RECONFIGURABLE HARDWARE RESOURCES IN ACCELERATOR CONTROL SYSTEMS

Dario Giove, Carlo De Martinis , Marco Mauri

INFN-Milan and University of Milan

Abstract

The development of modern accelerator control systems has taken advantage of the possibility to use standard architecture designs based on the experience gained in industrial applications. Communication buses, board formats, operating systems, network protocols and operator interface software are the main elements of this new approach. In this paper we will discuss the way to apply this method also to the design of electronic boards which call for custom design of particular circuits and capabilities. The use of FPGA based standard modules along with the possibility to customize them using a standard LabVIEW environment to obtain reconfigurable hardware resources will be presented.

INTRODUCTION

In the last years algorithm reconfigurable devices such as field programmable gate arrays (FPGA) and digital signal processors (DSP) have been widely used in the industrial world as system-on-chip (SoC) applications. Those devices make it possible to develop a high performance system in a short period of time, and moreover it is possible to have a small quantity of devices at reasonable costs. The structure of SoC devices with FPGA is fully flexible and can respond to change of the logic of a control system easily. The capability of modifying the logic enables us to easily implement future additions to the system. These features, along with increased performance demands, have fuelled the push of digital technology deeper into the controlled devices. The justifications often quoted for this push to digital include reproducibility, increased stability, increased resolution, and decreased infrastructure costs (networks replace control wiring).

The actual trend in the design of accelerator control systems is to use as much as possible commercial off-the-shelf (COTS) products. The use of COTS products as elements of larger systems is becoming increasingly commonplace. Shrinking budgets, accelerating rates of COTS enhancement, and expanding system requirements are all driving this process. The shift from custom development to COTS-based systems is occurring in both new development and maintenance activities.

Nevertheless the need to integrate different subsystems coming both from the industrial market and from international research collaborations, and the special requirements arising from the operation of particular piece of equipments calls for the development of custom boards along with the related firmware/software.

These developments may require an effort which is not allowed by the skill and the man power available in the control teams and/or a cost which is in excess with respect to the equipment to be controlled.

The possibility to apply FPGA based design to address these demands has been a successful option in a lot of different situations.

In this paper we will present and discuss a further evolution of this concept due to the last year announcements by National Instruments of a new software platform along with the first related devices.

Since 1992 the development environment named LabVIEW has been the most impressive evolution in the world of data acquisition and analysis programming. The capabilities offered by this language have found a widespread use in the industrial and research world making it a de-facto standard. In the last years more efforts have been devoted to address specific needs which required a less general approach (vision, real time, SCADA, etc.) In 2003 National Instruments (NI) extended the LabVIEW graphical development environment to FPGAs. By configuring the FPGAs on special designed NI RIO hardware, one can offload tasks from Windows based or Real-Time host machine and achieve a level of determinism only possible on a hardware platform.

The possibility to use the same environment and the same techniques to write code for general purpose control applications and for device programming has been a real revolution in the world of modern programming. The need to have engineers skilled in VHDL language and to develop special interfaces between FPGA designs and control computers is now no more true.

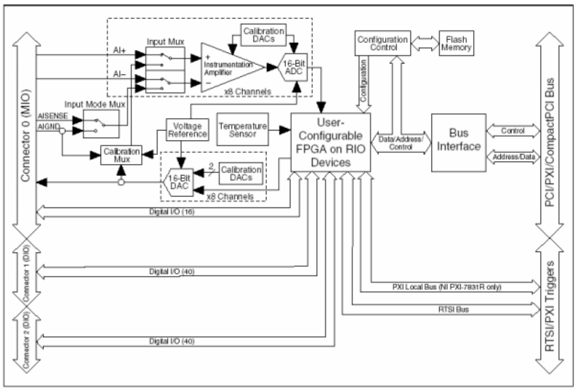

Fig. 1 Block Diagram of the NI-7831 board

RECONFIGURABLE I/O DEVICES

The block diagram of the first board (named NI-7831R) developed by NI to support the new architecture is shown in fig. 1

The NI 7831R is based on a reconfigurable FPGA core surrounded by fixed I/O resources for analog and digital input and output (8 ADC channels, 8 DAC channels, 96 DIO channels). It is possible to configure the behavior of the core to match the requirements of the measurement and control system. LabVIEW logic and processing may be implemented in the FPGA of the R Series device. Typical logic functions include Boolean operations, comparisons, and basic mathematical operations.

Multiple functions may be available efficiently in the same design, operating sequentially or in parallel. Software accesses the R Series device through the host computer bus interface, and the FPGA connects the bus interface and the fixed I/O to make possible timing, triggering, processing, and custom I/O functions using the LabVIEW FPGA Module. Each fixed I/O resource used by the application uses a small portion of the FPGA logic. The bus interface also uses a small portion of the FPGA logic to provide software access to the device. The remaining FPGA logic is available for higher level functions.

The NI 7831 R uses a XILINX Virtex II FPGA with 5120 logic slices (equivalent to 11520 logic cells), 80 Kbytes of internal memory and a timebase clock of 40 MHz.

The architecture of the NI 7831 R allows to address issues related to the possibility to develop a multiprocessors architecture in PCI and cPCI systems. Asymmetrical multiprocessing ties together different FPGA boards. Each board is "intelligent" in that it has its own processor and copy of the operating system. Fig. 3 shows this basic configuration, with the user interface and the network part of the system implemented on the host CPU, and additional real-time processing power provided by I/O modules based on FPGAs.

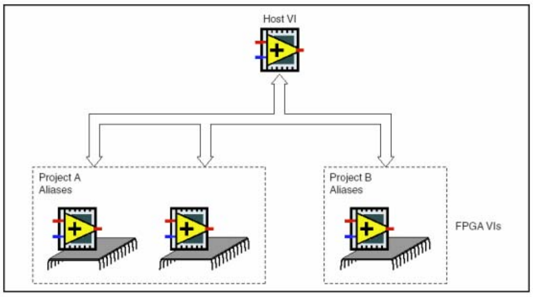

Fig.2 Multiple FPGA boards working in a parallel fashion

LABVIEW FPGA MODULE

The LabVIEW FPGA Module provides the same graphical programming environment for the creation of code (FPGA VIs) as LabVIEW does for standard VIs. The LabVIEW graphical programming environment includes front panels and block diagrams, powerful editing tools, and a wide range of included functions.

It is possible to design FPGA code that allows the FPGA device to operate independently of the rest of the system. This results in the fact that a robust FPGA code that use the ability to operate independently and continue to run even if the *host computer*—the computer that controls and monitors the FPGA device—crashes. Furthermore, it is possible to design the FPGA code to store data on the FPGA until the host computer can retrieve the data. Another advantage of the FPGA Module is parallel execution of block diagram operations in an FPGA code. Portions of the block diagram that do not depend on other portions execute in parallel on the FPGA device. For example, multiple independent While Loops on a block diagram each have independent portions of hardware. Therefore, the multiple independent While Loops run simultaneously on the FPGA device.

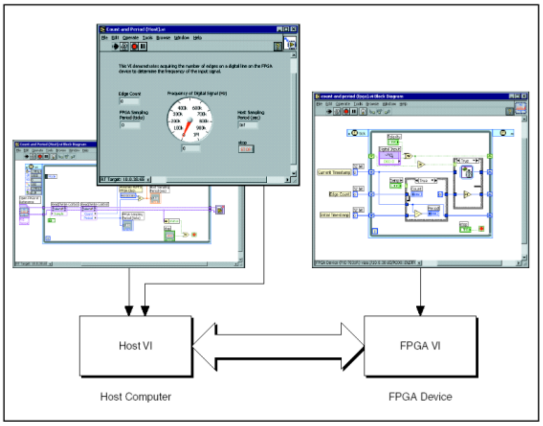

Fig.3 Communication architecture between host and FPGA devices

FPGA AND HOST COMPUTER COMMUNICATIONS

After a FPGA VI has been loaded on the FPGA device, we have to study carefully the way to communicate with it. Depending on the application requirements, there are two possible architectures to communicate with the FPGA VI: interactively or programmatically. Use Interactive Front Panel Communication to communicate with the FPGA VI directly from the front panel of the FPGA VI. Use Programmatic FPGA Interface Communication to communicate with the FPGA VI from a VI running on the host computer. The VI running on the host computer is called the *host VI*.

Systems that require to create a multi tiered application with the FPGA device as a component of a larger system or that need to perform operations not available on the FPGA device, such as floating-point arithmetic, have to follow the programmatic communication model.

To investigate the performances that can be obtained in such a model we have developed a driver between a CPU board and an FPGA board. The driver has been designed so that the FPGA board set an interrupt to the CPU and then it measures the time elapsed before getting back an ACK signal (in order to measure the latency to the interrupt and the start time of the interrupt service routine) and the time to get 40 byte of data. The CPU runs standard Windows 2000 operating system and it is based on Pentium III 1 GHz component. The elapsed time has been measured using internal counters on the FPGA and the tick resolution is of the order of 25 ns. Figure 4 shows the basic code in the FPGA

The results obtained may be considered promising since the cycle involving only the ACK signal requires 31 microsec. and the further reading of 40 bytes adds 34 microsec. The jitter in the measurements is very low (of the order of 3%)

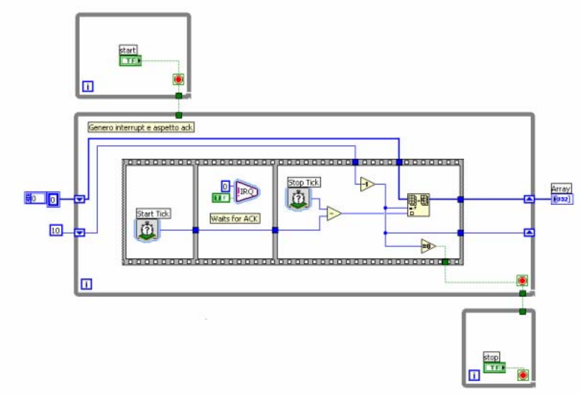

Fig.4 FPGA LabVIEW code to measure communication performances

APPLICATIONS

We have developed a few applications based on the items so far discussed. The design of each one of these applications allows an easy integration within already existing systems.

Programmable Waveform Generator

The first application has been an instrument able to generate a set of correlated waveforms each one with different frequency and duration. The typical time base of the device is of the order of 25 ns. in order to generate square waves up to 10 MHz and with the capability to modify the frequency of up to 6 orders of magnitudes without any discontinuity in the output. The application of this board is typically as a time generator for fast data acquisition sampling or as a complex trigger in sequence based devices. A single board solution has been designed and the whole application requires nearly 30% of the available resources.

Serial Bus Simulator

The second application has been devoted to build a simulator of complex protocols based on serial communications with speeds up to 1 Mbit/s. Usually these protocols refers to components coming from defence applications and that are no more available due to obsolescence reasons. We have implemented and tested a single board solution able to drive up to 4 of these links.

Timing Generation and Distribution

The third application refers to the possibility to design a master timing generator and the related peripheral receiving modules, to be used as the basic building block for a digital distributed timing system with a resolution of the order of 1 microsec.

The master timing generator has been designed using 2 FPGA boards inserted in a cPCI chassis with a general purpose Pentium IV based CPU. The first FPGA board handles the schedule of the events as it gets from the CPU every 500 msec. The second encode the timing events (max 16 bytes long) emulating a Manchester biphase serial encoder (in order to have a 1 Mbit/s channel with synchronization embedded) and waits for single digital events. The serial bus may be converted in optical to reduce the possibility to suffer noise from the environment. The optical signal may be distributed using passive optical splitters.

The peripheral modules are based on a single board design and they integrate both the equivalent Manchester biphase serial decoder and a local intelligence to handle the timing words and generate digital events as single triggers, waveforms, state transitions. The peripheral module may communicate with a host CPU to share these events on a cPCI bus (taking as a reference the performance above discussed) or to get more complementary information received from high level interfaces (Ethernet).

CONCLUSIONS

We have developed a few applications with single FPGA boards and others are in progress. The technology and the tools available by National Instruments seem very interesting to be used in accelerator control system applications.

ACKNOWLEDGEMENTS

We would like to acknowledge very helpful discussions and the valuable support received by ing. Paolo Ghidetti (SIDeA SpA – Milan).