An ATM Interface with facilities for Traffic Generation and Monitoring.

Technical Report

Gerald Tripp

The Computing Laboratory
The University of Kent at Canterbury

1. Introduction

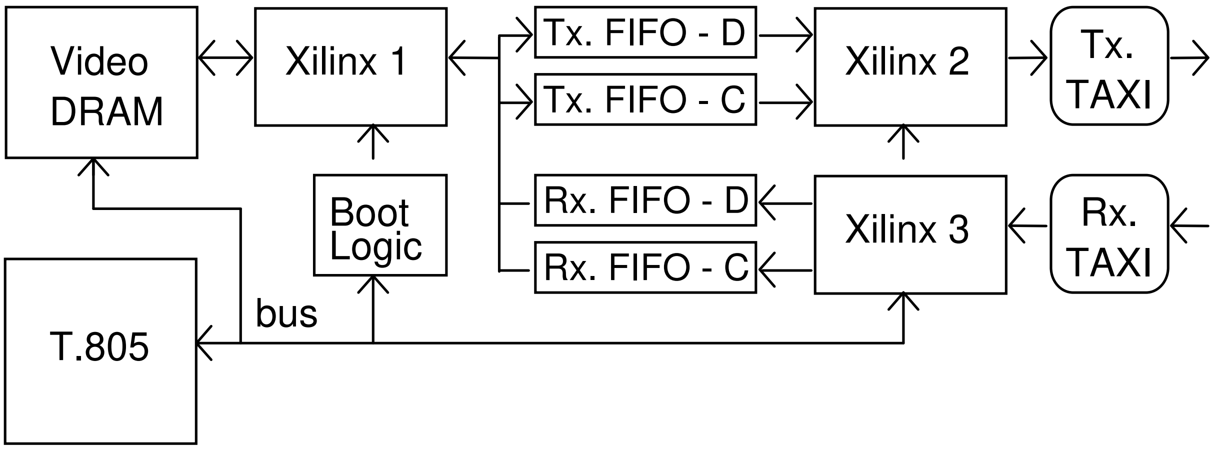
This paper describes an ATM interface with additional facilities to also allow it to act as both a traffic monitor and traffic source. High resolution timing is provided to enable all received cells to be time stamped on arrival and the information passed back to the user. For transmission, a cell has an associated target transmission time and the transmit hardware will block the head of the transmit queue until the target transmit time is reached.

This interface is intended to serve two purposes: as a conventional interface for ATM based projects and also as a piece of test equipment to act as a traffic source or monitor (or both) for use in experiments on ATM network performance. As a conventional interface, the control over transmit time for individual cells gives facilities for implementation of traffic shaping schemes. As a piece of test equipment, the control over precise transmit times and the recording of cell receive times allow accurate measurements to be made on network performance which would not be possible with a conventional interface.

2. Hardware Description

This ATM interface has been designed as a **TRA**nsputer **M**odule (or TRAM) [1]. This module can be used as part of a set of TRAMs out of which a multi-transputer system can be constructed or it can be mounted as a single module on a transputer motherboard just to act as an interface for a particular system. Data connections off board are via the four transputer links, which can each run at up to 20 Mbits/sec. Adapting this design to produce a card that is specific to a particular I/O bus would be possible without much additional work.

This interface card is constructed as a size 8 TRAM which has dimensions of 8.75" x 3.66". ATM input and output connections are provided on board as 50Ω SMB coaxial connectors. Local processing on the card is provided by a single transputer - a 20 MHz T.805 [2].



The local memory for the transputer is provided using 512 KBytes of VRAM (Video DRAM). The VRAM serial port is used for I/O, as will be described later. FIFO buffers are provided for both transmit and receive cell streams - the FIFO buffering in each direction is provided as a large data FIFO and a small control FIFO. The bulk of the logic is implemented as FPGAs (Xilinx™ [3] chips). The physical layer ATM interface is provided using the AMD TAXIchip™ set [4] which provides 4B/5B coding - the physical media is 50Ω coax.

2.1 Transputer and memory map

The transputer used in this interface is a T.805. The current version uses a 20 MHz part, although this could be upgraded later subject to the memory timing being adjusted accordingly. The transputer memory is provided as VRAM - being implemented as 4 off (128K x 8) memory chips - Toshiba TC528128BZ-80 [5]. With the current transputer part, the memory is accessed by selecting a 4 clock period memory cycle (i.e. 200ns).

This configuration of memory, provides a 32 bit Random Access port to the transputer. This memory appears in two places in the transputer memory map: one is the conventional location for transputer memory and the other is a shadow copy that is used for data transfer cycles - which are described later in this paper. The serial access ports of each chip are commoned together to give a multiplexed 8 bit wide serial port to the I/O system - thus giving a byte stream. The serial access and data transfer cycles are described later.

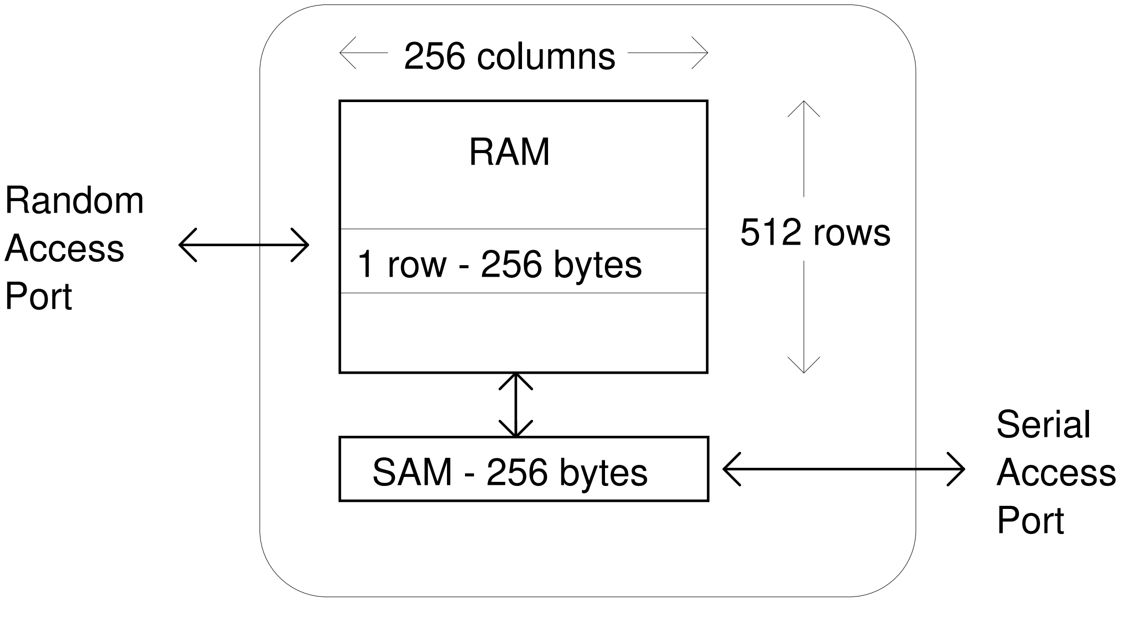
™Xilinx is a trademark of Xilinx, Inc.
™TAXIchip is a trademark of Advanced Micro Devices, Inc.

#80000000 - #8007FFFF	main memory
#C0000000 - #C007FFFF	main memory - access for data transfer cycles
#40000040	Xilinx boot: Control Register - CSR
#40000000	Xilinx boot: BOOT & Xilinx status
#40-#7C	Xilinx I/O Registers

To enable the transputer to access the Xilinx chips for simple I/O, these chips appear in the transputer memory map and have 16 word addresses shared between them. This is currently decoded as 4 word addresses each. Xilinx chips 2 & 3 have an 8 bit wide port which appears as the bottom 8 bits of the 32 bit word. Xilinx chip 1 has a 16 bit wide port - although currently only the bottom 8 bits of this is used. The Xilinx chips are un-programmed when the card is first switched on, and need to be loaded by the transputer at run time. As at this time there will be very little control logic operating on the circuit board, a small piece of logic is provided to bootstrap the Xilinx chips - this is also memory mapped and has a simple interface.

2.2 Video Memory

The main transputer memory for this interface is provided using Video DRAM (or VRAM). This type of memory is intended primarily for use in video display cards. Unlike ordinary DRAM, this memory is dual ported - it has a conventional Random Access port similar to that found on DRAM, and also has a high speed serial port.



Video RAM

The serial port is connected to a small area of Serial Access Memory (SAM) which can be read or written sequentially. Each of the memory chips used in this interface has an area of RAM which is organised as 512 rows of 256 bytes. The SAM is also 256 bytes and can be copied to or from any of the rows in the RAM.

Once set-up, the Serial Access port can be operated independently from the Random Access port. This is commonly used in video displays, where a row of pixels is loaded into the SAM and then output via the Serial Access port in real time during the line scan. This type of memory is also useful for high speed interfaces, as it gives a method of performing a type of high speed direct memory access - the serial port can operate at clock rates of up to 33 MHz.

Set-up of the serial port is done via the Random Access port. As well as the standard read/write operations, it is possible to perform **data transfer cycles** which move data between the SAM and a row of the RAM. This operation also selects whether the Serial Access port is to be left in input or output mode. The selection of a data transfer cycle is done by asserting the DT signal[3] - the various operations possible are shown below.

DT	Memory Cycle	SE-	Operation Name	Transfer	Serial Port mode
0	READ	X	Normal Read	-	-
0	WRITE	X	Normal Write	-	-
1	READ	X	Read Transfer	RAM -> SAM	Output
1	WRITE	1	Psuedo Write	-	Input
1	WRITE	0	Write Transfer	SAM -> RAM	Input

In terms of this interface, the DT signal is generated by accessing the memory in its shadow locations as shown in the memory map above. The SE- signal is generated by the FPGA - Xilinx 1. Some parts of the contents of the address and data busses are also used during some of these three special memory cycles, as shown below:

Operation Name	MD[0-7]	MA[0-7]	MA[8-16]
Read Transfer	-	Set SAM pointer	Select row
Psuedo Write	-	Set SAM pointer	-
Write Transfer	Write Mask	Set SAM pointer	Select row

The write mask selects which bits of the words in RAM to update - this is useful when the VRAM is used for graphical displays, but for this system the write-mask is set to all 1's so that all the bits in a word are updated. The SAM pointer selects the next location to read/write in the SAM. During a read or write transfer, a row in RAM is selected for transfer to/from the SAM. Access to the SAM from the serial port will start at the address specified by the SAM pointer. When this address reaches 255, it wraps around to 0. Hence the SAM can be read/written many times if required.

In this particular interface, four of these chips are used to give 32 bit wide memory. Because of this, a 32 bit wide write mask is used across the four chips, and the addresses referred to above become word addresses and are derived from the Transputer byte address shifted right by 2 bit positions.

As four of these chips are used, there are actually 1024 bytes of SAM available for transfers. The serial access ports for these four chips have the data busses commoned together, hence giving a byte wide data path - the control signals for the serial port are kept separate and need to be accessed in turn to read or write the SAM(s) in byte address order.

[3]This is multiplexed with the OE enable signal and picked up during the early part of the memory cycle.

2.2.1 Use of Video RAM serial port

The first point to note is that the serial mode needs to operate within a single 1024 byte page of memory[4] - if a transfer reaches the end of the 1024 byte page, it simply wraps around to the start. Subject to this constraint, transfers of any length within a page can take place. However, care needs to be taken when using this interface especially on input, when the current ownership of any page needs to be considered.

Output mode

The serial port is very easy to use in output (or read) mode. The first word of the buffer that you wish to output is read in its shadow location in memory. This will copy the appropriate 1024 byte page into the SAM and set the SAM pointer to the location specified in the read. Xilinx 1 can now read data from the SAM starting at the given address.

> Read Transfer from start of buffer
> <data output via serial port>

Input mode

The serial port is more difficult to use in input (or write) mode, as this has a side effect on the main memory. The serial port itself will default to output mode, so the direction of the serial port needs to be reversed before data can be read in. Another point to note, is that the write transfer will update the whole of the 1024 byte page of memory, even if only a few bytes have been read into the SAM. Because of this, the standard method of access is first to perform a read transfer cycle to pre-load the SAM with the existing RAM contents, then change the serial port to input mode, update the SAM with input and finally put the updated page in place.

> Read Transfer from start of buffer
> Pseudo Write transfer to start of buffer
> <data input via serial port>
> Write Transfer to start of buffer (value = write-mask).

Another important point to note, is that between the Read Transfer and the final Write Transfer, the page of memory effectively belongs to the I/O system. Any updates to this page made by the processor will be overwritten by the write transfer. The choice between Write Transfer and Pseudo Write Transfer is determined by the value placed on the SE- signal by Xilinx 1 - see later.

2.3 FIFO Buffers

Hardware FIFO buffers are provided in the transmit and receive cell streams. In this implementation there are two buffers in each direction: a 512 byte control FIFO and a 4096 byte data FIFO. The data FIFO is used to hold the contents of the cells, including the header and the control FIFO is used to pass any other information such as length and timing information. The ATM ends of these FIFOs are controlled by the byte clocks generated for each of transmit and receive - by having separate control and data FIFOs, the cell contents can be passed as required without any delays being caused by the

[4] The memory chip used for this interface has a number of features to allow split page transfers to take place, but access to these features is not available in this design.

passing of control information. A second advantage is that the use of two FIFOs in each direction allows for a certain amount of reordering of data: the length of a cell is only known at the end of its reception, but this can be passed via the control stream and read by Xilinx 1 ahead of the cell contents.

2.4 Line Input/Output

The physical layer is implemented using the AMD TAXI chips [4]. This is one of the many standards currently in use for ATM and was chosen because of its current use in the academic community for ATM research. The TAXI chips use the 4B/5B line encoding scheme as used by FDDI - where a 5 bit symbol is used to carry 4 bit data items or various control symbols. The channel itself runs at a raw data rate of 125 Mbit/sec - therefore giving a maximum user data rate of 100 Mbit/sec.

The TAXI chips provide the user with a parallel port, which allows the transfer of pairs of 5 bit symbols - either 8 bits of data or 1 of 16 different control symbol pairs. This allows the interconnected ATM hardware to run as a byte wide data path with a clock rate of 12.5 MHz.

The external interface provided by this card is 50Ω unbalanced, via on board SMB coaxial connectors.

3 Xilinx One

The current version of this design is X1E.

This chip provides an interface between the video memory and the FIFO buffers. Transfers take place to/from 64 byte records in video memory - the length of 64 being chosen as it divides into 1024 - giving 16 records per 1024 byte page. On transmit, these records are read from video memory and the appropriate parts loaded into the transmit FIFOs. On receive, these records are written into video memory, with the record contents being generated from data read from the Receive FIFOs - unused fields being written as 0.

Up to 16 records can be transferred in either direction in a single burst - with the transfer halting at appropriate points such as transfer complete or receive FIFO empty.

3.1 Data Format

Data will normally be transferred in 64 byte records[5]. In the tables below, the lowest address is shown on the right hand side - hence *Timer byte 0* will be at byte offset 0 and *Body 48* will be at byte offset 63 (from the start of this 64 byte block).

A standard format for the 64 byte record has been devised and is shown below; although many of the fields in this record are currently unused. The layout of the block is such that both the header and the cell body start on word boundaries - thus making software access to these fields faster. This requires the cell to be transferred as a 52 byte quantity - this is quite valid as the HEC field is created on transmission automatically and checked on reception to generate a syndrome value. The syndrome should be 0 if there are no errors in (header) transmission - some non-zero values should allow error correction to take place but this is of doubtful value after the cell has been through 4B/5B encoding.

Standard format:

Timer byte 3	Timer byte 2	Timer byte 1	Timer byte 0
Hard Status	Length	Syndrome	Soft Status
AAL5 CRC 3	AAL5 CRC 2	AAL5 CRC 1	AAL5 CRC 0
Header 4	Header 3	Header 2	Header 1
Body 4	Body 3	Body 2	Body 1
Body 8	Body 7	Body 6	Body 5
etc			
Body 48	Body 47	Body 46	Body 45

[5]Assuming a 53 byte cell.

3.1.1 Transmitter Format

The transmit logic (X2C), has the advantage of being designed after the standard block format had been defined. Hence this operates with a subset of the standard format as follows:

0	0	Timer byte 1	Timer byte 0
Status	Length	0	0
0	0	0	0
Header 4	Header 3	Header 2	Header 1
Body 4	Body 3	Body 2	Body 1
Body 8	Body 7	Body 6	Body 5
etc			
Body 48	Body 47	Body 46	Body 45

Note: the Length field does not include the HEC field which is generated by the ATM Tx logic. The length of a standard cell is therefore given as 52.

3.1.2 Receiver format

The current cell receive logic (X3C) provides blocks in an old format. This currently requires some of the fields to be swapped in software to maintain compatibility with the standard format.

0	0	Timer byte 1	Timer byte 0
Status (0)	Length	Syndrome (0)	0
Header 1	0	0	0
Syndrome	Header 4	Header 3	Header 2
Body 4	Body 3	Body 2	Body 1
Body 8	Body 7	Body 6	Body 5
etc			
Body 48	Body 47	Body 46	Body 45

The length of a cell is given as 53 as this currently includes the syndrome.

3.2 Operation

This chip acts as a half duplex, fast interface to the FIFOs. At any time that transmit or receive are ready, a transfer of up to 16 cells can be initiated in the appropriate direction. On transmit this will normally continue until completion - unless it is cancelled. On receive the transfer can terminate for a number of different reasons - with the reason for termination given in the status register. As the received cell stream is liable to corruption if the receive FIFOs overflow, the first byte in each FIFO for a cell is flagged[6] with bit 8 set to 1 and this checked on transfer - if this bit is incorrect then this indicates that the receive FIFO contents may be corrupt and the transfer will terminate.

One of the problems that slows the receive transfer is the possibility of hitting Receive FIFO empty. It is always possible to read the receive data FIFO without checking as the transfer of the cell contents will only begin after the cell length field is received via the control FIFO. For the control FIFO however, it is possible that the cell may still be arriving as the transfer of that cell is in progress - because of this the status of the control FIFO will normally require checking before each read. This can cause delay as due to pipeline delays in and out of the chip, it is necessary to wait for a number of cycles after a FIFO read in case the status changes. Because of this a option called **X1FAST** has been implemented, where if during a transfer the receive data FIFO is detected as being over half full, then the control FIFO will be read without checking status for the rest of the current transfer. This option should only be used if the size of the receive data FIFO is at least 2048 bytes (it is currently 4096 bytes) to ensure that there are already enough cells in the receive FIFOs to allow the transfer to complete.

Interrupts can be generated on completion of transfers and also to indicate that transfers can take place. Receive is deemed to be ready if there is data in the receive FIFOs; transmit is ready if both the transmit FIFOs are less than half full - hence enabling a full length burst to take place without the FIFOs becoming full. Once an interrupt has been generated, then all interrupts will be disabled.

3.2.1 Specification of Internal Interface to FIFOs

The data written into the transmit FIFOs for each cell transferred is as follows:

Item	Size (bytes)	FIFO	Bit 8 set?
Timer byte 0	1	Control	No
Timer byte 1	1	Control	No
Length	1	Control	No
Cell	<Length>	Data	No
Status	1	Control	No

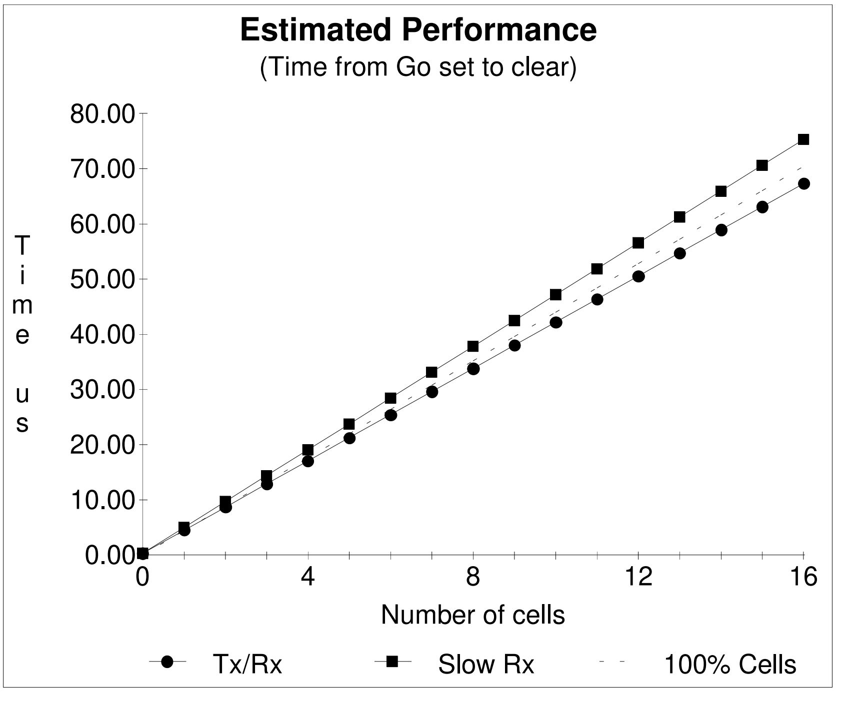
The data read from the receive FIFOs for each cell transferred is as follows:

Item	Size (bytes)	FIFO	Bit 8 expected to be set?
Timer byte 0	1	Control	Yes
Timer byte 1	1	Control	No
Length	1	Control	No
Cell	<Length>	Data	First byte only

[6]The FIFOs are actually 9 bits wide.

3.3 Performance

Below, is the estimated performance of this chip, based on the expected operation of the hardware[7]. This is quite straight forward to calculate for transmit, as it will normally run to completion without having to wait for any other part of the system. For receive, the performance is affected by the state of the receive FIFOs, as these can become empty during a transfer and hence cause the hardware to wait (or terminate the transfer). As mentioned above, there is also the problem that immediately after reading from a FIFO, the hardware is unsure of the FIFO status because of the heavy pipelining used in the hardware design.

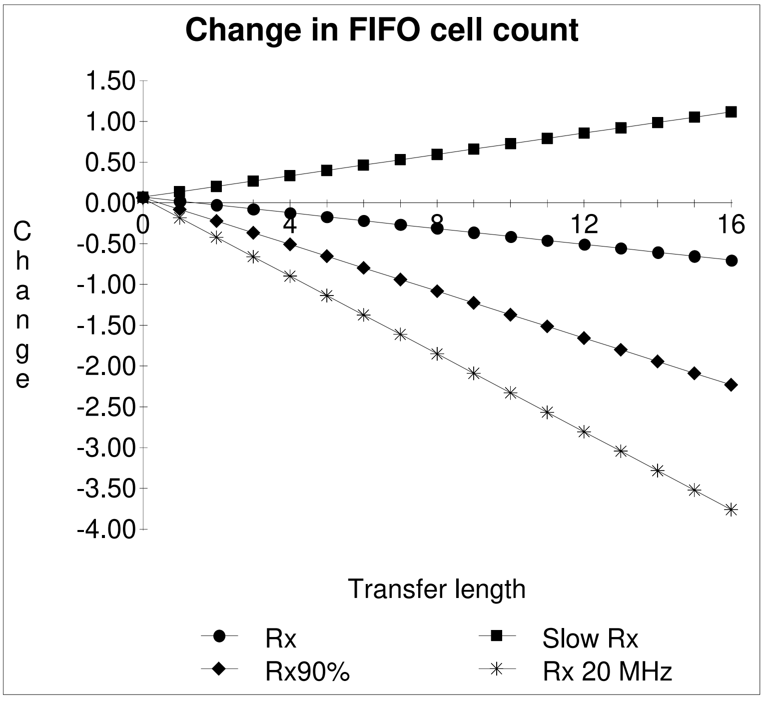


When the Receive data FIFOs are already half full, then the Rx transfer will take the same period of time as a Tx transfer of the same length. When the Receive data FIFOs is less than half full, then the performance will be slightly slower - due to the extra checking required. The figures quoted above are the times taken from first setting the GO bit on the interface to it being cleared by the chip to indicate end of transfer. Transmit and Receive done interrupts are generated during the processing of the final cell transferred and will typically be 4.0 µs earlier[8] - this is provided to give some overlap with the transputer interrupt latency. The receive FIFOs are assumed to never become empty.

It can be seen from the graph above, that the transfer rate is quite close to the maximum cell rate (100% cells) - the transfer rate being slower than cell rate for slow receive and faster for fast receive and for transmit. The transfer rate is limited mainly by the clock rate of Xilinx 1. This is currently running at 16 MHz, which is limited by the speed of asynchronous interfaces to the FIFOs and the Video Memory. A cell can arrive from the network every 4.4 µs, and Xilinx 1 has to go through a minimum of 64 clock cycles just to traverse a block in VRAM - even if the field is not used and just set to 0.

[7]These figures need to be verified by measurement.
[8]Except for the case when the transfer length is 0.

As a more illustrative example, below are the average effects on the FIFO length (as measured in cells) after performing transfers of various lengths. Both slow and fast receive are shown at 100% cell arrival rate, also shown is fast receive with 90% cell arrival rate.

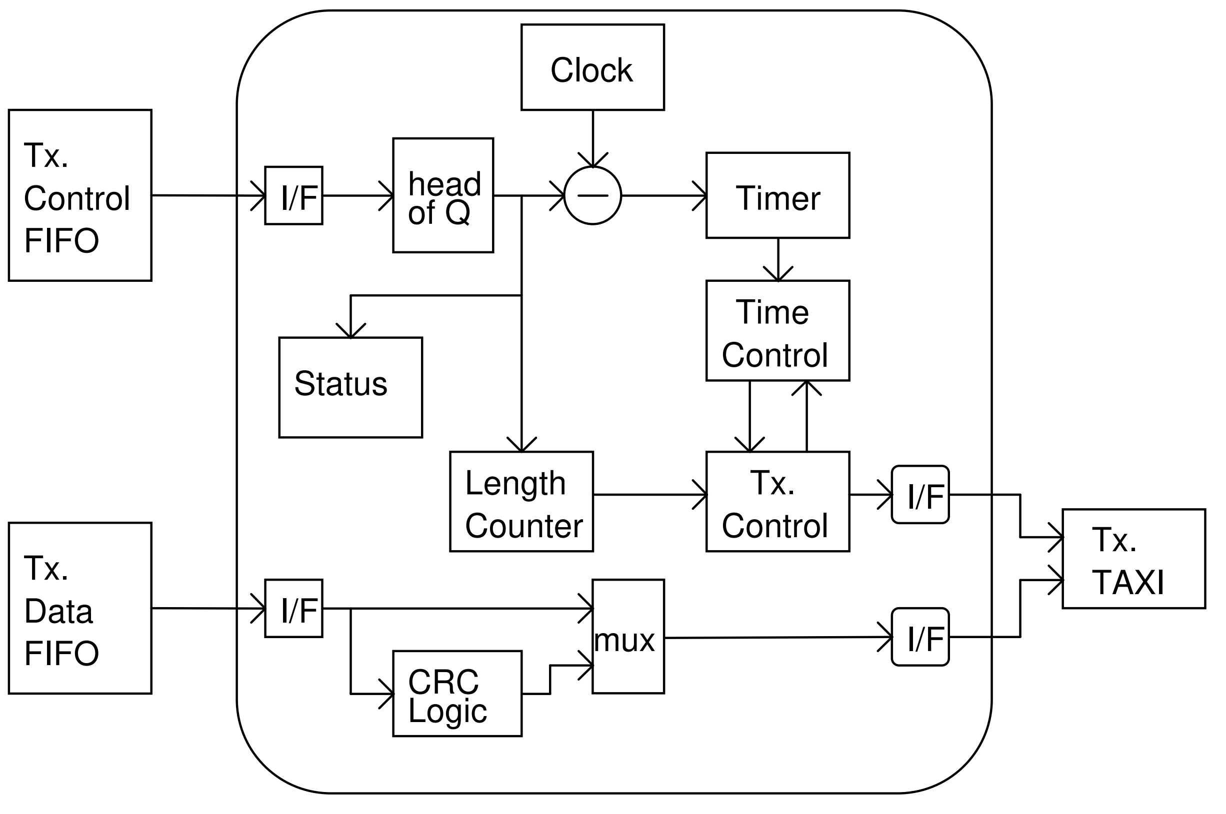


For interest only, the fourth plot shows fast receive operating at 100% arrival rate, but with the Xilinx 1 clock rate increased to 20 MHz. It is hoped that this increase in speed may be possible with a re-design of the interfaces to the VRAM and FIFOs.

The performance figures shown above, only show the performance of receive when the receive FIFOs stay non empty. If the receive logic detects that the receive control FIFO is empty at the start of a possible cell transfer, then the transfer will normally be terminated with a status report of empty FIFO (X1EMPTY). It is possible however for the transfer of a cell to start at a point at which the first (and only) cell in the FIFO has not yet been fully received - in this case the cell transfer will wait on each item from the control FIFO and hence not start the data transfer until the <length> field has been read. Whereas the wait on each byte from the control will normally be negligible - as both ends are running at similar speeds - the delay in waiting for the length byte can be quite long as this will not be placed into the control FIFO until all of the cell contents have been written into the data FIFO - in this case giving an extra delay of about 4 μs. In theory, this could happen for consecutive cells and give an unusually long transfer time. A 16 cell transfer time of the order of 135 μs is probably possible in extreme cases - although an exact cell rate would need to be sustained to keep the receiver in this state. Although this doesn't really cause any problems with receive - as the receive FIFOs have to consistently contain only part of a cell - it could tie up resources which could be require for transmit. If this proves to be a problem, then an option to terminate the current transfer after such an event could be provided - so only one cell is transferred piecemeal.

4. Xilinx Two

Xilinx two is the ATM cell transmission logic. This reads cells and their control information from the transmit FIFOs and formats these for transmission on the ATM output. A rough schematic of this is shown below.



An unusual feature about this ATM cell transmitter, is that it allows the transmission time of cells to be controlled. The required transmit time for a cell is placed in the 64 byte record for that cell and transmission will be delayed by the transmitter until that time is reached.

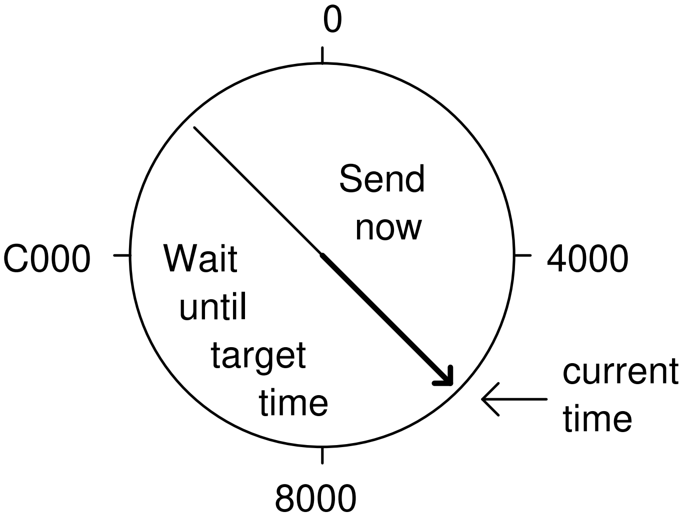
4.1 Timed Transmission scheme.

The transmit logic contains a free running clock. This is a 16 bit counter which is clocked from the transmit byte clock from the TAXI chip. This clock has a period of 80 ns, so the repeat time for the clock is about 5 ms. This clock can be read by software running on the transputer and reset to 0 if required.

The timed transmission scheme operates by subtracting the current time (from the local clock) from the target transmit time (for the cell). The result of this subtraction is loaded into a second counter which is also clocked from the transmit byte clock. The value in the counter is decremented on each clock tick, and when the value becomes negative, this is used as an signal to transmit the cell. In some cases, this will cause the cell to be transmitted immediately as the target transmit time for the cell has already passed; if this is not the case then the cell will be delayed until the target time is reached[9].

[9]To be exact, transmit time is actually target time + k. The value of the constant k is small but has not yet been

It is easy to visualise this by thinking of the current time as the hour hand on a clock - target times of up to 6 `hours' ahead will be waited for. Times greater than 6 hours ahead look as if they are in the period of up to 6 hours ago and are already late - so send immediately.

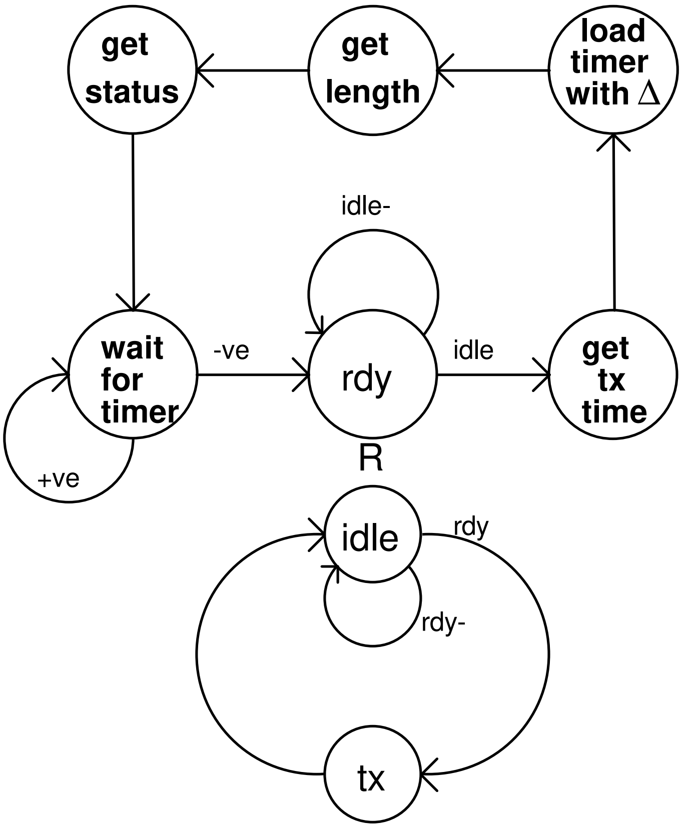


Using this timed transmission scheme, it is possible to have a transmit queue of cells, sorted in transmit order and with each cell flagged with its target transmit time.

4.2 Implementation.

The control function that this chip needs to perform is quite complex, as it has to deal with both cell transmission and timing. To simplify the implementation, the control logic is constructed as two separate state machines: the cell transmitter and the cell timer. These operate independently with the cell transmitter reading cells from the data FIFO and formatting these for transmission and the cell timer reading the control information for a cell and waiting for the correct time to transmit. These two state machines will rendezvous at a point where the cell timer decides it is time for the next cell to be transmitted and the cell transmitter is idle. This means that whilst the current cell is being transmitted, the cell timer can be pre-fetching the control information for the next cell and start the process of running the timer for deciding the time of its transmission.

calculated.



Simplified state transition diagram

The transmitter can achieve 100% of the channel, with a 53 byte cell[10] so long as there are always enough cells in the transmit FIFO to allow for the 1 cell look ahead. With the transmission of back to back cells, there will be 1 cell transmitted every 55 clock periods - this is made up of the cell start flag (TT), 53 data tokens for the cell itself and the minimum of 1 sync flag (JK) before the next cell

4.3 Empty transmit FIFOs.

As with Xilinx 1, this chip also has to cope with the problem of possibly empty FIFOs. To avoid problems with the data FIFO, Xilinx 1 will not write the last byte (Status) into the control FIFO until the whole cell has been written into the data FIFO. This means that when all four bytes of control information have been read from the control FIFO, then the specified number of bytes for that cell must already be in the data FIFO. Using this `bracketing' system means that the only FIFO that needs to have its status checked is the control FIFO - this is beneficial as only four bytes need to be read from this FIFO during one cell time and hence delays in checking status don't delay transmission.

As mentioned in the section on Xilinx 1, checking the status of a FIFO (for empty/not empty etc.) can be difficult as the pipelined interface between the Xilinx chips and the FIFOs cause the FIFO status to be delayed by several cycles and hence not immediately available after performing a FIFO read. To avoid the cell timing logic from having to cope with these FIFO status problems, the control FIFO read operation is split off into a third state machine that is responsible only for holding the first item

[10]Xilinx 2 can create cells of any length specified in the 8 bit length field. However, Xilinx 1 only supports the lower 6 bits of the length field. Any cell of length 4 or greater will have a HEC field inserted and hence gain 1 in length.

from the control FIFO. This state machine generates a ready signal to the cell timing logic to indicate that it currently holds the head item from the control FIFO: the cell timing logic can produce an acknowledgement which will dismiss this item and cause the head of queue logic to fetch the next item from the control FIFO.

4.4 Ideas for Future Work

One of the values which is not currently used is the status byte, which Xilinx 1 generates from the hard status field in the 64 byte record. At present this is only used as an indicator from Xilinx 1 that all the data for a cell have been written into the transmit data FIFO. Xilinx 2 reads this byte only as a flag to tell it that it can proceed with transmission of the cell.

The real purpose of this field for is to allow per cell control information to be passed to the transmitter. No control bits have been implemented yet, and the transmitter currently just ignores its contents. One suggestion for future use of this field is to have a transmit enable/disable bit. This would be used to indicate whether any 64 byte record actually required a cell to be transmitted or whether it was currently unused. If the transmitter received a disabled (or dummy) cell, then timing and control would proceed as usual, but nothing would be output on the ATM channel.

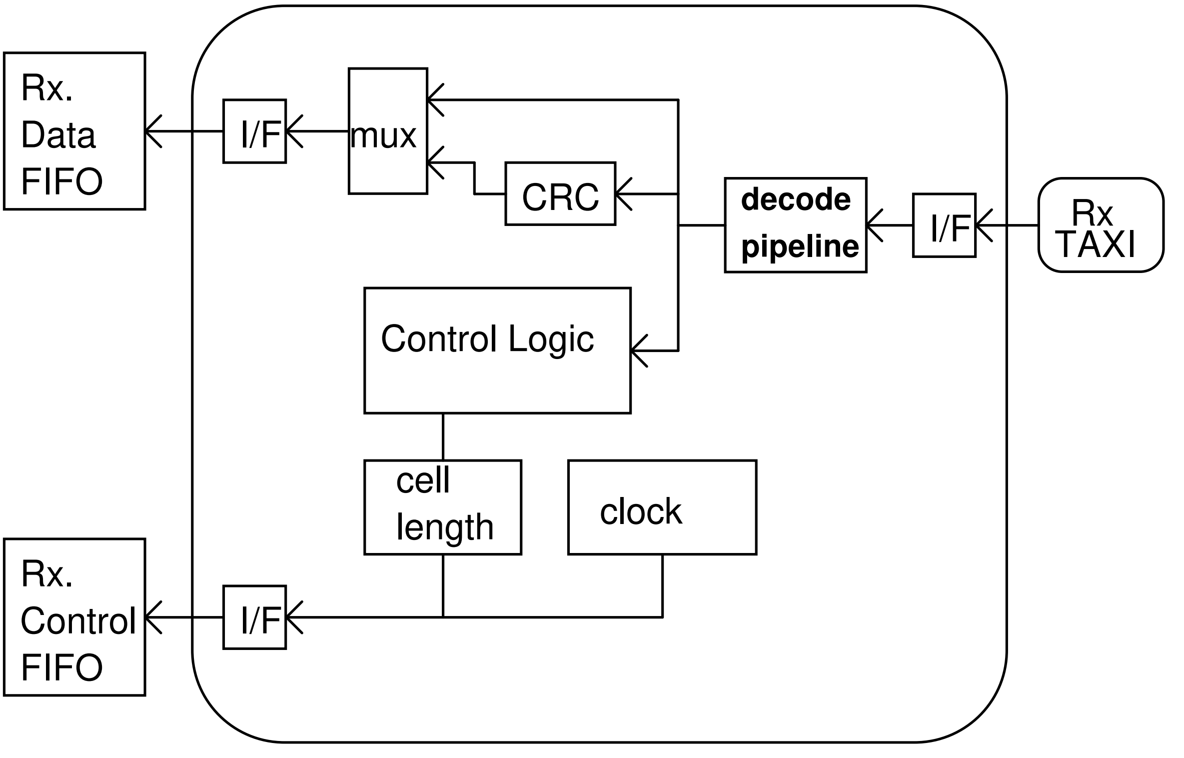
4.4.1 Use of dummy cells

One use of such a facility would be so that a template of cells with a fixed overall cell rate could be created which could be used if required by any of a number of streams. A second use is associated with the repeat period of the transmitter clock. If the gap between consecutive cells is greater than half the repeat period of the transmit clock then extra work would be required in software to delay the transfer of the cells into the transmit FIFOs, as the cells would otherwise be transmitted before the required time. To avoid this, dummy cells could be inserted into the transmit stream that activated the cell timing logic but caused nothing to be transmitted. Using this scheme, about 40 dummy cells per second would be required for a completely idle stream.

5. Xilinx Three

The current version of this chip is X3C.

The function of this chip is to receive cells from the ATM input and save them in the receive FIFOs. A rough schematic of this is shown below:



Rough schematic of Xilinx Three

A significant point about this implementation of an ATM cell receiver, is that it keeps a record of the arrival time of each cell, which is saved along with the cell contents.

5.1 Receive Process

The receive process consists of waiting for a start of cell delimiter (TT) to arrive from the line and then to take up to the next 53 data items as cell contents. The sync symbol (JK) may be present within the cell if required[11] and will be ignored. Cells are normally expected to be 53 bytes long and will be truncated if of greater length. Although cells should be transmitted with at least one sync symbol between adjacent cells, this constraint is not required by the receiver which can accept cells back to back. On reception, the cell contents will be saved in the receive data FIFO. The HEC CRC is calculated on the cell header and the syndrome generated is saved in place of the HEC field. The syndrome should be 0 for a correctly received header.

[11] These may be generated by slow cell transmitters.

5.2 Time Stamps and Control Information

As well as copying the cell contents into the receive data FIFO, some control information is also saved in the receive control FIFO. To enable the reception of short or `runt' cells, each cell has an associated length field - this should normally be 53. As well as this, the receiver has a local 16 bit clock running (like the transmitter) which is clocked from the TAXI receive byte clock. This clock has an 80 ns period and hence gives a clock repeat period of about 5ms. At the start of the reception of each cell, a copy is made of the clock and this is written into the receive control FIFO. At the end of the cell reception, the cell length is written into the control FIFO. A second benefit of transferring the cell length is that this acts as a marker to indicate that the whole of the cell has been written into the data FIFO and thus enables Xilinx 1 to read the cell in a single burst without any waiting.

5.3 Implementation

The receiver is implemented as a simple state machine which is event driven from the ATM input. The control logic is kept simple by pre-processing the ATM input with a decode pipeline. This `input pipeline' performs some simple pre-processing and presents the control logic with a token to indicate what has arrived: DATA, TT, NOP or ERROR - the DATA token being accompanied by 1 byte of data.

5.4 Performance

The ATM cell receiver should run at 100% cell rate, with 53 byte cells. Any problems will be caused by the FIFOs overflowing due to the cells not being transferred fast enough by Xilinx 1. At present the interface is rather simple in its operation - the FIFOs are always assumed to be ready (i.e. not full) and data is written into these receive FIFOs without checking the status. This will of course cause problems when the FIFOs are full, as some or all parts of a cell will be lost. As the FIFOs are actually 9 bits wide, a simple confidence check on the data integrity is performed by using the top bit to indicate the first byte of transfer. This is used on both receive FIFOs, so the control FIFO will have the top bit set for the first timing byte for a cell and the data FIFO will have the top bit set for the first byte of cell contents. This top bit is checked by Xilinx 1, which will abort the transfer if the bit appears to have the wrong value.

Ideas for solving problems with FIFO overrun are covered in section 5.5.4

5.5 Ideas for Future Work

The receiver was the first of these chips to be designed, and early experience of using this chip have enabled improvements to be introduced to the later chips before the designs were complete. As the first chip designed, the receiver has probably got the largest `wish list' for improvements!

5.5.1 Dummy cells

As with the transmitter, the 5ms clock repeat period causes problems for the software. To ensure that it is able to uniquely identify the arrival time for a cell, the software device driver would need to run its own clock. This would be needed to determine which 5ms period the cell arrived in. With the transmitter, it is proposed that this problem be addressed by the queuing of dummy cells which will never be transmitted. It is possible to use this same approach with the receiver and have the receive logic generate dummy cells when the receiver is idle. These could be generated each time the clock wraps around to 0, and used by the software to increment the more significant part of a software clock.

As with the transmitter, it is only possible to pass dummy cells around if these can be distinguished from the normal cells. By following the same model as the transmitter, a status byte could be returned from the receiver through the control FIFO to indicate what is being passed. This has several uses as it enables the receiver to pass back other status information about the cell.

5.5.2 Changes to FIFO Data Format

The format of the data in the receive FIFOs doesn't fit in too well with the 64 byte record that has now been standardised for use by Xilinx 1. In particular, passing the syndrome in place of the HEC gives this value too late to fill into its required field in the 12 byte header. The solution to this is to pass the syndrome via the control FIFO rather than the data FIFO. The other new value to pass via the control FIFO is the status byte that is mentioned in the previous section. This would give the format of the data in the receive FIFOs as follows:

Control FIFO		Data FIFO
Timing byte 0		Cell Header bytes 1-4
Timing byte 1		Cell Body bytes 1-48
Syndrome		
Length		
Status		

5.5.3 Changes to Implementation

The implementation of this chip would probably follow the model of the transmitter and have separate control and data state machines which rendezvous at appropriate points.

5.5.4 Coping with FIFO overrun

Half full

Ideally, it would be best to only write a cell into the FIFOs if it was known that enough room was available - unfortunately, the only status available from the FIFOs is Empty, half Full & Full. Although full may not be set before a cell is saved in the FIFOs, there may less space than that required for a complete cell - it is likely therefore, that only part of the last cell is saved in the FIFOs. One sure way of solving this problem is to use the half full indication as a full flag, as if the FIFOs are less than half full at the start of a cell then there will easily be enough space for the rest of a cell. Unfortunately, using the half full indication means that only just over half the buffer space is available for use and also means that the fast block transfer by Xilinx 1 on FIFO half full is no longer used so regularly.

Several other schemes are possible for coping with overflowing FIFOs - two of which are detailed below.

Halt on FIFO overflow

With the proposed new format, the data FIFO will hold 78 cells and the control FIFO, information about 102 cells[12]. Because of this, when it is detected that the data FIFO has overflowed, there will still be space in the control FIFO for that cell's control information. The last piece of information to be written into the control FIFO is the status field - which could have a status bit that indicates whether all the cell was correctly written into the data FIFO. If reception was halted at this point, then the cell would be removed from the FIFO later by Xilinx 1 leaving the FIFOs empty. In this case, as there were 1 or more bytes short in the data FIFO then the final byte saved would be read 2 or more times. The software would be able to tell that the cell had possibly been corrupted by the value in the status field. Hence after all data had been removed from the FIFOs, the FIFOs would be in a clean state to accept new cells.

When the device driver detects a corrupted cell - as indicated by the status field - it would know that FIFO overrun had occurred which had caused reception to be suspended. At this point, the device driver could start reception again into the empty FIFO.

Suspend until less than half full

The problem with the last method is that it still requires processor intervention and all incoming cells are ignored for the time it takes Xilinx 1 to transfer 78 cells. A compromise solution could be for the receiver to suspend cell reception until the data FIFO is half full. This would block reception for only half the period of time and also would not require intervention by the processor[13] to start cell reception again. The only problem with this method is that it would leave a cell in the data FIFO that had some of its data missing. This could be fixed by Xilinx 1 - by checking the status flags and stopping the read from the data FIFO when the FIFO was empty or when the head of the next cell was found - as indicated by the top bit being set. The missing contents of the cell could be padded with 0 or copies of the last valid data byte. This would make more work for Xilinx 1, in that it would need to perform this reframing operation and sometimes save the head of the next cell for future use.

This method would be a good compromise solution, although it would tend to lose bursts of cells under heavy load. Its possible implementation will depend on the availability of space in Xilinx 1.

5.5.5 Problems with `runt' cells

The reception of `runt' cells causes a few problems for the receiver. These are cells that are shorter than the standard cell length of 53 bytes. The receiver can normally handle short cells with no lasting problems - as the cell will just be associated with a non standard length field. One problem however is caused by the ability to have sync symbols within a cell, as when a short cell is received, the receiver will continue to wait for the remainder of the cell - which could legally arrive some time later. The end of the cell is only known for certain when the start delimiter for the next cell arrives, which means that a short cell will be left incomplete until this point. A time-out on cell arrival could be implemented, but this would probably be in breach of the transmission standard.

[12]The FIFOs are 9 bits wide, with the data FIFO holding 4096 words and the control FIFO 512 words.
[13]The cell receive chip doesn't have access to the rx FIFO empty flags.

A second problem that exists with the current release is that receiving a very short cell (0-2 bytes long) causes only part of the control information for that cell to be saved and hence corrupts the format of the contents of the control FIFO. This would cause Xilinx 1 to abort on a format error. This problem should be fixed with the next release although this will cause loss of a complete cell if more control information needs to be handled than there is time for (cell lengths 0-4 bytes).

6 Conclusions

The initial version of this ATM interface is now complete and (as of January 1995) two of these cards are currently in operation. Current software for traffic generation and traffic monitoring can sustain cell transfer between two cards (without an intermediate switch) at rates of up to about 140,000 cells/second with 0 cell loss and up to about 180,000 cells/second before cell loss becomes significant.

6.1 Performance

As a standard ATM interface, the current performance figure should give no problems. As a piece of test equipment, it would be useful if it could handle sustained bursts of 100% channel utilisation for both traffic generation and monitoring. At present, it will be able to handle 100% channel utilisation but only for short bursts of cells. Determining the maximum length of cell burst possible would actually make both an interesting calculation and experiment - this is likely to be quite a lot larger than the number of cells that can be stored in the FIFO.

6.1.1 Performance Improvements

Limits in performance are caused by the combined effect of the software and Xilinx 1. Although Xilinx 1 is capable of operating faster than 100% cell rate[14], it is only slightly faster than this and would present the software with a crisis time of about 3 µs between 16 cell transfers. As the speed of Xilinx 1 is very close to the cell rate, small changes in the time taken by Xilinx 1 make a significant difference to the overall performance. Part of this problem is caused by the use of 64 byte records for specifying cells in transputer memory. As the video memory is read via a serial access port, unused locations in the record need to be accessed even if there is no need to read or write that location

The time taken to read the unused fields in the record adds 8µs to the transfer time - it is difficult to know how to avoid this, without changing to a different arrangement of cell records in memory - which might create significant extra software overheads in address calculation. The 64 byte record also allows for future improvements to be made to the hardware without radical software changes.

There is a small amount of per cell overhead in Xilinx 1 which might be reduced, but a more significant improvement would be to increase the clock rate of Xilinx 1. The clock rate is currently 16 MHz and it may be possible to increase this to 20 MHz by modifying Xilinx 1 to improve the timing on the interfaces to the FIFOs and the VRAM. This would reduce the 16 cell Tx transfer to 53.85µs, which would increase the software crisis time (to keep up with 100% cell rate) from 3.09 µs to 16.55 µs. Experiments with increasing the Xilinx 1 clock rate to 20 MHz (and thus currently running the hardware slightly out of spec.) show that it is possible to run the traffic generation and monitoring systems described above at 100% cell rate with no cell loss.

[14]With receive, only if the receive FIFOs are at least half full.

6.2 Upgrades to Xilinx designs

A number of changes to the Xilinx designs have been proposed in sections 3-5. The most important of these is the rework of Xilinx 3 to provide a status field for each cell and to pass the syndrome via the control FIFO. This in turn will allow the receive record format to be updated to match that of the transmitter and hence provide word alignment for the cell header. This will require a rework of Xilinx three and minor updates to Xilinx 1 to utilise the new FIFO and record formats.

References

[1] The Transputer Applications Notebook. INMOS 1989.

[2] IMS T805 transputer. Inmos, May 1989.

[3] The Programmable Gate Array Book. XILINX 1992.

[4] Preliminary data sheet Am7968/Am7969 TAXIchip™ Integrated circuits. Advanced Micro devices May 1989.

[5] Preliminary Technical Data. 131,072 Words x 8 bits Multiport DRAM - TC528128BZ-80, Toshiba 1990.

™TAXIchip is a trademark of Advanced Micro Devices, Inc.

Appendix A
Standard Bootstrap interface

This interface is provided directly by a logic array on the TRAM card itself and is therefore available before the rest of the Xilinx chips have been programmed. This interface allows the three Xilinx chips to be booted and also allows all I/O circuitry on the TRAM card to be reset - including the Xilinx chips. In the latter case, this reset can be used to perform a soft reset of the Xilinx chips without causing a reboot.

A.1 Control register

Name: csr
Address: #40000040

The control register is a read/write register that sets the output of two on board control lines.

R/W	-	-	-	-	-	1	XIRESET	XIPROGRAM

The **XIPROGRAM** bit is active high. If this is set to 1 it will force the **done** signal on the card to a 0 state. This signal is a wired-and signal that connects to all three Xilinx chips and is locally pulled up by a resistor.

The **XIRESET** bit is also active high. This will activate the **reset** signal on the card, which will reset the various I/O chips and FIFOs and will also give a reset signal to the Xilinx chips. The effect on the Xilinx chips will depend on their current state and will be explained later.

Bit 2 of the control register will always read back as 1. The top 5 bits of the register are undefined.

A.2 Status Register

Name: xistatus
Address: #40000000

The status register is read only and will give the current status of three status bits from the Xilinx chips. There are no side effects of reading from the status register.

R	-	-	-	-	-	XIREADY	XIINIT	XIDONE

The **XIREADY** bit gives the value of the **rdy** signal which indicates that the first Xilinx chip is ready to accept a byte of data on the Xilinx boot port.

The **XIINIT** bit gives the value of the **init** signal which shows that one or more of the Xilinx chips is currently going through an initialisation phase and no boot information should be sent yet. The initialisation phase takes between 11 and 33ms. The main part of the initialisation phase is only entered after a power on reset to allow the power supply time to stabilise.

The **XIDONE** bit gives the value of the **done** signal which is high when the all the Xilinx chips have been successfully loaded.

A.3 The Boot port

Name: xiboot
Address: #40000000

This port is used to write configuration data into the Xilinx chips. This port is only ready to take a new byte of data when the **XIRDY** bit of the status register is set to 1.

A.4 Xilinx boot sequence.

On power on, the Xilinx chips will first go through a 11 to 33 ms initialisation phase, followed by a 195 to 580 us memory clear phase. After this, the Xilinx chips will first wait for the reset signal to go inactive and then the **init** bit in the status register will go to 0 to indicate that the Xilinx chips are ready for loading.

The Xilinx chips are now in the **configuration mode** and are ready to be loaded with the Xilinx configuration information. Asserting **reset** at this point will take the Xilinx chips back to the memory clear phase.

Whilst in the **configuration mode**, the Xilinx chips will accept data on the **boot** port. After receiving each byte of data, the **rdy** bit of the status register will go to 0 for an undefined period, whilst the byte is being loaded. During configuration mode, the **done** bit of the status register is 0.

After all the data has been loaded into a Xilinx chip, it will go into an **operational** state and stop pulling the **done** signal to 0. After all three chips have been loaded, the **done** signal will be pulled to a 1 by an on board pull-up resister.

When the Xilinx chips are in an **operational** state, they can be reset, by taking the **reset** signal to 1 for a short while (>6μs). This will just reset the user logic and not cause a reprogram cycle. While in the operational state, the chips can be reprogrammed by pulling **done** to 0 by setting the **prog** bit in the control register. The documentation is ambiguous at this point. The 1->0 transition on **done** is indicated as initiating a reprogram sequence - whereas elsewhere it states that a **reset** and a taking **done** to 0 are required. The latter looks to be a more conservative approach as the extra reset is unlikely to cause any problems.

The recommended boot sequence at any time is as follows:

```
#define XIDONE      1
#define XIINIT      2
#define XIREADY     4

#define XIPROGRAM   1
#define XIRESET     2

csr = XIPROGRAM | XIRESET;
wait a minimum of 6µs
csr=0;
wait until not xistatus&XIINIT

repeat
        Note: (xistatus&XIDONE) should be false
        wait until (xistatus&XIREADY) is true;
        write byte of config. data to: xiboot
until all data sent

(xistatus&XIDONE) should now go to true - may be a slight delay.

csr = XIRESET /* reset all three Xilinx chips, FIFOs and taxi chips */
wait for 6µs
csr=0
```

Appendix B
Register Interfaces to Xilinx chips

B.1 Xilinx One - X1E (VRAM-FIFO Interface)

B.1.1 Control/Status Register

Name: x1csr
Address: #40

W	X1GO	X1WAIT	X1FAST	X1INPUT	X1IETXR	X1IETXD	X1IERXR	X1IERXD
R					X1TXR	X1TXD	X1RXR	X1RXD

X1GO - Start transfer. This bit is cleared at the end of a transfer.

X1WAIT - Don't terminate a read transfer on the Rx. control FIFO being empty.

X1FAST - For receive: if the data FIFO is half full at the start of or during the transfer then stop checking the FIFO status bits.

X1INPUT - Set to 0 for transmit, or 1 for receive. Setting this bit to 1 will forces all SE-inputs to the video memory to 0 - as is required during Write Transfer cycles. **This bit should not be set if the video memory is currently in output mode as this will cause all four memory chips to enable data onto the serial data bus.**

X1TXR - Transmit Ready - both data and control FIFOs are less than half full

X1TXD - Transmit Done - a Transmit transfer has completed

X1RXR - Receive Ready - both data and control FIFOs are non empty

X1RXD - Receive Done - a Receive transfer has completed

X1IExxx - Interrupt enable for status bit X1<xxx>.

Notes: The interrupt enable bits are all cleared on an interrupt acknowledge from the Transputer.

The X1TXD and X1RXD bits are both generated from a single DONE bit which is used to generate X1TXD and X1RXD depending on the value of the bit `Input'. If the value of the bit `X1INPUT' is changed after a transfer, then this will change which of X1TXD or X1RXD is set.

The control bits: X1INPUT, X1WAIT and X1FAST should be set before the X1GO bit.

B.1.2 Counter Register

Name: x1cell
Address: #44

R/W	0	0	0	X1CC4	X1CC3	X1CC2	X1CC1	X1CC0

Count of the number of cells required to be transferred from FIFO to VRAM (max 16). If the transfer runs to completion, then the transfer will end with this register set to 0.

Writing to this register will also clear the Rx/Tx Status register and the DONE bits X1RXD and X1TXD.

B.1.3 Tx/Rx. Status Register

Name: x1txrxs
Address: #4C

R	X1ABORT	X1HALF	X1RES1	X1FORM	X1HALT	X1NOT53	X1EMPTY	X1COUNT

X1COUNT - Transfer ends on x1cell = 0.

X1EMPTY - Transfer ends on Control FIFO empty and X1WAIT not set.

X1NOT53 - Transfer ends on cell length not 53 bytes.

X1HALT - Transfer halted by software - by clearing X1GO bit.

X1FORM - Format error in data from Rx FIFOs: Expected D8=0, got D8=1 OR expected D8=1, got D8=0.

X1RES1 - Reserved.

X1HALF - Rx Data FIFO was half full at some point during the transfer.

X1ABORT **-** Stopped part way through transfer. Either on one of the format errors or by user abort generated by writing to this register.

B.2 Xilinx Two - X2C (cell transmit)

This Xilinx chip (X2C) handles ATM transmit. This version reads ATM cells from the Tx FIFOs and transmits these over the TAXI output at the specified time.

B.2.1 Command and Status Register

Name:	x2csr
Address:	#60

R	X2TXCEF	X2TXDEF	X2MK	X2CN	X2T1	X2WT	X2IDLE	X2GO
W	_	_	_	_	_	_	_	

This status register enables the transmitter and also gives an indication of its internal state.

X2GO - Enable Transmit

The following are provided for hardware debugging.

X2IDLE - Transmitter is idle, waiting at rendezvous for timer logic

X2WT - Timer logic waiting for first time byte

X2T1 - Timer logic waiting for second time byte

X2CN - Timer logic waiting for cell length

X2WT - Timer logic waiting for status byte

X2TXCEF - Tx control FIFO empty flag (active low)

X2TXDEF - Tx data FIFO empty flag (active low)

X2C should idle with X2IDLE and X2WT

B.2.2 Counter Clear

Name:	x2txcclr
Address:	#68

Writing to this address clears the Transmit counter

B.2.3 Counter Save

Name: x2txcsave
Address: #6C

Writing to this address, saves a copy of the Transmit Counter in an I/O register

B.2.4 Counter low byte

Name: x2txclow
Address: #68

Reading from this address, gives the low byte of the transmit counter as saved above

B.2.5 Counter high byte

Name: x2txchigh
Address: #6C

Reading from this address, gives the high byte of the transmit counter as saved above

B.3 Xilinx Three - X3C (ATM cell receive)

This version decodes ATM cells from the receive TAXI interface and writes them into FIFOs. The complete cell is written into the data FIFO - with the HEC field being replaced by the syndrome. The control FIFO will contain 3 bytes:

- Time stamp (low byte)
- Time stamp (high byte)
- cell length

The cell length field is written after the complete cell has been written to the data FIFO.

To aid synchronisation, the first *byte* of the cell contents and the first *byte* of control information will have the top bit (bit 8) set to 1 - all others will have this bit set to 0.

B.3.1 Command Register

Name: x3csr
Address: #70 (Read/Write)

R/W								X3GO

GO - Bit controlling ATM Rx. 0 for idle, 1 to enable.

B.3.2 Machine State

Name: x3state
Address: #74 (Read only)

R	X3RES	X3BODY	X3H5	X3H4	X3H3	X3H2	X3H1	X3IDLE

Of bits 0-6, one bit should contain 1, the others 0. This shows the current internal state. Should read as X3IDLE unless within a cell. Provided for hardware debugging.

B.3.3 FIFO status

Name: x3fifostate
Address: #78 (Read only)

R	0	0	0	0	X3RXCFF	X3RXCHF	X3RXDFF	X3RXDHF

Values of receive FIFO status flags. Provided for hardware debugging.

X3RXCFF - Rx control FIFO full (active low)

X3RXCHF - Rx control FIFO half full (active low)

X3RXDFF - Rx data FIFO full (active low)

X3RXDHF - Rx data FIFO half full (active low)