PCI Pamette
Generic PCI board
Mark Shand
shand@pa.dec.com

digital

Systems Research Center
The PAM project, standing for programmable active memories, originated at INRIA, the French national computer research institute, in 1987, under the leadership of Jean Vuillemin. Shortly after its inception the project moved to Digital’s Paris Research Laboratory where it stayed until mid-1994 when the lab was closed due to budget cuts at Digital.

The goal of the PAM project was to study the construction, programming and use of FPGA-based reconfigurable coprocessors. The goal of these coprocessors, attached to the bus of a host workstation, was to deliver an order of magnitude or greater performance improvement across a range of applications. Two compute-oriented reconfigurable computing engines were built, Perle-0 in 1988 and DECPeRLe-1 in 1991-1992.

Experience in developing the TURBOchannel based interface to DECPeRLe-1, also based on FPGAs, lead us to investigate smaller I/O oriented PAMs. For these we coined the name Pamette. Prior to PCI Pamette V1, two were built, TURBOchannel Pamette, also known as 4Mint, and PCI Pamette V0. TURBOchannel pamette was a 2x2 matrix of Xilinx 4010s attached directly to Digital’s TURBOchannel bus. PCI Pamette V0 was a prototype board which allowed us to investigate the implementation of PCI interfaces in Xilinx 4000 series FPGAs.

### Background

- PAM Project - Paris Research Laboratory
- **PAM** - Programmable Active Memory
  - FPGA-based reconfigurable coprocessors
- Reconfigurable Computing Engines
  - Perle-0
  - DECPeRLe-1
- Modest I/O oriented coprocessors
  - TURBOchannel Pamette (4Mint)
  - PCI Pamette V0
DECPeRLe-1 Review

- Vintage 1991-1992
- 16+7 XC3090-100
- 200k gates @ 25MHz
- 4 MB fast static RAM (400MB/s)
- 100 MB/s TURBOchannel host link
  - http://pam.devinci.fr/

For people unfamiliar with DECPeRLe-1, this slide recalls some of the key points. The board was prototyped in 1991 and 20 systems were produced in 1992. About half these were sold (at production cost) to collaborating universities and research centres around the world.

DECPeRLe-1 consisted of a core 4x4 computational matrix of Xilinx XC3090-100 FPGAs augmented by 7 peripheral FPGAs for RAM address generation, switching and marshalling data in the host interface. The FPGAs resources provided the equivalent of 200k gates which could be clocked at up to 25MHz for typical circuits. The board contained four 1MB banks of static RAM each capable of 100MB/s sustained data transfer. The host link interfaced to TURBOchannel and, through DMA provided burst transfer rates of 100MB/s and sustained rates of 80-90MB/s.

A number of compute-intensive applications have been implemented on DECPeRLe-1. The Laplace heat equation solver holds the record for the highest raw compute power -- 6 billion additions per second for a total equivalent compute power of 39 Gigainstructions per second. The RSA decryption implemented in 1992 is still a factor of six faster than the fastest microprocessor in 1996. DECPeRLe-1 was successfully applied to several high energy physics triggering problems at CERN in Geneva. Here the problems were distinguished not only by a high compute requirement, but data inputs rates in the range 100-200MB/s.

Most of the publications relating to DECPeRLe-1 are available on the worldwide web at two sites, one in France and one in California.
PCI Pamette is a modest beast compared to DECPeRLe-1. Its lineage can be traced back to the DECPeRLe-1 interface board, known internally as 3Mint, which spawned the TURBOchannel Pamette, also known as 4Mint.

The last few years of microprocessor development have made it increasingly difficult to build FPGA coprocessors that are closely coupled to CPUs. A typical modern microprocessor has several GB/s of bandwidth to its register file and on-chip cache, but to connect to these requires reimplementing the CPU chip and can only be done working with the CPU designers. Off-chip cache and main memory often have bandwidths of several hundred MB/s, but usually use proprietary protocols which change from one machine design to the next.

By the time we reach standard I/O buses like PCI the available bandwidth has dropped to 100 MB/s. Even in high end 64-bit PCI systems the most we can obtain is 250 MB/s, over an order of magnitude less than the bandwidth to the first level of the memory hierarchy. The same story applies to latency of transfers.

PCI Pamette increases the efficiency of constrained standard I/O buses by allowing data specific preprocessing and reformatting and optimized data movement without loading the host CPU.
A modest off-chip FPGA resource can perform application specific data reformatting and filtering, for instance, inserting guard bits so that byte-wide data can be packed into 64-bit words and added byte-parallel without fear of overflow into neighbouring fields.

Real-time front-ends can be implemented in the FPGA that can respond at speeds unattainable by software and can buffer data so that the requirements on software response time are greatly relaxed.

The Pamette and its CAD and Runtime support software allow rapid prototyping.

If the anticipated volume is low the Pamette plus application specific bitstream can be the final product.

The Pamette is fully reconfigurable, even the PCI interface. This opens the possibility for logic-analyzer type functions, but also for generating traffic that can test, and stress, the host PCI and memory system.
The PCI Pamette V1 has footprints for 5 Xilinx 4000 series FPGAs in PQ208 packages. Primarily we use the 4010E, but we could use 4013E, 4028EX or 4003H parts. One FPGA serves as PCI interface. This is more or less dictated by the PCI specification which only allows a 10pF load per signal (one pad). The other four FPGAs are organized in a simple two by two matrix. For a board with so few FPGAs, a dedicated programmable switch is overkill.

The PCI interface is 5 volt 64-bit. Of course the majority of hosts are only 32-bit and in these systems the board, following the PCI specification, functions as a 32-bit board. We use a 4010E for the interface FPGA because it is the smallest 4000 series device that can connect to a 32-bit bus on one side and still have some rows spare. We consider the internal clock distribution on larger devices like the 4013E to introduce too much delay for PCI. The interface FPGA is loaded at power-on from a serial ROM. The host may then load the matrix, or user, FPGAs via the interface FPGA. However the interface FPGA is itself reconfigurable, we’ll talk more of this in a couple of slides.

The board has two banks of 16-bit wide by 64k SRAM, and connectors for industry standard 72-pin SIMM DRAM modules which permit from 4MB to 256MB of DRAM to be attached.

Any I/O oriented board can not anticipate all the types of transducers that might be required for external connections. Instead, we rely on simple mezzanine format daughter boards. Rather than invent our own format, we use IEEE P1386, the CMC/PMC format.

The clock generation scheme provides copies of the PCI clock plus an arbitrary frequency user specified clock.
This slide shows how the various components described in the previous slide are interconnected.

Essentially the front two user FPGAs connect to private scratchpad SRAMs and funnel data from the rear two user FPGAs to the interface FPGA. The rear two user FPGAs connect to the daughter board connectors and the DRAM SIMM sockets.

The interface FPGA controls download and readback of the user area and generation of clocks. User FPGAs can be individually reconfigured without affecting other FPGAs, however when multiple FPGAs are reconfigured or their configurations are read, this is done in parallel.
This slide gives details on the widths of the various buses on PCI Pamette V1. Note the narrow width of the bus between the interface FPGA and the front two user FPGAs relative to the 64-bit PCI. We would always like to have more pins in multi-FPGA systems. However in this case we can run this link at 66MHz and thus match the bandwidth of the 64-bit PCI.

Likewise the path between the rear two user FPGAs is narrow compared to the other buses in the user matrix. This width was dictated by the number of pins remaining after the pins required to connect to the mezzanine card and DRAM.
One challenge in implementing a PCI interface is the clock. Only one load is permitted on the clock input, and, because the period may change by arbitrary amounts on any cycle, any Phased-Locked Loop (PLL) based solution cannot claim compliance. This is unfortunate because the slowness of the internal FPGA clock distribution creates a strong temptation to precompensate the shift with a PLL. Of course we know that on almost any system we care about running on the PCI clock will be rock-solid.

The solution adopted on PCI Pamette V1 is to take the PCI clock directly into the interface FPGA on a primary clock input pin and send a copy of it is sent out through a neighbouring I/O pin to a PLL. The feedback path of the PLL also passes into and out of the FPGA thus matching the delay on the reference clock. Consequently, on systems with stable PCI clocks the PLL provides zero-skew copies of the PCI clock. These copies are connected to the user FPGAs. If the PCI clock is not stable, a different source, for instance the FPGA internal oscillator, can be fed to the PLL. In either case the interface FPGA runs directly off the PCI clock. An additional feature of this scheme is that the reference clock or the feedback may be divided down in the output pads to create zero-skew clocks that are double or half the PCI clock.

A second copy of the PCI clock is used as reference input to a programmable clock generator. This device can produce any frequency from 400kHz to 100MHz in steps of approximately half a percent. By passing this clock though the interface FPGA before the buffer which distributes it to the user FPGAs we gain the ability to stop it, or issue bursts.
This slide covers some specific capabilities of PCI Pamette V1.

The interface FPGA is normally loaded from serial ROM at power-up time. However an application in the user FPGAs may reconfigure the interface from a configuration stored in SRAM, or may change the configuration in the serial ROM so that this new configuration is used at each subsequent power-up.

The ability to reconfigure the interface in application specific ways makes the board very useful in system measurement applications for instance, where the interface could be adapted to capture every bus transaction and log them to local DRAM. It also opens the way for innovative uses of the PCI bus among cooperating boards.

Our mezzanine board facility uses the IEEE P1386 specification for Common Mezzanine Cards (CMC). This is a layered specification. At the base level I/Os, power and ground are defined. At the next level two bus protocols are offered, PCI and SBUS, with scope for others to be added in the future.

Custom designed I/O boards for PCI Pamette V1 can use just the CMC level and use a simple application tailored protocol. Alternatively, by appropriate programming of the user FPGAs, commercially available PMC (PCI Mezzanine Card) or SMC (SBUS Mezzanine Cards) may be used.

Capabilities

- Reconfiguration of interface
  - Bus snooping (compress & log to DRAM)
  - PCI protocol innovations
- Mezzanine board
  - IEEE P1386 CMC/PMC/SMC
  - PMC - uses PCI protocol
  - CMC - uses application specific protocol
This slide is above all a disclaimer. PCI Pamette V1 comes with FPGA circuit generation system derived from the DECPeRLe-1 CAD system. In its current incarnation this is known as PamDC. Members of the PAM project and many of our collaborators have found this CAD package an efficient way to generate high performance FPGA designs, but it is by no means the only way to generate designs for PCI Pamette V1. If you are happy with your existing CAD flow, be it schematics or some hardware description language like Verilog or VHDL, you may use it to generate designs for this board.

PamDC is not a compiler from C++ into circuits. That is a hard problem that we don’t know how to solve in an efficient way. PamDC is a library of C++ routines that, linked with your code, allow you to generate Xilinx xnf files based on declarations of circuit elements and nets embedded in the execution of your code. We have demonstrated its use on dense, high-performance designs for Xilinx 3000 and 4000 series devices, but it does require a good understanding of the underlying FPGA technology and often requires embedding considerable amounts of placement information with the user’s logic declarations.

### Programming Tools

- **Code-based - PamDC**
  - C++ based circuit description library
  - Full control of placement and technology mapping
  - Good for datapaths, code reuse, simple changes
- **Schematics**
- **HDL**
Digital wants to be as open as possible with PCI Pamette and its supporting technology, while protecting its investments. We encourage you to review our non-disclosure agreement on the web and if you find it acceptable download the PCI Pamette documentation and software. For research and academic use the PCB database and software is available on an “as is” royalty free basis. For commercial terms please contact us using the information in the web page.

PCI Pamette V1 is currently supported under Digital UNIX on Alpha or Microsoft Windows/NT on Intel x86.