Fusion Engineering and Design 128 (2018) 122–125

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

A framework for the integration of the development process of Linux FPGA System on Chip devices

A. Rigoni[a], G. Manduchi[a,*], A. Luchetta[a], C. Taliercio[a], T. Schröder[b]

[a] Consorzio RFX (CNR, ENEA, INFN, Università di Padova, Acciaierie Venete SpA), Padova, Italy
[b] Max-Planck-Institut für Plasmaphysik, D-17491 Greifswald, Germany

ARTICLE INFO

Keywords:
FPGA
System on chip
ADC
Timing systems

ABSTRACT

System on Chip is a hardware solution combining different hardware devices in the same chip. In particular, the XILINX Zynq solution, implementing an ARM processor and a configurable FPGA on the same chip, is a candidate technology for a variety of applications of interest in fusion research, where FPGA fast logic must be combined with CPU processing for high-level functions and communication. Developing Zynq based applications requires the development of the FPGA logic using the XILINX Vivado IDE, mapping information between the FPGA device and the processor address space, developing the kernel drivers for interaction with the FPGA device and developing the high level application programs in user space for the supervision and the integration of the system. The paper presents a framework that integrates all the above steps and greatly simplifies the overall process. The framework has been used for the development of a programmable timing device in Wendelstein 7-X. The development of new devices integrating data acquisition and timing functions is also foreseen for RFX-mod.

1. Introduction

The use of FPGA based solutions in control and data acquisition systems (CODAS) for nuclear fusion devices has been in the past rather limited if compared to other physics experiments such as accelerators. This fact is mainly due to the different requirements: while in accelerators it is necessary to handle a very large amount of fast events from detectors, requiring fast data reduction on the fly based on coincidences, in fusion experiments a lower number of channels is used, typically requiring the acquisition of input signals for data storage and possibly real-time control. Therefore, fusion experiment use more conventional electronic devices such as transient recorders, replaced in recent long lasting experiments by Analog to Digital (ADC) devices supporting a continuous output data stream. Moreover, the dynamics of the phenomena controlled in real-time, such as plasma stability, require in most cases a response time of the order of milliseconds, whereas the control of the fastest phenomena such as vertical stabilization in tokamaks require a response time of the order of 100 μs. These requirements can be satisfied using the current computer technology making therefore the use of general purpose computers preferable over specialized FPGA solutions. Developing FPGA solutions requires in fact skills and expertise in the Hardware Description Languages (HDL) and hardware interfaces. Considering also that the integration of custom FPGA systems in CODAS normally requires developing some kind of specialized communication protocol, the amount of required human resources to implement such solutions is often unaffordable, especially in small laboratories. For this reason, FPGA solutions have been in the past limited to specific applications in diagnostics [1,2]. A notable exception is certainly represented by the RIO FPGA architectures [3] (Compact RIO and Flex RIO) which provide an easy FPGA programming and integration via LabVIEW and have been widely adopted for plasma control [4] and other diagnostic applications [5]. This solution, proposed by National Instruments, aims at leveraging the power of FPGA by removing the main barriers in their usage, that is the expertise required in HDL programming and interfacing with the rest of the system. This solution is however quite expensive and closed to the specific choice in hardware and in the programming environment. A new modern approach for the integration of high-level software components with the power of the FPGA logic design is obtaining growing attention in the market of embedded technologies and exploits the System on Chip (SoC) solution that combines different hardware devices in the same chip. The main hardware competitors leading the SoC FPGA market are Intel/Altera and Xilinx, both proposing almost the same development solutions but with their own proprietary software. In particular the XILINX Zynq architecture [6], implementing an ARM processor and a configurable FPGA in the same chip, is a valuable candidate technology for a variety of applications of interest in fusion research, where FPGA fast logic can be combined with software functions carried out by a CPU

* Corresponding author.
E-mail addresses: andrea.rigoni@igi.cnr.it (A. Rigoni), gabriele.manduchi@igi.cnr.it (G. Manduchi).

https://doi.org/10.1016/j.fusengdes.2018.01.042
Received 9 June 2017; Received in revised form 28 December 2017; Accepted 17 January 2018

for high level functions and communication.

A considerable number of heterogeneous hardware from many vendors have been released profiting of the high integration of SoC devices. The main advantages that these chips brings to the programmable logic are the possibility to interface and share hardware features that are typical of a complete system such as the DMA controller and external interfaces like Ethernet or SATA.

Many software solutions have been also proposed, to guide the developer through the non-trivial mechanisms of the FPGA to system interfacing, as well as covering different programming approaches: from low level synthesis of Verilog and VHDL hardware description, to the higher level toolchains that compile real programming languages like SystemC OpenCL and others [7,8].

In this paper we present yet another choice named *Anacleto* (Another auto config for logic evaluation toolchains), particularly targeted to the GNU Linux embedded devices, that has been developed by RFX consortium and aims at proposing a unified standard workflow to the FPGA developer for programming both the logic and the software components in a uniform and portable way.

It is worth noting that in the development of Anacleto SoC projects the knowledge of HDL, unlike other solutions such as the National Instrument RIO LabView interface, is not hidden by the framework. The aim of this framework is indeed not to provide a new programming interface, adding another layer of logic, but to ease the development process with established well known open-source build tools. In this way Anacleto can be a way to access the low level machinery of the FPGA programming easily and uniformly, and a much cheaper solution in respect of RIO. Moreover, at this low abstraction programming level, many of the features that are usually involved are already provided free of charge by the chip vendors or with a reasonable license fee by external contributions, keeping a door open to a wide market of existing solutions.

The first candidate applications for SoC devices are timing systems, data acquisition preprocessing and fast computation. Timing systems represent a classical field of applications for FPGAs and have been implemented both in custom systems [9] and commercial products [10]. A typical timing application uses a synchronization clock signal distributed, normally via fiber optic, to all the timing devices and possibly propagating asynchronous events. The FPGA provides the generation of the required timing signals (clocks, triggers, …) based on current configuration loaded in the system using some kind of hardware interface such as PCI. A processor would introduce in this case more flexibility in the management of the configuration, letting, for example, the configuration be uploaded via the network.

Integrating configurable FPGAs in data acquisition would provide much more flexibility in data management introducing features not currently supported by ADC devices. An example is the possibility of managing deferred triggers communicated via network. Using the network to communicate triggers in data acquisition introduces delays that may compromise the precision in the reconstruction of the acquired signal. However, if a trigger message also carries the exact trigger time, and assuming that all devices have a precise knowledge of time (e.g. using IEEE 1588 timing protocol), it is possible to provide a correct reconstruction of the signal using an internal circular buffer maintaining a signal history lasting at least the delay in trigger communication [11]. The use of a configurable FPGA in data acquisition could also allow a significant reduction of the required front end when integrated signals from electromagnetic probes are acquired. In this case it would be possible to avoid analog integration before data acquisition moving integration to FPGA processing during acquisition.

Fast computation carried out by FPGA allows using more sophisticated algorithms in real-time plasma control retaining at the same time the flexibility provided by a computer system. The same approach could be used for new data processing algorithms such as feature detection from acquired video frames. In this case the processor would supervise data transfer and the FPGA would carry out intensive computing for feature detection. It is worth stressing the fact that FPGA solutions are more difficult to develop in respect of CPU based ones, and therefore the latter is preferred, provided it can satisfy the required timing constraints. As a rule of thumb, CPU based solutions should be considered when the order of magnitude of the required reaction time of the system is 100 μs or larger. Shorter times normally require FPGA implementations, however other factors may affect the choice, such as memory access issues that may reduce performance regardless the computational power, as happens also in large distributed computation carried out by General purpouse Graphical Processor Units (GGPUs) [12].

As for other FPGA solutions, SoC systems require skills and experience. For example, developing Zynq based applications requires (1) the development of the FPGA logic, (2) mapping information between the FPGA device and the processor address space, (3) developing the kernel driver for interfacing user software and the FPGA device and (4) developing the high level software applications in user space for the supervision of the system and its integration in the central CODAS. For this reason we have implemented a framework that integrates the above steps. The framework, described in the next section, makes the overall process easier, especially the integration of the FPGA components and the processor by coordinating all the required tools and by providing a set of templates that can be adapted to the specific application.

2. Framework components

Anacleto uses the Autotools [13] build infrastructure to organize the most general FPGA workflow acting like a standard toolchain compilation led by GNU make targets. The development process remains quite complex because many components in the final device board must be orchestrated (i.e. the kernel configuration, the customization of drivers to handle the newly created device, and so forth) but nevertheless the compilation is managed almost in automatic manner and, once the project is properly defined, all the steps are covered by Makefile targets that can be chained in a single make run. In order to develop a SoC application, it is necessary firstly to select the hardware system. Because we decided to make use of the Xilinx Zynq devices, as a first attempt, three low-cost solutions have been considered: RedPitaya [14], ZedBoard [15] and Parallella [16]. RedPitaya is intended to be used as a stand-alone system for handling digital and analog I/O signals. This board hosts ready to use ADC and DAC components and therefore could result best suited for developing small self-contained applications, but for the same reason it shows a reduced flexibility in respect of the other two for the configuration of the I/O pins. The other boards are intended to be hosted in a carrier board and therefore mount no additional I/O devices. In particular, Parallella is targeted towards computing intensive applications and hosts an additional processor with 16 cores.

Several other software components, all free of charge, are required for developing a SoC application an deploying it into the target board. First of all, it is necessary to download from XILINX the Integrated Development Environment (IDE) tool VIVADO for HDL programming (Verilog and VHDL are the supported languages). In order to be used on a specific target, VIVADO requires a target-specific configuration, provided by the board developer, which specifies how the processor is configured in that particular board. Currently only Red Pitaya configuration is managed in the framework, but it is foreseen that configuration files from Zed Board and Parallella will be included, adding the choice of the target board in the configuration steps. VIVADO provides a set of configurable Intellectual Property (IP) components that carry out the connectivity between the processor (dual core ARM Cortex A9 in the Zynq chip mounted on Red Pitaya) and the FPGA application. When no DMA is involved, communication between the processor and the FPGA application is carried out by a configurable number of 32 bit registers and, optionally, one or more interrupt lines. When the developer creates a new project for a FPGA application, the IDE creates a set of IP components, carrying out the handshaking with the internal