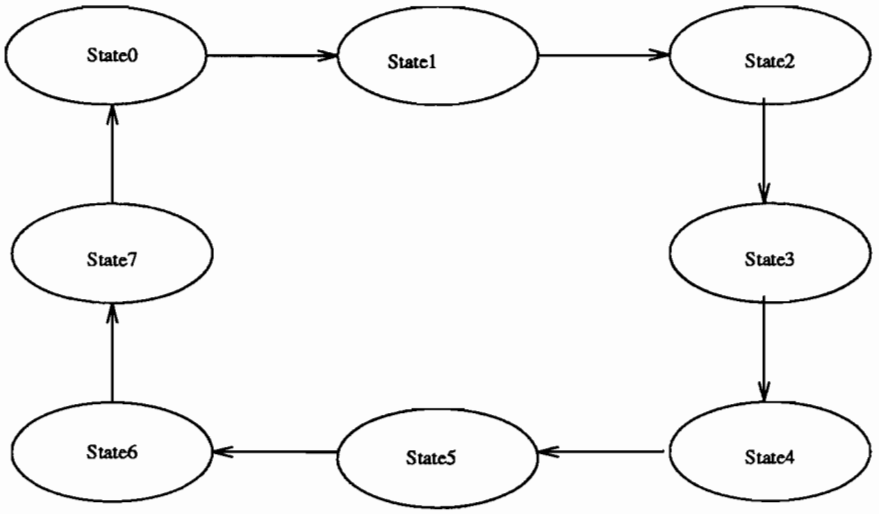


Figure 4-9: State machine of OC-12c Chip2 receiver

chronous with the AN2_WordClk_H.

- ATM_Grant_H(3:0): This is an input signal which indicates that a request has been granted. The appropriate FIFO register is read out by enabling the tri-state buffers at the output of the register. The output IOBs are also enabled and the data is written into the SRAM.

- Stream_ValidCell_H(0): This signal is used by the state machine to store the input words into the FIFO buffers.

- Stream_ValidHdr_H(0): This signal is used by the state machine to re-arrange the header bits.

- EnClkBuf_H(7:0): This signal is generated by the state machine to enable the appropriate FIFO buffer.

- EnTriSt(7:0): This signal is generated by the state machine to enable the output tristate buffers of the FIFO registers.

The timings of the control signals to generate an *ATM_Request* is shown in Figure 4-10. Figure 4-11 shows the timings for the control signals to output the

data.

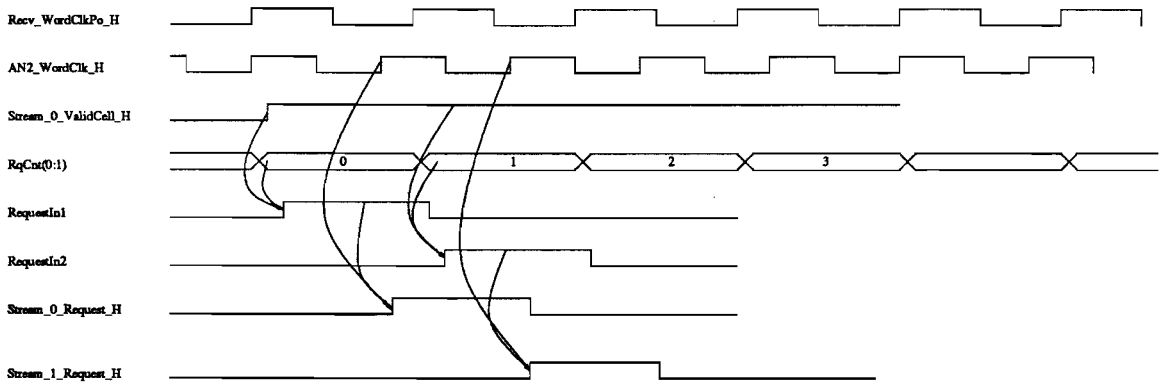


Figure 4-10: STS-12c ATM Receiver - Chip2 Timing to generate an *ATM_Request signal.*

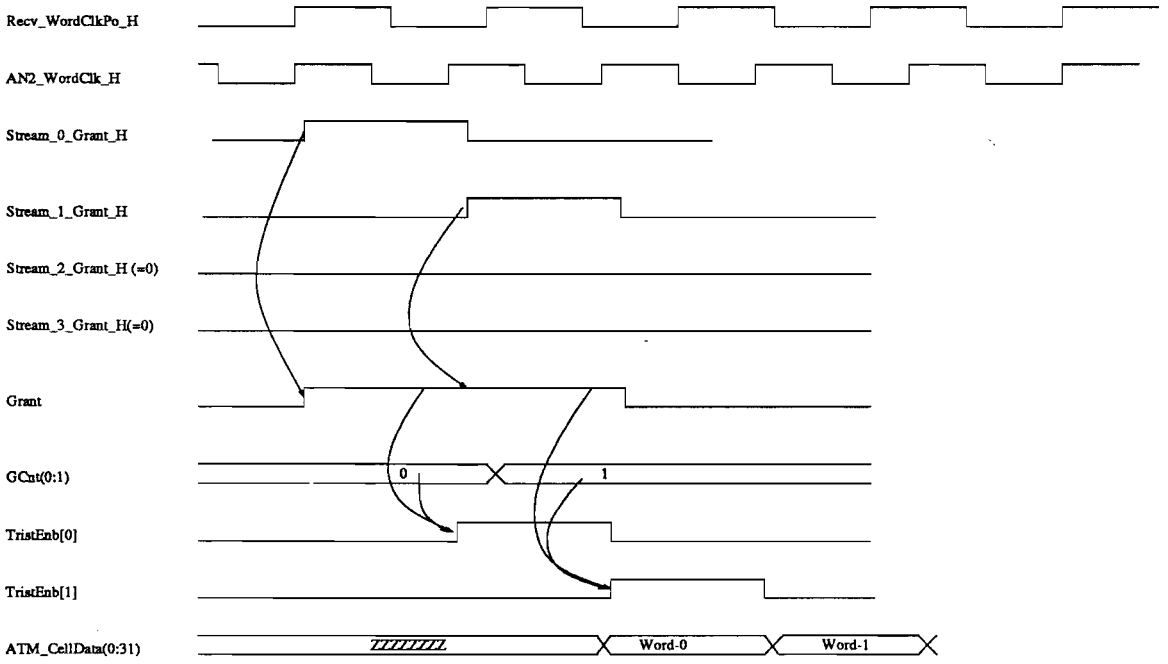


Figure 4-11: STS-12c ATM Receiver - Chip2 Timing to generate the output data.

4.3 Architecture of the 4 x STS-3c mode

4.3.1 Datapath of 4 x STS-3c delineator Chip1

Chip1 of the 4 x STS-3c architecture performs the following functions:

- descrambles the incoming ATM cells,
- computes the HEC byte and compares it to the incoming HEC byte, and
- synchronizes with the incoming data stream by locating valid cell headers in the input stream.

A block diagram of the Chip1 data path is shown in Figure 4-13. This design maps into the same two XC3195 chips that are used for the STS-12c mode. In this chip, there are four completely independent state machines controlling the four independent input channels. A functional block diagram of the chip is shown in Figure 4-12. The input data is latched at the IOBs. It is then descrambled. Next, the HEC is computed on the header and compared with the received HEC byte. Idle cell detection logic is implemented next.

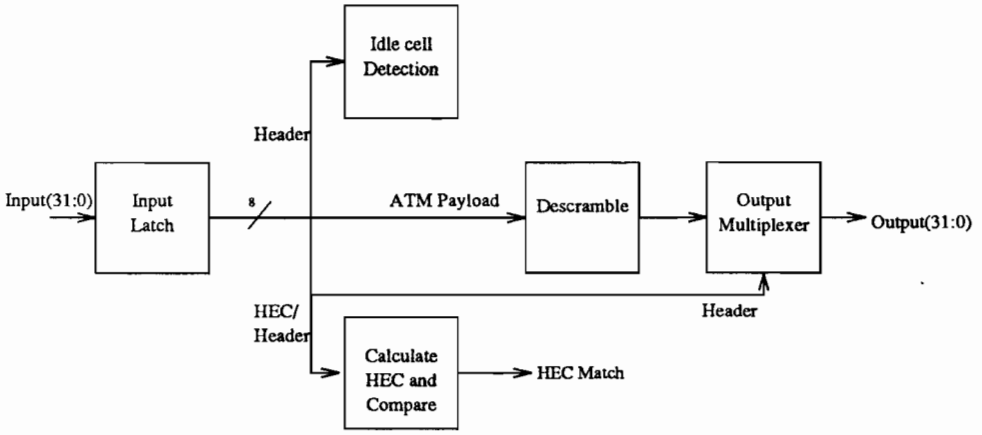


Figure 4-12: Functional block diagram of 4 x STS-3c ATM Delineator - Chip1.

The input, which is 32 bits wide, is formed from byte interleaved data from the four independent channels. Thus bits (7:0) belong to channel 1, (15:8) belong to

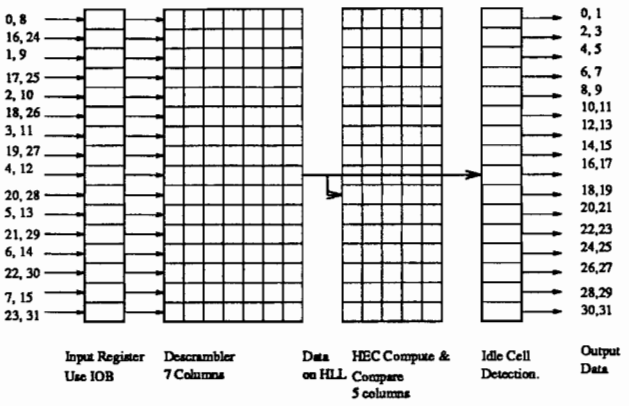


Figure 4-13: 4 x STS-3c ATM Receiver - Chip1.

the second channel, and so on. The input is latched using a 51.44 ns period clock at the IOBs. The data is then descrambled. The polynomial used for descrambling is the same as the one used in STS-12c mode. A conceptual diagram illustrating the operation of the descrambler is shown in Figure 4-14. For this mode, an 8 bit parallel version of the descrambler is used, instead of the 32 bit parallel descrambler that was used in the STS-12c mode. The VHDL code to perform descrambling are given below:

```
FOR I IN 0 TO 7 LOOP
  Desrambled_bit(I)            <= Input_bit(I) XOR Descramble_state_bit(42-I);
  Descramble_state_bit(I)      <= Input_bit(7-I);
  Descramble_state_bit(42 - I) <= Descramble_state_bit(34 - I);
  Descramble_state_bit(34 - I) <= Descramble_state_bit(26 - I);
  Descramble_state_bit(26 - I) <= Descramble_state_bit(18 - I);
  Descramble_state_bit(18 - I) <= Descramble_state_bit(10 - I);
END LOOP;
  Descramble_state_bit(10)     <= Descramble_state_bit(2);
  Descramble_state_bit(9)      <= Descramble_state_bit(1);
  Descramble_state_bit(8)      <= Descramble_state_bit(0);
```

The organization of the descrambler in the Xilinx chip is shown in Figure 4-15. It takes seven columns to implement the descrambler. Only the payload portion of the ATM cell is descrambled. The generation of the necessary control signal is described in the section on the state machine.

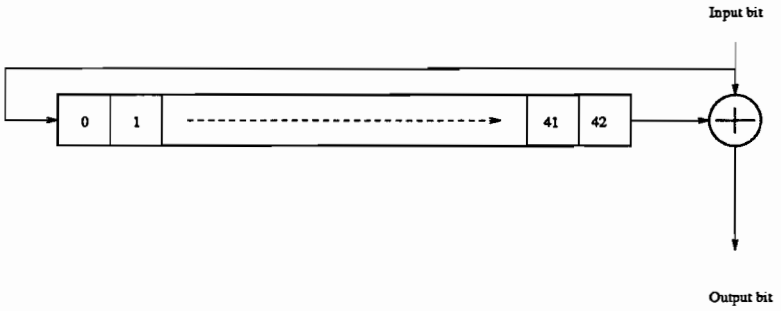


Figure 4-14: ATM self synchronous descrambler

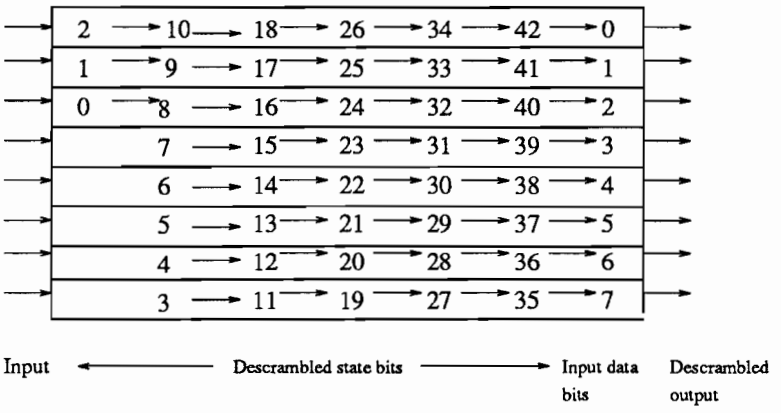


Figure 4-15: Organization of the ATM descrambler for the 4 x STS-3c mode

The next four columns are used to generate the HEC from the input header word. The polynomial used for HEC generation is the same as for the STS-12c mode. For this mode, an eight bit parallel HEC calculation circuit is implemented.

The next column is used to compare the computed value of the HEC to the incoming HEC byte. An *HECMatch* signal is asserted to indicate that the values match. This signal is used in the state machine.

The next column is used to detect idle cells. The idle cell header bit pattern is shown in Figure 4-7. If the input is an idle cell, then an *IdleCell* signal is asserted, which causes the Stream_ValidCell_H signal to go low. This prevents storing idle cells in the SRAM. Table 4.3 shows the pipeline delay through the first chip. The state that this table refers to is the state of the 53 bit state machine. As mentioned earlier, there are four independent state machines, one for each channel.

4.3.2 State machine for Chip1

There are four independent state machines, one for each channel, each of which perform the following functions:

- ATM Delineation: The state machine starts with *Hunt*. When the first HEC match occurs, the state goes to *Presync* and then after 6 consecutive HEC matches, the state goes to *Sync*.

  In order to get the first HEC match in the hunt state, the HEC is computed over four bytes and compared to the fifth incoming byte. The sixth clock period is used to clear the feedback registers where the computed value of HEC was stored. By looking for an HEC match in a six clock-period window, the first HEC match is assured to be found within six ATM cell periods.

- Generate the following signals for use in the data path:

  - ClearFBRegs_H: A signal used to clear the feedback HEC registers. The feedback registers will be cleared before a new HEC calculation.

Time tick	Action
T0	Header Byte-1 latched into Chip1 IOBs.
	State = 46
	ClearFBRegsL signal after being latched
	ResetIdleCell signal after being latched
	DisableDescrambler signal true (after latch).
T1	Data latched out of the de-scrambler
	State = 47
	DisableDescrambler signal true (after latch)
T2	Data latched out of Chip-1.
	State = 48
	DisableDescrambler signal true (after latch)
T3	Data latched into Chip-2.
	State = 49
	DisableDescrambler signal true (after latch)
T4	HEC byte latched into Chip-1.
	Byte4 signal generated (after latch).
	State = 50
T5	IdleCell signal generated.
	HEC Computed compared with HEC received.
	State = 51
	State-51 latched out as *Stream_ValidHdr*.
T6	Fourth byte of the header latched into chip-2.
	HECMatch signal is generated.
	IdleCell signal delayed by 1 clock period.
	State = 52
T7	De-interleaved header word at the input of header re-arrange register.
	HEC byte at the input of the de-interleave register.
	Stream_ValidHdr used to disable de-interleave register (to discard the HEC byte) and to re-arrange header word.
	Stream_ValidCell signal generated.
	State = 0
T8	Header word at the input of FIFO register.
	Stream_ValidCell signal used to disable FIFO input latching.

Table 4.3: Pipeline delay through Chip1.

- ComputeHEC_H: When high, the HEC computing circuitry is enabled. This signal normally remains high. It is used to disable the HEC calculation when the path overhead or the SONET overhead occurs between the header bytes.
- Descramble_H: Used to enable descrambling.
- ResetIdleCell_L: This signal is used for idle cell detection. It is used to reset the result of the previous idle cell detection.
- Byte4_H: This signal is also used in the idle cell detection logic. It is used to indicate the fourth byte of the header. This is necessary because in an idle cell, some bits of the fourth byte are *don't care* bits.

- Stream_ValidCell indicates that the cell has a valid HEC and that it is not an idle cell. It changes at cell boundaries and is used by the second chip.
- Stream_ValidHdr is used by the second chip to re-arrange the header appropriately.

The state machine to count the bytes in an ATM cell is a one hot state machine. It has 53 states, 0 to 52. The state transition diagram is shown in Figure 4-16. The one hot state machine is enabled by the first HEC match when the receiver is in *Hunt* state. The one hot state machine is reset if there is a HEC mis-match during the *PreSync* state or if there are six consecutive HEC mis-matches during the *Sync* state.

The timing of the control signals is shown in Figure 4-17.

4.3.3 Chip2 Datapath for the 4 x STS-3c mode

The second ATM delineator performs the following functions for each of the four channels.

- Re-arranges the header bits appropriately.

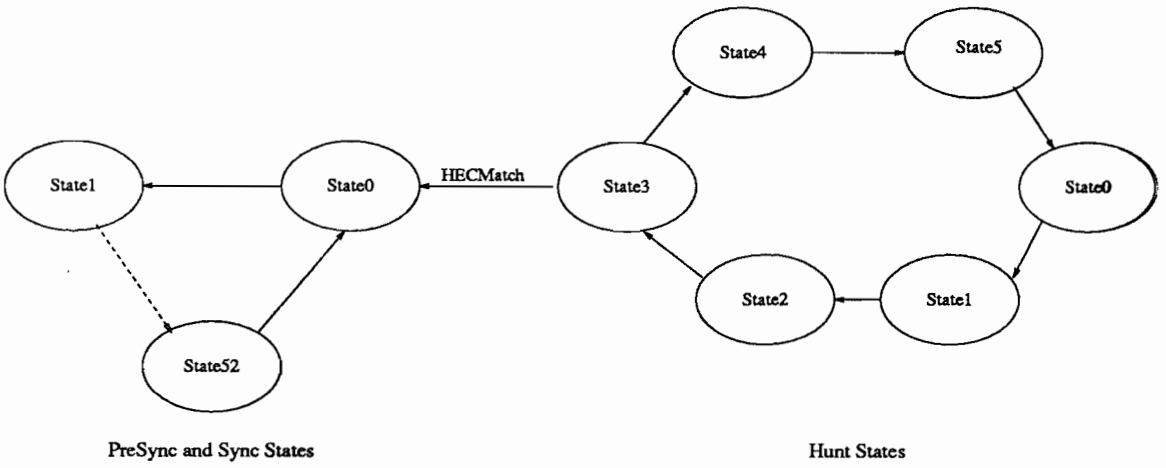


Figure 4-16: 4 x STS-3c ATM Receiver - Chip1 state transition diagram.

- The incoming data is byte interleaved, as mentioned earlier. This data is de-interleaved in this chip so that the output can be separated into words formed of one of the four streams.
- The path-overhead byte and the HEC byte are removed from the data stream in this chip.
- Buffers the de-interleaved words until resources are available to write the word into the SRAM.

The input IOBs of the second FPGA are used as a latch. The input is from the top and bottom pads, as shown in Figure 4-18. The figure shows the architecture for a single channel. The input is distributed along the top and bottom IOBs of the FPGA to reduce the interconnection delay between the de-interleave register and the input IOBs.

The data is de-interleaved by shifting the data down at 51.44 ns intervals in the *Channel register.* It takes at least four 51.44 ns clock periods to generate a complete word for a stream. It would take more than four periods if path overhead, HEC or SONET overhead occurred while the data was interleaved. If the *Stream_ValidCell_H* signal is inactive (low), the de-interleave register is disabled.

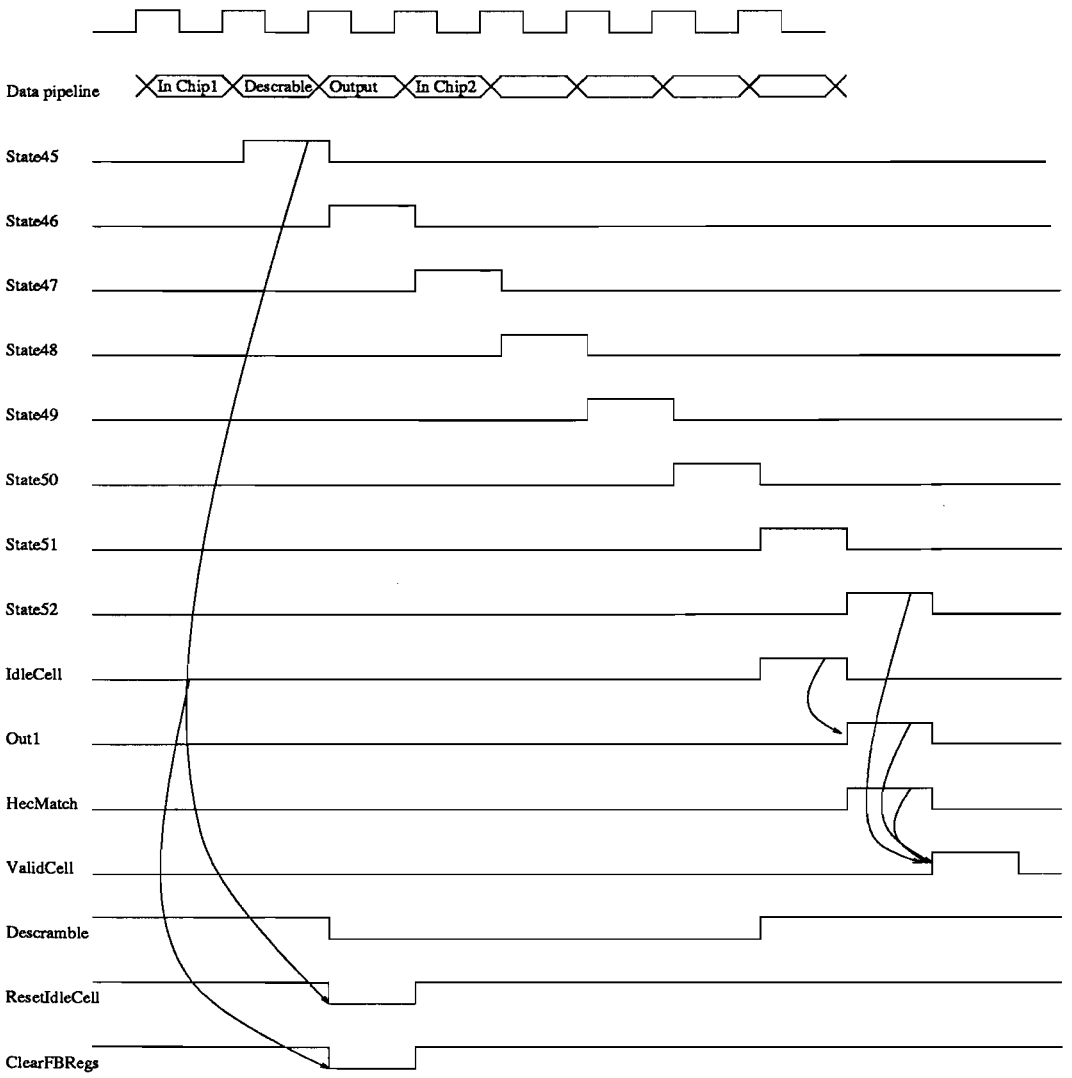


Figure 4-17: 4 x STS-3c ATM Receiver - Chip1 control signal timings.

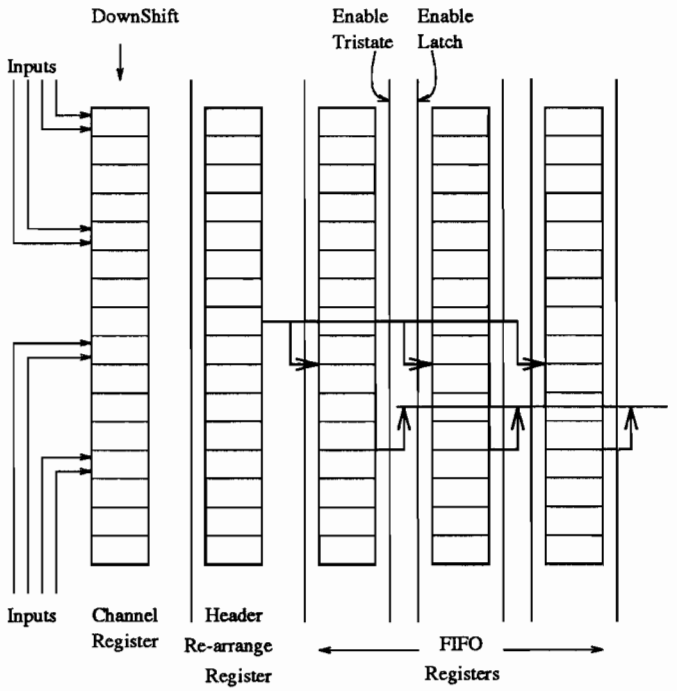


Figure 4-18: 4 x STS-3c ATM Receiver - Chip2.

After a complete word is assembled, it is output to the *Header Re-arrange* register. If the word was a valid header, the *Stream_ValidHdr* signal for that stream goes high and the bits are re-arranged and latched at the output of *Header Re-arrange* register. If the word was not a header, it is just latched out of the register. The pipeline delays for the signals *Stream_ValidHdr* and *Stream_ValidCell* are shown in Table 4.3. Table 4.4 shows the pipelining delays in generating a request and Table 4.5 shows the pipeline delay in reading the data out of the chip after getting an *ATM_Grant* from the *Receive Buffer Control* Chip.

The output of *Header Re-arrange* register is written into one of the three FIFO registers. These registers are provided for delays in the write of the un-interleaved word into the SRAM.

Time tick	Action
T0	First byte latched into de-interleave registers.
T1	Second byte latched into the de-interleave registers.
T2	Third byte latched into the de-interleave registers.
T3	Fourth byte latched into the de-interleave registers.
T4	Complete de-interleaved word latched out of the header re-arrange register.
	Request counter incremented (on the rising edge of the next AN2 Clock).
T5	Request is latched out of the chip on the rising edge of the AN2 Clock.

Table 4.4: Pipeline delay for generation of *ATM_Request* signal.

Time tick	Action
T0	Grant signal latched into the chip.
T1	Request counter decremented.
T2	Data enabled to be put on horizontal long lines.
T3	Output IOBs are enabled. Data is read out of the chip.

Table 4.5: Pipeline delay for reading the data out of Chip2.

4.3.4 State machine for Chip2

The state transition diagram for the request grant states is shown in Figure 4-19. There are two independent finite state machines, one for writing into the buffers and another for reading out of the buffers. Data is written into or read from one of the three buffers depending upon the state of the respective state machine. After a read or write the state of the respective state machine transitions so that the next time the next buffer is used for reading or writing.

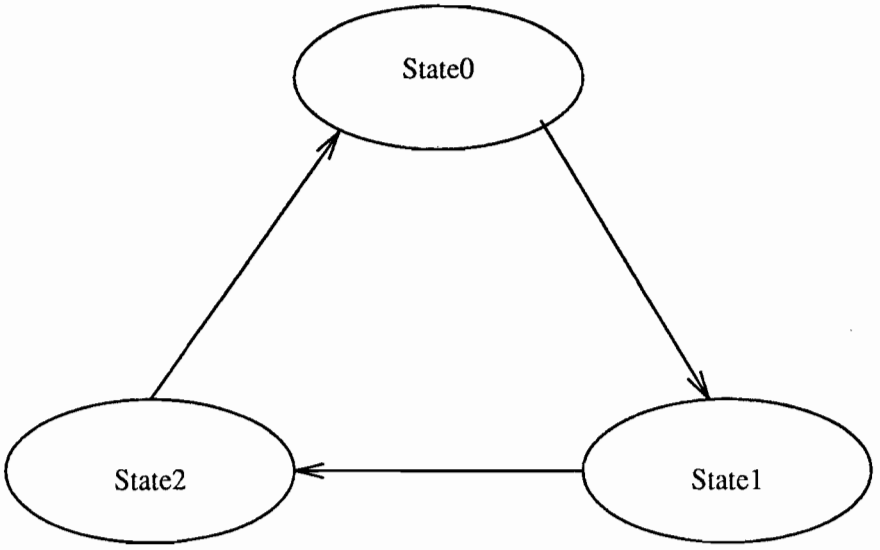


Figure 4-19: State machine of OC-3c Chip2 receiver

For example, if the request state machine is in State0, then the 0th buffer is used to write the next word. After writing, the request state machine would get into State1. The grant state machine operates similarly. After receiving the first grant, data is read from the 0th buffer. After reading data, the state of the grant state machine goes to State1.

Hence, during each request state, one of the three buffers is enabled for writing a 32 bit data word. During each grant state, one of the three buffers is enabled to write the data on to horizontal long lies. The request state determines which of the three buffers to write. The grant state determines which of the three states to read.

The state machine signals for one channel are listed below. The state machine is repeated for each channel, for a total of four copies.

- ATM_Request_H(3:0): This signal is asserted to issue a request for writing a word to the SRAM. This signal is synchronous to the AN2_WordClk_H. A request is issued when a valid word has been written into one of the three FIFO registers. The write to the FIFO registers is synchronous to the ATM_DataOutClk_H. Hence, a pulse shortening circuit is used to shorten the ATM_Request_H signal and make it synchronous with the AN2_WordClk_H.

- ATM_Grant_H(3:0): This is an input signal which indicates that a request has been granted. The appropriate FIFO register is enabled to pass data to the output bus. The output IOBs are also enabled to pass the data through so that the data is written into the SRAM.

- Stream_ValidCell_H(3:0): This signal is used by the state machine to store the input words into the FIFO buffers.

- Stream_ValidHdr_H(3:0): This signal is used by the state machine to re-arrange the header bits.

- EnClkBuf_H(2:0): This signal is generated by the state machine to enable the appropriate FIFO buffer. It is used as a one of n code.

- EnTriSt(2:0): This signal is generated by the state machine to enable the output tristate buffers of the FIFO registers. It is used as a one of n code.

The timing diagram illustrating generation of a *ATM_Request* signal is shown in Figure 4-20. Figure 4-21 illustrates the generation of the *ATM_CellData(0:31)* when an *ATM_Grant* signal is received for a channel.

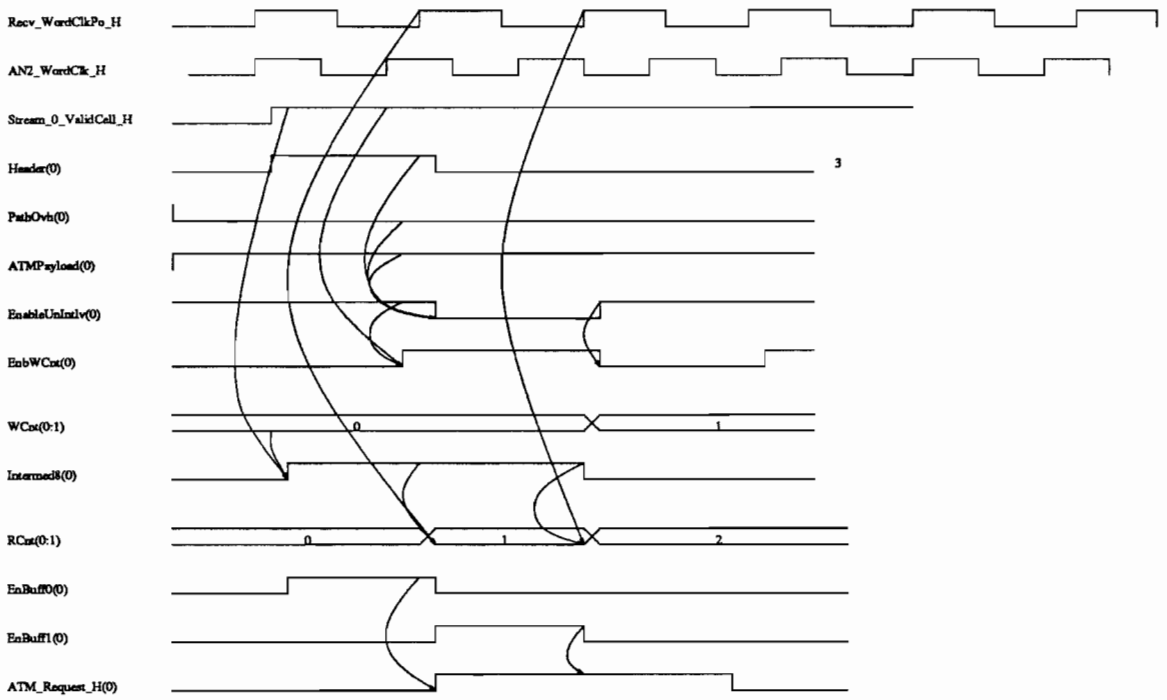


Figure 4-20: 4 x STS-3c ATM Receiver - Chip2 control signal timings to generate an ATM_Request signal.

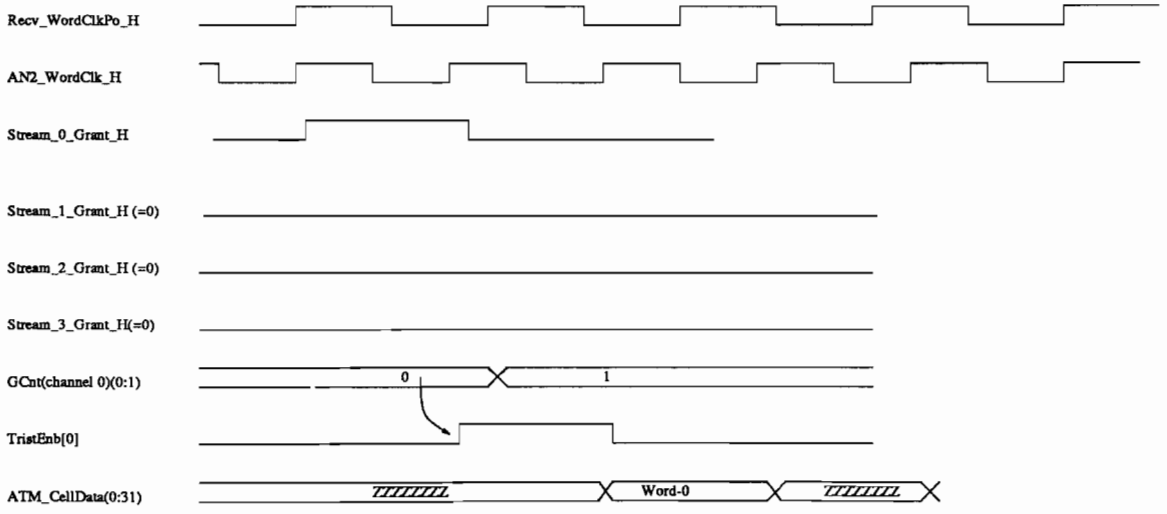


Figure 4-21: 4 x STS-3c ATM Receiver - Chip2 control signal timings to generate ATM_CellData(0:31).

4.4 Design for the STS-12c receiver using 51.44 ns clock

The design presented earlier uses a 12.86 ns clock to perform ATM cell delineation. When the state is in *Sync*, the earlier design used the 12.86 ns clock to discard the HEC byte and the path overhead byte and to re-order the output into 32 bit words. The design presented in this section uses a 51.44 ns clock to perform all of these functions.

The HEC byte in the received data stream could be in one of the four positions shown in Figure 4-22. When the delineator is in *Hunt* state, it looks for the first HEC match by searching for a valid HEC corresponding to position (a) in Figure 4-22. Once the first HEC match is found, the design looks for the HEC byte in positions (b), (c), (d) and (a) in the same order every 53 bytes. If the path overhead byte was present during a cell period, then the HEC byte for the next ATM cell will skip a position in Figure 4-22. That is, if the next HEC byte was expected in position 4-22(a) and the path overhead byte was encountered before the HEC byte, the next HEC byte will now be expected to occur in position 4-22(b).

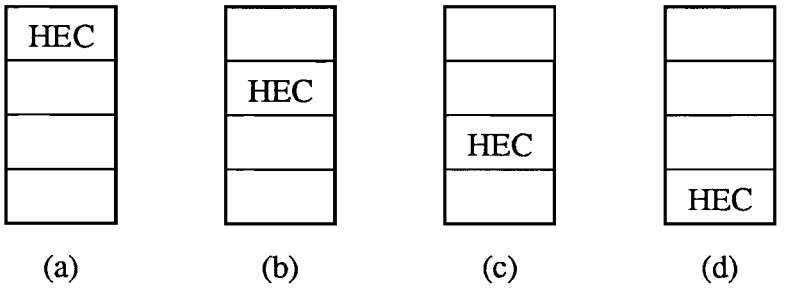


Figure 4-22: Position of the HEC byte in the output data stream.

The Xilinx XC3195 floor plan is as shown in Figure 4-23. The input data ordering is also shown in the figure. The input data is latched into the first register of each of the four sets of registers. The input to the second register of

each register set is the output of the first register of the same set. So, the data in the second register is delayed by one clock cycle. The input byte ordering for each of the four sets is shown conceptually in Figure 4-24. Note that the bits are ordered as shown in Figure 4-23. Figure 4-24 (a) shows the byte ordering within set 0. Figure 4-24 (b) shows the byte ordering for set 1. Figures 4-24 (c) and (d) show the byte ordering within barrel shift register sets 2 and 3 respectively. In this figure, bytes 0(0), 1(0), 2(0) and 3(0) correspond to the first register of the register set. Bytes 0(1), 1(1), 2(1) and 3(1) correspond to the second register of each register set. The bytes 0(1), 1(1), 2(1) and 3(1) are delayed by one 51.44 ns clock with respect to bytes 0(0), 1(0), 2(0) and 3(0). Due the order of the bits in the input bus, the maximum vertical net length to realize the byte order shown in Figure 4-24 is 3 CLBs.

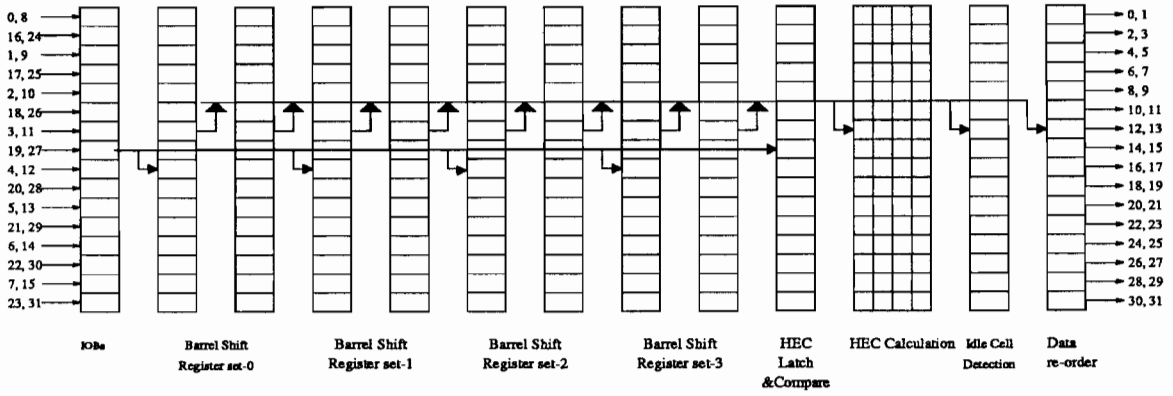


Figure 4-23: Receiver design for STS-12c mode using 51.44 ns clock.

The output data word is formed by selectively enabling the output tristate enable signals to latch the data on horizontal long lines. Figure 4-24 shows the way in which the bytes in the eight barrel shift registers are connected to the long lines through the tri-state buffers. Note that the output data on the long lines are always in the right order, which is byte 0, 1, 2, and 3.

This process will be clear with an example. If the previous HEC byte was in the position shown Figure 4-22(a), then barrel shift register set 1 will be used to form the output for the next cell. The byte configuration for barrel shift register

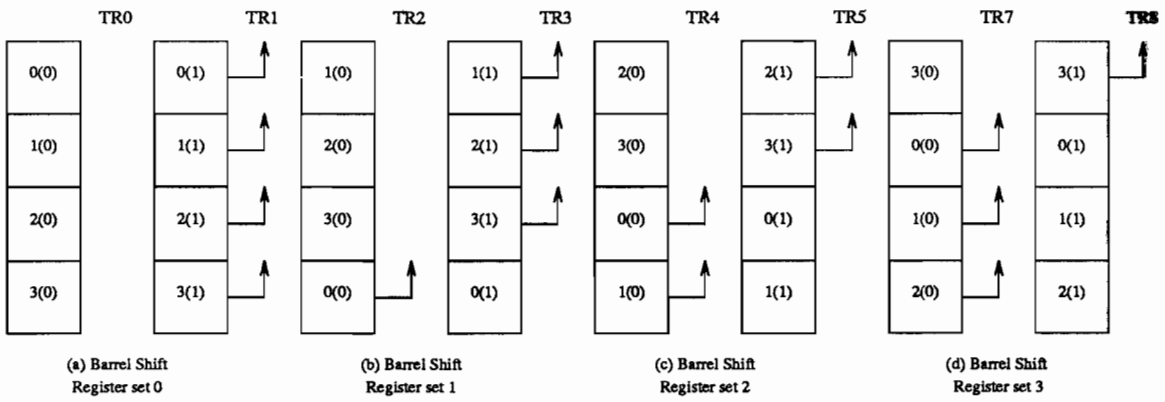


Figure 4-24: Input byte ordering for the barrel shift registers.

1 is shown in Figure 4-24(b). To form an output word out of this register, bytes 1, 2, and 3 are latched out of register 1 and byte 0 is latched out of register 0. Note that the word in register 1 was latched one clock cycle before the word in register 0. Hence, the output word ordering is actually 0, 1, 2 and 3, which is the right order.

Output will continue to be generated from barrel shift register set 1 until the next HEC byte or path overhead byte is encountered. After discarding the HEC or path overhead, output will be read from barrel shift register set 2. To form an output word out of this register, bytes 2 and 3 are latched out of register 1 and bytes 0 and 1 are latched out of register 0.

Barrel shift register set 2 will be used to form the output if the HEC byte was removed from position 4-22(b) and barrel shift register set 3 will be used if the HEC byte was removed from position 4-22(c). If the HEC byte was removed from position 4-22(d), then register set 0 will be used to output the data for the next cell. After discarding four bytes, which could be three HECs and a path overhead byte or four HEC bytes, one output write operation will be skipped.

Discarding the path overhead byte is similar to discarding the HEC byte. The path overhead byte will always be present in the least significant byte position, that is as in Figure 4-22(d).

Since the HEC can be latched in any of the four positions, the comparison logic

is present in all of the four positions and the computed HEC value is compared with all four bytes. When the HEC is present in one of the barrel shift registers, it is copied to the HEC latch and compare register.

In addition to delineation and header verification, idle cell detection and descrambling must be performed. An idle cell detection circuit is implemented using one column. The ATM self synchronous descrambler is implemented in the second delineator chip.

Apart from performing ATM cell delineation, the state machine for this design will keep track of the position of next HEC byte in the input data stream and increment its position count by one if a path overhead byte is encountered during an ATM cell. The state machine also enables the tri-state buffers selectively to form valid 32 bit words at the output. The state machine inputs are listed below.

- PathOvh: Indicates that the path overhead byte is at the input of 0(0) byte in Figure 4-24.
- PathOvhD1: This is the PathOvh signal delayed by one word clock, that is 51.44 ns.
- Count: The state of the 0-12 counter is used to track the position and the presence of HEC byte in the input stream.

The state transition diagram for the state machine is given in Figure 4-25. As shown in the figure, there are two finite state machines. One of them operates when the receiver is in *Hunt* state and the other state machine operates when the receiver is in *PreSync* or *Sync* states. When the receiver is in *Hunt* state if the computed HEC matches the received HEC, the receiver gets into *PreSync* state. When the receiver is in this state, a HEC mis-match would cause the receiver to get back into *Hunt* state. When the receiver is in *Sync* state, it will get back into *Hunt* state if there are six consecutive HEC mis-matches.

When the receiver is in *State0*, HEC is expected in position (a) in Figure 4-22. HEC is always expected in this position when the receiver is in *Hunt*. Hence, when the receiver transits from *Hunt* state to *PreSync* state, it gets into *State1* in Figure 4-25.

The outputs of the state machine are the *TRx* signals used to enable the appropriate tristate buffers to form the output. The state machine also generates a signal called *Disable_Count* which is used to disable the clock to the data output stage. This signal is used to generate ATM_DataOutClk_H, which is used in Chip2 as the word clock for all the input operations.

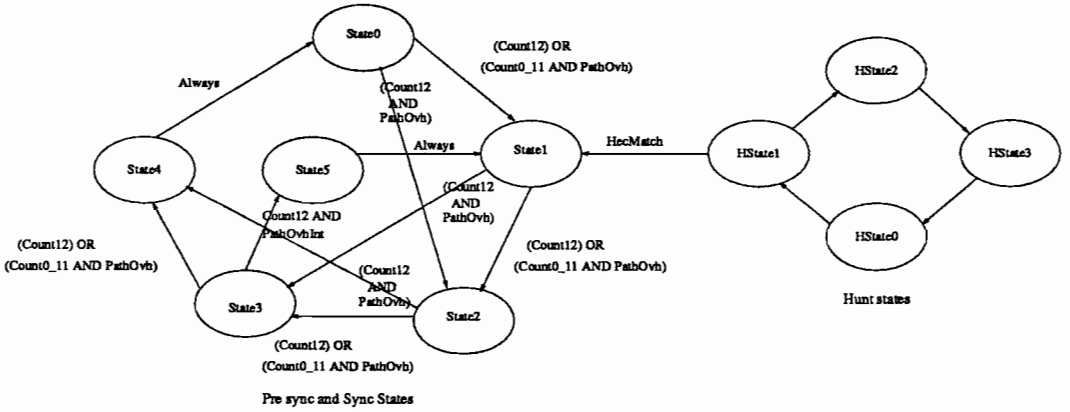


Figure 4-25: STS-12c ATM Receiver - Chip1 state transition diagram.

A VHDL description of the state machine is given in Appendix G. This state machine description can be used along with the state diagram to decode the state machine. A timing diagram illustrating the control signals in STS-12c design is shown in Figure 4-26.

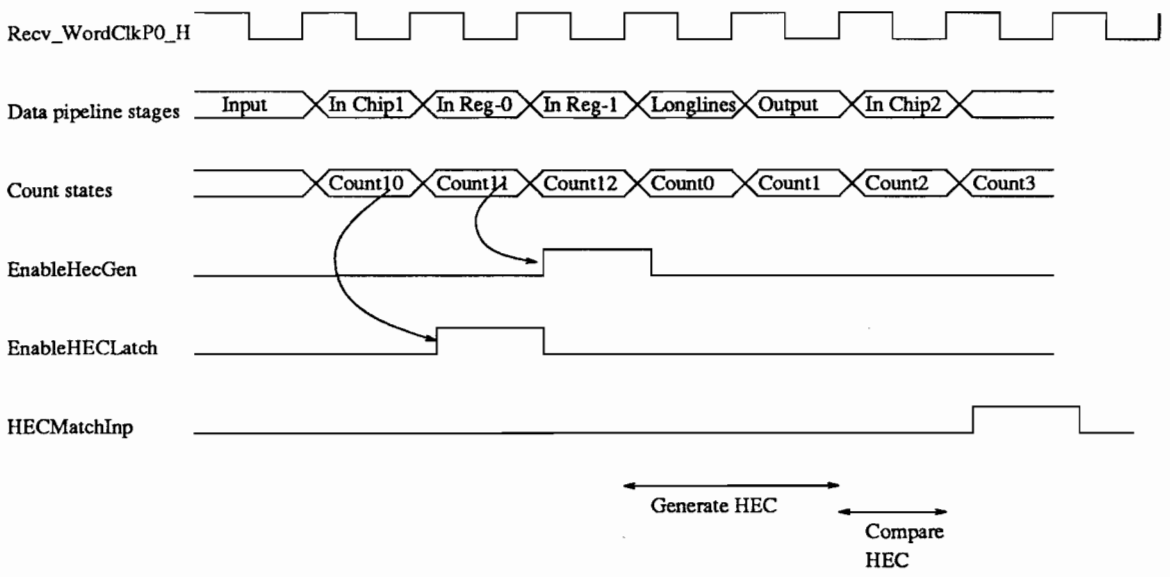


Figure 4-26: STS-12c ATM Receiver - Chip1 control signal timings.

4.5 Summary

The STS-12c and the 4 x STS-3c transmitters have been verified by simulating the VHDL model using Mentor Graphics tools. The 4 x STS transmitter and the STS-12c design that uses 51.44 ns clock have also been verified using the VHDL generated from Modula-3 code. The delays are less than 46 ns for the 51.44 ns clock designs and are less than 12.0 ns for the 12.8 ns clock designs.

Chapter 5

Performance Scalability in ATM Networks

In this chapter, ATM functions in the B-ISDN framework are explored in order to identify the potential bottlenecks in designing systems based on SONET and ATM. Dependency relationships are used to illustrate and analyze the designs under mild assumptions and for varying degrees of bit-level parallelism. First the data flow of the receiver is analyzed to identify the critical delay path in that subsystem. Next, the transmitter is analyzed to find its critical delay paths. The design considerations imposed by SONET framing are then discussed. The next section analyzes the data flow to implement AAL-5, the preferred ATM adaptation layer for data networks. Finally, the results derived in the earlier sections are analyzed to establish an overall limit on the performance of these systems.

5.1 Analysis of the ATM Receiver

When the raw data stream from a physical layer such as SONET is received, it is processed to perform the following functions before further processing by higher layers:

- ATM cell delineation is performed to separate and store ATM cells,
- the receiver descrambles its input payload to recover the original data,
- idle cells are detected and discarded, and
- cells with the incorrect HEC are discarded.

The dependency relationships for the various components of an ATM system will be expressed in a series of informal graphs. These graphs do not depict implementation details, only dependency relationships. In order to study the ultimate bounds on performance, high degrees of parallelism will be investigated, although such implementations will almost certainly not be practical with the technology of the foreseeable future. It will be assumed, except where noted otherwise, that only one instance of each of the major functional blocks will be used; those cases where relaxing this constraint can be used to improve performance will be specifically noted.

Since the processing performed by the receiver on the data stream differs significantly depending on the synchronization state of the receiver, the dependency relationships in different states will be studied separately. Cell synchronization, also known as delineation, is achieved by using the HEC (Header Error Check) field in the ATM cell header to determine if synchronization has been attained [1]. Initially, in the *Hunt* state, the delineation unit checks each input stream, byte by byte, to see if the HEC of the first four bytes of the assumed header field matches the fifth byte. If it does, the process passes to the *Pre-sync* state. When the system is in this phase, a single HEC mismatch will cause the state machine to go back into *Hunt* state. In *Pre-sync*, the unit checks for valid headers after every 53 bytes. If HEC matches six consecutive times, the unit assumes that it is synchronized with the received stream and thus the state machine moves to the *Sync* state. The delineation unit then checks every 53 bytes to see if synchronization is maintained. If it encounters seven consecutive cells for which the HECs

do not match, it re-initializes itself and begins to hunt for synchronization once again.

When the receiver is in *Hunt* state, it scans the input data to find the beginning of an ATM cell. This is shown in the dependency relationships in Figure 5-1. If the HEC computed over the first four bytes matches the fifth byte, the state machine assumes that it is the beginning of a valid ATM cell and the state of the system goes to *Pre-sync*. The HEC calculation logic calculates the HEC from the input data. The HEC comparison logic compares the calculated value of the HEC to the received HEC value. The result of this comparison is used by the state machine to determine the next state. Although it is recommended by the standards that each iteration in this figure should take one byte time, each iteration can take one byte, word, or an entire cell time, as long as all the bytes of the input stream are scanned and synchronization is attained within the time required by the standards [1, 3]. Under these assumptions, it will become apparent that the *Hunt* state will not impose the critical limits on performance.

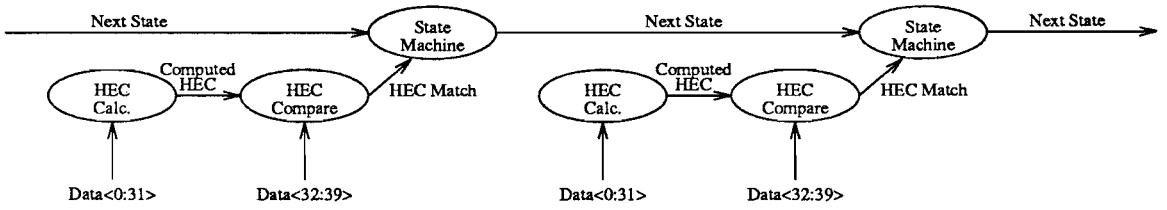


Figure 5-1: Dependency relationships for the receiver in Hunt state.

The dependency relationships for the receiver when it is in the *Pre-sync* and *Sync* states are shown in Figure 5-2. The state machine will generate control signals for the block that delineates or separates the header, HEC, and payload. The next state of the state machine will depend on the result of the HEC calculation as discussed earlier. The header data is input to the idle cell detection, the HEC calculation logic and to the memory write logic. The payload data is input to the descrambler. The unscrambled data out of the descrambler is input to the memory write logic. When the state of the system is in *Pre-sync*, a HEC

mismatch occurs, or when the input cell is an idle cell, the data is not written to memory.

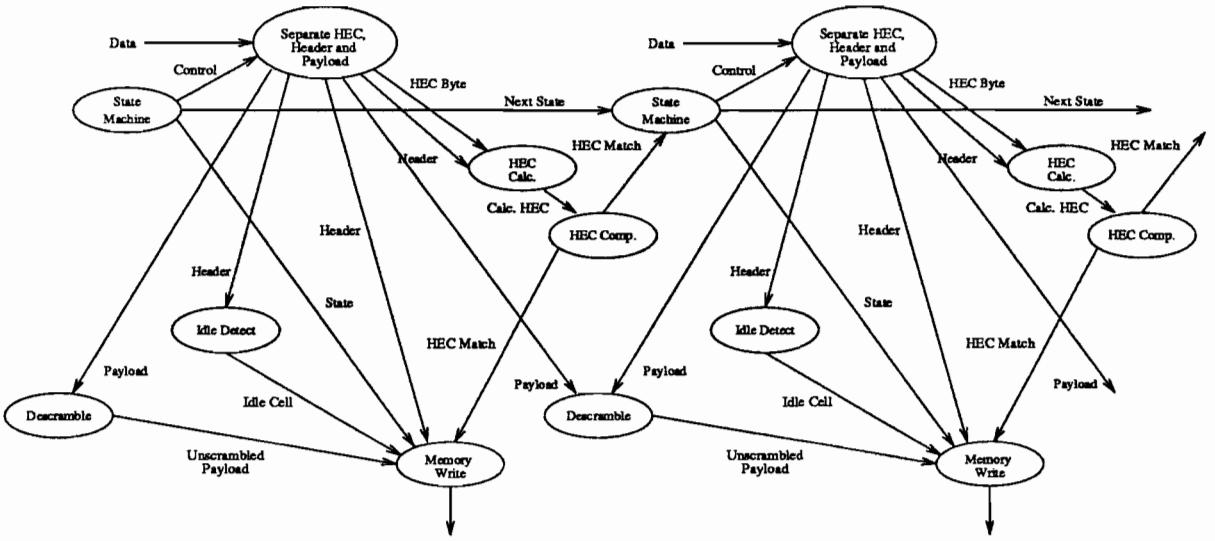


Figure 5-2: Dependency relationships for the receiver in Pre-sync or Sync states.

5.1.1 Constraints

The HEC calculation and HEC comparison, and self-synchronous payload descrambler, are critical elements in the ATM cell processing at the receiver.

The critical paths for the receiver will involve the cell delineation functions, since these operations must be completed prior to the processing of the next cell in order to verify header correctness and indicate these results to the state machine, which must be updated on a per cell basis as required by the standards [1, 3]. In addition, since the inputs to the descrambler function cross cell boundaries, the delineation function for the previous cell must be complete prior to the beginning of descrambling of the current cell to prevent invalid state data from being used in the descrambling function. Specifically, the critical path for the receiver will then be the path from the state machine, through the header/payload separation, HEC calculation and HEC compare, and back to the state machine.

While the speed of memory operations may be critical if a serial approach to writing cells is used, any restrictions of this sort can be eliminated by using multiple paths with interleaved access, so this function will not, in general, be constraining. Similar comments may be applied to the idle cell detection function. If only one path to memory is used, however, as would be common in the current technology, the descrambler block may limit the throughput.

In the next three subsections, the ATM cell delineation (including HEC) block, the descrambler, and the idle cell detection block are analyzed in detail to quantify the performance of these elements.

5.1.2 Analysis of ATM Cell Delineation

The constraints imposed by the state machine and descrambler imply that a critical path will be through the HEC logic. Figure 5-3 shows the dependency relationships for ATM cell delineation.

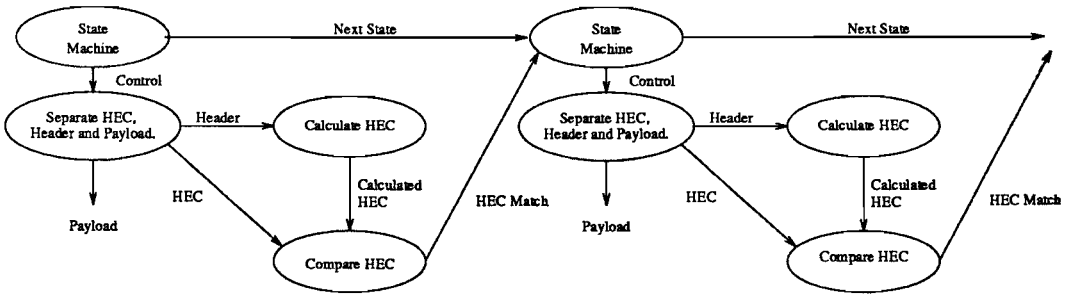


Figure 5-3: Dependency relationships for critical sections of the delineation path.

Several definitions are used in this and subsequent analyses. The time taken to compute the HEC will be $T_{HEC}(k)$, which is a function of the input data word parallelism, k bits. The time to compare the computed value of the HEC with the input byte is T_{Comp}. The time required to compute the next state will be denoted by T_{State}. The time taken to separate the header, HEC, and payload is $T_{Separate}$. For any particular technology, the combined register setup, hold and propagation delay will be T_{Reg}, and the average single gate delay will be T_{Gate}. The time taken

to distribute signals will be denoted by T_{Route}. Finally, the input/output data word clock period will be T_{Clk}.

The HEC is computed using the generator polynomial

$$1 + x + x^2 + x^8. \tag{5.1}$$

Using methods similar to those in [20], the number of XOR terms required to implement the HEC calculation for a given parallelism was calculated. Figure 5-4 shows the maximum number of XOR terms for HEC computation using different degrees of parallelism. If a 32 bit parallel design is used to compute the HEC, then a maximum of 20 XOR terms are present in any equation. If two input XOR gates are used, the operations can be performed using an XOR tree that is $\lceil log_2 20 \rceil = 5$ deep. This assumes a tree structure for the implementation, which is the most efficient structure. The restriction to two-input gates is reasonable, as this is often imposed by implementation technologies such as standard cell libraries.

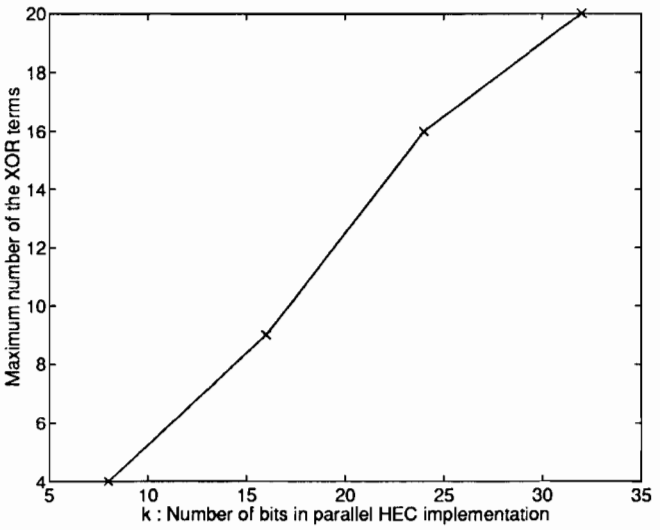


Figure 5-4: Depth of HEC XOR array versus parallelism.

Two cases need to be considered, that is, input word sizes larger than 32 bits and less than 32 bits, since the requirement for feedback in the HEC calculation

changes at this boundary. If an 8, 16, or 24 bit parallel design is used, for example, then the HEC calculation will require more than one iteration, since the HEC is computed over 32 bits of the header. Hence, the intermediate HEC values will have to be saved in a register and used during the next clock cycle. This implies the following bound on the HEC computation time,

$$T_{HEC}(k) = \{T_{XOR}(k) + T_{Reg} + T_{Route}\}\lceil\frac{32}{k}\rceil, \tag{5.2}$$

where $T_{XOR}(k)$ is the maximum delay through the HEC XOR tree for the given level of parallelism. In this case, the word clock is constrained to be

$$T_{Clk} \geq T_{XOR}(k) + T_{Reg} + T_{Route}. \tag{5.3}$$

If a parallel design of 32 bits or greater is used, the HEC computation time using a combinational logic tree implementation will be

$$T_{HEC}(k) = T_{XOR}(k) + T_{Reg} + T_{Route}. \tag{5.4}$$

The word clock constraint in this case is the same as that in Equation (5.3).

The HEC logic computation time, $T_{XOR}(k)$, is constant for $k \geq 32$, since this computation reduces to a 32-input, 8-output combinational logic function for such cases. Since no feedback is involved in the HEC calculation for $k \geq 32$, this computation can be pipelined at the individual gate level to provide further increases in clock rate. If such a strategy is pursued, the clock is constrained by

$$T_{Clk} \geq T_{Gate} + T_{Reg} + T_{Route}, \tag{5.5}$$

where

$$T_{HEC}(k) = 5T_{Clk}. \tag{5.6}$$

The Equations 5.5 and 5.6 can be extended to derive $T_{HEC}(k)$ using 2 or 3 stage gate level pipelining also. In order to meet the restrictions imposed by the state machine and the descrambler input sequencing, that is, the requirement that the indicated computations be completed within one cell period, the bound on the word clock for a given level of parallelism is

$$T_{HEC}(k) + T_{Comp} + T_{State} + T_{Separate} \leq \frac{424 \text{ bits/cell}}{\text{k bits/word}} T_{Clk} \tag{5.7}$$

for $0 \leq k \leq 424$.

A 32-bit parallel HEC calculation circuit was implemented in the $2\mu m$ SCMOS MOSIS process [15] using the MSU standard cells [18]. It was used to estimate the various parameters in the above equations. The maximum delay through any combinational path in the HEC calculation was found to be 23 ns. This is the delay through the largest XOR tree in the design, which is five gates deep. From the maximum net length in the design, the distributed RC delay was computed [13] and found to be less than 1 ns.Hence, for these calculations, the routing delays can be neglected, at least in this technology.

Table 5.1 shows $T_{HEC}(k)$ and the throughput k/T_{Clk} for parallel HEC designs in the MOSIS $2\mu m$ CMOS process using the MSU standard cells. The values for the 8, 16 and 24 bit designs were calculated using Equations (5.2), (5.3), and (5.7). The corresponding values for the larger designs were calculated using Equations (5.3), (5.4), and (5.7) or (5.5), (5.6), and (5.7), whichever set provides higher performance. T_{Comp}, the time taken to get the result of comparison is assumed to be two gate delays since it can be done by performing an XOR and an AND. T_{State} is also two gate delays, since the state machine is normally implemented as either *sum of products* or a *product of sum* forms, and it takes atleast two gate delays in either case. $T_{Separate}$ is assumed to be a single gate delay.

k	XOR tree depth	T_{XOR} (ns)	T_{Reg} (ns)	T_{Gate} (ns)	T_{Clk} (ns)	Throughput (Mb/s)
8	2	9.5	13	4.5	22.5	355
16	4	19	13	4.5	32	500
24	4	19	13	4.5	32	750
32	5	23	13	4.5	17.5	1829
64	5	23	13	4.5	17.5	3657
128	5	23	13	4.5	26	4923
256	5	23	13	4.5	36.5	7013
424	5	23	13	4.5	59	7186

Table 5.1: HEC throughput for different degrees of parallelism.

5.1.3 Analysis of Idle Cell Detection Logic

This block is used to detect idle cells in the input data stream. Idle cells are detected so that they need not be stored in the memory for further processing. Idle cell detection should be completed before the end of one cell period if one functional block is used. The dependency relationship for idle cell detection logic is shown in Figure 5-5. The header and payload are separated and only the header is input to the idle cell detection logic. The output of the idle cell detection block controls memory write logic. If the input is an idle cell, then that cell will not be written to memory.

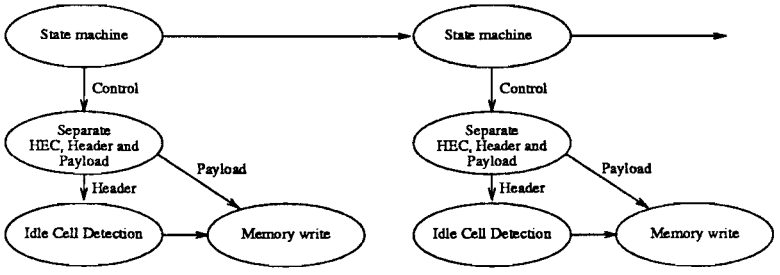


Figure 5-5: Dependency relationships for idle cell detection datapath.

Idle cell detection should be completed before a memory write is performed, so that idle cells are not written to the memory. If a single idle cell detection circuit

is used, then idle cell detection should be completed before the next cell arrival. The constraint due to the idle detection circuitry is thus

$$T_{idle\_det} \leq \frac{424 \text{ bits/cell}}{\text{k bits/word}} T_{Clk}, \tag{5.8}$$

where T_{idle_det} is the time taken to detect an idle cell, given the appropriate header bits. Typically T_{idle_det} is a few gate delays. By increasing the number of idle cell detection circuits, this constraint can be overcome.

5.1.4 Analysis of ATM Descrambler

ATM self-synchronous descrambling is performed prior to the storage of the cell payloads. The descrambled data is generated according to the polynomial [1]

$$x^{43} + 1, \tag{5.9}$$

as shown in Figure 5-6. This figure implies that the 43rd previous input bit is XORed with the present input bit to obtain the actual data. Hence, for any parallelism, data from the previous clock cycle is required.

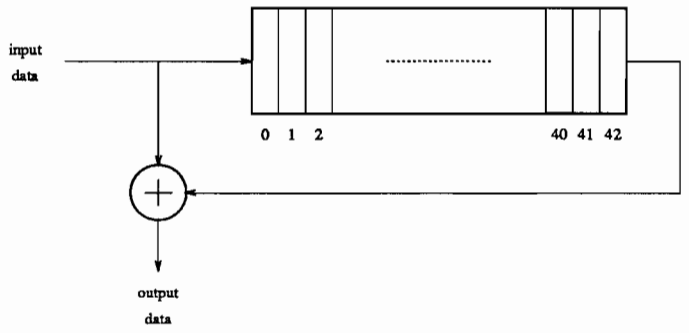


Figure 5-6: Self-synchronous descrambler (serial implementation).

If less than 43 bits are descrambled in parallel, the state information for at least a part of the present output will be the bits stored during previous clock periods. Hence, the state information should be stored in the register and routed to the

k	T_{Gate} (ns)	T_{Reg} (ns)	$T_{Descramble}$ (ns)	Throughput (Mb/s)
8	4.5	13	17.5	457
16	4.5	13	17.5	910
32	4.5	13	17.5	1828
64	4.5	13	17.5	3657
128	4.5	13	17.5	7314
256	4.5	13	17.5	14628
424	4.5	13	17.5	24228

Table 5.2: Descrambler throughput for different degrees of parallelism.

appropriate location before the beginning of the next clock. The computation time will thus be given by

$$T_{Descramble} = T_{Gate} + T_{Reg} + T_{Route}, \tag{5.10}$$

where T_{Gate} is the time taken to calculate the XOR of two bits.

If greater than 43 bits are descrambled, the state information for the first set of 43 bits will be obtained from the data stored in the registers from the previous clock cycle and the state information for all the other sets of 43 bits will be the data from the previous set of 43 bits. Since we are assuming routing delays are very small compared to T_{Reg}, the computation time will be the same as in Equation (5.10).

Table 5.2 shows the throughput for different values of parallelism that can be achieved with the $2\mu m$ MOSIS SCMOS process, where it is assumed that only one descrambler unit is being used.

5.1.5 Results for the ATM Receiver

The results of the analysis for the ATM receiver are presented in Table 5.3. The table also shows the nearest major STS signal that can be processed using this

k	Throughput of the descrambler (Mb/s)	Throughput of the HEC circuit (Mb/s)	Overall throughput (Mb/s)	STS signal
8	457	355	355	STS-3c
16	914	500	500	STS-3c
32	1828	1829	1828	STS-12c
64	3657	3657	3657	STS-48c
128	7314	4923	4923	STS-96c
256	14628	7013	7013	STS-96c
424	24228	7186	7186	STS-96c

Table 5.3: Overall throughput of the receiver.

technology (note that some of these STS signals, such as STS-48c, are not currently standardized). It can be seen from this table that the HEC circuit limits the overall throughput for all degrees of parallelism.

Using first order scaling [13], the overall throughput was derived for values of λ, the minimum feature size for a technology, where λ ranged from $2.0\mu m$ to $0.2\mu m$. The results are summarized in the graph shown in Figure 5-7. A particularly interesting trend is that the throughput appears to be more heavily dependent on parallelism than feature size.

5.2 Analysis of the ATM Transmitter

The transmitter performs the following functions to transmit valid ATM cells in SONET frames:

- ATM cells are formed from the header and payload data and are arranged in a SONET frame,
- the payload is scrambled before transmitting,
- idle cells are inserted when actual data is not available for transmission.

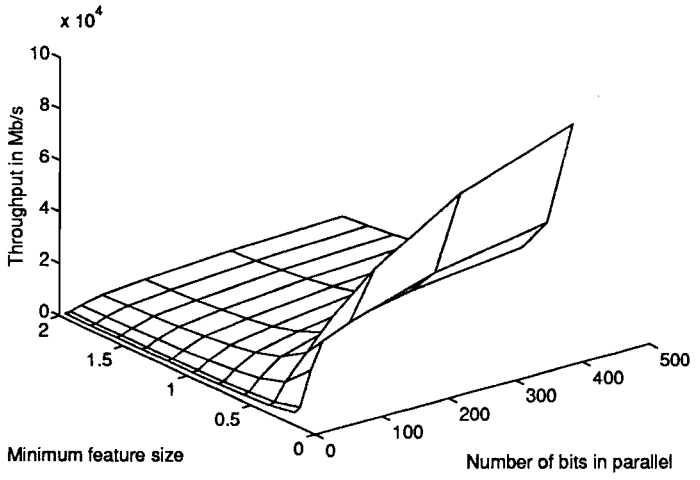


Figure 5-7: Throughput dependence on technology and parallelism for the receiver.

The dependency relationships for the transmitter are illustrated in Figure 5-8. The payload and the header are read in from the memory. The state machine enables the HEC calculation logic when header data is at its input. The state machine disables the scrambler when its input has header data. Idle cell generation is enabled when actual data is not available for transmission. The memory read logic indicates that no data is available for transmission to the idle cell generation block. The cell construction multiplexer constructs the ATM cell by multiplexing the header, HEC byte, scrambled payload, and the idle cell header.

The analysis of the HEC calculation for the transmitter is different from the analysis of the HEC in the ATM delineation path of the receiver because there is no constraint to complete the HEC calculation within one clock period in the transmitter. Table 5.4 shows $T_{HEC}(k)$ and the throughput k/T_{Clk} for parallel HEC designs in the MOSIS $2\mu m$ CMOS process using the MSU standard cells. The values for the 8, 16 and 24 bit designs were calculated using Equations (5.2) and (5.3). The corresponding values for the larger designs were calculated using Equations (5.3) and (5.4) or (5.5) and (5.6), whichever set provides higher

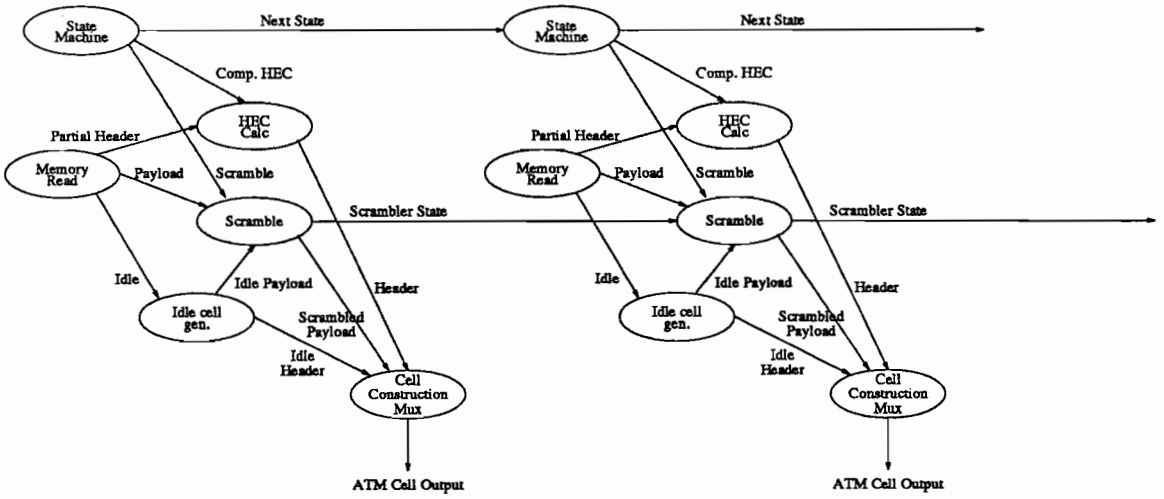


Figure 5-8: Dependency relationships for the transmitter.

performance.

The idle cell generation is similar to idle cell detection. The ATM self synchronous scrambler is significantly different from the descrambler, and in fact imposes the critical throughput constraint on the ATM receiver/transmitter pair.

5.2.1 Analysis of ATM Scrambler

ATM self-synchronous scrambling is performed on the cell payload prior to transmission, using the polynomial [1]:

$$x^{43} + 1, \tag{5.11}$$

as shown in Figure 5-9. This figure implies that the 43rd state bit is XORed with the present bit to produce scrambled data. The scrambler is different from the descrambler in the sense that the state bit is derived by applying the above equation to the previously computed outputs of the scrambler, whereas for the descrambler, the state bit is the input bit itself.

Because this structure is similar to a CRC calculation, the methods in [20] can be applied to determine the complexity of logic for any degree of parallelism k.

k	XOR tree depth	T_{XOR} (ns)	T_{Reg} (ns)	T_{Gate} (ns)	T_{Clk} (ns)	Throughput (Mb/s)
8	2	9.5	13	4.5	22.5	355
16	4	19	13	4.5	32	500
24	4	19	13	4.5	32	750
32	5	23	13	4.5	17.5	1829
64	5	23	13	4.5	17.5	3657
128	5	23	13	4.5	17.5	7314
256	5	23	13	4.5	17.5	14628
424	5	23	13	4.5	17.5	24228

Table 5.4: HEC throughput for different degrees of parallelism for the transmitter.

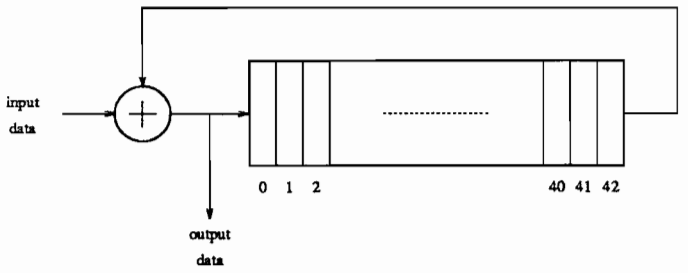


Figure 5-9: Self-synchronous scrambler (serial implementation).

k	T_{XOR} (ns)	T_{Reg} (ns)	T_{Clk} (ns)	Throughput (Mb/s)
8	4.5	13	17.5	457
16	4.5	13	17.5	914
32	4.5	13	17.5	1828
64	9	13	22	2909
128	13.5	13	26.5	5818
256	27	13	40	9660
424	45	13	58	13677

Table 5.5: Scrambler throughput for different degrees of parallelism.

The bound on the clock period can thus be found to be given by

$$T_{Clk} \geq T_{Reg} + T_{Route} + log_2(\frac{k}{43})T_{Gate}. \tag{5.12}$$

As parallelism increases, the routing delay factor in this equation increases in a non-linear fashion and will become significant at high degrees of parallelism.

Table 5.5 shows the throughput that can be achieved in the $2\mu m$ MOSIS SCMOS process.

5.2.2 Results for the ATM Transmitter

The results of the analysis for the ATM transmitter are presented in Table 5.6. It can be seen from this table that for 8 and 16 bit parallel designs, the HEC circuit limits the overall throughput, whereas for 32 bits and higher levels of parallelism, the scrambler circuit limits the overall throughput.

Using first order scaling [13], the overall throughput limits imposed by the HEC and scrambler circuits were derived for various values of λ, the minimum feature size for a technology. The results are summarized in the graph shown in Figure 5-10.

k	Throughput of the scrambler (Mb/s)	Throughput of the HEC circuit (Mb/s)	Overall throughput (Mb/s)	STS Signal
8	457	355	355	STS-3c
16	914	500	500	STS-3c
32	1828	1829	1828	STS-12c
64	2909	3657	2909	STS-48c
128	5818	7314	5818	STS-96c
256	9660	14628	9660	STS-96c
424	13677	24228	13677	STS-96c

Table 5.6: Overall throughput of the transmitter.

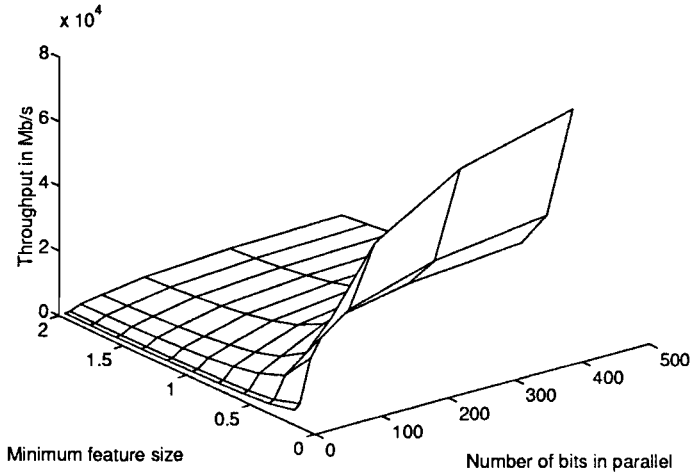


Figure 5-10: Throughput dependence on technology and parallelism for the transmitter.

5.3 SONET Considerations

SONET forms the physical layer of B-ISDN [22]. In a SONET frame, the payload can drift with respect to the frame itself. This capability is provided so that the receiver and the transmitter need not be in perfect synchronization. Because the position of the payload within the frame cannot be determined precisely from frame to frame, there are potential implications for the operation of systems which insert and retrieve ATM cells from the SONET payload.

The position of the path overhead byte is defined as the first byte in the synchronous payload envelope (SPE). Hence, to delineate ATM cells that are part of the SONET frame, it is necessary to separate the path overhead byte from the rest of the payload. In steady state, the path overhead byte always occurs in the same position within a frame. But since the synchronous payload envelope can drift within the SONET frame, the position of the path overhead byte will change. There may be an invalid byte in the data stream when the SPE, and hence the path overhead byte, drift forward relative to the SONET frame. This byte, as well as the overhead byte itself, must be skipped to maintain ATM synchronization. There is also a possibility that the SPE will drift in a reverse manner, in which case the path overhead is present in the stuff byte, which is a location reserved in the SONET overhead for reverse drift of the SPE within the SONET frame. The byte following the stuff byte location must be recovered in order to maintain ATM cell synchronization in this case.

The presence of a valid stuff byte, also known as the H3 byte, is indicated by the values in the H1 and H2 bytes [22] of the SONET overhead. In the worst case, the path overhead byte could be in the stuff byte of the SONET overhead, in which case the time available to indicate the path overhead is the time between the H2 and H3 bytes.

Normally, the input to the ATM receiver chip is masked during the SONET section, line, and path overhead periods. In order to indicate the presence of a

valid SPE word in the stuff byte, the SONET section and line termination unit should unmask the input to the ATM receiver chip and, if the path overhead is present in the stuff byte, the appropriate control signal should be asserted. The time available for this control signal to reach the ATM receiver with respect to the actual data containing the path overhead is critical, since the receiver should discard the path overhead byte before ATM processing.

In order to resolve this problem, the data can be pipelined to allow processing of the H1 and H2 bytes before the data is forwarded to the ATM receiver. The detection and elimination of the various SONET overhead components from the data stream should not, therefore, impose a critical performance constraint on ATM/SONET receivers. The insertion of ATM cells into a locally-generated SONET frame at the transmitter should not impose critical limitations.

Other SONET-related functionality, in particular processing of overhead information such as the byte interleaved parity codes, may in practice limit the performance of that physical layer. These issues are, however, beyond the scope of this appraisal; further, the current capabilities of SONET integrated circuits appear to indicate that the SONET functions will not impose the critical constraints in B-ISDN.

5.4 AAL-5 Considerations

AAL-5 is a simple and efficient ATM adaptation layer [21], recommended primarily for data communication networks. It has gained wide acceptance, as indicated by its use in many recently developed ATM local area network products.

The maximum size of an AAL-5 protocol data unit (PDU) is 65536 bytes. The AAL-5 packet is segmented into ATM cells for transmission, and reassembled upon receipt. All of the ATM cells other than the last cell of the PDU contain 48 bytes of the AAL-5 PDU data in the payload portion of the ATM cell. The format

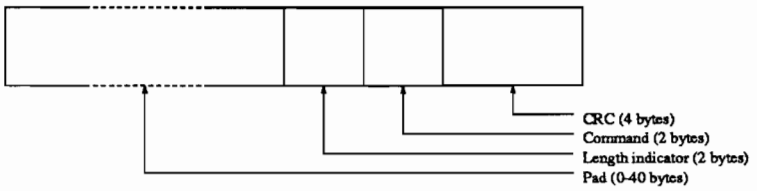


Figure 5-11: Last ATM cell in an AAL-5 PDU.

of the last ATM cell is as shown in Figure 5-11. The length indicator indicates the length of the AAL-5 packet in bytes. Padding is used to fill the last ATM cell so that the AAL-5 trailer is aligned to the end of the cell.

Since the delineation of the end of the AAL-5 PDU and the insertion or recovery of the trailer are straightforward, these functions should not limit the performance of AAL-5 transmitter or receiver units. The only other problematic function might be the length counter, which is used to recover the payload. If the circuitry is byte oriented, this function reduces to count and compare operations. If wider word sizes are used (more parallelism), more complex operations may be required. In any case, the throughput constraints due to the payload length processing circuitry are unlikely to compare to those from the CRC calculation, as will be seen.

In the following discussion, the implementation of the CRC circuit for AAL-5 is analyzed to find the maximum throughput that can be achieved. Given the relative simplicity of the other elements of this AAL, generating the CRC is going to be the limiting factor in implementing AAL-5.

The AAL-5 CRC is generated over the entire AAL-5 PDU. The AAL-5 CRC is computed using the generator polynomial

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

Since the maximum size of an AAL-5 packet is 64 kilobytes, the entire CRC cannot be calculated with a single parallel circuit. Hence, for all implementations

k	XOR tree depth	T_{XOR} (ns)	T_{Reg} (ns)	T_{Clk} (ns)	Throughput (Mb/s)	STS Signal
8	3	13	13	26	307	STS-3c
16	4	19	13	32	500	STS-3c
32	5	23	13	36	888	STS-12c
64	6	28	13	41	1560	STS-24c
96	6	28	13	41	2341	STS-24c
128	7	33	13	46	2782	STS-48c
256	7	33	13	46	5564	STS-96c
424	7	33	13	46	9217	STS-96c

Table 5.7: Throughput for different degrees of parallelism in AAL-5 CRC.

of the CRC circuit, feedback is necessary. For a parallel implementation of CRC calculation, the constraint on the clock period will be given by [20]

$$T_{Clk} \geq T_{XOR}(k) + T_{Reg} + T_{Routing}, \tag{5.13}$$

where the same notation as defined earlier is used, This equation implies that the XOR operations should be completed and the results latched before the beginning of the next clock period. The routing delays will again be assumed to be small, and so that term will be neglected.

The number of XOR terms required to implement the CRC calculation for a given level of parallelism was calculated and is displayed in Figure 5-12. The depth of the XOR tree is calculated from the maximum number of terms in the parallel CRC equations for that value of k. Table 5.7 shows the throughput for 8, 16, 24 and 32 bit parallel CRC designs implemented in the MOSIS $2\mu m$ SCMOS process using the MSU standard cells. The table also shows the STS signals that can be processed using this technology. The effects of first order scaling on these results are shown in Figure 5-13 for decreasing feature size when CMOS technology is used.

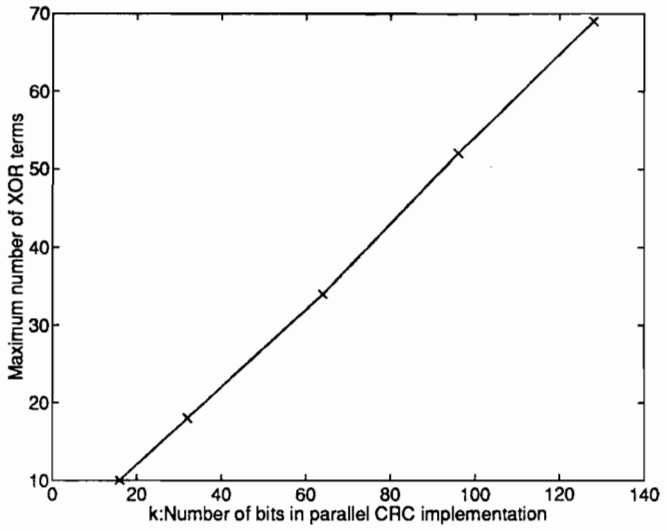


Figure 5-12: Number of XOR terms versus parallelism for AAL-5 CRC.

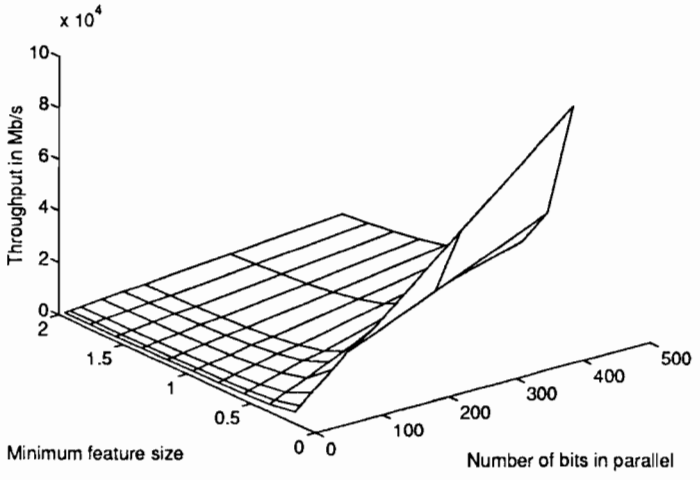


Figure 5-13: Throughput dependency on technology and parallelism for AAL-5 CRC.

k	Throughput of transmitter (Mb/s)	Throughput of receiver (Mb/s)	Throughput of AAL-5 (Mb/s)	Overall throughput (Mb/s)
8	355	355	307	307
16	500	500	500	500
32	1828	1829	888	888
64	2909	3657	1560	1560
128	5818	4923	2782	2782
256	9660	7013	5564	5564
424	13677	7186	9217	7186

Table 5.8: Overall throughput for different degrees of parallelism in ATM interfaces.

5.5 Conclusion and Discussion

The critical elements that determine the overall performance of the principal ATM interface portions of a B-ISDN network can be derived from the preceding analysis. The limits on the overall performance can be separated into two cases, the case in which the performance of functions above the ATM layer are not considered, and the case in which AAL-5 is used.

By comparing the results for the ATM transmitter and the receiver sections, it can be concluded that the transmitter determines the overall performance of the ATM layer alone. This bound is a result of HEC throughput limitations for small word size k, and scrambler throughput constraints for larger degrees of bit-level parallelism. If AAL-5 is used, the AAL-5 CRC computation circuit will determine the overall throughput of the entire system except for $k = 424$ (cell level parallelism), in which case the transmitter limits performance. The results for these cases are summarized in Table 5.8.

In the course of this analysis, a number of assumptions have been made which may not hold in practically realizable systems. For example, this analysis assumes that the required package input/output and interconnect data rates can be

achieved, though this may not always be the case, and will depend on the packaging technology used. In addition, due to chip area limitations, the amount of pipelining needed to attain the maximum throughput rates may not be realizable in current and near-term technologies. Finally, the difficulties in routing highly parallel designs were not considered in detail, and may prove intractable.

The results presented in this work provide an upper bound on the performance of ATM/SONET networks, although reaching these bounds with practical implementations will be extremely difficult. It has been shown that bit-level parallelism provides the key to high performance implementation of ATM networks. Moderate degrees of parallelism are currently being used in SONET/ATM systems in the MAGIC gigabit testbed [19], and thus the practicality of these techniques has been demonstrated.

Chapter 6

Conclusions

Architectures for performing the principal ATM receive and transmit functions in a SONET network were presented. The novel architectures presented here use the same chip set in the same system configuration to realize either an OC-12c network or four independent streams of OC-3c. The transmitter uses one Xilinx XC3195 chip and one Xilinx XC3195 chip. The receiver uses two Xilinx XC3195 chips. Although architectures based on custom ASICs have been presented previously to perform part of the 4 $\times$ OC-3c functions [8], this design supports both 4 $\times$ OC-3c and OC-12c modes. Further, implementation based on FPGAs allows rapid prototyping of systems while standards are still evolving, which is more difficult with the traditional custom ASIC approach.

The critical elements that determine the overall performance of the principal ATM interface portions of a B-ISDN network have been analyzed. The limits on the overall performance were separated into two cases, the case in which the performance of functions above the ATM layer are not considered, and the case in which AAL-5 is used. Each case was analyzed separately to get and upper bound on the performance of ATM/SONET networks.

Appendix A

OC-12c Transmitter

A.1 Behavioral model of OC-12c transmitter: Chip1

A.2 Behavioral model of OC-12c transmitter: Chip2

A.3 Simulation results for OC-12c transmitter

A.4 OC-12c Transmitter Chip1 FPGA Layout

A.5 OC-12c Transmitter Chip2 FPGA Layout

Appendix B

4 x OC-3c Transmitter

B.1 Behavioral model of 4xOC-3c transmitter: Chip1

B.2 Behavioral model of 4xOC-3c transmitter: Chip2

B.3 Simulation results for 4xOC-3c transmitter

B.4 4xOC-3c Transmitter Chip1 FPGA Layout

B.5 4xOC-3c Transmitter Chip2 FPGA Layout

Appendix C

OC-12c Transmitter and Receiver Simulation

Appendix D

4xOC-3c Transmitter and Receiver Simulation

Appendix E

OC-12c Receiver

E.1 Behavioral model of OC-12c receiver: Chip1

E.2 Schematics of OC-12c Receiver: Chip1

E.3 Behavioral model of OC-12c receiver: Chip2

E.4 Schematics of OC-12c Receiver: Chip2

E.5 OC-12c receiver Chip1 FPGA Layout

E.6 OC-12c receiver Chip2 FPGA Layout

Appendix F

4 x OC-3c Receiver

F.1 Behavioral model of 4xOC-3c receiver: Chip1

F.2 Schematics of 4xOC-3c Receiver: Chip1

F.3 Behavioral model of 4xOC-3c receiver: Chip2

F.4 Schematics of 4xOC-3c Receiver: Chip2

F.5 4xOC-3c receiver Chip1 FPGA Layout

F.6 4xOC-3c receiver Chip2 FPGA Layout

Appendix G

OC-12c Receiver using 51.44 ns clock

G.1 Simulation results of the receiver

G.2 FPGA layout of the OC-12c receiver: Chip1

G.3 State machine of the OC-12c receiver using 51.44 ns clock

```
------------------------------------------------------------
--
-- Acestes Project -- VHDL Source Code
--
------------------------------------------------------------

-- Define external libraries

LIBRARY mgc_portable, std, xilinx_lib;

USE mgc_portable.qsim_logic.all;
USE std.standard.all;

-- Define entity

ENTITY Delin1Stm IS
  PORT (
        Recv_WordClkPO_H: IN qsim_state;
        Count: IN qsim_state_vector (3 downto 0);
        PathOvh: IN qsim_state;
        DisableCount: OUT qsim_state;
        HECLatch: OUT qsim_state;
        HECPos: OUT qsim_state_vector (1 downto 0);
        PathOvhD1: IN qsim_state;
        TR10_2: OUT qsim_state;
        TR13: OUT qsim_state;
        TR2: OUT qsim_state;
        TR31_2: OUT qsim_state;
        TR3_3: OUT qsim_state;
        TR4: OUT qsim_state;
        TR52: OUT qsim_state;
        TR53: OUT qsim_state;
        TR7: OUT qsim_state;
        TR8: OUT qsim_state
        );

ARCHITECTURE behavior OF Delin1Stm IS

        SIGNAL  State0: qsim_state;
```

```
        SIGNAL  State1: qsim_state;
        SIGNAL  State2: qsim_state;
        SIGNAL  State3: qsim_state;
        SIGNAL  State4: qsim_state;
        SIGNAL  State5: qsim_state;
BEGIN
Process ( Recv_WordClkP0_H)

BEGIN

If ( find_state(Recv_WordClkP0_H) = 'R') then

        If (State0 = 1) and 0 <= Count <= 11 and Not(Path0vh) then

                TR10_2  <= '1';
                TR13    <= '1';
                State0  <= '1';
        End If;

        If (State0 = 1) and 0 <= Count <= 11 and Path0vh then

                TR10_2  <= '1';
                TR2     <= '1';
                State1  <= '1';
        End If;

        If (State0 = 1) and Count = 12 and Not(Path0vh) then

                State1  <= '1';
                TR2     <= '1';
                TR31_2  <= '1';
                TR3_3   <= '1';
        End If;

        If (State0 = 1) and Count = 12 and Path0vh then

                State2  <= '1';
                TR31_2  <= '1';
                TR4     <= '1';
        End If;


        If (State1 = 1) and 0 <= Count <= 11 and Not(Path0vh) then

                TR2     <= '1';
```

```
        TR31_2  <= '1';
        TR33    <= '1';
        State1  <= '1';
End If;

If (State1 = 1) and 0 <= Count <= 11 and PathOvh then

        TR31_2  <= '1';
        TR4     <= '1';
        State2  <= '1';
End If;

If (State1 = 1) and Count = 12 and Not(PathOvh) then

        State2  <= '1';
        TR4     <= '1';
        TR52    <= '1';
        TR53    <= '1';
End If;

If (State1 = 1) and Count = 12 and PathOvh then

        State3  <= '1';
        TR52    <= '1';
        TR7     <= '1';
End If;

If (State2 = 1) and 0 <= Count <= 11 and Not(PathOvh) then

        TR4     <= '1';
        TR52    <= '1';
        TR53    <= '1';
        State2  <= '1';
End If;

If (State2 = 1) and 0 <= Count <= 11 and PathOvh then

        TR52    <= '1';
        TR7     <= '1';
        State3  <= '1';
End If;

If (State2 = 1) and Count = 12 and Not(PathOvh) then

        State3  <= '1';
```

```
        TR7     <= '1';
        TR8     <= '1';
End If;

If (State2 = 1) and Count = 12 and PathOvh then

        State4  <= '1';
        DisableCount    <= '1';
End If;

If (State3 = 1) and 0 <= Count <= 11 and Not(PathOvh) then

        TR7     <= '1';
        TR8     <= '1';
        State3  <= '1';
End If;

If (State3 = 1) and 0 <= Count <= 11 and PathOvh then

        State4  <= '1';
        DisableCount    <= '1';
End If;

If (State3 = 1) and Count = 12 and Not(PathOvh) then

        State4  <= '1';
        DisableCount    <= '1';
End If;

If (State3 = 1) and Count = 12 and PathOvh then


        State5  <= '1';
        DisableCount    <= '1';
End If;

If (State4 = 1) then
        TR10_2  <= '1';
        TR13    <= '1';
        State0  <= '1';
End If;

If (State5 = 1) then
        TR31_2  <= '1';
        TR3_3   <= '1';
```

```
                TR2     <= '1';
                State1  <= '1';
        End If;
End If;

END behaviour;
```

Appendix H

Simulation results of the ATM receiver connected to Receive Buffer Controller

H.1 Simulation results for OC-12c mode

H.2 Simulation results for 4 x OC-3c mode

Bibliography

[1] ATM Forum. ATM User-Network Interface Specification Version 2.0, June 1, 1992.

[2] Gary J. Minden, Joseph B. Evans, David W. Petr, Victor S. Frost. An ATM WAN/LAN Gateway Architecture. Second International Symposium on High Performance Distributed Computing, 1993.

[3] 1990 CCITT Study Group XVIII Recommendation I.150. B-ISDN Asynchronous Transfer Mode Functional Characteristics. CCITT, Geneva, 1990.

[4] M. A. Khan and G. Delgrossi. Implementation of an ASIC to Interconnect SDH and ATM Networks. In *IEEE Custom IC Conf.*, May 1992.

[5] Bellcore Technical Reference TR-NWT-000253. Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria. Bellcore Technical Reference Issue 2, Bellcore, Dec 1991.

[6] Nim K. Cheung. The Infrastructure for Gigabit Computer Networks. *IEEE Communications Magazine,* April 1992.

[7] J. Bryan Lyles, Daniel C. Swinehart. The Emerging Gigabit Environment and the Role of Local ATM. *IEEE Communications Magazine,* April 1992.

[8] G. R. Lalk, B.S. Davie, W.S. Marcus and K.C. Young, Jr. OC-12/STS-3c/ATM Interface for Gigabit Network Applications *Electronics Letters* October 1992.

[9] Ben Lisowski, Louise Reingold. Sprint's Evolution to Broadband ISDN. *IEEE Communications Magazine,* August 1992.

[10] ANSI Committee T1 Contribution T1S1.5/91-449. AAL 5 – A New High Speed Data Transfer AAL. Bellcore Technical Reference Issue 2, IBM et al, Dallas, Texas, Nov 1991.

[11] Broadband ISDN Transport Network Elements Framework Generic Criteria, Bellcore Framework Technical Adv FA-NWT-001109, issue 1, Dec, 1990.

[12] Hiroshi Ishikawa. NTT's Architecture : Evolving from Narrowband. *IEEE Communications Magazine,* August 1992.

[13] Neil Weste, Kamran Eshraghian Principles of CMOS VLSI Design *Addison-Wesley Publishing Company,* June 1988.

[14] A. S. Tanenbaum Communication Networks *Prentice Hall,* 1992.

[15] Information Sciences Institute, University of Southern California, Los Angeles, California. *MOSIS Scalable & Generic CMOS Design Rules*, 1988.

[16] Institute for Technology Development *Advanced Microelectronics Division SCMOS Double Metal I/C Standard Cell Library*, Mar 1990.

[17] Tong-Bi Pei, Charles Zukowski High Speed Parallel CRC Circuits in VLSI *IEEE Transactions on Communications,* Vol 40, No. 4, April 1992.

[18] Institute for Technology Development - Advanced Microelectronics Division, Mississippi State University. *Scalable CMOS (SCMOS) Standard Cell Library: dlm V2.2*, March 1990.

[19] G. Minden, J. Evans, D. Petr, and V. Frost. An ATM WAN/LAN gateway architecture. In *2nd IEEE Symp. High Perf. Dist. Comp.*, July 1993.

[20] T.-B. Pei and C. Zukowski. High speed parallel CRC circuits in VLSI. *IEEE Trans. Comm.*, 40(4), Apr 1992.

[21] ANSI Committee T1 Contribution T1S1.5/91-449. AAL 5 – A New High Speed Data Transfer AAL. Bellcore Technical Reference Issue 2, IBM et al, Dallas, Texas, Nov 1991.

[22] Bellcore Technical Reference TR-NWT-000253. Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria. Bellcore Technical Reference Issue 2, Bellcore, Dec 1991.