SRC Technical Note
1998 - 012

June 1st, 1998

FireLink: A Reconfigurable Firewire Testbed

Oskar Mencer, Mark Shand, Michael J. Flynn

digital

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

http://www.research.digital.com/SRC/

Abstract

We describe the design and implementation of a reconfigurable high-speed interface for IEEE 1394 Firewire and measurements obtained with this system. For the physical and link layers we use a commercial chip-set. Field-Programmable Gate Arrays on the PCI Pamette and a secondary FireLink board are used to implement a simple transaction layer interfacing the link layers to the host CPU.

1 Introduction

Firewire, IEEE standard 1394, is a serial interconnect developed by Apple and Texas Instruments for the consumer market. The standard regulates the physical and link layers. In addition, the standard proposes a generic transaction layer. Primary target applications are high volume Audio, Video and computer peripheral connections.

Firewire highlights are: 100 Mbits/s to 3.2 Gbits/s of physical transport[1], user friendly hot plugging, low cost and a non-proprietary specification. Figure 1 puts Firewire into perspective to other interconnect technologies.

Today, the bandwidth of Firewire is five times smaller than current PCI bus bandwidth. Comparing the projections of the various interconnect technologies, Firewire has the potential to achieve about half the bandwidth of the High-Performance Parallel Interface (HIPPI).

Firewire is of special interest because it represents the first serious attempt to provide a standardized high bandwidth interconnect for both computers and consumer electronics. Consumer devices such as all digital video cameras are already available and Microsoft Corp. is promoting the use of Firewire in future PCs.

2 FireLink and the PCI Pamette

PCI Pamette is an FPGA board developed at DIGITAL Systems Research Center as a follow on project to PAM [1] - Programmable Active Memories - one of the first successful custom computing machines.

The PCI Pamette contains a powerful PCI interface, four Field Programmable Gate Arrays (Xilinx XC4000 series), and IEEE P1386 Common Mezzanine Card connectors to the real world.

Past projects with the PCI Pamette include :

- PCI bus exercising and analysis[7].

[1] at the time of this project, 200 Mbits/s chipsets are available from Texas Instruments.

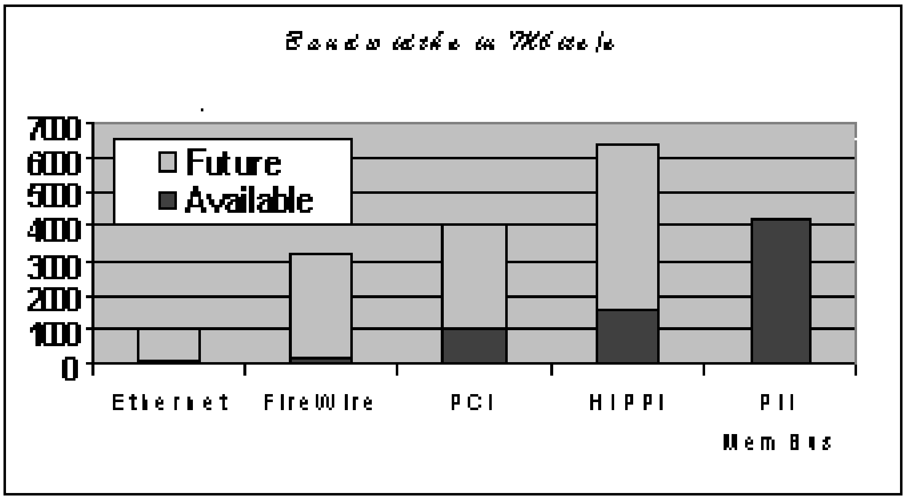

Figure 1: In this figure, various network technologies are compared to Firewire. Currently available bandwidths and future bandwidths are compared. The technologies are: Ethernet, Firewire, PCI bus, HIPPI the High Performance Parallel Interface, and the Pentium II main memory bus.

- Scientific image acquisition and telescope control for solar astronomy [8] [9] [10].
- Rapid prototyping of a wireless LAN host adapter[11].
- Lightning: A distributed graphics frame-buffer[6].

To provide a direct Firewire interface to PCI Pamette, Didier Roncin developed a custom PCI Pamette daughter board called FireLink. FireLink consists of two fully independent Firewire Channels and 1 Xilinx XC4010E for control. FireLink can therefore also be used as a bridge between two Firewire networks. Both channels are three port however due to physical space constraints one channel only provides connectors for two of its ports. The XC4010E on the FireLink board is programmed from PCI Pamette using a jtag interface. This same interface can be used to examine the internal state of the running FPGA. Strict compliance with the Firewire standard and flexible FPGA technology make this board a powerful platform for exploring Firewire.

2.1 FPGA design with PamDC

PamDC was developed as part of the PAM project[1] described above. The design is described structurally in C++. Running the resulting program creates a Xilinx netlist file which is passed on to place-and-route tools.

The advantage of PamDC is that the designer has full control over placement. Technology mapping can be done automatically or by the designer. This is especially important in order to make efficient use of the fast carry chain, available in Xilinx XC4000 FPGAs. PamDC gives the designer total control over the design, which with some effort can result in maximal performance and minimal area.

The drawbacks of PamDC are the relatively high effort needed to create structural designs on a very low level.

2.2 Reconfigurable Hardware based Development Methodology

Reconfigurable hardware allows us to use an incremental design approach. We use instrumentation to aid in debugging and optimization of the design. At runtime, SRAM memory on the PCI Pamette board is used to log data from the FPGAs. This data can afterwards be evaluated. Results of cycle-by-cycle simulation are used to validate the real-time data obtained by instrumentation. This methodology was successfully employed in debugging five FPGAs across three circuit-boards.

Our methodology also enables us to explore various design alternatives and obtain measurements about the environment of our design (e.g. details about the link and physical layer chip-set.)

3 Inter-chip Communication

3.1 Communication with the Link Layer Chips

Firewire transactions are initiated by writing to specific registers on the link layer chips, i.e. sending a message requires writing the control data from the message header into control registers, writing the first data word into a start register and writing all following data words into a second write register until fin ally the last data word of the message is written into a third 'end-of-block' register.

After the last word is written to the link layer, the data is transferred to the physical layer. After all the data is transferred to the physical layer, the link layer chip is ready to receive new data.

The latency for reads and writes of registers in the link layer is defined in the documentation to take "up to 9 clock cycles". This variable latency of reads and

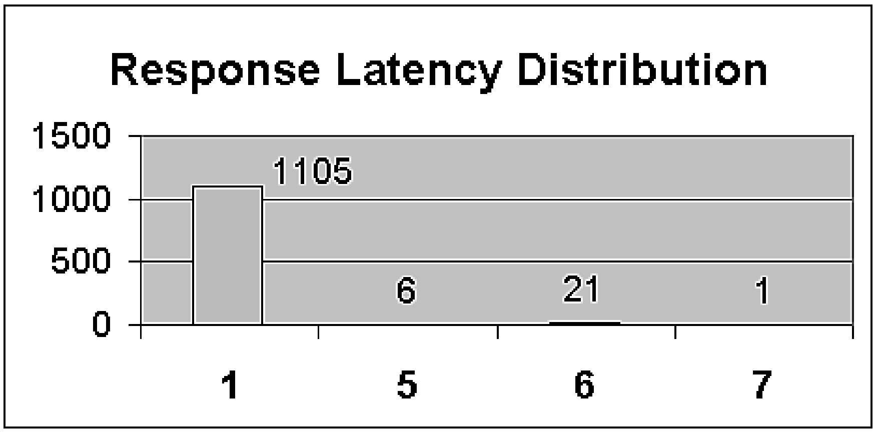

Figure 2: The figure shows the distribution for the latency of accessing the link layer chips in link layer clock cycles. Out of 1133 consecutive requests to the link layer, 1105 responses came on the following clock cycle.

writes of the link layer requires adequate buffering and synchronization between the PCI Pamette and the link layers. A block diagram of our buffer design on the FPGAs is shown in Figure 3.

In order to gain better understanding of the meaning of “up to 9 clock cycles”, we used our instrumentation and logging framework to get data about the actual distribution of read and write latencies. Figure 2 shows the distribution of link layer access latencies.

3.2 PCI and PCI Pamette Fundamentals

PCI is a high performance, 32-bit or 64-bit bus with multiplexed address and data lines. The bus is intended for use as an interconnect mechanism between highly integrated peripheral controller components, peripheral add-in boards, and processor/memory systems.

The fundamental transaction in PCI is an address followed by a burst of data. Starting with version 2.1 of the PCI Specification[12] there is some scope for splitting transactions into address and data phases, but this feature is primarily intended for use in the construction of deeply hierarchical bridged bus systems.

In most uses the read operation suffers a performance penalty in comparison to the write operation because the read target must decode the transaction address and, on the basis of this address, drive the appropriate data onto the bus. This address dependency introduces latency between the address and data phases on

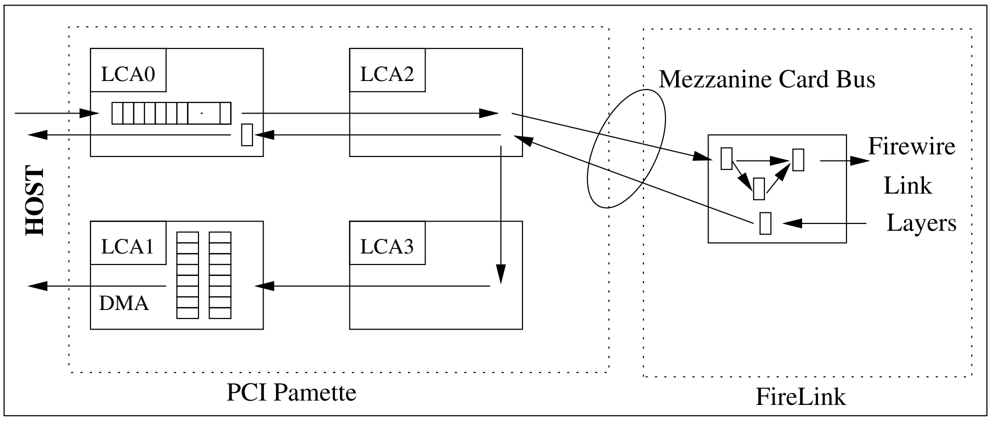

Figure 3: The figure shows five Xilinx XC4010E FPGAs. LCA0-3 are the four user programmable FPGAs on the PCI Pamette. The FireLink board has only one FPGA that communicates with both Firewire Channels. The buffers in LCA0 are FIFO buffers, while in LCA1, local SRAM is used to buffer data for DMA to host memory.

read transactions. Write operations in contrast suffer no such penalty because the initiator of the write can prepare the address and data ahead of the transaction and pipeline information onto the bus.

For these reasons we try wherever possible to use PCI write transactions for performance critical data transfers. Data that the host CPU wishes to send to a remote system is sent to the PCI Pamette using programmed I/O write operations. In this case we take advantage of the CPU write buffers by writing to sequential addresses in the memory space to which the PCI Pamette has been mapped and relying on the write buffers to aggregate these transfers into longer bursts. Data arriving from a remote system on the other hand is moved into the host memory system using direct memory access (DMA) in which the PCI Pamette autonomously requests the PCI bus and initiates writes to main memory buffer addresses that have been provided in advance by the CPU. Rendezvous with the CPU for received data is currently accomplished by polling of main memory by the CPU.

Data flows in two directions across the PCI bus. The reason for using CPU writes in one direction and DMA in the other is that CPU writes to PCI are the most efficient way to transfer data from CPU to PCI and DMA initiated by a peripheral card (i.e. PCI Pamette) is the most efficient way to move data from PCI to main memory of the host.

```
Send (sendBlock, length, destination, A);

if (ReadAck(A) != SUCCESS) { ERROR }

if (Recv(B, recvBlock, length) != SUCCESS) { ERROR }
```

Figure 4: The figure shows a typical sequence of function calls for sending a block of data on channel A and receiving a block of data on channel B.

3.3 Communication between PCI and the Link Layers

We use PCI Pamette in a high-performance mode where main memory can be mapped onto resources on the FPGA board.

In order to achieve maximal performance, we transfer data to FireLink with write bursts from the CPU, and move the arriving data from FireLink into the host's main memory with DMA bursts initiated by the PCI Pamette.

CPU write bursts on the PCI bus are achieved by mapping our entire region of main memory onto one register on the link layer chip. Consecutive writes to this region create PCI bursts that allow us to write to the link layer chips at maximal speed (see section on Peak Bandwidths and Latencies).

In our case, DMA cycles are initiated by the PCI Pamette transferring a block of data from LCA1 to main memory. For efficiency reasons, we chose a block-size equal to the host cache-line size.

4 The FireLink Software Library

In order to control the Firewire network through the PCI Pamette, we created a software library that communicates with Firewire through the PCI Pamette runtime environment. The software interface to FireLink is also designed with performance as the main target. We chose the C programming language and macros for implementing the software interface to the FireLink hardware. The size of the library is around 1000 lines of code.

The two data abstractions are:

- Quad-Packet [32 bits]
- Burst-Blocks, consisting of multiple Quad-Packets

Figure 4 shows a sequence of *send* and *receive* function calls for Burst-Blocks. What actually happens is that the CPU sends the header for a Burst-Block and afterwards the data, to the asynchronous send FIFO in the link layer of channel A. Next, the link layer sends the data to the physical layer. The physical layer signals the link layer when the acknowledge from the other side arrives. *ReadAck* waits for this signal to arrive in main memory.

Our design support two ways to receive data. In normal operation mode, DMA is enabled. The FPGA on the FireLink board automatically reads the arriving data and transfers it to LCA1 where the data is stored in local SRAM memory until the PCI arbiter grants a DMA cycle to the PCI Pamette.

In development and debugging mode DMA is disabled. The CPU has to initiate a PCI read in order for the data to be read out of the link layer and get transferred to LCA0. Although it is much slower to read out received data on a block by block basis, this mode has proven to be effective for debugging and testing purposes.

5 Peak Bandwidths and Latencies

We measure peak bandwidths and latencies achievable with the software library described above.

Figure 5 shows the resulting latencies for the three operations:

1. *Send*: write-bursts from CPU to Firewire Link Layer
2. *ReadAck*: wait for physical acknowledge
3. *Recv*: DMA from Link Layer to main memory of the CPU

Send, the time to write a burst-block of a specific size into the transmit FIFO, has a maximal bandwidth of 70 Mbits/s for block sizes of 240 quad-packets. This is achieved by mapping a region in main memory onto the transmit FIFO in the link layers, and forcing the CPU to issue write bursts on the PCI bus.

ReadAck, the time for the link layer to send all the data from the transmit FIFOs to the destination and receive an acknowledgment from the other side, translates into a bandwidth of 140 Mbits/s. According to the documentation from Texas Instruments, the physical layer transmits data at 200 Mbits/s. Therefore about 60 Mbits/s are lost waiting for a physical acknowledge.

Recv, the time to move the data from the receive FIFO at the link layer chips to main memory, results in a bandwidth of 110 Mbits/s. This is achieved by DMA bursts from the PCI Pamette board to main memory.

The assembly instruction *rpcc* was used to instrument the code. This allowed us to measure latencies with the granularity of processor clock cycles. All data reflects latencies for the runtime library calls and macros.

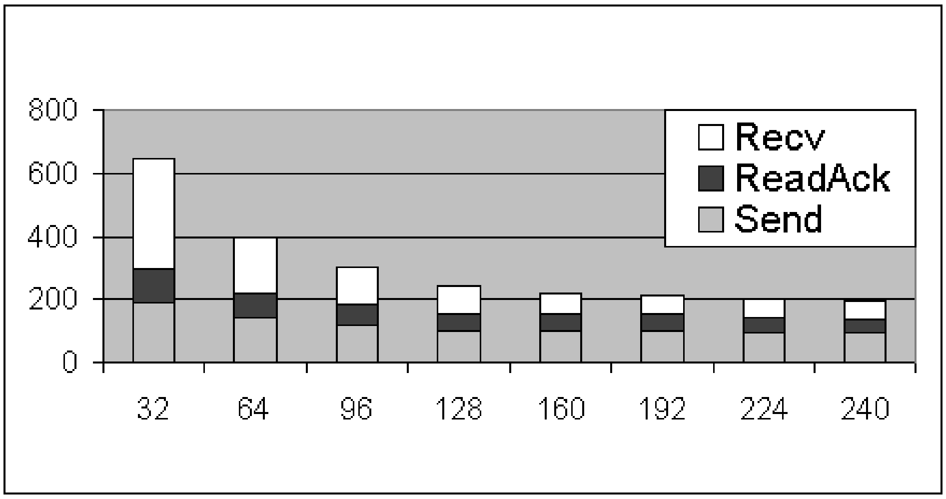

Figure 5: The figure shows the latencies for each of the three basic operations, which are provided by the software library. The different columns stand for different burst-block sizes.

6 Future Work

We used the experience gained in the project presented above to design PAM-Blox [5], an object-oriented library and design methodology on top of PamDC. The major goal of PAM-Blox is to create additional layers of abstraction on top of PamDC – simplifying FPGA design while keeping the performance over area advantage of low level FPGA design.

Possible extension to the FireLink project include exploring the scalability of FireLink to handle large systems. A software interface to Memory Channel technology would integrate all the Memory Channel applications with Firewire.

A more esoteric project would be to think of the FireLink hardware as a cost-efficient distributed system of programmable active memories (PAMs). The objective could be to speed up computation intensive parallel applications by accelerating each node with custom FPGA designs.

7 Acknowledgments

We would like to thank Didier Roncin for building the FireLink board. We especially thank DIGITAL Systems Research Center in Palo Alto for sponsoring the development of this hardware and providing one of the authors with a summer-

internship to explore the design space of a reconfigurable transaction layer for Firewire.

This research is also partly supported by DARPA Grant Nr. DABT63-96-C-0106.

References

[1] P. Bertin, D. Roncin, J. Vuillemin, *Programmable Active Memories: A Performance Assessment*, ACM FPGA, February 1992.

[2] W.H. Mangione-Smith, B. Hutchings, D. Andrews, A. DeHon, C. Ebeling, R. Hartenstein, O. Mencer, J. Morris, K. Palem, V. Prasanna, H. Spaanenburg, *Seeking Solutions in Configurable Computing*, IEEE Computer Magazine, December 1997.

[3] The PCI Pamette FPGA board at DEC Systems Research Center, *http://www.research.digital.com/SRC/pamette/*

[4] The homepage of the FireLink board at the PAM website, *http://pam.devinci.fr/hardware.html*

[5] The homepage of PAM-Blox, object-oriented FPGA design. *http://umunhum.stanford.edu/PamBlox/*

[6] M. Eldridge, J. Owens, K. Proudfoot, P. Hanrahan, A. Gupta, K. Li, "Lightning", *http://www-graphics.stanford.edu/projects/flashg/lightning/*

[7] L. Moll and M. Shand, "Systems Performance Measurement on PCI Pamette," *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1997.

[8] M. Shand, "Flexible image acquisition using reconfigurable hardware," in *FPGAs for Custom Computing Machines (FCCM'95)*. IEEE, April 1995,

[9] M. Shand, W. Wei, and G. Scharmer, "A 3.8ms latency correlation tracker for active mirror control based on a reconfigurable interface to a standard workstation," in *Photonic East Symposium '95*. SPIE, October 1995.

[10] J. Sanchez Almeida, M. Collados, V. Martinez Pillet, V. Gonzalez Escalera, G.B. Scharmer, M.Shand, L. Moll, E. Joven, A. Cruz, J.J. Diaz, L.F. Rodriguez, J. Fuentes, L. Jochum, E. Paez, B. Ronquillo, J.M. Carranza and T. Escudero Sanz, "The IAC Solar Polarimeters: Goals and

Review of Two Ongoing Projects," *1st Advances in Solar Physics Euroconference*, eds. Schmieder B., del Toro Iniesta J.C., and Vazquez M., ASP Conf. Ser., 1997.

[11] Tom McDermott, Phil Ryan, Mark Shand, David Skellern, Terry Percival, and Neil Weste. "A wireless LAN demodulator in a pamette: Design and experience." in *FPGAs for Custom Computing Machines (FCCM'97)*. IEEE, April 1997.

[12] *PCI Local Bus Specification 2.1,* PCI Special Interest Group, 1995.