many full cells and how many full cells with AAU bits set are in the FIFO at any given time.

On the transmission side, it is again the main EATM controller that maintains the count, feeding transmit cell events one at a time to the transmission subsystem. The transmission subsystem acknowledges an event only after it has completely transmitted the cell.

2.7 TAXI Framing

The TAXI controller is responsible for the framing of cells. In the transmit direction this is relatively straightforward. A start of cell (SOC) code must be generated, followed by four bytes of header (read from the TXFIFO) followed by the header error check (HEC) which is generated by the TAXI controller itself. This is then followed by the 48 byte payload.

In the receive direction the TAXI controller must cope with possible framing errors. The receiver can be in two states, hunting or cell processing. In the hunting state the receiver places no information in the RXFIFO. The receiver stays in the hunting state until a SOC is received. It places a SOC code in the FIFO and then begins copying header bytes into the FIFO and checks the HEC. It writes a code into the FIFO (with the 9th bit set) indicating whether the CRC was correct or not. The controller also remembers if the AUU bit was set. When 48 bytes more are received the TAXI controller enters the hunting state.

During the time the receiver is cell processing, it may of course find an other SOC code. If it does this, it simply writes a special code into the FIFO and carries on. Thus the RXFIFO always contains full cells. The receiver, by continually entering hunt mode at the end of each cell will relock onto the correct cell structure after a short cell has been received. The price paid by this simple scheme is that a short cell will also cause a full cell immediately following it to be corrupted.

References

[Hitachi 91]
HD64981F EISA Slave Interface Controller (ESIC), Hitachi, June 1991.

2.5 Transmission Errors

These are likely to be sampled periodically without using interrupts.

2.6 TAXI Subsystem Interface

The TAXI transmission subsystem consists of two AMD TAXI devices, one for transmission, one for reception, the TAXI control device implemented in a XIlinx PGA, two asynchronous FIFOs, ECL buffers, some LEDs and various descrete components.

The physical interface presented to the main part of the board consists of a subset of the signals connected to the two FIFOs and the TAXI Control PGA:

RXFIFO	rxData[0..9]
	rxFifoRdBar
	rxFifoFull
	rxFifoEmpty
TXFIFO	txData[0..9]
	txFifoWrBar
	txFifoFull
	txFifoEmpty
TAXICONTRL	rRxCell
	aRxCell
	rRxBit
	aRxBit
	rTxCell
	aTxCell

The FIFO signals are self explanatory, forming nine bit data paths in each direction. The signals to the TAXICONTRL device are in request-accept (r-a) pairs each controlling an asynchronous handshake.

An r-a pair is used in the following way. A *requestor* indicates that an event has occured by raising the r-signal. It may only do this if the a-signal is low. The *acceptor* accepts this event by raising the a-signal. Upon seeing the a-signal go high, the requestor drops the r-signal. Finally the acceptor can drop the a-signal when it sees the r-signal go low, completing the handshake.

The rRxCell-aRxCell pair is used to indicate that a complete cell has been received and placed in the receive FIFO. In this case the requestor is the transmission subsystem; the acceptor is the main ETAM controller. The rRxBit-aRxBit pair is similar but used to indicate that a complete cell with the ATM User-User bit set has been placed in the receive FIFO. (A cell with the AUU bit set will cause both events to be signalled.)

The rTxCell-aTxCell pair is used to indicate that a complete cell is in the transmit FIFO and ready for transmission; here the requestor is the main EATM controller and the acceptor is the transmission subsystem.

The events generated by the transmission subsystem, the RxCell and RxBit events, are counted by the main ETAM controller. Thus the EATM controller has a count of how

2.2.2 Initialising the ESIC

Note that the ESIC registers are byte-wide! Copy stuff in here from ESIC doc. Note option of memory mapping EATM Interface registers.

2.3 Receive Operation Example

The device driver enables interrupts for RXBIT and RXFHF. It maintains a list of partially assembled AAL5 units and their CRC-32 residues. (It expects to process an AAL5 unit without interleaving but is capable of interleaving.)

On a receive interrupt it reads the header and determines the state of reassembly on that VCI and on the VCI just processed in order to set up the CRC-32 register:

```
IF (old_vci = new_vci) AND (old_not_finished) THEN
    <hit the RXCRCCONT>
ELSE IF (old_not_finished) THEN
         <save old CRC>
     END;
     IF (new_not_finished) THEN
         <restore old CRC>
     ELSE
         <init CRC to init val>  (* also handles old = new but finished)
     END;
END;
```

If the AUU bit was not set in the header, the driver then reads the data (twelve 32 bit transfers) into the appropriate place, otherwise it reads all words and checks that the CRC-32 register is zero.

If a header error check was found, the buffer just written to now has 48 bytes of junk in it and appropriate action should be taken. (This might be no action.) If the AUU bit was set, the CRC should be checked to see if is correct.

The driver should now look to see if there are any more cells and read them. If there are no more cells, the driver clears the RXBIT bit in the STATCON register and again checks for any cells, repeating as necessary.

2.4 Transmit Operation Example

When the driver has a cell(s) to send it checks to see if there is room in the transmitter fifo. If not, it enables the interrupt on TXROOM and appends the cell to any queue it has.

For each cell for which there is room, the driver first writes the header and then sets (or continues) with the TXCRC. Unlike the receive side, there is no need to explicitly issue a CRC continue. The data is then written out, either 12 writes in a row, or in the case of the last cell of an AAL5 unit, up to but not including the CRC. The TXCRC is then read back and then written as data followed by any remaining data words.

2.1.4 RXCRC Register

This readable and writeable register contains the current state of the CRC calculation. When written to after the RXHDR register is read it starts the data running through the CRC-32 logic.

2.1.5 RXCRCCONT Register

This dummy register is written to (the value is ignored) to start the CRC-32 logic. It allows the CRC residue in the CRC-32 register to be used without having been read and restored.

2.1.6 RXTHRES Register

This writeable register is used to set the receive fifo threshold (which is in cells not bytes).

2.1.7 TXHDR Register

This writeable register should be written with the ATM cell header (excluding the header error check) before the data parts are written. The header is not run through the CRC-32 hardware.

2.1.8 TXDATA Register

This writeable register should be used to write the data bytes (4 bytes at a time) of a transmitted cell. These bytes are run through the CRC-32 hardware.

2.1.9 TXCRC Register

This is the CRC-32 residue register which can be read or written. Note that to send a CRC it should be read from the TXCRC register and then written to the TXDATA register — this is done in mid cell.

2.1.10 TXTHRES Register

This writeable register is used to set the transmit fifo threshold (which is in cells not bytes).

2.2 Initialisation

2.2.1 Finding the Sucker

As for all EISA boards, the EATM Interface should be found by investigating the EISA ID locations at 0xzc80 to 0xzc83. (need to chase a manufacturers code)

- RXBIT: This read/write bit is set by the EATM Interface whenever an ATM User-User (AUU) bit is found set in a cell header and the cell has been completely received. It is intended that this be reset by the device driver. Note that a new cell (with the AUU bit set) may arrive inbetween the time the device driver reads all cells and the time it clears this bit. A device driver should therefore check the RXCELL bit after it clears the RXBIT bit.
- RXFTHR: This read only bit is set when the number of cells in the receive FIFO exceeds the threshold.
- TXFTHR: This read only bit is set when the number of cells in the transmit FIFO is below the threshold.
- ERROR1: This read/write bit is set by the EATM Interface whenever a transmission error is detected. It is cleared by the device driver.
- ERROR2: This read/write bit is set by the EATM Interface whenever ? transmission errors are detected since it was last cleared. It is cleared by the device driver.
- INTERR1: This read/write bit, when set enables a set ERROR1 bit to cause an interrupt.
- INTERR2: This read/write bit, when set enables a set ERROR2 bit to cause an interrupt.
- INTTXFTHR: This read/write bit, when set enables a set TXFTHR bit to cause an interrupt.
- INTRXFTHR: This read/write bit, when set enables a set RXFTHR bit to cause an interrupt.
- INTRXCELL: This read/write bit, when set enables a set RXCELL bit to cause an interrupt.
- INTRXBIT: This read/write bit, when set enables a set RXBIT bit to cause an interrupt.

2.1.2 RXHDR Register

The RXHDR register is a read only register that should be read at the start of a cell. It reads a 32 word from the receive FIFO but does not run the 32 bits through the CRC-32 logic.

2.1.3 RXDATA Register

The RXDATA register is a read only register that should be read (12 times in fact) after the header has been read and the CRC-32 initialised (see below). Reading from this register reads from the receive FIFO but also runs the data through the CRC-32 logic.

Location	Name	Access	Description
0xz000	STATCON	R/W	status and control
0xz004	RXHDR	R	read cell header
0xz008	RXDATA	R	read cell data
0xz00c	RXCRC	R/W	rx crc register
0xz010	RXCRCCONT	W	rx crc continue
0xz014	RXTHRESH	W	rx fifo threshold
0xz018	TXHDR	W	write cell header
0xz01c	TXDATA	W	write cell data
0xz020	TXCRC	R/W	tx crc register
0xz024	TXTHRESH	W	tx fifo threshold

Table 1: EATM Interface IO Registers ('z' indicates slot number)

Bit	Name	Access
15	RXCELL	R
14	RXHDRERR	R
13	RXBIT	R/W
12	RXFTHR	R
11	TXFTHT	R
10	ERROR1	R
9	ERROR2	R
5	INTERR1	R/W
4	INTERR2	R/W
3	INTTXFTHR	R/W
2	INTRXFTHR	R/W
1	INTRXCELL	R/W
0	INTRXBIT	R/W

Table 2: STATCON Register Format

  - interfaces the tx and rx FIFOs to the EISA bus
  - handshakes with the transmission subsystem to keep track of cells in the FIFOs

- RXFIFO: This is intended to be a 32K by 9 bit FIFO but it pin compatible with FIFO devices of various sizes.
- TXFIFO: This is intended to be a 4K by 9 bit FIFO but it pin compatible with FIFO devices of various sizes.
- Transmission subsystem controller: This Xilinx device is based on existing Fairisle and Saphire taxi transmission controllers. It interfaces the FIFOs to the TAXI devices, performs cell framing and header CRC functions. It also handshakes with the main EATM Interface controller.
- TAXI devices: These are the two AMD TAXI devices and the associated ECL and analog gubbins.
- Bus drivers: These are four bidirectional latches which drive the EISA data lines. They are controlled by the main EATM Interface controller Xilinx device.
- FIFO Pulses: A delay line and a PAL will be used to generate the tx FIFO write and the rx FIFO read signals, allowing the main EATM Interface controller to behave in a completely synchronous manner.
- miscellaneous: A PROM to program the Xilinx devices and a PAL or PROM to supply the EISA identification information will be required. (It is possible that the Xilinx device can supply the EISA identification.)

2 Programmer's Interface

2.1 EATM Interface Register Set

The EATM Interface uses the Hitachi HD64981F ESIC [Hitachi 91] to implement the EISA protocol. As well as the ESIC registers, which are located at 0xzc80, where 'z' is the EISA slot number, the EATM Interface has eight 32 bit registers which are IO mapped, and in addition can be memory mapped. The IO mappings of these registers are shown in table 1. When these register are memory mapped they will appear in contiguous memory starting from the base address configured in the ESIC.

These registers and their use are decribed in detail below.

2.1.1 STATCON Register

The status register has the format shown in table 2. The meanings of the individual bits are as follows:

- RXCELL: This read only bit, when set, indicates that at least one complete received cell is available in the EATM Interface .
- RXHDRERR: This read only bit, when set, indicates that the last cell read from the PATM had a header error check.

EISA ATM Interface Card
DRAFT 0.0

Ian M Leslie
Derek McAuley

February 18, 1993

This document is preliminary. The interface is not completely designed at time of writing and significant changes are to be expected. A new version of this document will be issued when the first prototypes have been made to work. This document should not be used as a basis for writing device drivers.

1 Overview

The EATM Interface is an EISA compliant ATM interface card. The physical transmission system used is ATM Forum 100 Mbps on TAXI 4B5B coding. The interface presented to the EISA bus is a simple cell interface with some support for AAL5. It can be accessed as a memory or IO slave. The board does *not* perform segmentation and reassembly.

The board will do per cell CRC-32 (as used in AAL5) calculations in both directions. It will be able to include a previous CRC residue in these calculations. For non VCI interleaved transmission and non VCI interleaved reception no special action is required by host software to chain CRC residues across an entire AAL5 unit. Host software will be required to save and restore CRC residues when VCI interleaveing within an AAL5 unit is performed. More details can be found in section 2 below.

The board will also be able to interrupt on a number of conditions. These include, received cell, received cell with ATM User User bit set, received cell FIFO over threshold, transmission cell FIFO under treshold and various error conditions. FIFO thresholds will be programmable by the host.

The major design goal of the interface is low cost. The first version will be built using Xilinx devices. The major components on the card are

- ESIC: This Hitachi ESIA Slave Interface Controller device handles the EISA bus peculiarities providing control signals to the main EATM Interface controller. It handles all issues to do with address configuration.
- EATM Interface controller: This Xilinx device is the heart of the interface and:
  - performs CRC generation in both tx and rx directions
  - contains all data and non-ESIC control registers