FPC2 Performance Measurements

Ian Pratt and Eoin Hyden

18th February 1993

Abstract

Although in use for some time, the maximum sustainable throughput of the FPC2 had not been measured accurately until this series of experiments. A number of different hardware configurations were tried in order to try and establish which factors most affect performance.

1 The Aim

We have taken 'maximum sustainable throughput' to be "the maximum data rate that can be forwarded continuously arriving on a single VCI, given no switch contention or other problems".

Due to the nature of the hardware however, the single VCI constraint is not a limiting one: the same rate could be sustained across many VCIs. Fabric contention resulting in NACKS obviously reduces the port controller throughput, so was ignored in this experiment. 'Other problems' covers things like errors on the transmission line. Many million cells have been shipped without error, but when one does occur (usually artificially generated), the FPC2 currently attempts to print information about its occurrence. This distracts it from cell forwarding, and would affect results. During all of these experiments, no transmission line bit errors were ever detected.

All data rates quoted in this document refer to payload data rates i.e. 48 octet cells. This was done because it was thought to be more meaningful than quoting rates for cells with headers.

The maximum payload data rate of the TAXI transmission links is 87.3 Mb/s. Ideally, a port controller should be able to cell forward at this rate, and have some CPU time left over for connection management etc.

Before these experiments it was predicted that the bottleneck in cell forwarding was software running on the ARM CPU. When the rate of cells arriving exceeds the rate at which they can be processed, cells will back up in the FPC2's 2000 cell input buffer. If the buffer fills, this is detected. These experiments reveal that with faster ARM parts, this is not always the case.

2 The Experimental Setup

A 4x4 switch fabric with three FPC2s was used for the experiments. One FPC was simply an Ethernet gateway to allow booting of the other two, and played no part in the experiment. The other two were both connected to ORL YES V2 boards in DEC Station 5000/25s via approximately 50m of twin coax. Both DEC Stations were running Wanda, but with different user processes. One ran a traffic source process, the other a traffic sink.

All the FPCs were running kernels with gateways and RARP daemons spliced on. The host info was setup with MSNL routes for both fabric and transmission in the standard way. The two FPCs connected to DEC Stations were given gateway controls that allowed them to forward on behalf of their attached machines.

The DEC Station running the traffic source attempts to connect to a known port on the sink machine. The FPCs setup the connection, and give it a number of seconds to settle down (allowing print statements to occur etc). The traffic generator then produces a regular traffic pattern over the connection that speeds up slightly every 10 seconds.

The FPC2 connected to the traffic source must control the Xilinx to cause cells to be VCI remapped and then sent out on the fabric with the correct route byte. The FPC2 attached to the sink does no further work after the connection has been established - data arriving from the fabric passes through the passive side of the port controller out through the transmission interface to the traffic sink where it is dropped.

The traffic generator was capable of generating bursts of x cells every y μs. These cell bursts were at the full transmission line rate of 87.3 Mb/s. A particular setting of x and y was repeated continuously for approximately ten seconds, during which time at least a million cells are shipped. Should the FPC2 not be able to handle this rate, its 2000 cell input buffer would begin to fill as cells arrive at a faster rate than it can process. When the 2000 cell buffer becomes full, this is detected by the FPC2 software which reports the fact and halts. The previous data rate was the one that the FPC2 could handle continuously. It is possible for this data rate to be an over estimate - cells could be arriving slightly faster than leaving and backing up slowly in the 2000 cell buffer. Given that we are sending a million plus cells at each speed iteration, this over estimate is bounded by 2000 over a million, 0.2%.

If the FPC2 can successfully cope with the data rate, the y parameter is decremented, so that x cell bursts are sent every $y - 1$ μs.

The reason for the 'cells in burst' parameter was to see how the traffic profile affected the maximum sustainable throughput. The experiments were repeated with a range of burst sizes between 1 and 35 cells. Bursts longer than 35 cells were not easily generatable, and results show that the things had settled down by this size.

3 FIQ Measurements

All the cell forwarding is carried out in the FIQ (Fast Interrupt Request) handler in the Wanda kernel. For more details look in the Xi3 bits document, (section 18).

The FIQ handler is entered on receipt of a cell. It attempts to forward the cell, placing it on the transmit queue, and then checks to see whether another cell has arrived. The code to do this is a small piece of hand crafted assembler written by Richard Black. Under heavy load, the FPC2 never leaves the FIQ routine as there is always another cell ready. When in this state, the port controller is unable to process any management cells that may have arrived. They just sit on a 'cells for Wanda' queue in the 2000 cell buffer.

The time spent in the FIQ can be observed by attaching a probe to the FPC2's yellow LED. The FIQ illuminates the LED on entry and switches it off on exit. The original purpose of this LED was to detect crashes in the FIQ code. While the maximum throughput experiments were in progress, the data rate at which the FPC2 began spending all of its time in the FIQ was noted.

All experiments were performed with the same Xi3 Xilinx firmware bits, and the same Wanda kernels. Attempts were made later on in the series of experiments to achieve a 20 MHz switch fabric speed instead of the usual 18 MHz, but this proved unreliable and was abandoned.

4 Results

The experiment was first performed on the lowliest of port controllers, 'Sole', an ARM 3 powered device with an FCLK (CPU clock) speed of 25 MHz and MCLK (Memory system clock) of 8.33 MHz. The fabric back plane speed was 18 MHz. A range of burst sizes were tried yielding the following results:

Note how as the burst size increases the CPU time is used more efficiently ie. the data rate required

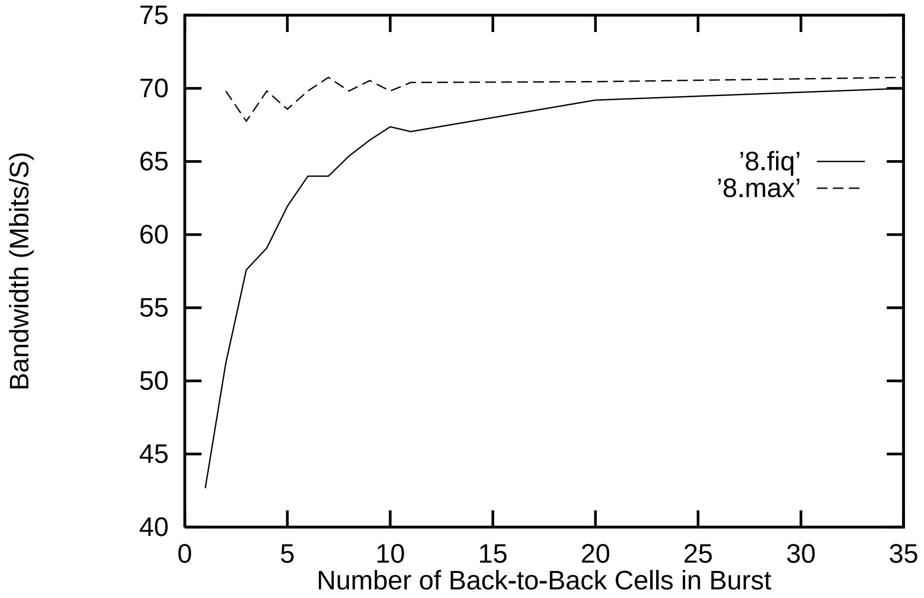

Figure 1: Burst size versus Bandwidth for ARM 3 CPU, FCLK = 25 MHz, MCLK = 8.33 MHz, FABRIC = 18 MHz.

to cause it to stay in the FIQ is higher for larger burst sizes. Note how the 'maximum sustainable throughput' figure is fairly constant over burst size. The larger bursts allow finer control over the data rate by the microsecond timer, and provide a more refined maximum figure.

Next, 'Sole' was upgraded to use a 10 MHz MCLK. Since all FPC2s have 10 MHZ MEMCs (Memory Controllers), this could (and should) be done to all existing FPC2s. This represents the 'standard' configuration for an FPC2, and is the most common amongst the ones we have.

Increasing the MCLK speed by 20% resulted in a 4% improvement in maximum throughput. This was thought quite disappointing, given that it was thought that the CPU was bound by memory accesses. This was apparently the case for Xi2 bits, but Xi3 bits requires fewer memory cycles. Note however how faster memory accesses speed up the FIQ context switch time, resulting in a sharper FIQ curve.

There exists a ceramic version of the ARM 3 CPU that is rated at 30 MHz and was used in the next experiment. With a 10 MHz MCLK it was found to achieve a maximum sustainable throughput of 76.8 Mb/s. This is 5% better than the 25 MHz ARM 3 with the same 10 MHz MCLK. Again this was a little surprising as a more dramatic speed up was expected.

Future port controller such as FPC3 will use ARM 6xx CPUs. There exists a single ARM 600 FPC2 that was used for this experiment. The ARM 6 has basically the same processor core as the ARM 3, except for a 32 bit address space. Instruction execution times are the same, and they both implement the same write through cache. The advantage of the ARM 600 is that it has an eight entry write buffer, which allows the CPU to continue working at FCLK speeds while writes are pending rather than having to slow to MCLK speeds on every write.

Due to a beta silicon bug on our ARM 600, we were only able to run MCLK at 8.33 MHz. FCLK was set at the maximum 25 MHz, the same as for ARM 3s. As can be seen from the results, the write buffer really does make a difference. In fact, the ARM 600 is sufficiently fast such that the FIQ code is no longer the bottle neck. The failure of the FPC2 in cell forwarding above 82.5 Mb/s is in fact due to a hardware failure of the Xi3 bits when operating at 18 MHz. At this speed they are unable to read data from the transmission line FIFO fast enough to avoid it filling.

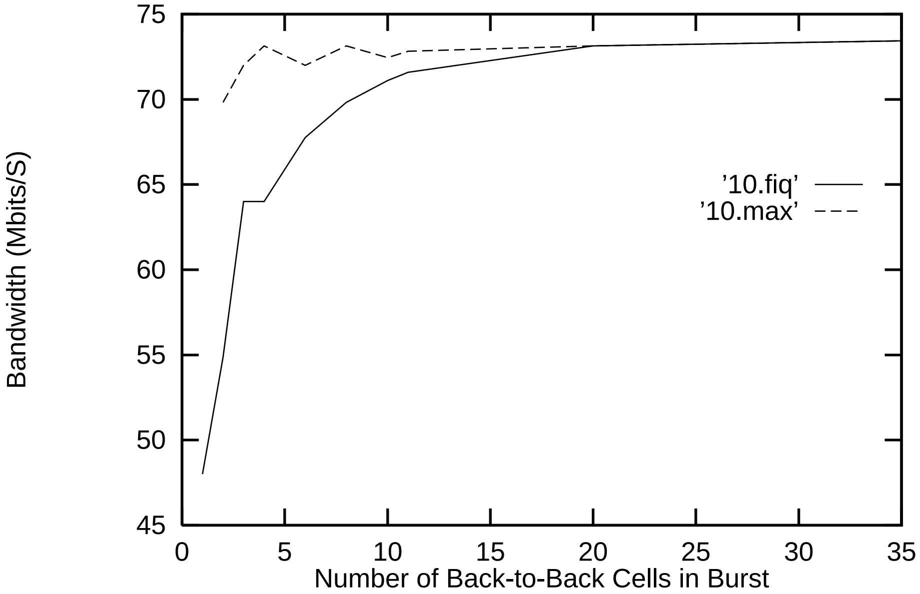

Figure 2: Burst size versus Bandwidth for ARM 3 CPU, FCLK = 25 MHz, MCLK = 10.0 MHz, FABRIC = 18 MHz.

This problem could only be fixed by up-rating the fabric speed to $\geq$ 19.1 MHz, where the Xilinx would be able to drain the FIFO at the full 87.3 Mb/s transmission line rate. Faster Xilinx designs capable of 20 MHz operation have been produced in the past, and probably could be again. The reason for this not being done is that it was not holding anyone's research up, and nothing was avaliable that could generate (or more to the point, receive) at such high data rates.

A summary of throughput results follows:

Processor	FCLK (MHz)	MCLK (MHz)	FABRIC (MHz)	Max Throughput (Mb/s)
ARM 3	8.3	25.0	18.0	70.7
ARM 3	10.0	25.0	18.0	73.4
ARM 3	10.0	30.0	18.0	76.8
ARM 6	8.3	25.0	18.0	82.5

Table 1: Maximum Sustainable Throughput figures

5 Effect of switch fabric speed

As mentioned previously, an 18 MHz fabric speed becomes a bottle neck at around 82.5 Mb/s. To ensure that this effect was understood, the ARM 6 experiment was repeated at 16.7, 15.0 and 13.33 MHz fabric speeds. The results scaled precisely linearly as expected:

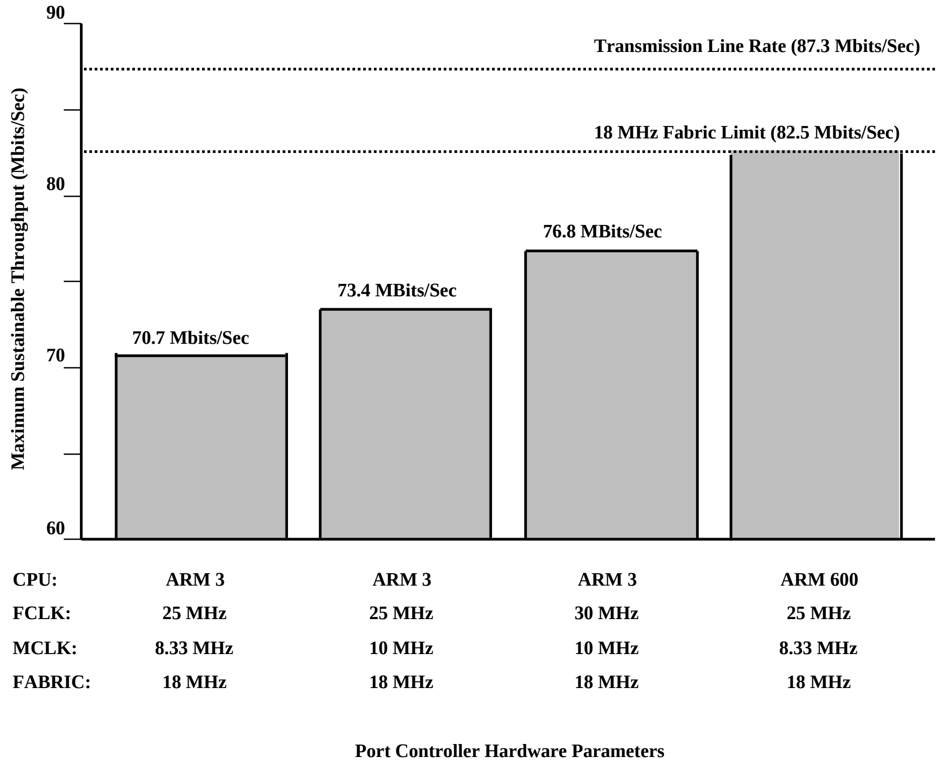

Figure 3: Maximum Sustainable Throughput for Various Configurations.

Fabric Speed (MHz)	Max Throughput (Mb/s)
18.0	82.5
16.7	76.4
15.0	68.8
13.3	61.0

Table 2: Maximum Sustainable Throughput figures.

6 Conclusions

The FPC2 works faster than had been originally anticipated. For a 16x16 switch we thus have a total throughput of over 1.3 Gigabits/Sec. These results suggest that it is quite possible that an ARM610 powered FPC3 will achieve the full TAXI line rate.

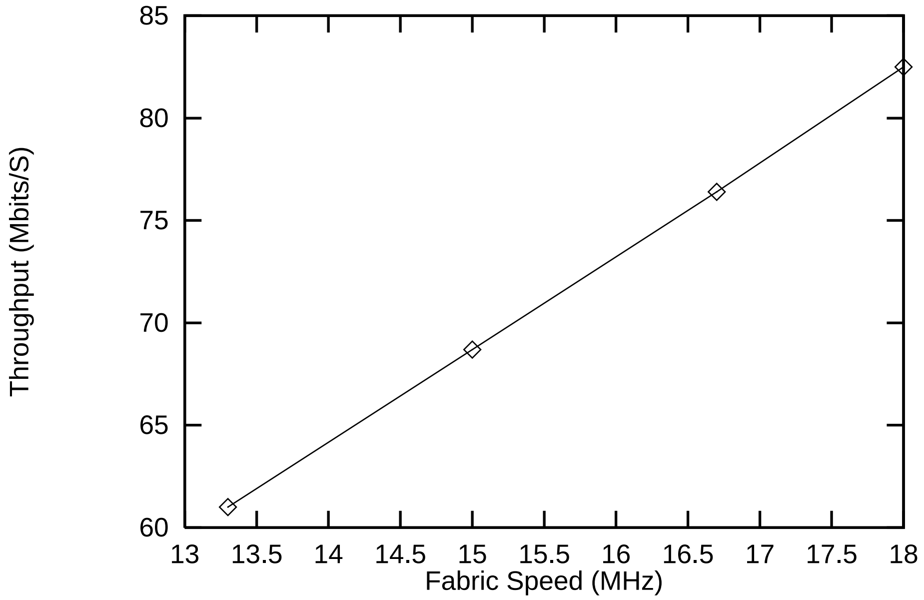

Figure 4: Maximum Sustainable Throughput versus Fabric Clock Speed.