Evolution of S-LINK to PCI interfaces

Wieslaw Iwanski

Henryk Niewodniczanski Institute of Nuclear Physics, Radzikowskiego 152, 31-342 Krakow, Poland
wieslaw.iwanski@ifj.edu.pl

Markus Joos, Robert McLaren, Jorgen Petersen, Erik van der Bij

CERN, 1211 Geneva 23, Switzerland
markus.joos@cern.ch, robert.andrew.mclaren@cern.ch, jorgen.petersen@cern.ch, erik.van.der.bij@cern.ch

Abstract

The S-LINK is a CERN developed standard that defines a point-to-point data link. In many applications and test systems the data transmitted over the link is moved to a PCI based computer. An overview of the evolution of S-LINK to PCI interfaces is given. The performance of these interfaces is presented and a description of the FILAR, a future PCI interface with four integrated inputs, is given.

I. Introduction

The S-LINK is a standard that defines interfaces of source and destination sides of a point-to-point data link featuring a bandwidth of up to 160 Mbytes/s. A link complying with the S-LINK specification [1] can be thought of as a virtual cable that can move data or control words from any layer of front-end electronics to the next layer of read-out.

In many applications and test systems the data transmitted over the data link is moved to a PCI [2] based computer. The first Simple S-LINK to PCI interface (SSPCI) [3], designed in 1997, was intended for a 32-bit/33 MHz PCI bus. Since then the PCI has evolved to 64-bit and 66 MHz offering a bandwidth of up to 528 Mbytes/s and motherboards with several fast and wide PCI segments have become commercially available. For such computers the S32PCI64 [4] S-LINK to PCI interface has been designed. It is able to move data from one plugged-in S-LINK Destination Card (LDC) to a host computer. In this interface, the full potential of fast PCI (528 Mbytes/s) is not entirely utilised, as the limiting factor is the speed of the LDC (160 Mbytes/s).

For systems with several data inputs running at full S-LINK speed, the FILAR [5] interface is envisaged. This module, currently under design, will have four S-LINK LDC channels integrated with four S32PCI64-like cores within one PCI controller.

II. SSPCI

SSPCI was the first S-LINK to PCI interface. Its hardware design, based on the AMCC S5933 PCI controller [6], was simple but the host computer had to use a complex protocol for the transfer of S-LINK data packets. It required many PCI cycles resulting in a typical overhead of 8 μs per packet. With a block size of over 10 Kbytes, the overall performance reached 117 Mbytes/s. Some ATLAS detectors like Muon or TileCal have been using this interface in their past and recent test-beam set-ups.

III. S32PCI64

In order to decrease the software and protocol overhead and to have a better PCI bus utilization, the S32PCI64 interface has been designed. This commercially available module [7] is intended for a 64-bit/66 MHz PCI, which potentially allows a throughput that is four times higher than that of the SSPCI. The block diagram of the S32PCI64 interface is shown in Figure 1.

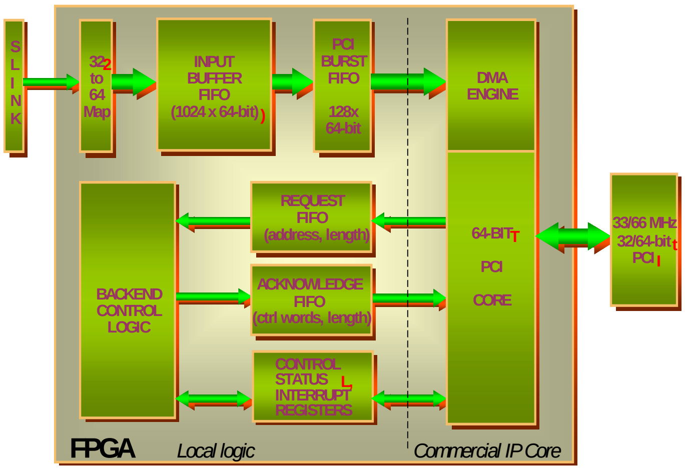
S L I N K
32 to 64 Map
INPUT BUFFER FIFO (1024 x 64-bit)
PCI BURST FIFO 128x 64-bit
DMA ENGINE
REQUEST FIFO (address, length)
ACKNOWLEDGE FIFO (ctrl words, length)
CONTROL STATUS INTERRUPT REGISTERS
BACKEND CONTROL LOGIC
64-BIT PCI CORE
33/66 MHz 32/64-bit PCI
FPGA
Local logic
Commercial IP Core

Figure 1. Block diagram of the S32PCI64

S-LINK and PCI connectors are directly connected to the FPGA hosting the firmware of the interface. Firmware consists of a 32/64-bit PCI core [8] and local logic. A commercially available PCI IP core provides the DMA engine and interface to the PCI, while the local logic receives the S-LINK data and controls the data movement.

Thirty-two-bit data words arriving from the S-LINK LDC are merged and moved to the Input Buffer FIFO. The depth of this FIFO allows the interface to accommodate up to 8 Kbytes of data. When the Input Buffer FIFO is 75% full, it will generate a flow control signal to the Link Destination Card so that the input buffer will not overflow.

The host processor can initialize the interface to receive up to 15 S-LINK data blocks by writing to the Request FIFO the PCI addresses where the data have to be stored and the maximum length of each data block to be received. After this, the interface can receive data blocks without needing any intervention of the processor.

As address and length parameters appear in the Request FIFO, the interface starts moving S-LINK data from the Input Buffer FIFO to the PCI Burst FIFO and counts the number of words. Only regular data words are moved, while the control words are extracted from the data stream and stored in the Acknowledge FIFO.

The depth of the PCI Burst FIFO is a compromise between the length of a typical data block in the ATLAS experiment at LHC and memory resources available in the FPGA hosting the interface logic. The PCI Burst FIFO is 128 words deep which allows the interface for bursts of 1kbytes size. This is the longest single PCI burst that the interface will perform. Data blocks larger than that will be segmented.

DMA transfers are fully autonomous; the interface becomes a Master on the PCI bus when it has received a request and data to move. When a current data block is moved entirely to the host's memory, a message containing the contents of the control words and the actual length of the data block is stored in the Acknowledge FIFO. Up to 15 different messages can be stored in this FIFO.

The occupancy of the Request and Acknowledge FIFOs is monitored in the status register of the interface and can be used to generate an interrupt. Up to six different events can prompt the interface to generate an interrupt.

The return lines and the other signals of the S-LINK destination card are set through the control register.

IV. Test Set-up

Hardware tests were made on an 800 MHz Pentium III PC with a SUPER 370DLE motherboard [9] running Linux. A SLIDAS board [10] plugged onto the S-LINK connector of the S32PCI64 interface generated input data while a VMEtro PBT-515BX PCI Bus Analyser [11] plugged into the second 64-bit/66 MHz PCI slot monitored an activity on the PCI bus. A PCI bus exerciser (PCI-Blaster) [12] benchmarked the memory and PCI bridge of the PC.

A. SLIDAS

The SLIDAS is a manually controlled, stand-alone data generator that can be connected to the S-LINK connector. It can generate a wide spectrum of data patterns of different length, contents and bandwidth.

B. PCI-Blaster

The PCI-Blaster is a module consisting of the S32PCI64 hardware and a dedicated firmware uploaded into its FPGA. It has been designed to sink or source a continuous data stream running at full PCI speed in order to test properties of the PC it is installed in. PCI-Blaster is fully software programmable through its PCI registers. Read and write modes are available and can be set-up simultaneously. No external devices are needed to run it. In PCI write mode, data of a known pattern is generated internally while running in PCI read mode any data is accepted. Data transfers can be set-up for a specified number of times or for an infinite loop.

V. Measurements

We measured the sustained throughput of one S32PCI64 interface and an aggregate sustained throughput of two S32PCI64 modules present on the same PCI. Application software, written for the test set-up, controlled the data transfers on a ‘one-by-one’ basis. This means that for each data packet transmitted to the host’s memory a new request has been submitted to the Request FIFO of the S32PCI64 interface. The test program polled the status register and didn’t use interrupts. A graph presenting the performance of the S32PCI64 interface is shown in Figure 2. One can see that the S-LINK bandwidth limits the overall performance of the system to 160 Mbytes/s or 320 Mbytes/s in the case of one or two S32PCI64 cards present on the PCI bus, respectively.

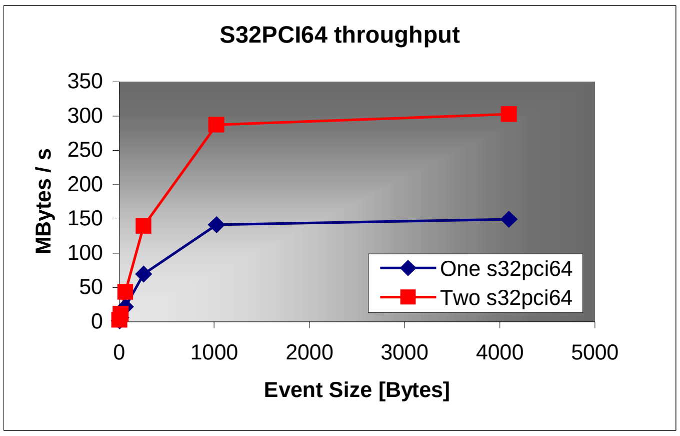

Figure 2. Performance of the S32PCI64

With the help of a PCI analyser, we measured the latency of single read and write instructions and the overhead caused by the interface. The overhead is understood as a number of PCI clock cycles (wait states) between addressing and data cycles of the single PCI instruction.

We measured also minimal possible period of consecutive commands of the same or different type. The asymmetry between the results obtained for read and write commands is a property of the PC being used for the tests.

The results of these measurements are shown in Tables 1 and 2.

Table 1: Overhead introduced by the S32PCI64

Type of instruction	Overhead
write	30 ns (2 wait states)
read	45 ns (3 wait states)

Table 2: Minimal interval between beginning of consecutive PCI cycles

Type of instruction	Interval
write-write	75 ns
write-read	105 ns
read-read	330 ns
read-write	345 ns

For each data block movement the S32PCI64 interface needs 5 PCI single transactions: 2 write operations to the Request FIFO and 3 read operations from the Acknowledge FIFO. The protocol overhead is reduced down to ~2.5 µs with respect to the 8 µs of the SSPCI.

VI. FILAR

With a theoretical performance limit of 528 Mbytes/s a 64-bit/66MHz PCI would support up to three S32PCI64 cards, each featuring a bandwidth of up to 160 Mbytes/s. As the number of this type of PCI slots in a PC is usually limited to two, an integration of more input links into one PCI interface would be advantageous. The FILAR interface (see Figure 3), which is under design, gives the possibility to receive data from up to four input links on a single PCI card.

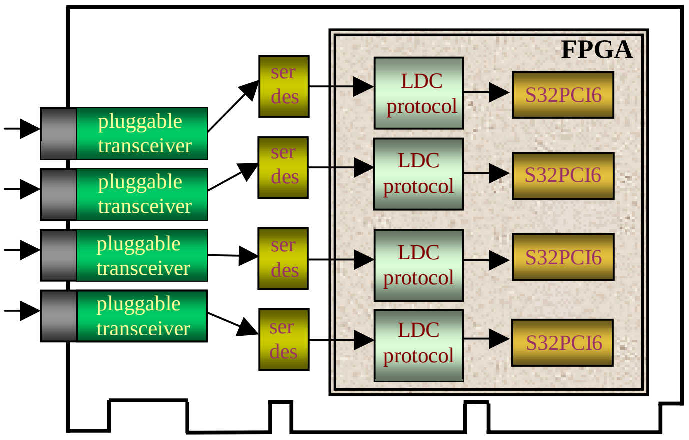

Figure 3. The FILAR module

Figure 3 presents block diagram of the FILAR module. It contains four 2.5 Gbit/s HOLA [13] S-LINK LDC channels integrated with four S32PCI64-like cores on a one PCI card. Each data channel can be individually enabled/disabled. The card will have pluggable fibre optical transceivers, which reduces the cost if modules with less than four data channels are needed. The temperature of the card can be monitored through a PCI register.

A prototype of the FILAR interface is scheduled for Q1/2003.

A. *FILAR Software*

The protocol of the FILAR is similar to that of the S32PCI64, with an additional reduction to at most three PCI transactions per packet. The dedicated software package, written in the context of the ReadOut Subsystem (ROS) [14], consists of a Linux driver and user library. Support for multiple FILAR cards and multiple channels in each card is provided. As the FILAR code has to coexist with other processes, the polling of the Acknowledge FIFO has been given up in favour of interrupts. In order to minimise the interrupt frequency, once in the interrupt service routine, the driver serves all channels including those that have not yet reached the interrupt threshold themselves. In combination with some other techniques to reduce the software overhead we have measured a total overhead (software + PCI single cycles) of 1.5 µs per S-LINK packet.

B. *FILAR Emulator*

To prepare the FILAR design and to test new software written for it we have built the FILAR emulator. The re-usability of existing resources helped as in the case of the PCI-Blaster. The FILAR emulator consists of the S32PCI64 hardware and a newly developed firmware package for the FPGA.

The FILAR emulator takes its data from the S-LINK connector and copies it inside the FPGA to the inputs of all four data channels. As the board contains only one XOFF line for flow control, the XOFF lines coming from the individual data channels are logically ORed inside the FPGA.

The emulator generates an interrupt when 24 or more events have been transmitted to the host's memory by anyone of the four data channels.

C. *Performance of the FILAR Emulator*

We measured the sustained throughput of the FILAR emulator with one, two, three and four data channels enabled. For the measurements we used the same test set-up as for the S32PCI64 with the SLIDAS data generator. Triggered by the interrupts from the FILAR emulator, the Linux driver read the Acknowledge FIFO and re-filled the Request FIFO in all working data channels. The test application in turn checked the data from the Acknowledge FIFO and provided new data to the Request FIFO. The VMEtro PCI analyser was used for bus traffic analysis. For randomly recorded full traces of 65000 PCI cycles a statistical function of VMEtro has been used to compute the throughput. Results include all protocol and software overheads. A graph showing the performance of the FILAR emulator is given in Figure 4.

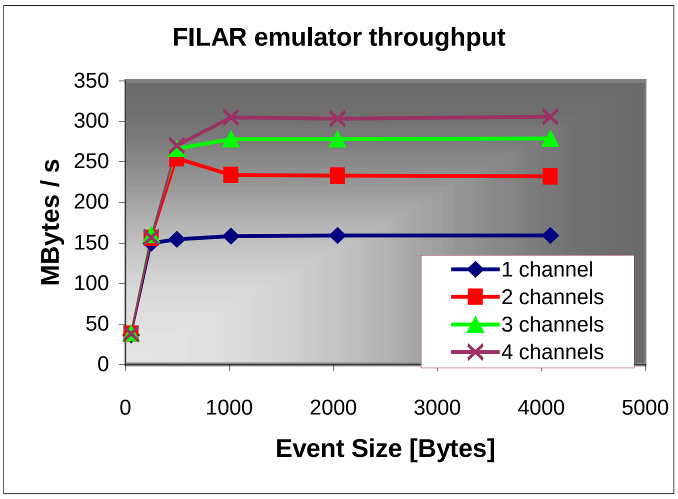

Figure 4. Performance of the FILAR emulator

We see an improvement in the performance of the FILAR emulator with the respect to the S32PCI64 interface for small packets. Also the bandwidth for one data channel is better than that in the S32PCI64.

The performance of the FILAR emulator running two, three or four data channels is compromised by a limitation of the S32PCI64 hardware. A flow control signal working for all channels here stops new data to come to the whole interface whenever any data buffer in one of the data channels is getting full. It thereby prevents other, already empty data buffers from being re-filled. This has the consequence that, occasionally, the emulator is entirely empty for up to 4.5 µs during a DMA. This shortcoming limits the overall performance of the FILAR emulator currently to 320 Mbytes/s but will not exist in the final design where the data flow in each channel will be individually controlled.

We investigate an option that further reduces an impact of single PCI cycles on overall bandwidth of the module. If it becomes necessary, we will try to replace single PCI accesses by DMA.

VII. Conclusions

In this paper we have presented the evolution of the S-LINK to PCI interfaces. The development in this area followed trends in modern electronics over past years. High-density field programmable gate array chips have replaced traditional integrated circuits while more intellectual property solutions have been used. A smooth transition from the S32PCI64 to the FILAR interface has made it possible to re-use a great part the software.

All these factors simplify the design process and improve testability and re-usability. Integration of several data channels on one board reduces the cost of the final system, which is the key issue in modern experiments.

VIII. Acknowledgments

The work was supported in part by the Polish State Committee for Research under Grant No. 620/E-77/SPUB-M/CERN/P-03/DZ 295/2000-2002.

This work was performed within the framework of the ATLAS Trigger and Data Acquisition project and we gratefully acknowledge the contributions of other members of the T/DAQ group.

IX. References

[1] http://www.cern.ch/HSI/s-link
[2] PCI local bus specification, Revision 2.1, June 1, 1995
[3] http://hsi.web.cern.ch/HSI/s-link/devices/slink-pmc/
[4] http://hsi.web.cern.ch/HSI/s-link/devices/s32pci64/
[5] http://edms.cern.ch/document/337904/1
[6] http://www.amcc.com
[7] http://www.no-el.krakow.pl/products.html
[8] http://www.plda.com
[9] http://www.supermicro.com
[10] http://hsi.web.cern.ch/HSI/s-link/devices/slidas
[11] http://www.vmetro.com
[12] http://edms.cern.ch/document/338516/1
[13] http://hsi.web.cern.ch/HSI/s-link/devices/hola/
[14] http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/ROS/ros.htm