Reconfigurable computing concept for the on-shore data acquisition system of a km^3-scale underwater neutrino telescope

A. Belias [a,*],

[a] *NESTOR Institute for Astroparticle Physics, National Observatory of Athens, 24001 Pylos, Greece*

Abstract

The on-shore data acquisition system of a km^3-scale underwater neutrino telescope is required to acquire continuously the digitized signals of 10000 photomultiplier tubes to apply filtering methods and search for signatures of candidate muon and neutrino events in real time. We propose the use of reconfigurable computing architectures, to perform the data acquisition tasks of a neutrino telescope its control system and the associated underwater observatory instruments.

Key words: KM3NeT, DAQ, reconfigurable computing, FPGA
PACS: 95.5.Vj; 01.30.Cc; 07.05.Hd

1. Introduction

An underwater neutrino telescope, based on the detection of Cherenkov light in water, requires many thousand photomultiplier tubes (PMTs) in order to cover a large sensitive effective area. The design study for a km^3-scale underwater neutrino telescope and multidisciplinary underwater observatory in the Mediterranean Sea [1] examines, amongst other aspects, different readout schemes, one of which being to send all digitized PMT data to shore to be processed in real-time. The required bandwidth is estimated to be $0.1Tb/s$.

Whereas a reasonable number of optical fibres can accommodate this data rate, it is too excessive for any data storing capacity, making the on-line filtering of the raw data imperative. For the data acquisition (DAQ) tasks foreseen on-shore of an underwater neutrino telescope we examine the uses of reconfigurable computing, focusing on commercial-off-the-shelf Field Programmable Gate Arrays (FPGAs).

2. Reconfigurable computing - FPGA based systems

Since originally proposed in the 1960s [2], Reconfigurable computing is becoming an increasingly important computing concept. Reconfigurable computing is intended to bridge the hardware software gap, achieving potentially much higher performance than *just* software, while maintaining a higher level of flexibility than *just* hardware.

Reprogrammable logic chips, of which FPGAs are considered here, permit the device hardware to be programmed multiple times after manufacture. A FPGA is a semiconductor device containing a collection of logic blocks, memory blocks and routing interconnect, with programmable input and output. With the advances of semiconductor process technology, FPGAs have gained high logic densities and a wide range of logical blocks and input/output signalling. Those device architectures have evolved

* Corresponding author.
Email address: belias@nestor.org.gr (A. Belias).
URL: www.nestor.org.gr (A. Belias).

with inclusion of features such as embedded processors, large embedded RAMs and high-speed serial I/O functionality, all of which exhibit an extremely flexible, high-speed processing potential.

The fine-grained programmability of the configurable logic fabric permits bit-wise hardware customization, enhancing performance for applications specific data widths. FPGA-based systems are of special interest since they are capable of migrating parts of their functionality between software and hardware implementations targeting optimal performance and best use of resources for a specific application.

2.1. *System Features*

The motivation for using FPGA-centric embedded systems is driven primarily by the following features:

- Speedup
  - Hardware executes faster than software, under the right conditions.
  - FPGAs are inherently parallel through a combination of spatial parallelism (i.e. multiple identical units) and temporal parallelism (i.e. pipelined units).
- Flexibility
  - FPGAs exhibit functional flexibility like software while retaining the execution speeds like dedicated hardware.
  - Through configurable logic, interconnects and interfaces, one can implement custom functional units.
  - The reconfigurable hardware logic can be reprogrammed multiple times.
- Cost
  - Development costs for an FPGA are less than for application specific circuits.
  - The same FPGA hardware can be re-configured for use in a different application.

Although an FPGA clock rate is typically slower that of a CPU, hardware implemented digital filters can process data at many times that of software implementations[3]. In a dedicated application specific implementation most of the transistors in a microprocessor spend nearly all of their time idle. In contrast, FPGA architectures make it possible to configure their transistors and peripheral I/O around the computing problem, and with a large number of data paths, offer the potential in performance improvements relative to conventional general purpose microprocessor architectures.

FPGA Pipelines lend themselves naturally to parallelism and a single design may include multiple parallel pipelines, with depths of hundreds or even thousands of cycles. This, combined with other parallelization schemes such as logic replication and concurrent scheduling of independent operations, gives FPGAs their performance potential.

An FPGA-centric implementation which leads to reducing the number of devices in an embedded system also reduces inter-device communication overheads, reducing the likelihood of data bottlenecks, I/O power consumption, and signal integrity problems while increasing development flexibility. This approach also makes algorithm partitioning easier because there are fewer devices between which algorithms must be split. Such features further increase the proportion of processing capacity that is available using FPGAs.

The logic within the FPGA can be changed if or when it is necessary, which has many advantages. For example, hardware bug fixes and upgrades can be administered as easily as their software counterparts. In order to support a new version of a network protocol, one can redesign the internal logic of the FPGA and send the enhancement to the affected applications. Once the new logic design is downloaded to the system and restarted, the new version of the protocol will be in use.

Another aspect of reconfigurable computing involves manipulation of the logic within the FPGA at run-time. This allows the design of the hardware to change in response to the demands placed upon the system while it is running. The FPGA acts as an execution engine for a variety of different hardware functions, some executing in parallel, others in serial, much like a CPU acts as an execution engine for a variety of software threads.

2.2. *System Architectures*

A type of reconfigurable computing architecture is one that combines FPGAs and conventional microprocessors. At its simplest it is a conventional microprocessor-based platform (PC motherboard) that has a FPGA-board connected to it. Although variations on the theme do exist, the typical instantiation of an FPGA in a system is in a co-processor model with the FPGA-board connected into the PCI, PCI-X or PCI Express buses. Boards

with several interconnected FPGAs are available as components-off-the-shelf hardware. The microprocessor platform provides the housekeeping functions and run the operating system and user interface.

A typical workflow in most co-processor models involves loading data into the FPGA with a user initiated Direct Memory Access (DMA) operation and loading the results back to main memory. This architecture has the advantage that PCI buses are ubiquitous through all general-purpose computing platforms and provide a solid and well-known development environment. However, because of the slave I/O nature of the interface care must be taken to keep the latency to the FPGA low. The I/O bus limitations can impact the bandwidth back to main memory and the limited PCI bandwidth can become a bottleneck when integrating the FPGA into higher bandwidth networks and file systems.

While FPGAs can create data paths with enormous throughput, the application specific designs typically fall short in the communication path between the FPGAs and the host processors. Recent developments in FPGA augmented computing enable the direct connection of the FPGA into the Front Side Bus (FSB) of the host microprocessor system [4], [5].

An FPGA plugs directly into a CPU socket, of a multi-CPU microprocessor system, allowing close access to the host processors and system resources.

Connecting directly to the FSB provides a high bandwidth, low latency connection between the FPGA, host processors and system memory offering the potential for dramatic performance improvements. A simplified programming model in virtual memory illustrates how FPGA-FSB coupling in a microprocessor architecture provides FPGA users with distinct performance advantages. Using virtual memory, the user can map a virtual address to access the memory of the FPGA. Without virtual memory, the user would need to read and write data back and forth between user and kernel space (similar to a disk drive access).

Another added benefit of using virtual memory is that common synchronization constructions such as barriers and semaphores can be used to synchronize the workings of multiple FPGA cards in the same host.

The PCI-bus type interconnect between FPGA modules and the host processor has made it necessary to copy large chunks of data into local banks of dynamic RAM.

This increases the complexity of the FPGA module and decreases the number of I/O pins which would be available to provide application specific interfaces for the user.

3. FPGA-augmented Data Acquisition

The recent report [1] released by the KM3NeT project discusses, amongst other, concepts in acquisition, processing, distribution and storage of data from a km^3-scale deep-sea neutrino telescope and a multidisciplinary underwater observatory. The neutrino detection method is based on Cherenkov light and the basic detector unit is a Optical Module (OM), employing photomultiplier tube(s) in a glass sphere with the high voltage supply and front-end electronics.

Owing to the large number of OMs (10000), the singles hits of each OM ($100kHz$) in the sea and the byte-size of each hit, the direct archival to storage is technical unfeasible. Regardless, of the front-end electronics implementation and any local trigger schemes in the deep-sea, the DAQ system should be able to accommodate a readout scheme where all data arriving to shore must be processed in real-time, including data from detector control and the associated underwater observatory instruments.

The basic functions to be performed by the DAQ system are outlined briefly:

- Aggregation - The continuous and dead-time-less readout of data from each OM into buffers of processing systems on-shore.
- Filtering - Integrity checks of the data, followed by trigger algorithms to reduce the background rates by factors of 10^4 to 10^5. The trigger algorithms use pattern recognition in local and extended areas of the detector acting on a snapshot of the data of the whole detector.
- Event Building - For any data passing any trigger criteria, an event will be built with all detector data around programmable time windows of the ensemble of OMs causing the trigger. Likewise, with programmable spatial windows, an event will be built, if certain detector areas for a given time period have satisfied trigger criteria.

Features of FPGA-augmented computing that are useful for the DAQ tasks are highlighted in the following:

- The bit data capabilities of FPGAs are superior to those of microprocessors and lend themselves

to filtering and pattern recognition algorithms.
- Signal processing algorithms such as Fourier transforms, convolutions and digital filtering also lend themselves very well to FPGA use. These algorithms are very computationally rich with relatively little decision structures.
- The advantage of reconfigurable FPGAs is that communications protocols and algorithm implementations can be tested in-situ without hardware changes to the host architesture. This allows adapting the resources of the reprogrammable hardware to particular demands of the neutrino telescope user community.
- FPGAs have been widely used in format conversion in the telecommunication industry with an extension into the media broadcast industry.
- The 'dual port' nature of FPGA designs allows data to "pass through" the FPGA and enables pre- and post-processing of data.
- FPGAs offer many channels of multi-gigabit signalling that provide direct access to the logic fabric.
- Various switched network fabrics (InfiniBand, Rocket I/O) can interface multiple FPGAs to accommodate the high transient rates of data, while handling the data reduction functions.
- With FPGA-augmented computing, scalability can be achieved through increase of FPGA systems in the same number of microprocessor host platforms.

4. Implementation on FPGAs

Unlike a microprocessor, an FPGA has no inherent functionality, and requires a bitfile to instruct the FPGA on how to interconnect its internal resources. The bitfile is the result of an automated physical synthesis process, based on a description of the behaviour required by the FPGA. One may either explicitly state the logical elements that are required in terms of base logical units of the FPGA, or they may describe the behaviour required in a more abstract manner and have the synthesis tools select the exact physical structure that implements this behaviour. Many designs use a mix of both techniques, which often compels developers to use a wide variety of discrete tools. Even for experienced software programmers, the efficient implementation of algorithms on FPGAs presents an initial learning curve requiring specialized understanding of digital design, however, vendor-specific tool flows have emerged which overcome this by providing a library of functional modules present in the FPGA and by abstracting away the complexities of the tool chain. Prior to deploying the code to the FPGA hardware, simulators are used to verify the FPGA behaviour, from simple logic up to complex system level.

5. Conclusions

The strength of reconfigurable computing stems from the capability to customize a hardware solution to a specific problem; yet retaining the flexibility of a software implementation. The emergence of FPGA-based systems within general purpose microprocessor platforms makes it possible to construct application specific embedded systems with reconfigurable components-off-the-shelf hardware and widens the scope of applications for which FPGA processing is viable.

The on-shore DAQ for a km^3-scale Neutrino telescope can draw considerable advantages from the FPGAs flexibility and performance to create optimized hardware architectures for each of the DAQ tasks, data aggregation, filtering and event building. Furthermore, reconfigurable computing schemes allow for optimal use of hardware resources and for additional, on-demand, algorithm and implementation changes in already deployed data acquisition hardware systems.

References

[1] KM3NeT Collaboration, KM3NeT, *Conceptual Design Report for a Very Large Volume Neutrino Telescope in the Mediterranean Sea*, April 2008.

[2] G. Estrin et al., *Parallel Processing in a Restructurable Computer System*, IEEE Trans. Electronic Computers, vol. EC-12, no.5, Dec. 1963, pp. 747-755.

[3] Tessier R., and W. Burleson, *Reconfigurable Computing for Digital Signal Processing: A Survey.*, Journal of VLSI Signal Processing, 2001.28, pp. 7-27.

[4] Silicon Graphics Inc., *Reconfigurable Application-Specific Computing User's Guide (007-4718-007)*, 2008.

[5] Ian McCallum, *Intel QuickAssist Technology Accelerator Abstraction Layer (AAL) 317481-001US*, 2007.