FORTH-ICS / TR - 221 July 1998

Design and Implementation of a 1.2 Gbit/s ATM Cell Buffer using a Synchronous DRAM chip

George N. Glykopoulos

ABSTRACT

High speed networks require high throughput memories to store cells or packets. Synchronous Dynamic RAM (SDRAM) chips provide both high storage capacity and high throughput, and are thus an appropriate technology for building such cell or packet buffers. One 16Mbit SDRAM chip with a 16-bit data interface provides 40K cell storage and 1.2Gbit/s throughput when used as an ATM cell buffer. The purpose of this work is to demonstrate this capability in a working prototype, gaining familiarity with the practical details of SDRAM and SONET operation, and PCB implementation with 100MHz clocks. The prototype that was designed, manufactured and successfully tested, is an 8-layer PCB with two SONET OC-3 links (2 x 155.52Mbit/s incoming and 2 x 155.52Mbit/s outgoing throughput), interfaced to an SDRAM chip using two ALTERA FPGA's. The board also includes a microprocessor interface and PLL-driven clock generators/drivers.

Design and Implementation of a 1.2 Gbit/s ATM Cell Buffer using a Synchronous DRAM chip

George N. Glykopoulos

Computer Architecture and VLSI Systems Division (CARV)
Institute of Computer Science (ICS)
Foundation for Research & Technology - Hellas (FORTH)
P.O. Box 1385, Science and Technology Park of Crete,
Heraklion, Crete, GR-711-10 GREECE
Tel: +30 81 391660, Fax: +30 81 391661
email: glykop@ics.forth.gr

Technical Report FORTH-ICS/TR-221 July 1998

Work performed as M.Sc. Thesis at the Univ. of Crete

ABSTRACT

High speed networks require high throughput memories to store cells or packets. Synchronous Dynamic RAM (SDRAM) chips provide both high storage capacity and high throughput, and are thus an appropriate technology for building such cell or packet buffers. One 16Mbit SDRAM chip with a 16-bit data interface provides 40K cell storage and 1.2Gbit/s throughput when used as an ATM cell buffer. The purpose of this work is to demonstrate this capability in a working prototype, gaining familiarity with the practical details of SDRAM and SONET operation, and PCB implementation with 100MHz clocks. The prototype that was designed, manufactured and successfully tested, is an 8-layer PCB with two SONET OC-3 links (2 x 155.52Mbit/s incoming and 2 x 155.52Mbit/s outgoing throughput), interfaced to an SDRAM chip using two ALTERA FPGA's. The board also includes a microprocessor interface and PLL-driven clock generators/drivers.

This text is available in Postscript form, by anonymous ftp, from server "ftp.ics.forth.gr".
Directory: "tech-reports/1998".
Files: "1998.TR221.ATM_Cell_Buffer_using_SDRAM.README,
"1998.TR221.ATM_Cell_Buffer_using_SDRAM.ps.gz"

Contents

1 Introduction **1**

1.1 This thesis . . . 1

2 Board Structure and Operations **3**

3 Interfacing with SONET/SDH at 155.52Mbps **6**

3.1 The SONET and SDH standards . . . 6

3.2 Electro-optical converters . . . 8

3.3 SONET/SDH ATM 155.52Mbps transceiver . . . 8

4 Synchronous Dynamic RAM: Architecture and Operation **11**

4.1 Throughput of SDRAM . . . 14

5 The ATM Cell Scheduler **16**

5.1 Receive Side Interface . . . 16

5.2 Transmit Side Interface . . . 17

5.3 Interface to the SDRAM Controller . . . 18

5.4 Scheduler architecture . . . 18

5.5 Dropping ATM cells upon cell memory overflow . . . 20

5.6 Internal memory organization . . . 21

5.7 Elastic buffer at input/output links: worst case analysis . . . 23

6 The SDRAM Controller **27**

7 MicroProcessor Interface **29**

8 Board Implementation **31**

8.1 High speed design techniques at the board level . . . 31

8.2 Design flow . . . 35

8.3 PCB Characteristics . . . 36

9 Board Testing and Results **38**

10 Conclusions and future work **40**

11 Acknowledgements **41**

A Reference designators **43**

B Schematics of the printed circuit board **45**

C Board operation and programming manual **50**

C.1 Power supply issues . . . 50

C.2 Jumper positions . . . 50

C.3 FPGA programming 51
C.4 Microcontroller interface 52
C.5 Connectors 52
C.6 LEDs 55
C.7 IMPORTANT NOTE 55

D Board pictures **56**

E Bill of materials **58**

F Manufacturers **59**

G Distributors **61**

List of Figures

1 The ATM cell buffer/forwarder in abstract form 3
2 Overview of the prototype board 5
3 STS-1 frame structure 7
4 STS-3 frame structure 7
5 The SONET/SDH ATM transceiver functional block diagram 9
6 Connections between the electro-optical converter and the SONET/SDH ATM transceiver. 10
7 SDRAM architecture. 11
8 ATM cell read operation from SDRAM. 13
9 ATM cell write operation into SDRAM. 13
10 Throughput offered by an SDRAM chip with various organizations (16, 8 or 4 I/O pins) for various burst sizes (I stands for interleaved accesses, NonI stands for non-interleaved). 15
11 Receive cell interface between the transceiver and the scheduler. . . . 16
12 Transmit cell interface between the transceiver and the scheduler. . . 17
13 Interface between the scheduler and the SDRAM controller. 18
14 Logical queue organization of the scheduler. 19
15 ATM cell scheduler block diagram. 22
16 Cells are residing in both banks to reduce bank conflicts and increase throughput. 23
17 The first byte of a cell is always written at the first byte position of a register: A) start of a cell k, B) end of cell k and start of $k+1$, C) end of $k+1$ and start of $k+2$. 24
18 Cycle 0: all registers at input links are completely empty and all registers at output links are completely full. 24
19 Worst case: all links require the same byte at the same time. The highest demand for internal memory occurs when they reach end-of-packet simultaneously. 26
20 The block diagram of the controller. 27
21 Address is given to SDRAM directly from the scheduler 28
22 Communication with an off-board MicroProcessor. 29
23 Clock distribution network. 35
24 Top layer of the board: reference designators 45
25 Bottom layer of the board: reference designators 46
26 Top layer of the board: etch 47
27 Bottom layer of the board: etch 48
28 Internal layer of the board: etch 49
29 Numbering of J1 connector pins (top view). 52
30 Picture of the final board (top view). 56
31 Picture of the final board (bottom view). 57

List of Tables

1 SONET and SDH equivalent designators 6
2 Reference designators of all components 44
3 J1 connector pin assignment. 54
4 Alarm indicated by each LED 55

1 Introduction

Modern ATM switches require high throughput memories in order to avoid blocking and to fully utilize the available link bandwidth. Modern memory technologies, such as Synchronous Dynamic RAM (SDRAM) and Rambus, offer significantly high throughput. Rambus is an expensive solution because it requires a macrocell interface to be integrated on the controller chip. On the other hand, SDRAM is simpler to use in a system without advanced ASIC's, and still offers peak bandwidth of 1.6Gbps (for 16-bit organization operating at 100MHz) with only one chip. There are a number of other memory technologies that can support high throughput, either on or off chip. On-chip memory can be made wide enough to support high throughput but their size is limited. Off chip memory (other than SDRAM) can support high throughput by building an interleaved or a bit sliced memory system by using multiple chips. These technologies are also expensive because they require a large number of chips to compensate for the limited throughput of each chip. Also, many chips means more difficult board layout. SDRAM is, thus, an attractive solution and is rapidly becoming popular.

On the other hand, in order to support the high throughput capabilities of SDRAM, a fast clock must be used. Frequencies of as much as 100MHz must be supported at the board level, which is not a trivial task. At such high frequencies, designs must adhere to strict design rules, or else the whole system may become inoperable. Issues like signal quality, minimum rise-fall time, signal reflection, controlled impedance, inductance, noise and many other must be addressed with great care.

Another high speed technology that is adopted as a promising high speed technology of the future, in both local and wide area networks, is ATM. A variety of reasons has led to this, including: (i) fixed-size cells make hardware simpler and faster, (ii) small cells allow low latency, and (iii) by using the same ATM equipment for both LANs and WANs, cost can be reduced through mass production. The most cost-effective physical medium over which ATM will operate is fibers. Fibers have higher transmission capacity than copper and are more reliable at the same time. Thus, ATM over SONET/SDH is becoming a promising technology for the future.

1.1 This thesis

In this direction, we decided the design and development of an FPGA-based prototype board, that will have two bidirectional links carrying ATM traffic over SONET/SDH and will use SDRAM as a large ATM cell buffer. Our primary goal was to demonstrate high clock frequencies at the board level and exploit the high throughput of SDRAM in an ATM- related design, through rapid prototyping of a board that would have flexibility and allow future extensions. The design was implemented on reprogrammable FPGA's which are relatively slow devices, so the architecture had to be simple to allow frequencies to rise. Besides high frequencies needed for the SDRAM operation, we had to support many different ones, in the presence of other slower devices operating at different frequencies. This was achieved using PLL based clock drivers/multipliers and numerous other external components.

This design allowed us to study in detail and gain knowledge and experience in the following areas:

- 155.52Mbit/s ATM over SONET/SDH with optical links, interfacing to a simple 8-bit UTOPIA bus.
- high throughput modern synchronous dynamic memories (SDRAMs), used as a large ATM cell buffer.
- high speed PCB design (100MHz), and how this speed is achieved using FPGA's.

The board that we implemented is used as an ATM cell buffer/forwarder. It has 2 input and 2 output links (or 2 bidirectional links) each running at 155.52Mbits/s and carrying ATM traffic. It uses modern Synchronous Dynamic Memory (SDRAM) to buffer cells, 2 FPGA's (ATM cell scheduler and memory controller) to operate on cells, a micro-controller interface for monitoring and initialization purposes, and numerous other components to support high frequency operation.

In section 2, we present the board structure in detail, the components it consists of and the operations it supports. Interfacing with SONET/SDH at 155.52Mbps (STS-3,STM-1) is explained in section 3. Section 4 describes SDRAM architecture and operation and explains why these memories are one of the best solutions for cell buffering. In section 5, the scheduler organization is described and the reasons it led us to choose the specific FPGA architecture are laid out. The SDRAM controller is presented in section 6, while interfacing with the micro-controller is explained in section 7. In section 8, various issues of the board design and implementation are presented and analyzed. We present simulation and actual test results of the board operation in section 9. Finally, section 10 gives our conclusions and describes future work.

2 Board Structure and Operations

The ATM cell buffer/forwarder can be seen in abstract form in figure 1. The Cell Scheduler receives incoming cells from SONET/SDH to UTOPIA interfaces and stores them in a 16Mbit SDRAM. The SDRAM is statically partitioned in two queues, one per output link. Cells are enqueued according to their VP/VC value and then, if the output link is free, the cell is dequeued and transmitted through the UTOPIA to SONET/SDH interface. Cut-through is not supported and VP/VC values are not translated in this prototype, because we focused in the simplicity and high speed of the hardware; yet, other more complex architectures can be supported through reprogramming of the FPGA's. Finally, a cell buffer size of 16Mbits is considered enough for this simple-functionality prototype.

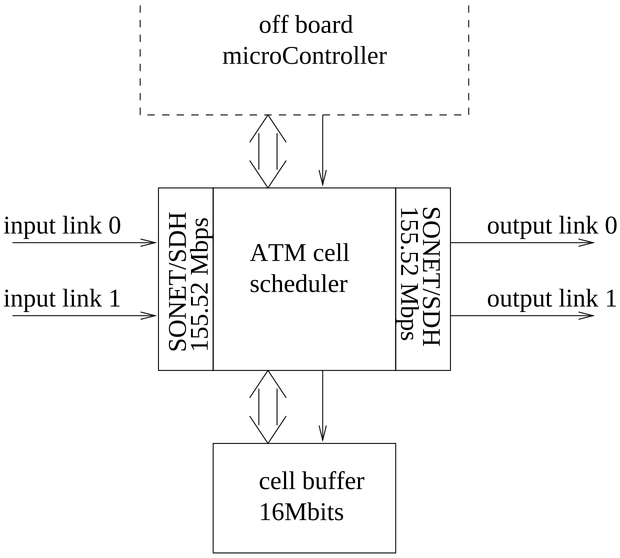

Figure 1: The ATM cell buffer/forwarder in abstract form

The board mainly consists of the following parts (see also figure 2):

- Electro-optical converters that convert the 155.52Mbps optical input to serial electrical and conversely the 155.52Mbps serial electrical output to optical signal. Since we have 2 input and 2 output links, we need two such converters.
- Two SONET/SDH ATM transceivers. At the receive side they convert the 155.52Mbits/s SONET/SDH bit serial input to an 8-bit wide ATM "UTOPIA" bus; cells can be read one byte at a time, at a speed of 19.44MHz (155.52 / 8) or more (due to the introduced SONET overhead bytes, the actual minimum rate is a bit less: 18.72MHz) At the transmit side the transceivers convert a 8-bit wide input to a bit serial output of 155.52Mbits/s.
- ATM cell scheduler. It is the core of the design because it performs all the operations on ATM cells (enqueue, dequeue). It maintains the head-tail pointers of the two output queues and drops cells when these overflow. The scheduler

uses internal buffers to accumulate entire cells, so that accesses to the SDRAM are performed in ”bursty” mode.

- SDRAM controller. The controller interfaces between the slow scheduler (50MHz) and the fast SDRAM (100MHz). This is not a full functionality controller since it issues a subset of the commands supported by the SDRAM and does not control the memory's address pins. The latter is left to the scheduler itself.
- 16Mbit dual bank synchronous dynamic memory (SDRAM). This memory serves as a cell buffer and is statically partitioned in two queues (one per output link). More specifically, each queue can store 19,784 ATM cells (the actual amount of cells is reduced to 18,432 due to a faster memory organization at the expense of wasting some buffer space).
- Two low voltage, low skew clock drivers. Implementation problems of the board dictated that 3 clock frequencies (with synchronized edges) had to be used: 25, 50 and 100MHz.
- Micro-controller interface (16-bit address, 8-bit data, various control signals) to allow initialization, programming, and monitoring of the board operation. The SONET/SDH transceivers connect to this interface with an 8-bit data bus and an 8-bit address bus making a number of operations available (interrupt handling, SONET or SDH enabling, interrupt masking, clock looping, etc.) The scheduler connects to this interface with a 16-bit address bus, an 8-bit wide data bus and many other control signals. Although many of these signals are not used at the time, they allow for future extensions.
- Two low speed but high precision clock oscillators. The SONET/SDH transceivers need a 19.44MHz input clock to generate the 155.52MHz clock for the bit serial outputs The second oscillator supplies the clock source for the rest of the board.
- PLL based clock generator with fully programmable output frequency. This generator allows us to test the board at various frequencies: low, if some mistake does not allow the design to operate at full speed, or higher than 100MHz, studying design's limits.
- 20 LEDs for various alarm indications generated by the SONET/SDH transceivers.
- numerous bypassing and decoupling capacitors combined with some inductors for power filtering and noise minimization.
- Terminating resistors for electromagnetic interference (EMI) suppression and signal reflection minimization.

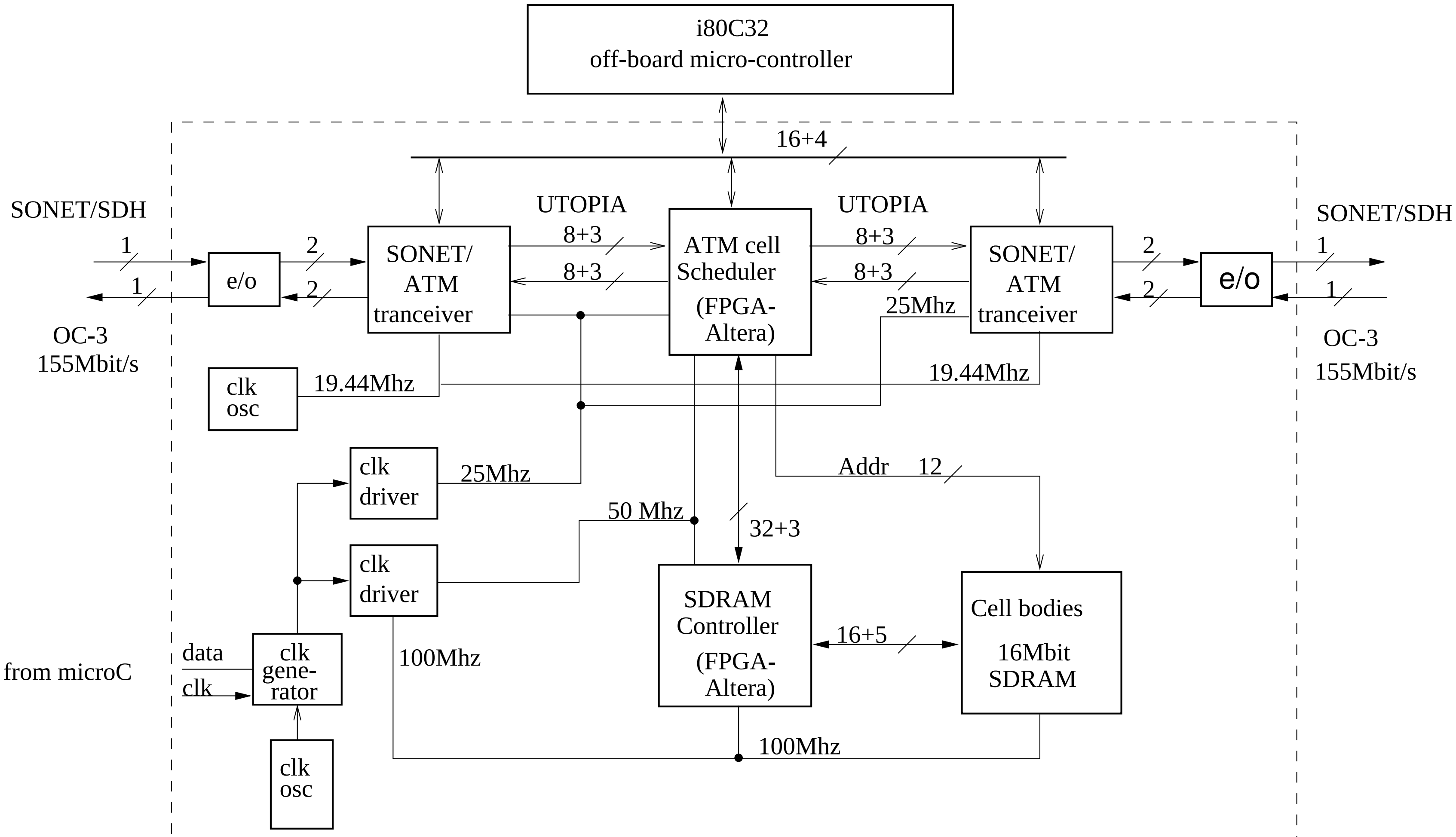

Figure 2: Overview of the prototype board

3 Interfacing with SONET/SDH at 155.52Mbps

3.1 The SONET and SDH standards

Synchronous optical network (SONET) is the standard for transmission over fiber optic medium. SONET provides the standard for interconnection of heterogeneous devices that may use optical interfaces. The basic SONET frequency is 51.84Mbps which is referred to as Synchronous Transport Signal level 1, STS-1. The International Telecommunications Union (ITU) developed SDH (Synchronous Digital Hierarchy) in 1988 which is based on SONET. The lowest rate in SDH standard is 155.52Mbps which is referred to as STM-1 (Synchronous Transport Module 1); its rate is three times that of STS-1. SONET/SDH standards have equivalent designators among various transmission rates; these are shown in Table 1.

SONET Designation	SDH Designation	Rate (Mbps)
STS-1		51.84
STS-3	STM-1	155.52
STS-9	STM-3	466.56
STS-12	STM-4	622.08
STS-18	STM-6	933.12
STS-24	STM-8	1244.16
STS-36	STM-12	1866.24
STS-48	STM-16	2488.32

Table 1: SONET and SDH equivalent designators

The building block for SONET is the STS-1 frame. Each frame consists of 9 rows and 90 columns as shown in figure 3. Each element in this matrix is one byte, that is the STS-1 frame is 810 bytes long. This frame is transmitted ”by rows”: the top row is transmitted first, left-to-right, then the second row and so forth. The first bit of a byte transmitted is the least significant bit.

The first 3 columns in each frame and one column in each payload field are allocated to OAM (operation and maintenance) and other end-to-end functions. SONET transmission system is defined as a hierarchy of layers: *photonic*,*section*,*line* and *path*. *Photonic* is responsible for the physical layer functions such as optical signal power, timing and dispersion characteristics. *Section* refers to optical cable segments between repeaters/regenerators that assist in rebuilding the optical signals. *Line* level is for synchronization, multiplexing, maintenance and protection of the switching functions. *Path* level is for end-to-end information transport. An overhead field in each SONET frame is associated with each one of these layers (except the photonic). *Section* overhead field consists of the first 3 columns in the first 3 rows in each frame. *Line* overhead consists of the first 3 columns in the remaining 6 rows,

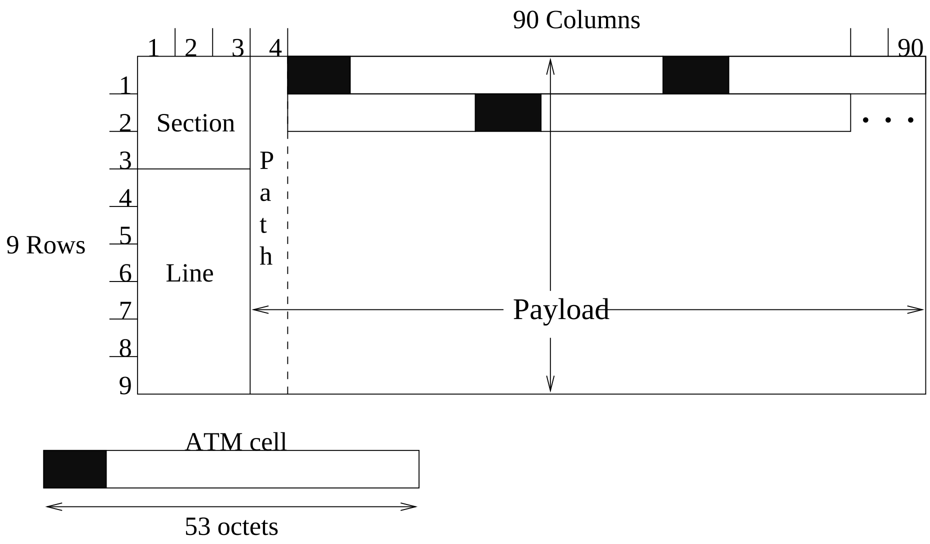

Figure 3: STS-1 frame structure

and *path* overhead is the first column in the payload field.

Each SONET frame is transmitted in 125µs. For STS-1 the frame is 9x90 = 810 bytes, which yields an 810bytes/125µs = 51.84Mbps transmission capacity. In order to create an STS-N frame (in our case STS-3 for 155.52Mbps): a) the first 3 columns from each STS-1 frame are put together (byte interleaved) creating the (3∗N)-column overhead field of the STS-N frame, b) the payload fields are byte interleaved together respectively. Thus, a (90∗N) x 9 frame is formed (270 columns x 9 rows in STS-3). This is shown on figure 4.

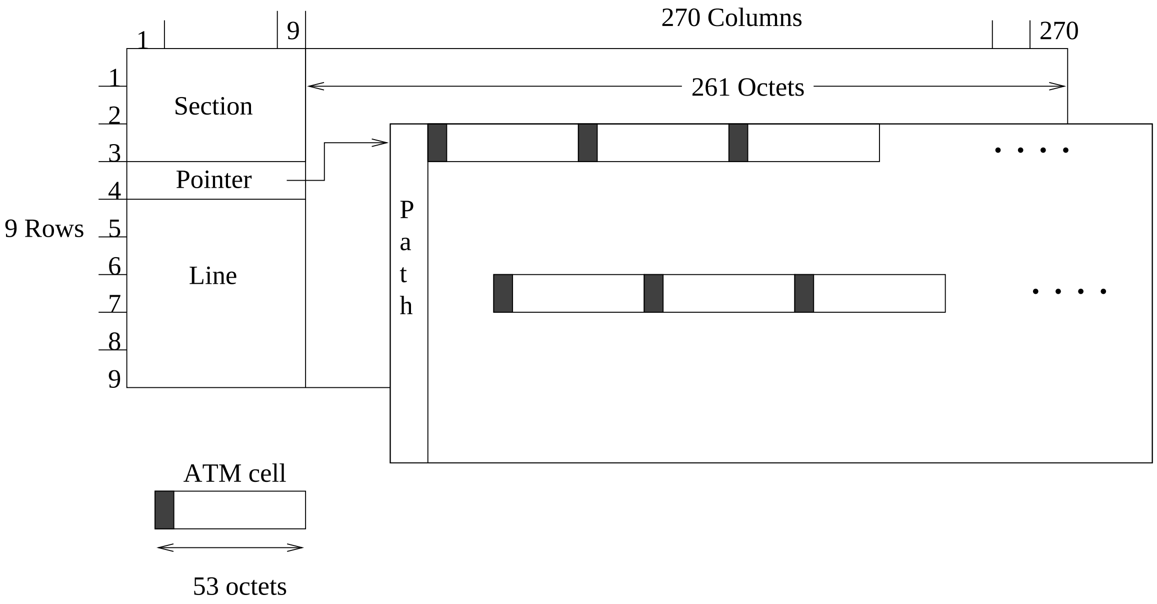

Figure 4: STS-3 frame structure

In order to accommodate for clock rate differences among various components in the network, the starting point of the payload is not at a standard position in the frame. Two bytes in line overhead field (referred to as H1 and H2 bytes) are used as pointer to point at the beginning of the payload in the frame, thus allowing the receiving device to locate the starting point of the payload. The values of H1 and H2 are incremented/decremented as frequency differences develop.

3.2 Electro-optical converters

For maximum reliability we chose to use fibers, instead of twisted-pair or coaxial cable, to communicate over SONET/SDH. Fibers have become the most cost effective and attractive medium for communication due to their high bandwidth, low loss, high transmission security and no vulnerability to electromagnetic interference.

Electrical signals are converted to optical using a Light-Emitting Diode (LED) or a laser and are sent into the fiber. On the other side, the electrical signals are regenerated using a photodetector, which can either be a PN photodetector or a PIN diode. Two types of fiber cables are in use: single mode and multimode. Single-mode is a very narrow fiber, which has the advantage of low scattering and low attenuation of light; however it is difficult to convert from this mode to another. On the other hand, multimode fibers are cheaper, have a wider diameter and are more easily connected to other fibers; yet, in multimode fibers light is deflected at the boundary of the fiber and signal dispersion-attenuation is quite higher. For a few tens of meters though, as in our prototype environment, signal attenuation creates no problem.

In our case, the AMP 269039-1 [5] multimode electro-optical converter was used. It is a bidirectional converter (elec. to optical and vice versa), its typical optical output signal is 1330nm wavelength and supports 156Mbits/s maximum data rate.

3.3 SONET/SDH ATM 155.52Mbps transceiver

In order to interface with the 155.52Mbits/s SONET/SDH lines, both at inputs and outputs, two transceivers are used (Texas Instruments TNETA1500 [10]). These single-chip devices are SONET/SDH ATM line interfaces that provide transportation of ATM cells over the SONET/SDH network at STS-3/STM-1 rates. Their main functionality is to insert and extract 53-byte ATM cells from/to STS-3/STM-1 synchronous payload envelope (SPE), providing a simple 8-bit wide data bus to other devices. A functional block diagram of the transceiver can be seen in figure 5.

On the the receive side, the TNETA1500 accepts 155.52Mbits/s differential bit-serial data, recovers the embedded clock signal, performs SONET/SDH frame alignment and serial-to-parallel conversion, identifies the SONET/SDH payload and establishes the ATM-cell boundaries. The ATM cells are extracted from the payload, descrambled and passed to the receive output FIFO for output to the next device (in our case, the ATM cell scheduler as can be seen in figure 2). This FIFO can hold 3 cells maximum and is reset upon overflow. If overflow occurs, three cells are lost; it is the responsibility of the scheduler to read from this FIFO at a high enough rate for this not to happen. Since serial data are written in FIFO at 155.52Mbps rate, bytes should be read from the FIFO at a rate of 19.44MHz minimum (the actual rate is a bit less due

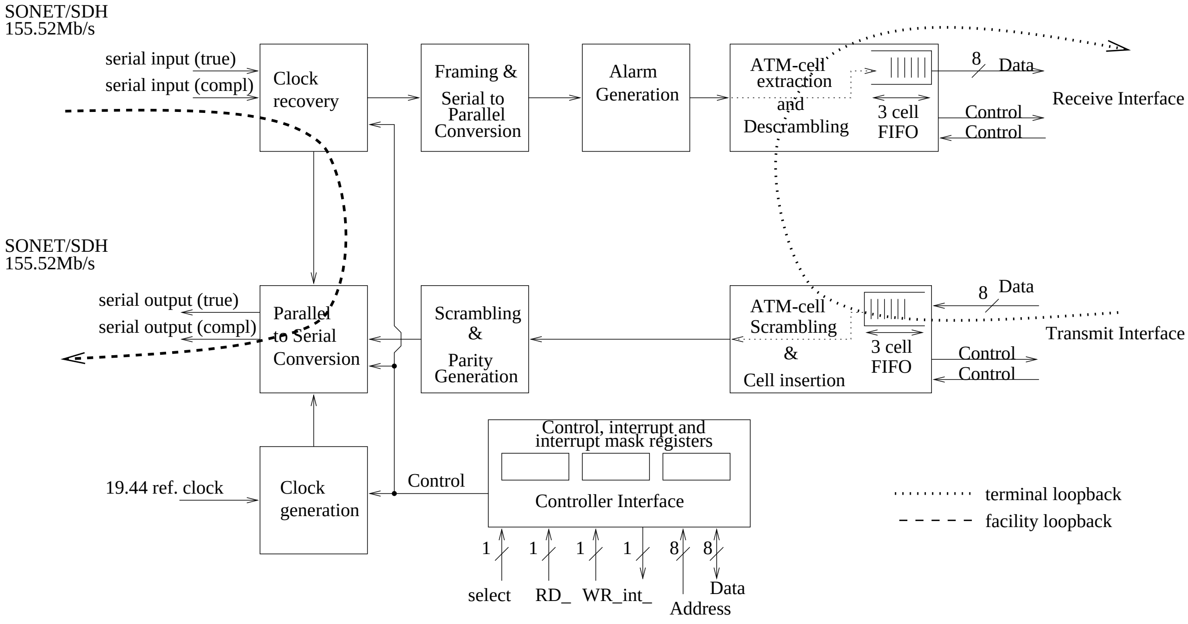

Figure 5: The SONET/SDH ATM transceiver functional block diagram

to the SONET overhead data that are not written in this FIFO). On the other hand, if the receive FIFO becomes empty the scheduler is notified and stops reading.

On the transmit side, complete 53-byte ATM cells are placed from the scheduler in the transmit input FIFO of the transceiver. These are then scrambled and inserted into an STS-3c/STM-1 synchronous payload envelope. This FIFO can hold 3 cells maximum and the scheduler is notified when it gets full. When this happens the scheduler must stop writing in the FIFO because any subsequent writes are ignored from the transceiver. FIFO may only get full if the scheduler writes in it faster than 19.44MHz (155.52/8). The SONET/SDH frame is then scrambled and converted to a differential bit-serial data stream for output. An analog PLL internal to the device is used to generate the 155.52MHz output clock from a low-speed 19.44MHz external oscillator.

For testing and debugging purposes the serial data input can be bypassed and the data coming from the 8-bit data transmit interface be given directly to the 8-bit data receive interface. This is called *terminal* loopback. Additionally, the high speed serial data input can be directly given to the high speed serial output for the same reason. This is called *facility* loopback. These functions can be seen in figure 5 with dotted and dashed lines.

The SONET/SDH transceivers use true and complementary positive ECL (PECL) serial data inputs and outputs. The low voltage swing of PECL, as well as the reduced vulnerability to external noise due to the differential operation, allow frequencies to rise. High frequencies of CMOS or TTL voltage level would induce too much noise and consume a lot of power. Except for the serial data input and output from the electro-optical converter, the transceivers receive the (differential) signal FLAGC which is asserted when the optical signal is lost. The connections between the electro-optical converter and the transceiver are shown in figure 6.

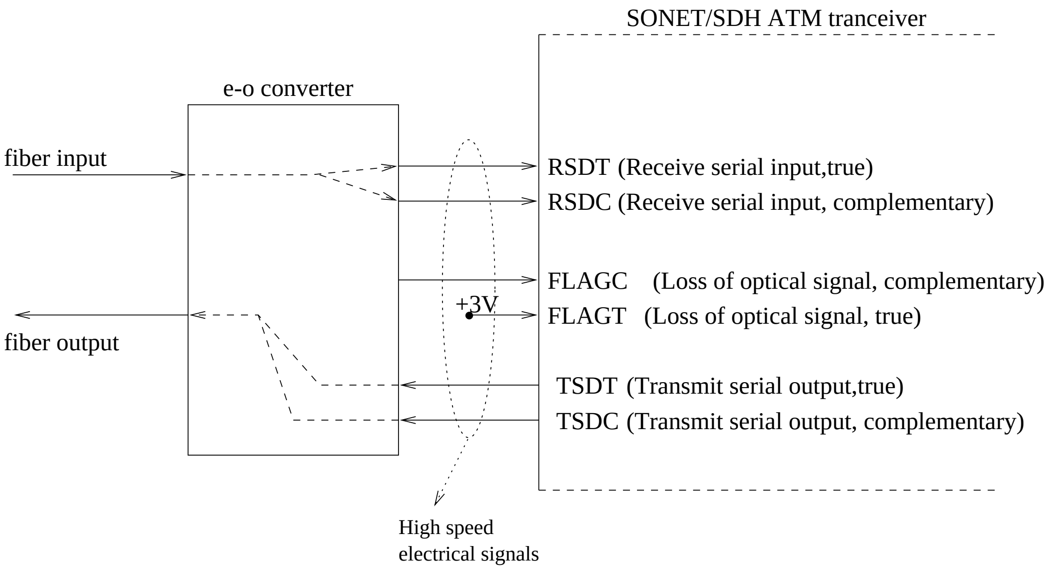

Figure 6: Connections between the electro-optical converter and the SONET/SDH ATM transceiver.

The transceivers offer a wide range of operations through the control interface. This interface has 8-bit data, 8-bit address and various control signals as can be seen in figure 5. The 8-bit control, interrupt and interrupt mask register can be read and written through this interface. The control register provides a means of controlling the operation of the device, e.g. disable or enable: error correction (for single-bit errors) of ATM-cell headers, terminal or facility loopback, dropping of unassigned ATM cells, idle or unassigned cell insertion, receive-clock looping etc. The interrupt register provides a means of monitoring the operation of the device. The transceiver reports various error conditions and alarms by setting the appropriate bit in the interrupt register. Reading this register through the control interface notifies an external device (e.g. processor) for these alarms. Some of the alarms generated are: loss of receive data, loss of frame, out of frame, loss of optical carrier etc. These alarms, also, cause the *interrupt* output signal of the device to go active low. Thus, the external device is first notified through this signal that an alarm has been generated and then the alarm can be identified by reading the interrupt register. If the external device does not want to be notified for specific alarms, it should set the corresponding bits in the interrupt mask register to avert the *interrupt* signal from being asserted. Reading the interrupt register causes the *interrupt* signal to go inactive and the interrupt register to be cleared.

4 Synchronous Dynamic RAM: Architecture and Operation

Synchronous Dynamic RAM (SDRAM) is a modern, inexpensive, random access memory, designed to meet the increased demand in data throughput. SDRAMs provide burst-mode (off-chip) accesses and offer (off-chip) transfer clock rates of up to 100MHz (as this report is written, new SDRAMs have been announced to achieve rates of up to 125MHz). This means that once the burst has started, all remaining bits are delivered at 10ns intervals. However, the random access delay is no better than a traditional DRAM (as will be explained later in this section). SDRAMs differ from DRAMs in that the the former use a clocked (synchronous) interface and have multiple banks. That is, every operation is initiated by asserting (or deasserting) the appropriate control signals (WE_, RAS_, CAS_, CS_, DQM) at the positive edge of the clock. The assertion or deassertion of these signals comprise the command given to the SDRAM. Also, data is read and written at the positive clock edge. An overview of the SDRAM architecture can be seen in figure 7.

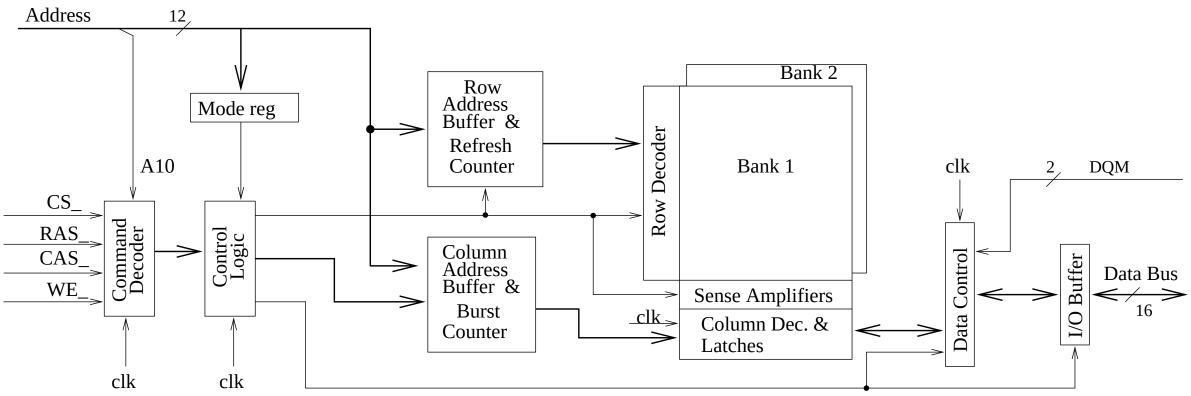

Figure 7: SDRAM architecture.

SDRAMs employ a mode register to allow various configurations according to user's needs: e.g. data can be read after one cycle of 33MHz or after 2 cycles of 66MHz or after 3 cycles of 100MHz from the time that the *Read* command is given, burst reads/writes can be 2,4,8 words or full page in size, address succession inside a burst can be sequential or interleaved, etc. Additionally, they have multiple banks in order to support on-chip interleaving of data and allow multiple pages of the DRAM to be opened simultaneously. SDRAMs allow automatic refresh, i.e. only the command for refresh need be given and no control of address pins is required.

In our case the NEC uPD4516161 [8] was used: it is a 16Mbit, dual bank SDRAM, with 512K 16bit words in each bank. Its organization is 2 banks x 2048 rows x 4096 columns, and has a 16-bit wide data bus. Its page (row) size is, thus, 256 16-bit words (4096 columns = 256 words x 16 bits/word). The programmed configuration that we used was appropriate for the highest clock speed - 100MHz (this induces a 3 clock cycle latency when reading, as we will see shortly). Also, the SDRAM was programmed for full page burst length, because every read/write was performed on a per cell basis (one read/write operation per cell). This full page burst is stopped when an entire ATM cell is read/written after 27 words, because otherwise the burst would go on for 229 more cycles.

Among the various operations supported by the SDRAM, only the following were used in our case, which suffice to fully support reading and writing of complete ATM cells (operation at 100MHz and burst length of full page are assumed):

- *Activate* command with the row address at the address pins. This command opens the bank and the row specified by the address. Then, a burst read/write to this row can be performed. No read/write can be performed if this command has not been issued. Accesses to the same row can be done without another *activate* command, whereas, for accessing other rows, this row must first be closed (see the *precharge* command below) and a new *activate* command with a new row address need be issued. Finally, the *activate* command induces 7 cycles latency if it is to be followed by the *precharge* command or 3 cycles latency if it is to be followed by other commands. That is, the *precharge* command cannot be given for the next 6 cycles following the *activate* command; other commands can only be given 3 or more cycles after the *activate* command.

- *Read* command with the column address at the address pins. This command will start reading from the column specified by the address, and will advance 16 columns (1 word) every cycle. The first data word will appear on the data bus 3 cycles (30ns) after the read command is given. This latency comes from the fact that the memory operates at high-speed; if it were to operate at 33MHz, the latency would be just one cycle (the actual time in both cases is the same: 30ns). New data words would appear on the data bus every cycle for 256 cycles (full page), if we did not stop this command after just 27 cycles. (27 cycles x 16 bits = 432 bits). The *Precharge* command is used to stop the read burst.

- *Write* command with the column address at the address pins. This command will start writing from the column specified by the address, and will advance 16 columns (1 word) every cycle. The first data word written in memory must be on the data bus at the same cycle that the write command is given. New data must be put on the bus every cycle, for 27 consecutive cycles in our case. Then, the burst must be stopped using the *precharge* command.

- *Precharge* command. This command performs two actions simultaneously: First, it stops the read/write bursts, and second, it closes the memory bank so that a new *activate* command for another row can be given. In our case, the *precharge* command is issued just after a complete ATM cell has been read/written. New *activate* commands cannot be given for the two cycles following the *precharge* command. This chip provides the option of precharging only the bank that is in use. We, however, for simplicity, always precharge both banks; to tell this to the chip, the 10th address bit is set at the same cycle the *precharge* command is given.

- *Refresh* command; this command initiates the automatic refresh sequence which lasts for 10 consecutive cycles. The address pins need not be controlled, since the SDRAM has an internal row counter, thus alleviating external controllers from extra complexity. All 4096 rows in the two banks must me refreshed at least once every 64ms, which yields one *refresh* command every 15.6µs average, or one every 1562 cycles (for 100MHz frequency). Thus, less than 0.65% (10 cycles every 1562) of the total throughput is lost due to refresh.

- *Register mode set* command with the proper register contents on the address bus. This register controls the burst length, read latency etc., thus the command must be issued at least once after the SDRAM has been powered on (and before any access is made).

In figure 8 and 9 we can see how a complete ATM cell is transferred from/to SDRAM.

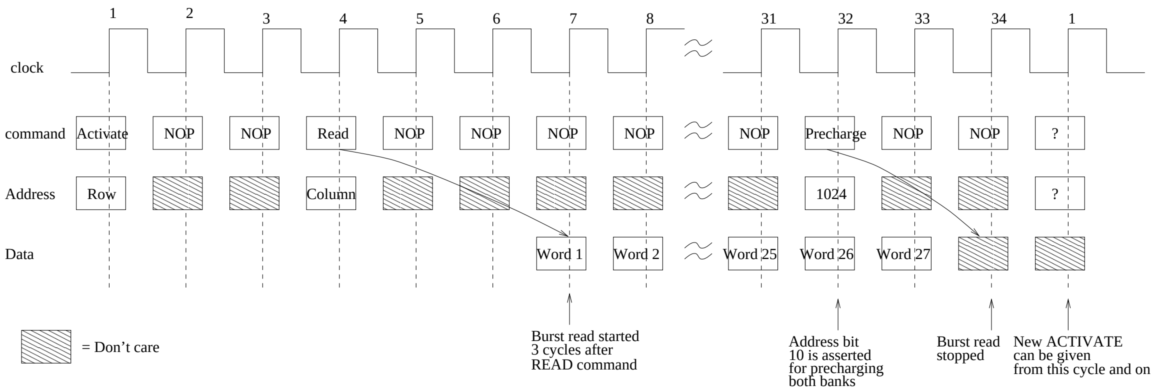

Figure 8: ATM cell read operation from SDRAM.

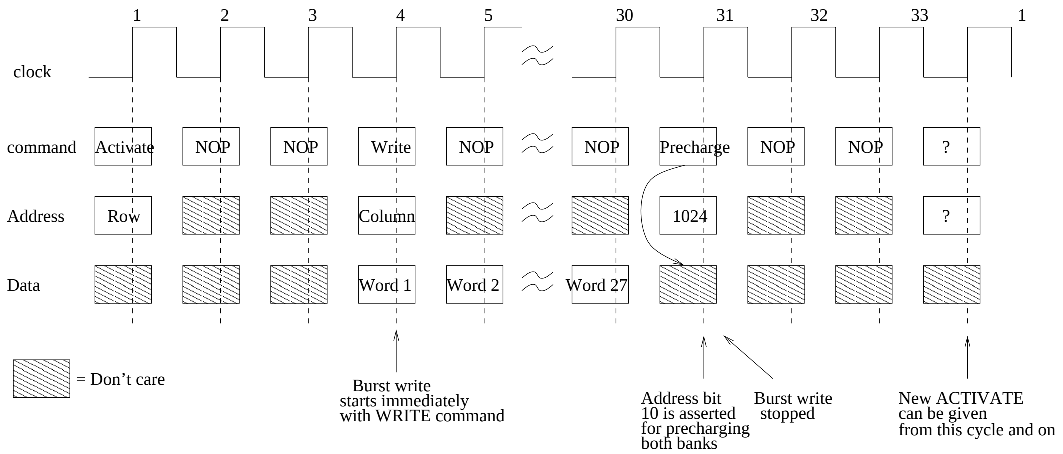

Figure 9: ATM cell write operation into SDRAM.

4.1 Throughput of SDRAM

From the above, we can understand that random accessing in SDRAM is not faster than in conventional DRAM. If we were to access just one word (16 bits), we would pay 10 cycles latency due to the SDRAM requirements: a) *activate* to *precharge* interval must be 7 cycles and b) *precharge* to *activate* interval must be 3 cycles. This is 100ns (way below our requirements), giving a total throughput of only 160Mbit/s.

Instead, by accessing 27 consecutive words, we have only 33 cycles latency for writing (3 for activating, 27 for writing, 3 for precharging) and 34 for reading (same as writing, plus one cycle extra latency, see figure 8). This yields a throughput of 424 bits (ATM cell) every 340ns which exceeds 1.2Gbps. That's why any 53-byte ATM cell resides in 27 sequential words in only one row.

It is obvious that throughput depends on the burst size. Larger burst sizes offer more throughput, approaching the peak throughput that the organization of the chip allows. One SDRAM chip with 16 I/O pins has 1.6Gbps peak throughput, whereas the chips with 8 and 4 I/O pins have 0.8Gbps and 0.4Gbps peak throughput respectively (operation at the 100MHz frequency is assumed). SDRAM, also, offers the advantage of interleaving accessing, i.e. activating a row in one bank and preparing to making an access to it at the same time an access to the other bank is already taking place. When alternating accesses from one bank to another continuously, one can exploit the peak throughput of the device, because activating and precharging is done "off-line". When interleaved accesses are not made, throughput depends on the size of the burst only. Figure 10 shows the throughput of one SDRAM chip with various organizations (4, 8 or 16 I/O pins) for various burst sizes. When supporting interleaved accesses the mean throughput is somewhere between the peak throughput of the device ("I" lines in figure 10) and the throughput offered when no interleaving is used ("nonI" lines in figure 10).

Thus, SDRAMs are most appropriate for cell storage, because large capacity at reduced cost together with high throughput is required and because accesses are quite predictable: all accesses are to 53-byte blocks which reside in consecutive locations.

In our case, one memory row, which is 256 16-bit words (4096 bits) in size, can hold at most 9 ATM cells ($\lfloor 256 words/27 words\ per\ cell \rfloor$). The total useful storage in a row is 9 cells $*$ 424 bits = 3816 bits, while the remaining 4096 - 3816 = 280 bits cannot be used since they cannot hold a complete ATM cell. This yields a 280 bit waste every 4096 bits, or 6.83% waste of the memory space. This memory waste was not considered significant as opposed to a) simplicity and uniformity of memory accesses, b) easy address calculation and c) fast and simple empty-full check. A maximum of 36864 cells (4096 rows $*$ 9 cells per row) can be present at any time in the SDRAM.

Multi-chip organizations using SDRAM can increase the offered throughput. When an ATM cell resides in more than one SDRAM chip, then a cell transfer will last less than having only one chip, due to the increased peak data throughput. However, the mean throughput will not be proportional to the total number of chips containing

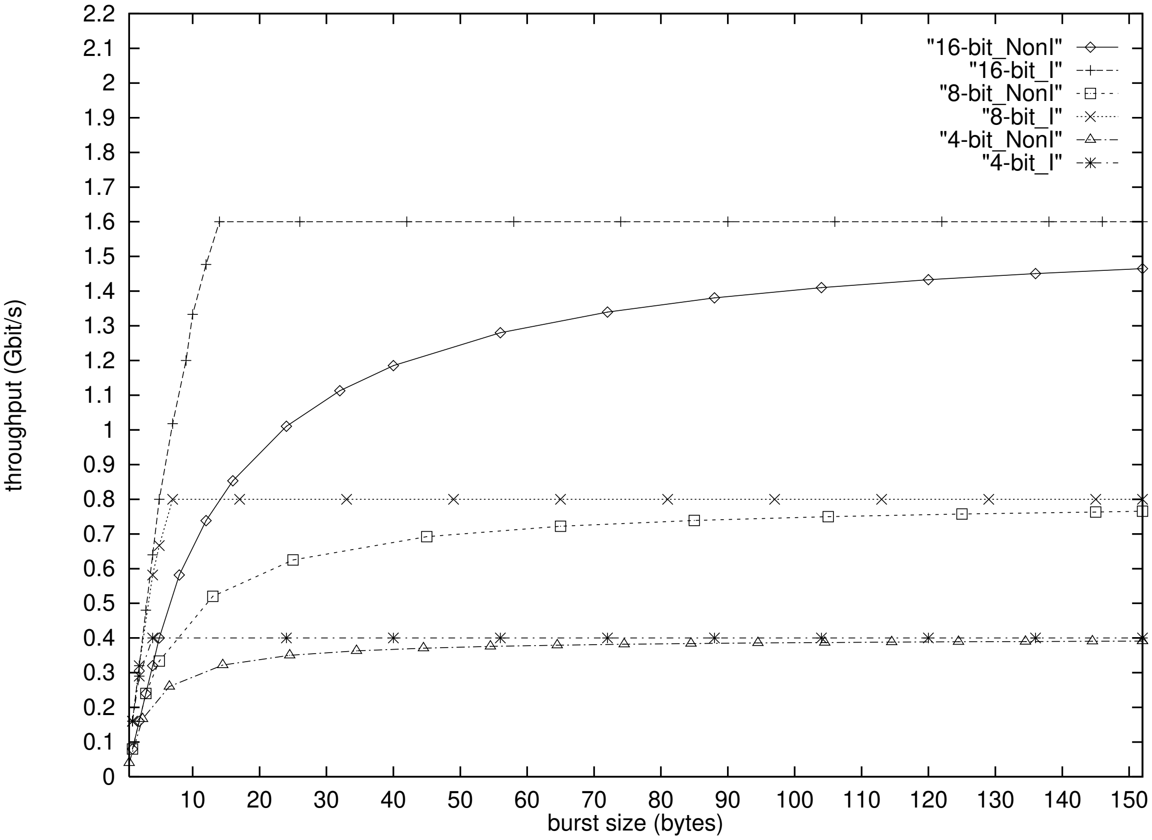

Figure 10: Throughput offered by an SDRAM chip with various organizations (16, 8 or 4 I/O pins) for various burst sizes (I stands for interleaved accesses, NonI stands for non-interleaved).

the cell. The reason for this is that the 7-cycle latency will still be paid in the multi-chip organizations. For example, the latency for a 2-chip organization is 21 cycles (14cc data burst + 7cc latency) and not 34/2=17cc (34cc is the latency for 1-chip organization).

On the other hand, when an ATM cell resides in only one SDRAM chip in multi-chip organizations, the total buffer space and the number of banks are increasing, when compared to the single-chip organization. Accesses going to different banks can be done simultaneously, increasing throughput. However, in these architectures, throughput is limited due to bank conflicts.

5 The ATM Cell Scheduler

The ATM cell scheduler is the core of the design. It implements all the operations on cells (enqueuing, dequeuing, dropping when buffers overflow etc.). The scheduler interfaces with the two SONET transceivers, thus servicing both input and both output links. The scheduler is implemented in a large FPGA which is much slower than SDRAM and consequently no direct connection between the two is possible. For this reason, a fast memory controller is used to interface between these devices. The controller is described in section 6. Since the SDRAM controller must communicate with the SDRAM at 100MHz over a 16-bit bus, it is convenient for the scheduler to operate at 50MHz (reasonable enough for a large modern FPGA) and interface with the controller over a 32-bit bus, thus achieving the same data throughput (16bits$*$100MHz=32bits$*$50MHz). However, timing requirements of the SONET/SDH transceivers do not allow their interfaces with the scheduler to operate at 50MHz; for this reason, a third clock running at 25MHz is used (the 19.44MHz minimum is satisfied). Thus, the scheduler reads/writes from/to the SONET/SDH transceivers at the (instantaneous) rate of 8-bits/40ns. These three clock sources are synchronized with each other (simultaneous positive edges), to avoid synchronization delays in our system. The scheduler interfaces are shown in detail in section 5.1, 5.2 and 5.3.

5.1 Receive Side Interface

The scheduler receives ATM cells from the SONET/SDH transceivers using the following signals: RRE_, RXFE_, RXCELL and RDAT. These can be seen in figure 11.

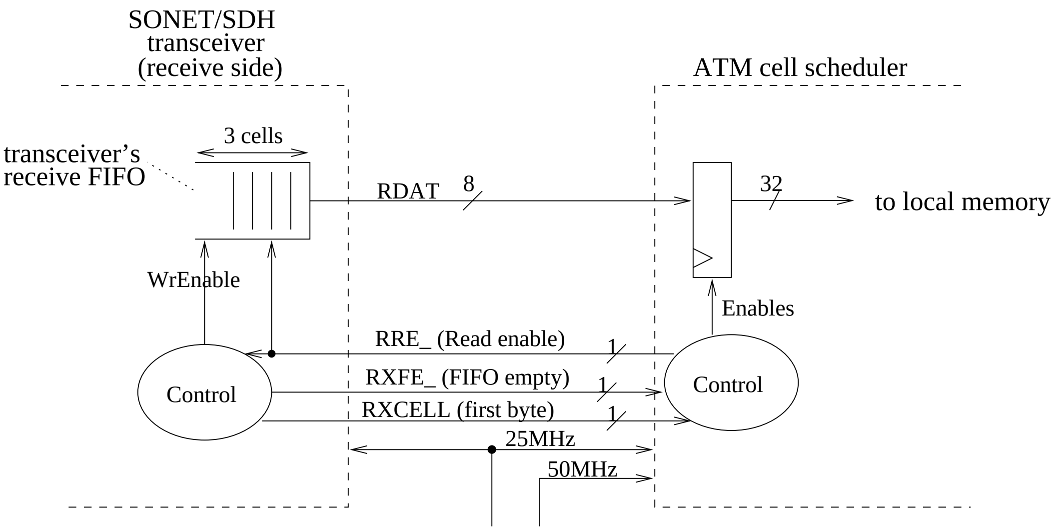

Figure 11: Receive cell interface between the transceiver and the scheduler.

Reading from the transceiver's FIFO is enabled by asserting the RRE_ signal (scheduler output). Bytes read from the FIFO are placed on the RDAT bus and are identified valid by the deasserted RXFE_ signal. RXFE_ goes active low when the receive FIFO is empty and the current byte on the bus is invalid. Thus, a byte on the bus is valid at a cycle a, if RXFE_ is deasserted in the same cycle a and RRE_ was asserted in cycle $a-1$. In our case, the RRE_ output is constantly kept asserted because the scheduler can read from the FIFO every cycle (if the FIFO never emptied). Of course, reading cannot be continuous because the scheduler reads the FIFO faster than it is written (25 vs. 19.44MHz), so the scheduler finds the FIFO empty periodically. Thus, ATM cells are received by the scheduler at the peak rate of 1byte/40ns, but at the average rate of 1byte/51.44ns (51.44ns is the cycle time of the 19.44MHz clock). Finally, the RXCELL signal identifies the first byte of the received ATM cell.

5.2 Transmit Side Interface

The scheduler transmits ATM cells to the SONET/SDH transceivers using the following signals: TWE_, TXAF_, TXCELL, TDAT. These can be seen in figure 12

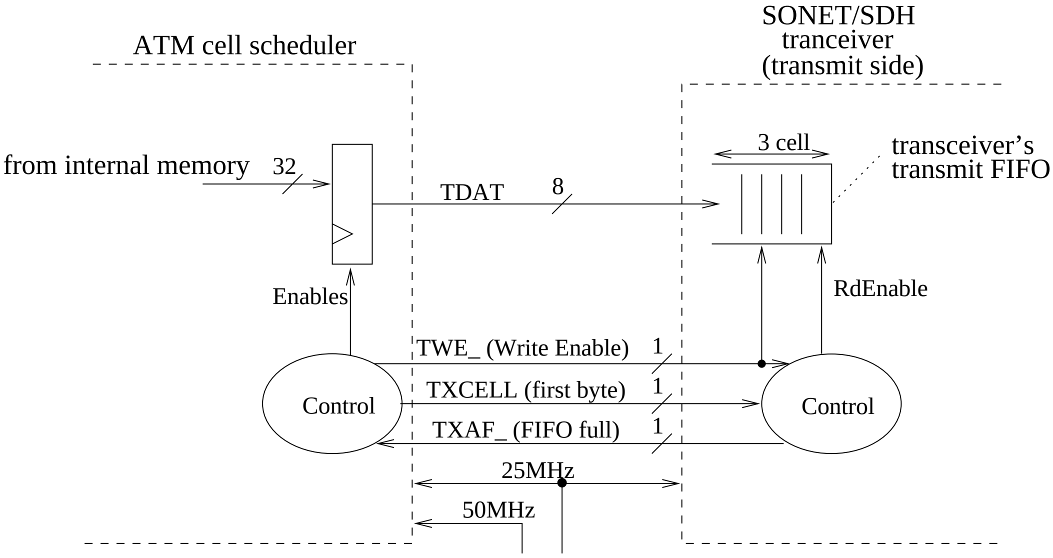

Figure 12: Transmit cell interface between the transceiver and the scheduler.

Writing to the transceiver's FIFO is enabled by asserting the TWE_ signal (scheduler's output). The transceiver notifies the scheduler that the FIFO is full by asserting the TXAF_ signal, in which case the scheduler must not transmit a byte in the next cycle, or else that byte will be ignored by the transceiver. Thus, a byte is actually transmitted at a cycle a if TWE_ is asserted in the same cycle a and TXAF_ was deasserted in cycle $a-1$. Thus, data is transferred at the peak rate of 1byte/40ns, but since the transceiver cannot transmit them to SONET/SDH that fast, the average rate must be less than or equal to 1byte/51.44ns (51.44ns is the cycle time of the 19.44MHz clock). Finally, TXCELL identifies the first byte of a transmitted ATM cell.

The scheduler uses the 25MHz clock signal both in the receive and transmit interface to synchronize with the transceiver, although the scheduler operates internally at 50MHz.

5.3 Interface to the SDRAM Controller

The scheduler communicates with the SDRAM controller with a 32-bit data bus and a 3-bit command. This can be seen in figure 13.

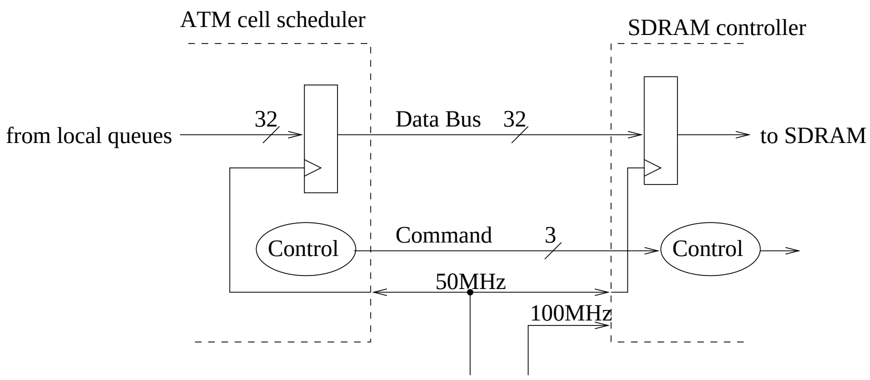

Figure 13: Interface between the scheduler and the SDRAM controller.

Synchronization is achieved through the 50MHz clock signal. The scheduler can issue a total of 6 commands to the controller: *Mode Register Set*, *Precharge*, *Read*, *Write*, *Refresh* and *No operation*. For a memory read or write operation the scheduler issues the *Read* or *Write* command and then, after a number of cycles puts (or reads) the data on (from) the data bus. Memory address is not given to the controller; instead it is directly given from the scheduler to the SDRAM. This will be explained in section 6.

5.4 Scheduler architecture

The scheduler is implemented in an ALTERA "FLEX10K40" FPGA [3], which has internal single-ported memory and allows us to fully exploit the SDRAM's throughput. As we saw in section 4.1, a complete cell must be transferred to/from the SDRAM without interruption, in order to achieve a 34 cycle latency for a cell transfer. The incoming cell (from an input link) may be interrupted at any time (no control over the input links exists), so if the cell is immediately forwarded to SDRAM as it is received, this memory transfer may be interrupted at any time. Subsequently, a complete ATM cell must first be buffered inside the scheduler, and then transferred in its entirety to the SDRAM; FPGA's internal memory is used to buffer the cell inside the scheduler. Also, since the scheduler is transmitting bytes to the transceivers at a rate higher than the transceiver can serve them (25 vs. 19.44MHz), it may be the case that the transceiver FIFO gets full, so the scheduler must stop transmitting. Consequently, if the cell transmitted to the output links was coming directly from cell memory, this cell memory read would be interruptible at any time. Instead, the cell read from the external memory (SDRAM) is first buffered in the scheduler's internal memory, and then it can be transmitted to the output link with any number of interruptions.

All these are implemented using four logical queues (one per link), called *local queues*, which are maintained in the scheduler's internal memory. Each local queue can hold 2 ATM cells. Two large output queues (one per output link) are implemented externally, in the cell memory (SDRAM). SDRAM is statically partitioned in these two queues. This static partitioning makes hardware simpler and faster because only two pointers are needed (*head* and *tail*) for each queue. On the other hand, memory under-utilization is observed when one queue overflows and cells are lost, while the other queue is still non-full. The local queues can be seen in figure 14 along with the two output queues.

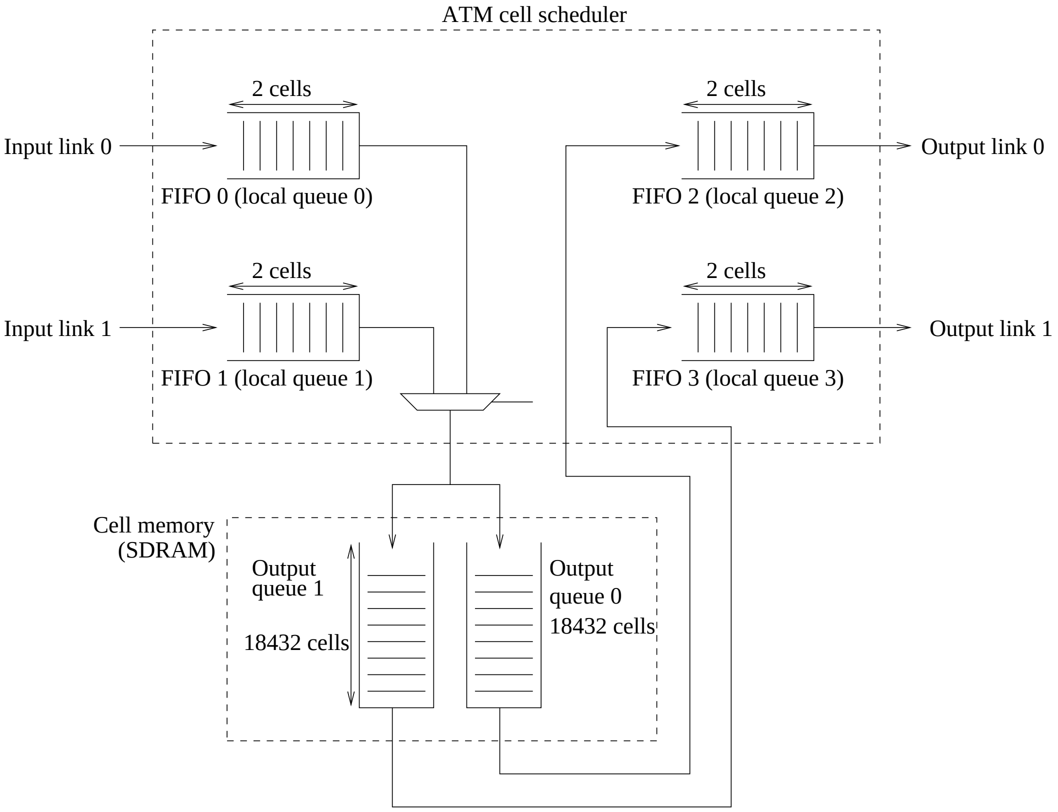

Figure 14: Logical queue organization of the scheduler.

As an ATM cell is being received from the input link, it is written in the appropriate local queue. When an entire cell exists in an input local queue, it is considered candidate for enqueue in the cell memory. In other words, the input local queues are the tails of the output queues residing in cell memory. It is guaranteed that a cell just arrived at one of the local queues will be enqueued in the cell memory before another cell time elapses, thus a 2-cell deep local queue suffices and overflow will never happen. When an output local queue has either one or no cell in it, a cell is dequeued (if it exists) from the corresponding queue in cell memory and written in the local queue. In other words, an output local queue is the 2-cell deep head of the corresponding output queue of the cell memory. It is guaranteed that if a cell in cell memory exists, it will be written in the local output queue at a fast enough rate to keep the output link utilization at 100%.

Thus, the main operations of the scheduler are:

- A complete ATM cell is gathered from the input links inside the internal memory (in the link's input local queue).

- A cell residing in an input local queue is enqueued, according to its VP/VC value, to the appropriate output queue in cell memory (SDRAM) if this latter queue is not full. Writing to cell memory is done in consecutive cycles without interruption.

- If an output local queue has space for at least one cell, a cell is dequeued from the corresponding queue in cell memory (if it is not empty) and written in this output local queue. Reading from the cell memory is done in consecutive cycles without interruption.

- If an output link is ready (the transmit FIFO of the transceiver is not full), and the corresponding output local queue is non-empty, bytes are read from that output local queue and are transmitted to the transceiver.

Cut-through is not supported in this design and the destination link of a cell is mapped statically to VP/VC values, i.e. routing is decided at the time of FPGA programming. This can be altered in future versions of the design, to allow VP/VC translation and routing table updates from the micro-controller.

The scheduler's block diagram will be seen in section 5.6.

5.5 Dropping ATM cells upon cell memory overflow

Cells destined to the same output link may arrive contiguously and from both input links, in which case the corresponding output queue is filled at a rate of twice the rate at which it is emptied. When an output queue in cell memory becomes full, the scheduler drops any further cells destined to this output queue, from whichever input local queue. It is reminded that an input local queue has the following characteristics: a) it is 2-cell deep, b) a complete cell must first be written in it before the cell is read out to the cell memory and c) it is read at a much higher rate than it is written. Thus, an input local queue will contain up to one complete cell at any time, and because we have two such queues, up to two cells can be dropped when an output queue becomes full.

A second option would be to drop only one cell every time an output queue becomes full. Choosing which cell to drop would be based on the link it came from: e.g. a round robin algorithm chooses cells to drop, alternating between those coming from input 0 and those coming from input 1. Dropping one cell every time an output queue becomes full leads to the output queue being filled up with a rate not higher than it is read. Thus, no overflow occurs. However, the first scheme is much simpler and not inferior in performance than this one; dropping up to 2 cells every time an output queue fills up, leaves one extra cell-space for the next cell going to the same queue (this "next" cell would be dropped with the second scheme). Thus, with the first scheme, the total number of the dropped cells is the same and thus, the output link will still be fully utilized.

A third option would be to stop receiving bytes from the SONET/SDH transceivers. But then, it is the transceiver's FIFO that would become full after a while. It is reminded that when a transceiver's FIFO overflows, up to 3 ATM cells are lost, some of them possibly destined to the other (non-full) output queue. With the first scheme, cells destined to the other (non-full) output queue do proceed normally.

5.6 Internal memory organization

We chose to implement the local queues in internal memory instead of a few thousand registers because in that case, control logic would be too complex and too slow. Internal memory of ALTERA's devices consists of a number of Embedded Array Blocks (EAB), each one being a 256x8bit synchronous SRAM (2048 bits per EAB). In our design, a minimum of $8*424=3392$ bits are required to support the four 2-cell deep local queues. This means that 2 EABs (4096 bits total size) would suffice as far as buffer space is concerned. On the other hand, we need an 1.6Gbps peak throughput to access the cell memory (SDRAM requires 16bits/10ns) and 0.8Gbps peak throughput to serve input/output links (4 links $*$ 8 bits $*$ 25MHz). Thus a total of 2.4Gbps must be supported by the internal memory. Each EAB has a 0.4Gbps throughput (8-bit at 20ns cycle because scheduler operates internally at 50MHz), which means that a total of 6 EABs are needed. We chose to use 8 EABs, (instead of 6), so that two 32-bit wide banks can be formed. This makes hardware much simpler, since all memory accesses are now made to 32-bit words. Also, every memory access is thus serviced by either of the two banks each, instead of varying combinations of EABs. The same holds true for the communication between scheduler and SDRAM controller, which is done over a 32-bit data bus.

The block diagram of the scheduler can be seen in figure 15. The internal memory is divided in two banks of 32-bit width each. The tristates drivers in this figure are shown for simplicity; ALTERA's FPGA's do not have internal tristates (except at the I/O pins), so multiplexors are used instead.

Under this scheme, however, if a cell resided in only one bank, then this bank would be continuously busy for 14 cycles during an enqueue/dequeue operation from/to the cell memory (SDRAM), because the 53 bytes of the cell reside in 14 words of the internal memory and must be read/written sequentially without interruption. If input/output links needed the internal memory at the same time that a transfer to/from the cell memory was taking place, they would have to wait for at least 14 cycles, since the memory is single-ported (not considering bank conflicts between the links which would increase this delay). This would increase the demand for local buffering at the links, that must wait for the internal memory to become available, and would make hardware more complex. Additionally, if at some time all links and cell memory were conflicting over the same memory bank, internal memory throughput would not be sufficient.

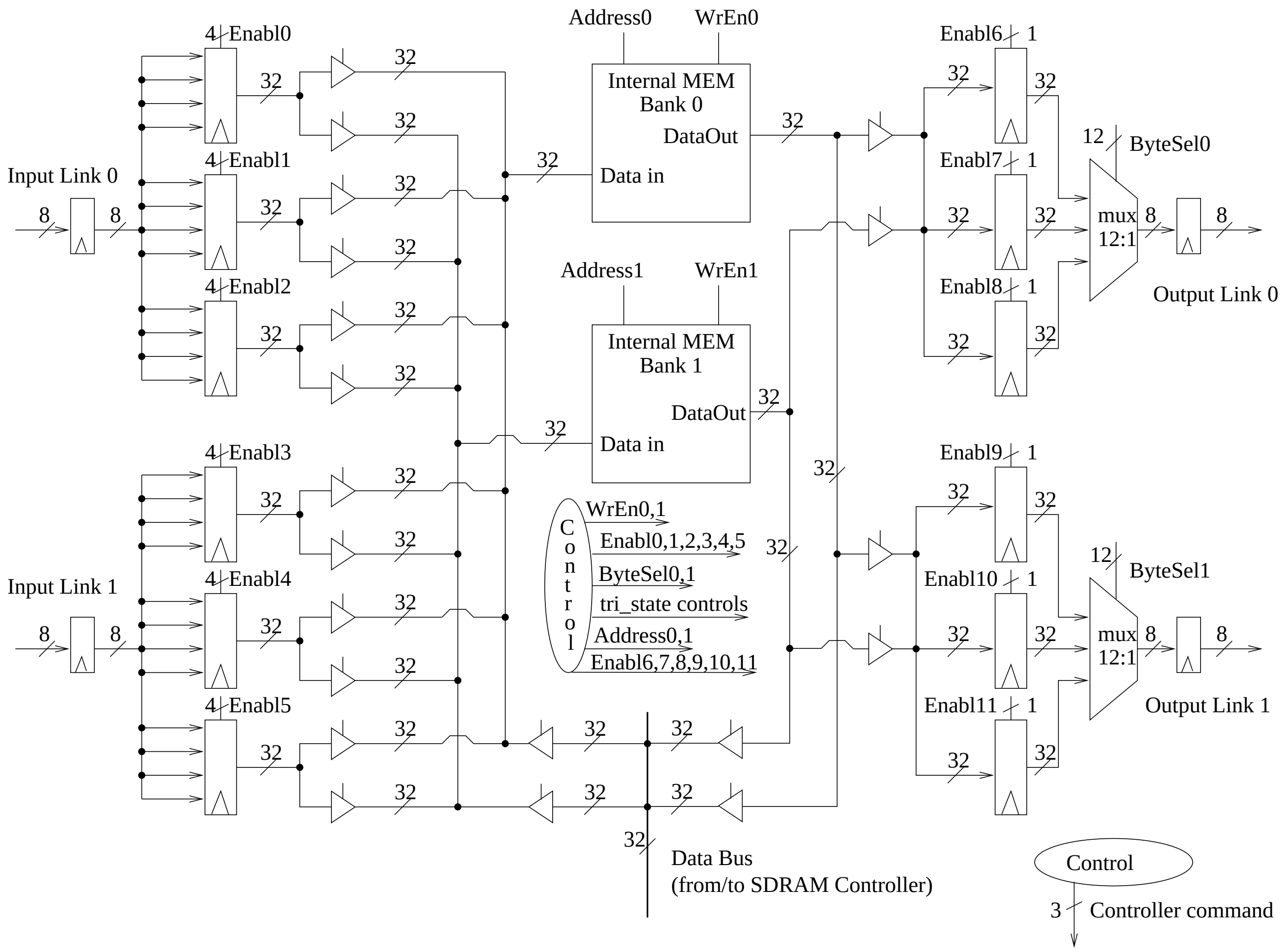

Figure 15: ATM cell scheduler block diagram.

For all these reasons, the following scheme was chosen:

- Two 32-bit wide memory banks are formed.
- Every cell resides in both banks.
- Bank 0 holds all even words of all cells (words 0,2,...,12)
- Bank 1 holds all odd words of all cells (words 1,3,...,13)

This is shown in figure 16. One memory word between cells is left unused in order to make addressing simpler (every cell starts at an address which is a multiple of 8).

Thus, every cell memory transfer will use each bank every other cycle, so each bank will be made available to input/output links after 1 cycle delay (maximum). Under this scheme, bank conflicts are minimized but still exist between input/output links and cell memory. Due to the inevitable remaining bank conflicts, additional buffering at the input and output links is required. The worst case is that all four links and the cell memory content for the same memory bank at the same time. In this case,

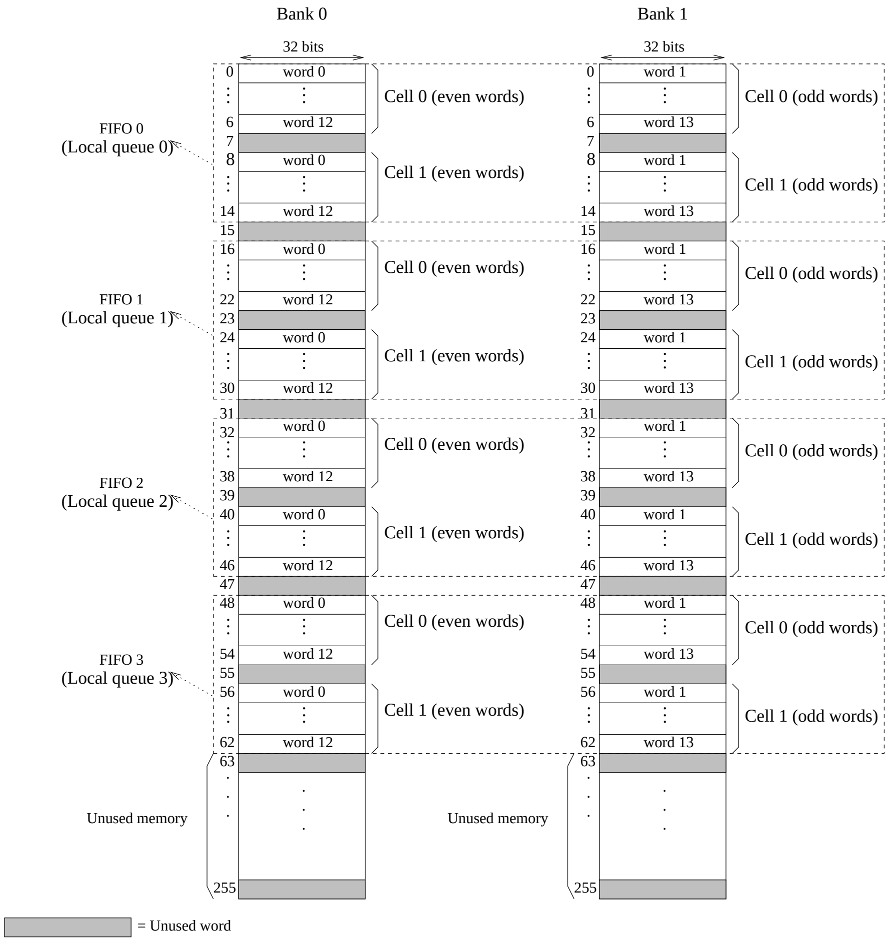

Figure 16: Cells are residing in both banks to reduce bank conflicts and increase throughput.

analysis showed that a 12-byte (or 3 32-bit word) elastic buffer should be used at the input and output links (shown in figure 15). This worst case is analyzed in section 5.7.

Internal memory is shared among all four links and the cell memory. Arbitration is made by using priorities. Ordering them from highest priority to lowest: cell memory, input link 0, input link1, output link 0, output link1. Although this seems unfair, the scheduler's architecture described above, guarantees that all links (even that with the lowest priority) will be utilized 100%. Finally, a simple round-robin algorithm is used to decide which local queue will transfer cells to/from the cell memory.

5.7 Elastic buffer at input/output links: worst case analysis

As already mentioned, 3 32-bit registers are used for buffering at each link, allowing fast reads/writes of 32-bit data from/to local queues in one clock cycle, or 20ns. In order to keep control logic simple, the first byte of every cell is always written in the first byte position of any register. This applies even when an ATM cell ends (see figure 17). It is reminded that input registers are written by incoming cells and their contents are then written into the input local queues, whereas output registers are written by outgoing cells, read from the output local queues, and their contents are read and transmitted on the links. When an ATM cell ends, an access to internal memory reads/writes only one byte (the last one) in these registers and not four bytes as the usual case is. Thus, we have a momentary increase in internal memory usage when the last-byte of the cell is being received/transmitted.

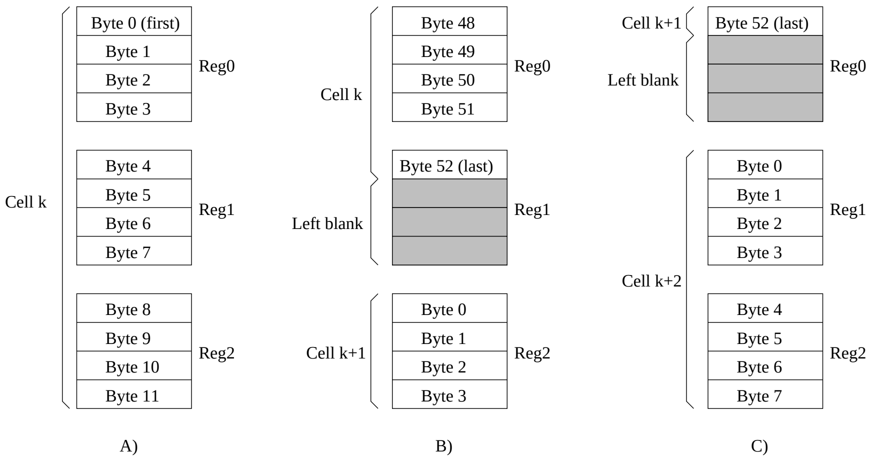

Figure 17: The first byte of a cell is always written at the first byte position of a register: A) start of a cell k, B) end of cell k and start of $k+1$, C) end of $k+1$ and start of $k+2$.

Let's assume now that all links transmit/receive the same byte of the cell at the same time, so they will reach the end-of-packet simultaneously. Also, assume that at some cycle (say cycle 0), all registers at the output links are full and all registers at input links are empty. The last byte of each and every cell is going to be written in Reg1 (at both inputs) and is already stored in Reg1 (at both outputs) (see figure 18).

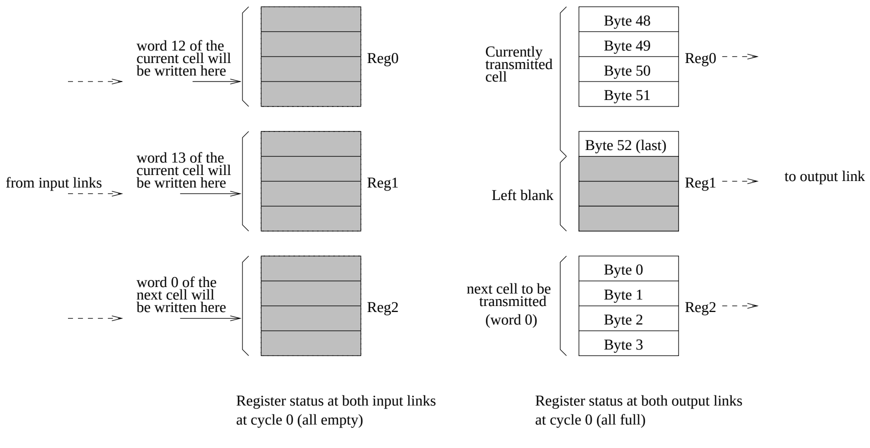

Figure 18: Cycle 0: all registers at input links are completely empty and all registers at output links are completely full.

Bytes are read/written from/to these registers every 2 cycles (50MHz), because the scheduler runs at twice the speed of the interface with the SONET transceiver (25MHz). At the same time, assume there is an ongoing transfer to cell memory, which has the highest priority since it must not be interrupted. What follows can be seen on figure 19. The first row in the table shows the cycle that things happen. The second and third row show who is using the internal memory, for example ”a1” means input link0-Reg1, ”c0” means output link 0-Reg0, etc.

At cycle 5, no register on any link needs accessing internal memory since no register is either completely full (at inputs) or completely empty (at outputs). At cycle 6, all links receive (or transmit) the 51st byte of the cell, so all Reg1s will need an access to the internal memory. At cycle 7 all links start contending for bank 1, because they need their next word. However, no register is served at cycle 7 since bank 1 is busy transmitting to cell memory. At cycle 8, link’s registers are beginning to be served, but as it can be seen, output link 1 will wait till cycle 14 to be served (it has the smallest priority). If it were not for Reg2 to provide byte 1 through byte 4 for transmission after cycle 9, output link 1 would have to be idle because Reg0 and Reg1 are empty at this time, although there are bytes in memory waiting to be transmitted.

The same situation applies for all links except input link 0: as can be seen in figure 19, input link 0 uses bank 1 at cycle 8 before Reg1 is emptied. This means that only two registers would suffice and Reg2 is, thus, unneeded. In fact, due to the deep pipeline that the scheduler employs, there is a 3 cycle extra latency from the time that bank 1 seems available, till the time that it will be actually accessed. Thus, Reg 2 is necessary for input link 0 too. Thus, 3 32-bit registers are used at every link.

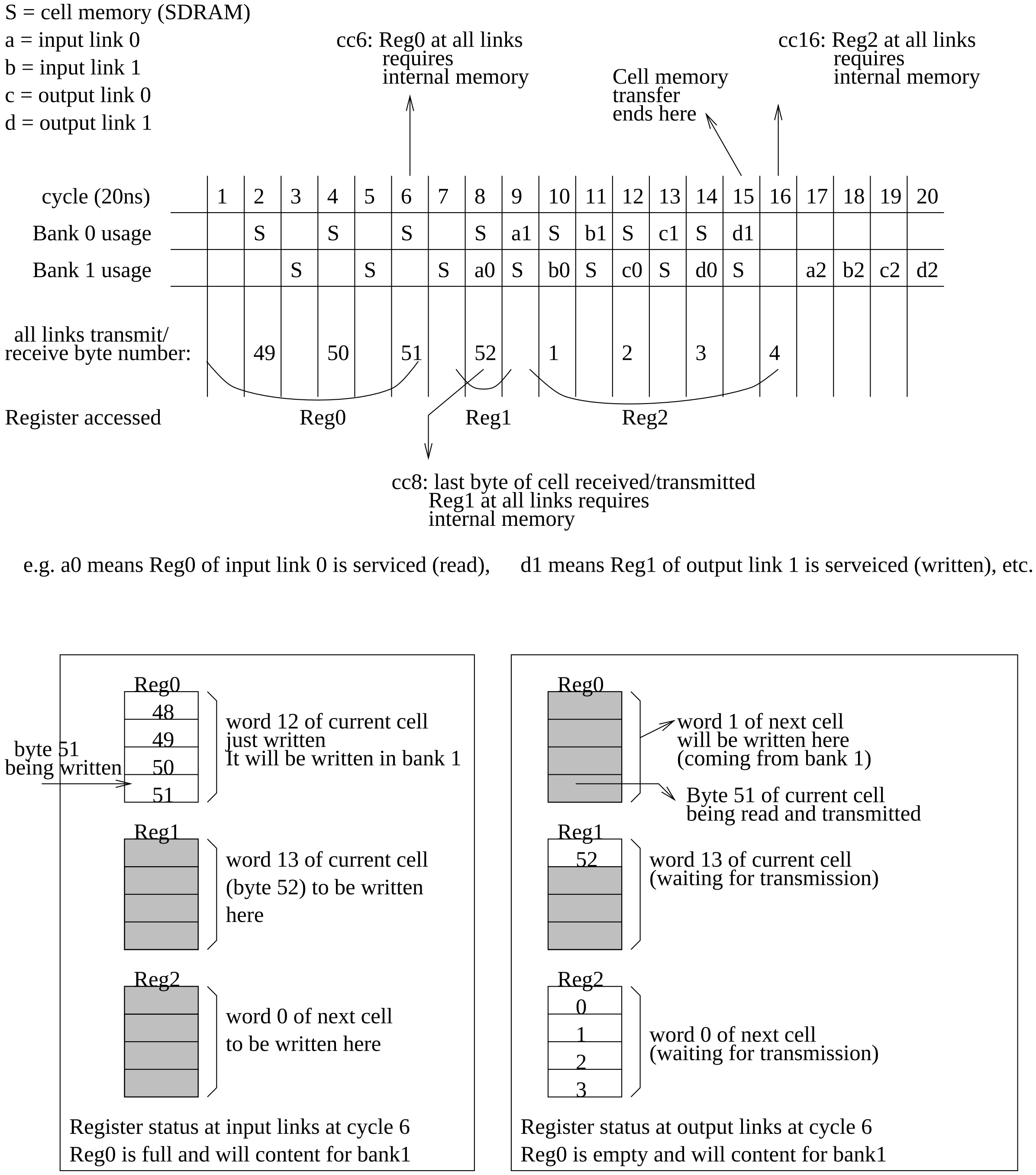

Figure 19: Worst case: all links require the same byte at the same time. The highest demand for internal memory occurs when they reach end-of-packet simultaneously.

6 The SDRAM Controller

In order to achieve the high speed requirements of SDRAMs (100MHz) and still keep the scheduler operating at 50MHz, a fast SDRAM controller is used to interface between the two devices. Operation of the scheduler at 100MHz was impossible since it resides in a quite large (thus relatively slow) FPGA. The SDRAM controller interfaces with a) the scheduler with a 32-bit wide data bus (called scheduler DataBus) and 3 control signals, and b) with the SDRAM with a 16-bit wide data bus and 5 control signals. Thus, the peak data throughput in both sides is 1.6Gbps. The controller's block diagram can be seen in figure 20.

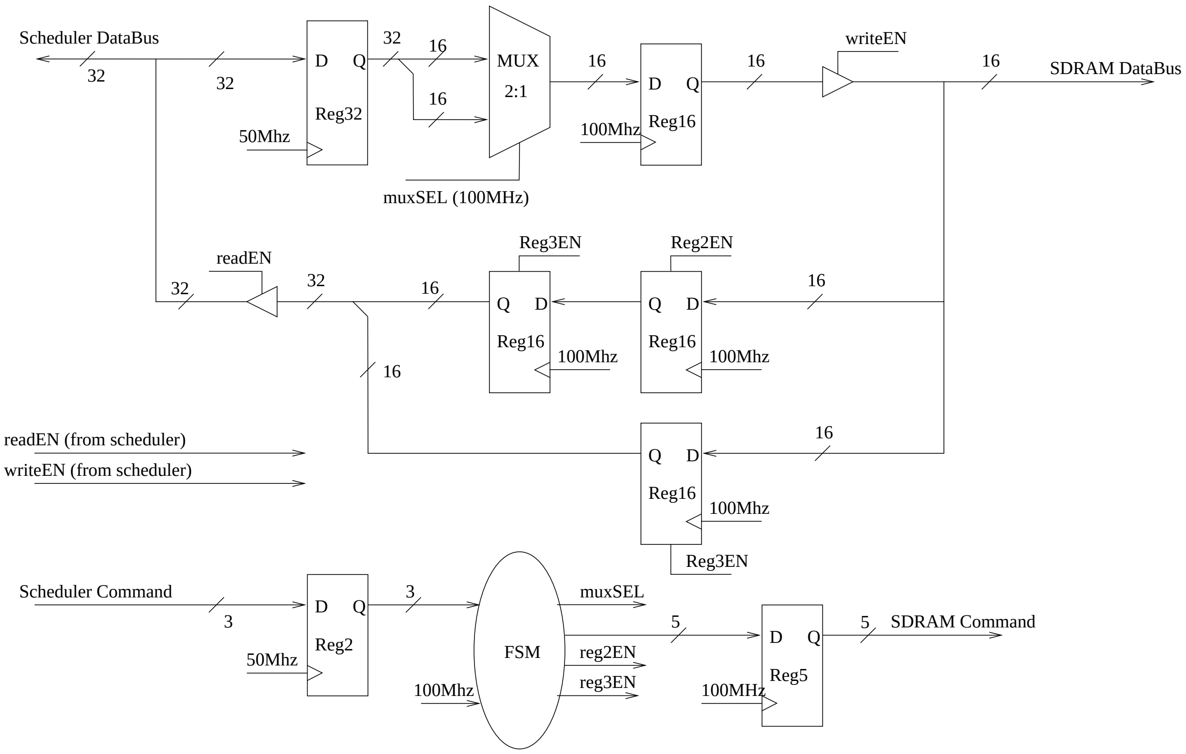

Figure 20: The block diagram of the controller.

The controller uses the 50MHz clock signal to synchronize with the scheduler. During a cell memory write, the controller accepts the 32-bit word to be written to cell memory from the scheduler. It also accepts the *Write* command. It then multiplexes the 32-bit word on the 16-bit data bus to the SDRAM, in two consecutive "fast" cycles (100MHz). It also outputs the *Write* command to the SDRAM. During a memory read operation the controller accepts a *Read* command from the scheduler and forwards it to the SDRAM. Then, it reads data from the memory bus for two consecutive cycles and outputs them on the scheduler bus. This data are kept steady (and valid) on the scheduler's data bus for two consecutive cycles, so that the "slower" scheduler can read them.

Achieving clock rates of 100MHz in modern FPGA's is not impossible. Yet, the FPGA which will operate at such frequencies, must be as small and simple as possible.

Efforts have been made to keep the SDRAM controller simple, even if these efforts would increase the scheduler's complexity. A couple of techniques, described below, were employed to keep the controller simple enough.

It is reminded that the row address to the SDRAM is given during the *Activate* command and the column address during the *Write* or *Read* command (see figure 8 and 9). But the *Read* or *Write* command is issued 3 cycles after the *Activate* command. That is, SDRAM address need not change every cycle, whereas data and command (control signals) do. Thus, address bits need not pass through the "fast" controller; instead they can come directly from the scheduler. This is possible because the scheduler is synchronized with the controller and knows exactly when the controller will issue commands to SDRAM. This is shown in figure 21. In this way we avoid the extra complexity and save I/O pins in the controller, because otherwise address buffering/forwarding over the faster bus should be performed by the controller.

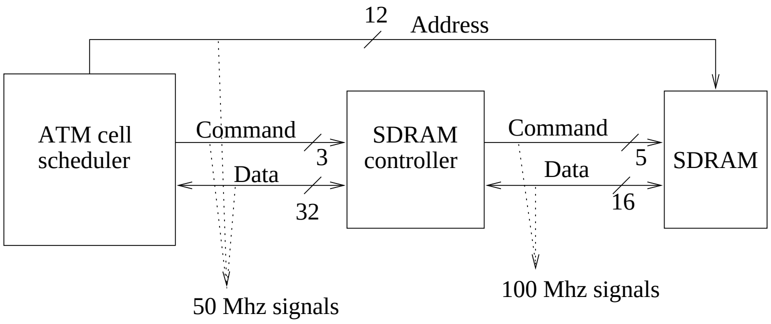

Figure 21: Address is given to SDRAM directly from the scheduler

In order to further reduce controller's complexity, refresh operation is left to the scheduler. Refresh is initiated by the scheduler and the controller just repeats it to the SDRAM. It is reminded that refresh must take place at least once every 1562 cycles, which demands a 11-bit counter. The controller was implemented in ALTERA's MAX7128S device [3], which has only 128-bits total buffer space; thus, more than 8% of the total buffer space was saved in this way. Finally, the *Mode Register Set* command is issued from the scheduler after power-up and the controller just repeats it to the SDRAM.

These efforts really kept the SDRAM controller simple, so operation at 113MHz was achieved during the simulation phase, exceeding our worst case requirements.

7 MicroProcessor Interface

The board interfaces to a microprocessor board for initialization and monitoring purposes. An i80C32 [7] based microprocessor board is used, which was developed at ICS-FORTH. Although i80C32 is old and slow, it suffices for our needs and avoids the complexity of modern processor interfaces. Due to the low speed at which the interface is running, the two boards are connected with just a ribbon cable (an easy and robust way of communication). Finally, the microprocessor board has been used extensively for a variety of applications in the past, so experience gained from previous work was quite helpful in our design.

The interface to microprocessor board involves: a 16-bit wide address bus, an 8-bit wide data bus and various control signals such as *Read*, *Write*, *PSEN* and an 8-bit unidirectional bus completely controlled by the processor. The microprocessor handles the transceiver's registers as external memory: the 8 LS bits of the address and the whole data bus is directly connected to the control interface of the SONET/SDH transceivers as can be seen in figure 22. It is reminded that this interface allows various operations on the transceiver such as monitoring, configuring and testing (see section 3.3). Microprocessor's interface does not include the *Select* signal required by the transceivers. For this reason, the 9th address bit was used; cell scheduler's logic and pins were used to decode this bit and select one of the two transceivers. Additionally, the *Read* and *Write* signals pass through the scheduler's logic to be converted to the *Read/Write_* signal required by each transceiver. Scheduler has been used to perform this conversion of the select signals, in order to avoid adding extra external logic.

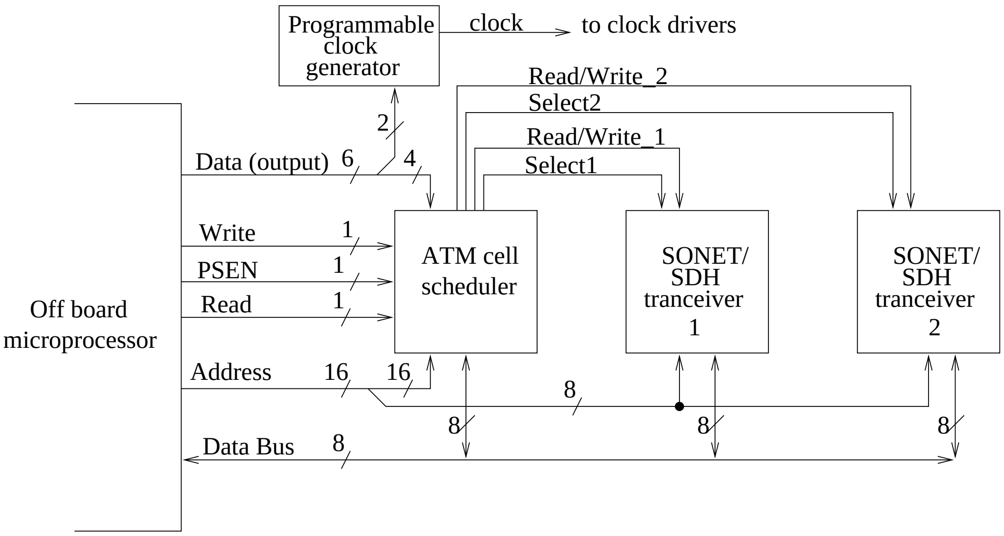

Figure 22: Communication with an off-board MicroProcessor.

Care has been taken so that the interface can be used for a variety of purposes, unforeseen at design time. For this reason, all microprocessor signals reach the cell scheduler, including 4 bits of the 8-bit unidirectional data bus, although they are not used at this time. It is reminded that the scheduler is implemented in a reprogrammable FPGA and can be redesigned (and reprogrammed) in the future to exploit these extra connections with the microprocessor. This communication has already been used (and was quite helpful) in the debugging phase of the design.

PSEN is used to identify memory accesses to the microprocessor's local memory and thus distinguishes them from the accesses to our board. Finally, 2 bits from the extra unidirectional data bus of the interface were used to program the programmable clock generator. All these can be seen in figure 22.

8 Board Implementation

The board implementation was quite complex, especially when compared with the operations it performs. The following reasons contribute to this:

- Not only is the board required to operate at high frequencies, but also it is expected to support five different frequencies. That is, there are simultaneously 155.52, 100, 50, 25 and 19.44MHz frequencies on the board.
- There exist 5V and 3.3V devices. SDRAM is a modern low voltage IC designed for low power and high speed. Additionally, SDRAM's clock input must be driven from a low voltage clock driver.
- There are analog parts in many of the components. Clock drivers have analog PLLs (Phase Locked Loop) to synchronize their output with their input frequencies. Analog PLLs are also used in the programmable clock generator and in SONET/SDH transceivers which recover the 155.52MHz clock from the bit-serial input. Also, the electro-optical converters use analog components to convert their inputs. Analog components are more vulnerable to noise than digital ones and thus, care must be taken to protect them when operating in noisy environments. A digital environment switching at 100MHz, as in our case, is quite noisy.
- The devices that were used in our design have high pin density. This raises the density of signal traces which must now be thinner and closer to each other, thus more susceptible to crosstalk.

In this direction a number of implementation issues must be addressed. These are analyzed in the following section.

8.1 High speed design techniques at the board level

The 100MHz requirement affected all aspects of the board design since it enforced very tight restrictions. The ATM cell scheduler uses pipeline techniques heavily to tear down the critical paths and improve its performance. The SDRAM controller, as already mentioned, was kept extremely simple to achieve the 100MHz speed. Besides the FPGA design, the rest of the board was thoroughly designed with high speed design techniques in mind. All ICs used, were chosen to exceed their expected performance, whereas board layout (clock distribution, device placement, power supply-ground) was designed with great care.

In general, the following rules must be followed when a high-speed switching environment must be designed [1] [2]:

Keep inductance low. In low speeds, resistance plays the most important role. In high speeds, inductance dominates over resistance. Inductance increases the impedance of the system, burdens current to flow and induces noise to the system, thus lowering performance. Traditional devices with long and thin pins appear increased inductance at high frequencies. That's why the devices used in our design were surface mount; surface mount devices have short pins as opposed to through hole devices that penetrate the board with their long pins.

Use proper bypass capacitors. High speed switching environments generate noise on power lines (or planes), due to charging and discharging of internal and external capacitors of integrated circuits. The instantaneous current generated by the rising and falling edges of the outputs causes power line ringing and ground bounce, endangering the proper operation of the system. The solution is the bypass capacitor, which stores electrical charge and releases it to the power line whenever a transient voltage spike occurs. In other words, it provides a low impedance power supply to the system, thereby minimizing noise. Every application needs a bypass capacitor of the proper capacitance value. Large capacitances suffer from increased inductance (and thus impedance) and cannot supply enough current at high frequencies. The smaller the capacitance value is, the higher is the frequency it can operate in. But small capacitances store little electrical charge, which may not be adequate for proper bypassing of the power lines. That's why a surface mount capacitor, with as small capacitance as our design requires, is the best solution. In our case, MLC (multilayer ceramic chip) capacitors were used, which have very small external inductance, due to their short leads, and negligible internal inductance. Additionally, capacitors must be placed as close as possible to the IC's power and ground pins, and must be equally distributed around the board. Finally, very large capacitors (hundreds of microfarads) are needed at the location where the power lines come into the board, because voltage fluctuations of the external power supply can be more severe and when they occur, they affect all devices on the board (they are not local to a fast switching device for example).

Keep the ground and power planes as close together as possible. Ground and power planes form a large capacitor; thus, when they are located close together, their capacitance increases. Large capacitance on these planes makes them less susceptible to voltage fluctuations and noise. Ground planes must not be located next to each other; they must be located next to the power planes to increase their capacitance. During the board's manufacturing, smaller (than typical) spacing between these planes was chosen, and both ground planes were located next to the power planes.

Use a proper termination scheme to avoid reflections. Nowadays, ICs have very fast edge rates (1ns typical) into 30pF (typical) capacitances. If the wires carrying such signals are electrically too long (the two-way propagation delay is longer than the rise or fall time of the signal they carry), then the signals are reflected at the destination and propagate back to the source. When it reaches the source, it distorts the signal and degrades the signal quality (overshoot or undershoot is observed). This reduces the noise margin of the digital circuits and creates electromagnetic interference (EMI). Clock signals are most sensitive to reflections because they are the most frequent switching signals and affect the whole operation of the devices. Generally, the characteristic impedance of the signal trace depends on: a)the width of the trace, b)the dielectric material around it (air, FR-4, etc.) and c) the distance from the power or ground plane that lies above or underneath. 50 Ohms is a typical characteristic impedance.

Thinner traces have higher characteristic impedance, whereas smaller distance from power or ground planes lowers characteristic impedance. Materials with higher dielectric constant around the trace, also lower the characteristic impedance of the trace.

The reflection is caused by the impedance mismatch between the signal trace and the receiver pin. That's why a proper terminating scheme is required, in which a *terminating resistor* of the same value as the signal's trace is placed close to the receiver pin. Thus, the impedance is the same all the way across the signal trace. A number of terminating schemes exist: shunt, AC, Thevenin, diode, series, all of them having advantages and disadvantages. In our case only the clock traces and the high-speed serial input/outputs of the transceivers were terminated. Thevenin termination was used because this was suggested by the manufacturers [10] [6].

Use controlled impedance traces. As already discussed, terminating resistors play a significant role in noise and electromagnetic interference suppression. But if the characteristic impedance of the signal trace is not known, no proper terminating resistor value can be selected. In our case only the traces carrying the high speed serial inputs/outputs of the SONET/SDH transceivers and the clock signals required to have known characteristic impedance (50 Ohms). Discussions with the board's manufacturer have been made to accurately define the width of the trace, the dielectric constant of the dielectric material and the spacing of the trace from the power/ground planes. Thus, the traces mentioned above, have 50 Ohms characteristic impedance.

Minimize crosstalk. Two signal traces traveling on the board parallel and close to each other are mutually affected by each other's electromagnetic field. This is called *crosstalk* and creates problems only at high speed signals, because low speed ones will have enough time to settle (till the next clock edge at which they will be sampled). Crosstalk is proportional to the total distance the signals travel parallel to each other and inversely proportional to their in-between spacing. Additionally, crosstalk can be minimized by using the ground and power planes as shielding layers. That is, the closer one signal is to its above (or underneath) plane, the weaker the electromagnetic field it radiates is. In our case, although some signals travel at a significant distance parallel to each other, their in-between spacing and their spacing to the shield layers has been chosen so that no problem with crosstalk will normally exist.

Protect analog circuits from noise. Analog or mixed analog-digital components are far more sensitive to noise than digital ones. This happens because analog components depend constantly on their inputs; instead, digital ones use their inputs only at the clock edge, when the signals will have already settled to a valid and steady value (the voltage swing will occur between the clock edges). Also, digital circuits induce much more noise than analog ones, because they switch outputs simultaneously at the clock edge, thus requiring sudden, high current flows. This sudden current flow creates voltage spikes, which can render the sensitive analog components inoperable. For this reason, filtering schemes using inductors and capacitors must be used to filter the power supply. Also, by using a split power and ground plane, one can isolate the digital from the analog supply in crucial locations of the board and thus minimize noise. In our case, different analog and power/ground planes were used in order to further isolate the two power supplies. The latter scheme, with the addition of both high and low frequency filters, forms an excellent protection mechanism for the analog components of the board.

Use low skew clock drivers to implement the clock distribution network. As clock frequencies become higher, clock signal quality becomes more and more important. In other words, as clock period get shorter, uncertainty and skew in the clock distribution network causes severer problems to the whole system. Besides clock skew, clock signal reflections degrade the signal quality and render the system unreliable. Thus the clock distribution system must guarantee minimum skew and minimum reflections. Clock skew is avoided if:

- Clock traces are all equal in length. The signal propagates down the wire with an (approximately) 130psec delay per inch, so if we need the clock edges to be very close to each other, we must design all traces to be of equal length.
- Clock outputs must be equally loaded. Larger capacitances affect the signal quality by making the edge rate lower; a large capacitance will be switched later than a small one. For a high speed design, capacitive loads must not differ by more than 5pF, or else clock skew is induced.
- Clock traces must be of equal characteristic impedance. Different impedances will demand different terminating resistors, and different resistors will affect the driving strength of the clock output. Smaller resistors will demand a stronger clock driver, or in other words, the edge rate of a clock trace terminated to a small resistor will be much lower (or slower) than expected. Additionally, due to the fact that signal traces passing through the board (inner signal layer) have lower characteristic impedance, they must be designed to be thinner (thin traces generally have higher characteristic impedance) to compensate for the impedance loss. Clock traces on the same layer must only be of equal width.

In our case, guaranteed skew clock drivers were used. IDT's PLL based 74FCT388915T-100 [6] device guarantees 250psec skew across all outputs of the same device, even if these outputs switch at different (but with synchronized edges) frequencies. This driver is able to produce simultaneously x1, x2 and /2 the internal VCO (voltage controlled oscillator) frequency, which can either be x1, x2 or /2 the input frequency. Thus, the driver's output frequency ranges from /4 to x4 the input frequency, but all these frequencies are not available at the same time. The clock distribution network can be seen in figure 23. As can be seen in this figure we need to have 1 output at 100MHz, 2 outputs at 50MHz and 3 outputs at 25MHz (in this figure we assume a 25MHz output from the programmable clock generator; other frequencies can be available by reprogramming through the program interface). All outputs are almost equally loaded (load on every signal is shown in parentheses). Since the above driver cannot support that many outputs, two such drivers were used. They were driven by the same clock source, thus all outputs coming from them had also synchronized edges. When more than one device is used at the same time, the manufacturer guarantees that the introduced skew will be below 1ns across any two devices, which satisfies our requirements. It must be noted that the 100 and 50MHz signals come from the same device (thus have only 250psec skew) and the second device is used to derive the slow 25MHz signals. 1ns skew for the 10ns and 20ns cycle time of our ”fast” clocks would be disastrous.

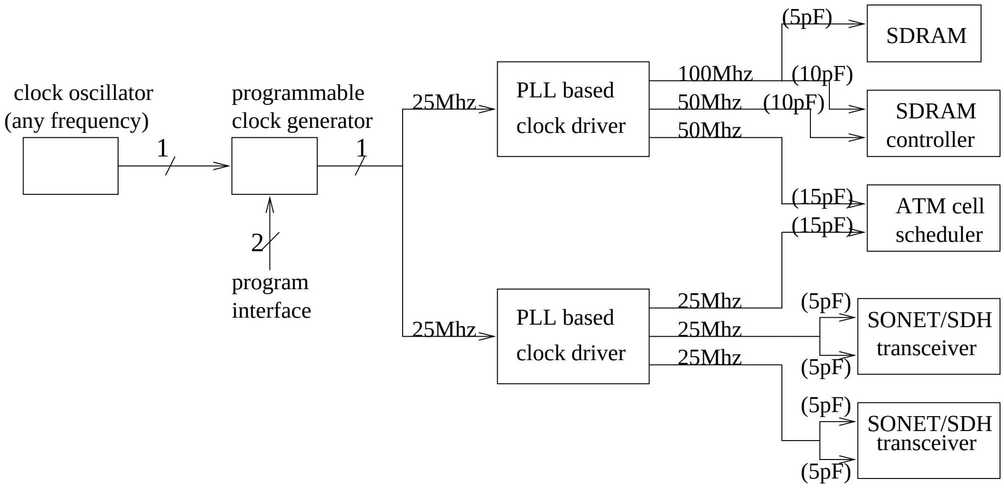

Figure 23: Clock distribution network.

8.2 Design flow

The ATM scheduler and the SDRAM controller are implemented in ALTERA’s FPGA’s. First these two FPGA’s were designed in ALTERA’s MAX+Plus II development tool. This is a highly user-friendly tool that allowed the design to complete in only a few months, although there was no previous experience in ALTERA’s devices. Throughout our design, ALTERA’s HDL programming language (AHDL) was used, together with a schematic editor; VHDL programming language was not used although supported by the tool. The ATM cell scheduler was implemented in FLEX10K40RC-208 device [3] which has 16Kbits on-chip memory and 40000 usable gates. 1800 LEs (logic elements) out of 2300 total were used (78% usage). Many unused pins were connected on the board and reserved for future use, and 8 pins were left unconnected. This gave a 96.16% usage out of total 208 pins of the device. 1303 lines of AHDL code were written combined with a little schematic design. The SDRAM controller was implemented in MAX7128SLC-84 device [3]; 121 out of 128 LEs total were used (94.53% usage), whereas none of its pins were left unconnected (few unused pins were reserved for future use). The AHDL code counts 144 lines only; again, a little schematic design was employed.

Next, a small number of simulations were carried out in order to quickly verify the functionality and performance of the chosen architecture. The cell scheduler and the SDRAM controller were simulated separately (one at a time). They were not simulated as a unit, because this was considered useless without the co-existence of the SDRAM model in the simulations (the MAX+Plus II environment did not allow for a simulation model of the SDRAM to be written). For this reason, Verilog-XL HDL from Cadence was used. This language allows the functional description of a circuit independent of the underlying technology. After the simulation model of the SDRAM was written in Verilog-XL, the cell scheduler and the SDRAM controller were also translated to Verilog-XL, using the MAX+Plus II verilog translator. All three devices were simulated together to verify the correct operation of the design. During the simulation phase, approximately 2 million ATM cells passed through the ATM scheduler (with various traffic patterns) without any mistake to be observed.

8.3 PCB Characteristics

The design of the printed circuit board (PCB) was the next step. Allegro v11.0 tool from Cadence was used. The final board is rectangular, 6.76in x 4.42in. It has 8 layers: 3 signal, 3 power and 2 ground. As already mentioned, besides one power and one ground plane that were used for the digital parts of the board, one extra power-ground plane pair was used for the analog parts. The third power plane powers the low voltage devices at 3.3V; no voltage regulator was used, because this would demand isolating so many parts of the 5V power plane that it would make the design far too complex. Employing extra planes for the analog parts is a conservative and expensive technique, but it makes the design more robust. On the other hand, 3 signal layers are considered quite few for such a dense design. Keeping the number of signal layers to 3 demanded that the whole routing procedure would be made manually. The Allegro router is a simple and cheap one and could not possibly complete the routing process in just 3 layers.

The board has 8 mil (1mil = 1 thousandth of the inch) minimum line width, 10 mil minimum spacing between lines and 25 mil minimum via diameter. The PCB has a total of 616 through vias (no blind or buried vias were used), 1309 pins (both surface mount and through hole), 420 signal connections and 577 connections to power supply and ground. These figures make the board dense enough. Thus, components are very close to each other, the total size of the board becomes smaller and signal reflections are minimized.

The layer stackup from top to bottom is:
TOP (signal)
Digital GND
Digital VCC 5V
INT (signal)
Digital VCC 3.3V
Analog GND
Analog VCC 5V

BOTTOM (signal).

Two clock oscillators and a total of 14 ICs were used: 2 FPGA's, 1 SDRAM, 4 inverters, 2 clock drivers, 1 programmable clock generator, 2 SONET/SDH transceivers, 2 electro-optical converters. Also, a total of 132 capacitors, 9 inductors and 64 resistors perform power filtering, signal termination and bypassing. The large number of passive devices makes the design more robust and allows us to exceed the required specifications (100MHz). Our goal was to reach as higher a frequency as possible.

The board's actual manufacturing took place in Stevenage Circuits Ltd. in Great Britain. This company also took over the assembly of the board because our equipment in ICS-FORTH was not able to assemble the high-pin dense FLEX10K40RC-208 (cell scheduler) and the TNETA1500-PCM (SONET/SDH transceivers). The exact way of calculating the manufacturing cost differs from one manufacturer to another, but the main factors raising cost are: large number of layers, existence of blind or buried vias, thin traces, dense traces, large number of pads, dense pads, large number of different pads, large number of vias, large number of different vias, small spacing between layers.

The total cost of the board was $2,150. The manufacturing cost $1,430, the assembly $120 and the materials $600. The manufacturing cost was high due to a) large number of layers (8 layers), b) thin traces (8 mil), c) small spacing between layers (5 mil instead of 7.5 mil typical), d) component pads on both sides of the board, e) 5 different via sizes f) large number of component pads and g) very dense pads of the FLEX10K40RC208 (ATM cell scheduler).

9 Board Testing and Results

After the manufacturing and assembly of the board by Stevenage Circuits Ltd. in Great Britain, we tested the board. After some initial verifications that no short-circuit existed and that the resistance between power supply and ground was reasonably high, the board was powered up. First, the clock signals were checked with the oscilloscope to verify that they provided the board with the appropriate frequencies. Then, the FPGA's were loaded with a simple program to test if they were correctly programmed. A few test points on the board, driven by FPGA's outputs, were sampled against some predefined values.

The next step was to check if the SDRAM was functional. A hardwired pattern was continuously written by the scheduler (through the controller) in consecutive locations in memory and then read back. The successful comparison of the data read back, proved that the SDRAM was initialized properly and operated well at the 100MHz frequency (maximum specification of the device).

A number of loopback tests followed. As already mentioned in section 3.3 (also see figure 5) the SONET/SDH transceiver can operate in two loopback modes: *terminal* and *facility*. During the performed *terminal* loopback, a hardwired pattern was transmitted by the scheduler to the transceiver's transmit FIFO; this pattern was compared against the looped-back data. This test verified that a well established connection between the scheduler and the transceiver, over the UTOPIA bus, existed. During the facility loopback test that followed, the correct interconnection between the electro-optical converters and the transceivers proved to be correct. This was the most crucial part of the tests because it checked the fastest signals on the board (155.52MHz). The incoming traffic was produced in a SparcStation 10 workstation connected to our board through an ATM switch, while the SONET transceiver was placed in the *facility* loopback mode.

All the above tests checked and verified the interconnection between any two devices. The board's functionality as a whole and without any loopback mode was to be examined next. The input links were again connected to two ports of the ATM switch and received ATM traffic from two workstations simultaneously. During these final tests, the board received ATM traffic at the mean rate of 75Mbit/sec from each input link. The ATM cell scheduler was instructed to forward the incoming traffic (through the cell memory) from input link 0 to output link 0 and the traffic from input link 1 to output link 1. Each output link was connected to the ATM switch which sent the traffic back to the workstation it came from. Returned data was compared against the data sent, revealing that the board is operating correctly. Approximately 70GBytes of data passed through the board for over an hour of continuous operation, during these tests, without any error to be observed.

It must be noted that the 75Mbit/sec data rate was the upper limit during all the above tests, because the workstation is slow and cannot receive and transmit at 155.52Mbit/s speed simultaneously. In order to overcome this, the next scheme was employed: Both workstations were ordered only to transmit data and not to receive, achieving rates of 110Mbit/s. Both these traffics (around 220Mbit/s) were forwarded to the same output link, congesting it. Then, the corresponding queue of the cell memory (SDRAM) overflew and packets were lost. Monitoring with the oscilloscope allowed us to verify that SDRAM was continuously becoming full throughout these transmissions.

Next, one workstation was ordered to transmit only (achieving 110Mbit/s), while the other both to transmit (achieving 75Mbit/s) and receive data. Both incoming traffics on the board were directed to the same output link (the one the workstation received data back). Again, the total throughput required was over the link capacity, so the cell memory overflew and packets were lost. This was verified by the workstation which reported this packet loss.

The testing and debugging of the board revealed only one minor mistake. Both clock drivers had their *reset* pin connected to the reset circuitry of the board. That is, when the board was reset through the reset button, the clock signals went down into inactive state. Due to the fact that the clock drivers are PLL based devices, the clock signals were reactivated quite some time after the reset button was released, and when all other devices had already begun to initialize. But without a valid clock signal they did not initialize properly and they entered illegal states of operation. This was revealed because although the board was completely functional after power up, it did not operate at all after a reset operation. This was easily corrected, by forcing the reset input pin of the clock drivers to always stay at the inactive state (disconnected from the reset circuitry and then connected to the power supply), so that the clocks were constantly running.

10 Conclusions and future work

Through this work we conclude that developing a high throughput off-chip buffer for ATM switches is possible by employing modern high-speed techniques and by using synchronous dynamic memories (SDRAM) for cell buffering. Reprogrammable FPGA's and commercial ICs as physical interfaces with SONET offer flexibility to the design and rapid prototyping. In this work, the prototype board manufactured is an 1.2Gbit/s ATM cell buffer/forwarder that uses a 16Mbit SDRAM chip. The board is 6.76in x 4.42in in size, uses FPGA's to operate on ATM cells, SDRAM to buffer cells and ICs to interface with four SONET/SDH 155.52MHz links. More complex architectures can be designed and tested, without changing the board; only the FPGA's need to be reprogrammed. Thus, future work can be easily verified on this board with little design time and no material cost. Many aspects of future architectures can first be tested on this board prior to implementation as an ASIC for achieving higher speeds.

The ATM cell buffer/forwarder that was presented here operates internally at 100MHz maximum and 77.76MHz minimum in order to achieve the high throughput requirements of servicing four 155.52Mbps SONET/SDH links that carry ATM traffic, with only one SDRAM chip. These memories use synchronous (clocked) interface to communicate with other devices, they provide high throughput by employing burst techniques and still remain inexpensive. In our design, two logical queues (FIFOs) are maintained in this memory (one for every output link) and cells are enqueued in them, as they arrive from input links, according to their VP/VC values. Cells are dequeued from these SDRAM queues and transmitted through the output links.

The design's operation was thoroughly tested through extensive simulations at gate level prior to being sent for manufacturing. Excessive testing was also performed in the laboratory after manufacturing proving that the board was fully functional. The final board achieved the clock rate of 100MHz at 25C ambient temperature without any errors being observed throughout tests. Operation at higher frequencies has not been attempted till now, because the 100MHz was the maximum required frequency. Future tests may reveal the design's frequency limits.

Future work can add a VP/VC translation table to the design. We did not implement a translation table because we focused on the simplicity and speed of the hardware. Yet, this table could either be static (decided at programming time) using PVCs, or dynamic by accepting *update* commands from the micro controller. Another extension could be the addition of cut-through capability. This would minimize ATM cell delays when the queues are empty and thus, no internal buffering is needed in that case.

11 Acknowledgements

I would like to thank all those who have contributed to this work. First, my professor Manolis Katevenis for his overall guidance. George Kalokerinos helped me most with almost every aspect of the board design. Also, I would like to thank Mixalhs Ligerakhs who helped me with the board assembly and the board test. Also, Manolis Stratakhs who guided me through the Allegro tool.

Next, I would like to thank Spyros Papadhmhtriou who wrote the simulation model of the SDRAM used during the simulations, and Fotis Kitsos who set up the ATM switch environment used during the tests.

Finally, I wish to thank Institute of Computer Science (ICS) of the Foundation for Research and Technology (FORTH) for the financial and technical support throughout this design. This work was also financially supported by the European Union ACTS (Advanced Communication Technologies and Services) program, through the ASIC-COM (ATM Switch for Integrated Communication, COmputation and Monitoring) project.

References

[1] Johnson, W. Howard, and M.Graham, "High speed digital design. A Handbook of Black Magic." Englewood Cliffs, NJ:Prentice Hall, 1993.

[2] Montrose, I. Mark, "Printed Circuit Board Design Techniques for EMC Compliance", IEEE Press, 1995.

[3] ALTERA, "Data Book" 1996.

[4] ALTERA, "MAX+PLUS II Getting Started".

[5] AMP, "269039-1 datasheet" at http://www.amp.com/catalog/index.html/ (registration needed).

[6] IDT, "74FCT388915 datasheet" at http://www.idt.com/docs/3052.pdf, 1995.

[7] INTEL, "Embedded Microcontrollers and Processors (Volume I)", 1992.

[8] NEC, "µPD4516161 datasheet" available at ICS-FORTH: /vlsi/projects/glykop/uPD4516161.pdf

[9] PMC-SIERRA, "S/UNI-LITE OPTICAL REFERENCE DESIGN (SORD)", 1995.

[10] Texas Instruments, "TNETA1500 datasheet" at: http://www-s.ti.com/sc/psheets/sdns021d/sdns021d.pdf, 1998.

[11] Texas Instruments, "TNETA1560/TNETA1500 SBUS Reference Schematic Notebook", 1995.

[12] Texas Instruments, "TNETA1561/TNETA1500 PCI Reference Schematic Notebook", 1995.

Appendices

A Reference designators

The following reference designators can be seen in the schematics of the board in appendix B:

Ref. Des.	Description
U1	ATM Cell Scheduler (FPGA)
U2	SDRAM controller (FPGA)
U3	SDRAM 16Mbits
U4	SONET transceiver 1
U5	SONET transceiver 2
U6	Electro-optical converter for transceiver 1
U7	Electro-optical converter for transceiver 2
U8	19.44MHz clock oscillator (for transceivers)
U9	clock oscillator (any freq. from 19.44 to 25MHz)
U10	programmable clock generator
U11	clock driver (50,100MHz outputs)
U12	clock driver (25MHz outputs)
U13	inverter (reset pins driving)
U14	reset button
U15 to U17	inverters (LEDs driving)
U18	Ledbar1
U19	Ledbar2
U20 to U23	Resistor Array, 220Ohm
C1 to C4	470uF capacitor
C5 to C11 C14 C16 to C22 C94 C106	10uF capacitor
C12 C13	4.7uF capacitor
C23 C127 C128 C130	1uF capacitor

continued in the next page

Ref. Des.	Description
C37 to C95 C97 to C105 C107 to C126	0.1uF capacitor
C24 to C29	0.01uF capacitor
C30 to C36 C129 C131 to C133	0.001uF capacitor
R1 to R10	82.5 Ohms resistor
R28 R29 R43 R44 R53 R54	100 Ohms resistor
R11 to R20	130 Ohms resistor 1/8 Watt
R21	130 Ohms resistor 1/4 Watt
R22 R23 to R27 R35 to R38 R45 to R48 R62 to R66	1 KOhms resistor
R39 R49	1.3 KOhms resistor
R40 R50	3.74 KOhms resistor
R30 to R32 R34 R41 to R42 R52 R55 to R61	4.74 KOhms resistor
L1 to L5	Ferrite bead (short lead)
L6 to L9	Ferrite bead (long lead)

Table 2: Reference designators of all components

B Schematics of the printed circuit board

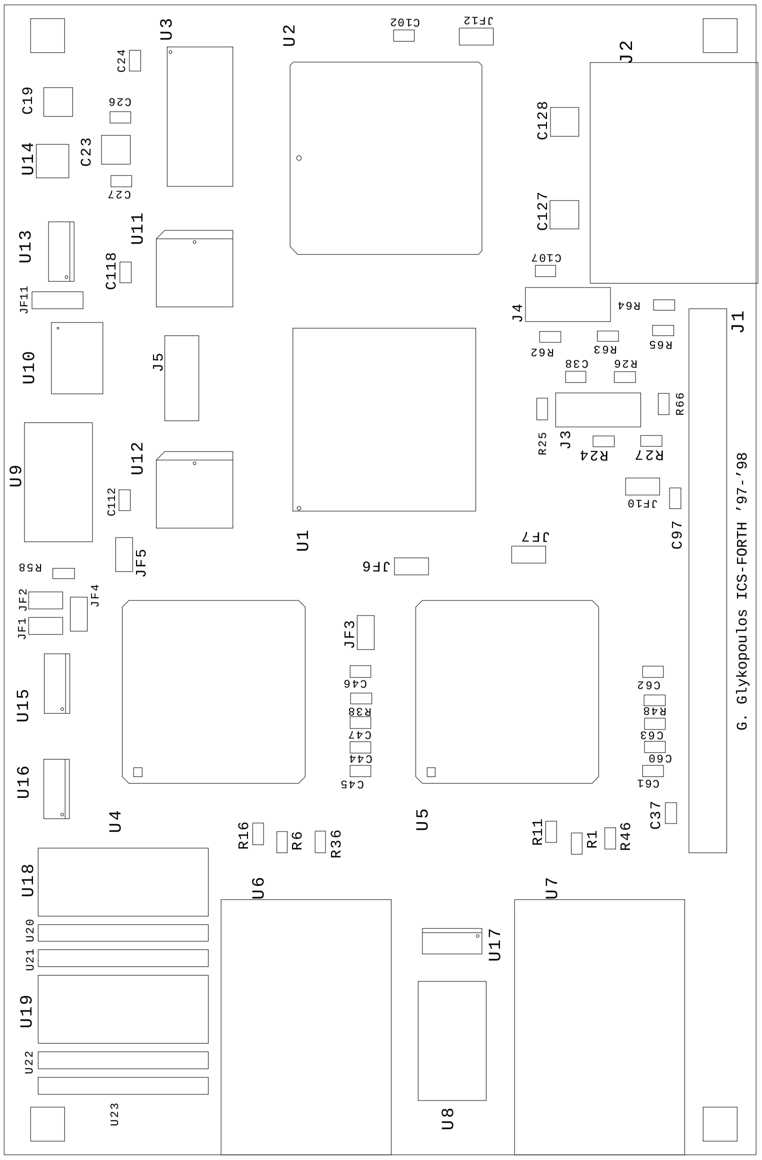

Figure 24: Top layer of the board: reference designators

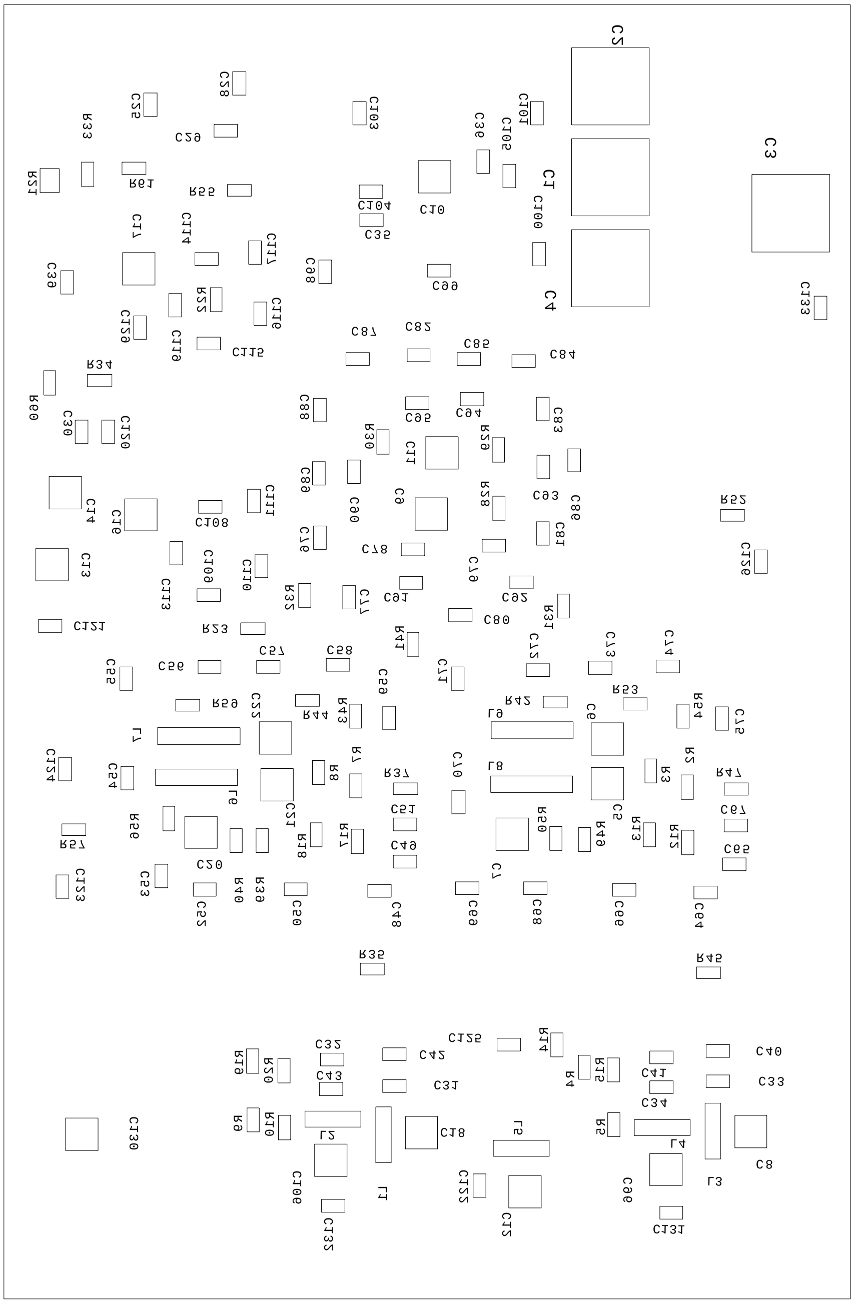

Figure 25: Bottom layer of the board: reference designators

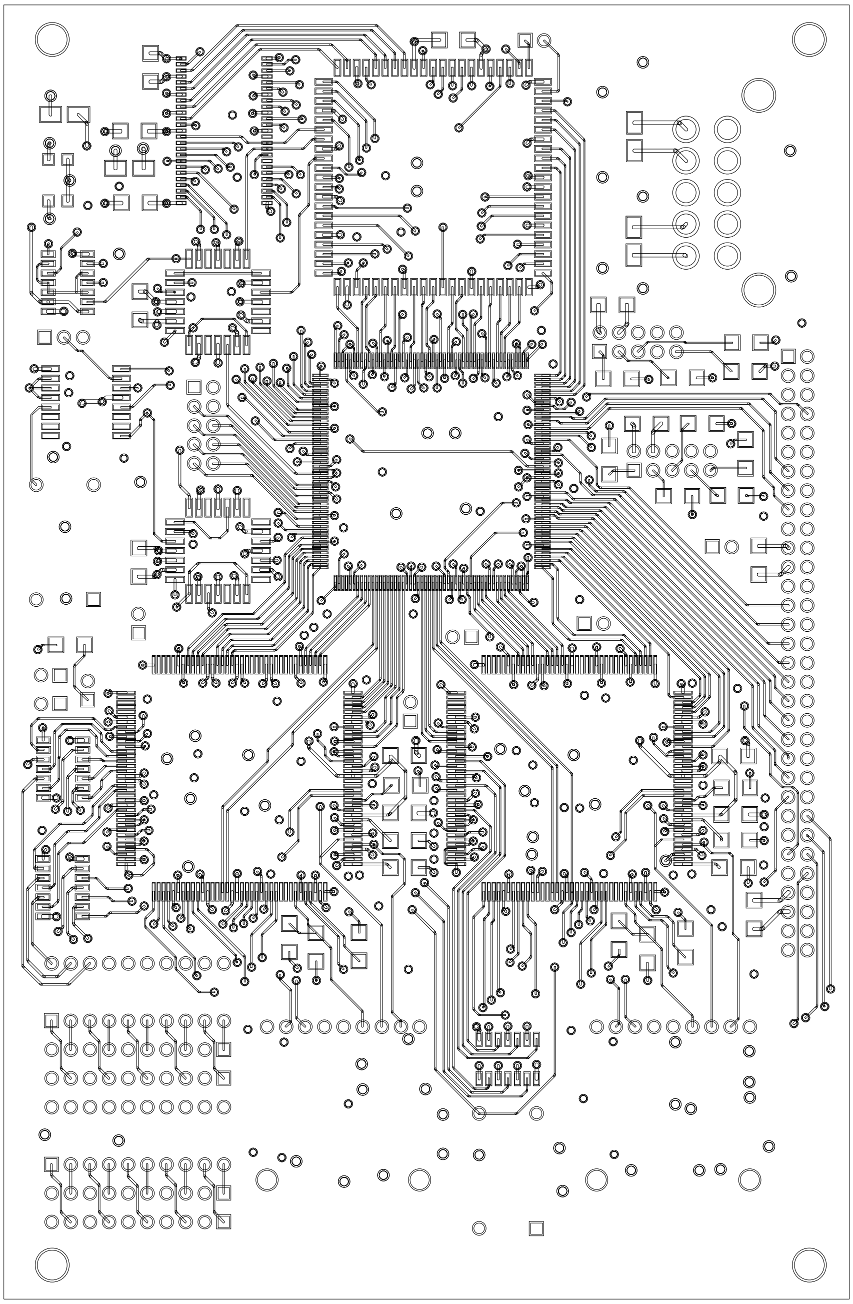

Figure 26: Top layer of the board: etch

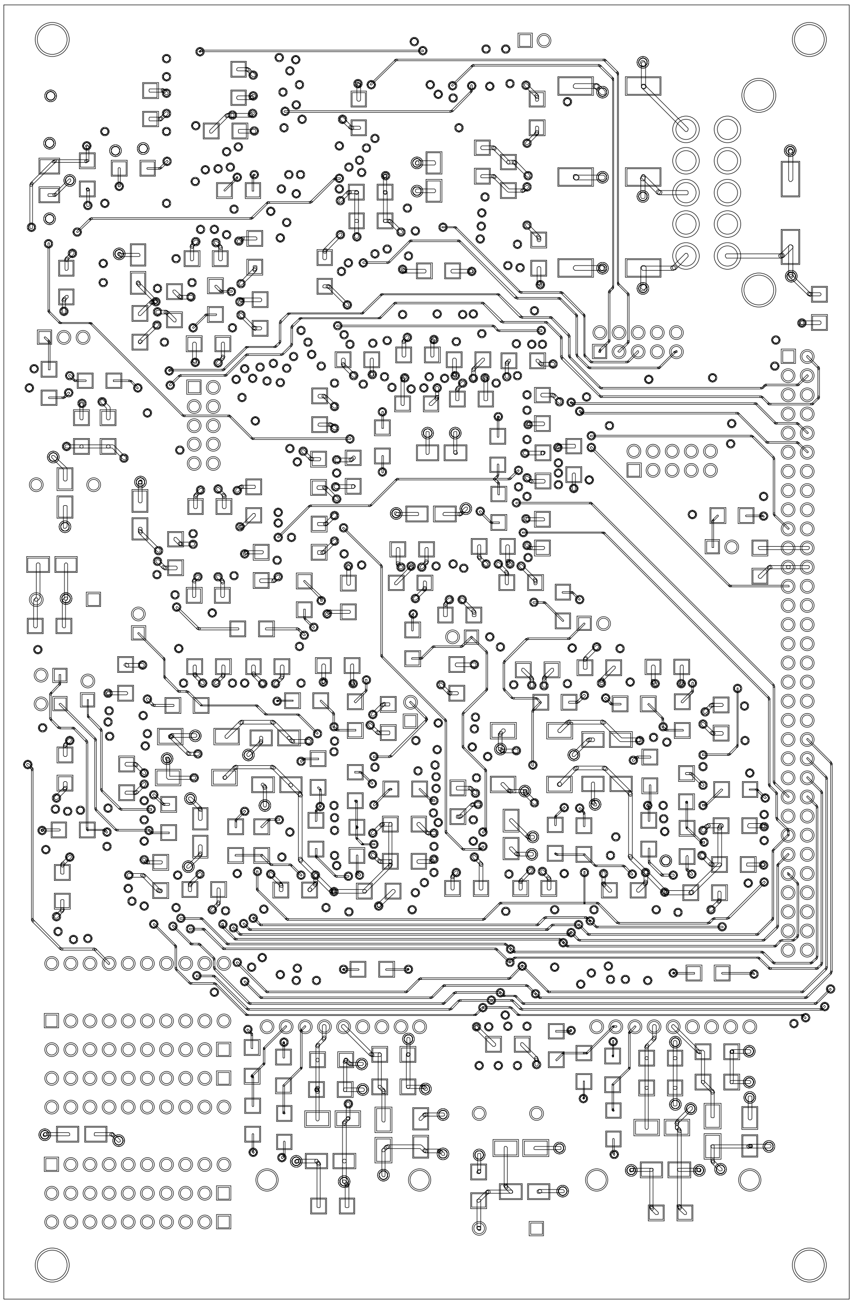

Figure 27: Bottom layer of the board: etch

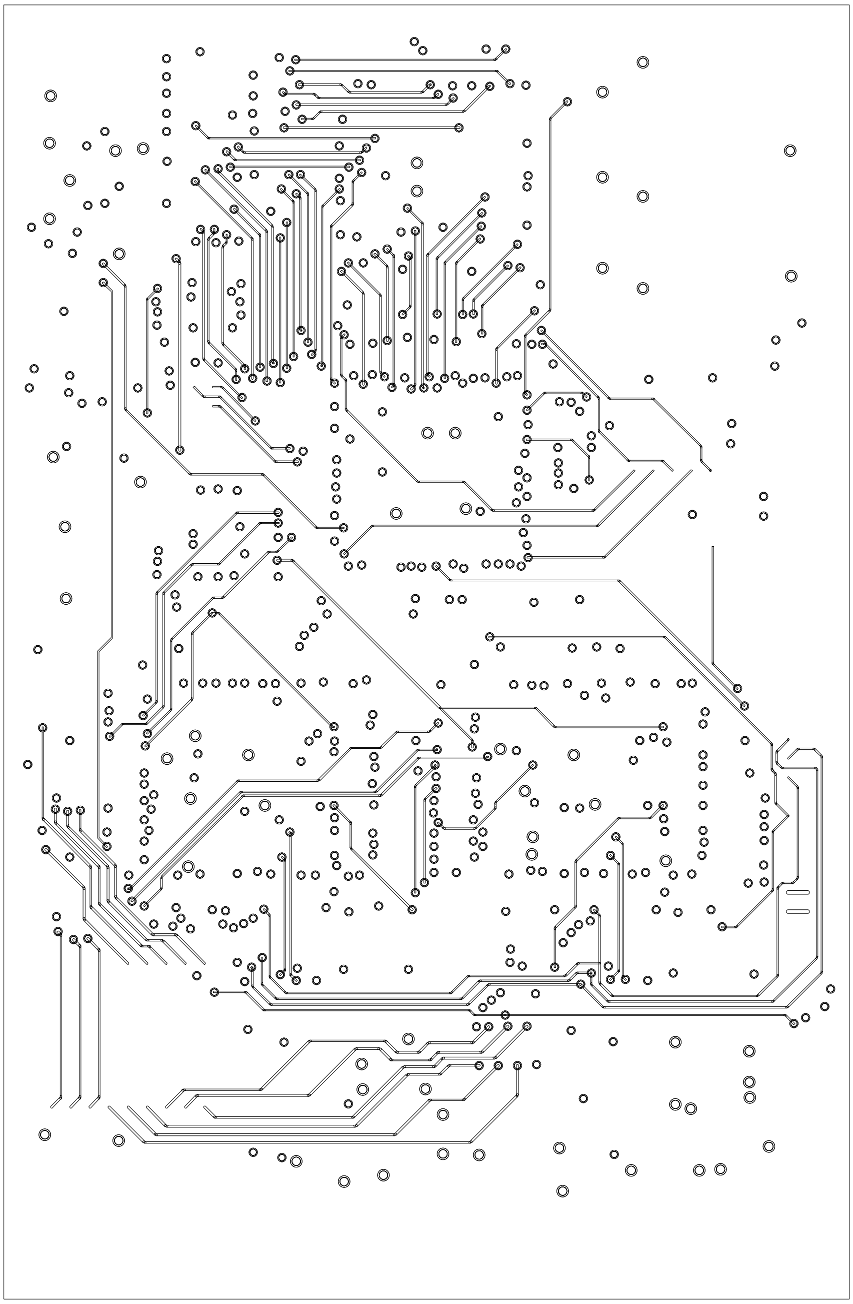

Figure 28: Internal layer of the board: etch

C Board operation and programming manual

Throughout this manual, consider the directions of the board (*up,down,left,right*) to be the directions of the label ”G. Glykopoulos ICS - FORTH ’97 - ’98”. (Consider the board position so as this label can be read normally).

C.1 Power supply issues

Always power all three voltage levels on the board, which come onto the board through the power connector (J2) at the following positions (from left to right): Analog 5V, Analog GND, Digital 5V, Digital GND, Digital 3.3V.

The analog 5V must be connected (short-circuit) with the digital 5V **before** reaching the power connector, because these two voltage levels must always be of the same value (small deviations may result in damaging devices).

The analog GND must be connected (short-circuit) with the digital GND **before** reaching the power connector.

C.2 Jumper positions

The following jumpers have the following operation when shorted (listed by their reference designator on the board):
JF1: Activates facility loopback on SONET transceiver 1.
JF2: Activates terminal loopback on SONET transceiver 1.
JF3: Activates SDH (instead of SONET) on SONET transceiver 1.
JF4: Activates loop timing on SONET transceiver 1.
JF5: Tri-states all TTL outputs of SONET transceiver 1.
JF6: Activates facility loopback on SONET transceiver 2.
JF7: Activates terminal loopback on SONET transceiver 2.
JF8,JF9: Do not exist.
JF10: Tri-states all TTL outputs of SONET transceiver 2.

One of **JF5** and **JPF10** must be shorted before programming the ATM cell scheduler. Both transceivers are on the same data bus and contention may result if their outputs are enabled at the same time. After programming the cell scheduler, it is guaranteed that they will not drive this bus at the same time, so both JF5 and JF10 can be left open.

The 3-way jumper **JF11**, when set to GND, sets the output of the programmable clock generator to the input frequency it gets. When set to VCC, it sets the output of the programmable clock generator to the programmed frequency. The VCC pin is the one closer to the edge of the board, whereas the GND pin is the innermost (towards the inner side of the board) pin. One of these pins must be always shorted to the middle one (it must never be left open).

JF12 is not a jumper. NEVER short JF12.

C.3 FPGA programming

U2 (SDRAM controller) is an EEPROM-based FPGA and is programmed through connector J4. It does not need reprogramming after power down. This device can be reprogrammed around 100 times. At the time of this report being written, it has already been reprogrammed 7 times.

U1 (ATM cell scheduler) is an SRAM based FPGA and is configured through connector J3. It can be configured (”configure” is the corresponding word for ”program” on a SRAM device) an unlimited number of times. It must be reconfigured every time after power up because it looses configuration when the power goes down.

Both FPGA’s (U1 and U2) can be reconfigured/reprogrammed during operation of the board. A reset operation may be needed after this.

In order to program the FPGA’s:

1. Connect the ByteBlaster cable to both the parallel port of a PC and on the corresponding 10-pin header on the board (J3 for cell scheduler, J4 for SDRAM controller).

2. Open the MAX+PLUS II environment, compile a design and then run the programmer. For more information see [4] chapter 4. All design and program files of the current design are available at ICS-FORTH:
/vlsi/projects/glykop/pld1.zip (ATM cell scheduler) and
/vlsi/projects/glykop/pld2.zip (SDRAM controller).
See README file in each zip file for more instructions.

3. One FPGA must be programmed at a time. In order to switch the ByteBlaster between J3 and J4, the board must be powered down. Since the ATM cell scheduler (U1) is an SRAM based device, it looses configuration after power down; thus, the SDRAM controller must be programmed first, then the board must be powered down and finally the cell scheduler can be configured.

4. When the programmer is running, press the ”configure” button (for the cell scheduler) or the ”program” button (for the SDRAM controller). A ”configuration complete” message should appear.

5. A reset operation may be needed for the correct initialization and operation of the loaded programs.

For more information on programming ALTERA FPGA’s, see [4] and [3].

NOTE: FPGA’s do not need reprogramming/reconfiguration after a reset operation.

IMPORTANT NOTE:
When reprogramming/reconfiguring the FPGAs, care must be taken as far as pin type is concerned (input, output or bidirectional pin). Pins on different FPGA’s (or,

generally, on different ICs) that are connected together, must never be programmed as outputs at the same time. All connections between pins can be seen in the Concept diagram of the design, or in the Allegro diagram or in the Allegro report files.

C.4 Microcontroller interface

A number of operations can be made on the board through the i80C32 [7] microprocessor board, such as reading and writing the transceivers' internal registers. In order to do this:

- Connect the board to the i80C32 microprocessor board through a 64-bit flat cable and through a 2x32 connector.
- Connect the microprocessor board to the EPROM emulator. The EPROM emulator must be connected on a PC serial port. The EPROM emulator downloads from the PC the program to be run by i80C32.
- Run program "ggboard.exe" on the PC connected to the EPROM emulator. A menu appears; follow the instructions appearing on screen. The program "ggboard.exe" is available at ICS-FORTH: /vlsi/projects/glykop/ggboard.exe

For more information on this interface see appendix C.5.

C.5 Connectors

JF12 is not a jumper. Its pins are connected to pins 52 and 54 of U2 (SDRAM controller), and are used for debugging purposes. NEVER short JF12.

J1 is the interface with the micro-controller. Connect it through a flat cable and with a 2x32 pin connector to the i80C32 board. The pin numbering of J1 connector can be seen in figure 29 (top view of the connector).

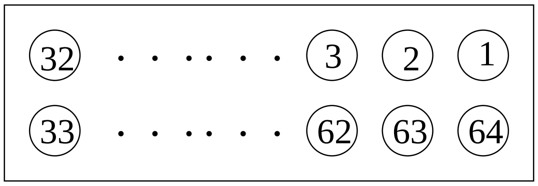

Figure 29: Numbering of J1 connector pins (top view).

Pin assignment of J1 is shown in table 3.

J2 is the power connector. Its circuits are (from left to right): Analog 5V, Analog GND, Digital 5V, Digital GND, Digital 3.3V.

J3 programs U1 (ATM cell scheduler). The ByteBlaster cable must be connected with pin #1 towards the inner part of the board (or towards U1, see appendix B).

J4 programs U2 (SDRAM controller). The ByteBlaster cable must be connected with pin #1 towards the inner part of the board (or towards U1, see appendix B).

J5 is a 10-pin header, used for debugging purposes. Its pins are connected to pins 195, 193, 192, 191, 189, 187, 186, 180, 175, 174 of U1 (ATM cell scheduler).

Pin num.	Signal name	Pin num.	Signal name
1	GND	33	NC
2	NC	34	NC
3	NC	35	GND
4	VCC 5V	36	VCC 5V
5	NC	37	Data 0
6	NC	38	Data 1
7	Read_	39	Data 2
8	Write_	40	Data 3
9	ALE	41	Data 4
10	PSEN	42	Data 5
11	VCC	43	Data 6
12	GND	44	Data 7
13	Addr 15	45	NC
14	Addr 14	46	NC
15	Addr 13	47	NC
16	Addr 12	48	GND
17	Addr 11	49	NC
18	Addr 10	50	NC
19	Addr 9	51	NC
20	Addr 8	52	NC
21	Addr 7	53	GND 5V
22	Addr 6	54	VCC 5V
23	Addr 5	55	NC
24	Addr 4	56	NC
25	Addr 3	57	NC
26	Addr 2	58	NC
27	Addr 1	59	Reserved connection to U1 pin 94
28	Addr 0	60	Reserved connection to U1 pin 96
29	VCC 5V	61	Reserved connection to U1 pin 95
30	GND	62	NC
31	NC	63	clk generator data
32	NC	64	clk generator sclk

Table 3: J1 connector pin assignment.

C.6 LEDs

LEDBAR 1 and LEDBAR 2 indicate the alarms generated from the two transceivers. LEDBAR 1 (at the right) indicates alarms from transceiver 1 and LEDBAR 2 (at the left) from transceiver 2. Especially, the alarms LOF, OOF and LOCA are asserted when the device is reset and stay asserted if the fiber is not connected at all. LED 10 of LEDBAR 1 is ON when analog 5V is powered and LED 10 of LEDBAR 2 is ON when digital 5V is powered. Make sure both these LEDs are ON when the board is operating. The corresponding LEDs for all alarms and VCC testing are shown in table 4.

LED num.	LEDBAR 1:Alarms for transceiv. 1	LEDBAR 2:Alarms for transceiv. 2
1	LOF	LOSRD
2	OOF	LOP
3	LAIS	LAIS
4	LOP	LOCA
5	LOS	OOF
6	LOSRD	LOF
7	LFERF	LFERF
8	LOCA	B1ERR
9	B1ERR	LOS
10	An. 5V	Dig. 5V

Table 4: Alarm indicated by each LED

On the ledbar, LED 2 is under LED 1, LED 3 is under LED 2 and so on.

C.7 IMPORTANT NOTE

Pin 2 of U13 inverter (reset signals generator) is forced away from its pad. Make sure this pin is ALWAYS unconnected to its pad for the correct operation of the board. Do NOT attempt to solder it again on its pad (or bring it closer to its pad).

D Board pictures

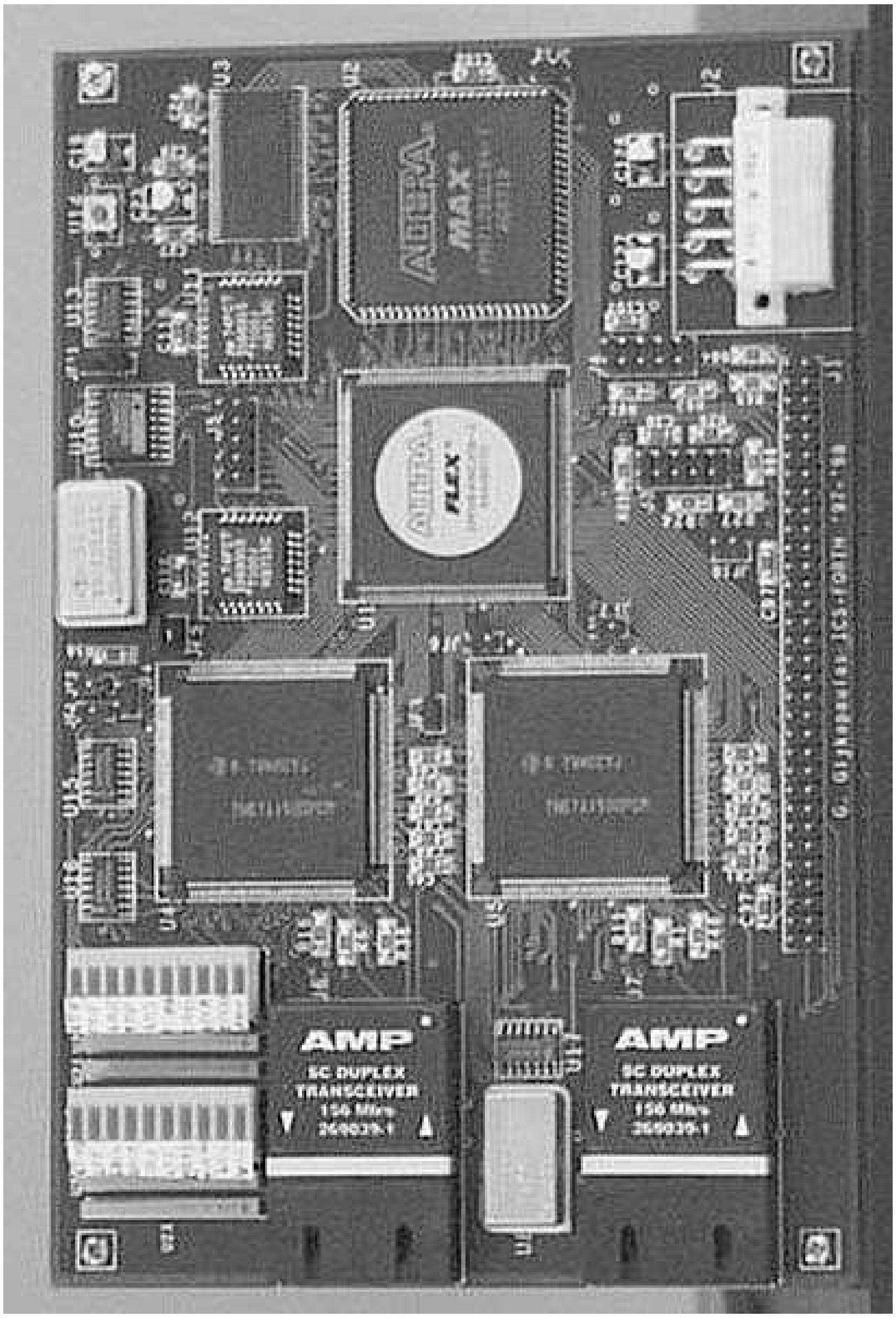

Figure 30: Picture of the final board (top view).

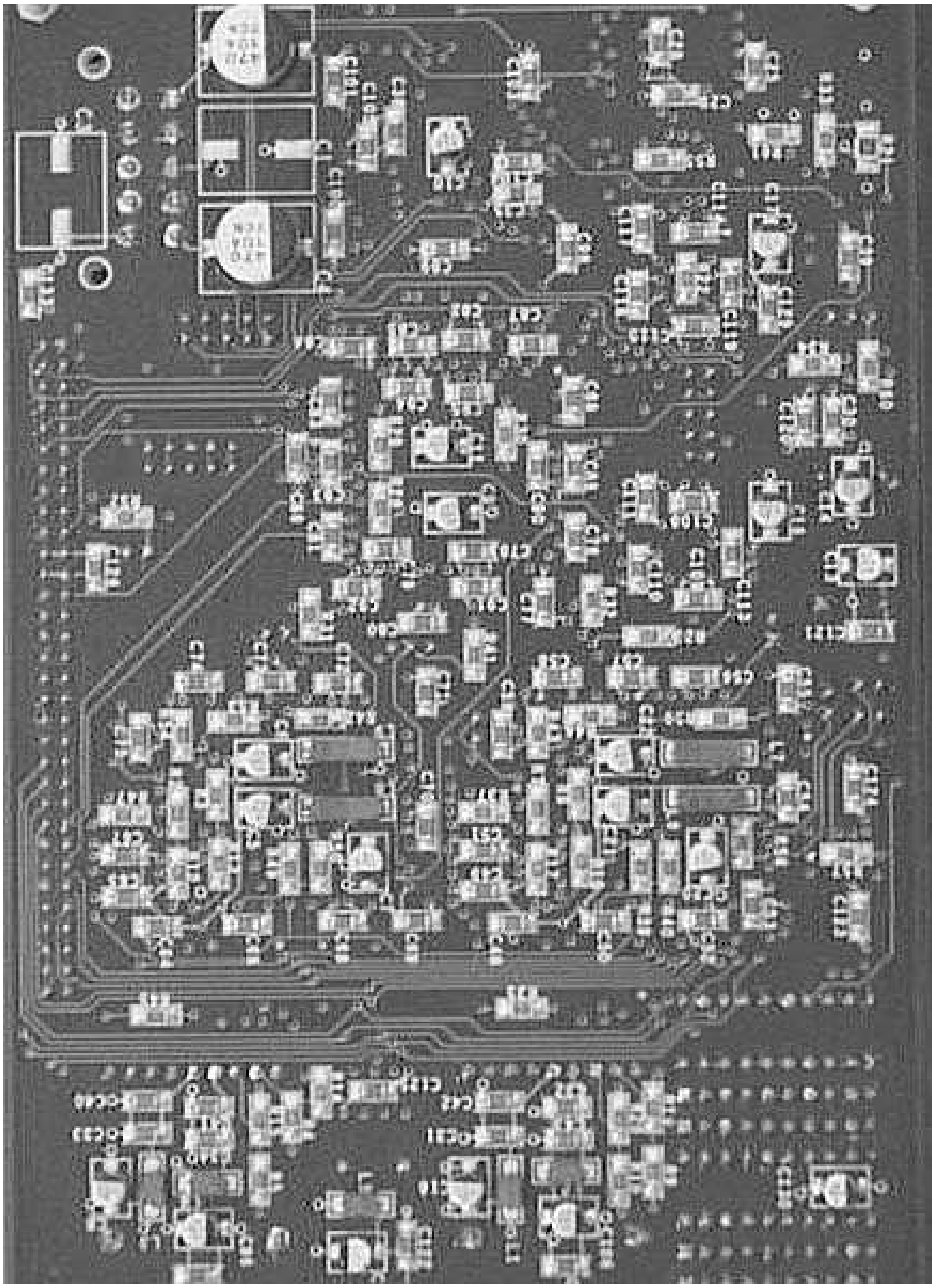

Figure 31: Picture of the final board (bottom view).

E Bill of materials

The following materials have been used:

Part/num	Manufacturer	Supplier	Description	Qty	Unit Price ($)
EPF10K40RC208-3	ALTERA	Arrow Elec.	FPGA	1	150
EPM7128SLC84-7	ALTERA	Arrow Elec.	FPGA	1	30
uPD4516161	NEC	Marshall Ind.	SDRAM 16Mbit	1	20
TNETA1500-PCM	Texas Instrum.	Arrow Elec.	SONET ATM transceiver	2	50
269039-1	AMP	Arrow Elec.	e/o converter	2	90
OSO-STC B015 19.44	OnSpec	OnSpec	oscillator 19.44MHz 15ppm	2	12
ICD2051-2	Cypress	Ambar Comp.	Programmable clock gener.	1	5
74FCT388915T-100J	IDT	Insight	PLL based clock driver	2	8
74ACTQ04SC	Fairchild	Digikey	Hex inverter	4	1
7914J-1-000E	Bourns Inc.	Digikey	Reset Button	1	1
MV57164	Quality Tech. Optoelectron.	Quality Tech. Optoelectron.	LEDBAR (10 LEDS)	2	2
4306R-102-221	Bourns Inc.	Digikey	Resistor Array	4	1
ECE-V1CA471P	Panasonic	Digikey	470uF capacitor	4	0.8
EEV-HB1C100R	Panasonic	Digikey	10uF capacitor	17	0.3
ECE-V1EA4R7R	Panasonic	Digikey	4.7uF capacitor	2	0.3
ECE-V1HA010R	Panasonic	Digikey	1uF capacitor	4	0.3
ECU-V1H104KBW	Panasonic	Digikey	0.1uF capacitor	88	0.3
ECU-V1H103KBM	Panasonic	Digikey	0.01uF capacitor	6	0.1
ECU-V1H102KBM	Panasonic	Digikey	0.001uF capacitor	11	0.1
ERJ-8ENF82R5	Panasonic	Digikey	82.5 Ohms resistor	10	0.1
ERJ-8ENF1000	Panasonic	Digikey	100 Ohms resistor	6	0.1
ERJ-8ENF1300	Panasonic	Digikey	130 Ohms resistor 1/8 Watt	10	0.1
ERJ-14YJ131	Panasonic	Digikey	130 Ohms resistor 1/4 Watt	1	0.3
ERJ-8ENF1001	Panasonic	Digikey	1.0 KOhms resistor	19	0.1
ERJ-8ENF1301	Panasonic	Digikey	1.3 KOhms resistor	2	0.1
ERJ-8ENF3741	Panasonic	Digikey	3.74 KOhms resistor	2	0.1
ERJ-8ENF4751	Panasonic	Digikey	4.75 KOhms resistor	14	0.1
2743019447	Fair-Rite	Fair-Rite	Ferrite Bead (short lead)	5	2
2743021447	Fair-Rite	Fair-Rite	Ferrite Bead (long lead)	4	2
39-29-1108	Molex	Digikey	Power Conn. (male header)	1	2.5
39-01-2100	Molex	Digikey	Power Conn. (conn. housing)	1	1

F Manufacturers

ALTERA (USA):
Phone: (408) 544-7104 (Product availability, pricing)
WEB: http://www.altera.com/
E-mail: sos@altera.com (technical support)

AMP (USA):
Phone: 717-564-0100
Fax: 717-986-7575
WEB: http://www.amp.com/
E-mail: http://www.amp.com/support/feedback.html (e-mail form)

Bourns (USA):
WEB: e-mail: http://www.bourns.com/
E-mail: http://www.bourns.com/asp/ModelSearch/HumanHelp.asp (e-mail addresses)

Cypress (USA):
Phone: (408) 943-2600
Fax: (408) 943-2741
WEB: http://www.cypress.com/
E-mail: webmaster@cypress.com

Fairchild (USA):
Phone: (207) 775-8100
Fax: (207) 761-6020
WEB: http://www.fairchildsemi.com/
E-mail: http://www.fairchildsemi.com/cgi-bin/tsg.cgi/ (tech support)

Fair-Rite (USA):
Phone: (914) 895-2055
Fax: (914) 895-2629
WEB: http://www.fair-rite.com/
E-mail: ferrites@fair-rite.com

IDT (USA):
Phone: 800-345-7015
Fax: 408-492-8674
WEB: http://www.idt.com/, http://www.idt.com/tools/support.html (tech support)
E-mail: ussales@idt.com (sales)

Molex (USA):
Phone: 630-969-4550

Fax: 630-968-8356
WEB: http://www.molex.com/
e-mail: amerinfo@molex.com (for USA), euroinfo@molex.com (for Europe)

NEC (USA):
WEB: http://www.nec.com/
E-mail: tech-support@nectech.com

OnSpec (UK):
Phone: 01203 642024
Fax: 01203 642073
WEB: http://www.onspec.co.uk/
E-mail: sales@onspec.co.uk

Panasonic (USA):
WEB: http://www.panasonic.com/pic/index-semico.html

Quality Technologies Optoelectronics (USA):
Phone: (408) 720-1440
Fax: (408) 720-0848
WEB: http://www.qtopto.com/
E-mail: http://www.qtopto.com/inforeq.htm (e-mail form)

Texas-Instruments (USA):
Phone: (972) 644-5580
Fax: (972) 480-7800
WEB: http://www.ti.com/
E-mail: http://www.ti.com/sc/docs/feedbk1.htm (e-mail form)

G Distributors

Marshall Industries (USA):
Phone: 619-627-4140
WEB: http://www.marshall.com/

Arrow Electronics (USA):
WEB: http://www.arrowsemi.com/
E-mail: mgoldberg@arrow.com (sales person)

Ambar Components Ltd. (UK):
Phone: +44 1296 397 396
Fax: +44 1296 397 439
WEB: http://www.ambar.memec.com/

Insight (USA):
Fax: 602-902-1180 WEB: http://www.insight.com/
E-mail: info@insight.com

Digikey (USA):
Phone: 218-681-6674
Fax: 218-681-3380
WEB: http://www.digikey.com/
E-mail: http://www.digikey.com/MKT/Contact.html (e-mail forms)

Other distributors:
All American (USA): 408-441-1300
Bell Industries (USA): 619-576-3290
Bell Microproducts (USA): 408-451-9400
Merit Electronics (USA): 408-434-0800
Pioneer (USA): 408-954-9100
Sterling Electronics (USA): 408-435-0835
Western Microtechnology (USA): 619-453-8430