EMBEDDED NETWORKED FRONT ENDS - BEYOND THE CRATE*

Lawrence R. Doolittle, LBNL, Berkeley, CA 94720, USA

Abstract

The inexorable march of Moore's Law has given engineers the capability to produce front end equipment with capabilities and complexity unimaginable only a few years ago. The traditional standardized crate, populated with off-the-shelf general-purpose cards, is ill suited to the next level of integration and miniaturization. We have reached the stage where the network protocol engine and digital signal processing can, and should, directly adjoin the analog/digital converters and the hardware that they monitor and control.

The current generation of Field Programmable Gate Arrays (FPGAs) is an enabling technology, providing flexible and customizable hard-real-time interfacing at the downloadable firmware level, instead of the connector level. By moving in the direction of a system-on-a-chip, improvements are seen in parts counts, reliability, power dissipation, and latency.

This paper will discuss the current state-of-the-art in embedded, networked front end controllers, and gauge the direction of and prospects for future development.

THE CRATE AGE

For decades, CAMAC[1] and VME[2] crates have formed the basis for new designs of accelerator front end equipment. These designs still make a certain amount of sense when 100% of the desired functionality can be assembled using off-the-shelf boards.

Crates have their origin in the times when no single board had, or could have, enough interface gear to run a piece of equipment. It was reasonable to line cards up in a crate to get enough digital inputs, digital outputs, analog inputs, and analog outputs to meet the needs of a system.

As a natural consequence of Moore's Law[3], the amount of functionality available on a board has risen progressively. Last year's system fits in today's crate, last year's crate fits on today's circuit board, and last year's circuit board fits on today's chip.

A side effect of this progression is that, for the fixed form factor of a crate, the number of connection points to a board is larger, so more wires are involved. To justify having the large cost overhead of a crate, people are motivated to "fill it up." This in turn leads to unmaintainably large cable bundles leading between each crate and patch panels that act as antennas for crosstalk. The cables and patch panels are inevitably hand-wired and not self-checkable. Worse, from a software perspective, is that unrelated systems are often grouped in one control computer, aggravating problems of coordination.

NETWORKED FRONT END CONCEPT

It's always true that developing circuit boards (including assembly, debugging, and calibration) is more expensive than buying a ready-made board. Hand-wired transition assemblies between the connectors as provided on ready-made boards and those on the equipment that needs interfacing, however, is even more expensive and notoriously unreliable. People therefore put such transition and signal conditioning equipment on circuit boards. From there, a slippery slope begins: the extra effort to add Analog/Digital conversion chips to the board is fairly small, and places complete control over the analog system performance in the hands of the engineer. The resulting large, and difficult to test, number of wires between conversion chips and the control system logic can be managed by connecting them directly to an FPGA. Finally, computer gear is sufficiently small and well understood that it makes more sense to think of the computer as an add-on to the custom hardware, than *vice versa*. Each integration step reduces the number of connectors, a perennial weak link in accelerator reliability. It also reduces the number of unrelated clock domains.

The familiar block diagram of figure 1 represents in most general form the resulting structure of modern control hardware. The FPGA provides a consistent (and small) latency digital feedback path between the ADCs (analog to digital converters) and DACs (digital to analog converters). Different applications have varying requirements for the speed, resolution, and channel count of ADC and DAC hardware.

While analog electronics has not shrunk as dramatically as digital, it has proved possible in many cases to simplify the analog signal path by pushing functionality into the digital domain[4]. This is an important step in bringing down the total hardware complexity, since the digital processing involves no additional chips.

Low speed housekeeping hardware normally involves at least a multi-channel ADC for power supply monitoring (including the current drawn by the FPGA core), plus temperature and electronic serial number. Communication between the FPGA and such housekeeping hardware normally takes place over bit-serial interfaces such as SPI^{TM}[5], I^{2}C^{TM}[6], or 1-Wire^{TM}[7]. While some might

* This work is supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. The SNS project is being carried out by a collaboration of six US Laboratories: Argonne National Laboratory (ANL), Brookhaven National Laboratory (BNL), Thomas Jefferson National Accelerator Facility (TJNAF), Los Alamos National Laboratory (LANL), E. O. Lawrence Berkeley National Laboratory (LBNL), and Oak Ridge National Laboratory (ORNL). SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy.

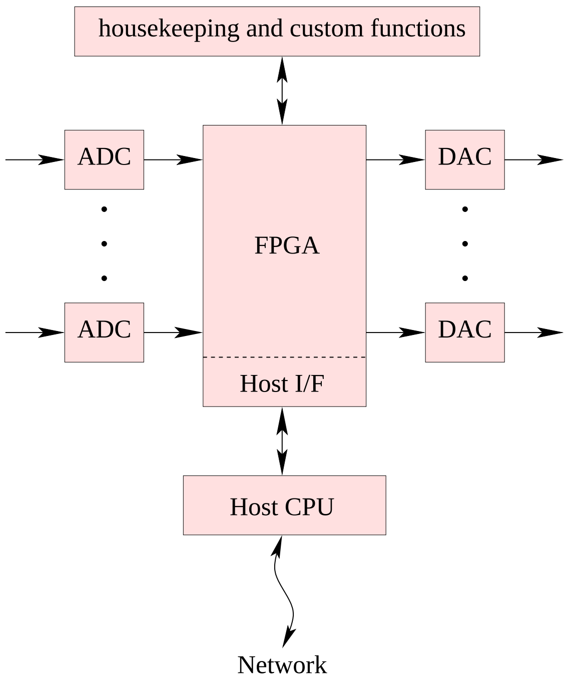

Figure 1: Familiar block diagram.

consider such housekeeping a frill, it has great value in operating, remotely troubleshooting, and maintaining these devices. The parts cost and board area required are minimal.

It is worth remembering why designs normally include both a CPU (central processing unit) and an FPGA. While both offer programmable functionality, each has strengths and weaknesses:

Computers

- multiple sources, very competitive market
- usually needs additional glue logic
- usually good throughput, but unpredictable latency
- widely understood, everybody thinks they know how to program one

FPGA

- chip design strongly protected by patents, two vendors dominate
- handles glue logic very well
- guaranteed latency normally designed-in
- reputation for being difficult to program

NETWORKED FRONT END EXAMPLES

The following examples show single-purpose devices that place information from, or control of, a device onto the Internet. They each represent a variation on the theme shown in figure 1.

Accelerator LLRF control

The SNS Interim Low Level RF system[9] is made up of connectorized RF plumbing, a custom circuit board with 4 × 40MS/s 12-bit ADCs, a 12-bit 80 MS/s DAC, a Xilinx XC2S150 FPGA, and a plug-on 200 MIPS single board computer. The assembly is 2U high in a 19” rack, and runs an EPICS server to provide network control over 100 MB/s Ethernet. All the power supplies are linear, to minimize electrical noise in the chassis.

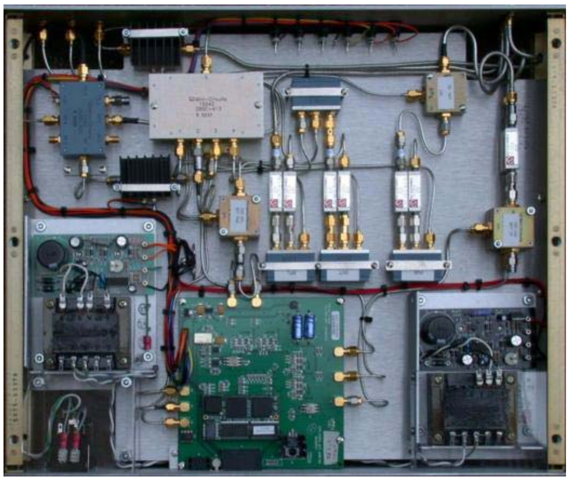

Figure 2: SNS LLRF Chassis top view.

Accelerator BPM readout

The SNS Beam Position Monitor system[10] is made up of 4 × 40MS/s 14-bit ADCs, 256K deep FIFOs, and a direct connection to a PCI bus. The chassis is a commercial 1U rack-mounted PC. A Quicklogic CPLD provides the PCI interface and custom interface logic.

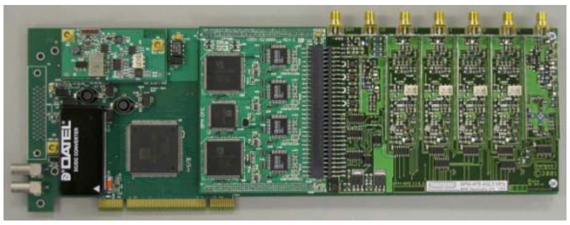

Figure 3: SNS BPM PCI card. Digitizers are in the center, downconversion mixers and filters are to the right.

Network Camera

The Elphel Model 303 High Speed Gated Intensified Camera[11] is a nice example of compact electronics connecting all the way from sensor array to Ethernet. A Xilinx XC2S300E FPGA performs (among other things) image compression from raw pixels to JPEG format. An ETRAX 32-bit CPU bridges the data to Ethernet, at a sustained rate of 15 frames (1.3 megapixels each) per second. This camera is powered by 48VDC through the Ethernet cable, compliant with the IEEE 802.3af standard.

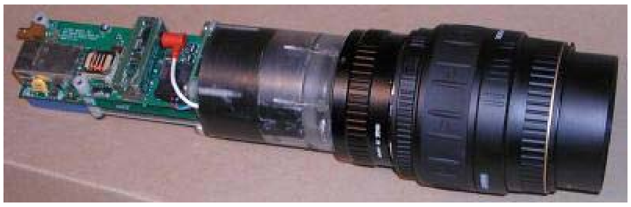

Figure 4: Camera assembly including CCD, acquisition, and networked computer

Home Audio

The LANPipe [12] is a simple network to audio bridge device. It includes a 16-bit CPU implemented on the XC2S30 FPGA, which in turn runs a simple UDP/IP network stack. The Ethernet chip includes both the PHY (physical) and MAC (logical) layer.

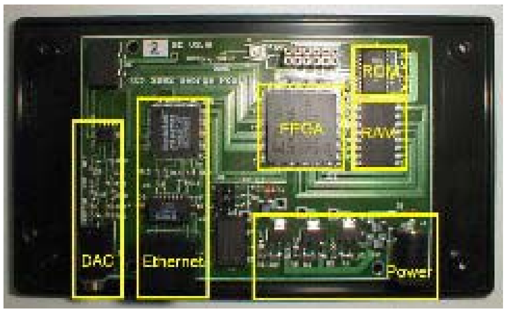

Figure 5: Annotated photo of LANPipe circuit board

BACK END IMPLICATIONS

As the front end functionality of a large installation is subdivided more finely, more network cables run back to the Global Control System. There is legitimate concern that network architectures be available to support this.

An example of modern large scale networked computing cluster is the University of Kentucky KASY0[8], which consists of 128 2.0 GHz Athlons. Its network has been demonstrated adequate to run software that requires a high degree of coordination between processors. With some additional of network gear (64 × 24-port switches), the KASY0 could provide 1408 network plugs (100 BaseT) for under US$33/plug. The hypothesized control system front ends would then be backed up with 400 GFLOPs and 1.5 GByte/second data throughput.

SYSTEM ON A CHIP

Normally this is a code word for combining the application specific logic with a host processor on a single chip. For technology reasons, DRAM and analog circuitry is almost never included in this chip. Both ASICs (Application Specific Integrated Circuits) and FPGAs can be used in SOC mode. Of course, the ASIC approach is only considered relevant when the production volume is large, greater than 100,000.

Besides potential system cost and space savings, a prime motivation for SOC architecture is to improve the interconnect between the high speed application logic and the host CPU. They are normally on different clock domains, and the bus interfaces to the CPU core are normally very much slower than the processor core. Typical numbers are 200-500 ns to transfer 16-32 bits. Compared to the speed of both the processor core (200-2000 MHz) and the FPGA plane (40-200 MHz), this is a serious bottleneck.

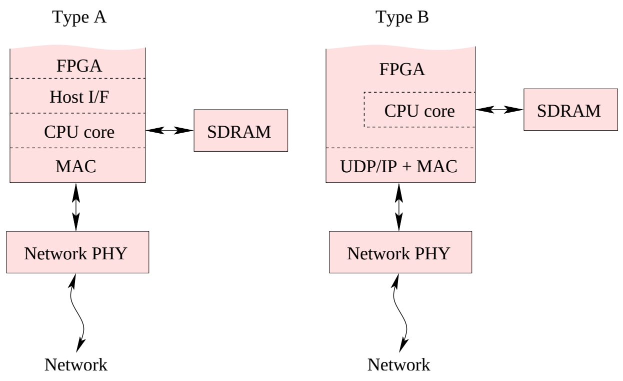

Figure 6: System-on-chip adaptations of the familiar block diagram.

Figure 6 shows two contrasting ways to adapt the familiar block diagram of figure 1 to SOC.

Type A keeps the processor in the datapath between the application hardware and the network. This approach is normally imagined as a hardware change only, where the existing software is moved onto the FPGA chip. That software is an existing body of high level code, including network protocols and a POSIX-capable operating system (like Linux). This concept makes large demands on the CPU and memory (at least 8 MiB, necessarily off-chip) by today's FPGA standards.

In the configuration of type B, the processor is by design kept out of the data path between the application hardware and the network. It is optional and (when used) reserved for local data manipulation, where the algorithm complexity is higher than plausibly programmed in dedicated HDL data path.

The high speed network connection is assumed to be simple, low latency transfers of raw data. The conversion to high level protocols can take place on commodity workstation-class hardware on the other end of the network cable.

Note there is community expertise running network protocols (especially Ethernet UDP/IP) in FPGA hardware without a traditional CPU. As wire speeds climb, having a CPU in the data path is likely to create more problems than it solves: at GB/s rates, even workstation-class CPUs get overloaded, so modern high speed NICs (network interface cards) are evolving into dedicated network co-processors.

"The cheapest, fastest and most reliable components of a computer system are those that aren't there."[13]

The flexibility and end-to-end integration of an FPGA-based SOC make it plausible to use Ethernet with a hard real time mind-set that is inconceivable using a CPU and a conventional MAC. Frame preamble and header information can be sent down the wire while results are still being acquired from the hardware.

There is not necessarily any hardware difference between approaches A and B. Approach A is more likely to work without the external memory chip, in part because of its simpler scope.

Soft CPU Cores

When the CPU is built with the FPGA fabric, just like the rest of the chip's functionality, it is called a soft core. Many such designs are published and/or sold, some of which are listed here.

name	source	bits	4-LUTs	MHz
PicoBlaze[14]	VHDL	8	152	40
SLC1657[15]	VHDL	8		
gr0040[16]	Verilog	16	257	50
xr16	schem	16	392	65
MicroBlaze[17]	N/A	32	1050	75
NIOS-16[18]	N/A	16	1100	50
NIOS-32[18]	N/A	32	1700	50
LEON SPARC[19]	VHDL	32	4800	65
Aquarius[20]	Verilog	32	5506	21
or1k[21]	VHDL	32	6000	33

None of these cores have an MMU (memory management unit). The speeds (and, to a lesser extent, the 4-LUT count) are only approximate since they depend on the speed and capability of the underlying FPGA. A 'N/A' in the 'source' column indicates that the source is not published, limiting the core's utility in a research context.

The advantages of a soft core are more competition, variety, and adaptability to the actual problem at hand.

Hard CPU Cores

When the CPU is built by the chip manufacturer on the same die as the FPGA fabric, it is called a hard core.

CPU core	chip	bits	MHz
PowerPC	Xilinx Virtex-IIpro	32	250
ARM9	Altera Excalibur	32	200
80C51	Triscend	8	

The first two of these designs include an MMU. Although less customizable, these cores have theoretically better cost/performance and speed-power product than a soft core. In today's FPGA generation, hard cores with external SDRAM are probably required for type A SOC implementations.

NETWORK CHOICES

At some point in the chain from hardware to operator, standards (as published by sanctioned standards bodies) are essential for communication between hardware built by different people at different times.

There are many historical standards for parallel bus attachments of peripherals to computers: CAMAC (IEEE-583), VME (IEEE-1014), VXI, GPIB (IEEE-488), SBUS (IEEE-796), ISA/AT, ATAPI (ANSI NCITS 317-1998 and later), PCI/cPCI. At the time of this writing, all are considered obsolete or dying, in many cases explicitly replaced with a serial equivalent. PCI sees extremely wide use, but is also very political, and many commercial interests appear eager to upgrade or replace it soon. Modern serial buses include Ethernet (IEEE-802), Firewire (IEEE-1394), Fibre Channel, USB, CAN (ISO 11898), SATA, and ATM.

Ethernet is both the oldest and most vibrant. It is in the heart of the wireless storm. Power Over Ethernet[22], which provides up to 13 W for peripherals over the same CAT5 cable as the network, is just taking off. Fiber and twisted pair transmission speeds are set for another jump in speed and/or availability. It's very hard to imagine any difficulty connecting Ethernet-based gear to the Internet anytime in the next two decades. The same cannot be said about *any* of the other listed protocols.

With ubiquitous CAT5 cable, 100BaseTX and 1000BaseT Ethernet will reach 100 m. On a fiber physical layer, 100BaseFX in full-duplex mode will reach 2000 m, and 1000Base-LX on a single mode fiber will reach 3000 m[23].

While not normally thought of as a hard-real-time link, point-to-point Ethernet does have deterministic latency. Direct links between networked front ends could take advantage of that to implement wide-area feedback and interlocks.

FIELD PROGRAMMING

FPGAs are an enabling technology. Their reconfigurability is an essential feature, allowing bugs to be fixed and features to be added to the hardware at a later date. This flexibility comes with a hardware price: some means of "booting" or "configuring" the FPGA must be included, and (to avoid losing the very feature that is so attractive) a mechanism must be included to make that configuration remotely updatable. When a conventional networked computer is part of the equation, the solution can be relatively easy: connect four JTAG leads to the computer's general purpose port, and have the FPGA activate only after the computer goes on line. This avoids dedicated Flash memory chips and all other hardware and software complexities. Normal software configuration control can place new FPGA configurations on a network server, where it will take effect on the next chassis reset or power cycle.

When interlocks are implemented with an FPGA, the equation changes: it has to be treated as a non-programmable device, and changes in functionality have to be accomplished by returning a unit to the bench, re-programming with specialized hardware, and the result re-tested before returning it to the field.

When the host computer is inside the FPGA, the configuration step involves a chicken-and-egg problem: the very computer (or computer-free network stack) that is needed to download FPGA configuration is implemented in the very hardware that needs configuration! FPGA vendors provide small and expensive Flash chips that can self-configure an FPGA at power up, but the infrastructure to reprogram these and restart the board (while leaving fail-safe options in place) is not easily understood. Since the chip count of an FPGA-based control board is normally very low to begin with, it seems imbalanced to add complex and fragile boot hardware.

CONCLUSIONS

Networked front end hardware has tremendous opportunity to make accelerator electronics simpler, cheaper, more featureful, better understood, and more reliable. By distributing the hardware closer to the gear it controls, field wiring becomes quieter and more maintainable. Standardized high speed network communications between front end modules and the global control system maximizes short and long term flexibility, and minimizes installation costs.

Since so much of the intellectual content of the devices will reside in its programming, it is appropriate to suggest widespread Internet-based collaboration within the community, as exists now in the SNS project and the EPICS collaboration. This “many eyeballs” approach can drive up quality and drive down costs compared with the “lock it in the desk drawer” approach.

Many more changes are on the horizon. Even with the demise of Moore’s Law looming in the next few years, imaginative applications of programmable digital circuitry will continue to enhance the performance and capabilities of front end hardware.

ACKNOWLEDGEMENTS

The author would like to thank the entire SNS team, especially controls, instrumentation, and LLRF groups, for their help as we all feel our way into the future.

REFERENCES

[1] IEEE-583-1982, Standard Modular Instrumentation and Digital Interface System.

[2] IEEE-1014, Standard for A Versatile Backplane Bus: VME-bus

[3] Gordon E. Moore, Cramming more components onto integrated circuits, Electronics, Volume 38, Number 8, April 19, 1965.

[4] Digital Signal Processing in Beam Instrumentation, M. E. Angoletta, DIPAC 2003, GSI, Mainz, Germany

[5] SPI is a trademark of Motorola, Inc. M68HC11 Reference Manual

[6] I^{2}C is a trademark of Philips Electronics N.V. http://www.semiconductors.philips.com/buses/i2c/

[7] 1-Wire is a trademark of Maxim.
http://www.maxim-ic.com/1-Wire.cfm

[8] http://aggregate.org/KASY0/

[9] http://recycle.lbl.gov/ ldoolitt/llrf/

[10] John Power, Beam Position Monitor Systems for the SNS LINAC, PAC2003

[11] http://www.linuxdevices.com/articles/AT2441343146.html

[12] http://www.thepowleys.com/lanpipe/index.php

[13] Gordon Bell, DEC laboratories

[14] http://www.xilinx.com/ipcenter/-processor_central/picoblaze/index.htm

[15] http://www.silicore.net/pr090903.htm

[16] http://www.fpgacpu.org/gr/index.html

[17] http://www.xilinx.com/xlnx/-xil_prodcat_product.jsp?title=microblaze

[18] http://www.altera.com/products/devices/nios/

[19] http://www.gaisler.com/

[20] http://www.opencores.org/projects/aquarius/

[21] http://www.opencores.org/projects/or1k/

[22] http://www.poweroverethernet.com/

[23] http://www.dslreports.com/faq/7800