FPC2 xilinx chip - Xi3
Design and Software

Mark Hayter and Richard Black

10th March 1994

Following the initial experiments on the FPC2 the port controller xilinx chip was redesigned resulting in the "Xi3" design. This document describes the interface of the Xi3 chip, and the current Wanda driving software. This version is the one typically currently used on the FPC2. The Xi4 design is a derivative used only for hardware testing and Xi4 bits should not be used in any port controllers. The Xi6 design is a retro fit of some of the compatibility fixes of Xi5 to the X3 design. At the date of writing the Xi6 design is less used mostly because of a lack of software support.

1 Changes made for Xi3 xilinx

The Xi3 xilinx chip for the port controller is almost a complete rewrite of the hardware from the earlier Xi2 device. There has been some "cut and paste" from the previous design, and some libraries are common, but most of the HDL files have been completely rewritten. The main noticable changes are:

- The system of transmit and receive queues in cell buffer 0 has been replaced by a single linked list, managed by the hardware.
- Data bus D[0..15] and D[30..31] are taken into the xilinx. These provide access to several registers.
- The status register has been replaced. Decoding of the address bus allows selection of what status information is to be read. In particular the interrupt request bits and the received buffer number may be read in a single operation.
- The arbiter has been changed (again). The word2fabric section now demands memory access. In other cycles the arm and fifo2word section get alternately offered use of memory. If the fifo2word section uses the offered cycle then it will not be offered memory in the next four cycles since it is known that it will not use them. A hacked up simulation indicates that this policy fractionally increases the expected wait for arm and f2w accesses if data is being moved at full speed. It makes the design simpler so Im going to do it, as it can always be changed later.
- The transmit buffer number now includes a retry count. The cell will be retried on NACK until the number of NACKs is equal to the retry count when the transmission system stops and interrupts.
- Only the "slowarm" interface is used.
- The rx jam condition has been fixed. Cells that do not come in fast enough are timed out.
- The NACK control logic has been put in a single module so different versions can be easily tried.

- A timestamp is put in word 0 of every rx cell. This has an 11 bit count which increments on every frame pulse, and also includes source information.
- A transmit cell can be marked for return to free queue without transmission. Note this process still takes a frame time.
- A cell can be requested from the free queue. It is marked by a bit in the VCI.
- The "goXilinx" strobe is done by writing a bit in the command register - this needs only one write rather than two through IOC. The IOC method is no longer possible.
- The detection of empty fifos has been changed. In particular the metastable state caused by the fifo briefly becomming empty should be less of a problem.

2 The Cell Buffer

As before the SRAM is divided up into cell buffers. Each is 16 words long. Figure 1 shows the organisation of a cell buffer as used for both transmission and reception. In an outgoing cell that

	Word 0	Word 1		Word 2..13	Word 14	Word 15
Transmit	In Local cell Route Fab PC VCIout	Remap index	FAS	Data	Link pointer to next buffer in Queue (may contain transmit details)	Remap vector Route Fab PC VCIout
Receive	Time Stamp	VCIin	FAS			

Figure 1: Cell Buffer Format.

is not marked as local the remap index is used as the "buffer number" to fetch the remap vector information from. *From December 1992 use of the local bit is discouraged as it will not be supported in future versions - all cells will go through the remap process.* The timestamp field is written with the value of the TimeStamp register just after the final data word is written.

3 The queue

The cell buffers fall into four classes: waiting collection by the reception FIQ, on the free queue, on the transmit queue, and being used by software. The cell buffers in the first three classes are linked together by cell buffer numbers in their link field (currently word 14). These form a single list with pointers in the xilinx chip marking the boundaries as shown in figure 2. The queue must be set up by the processor before starting the xilinx. The queue is automatically maintained because the processor adds buffers for transmission to the tail. Once these are sent they form part of the free queue before being used for reception. Buffers not in the queue are assumed in use by software, and must be entirely managed by software. This works well while the processor is not a net source or sink of cells. Methods exist for doing fake transmissions and receptions to allow software to correct any imbalance.

4 Registers

The new xilinx chip has more registers than the previous version, some of these may be read and written by the processor. There are no checks that the processor is putting sensible values into the registers, or that writes are done at sensible times. The registers are:

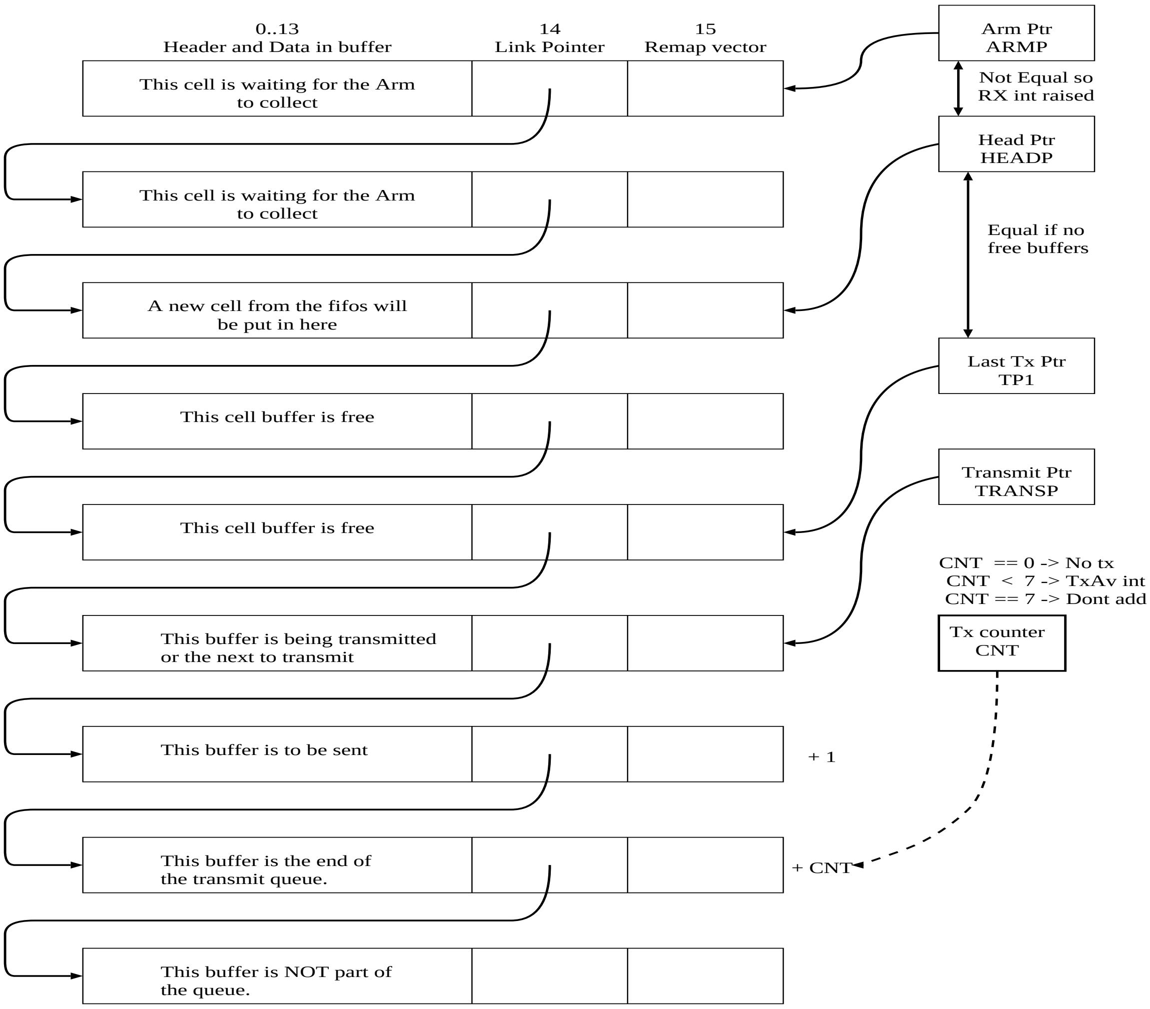

Figure 2: New xilinx queue structure.

ArmP. Points to the start of the queue structure. Buffers between ArmP and HeadP are ready to be collected by the processor. Can be read and written by processor, optionally reading may have the side effect of stepping it on to the next buffer in the queue if ArmP != HeadP.

HeadP. Points to the head of the free queue, and the cell that is currently being received. May be written by the processor. After a cell has been received HeadP steps on to the next buffer in the queue.

TP1. Points to the last buffer in the free queue - the transmit queue hangs off the link pointer of this buffer. May be read and written by the processor. If HeadP becomes equal to TP1 then the free queue is empty and an interrupt raised. While the free queue is empty cells will only be read from the RxFifo, any cells read will be discarded and the TOG status bit set.

TransP. Points to the buffer being transmitted, or that has just been transmitted (if there are no more to send). When this register is updated by the xilinx its old value is moved into TP1. May be written by the processor with no effect on TP1. TransP is advanced just before sending a cell (unless the cell is being automatically retried). After a NACK causing the retry limit to be reached TransP will point to the cell NACKed, and transmission will stop. To restart discarding the NACKed cell a “GoTx” command should be issued. To restart with further retries TransP should be loaded with the value read from TP1 and CNT incremented before the “GoTx” command. To reorder the transmit queue the new queue should be built under the link pointer from the buffer pointed to by TP1 and TransP written with the value from TP1 - note that since TP1 is unaltered by processor writes of TransP it will remain valid even in the case of an immediate NACK. TransP, and thus the cell link pointer, also contains two control bits **local** which when set prevents VCI remapping taking place[1] and **NoTx** which marks the cell for dummy transmission. A dummy transmission returns the cell buffer to the free queue by passing it through the transmission machinery without injecting it into the fabric – this still takes a whole frame period.

Remap. The remap register is internal to the xilinx and provides a temporary store for the VCI used in a VCI remap operation. The remap register is short lived, and relies on the pipeline for the correct address to be formed.

CNT. The CNT register stores the number of outstanding transmission requests, ie the number of valid buffers after TransP. The processor may increment the count as a side effect of writing memory or registers. The transmit available interrupt is raised if the count is less than 8, the counter will not increment if this is not the case. This is done to restrict the fifo queueing in hardware to 8 buffers, allowing software control of larger queues.

Command. The command register is writable by the processor to issue commands to the xilinx chip. These currently consist of “goTx” which starts the transmission system (this replaces pulsing the ramRdGo signal), and “RqW” which requests a buffer from the free queue for use by Wanda. These two commands are both initiated by writing the command register with a particular bit set, these bits are not stored so there is no need for the set-wait-clear that was required on the previous version of the chip. A buffer request from Wanda is serviced whenever the receive system is looking for a start of cell. A cell for Wanda has the timestamp put in words 0 and 1 of the buffer, thus ensuring that bit 15 of the VCI is set so it can be recognised. Only one request may be outstanding at any time, so Wanda should wait for one to be serviced before issuing another, or regard the request as an unreliable operation.

TimeStamp. The TimeStamp register is inserted in word 0 of a successfully received cell buffer just prior to advancing HeadP - this happens whether the buffer was truly recieved or was requested by Wanda. The TimeStamp contains an 11 bit count, which increments every fabric frame pulse (ie 64 clock cycles). The top two bits indicate the source of the cell (for Wanda, rx fifo, loopback fifo) to assist debugging. Also the “latched status” bits are inserted for reasons that I don’t propose to go in to! (Actually these may be useful for debugging too). Bit 15 of the TimeStamp is always 1, so that the timestamp can be used as the “VCI/FAS” word for cells to Wanda. If the timestamp is read by the processor (rather than being read indirectly via a recieved cell) the source information will probably not be useful since it is only valid when a cell is actually being recieved.

Status. The status register is read only and can be read with either ArmP or TP1. Most of the bits in the status register record conditions that have been seen by the xilinx, these are latched until a *status-read-and-clear* is done (allowing low level software to examine the status but only higher levels to record details).

[1] Use of this bit is now discouraged

It should be noted that a cell timeout (TMO) error is likely to provoke stray bytes (BYT)[2], and a cell thrown on ground (TOG) error will always provoke stray bytes.

IrqStat. The IrqStat register is read only and can be read with either ArmP or TP1. When read with ArmP there is an option to cause ArmP to follow the link to the next cell buffer. This reload will only happen if ArmP != HeadP (ie there are cells to collect). The information in IrqStat is an expanded form of the interrupt request lines presented to IOC, thus after an interrupt one read will get both the reason for interrupt and collect the buffer if it was a receive interrupt.

Figure 3 shows the format of the various registers. The xilinx chip has no access to data bits 16..29 so these are undefined on read and may be written with any value.

	31	30	29..16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TransP				Local	Retry count			NoTx	Buffer number to transmit										
Status (irq)	Source		Any	Sta	TxS	FrE	TxA	Rx	ArmP if A17==0, TP1 if A17==1										
Status (status)	Source		Any	togo	tmo	tog	byt	soc	ArmP if A17==0, TP1 if A17==1										
TimeStamp	Source		Any	1	tmo	tog	byt	soc	Count, increments every frame pulse										
Command Reg														RqW				GoTx	
ArmP/HeadP/Tp1/Remap									Buffer number										

Local - Cell source is local, don't VCI remap
Retry Count - Number of times to retry on NACK
NoTx - Don't transmit, just put on free queue

togo - There are cells to be transmitted
tmo - (latched) A cell was timed out on rx
tog - (latched) A cell was lost since no buffers
byt - (latched) Stray bytes were seen
soc - (latched) Extra SOCs were seen

Source: 00 - LpFifo, 01 RxFifo, 10 WandaRq

RqW - Set to request cell for Wanda
GoTx - Set to enable transmission system

Sta - Latched status bit(s) set
TxS - Transmit stopped, ie RESET or NACK
FrE - Free Queue is empty
TxA - Transmit buffers availible
Rx - Recieve buffers to collect

Figure 3: Transmit buffer pointer and status regs.

5 ARM accesses

The processor interface is based on the "slowarm" interface developed for the previous version of the xilinx design. This ensures resynchronisation at all possible points and thus allows any combination of processor and fabric clock speeds. This is done at the expense of making accesses a minimum of three processor REF8M cycles.

The access address is (incompletely) decoded to give access to the buffer SRAM and the xilinx registers. Tables 1 and 2 give the addresses used to decode read and write accesses. Write accesses to a register use address bits to select which register is written, by setting more than one of these bits several registers may be written (with the same value) in one cycle.

[2]The xi3tester method of generating short cells also causes extra SOCs

A18	A17	A16	A15	A14	Base	Notes
0	x	x	x	x	0x3180000	Read memory
1	0	0	0	0	0x31C0000	Read Armp, irqstat, reload ArmP
1	0	0	1	0	0x31C8000	Read Armp, irqstat, no reload
1	0	1	0	0	0x31D0000	Read Armp, status, no reload
1	0	1	1	0	0x31D8000	Read Armp, status, clear latched
1	1	0	0	0	0x31E0000	Read TP1, irqstat
1	1	0	1	0	0x31E8000	Read TP1, irqstat
1	1	1	0	0	0x31F0000	Read TP1, status
1	1	1	1	0	0x31F8000	Read TP1, status, clear latched
1	x	x	x	1	0x31F4000	Read TimeStamp

Table 1: ARM read addresses

A18 (reg)	A17 (inc)	Select Bit	Base Adr	Notes
0	0	x	0x3180000	Write memory
0	1	x	0x31A0000	Write memory and inc CNT
1	1	A10	0x31C0400	Write HeadP
1	1	A14	0x31C4000	Write ArmP
1	1	A15	0x31C8000	Write TransP
1	1	A16	0x31D0000	Write CmdReg
1	1	A10	0x31E0400	Write HeadP and inc CNT
1	1	A14	0x31E4000	Write ArmP and inc CNT
1	1	A15	0x31E8000	Write TransP and inc CNT
1	1	A16	0x31F0000	Write CmdReg and inc CNT

Table 2: ARM write addresses

The processor has to setup the queue structure before starting the xilinx chip. The chip resets with all registers equal to zero and transmission stopped. HeadP will be equal to TP1 so reception is also disabled. Thus the last step of the initialisation code is to set HeadP and issue a "GoTx" command. The processor must save the buffer number of the tail of the transmission list since this information is not stored anywhere else.

To transmit a buffer the processor should check that there are transmit slots available. When there are the new buffer is added to the transmit queue by writing its buffer number and transmit flags into the link word of the transmission tail - this write should have the "increment CNT" bit set in its address. A write with increment when the transmit queue is full will not change CNT, but *will* modify memory, if this happens things are likely to go wrong.

When a receive interrupt is seen the buffer number may be obtained by reading the value of ArmP. If this read is done with reload then ArmP will be moved to the next queue entry, and the receive interrupt bit will be updated. Since a reload is only performed if there was a buffer to be collected it is safe to do a read-with-reload of the IrqStat/ArmP register at the start of an ISR, allowing a single read to determine both the interrupt reason and collect the next buffer.

6 Interrupts

Three separately maskable FIQ interrupts may be signaled by the xilinx to IOC. Table 3 gives the IOC FIQ bit details. The Error interupt is raised if the transmission system stops (because of a reset or NACK) or the free queue becomes empty. The IrqStat register may be read to determine the cause of the error. For the meaning of the other bits in this register see the general port controller document.

bit	Name	Use
0	Xifiq[0]	Set on Rx Cell
1	Xifiq[1]	Set on Transmit Availible
2	Xifiq[2]	Set on Error Condition

Table 3: IOC FIQ Status register

7 Known problems with the Xi3 bits

The Xi3 timing out of cells has caused problems when correct but slow cells are recieved. This is seen for example when the source is a "Yes" board using PIO on its transmit side. There is a version of Xi3 with timing out disabled, it is intended that one be generated where two frame pulses have to occur between incomming bytes to generate a timeout.

The currently used versions of the Xi3 bits (which all appear to operate at 18 MHz) are:

T7Hmilton07 Basic version. Timeout enabled on cells, endgame problem.

T7HmiHACK T7Hmilton07 with timeout disabled.

upX4 Endgame problem fixed, timeout enabled on cells.

TYUlelaps01 Timeout enabled on bytes but broken, other known bugs fixed.

TYUle01HACK Timeout disabled, other known bugs fixed.

8 The interrupt handler

8.1 Issues

Many of the issues which caused problems in the Xi2 design have been removed however these changes have introduced subtle changes to the design requirements for other parts of the system.

8.1.1 Delay free and Queues

The delay free problem has been completely removed by the handling of the free list entirely within the xilinx chip. The xilinx chip now automatically frees a cell buffer when it is actually transmitted simply by loading the old value of TransP into TP1 on the next relaod of TransP. This removes the problem of management of the free list from the software. It adds some complexity, however, in the case when the Wanda network management software needs to free a cell. This is because a special "NoTX" transmission must be performed.

The major change is in the form of the queues. It is no longer possible to have the queues with a dummy buffer element on them, as the freeing of buffers is no longer done on queue removal. Instead a queue must potentially contain zero buffers. This in turn means that the queue can no longer be designed to be safe in the presence of concurrent access and that it is now necessary for the Wanda networking code to disable FIQs when accessing the queues. This is less important than it was on the Xi2 design because no freeing of buffers is involved.

8.2 Data Structures

For the Xi3 design there are now only two data structres manintained by the interrupt software.

8.2.1 Cell Queues

Cell queues are used to put cells on when they arrive, and to take them off when they are to be transmitted. Some of these queues will be both generated and consumed by the FIQ code, and some will be generated or consumed by the Wanda networking code. There must also be a queue of buffers to be pseudo-transmitted, that is, transmitted without the data being sent "NoTX".

The Xi3 design software still maintains the queues as a head and tail pointer and chaining in the a map of indecies. The principal change from the Xi2 software is that NULL queues are now possible. They are represented by having a zero value in the head pointer.

Manipulation has changed so that for addition if the head is null then the new buffer number is written to both head and tail. This can be performed with a store multiple. Otherwise the link pointer is inserted into the previous tail and the new tail written.

For a removal operation if the head is zero then the queue is empty. There is also a special case if the head and the tail are equal because the queue then goes empty and the head must be written with zero. Otherwise the head is written with the link entry of the previous head.

Again no link entry of zero is ever required with this system aiding efficiency.

8.2.2 The forwarding map

The forwarding map has an identical form to the case of the Xi2 design when remapping is available. The following must be encoded:

- The queue on to which the buffer is to be placed. This is required for all entries in the map.
- Whether to reconsider servicing transmit available interrupts. As explained earlier normally the Xilinx transmit queue cannot be kept full and so the transmit available interrupt is frequently disabled. If the data is to be forwarded this flag must be set to cause despatching of this interrupt.
- Whether to post a normal IRQ interrupt to Wanda. This is if the queue is to be consumed by Wanda and will allow Wanda deal with the new data.
- The new VCI and routeing information that the cell should be transmitted with. This is only required if the Xilinx bits do not support hardware remapping. Again this is only required if the cell is to be forwarded.

It is noted that a single bit is required to indicate which interrupt to enable. Performance is improved by encoding this bit in the same word as the queue pointer. The word stored is in fact the queue pointer shifted left by one bit with the bottom bit used to encode which interrupt to perform. A single right shift both recovers the queue address and places the controlling bit in the carry flag for the conditional branch.

8.3 Algorithms

8.3.1 Interrpt dispatching

One of the major advantages of the Xi3 xilinx hardware is that it is designed to give as much information as possible to the software when it is trying to find out what the status of the chip is. Previously the software had to read the IOC FIQ status register and then dispatch on various conditions of this. However the new design allows a single slightly slower read cycle (ArmP, irqstat, reload) from the xilinx chip to to all of the following:

- Status of receive available condition
- Status of transmit available condition
- If transmission has stopped because of a Nack
- If a special condition interrupt (free list empty) has arrisen

- If there are any transmision line related error conditions (latched status) pending
- The buffer number of the cell at the head of the receive queue (if applicable)
- Removing the buffer that has arrived from the head of the receive queue (if applicable)

When one of these loads is done it necessitates that no matter what the other conditions are that a recieve buffer interrupt must be dispatched before the register is read again.

The priority of the dispatcher is to nack and special condition interrupts. If a special condition interrupt is pending then code takes a machine check (panic) just like the Xi2 software. The transmit available code takes the next higher priority (assuming it is enabled in the software interrupt mask) and receive takes least priority.

The latched status bit is only ever looked at when there is no other work to do. If it is set a normal IRQ is enabled allowing the higher level sceptic code to investigate the status register for the exact condition.

8.3.2 Arriving cells

This is essentially the same as Xi2 software except the buffer number is already available from the interrupt dispatcher. The VCI is recovered from the received cell and indexed into the forwarding map. The cell is added to the specified queue and the appropriate interrupt is enabled.

8.3.3 Departing cells

If enabled, on a transmit available interrupt the code searches a number of queues for a cell to transmit. This could in general be any number of queues but the current Wanda networking code uses only three queues; one for all traffic being cell forwarded through this node, one for management or other traffic being generated at this node, and the third for cells to be added to the free list by null transmission. If a queue is non empty the head cell is removed and written (together with the appropriate control flags) to the Xilinx transmit queue with a strobe. This atomic operation adds the buffer for transmission, increases the count of cells to be transmitted and also the data written makes up the link in what will become the free list when the buffer is freed.

The current position within the Xilinx transmit queue is kept in a FIQ state register. The new position is easily calculated after the write as it is simply the link entry in the buffer number just written.

As an additional efficiency, the code to remove a cell from any of the three queues is implemented as an inline examination of the heads of each of the queues in turn with conditional branching to a single section of code which actually does the removal. This ensures at most one pipline fill for the entire transmit available servicing code.

As noted above at the end of the tranmsit queue available interrupt the code must check the value read in the interrupt dispatcher (which is preserved in a register) and jump directly to the receive code if appropriate.

8.3.4 Blocked cells

The Fabric generates a NACK whenever there has been contention, either for an output port or due to blocking within the fabric, and this port controller's cell has lost. A NACK is also generated by the destination port controller if its fifo is becoming full. On seeing a NACK the Xilinx may try between zero and seven additional times before generating a FIQ interrupt and stopping transmission.

On Xi3 design hardware it is much easier to deal with a NACK interrupt than on Xi2. The possible options supported by the hardware are described in section 4.

The default software again implements a naive sixteen attempts for every cell. Eight of these attempts are performed on each occasion by the xilinx chip by setting the retry value in the previous link pointer to seven. However this means there must be retries at the software level.

Like the Xi2 software, the Xi3 software uses a mechanism which is not perfectly accurate but which is almost so and much cheaper to implement. The number of the buffer which caused the last nack is kept in a FIQ state register. When a nack occurs the buffer number which was nacked is read from TP1. If it was not the same as the last time then the state register is updated and TransP is written with the buffer number obtained from TP1, which causes it to be retried. In either case the xilinx command register is then written with a “GoTX” command which restarts transmission. This is highly efficient compared with the Xi2 case.

9 Performance

It is difficult to determine the exact performance of the port controller because there are so many differing factors which all influence the final figure. Amongst others there are:

- Cpu type - Arm 3 or Arm 600
- Cpu FCLK speed.
- Cpu MCLK speed.
- Fabric clock speed.
- Exact Xilinx bits version.
- Software changes as appropriate for configuration of above.
- Measurement error.

The performance is usually measured by the time the FIQ is active in dealing with a mini-burst of eight back to back cells on a circuit which is sending regular mini-bursts. A substantial number of samples are taken and averaged. The performance is based on the time and the number of user data bits passed - that is on the forty eight octets only.

Performances have been observed of more than 52 Mbits/sec for an Arm 3 and over 68 Mbits/sec for the Arm 600.