YES V2: Preliminary Performance

Eoin Hyden and Ian Pratt

February 17, 1993

Abstract

This document presents the results of some experiments performed using various configurations of: DECstation 5000/25[1] with YES board; Fairisle port controller with switch; and ATM camera.

1 Introduction

The recent availability of an ATM interface for a general purpose workstation such as the DECstation 5000/25 has enabled us to try out parts of the Fairisle network in a manner not previously possible. For example, the processing power of the DECstation, coupled with the system high-resolution timer mean that it can be used to generate traffic for testing parts of the network[2].

This document presents the results of two simple experiments involving the YES board. The first involves the display of a video stream generated by the ATM camera and the second measures data throughput between two user-mode processes which communicate via a YES V2 board which has its transmitter output looped back to is receiver input. These experiments are not meant to be definitive; their main purposes are to show that the equipment behaves as expected and to provide an indication of what to expect in the future.

2 ATM Camera Video

In this experiment, the ATM camera was connected to a Fairisle port controller. The port controller in turn was connected to a YES V2 interface via a 100Mbit/s TAXI link, and the YES board was installed in the DECstation 5000/25. The DECstation was running the Wanda kernel.

A user-mode application was written to display a single stream of 8-bit colour video[3] from the camera. The application controls the camera by opening an MSNL connection to it and sending control cells which the camera receives and interprets.

After configuring the camera, the application sits in a loop requesting the next field of video from the camera then displaying it on the DECstation colour frame buffer. When it transmits a field of video, the camera digitizes a scan line at a time and transmits this as a sequence of cells. Each scan line appears one the ATM interface as one segment delineated by the appropriate FAS start and stop bits.

An interesting aspect of the ATM camera is that it buffers at most two video scan lines, so it injects into the network, segments of digitized scan lines about 64 μs apart.

[1] 25MHz MIPS R3000 processor with 64KB instruction cache, 64KB data cache and 12.5MHz TURBOchannel.

[2] Such work is described in another part of this book.

[3] Each pixel in the video is represented by 3 bits of red, 3 bits of green and 2 bits of blue.

This means that the system network code on the receiving end (i.e. the Wanda/MSNL/yes code) must be able to receive, decode and store one segment of cells at least every 64 μs[4].

2.1 Results

The following results were measured by attaching a logic analyser to the read strobe and write strobe pins of one of the receive FIFOs on the YES board while the application was displaying a video stream. The stream consisted of frames of 288 scan lines. Each scan line was sent as a segment of 9 cells. The first cell of a scan line is used to contain header information such as frame and line number, and the rest of the cells contain 48 bytes of pixels, so the scan lines in this experment are $(9-1) \times 48 = 384$ pixels wide. While displaying this size of image, the application achieved a frame rate of 12.5 frames per second.

During reception, the YES driver can process scan lines from the receive FIFO into Wanda IO buffers such that the time between reading the FAS for the first cell in consecutive lines is 72 μs. This means that the raw bandwidth (i.e. the bandwidth occupied by the 4-byte cell headers as well as the payload, giving 52 bytes per cell), averaged over one scan line is

$$9 \text{ cell} \times \frac{52 \text{ byte}}{\text{cell}} \times \frac{8 \text{ bit}}{\text{byte}} \times \frac{1}{72 \times 10^{-6} \text{ second}} = 52.0 \times 10^6 \text{ bit/second}$$

and the bandwidth occupied by user data (i.e. 48-byte payloads only) is

$$9 \text{ cell} \times \frac{48 \text{ byte}}{\text{cell}} \times \frac{8 \text{ bit}}{\text{byte}} \times \frac{1}{72 \times 10^{-6} \text{ second}} = 48.0 \times 10^6 \text{ bit/second}$$

After reception of the first cell header, the YES driver uses some CPU time to locate the appropriate association structure for which the segment being received is destined and allocate an IO buffer. Similarly, after the last cell of the segment has been received, the driver uses some CPU time to call up through the MSNL protocol code with the user data (scan line) which it has just received. After reception of the first cell, the YES driver code enters a fast path during which it assumes that, more often than not, the next cell to arrive at the interface will be for the same VCI as the previous cell. While in this fast path, cells can be extracted from the receive FIFO without the overheads of locating association structures, allocating IO buffers and upcalls so, during this time, higher bandwidths are observed between the receive FIFO and memory.

Figure 1 depicts a typical trace as generated using a logic analyser to record the behaviour of the read strobe on the YES interface's receive FIFO.

In this figure, the cell header is read from the YES receive FIFO by the processor, generating a 200 ns-wide programmed IO cycle. The driver takes 1.8 μs to check the header information and initiate a 12-word DMA of the payload which takes 1 μs. The remaining 2.4 μs are consumed by updating pointers and counters and checking for the end of a segment and availability of the next cell. During this period, the raw bandwidth measured is

$$1 \text{ cell} \times \frac{52 \text{ byte}}{\text{cell}} \times \frac{8 \text{ bit}}{\text{byte}} \times \frac{1}{5.4 \times 10^{-6} \text{ second}} = 77.0 \times 10^6 \text{ bit/second}$$

and the bandwidth occupied by user data is

$$1 \text{ cell} \times \frac{48 \text{ byte}}{\text{cell}} \times \frac{8 \text{ bit}}{\text{byte}} \times \frac{1}{5.4 \times 10^{-6} \text{ second}} = 71.1 \times 10^6 \text{ bit/second}$$

[4]Some leeway is provided by the receive FIFOs on the YES board.

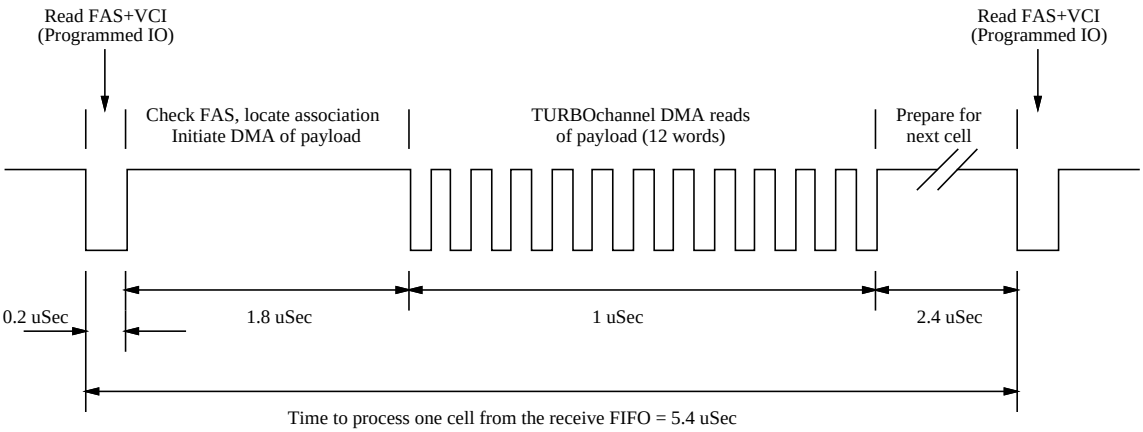

Figure 1: Timing of cell reception by Wanda/YES driver software.

3 Loopback Experiment

In this experiment the transmit and receive sides of the YES interface were connected together via a loopback connector. The DECstation was running the Wanda kernel with two user-level applications: a source and a sink. Messages were generated by the sink and stamped with the time as read from a timer of 1 μs resolution. They were then sent to the sink via the YES interface.

Upon reception of a segment, the timestamp was read by the sink, compared with the current time as read from the timer and the time taken for the message to travel from the source to the sink was recorded.

Reception of the segment was acknowledged by returning a single acknowledgement cell to the source. Upon reception of the acknowledgement by the source, the cycle was repeated. After 10^4 segments had been sent, the average trip time of the messages was recorded.

3.1 Results

The throughputs measured for messages of various sizes are plotted in figure 2.

Sampling the read strobes and write strobes YES board FIFOs during the experiment gave an indication of the comparative data rates into and out of the YES board. For a message of size 22 cells, the time to load the cells into the YES transmit FIFO from memory was measured to be 73.0 μs and the time to read the same cells from the YES board receive FIFO was measured to be 137.2 μs[5].

These times can be used to calculate the raw bandwidths obtainable by the driver in both directions between memory and the YES transmit and receive FIFOs. In the case of the receive side, when data is moving from the receive FIFO to memory, the raw bandwidth (user data plus cell headers) obtained is

$$22 \text{ cell} \times \frac{52 \text{ byte}}{\text{cell}} \times \frac{8 \text{ bit}}{\text{byte}} \times \frac{1}{137.2 \times 10^{-6} \text{ second}} = 66.7 \times 10^6 \text{ bit/second}$$

[5]The difference is caused by the extra processing required by the driver receive side. The receiver has to read the first word from the FIFO using a (slow) programmed IO cycle, check the FAS and can then DMA the remaining 12 words. The transmit side writes the FAS information before the start of the message data, taking advantage of the Wanda IO buffer design, then DMAs 13 words.

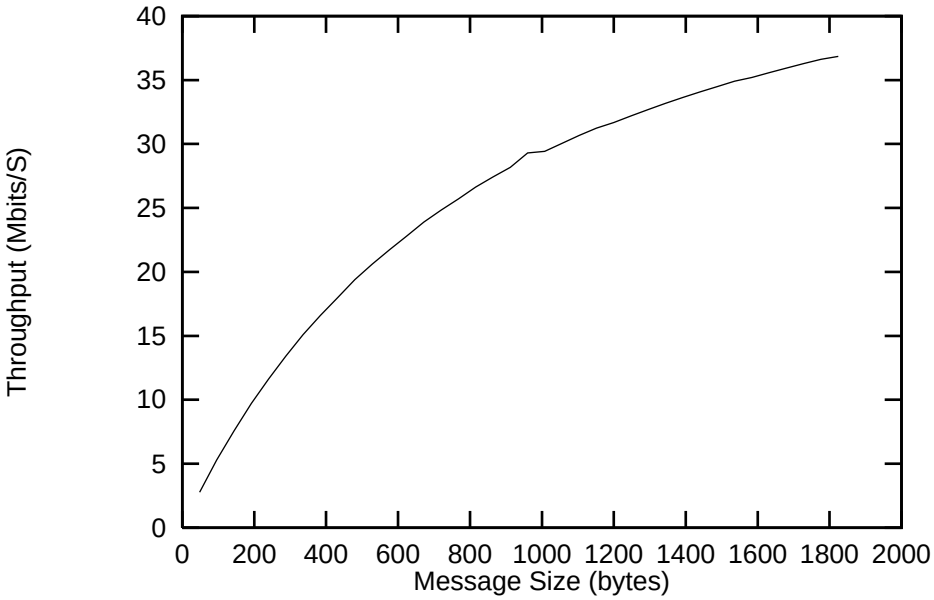

Figure 2: Throughput between two user processes versus message size.

and the bandwidth occupied by user data is

$$1\text{cell} \times \frac{48 \text{ byte}}{\text{cell}} \times \frac{8 \text{ bit}}{\text{byte}} \times \frac{1}{137.2 \times 10^{-6} \text{ second}} = 59.7 \times 10^6 \text{ bit/second}$$

while in the case of the transmit side, when data is moving from memory to the YES board's transmit FIFO, the raw bandwidth (user data plus cell headers) is

$$22 \text{ cell} \times \frac{52 \text{ byte}}{\text{cell}} \times \frac{8 \text{ bit}}{\text{byte}} \times \frac{1}{73.0 \times 10^{-6} \text{ second}} = 125.4 \times 10^6 \text{ bit/second}$$

and the bandwidth occupied by user data is

$$1 \text{ cell} \times \frac{48 \text{ byte}}{\text{cell}} \times \frac{8 \text{ bit}}{\text{byte}} \times \frac{1}{73.0 \times 10^{-6} \text{ second}} = 112.2 \times 10^6 \text{ bit/second}$$

The interesting point about this result is that the data rate out of the transmit FIFO going to the TAXI chip is less than 100Mbits/s, so it is possible to fill up the transmit FIFO with transfers as small as 13 words.

4 Summary

The two experiments presented have provided some early measurements of the performance of the YES V2 ATM interface in a DECstation/Wanda environment. It is worth keeping in mind that these numbers are likely to change as we gain more experience with the YES V2 interface. This is particularly the case with respect to the design of future kernel and device driver software.