HIGH LEVEL FPGA PROGRAMMING FRAMEWORK BASED ON SIMULINK*

B. Fernandes†, P. Gessler, C. Youngman, European XFEL GmbH, 22761 Hamburg, Germany

Abstract

Modern diagnostic and detector related data acquisition and processing hardware are increasingly being implemented with Field Programmable Gate Array (FPGA) technology. The level of flexibility allows for simpler hardware solutions together with the ability to implement functions during the firmware programming phase. The technology is also becoming more relevant in data processing, allowing for reduction and filtering to be done at the hardware level together with implementation of low-latency feedback systems. However, taking advantage of the flexibility and possibilities offered require significant amount of design, programming, simulation and testing work usually performed by FPGA experts.

A high-level FPGA programming framework is currently under development at the European XFEL in collaboration with the Oxford University within the EU CRISP project. This framework allows for people unfamiliar with FPGA programming to develop and simulate complete algorithms and programs within the MathWorks Simulink graphical tool with real FPGA precision. Modules within the framework allow for simple code reuse by compiling them into libraries, which can be deployed to other boards or FPGAs.

INTRODUCTION

Located in Hamburg, Germany, the European X-Ray Free-Electron Laser facility (European XFEL) will generate intense ultra short coherent X-Ray flashes for scientific applications. These short bursts last for 600 µs and have a 4.5 MHz repetition rate [1]. Bandwidths of 10 GBytes of data per second are expected from 2D pixel detectors currently being develop, while other detector types, like systems based on fast digitizing and analog-to-digital converters, can go up to 60 MBytes.

Considering this scenario, the European XFEL requires a hardware framework which can cope with the expected data bandwidth while offering flexibility and scalability. High-speed data throughput and processing units are necessary to collected and reduce the amount of data to be store.

The main Hardware platform is based on MicroTCA.4. It offers high bandwidth communication between boards and CPU via PCI Express (PCIe) and between boards via point-to-point communications. Processing of data occurs in CPUs, DSP and Boards/Digitizers with FPGAs [2].

FPGA technology offers a high level of flexibility. Algorithms can be continuously upgraded without having to change the hardware. The code can be reused and easily integrate into different projects.

HIGH-LEVEL HARDWARE PROGRAMMING TOOLS

Proprietary solutions for graphical high-level FPGA programming are available on the market, the most popular ones being Simulink from MathWorks [3] and LabVIEW from National Instruments [4]. Both frameworks offer a graphical programming language for modeling, simulating and analyzing a multidomain dynamic system, making them an appealing alternative to Hardware Descriptive Languages (HDL). Xilinx libraries are available for the Simulink framework which include a considerable number of modules optimized for Xilinx FPGA boards and incorporate the generation of ISE projects based on the users module [5]. Recently, MathWorks introduced the HDL Coder, a feature that allows for generation of synthesizable HDL code from Matlab functions which is compatible with any FPGA [6].

Focused on signal processing algorithms, System Studio from Synopsys presents a model-based design tool that offers an extensive library of advance protocols and standards for wireless and digital signal processing domain [7]. The designs can be exported as C/C++ functions or as SystemC modules, the latter compatible with any hardware.

Xilinx also offers the possibility of schematic-based design, where the top-level design consists of a schematic sheet that refers to several lower-level macros. These may be schematic-based modules, CORE Generator software modules or HDL modules.

However when starting a new design, in either of the previous frameworks, it still requires that the user provides information such as Pin placement and clock constraints to achieve optimal performance. In particular, on the previous two frameworks, users need to be familiarized with other software frameworks (such as ISE) in order to synthesize their code and program the FPGA.

Other solutions generally involve a wrapper of some sort around a HDL language or a language package whose code can be converted to synthesizable HDL code [8, 9]. This approach makes development more accessible for users familiar with other programming languages at the cost of losing some of the features that FPGA programming languages provide and development of a custom compiler to generate synthesizable code.

A more suitable solution to our applications is being develop by the CASPER project [10]. The framework is based on the Simulink tool and uses custom libraries. Libraries consist of algorithms and board specific blocks

* Work partially supported by European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement nº283745.

† Bruno.fernandes@xfel.eu

ISBN 978-3-95450-139-7

that implement the features available for that board. In this format, a complete FPGA project is developed within Simulink. Users simply add all required board blocks needed to develop their algorithm and the Xilinx Simulink tools translate the final design to an ISE project. Integration of user defined registers was achieved with use of Xilinx EDK. The drawback of the CASPER framework is that the available libraries are compatible only to the boards being used on the project. The development of these blocks is reported to be very time consuming.

FPGA PROJECTS STRUCTURE

Top level FPGA projects on the European XFEL are developed in VHDL and separated in two specific modules: Board and Application (see Fig. 1). The Board module includes all code related and specific to configure and monitor the Board features and also provides the Board interfaces (e.g. ADC/DAC, DDR2 controller, Clocks, etc.) to the Application module. The latter implements user algorithms and applications which may be Board specific. This fixed structure allows for easy porting and update of both board and application code. It provides an environment for firmware programming for standardized and future hardware.

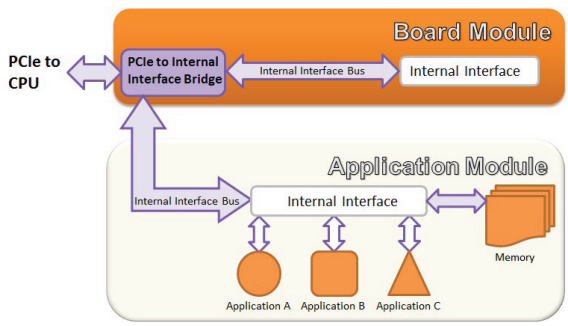

Figure 1: FPGA projects structure.

Since the basic Input/Output (IO) Board Interface are provided on the Application module, users can focus on the development of their application. Expert designers can continuously improve and further develop the functionalities available on the Board while not interfering with developments on the Application module.

Projects include an Internal Interface bus to define user registers and memories which can then be access by PCIe devices. The Internal Interface is a bus protocol which includes a VHDL API package that eases register access and definition. Register properties such as access rights, type (internal or external bus register), bit size, data type (unsigned, signed), number of integer and fractional bits, memory length together with a brief description are defined in VHDL table types. Assignment of address and generation of internal bus registers are preform automatically during the compilation process. This allows register definition to be preformed at the hardware level. An XML file with all register information, including the assigned addresses, is available after compilation. The Karabo software is able to interper this file and provide a user friendly environment which allow users to communicate with the design on the FPGA [11].

Application Development

FPGA programming is time intensive and requires specialists, however most applications and algorithms are conceptualized by users unfamiliar with FPGA programming.

For the European XFEL, a high level FPGA framework is being developed that allows for users with no prior HDL knowledge to develop their algorithm modules which can be integrated in a top VHDL project. The framework should allow user defined registers and memories compatible with the Internal Interface protocol and to port applications to different projects.

The framework being developed is based on the Simulink tool. Use of a graphical programming language allows for structured algorithms modules, reusability of existing blocks, while making debugging, simulation and development much faster. Integration of Simulink with Matlab also allows for powerful test environments, namely the usage of real data and integration of processing algorithms developed in other programming languages in simulation, time and frequency domain analysis as well as detail signal analysis. Currently, all boards designed for the European XFEL incorporate Xilinx FPGAs, making the available Xilinx Simulink library for the tool an asset.

FRAMEWORK WORKFLOW

The user design environment should be automatically setup for the target board and blocks that simulate the behavior of available features must be available so that algorithms and applications are develop within realistic scenarios. The framework must allow for user defined registers and memories and generate the necessary logic to later communicate with the bus protocol.

Simulink based applications should be included on the Application Module of a FPGA project. The interface is fixed by the Application Module IO since all features must be available together with the necessary bus protocol ports. Our FPGA project structure abstracts the user from the integration of his module in the top level module since all the require information concerning Pin placement, clock routing and Board features interface are already defined on the FPGA top level project.

Board Definition

All projects begin with the integration of the Project Definitions block in the design. With this block the user defines for which board the algorithm is going to be develop as well as the bus protocol (see Fig. 2).

The block will generate the IO available for the chosen board and examples of expected input signals. The user can remove IOs which are not suitable for the algorithm being development and reinsert them later if necessary.

ISBN 978-3-95450-139-7

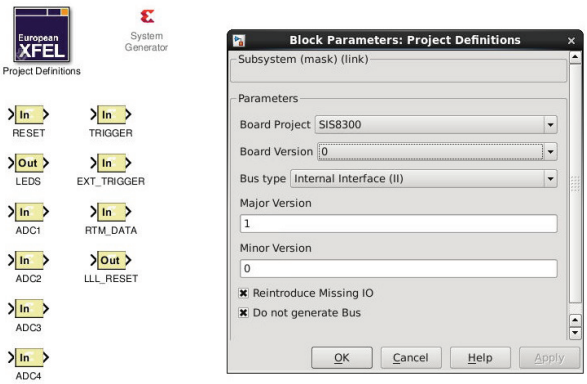

Figure 2: Board definition in a Simulink project.

All Xilinx based algorithms in Simulink need to have the System Generator block, which defines the target FPGA as well as clock frequency constraints. The Project Definitions block automatically includes the System generator block and defines the previous parameters according to the board.

User Registers and Memories

The BUS block allows for users to define registers and memories which are later accessible by the chosen protocol (see Fig. 3). While developing the algorithm, user registers are treated as IO ports, abstracting the user from hardware concepts such as write and read cycles, address indexing and data latching. The data saved in memories is also accessible in the Matlab workspace.

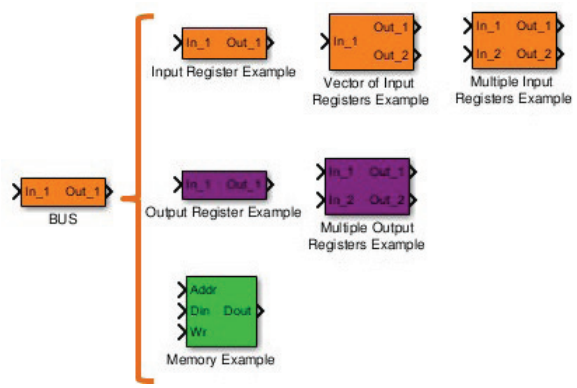

Figure 3: BUS block example. The block regenerates according to the parameters defined by the user.

The BUS block has a set of parameters which are used to regenerate the block properties according to the values specified.

Board Specific Blocks

Specific blocks are available on the library which will accurately simulate the behavior of features available on the board.

As an example, in the European XFEL a Low Latency Protocol (LLP) is used to distribute machine experimental related data for feedback and VETO type systems. This protocol is used on different interfaces such as backplanes and SFP connectors. The XFEL Simulink library includes a Transmitter and a Receiver LLP block whose behavioral is identical to that expected on the FPGA. When included, the block recognizes for which board the algorithm is being developed and presents the different interfaces available for that board as well as number of connectors. Users can easily simulate the transmission of data with this protocol and incorporate it in their algorithm.

Algorithm Integration

Once development of the application is completed, the whole design must be processed to integrate the top level FPGA project. A separate Simulink design is generated based on the user algorithm with user defined registers and memories being replaced with suitable blocks which are compatible with the chosen bus protocol. A netlist file of this final design is then generated and inserted on the ISE project. The resulting bit file is then used to reprogram the target FPGA. All the previous steps will be done automatically, the user needs only to inform the framework that the algorithm is ready to be compiled. Currently, the user has to manually add the generated Simulink files into the ISE project in order to generate the final bit file.

In the case of the Internal Interface, which normally works in a different clock domain than the algorithm, each user register/memory is replaced with a Multiple clock domain register/memory. A separate module assigns different addresses to each register and generates the necessary logic to communicate with the bus protocol while taking into account the information of each register (such as bit size, read and write access, memory length, etc.). The user is also given the option to generate a standalone module of his algorithm in order to include it on the XFEL Simulink library. User register definitions are also compiled in a XML file to be interper by the Karabo software framework.

Figure 4 shows the workflow of a simple example develop in this framework.

FURTHER DEVELOPMENT

As of now, the framework supports user defined registers and memories compatible to the Internal Interface protocol. It is planned to integrate additional protocols, namely Ethernet, Wishbone and UART. This will allow the framework to be flexible and compatible with other boards and projects outside of the European XFEL, with developed algorithms being compatible and accessible to a larger user community.

Additionally partial reconfiguration in the FPGA projects structure, i.e., the ability to dynamically modify blocks of logic by downloading partial bit files while the remaining logic continues to operate without interruption [12] will be incorporated. This process reduces synthesis time and drastically enhances the flexibility of FPGAs.

ISBN 978-3-95450-139-7

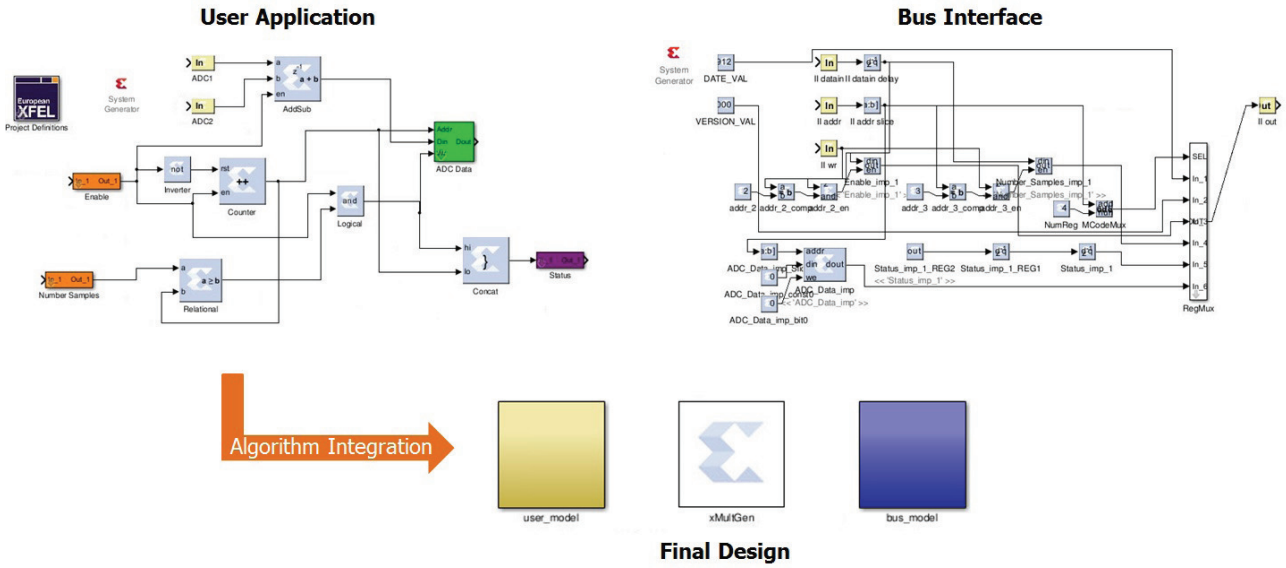

Figure 4: Design example. The Bus Interface logic is generated during Algorithm Integration based on user defined registers. The bus_model block includes the Bus Interface logic while a copy of the original algorithm is wrapped on the user_model block.

CONCLUSION

The described high-level FPGA programming framework allows users unfamiliar with FPGA programming to easily develop, simulate and integrate their algorithms into large FPGA projects.

Current European XFEL FPGA projects are already being develop in this framework with positive results. Users with basic digital electronic knowledge were able to develop entire algorithms and successfully test and integrate in existing FPGA projects.

REFERENCES

[1] M. Altarelli et al., "XFEL: the European X-ray Free-Electron Laser technical design report", DESY XFEL Project Group (2006)

[2] C. Youngman et al., "Electronics Developments for High Speed Data Throughput and Processing", TUPPC086, ICALEPCS-2013, to be published.

[3] MathWorks Simulink, http://www.mathworks.com/products/simulink

[4] NI LabVIEW, http://www.ni.com/labview

[5] Xilinx ISE, http://www.xilinx.com/products/design-tools/ise-design-suite/index.htm

[6] MathWorks HDL Coder, http://www.mathworks.com/products/hdlcoder

[7] Synopsys System Studio, http://www.synopsys.com/Systems/BlockDesign/DigitalSignalProcessing/Pages/SystemStudio.aspx

[8] MyHDL, http://www.myhdl.org.

[9] Milkymist Labs, "Migen" http://www.milkymist.org.

[10] Collaboration for Astronomy Signal Processing and Electronics Research (CASPER), http://casper.berkeley.edu.

[11] B. C. Heisen et al., "Karabo: An Integrated Software Framework Combining Control, Data Management and Scientific Computing Tasks", FRCOAAB02, ICALEPCS-2013, to be published.

[12] Xilinx Partial Reconfiguration, http://xilinx.com/tools/partial-reconfiguration.htm.

ISBN 978-3-95450-139-7