Embedded controllers for local board-control

Flavio Fontanelli, Giuseppe Mini and Mario Sannino*,
Zbigniew Guzik†,
Richard Jacobsson, Beat Jost and Niko Neufeld‡
*INFN Genoa
Genoa, Italy
†Institute for Nuclear Research
Warsazwa, Poland
‡CERN
Geneva, Switzerland
Email: niko.neufeld@cern.ch

***Abstract*— The LHCb experiment at CERN has a large number of custom electronics boards performing high-speed data-processing. Like in any large experiment the control and monitoring of these crate-mounted boards must be integrated into the overall control-system. Traditionally this has been done by using buses like VME on the back-plane of the crates. LHCb has chosen to equip every board with an embedded micro-controller and connecting them in a large Local Area Network. The intelligence of these devices allows complex (soft) real-time control and monitoring, required for modern powerful FPGA driven electronics. Moreover each board has its own, isolated control access path, which increases the robustness of the entire system. The system is now in pre-production at several sites and will go into full production during next year. The hardware and software will be discussed and experiences from the R&D and pre-production will be reviewed, with an emphasis on advantages and difficulties of this approach to board-control.**

LHCb-PROC-2005-009
04/06/2005

I. Introduction

Traditional board-control has used buses like VME or Compact PCI to control custom or commercial electronics-boards. This approach has been proved to work well, however it has also shown to suffer from at least two draw-backs:

- Intelligence is centralized on the crate-controller, which depending on the task of the controlling entity can be a bottle-neck.
- The control-paths of a comparatively large number of boards are shared and linked together. A misconfigured or badly behaving device can block the whole chain.

LHCb has therefore chosen a different approach and equips every electronics board with a micro-controller, which is accessed via a dedicated Local Area Network (LAN).

In this paper we are first introducing the concept of a micro-controller, then we give a brief overview of the micro-controller, which has been chosen for LHCb. In the next section we present the hardware of the micro-controller and how board-control is achieved. In the third section we present the software framework which has been developed to operate a large number of micro-controllers in a distributed system. Finally, in the conclusions, we will review the advantages and drawbacks of this approach in the light of our present, five years of experience with the system now widely used in pre-production environments.

II. Micro-controllers

Micro-controllers are basically microprocessor systems on a chip (SOC), that is a micro-processor, with all, or most, associated electronics in a single chip. To create a full computer usually one only needs to add a few discrete components (such as connectors), and, if needed in larger amounts, memory, which becomes considerably cheaper when standard DRAM chips are used as opposed to built-in memory on chip.

Micro-controllers are ubiquitous in modern electronics, particular in consumer electronics such as cell-phones, MP3 players and many more.

They exist based on a wide variety of CPU architectures such as PowerPC, probably the most popular in the embedded world, ARM and also i386 (Intel) compatible.

The widest range of readily available software is definitely given for the i386 architecture, which is the reason LHCb has chosen it.

Rather than building our own micro-controller, we chose to buy a complete micro-controller with a standardised interface. The chosen product is the SM520PCX from Digitallogic [1]. It is based on the AMD ELAN 520 micro-controller [2] and comprises all necessary hardware on a plugin board of 85 x 66 mm^2. It is shown in Fig. 1. The ELAN520 is a i486 compatible processor running at up to 133 MHz, with all standard functionalities of a PC and in addition extra precise hardware timers, a hardware watch-dog for automatic reboot and GPIO lines. The SM520PCX is only one of a family of pin-compatible embedded PCs from Digitallogic.

III. Interfacing to custom electronics boards

A PC is not "naturally" suited to directly control resources on an electronics board. Most custom chips in LHCb are accessed via I2C or JTAG (IEEE 1394). JTAG is also widely used for in-situ programming (ISP) of configuration devices such as EEPROMs and FlashRAMs. For high-speed data-transfer and register access to FPGAs or memories these two methods are unsuitable. A parallel bus is needed. PCs offer two choices, the aged ISA bus and the more modern PCI. ISA is disfavored nowadays in the PC world, and is also not very

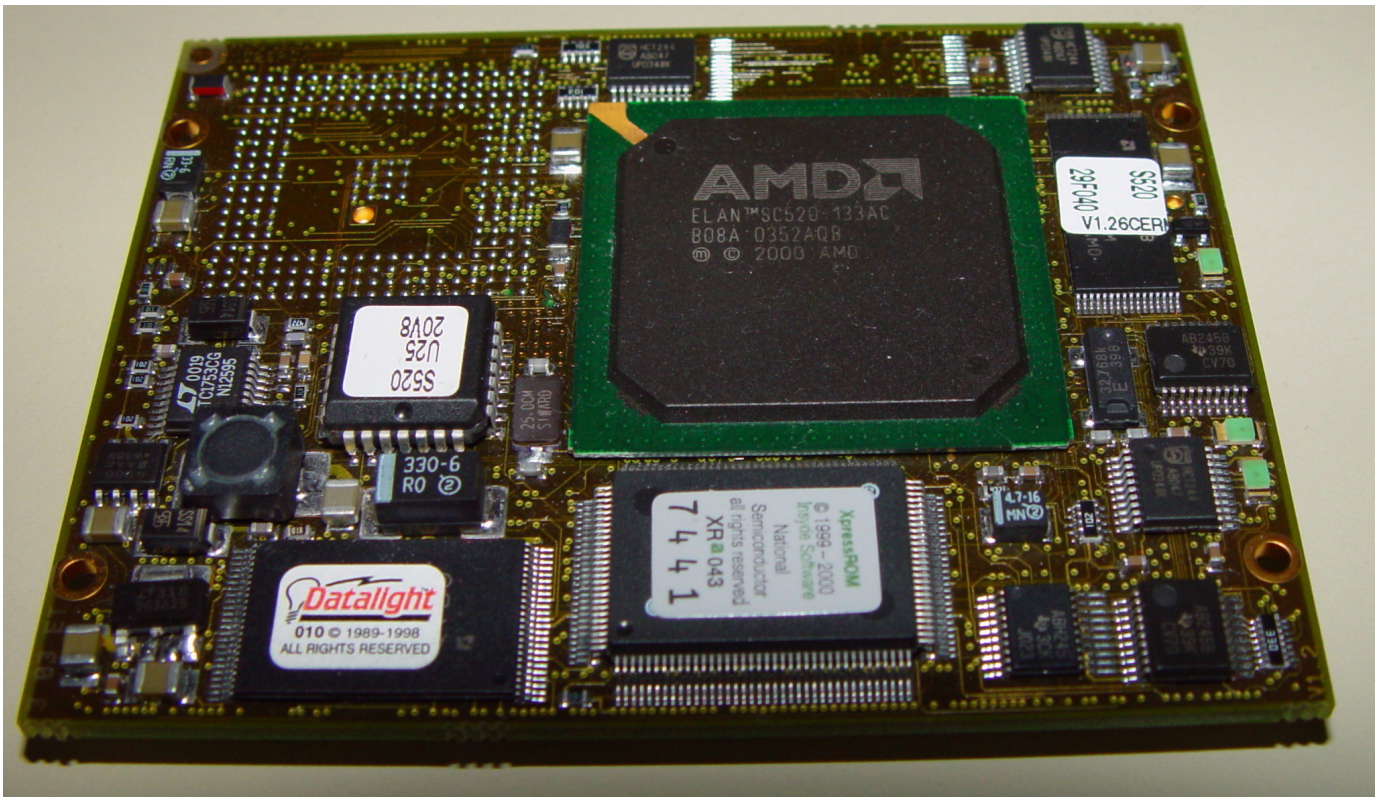

Fig. 1. Top-side of the SM520PCX embedded PC. The micro-controller together with some external components like the FLASH-RAM for the BIOS can be seen. A standard SO-DIMM memory-module, normally used in portable computers, is on the back.

popular with electronics engineers, for quite a few reasons, not the least being that it is a non-multiplexed bus.

PCI on the other hand, due to its signaling, which relies on reflective coupling, has impractical trace length restrictions (our electronics boards are typically 9U x 400 mm). We have therefore chosen to provide a simple local[1], parallel bus such as it is generated by a PLX PCI bridge [3].

Neither of these can be found on PC based micro-controllers, and also on other micro-controllers, their are usually only one or two I2C chains and only few and/or slow JTAG chains, while ISP in an environment where frequent reprogramming is necessary, definitely requires a high-speed JTAG chain.

We have therefore decided to build a small glue-board to create the required interfaces and connect them via PCI. Two approaches for the glue-card have been followed - an ASIC based and a purely FPGA based one.

In Fig. 2 the ASIC-based glue-card is shown. It is built-around a PLX PCI9030 PCI bridge, JTAG controllers from Texas Instruments and I2C controllers from Philips. It also includes a bus-switch to electrically isolate automatically the glue-card from the carrier-board, when the PC is rebooted. This board provides 4 I2C chains of 40 or 400 kHz, 3 JTAG chains of up to 2 MHz[2].

[1]Local here means not going out of the board.

[2]We have bench-marked the programming time of a 16-MBit ALTERA EPC16 programming device to be better than 10 s.

The functionality can also be implemented in an FPGA, which in this case acts directly as the PCI target and internally generates the I2C and JTAG required. This board is shown in Fig. 3.

When reading from registers or memories attached to the local bus from PCI, transfer performances better than 20 MB/s were achieved[3]. This is more than can be sent via a 100 Mbit/s Ethernet connection, and also more than be reasonably treated by a 133 MHz processor.

IV. Software for building a control-system

The micro-controller and glue-card solution presented above will be used to control, configure and monitor over 400 boards of approximately 15 different types. The R&D for these boards is done in many laboratories across Europe. We have therefore tried to provide a simple generic software distribution mechanism, which allows small, easy installations for a few boards, as well as larger multi-server setups and at the same time guarantees that updates of software are centrally transmitted to all installations, so that problems are not only fixed where they occur, but also pre-preempted at other sites.

A. Embedded software

We have chosen Linux as the OS of the micro-controller. The main reason is the easy customizability of the kernel for

[3]For FPGA registers this obviously depends of course strongly on the implementation on the carrier board.

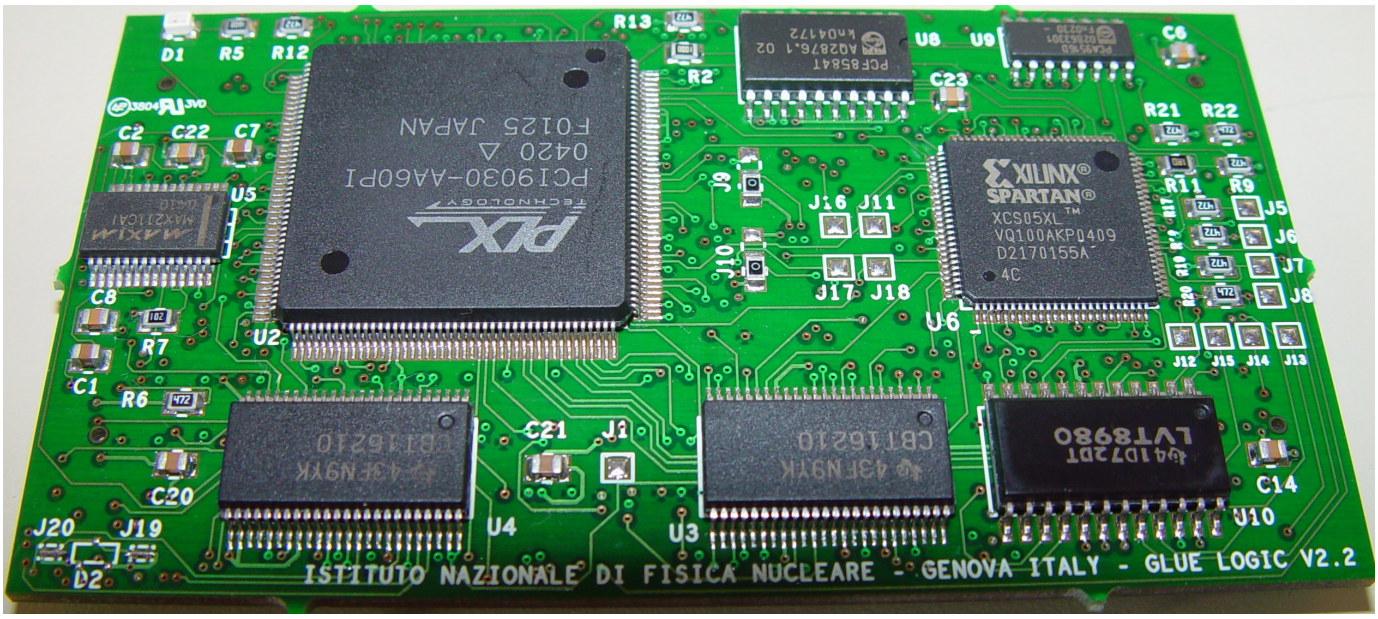

Fig. 2. The ASIC based glue-card.

Fig. 3. The FPGA based glue-card.

embedded, disk-less operation. In some R&D projects however Windows has also been used, which allowed using commercial software packages. All required functionality has since been ported to Linux.

On top of a slightly customized kernel we run the standard CERN Linux distribution, which is derived from the joint Fermilab and CERN Scientific Linux project [4].

The custom software, drivers and user-libraries to access the resources on the system are packaged and distributed in the same manner as the base-system, using the standard Redhat package management system (RPM) [5]. This is described in the next section.

Except for the device driver no special software, cross-compiler or anything is required. Editing, compiling and to the extent of not accessing the actual board hardwared debugging can be done on any Linux PC. Most developers run all development tools on a server, which is sharing the files with the embedded PC via NFS.

B. Management software

In a large collaboration it is necessary to disseminate the software quickly and transparently to all teams. Only in this manner can effective peer-help be achieved, because one can rely on a common software base. In our case this task is made more difficult because the teams are geographically very widely spread. There is also no common intranet, like in some large companies. A shared NFS server for the software for example, while from a performance point of view certainly possible, is not an option in the age of firewalls, net-filters and the like. We have based our software distribution therefore on HTTP and a simple automatic update mechanism. This requires a minimal administration effort by the users and can be even fully automatized.

In this manner all security updates and software patches, which concern packages not directly managed by us - and this is the vast majority - benefit from the effort of the respective support teams. It is thus not riskier to put any of the embedded controllers on the LAN or Internet, than it is to put a standard desktop PC running Scientific Linux. This allows *direct* remote accessibility of machines, a fact that greatly eases remote, rapid debugging and support.

The system relies only on a web-server and a few simple scripts, which are documented at our webserver [6].

C. Highlevel software

At a higher level all control functionality in LHCb is integrated in the Experiment Control System (ECS), which is based on a commercial SCADA system. This system provides alarm-handling, finite-state machine modeling, graphical user interfaces and the like. More details on this layer and its capabilities can be found in [7] and [8].

V. Conclusions

A. Commercial solution

Choosing a commercial solution for the micro-controller board, had the great advantage to leverage on wide-spread experience in PC main-board design in industry, not so much available in-house. Moreover, the design against a standardized connector pin-out proofed to be very valuable because it effectively shielded us from problems with component obsolescence.When the producer of the original micro-controller went out of business, Digitallogic redesigned a board with the identical connector and mechanical dimensions around a different micro-controller from AMD, the ELAN520 we are using today. This extra design effort, was completely (also price-wise) transparent for us[4]. Had we integrated the micro-controller directly we would have been forced to do a complete redesign. Unfortunately also commercially available components can have unexpected problems. Very late in the course of our tests it turned out that due to the specific connector used on the SM520PCX module, it was not possible to drive Ethernet from the card over more than 20 m at 100 MBit/s. Apparently verification had only been done at 10 MBit/s or short cables, where the problem does not occur.

Another disadvantage is the need of a separate glue-board, which necessitates additional connectors and routing of many signals, which makes integration on carrier-boards somewhat cumbersome. A micro-controller, which already has several of the interfaces (I2C, JTAG) available on chip and is combined with an FPGA part, would allow to design everything really on a single chip. Such chips exist already, unfortunately not yet based around an i386 compatible core, which we believe is a key advantage for long-term, low-effort maintenance of a large distributed system.

There is currently a lot of interest in these systems, as can be seen in these proceedings, for example see [9] and [10].

B. Linux as general purpose OS

Using Linux proved to be an excellent choice. Users are immediately at ease, practically any software available on normal desk-top machines runs easily, with the unfortunate exception of some proprietary FPGA programming tools, where extensive reverse-engineering was required to provide the functionality from Linux. The fact of not using a hard real-time version of Linux nowhere proved to be a problem. High-speed time-critical control is anyhow better done on the carrier-board itself in state-machines on the FPGAs. Since each PC controls only one board, time-critical tasks can always resort to polling, which allows for an excellent response time, if a low rate of small but unpredictable extra delays can be accepted. Critical tasks, which should not be interrupted could be added to the kernel. Linux can be made very-light weight and still be kept up-to-date with whatever is in fashion in the "real world". This is very important in the conception of a system which will have to run for some 15 years.

C. Comparison with crate-controllers

The combined cost of the micro-controller glue-card combination described in this paper is about 220 Euros. For a

[4]The only real difference is a slightly higher power-consumption of the ELAN520, which uses on average 1.4 Ws, a value perfectly acceptable for our purposes.

full crate with 20 user-slots this gives a cost of 4400 Euros to which the cost for an Ethernet switch (starting from 300 Euros) have to be added. A modern, performing crate-controller costs at least as much. Thus cost is certainly no obstacle for the micro-controller solution, which we are convinced is better scalable and more robust than the bus-based one.

ACKNOWLEDGMENTS

The authors would like to thank the whole LHCb Online team and the many users of this system for their help and feed-back.

REFERENCES

[1] Digitallogic, Switzerland http://www.digitallogic.ch
[2] Advanced Micro Devices, http://www.amd.com
[3] PLX, http://www.plxtech.com
[4] Scientific Linux, http://www.scientificlinux.org
[5] RPM, http://www.rpm.org
[6] LHCb Online webpages, http://cern.ch/lhcb-daq
[7] S. Schmeling, Common Tools for Large Experiment Controls - A Common Approach for Deployment. Maintenance and Support, these proceedings
[8] C. Gaspar and B. Franek, Tools for the Automation of Large Physics Experiments, these proceedings
[9] J. Alme et al., The Control System for the ALICE TPC Detector, these proceedings
[10] S. Silverstein et al., A Simple Linux-based Platform for Rapid Prototyping of Experimental Control Systems, these proceedings