A MULTI-LAYERED XML SCHEMA AND DESIGN TOOL FOR REUSING AND INTEGRATING FPGA IP

Adam Arnesen, Nathaniel Rollins, and Michael Wirthlin

NSF Center for High-Performance Reconfigurable Computing (CHREC)
Dept. of Electrical and Computer Engineering
Brigham Young University
Provo, UT, 84602, USA
adamarnesen@ieee.org, nhrollins@gmail.com, wirthlin@byu.edu

ABSTRACT

Reconfigurable computing systems remain difficult to use and program. One way to increase design productivity for these systems is through reuse of previously developed and verified intellectual property (IP) cores. This paper presents CHREC XML, a XML schema that facilitates IP reuse by encapsulating the details of reusable IP cores at multiple levels of abstraction. This schema is independent from any design language or tool and can be used by any tool to understand many details about the interface of a reusable circuit. An IP integration tool was also created based on this schema to demonstrate the ease of IP reuse when cores are described in this meta-data description. This IP integration tool allows a designer to easily select and integrate IP cores from a variety of languages/tools and automatically run the appropriate tools to generate the cores in a form usable by downstream implementation tools.

1. INTRODUCTION

There is growing interest in using reconfigurable platforms such as FPGAs to perform application-specific computing. Despite the interest in reconfigurable computing, low design productivity is still a major limitation to its more widespread adoption. Design productivity could be significantly improved through proper reuse of previously developed highly parameterized intellectual property (IP) Cores [1]. The successful application of reuse within reconfigurable computing has the potential to significantly improve design productivity.

This paper presents the CHREC XML schema and IP integration tool which facilitate the reuse of FPGA IP cores from a variety of languages and environments. This tool provides a user interface to structurally interconnect IP cores, automatically synthesize wrappers for several languages and tools, and automatically run the various tools needed to synthesize and interconnect the IP into a single design. CHREC XML and its IP integration tool simplify the process of reusing IP from a variety of languages and tools.

This work was supported by the I/UCRC Program of the National Science Foundation under Grant No. 0801876.

2. IP REUSE

Reuse of IP can be difficult and time consuming. Reusing a digital circuit requires the designer to: 1) Select the appropriate circuit core, 2) understand the details of the core, 3) create interface circuitry to integrate the core into the system, and 4) verify the core within the system. In order to make reuse a viable solution, these needs must be overcome and reuse costs must not exceed 30% of the cost of creating the same core from scratch [2].

Another challenge to core reuse is the variety of languages and tools used to create high-performance cores. Reusable cores are commonly written in VHDL, Verilog or other languages. There are also a variety of tools for distributing IP cores (i.e., Xilinx coregen, Altera MegaCore, JHDL, etc.). While these tools facilitate the delivery of IP cores, it can be difficult to integrate cores from such a wide variety of sources.

3. IP-XACT

An emerging IP reuse strategy is the IP-XACT standard from The Spirit Consortium [3]. IP-XACT defines an XML schema for describing reusable circuit cores in a vendor neutral manner. Targeted primarily for System-on-Chip (SoC) design, IP-XACT defines the busses, ports, configuration, and properties of a reusable core to facilitate core reuse at a high level. IP-XACT enabled tools allow designers to drag-and-drop complex IP into a design and automatically use third party tools to generate and verify SoC designs.

The IP-XACT schema provides many basic XML elements that are useful for describing any reusable hardware core. Several of these are as follows:

Component Naming: A component naming element defines component names and is used to differentiate between cores. It enables the user to easily find the desired core in a large library.

Port Information: The port element defines ports and their parameters. It includes port naming, bitwidths and underlying HDL types, e.g. `std_logic_vector` etc., and port direction.

File Sets: File sets associate external files such as HDL or software with the meta-data core description.

Component Generators: Generators are the interface which allows IP-XACT enabled design environments to communicate with and use external tools and contain information needed to access external tools.

These and other XML elements provide enough information about a reusable core to enable higher level SoC design tools to identify, use, and integrate the core into a larger SoC design.

4. UNIQUE DESCRIPTION REQUIREMENTS FOR RECONFIGURABLE COMPUTING IP

While the IP-XACT schema appears to be well suited for SoC design, the schema does not adequately address the description needs of IP for reconfigurable computing. Reconfigurable IP tends to have many interrelated parameters and therefore any meta-data description of this IP must have strong support for parameters and parameter evaluation. Reconfigurable IP are also usually more fine grain than SoC IP and they use ad-hoc, IP specific communication interfaces rather than predefined bus interfaces. Any description for RC IP needs to support arbitrary communication interfaces. This section will address these unique description requirements of reconfigurable computing IP.

4.1. Parameterization

Parameters are an essential part of reusable IP cores. The flexibility that comes with parameterization enables a single core to represent many non-parameterizable (static) cores. Therefore, a core with more parameters can be used in more design situations than a core with fewer parameters. While highly parameterizable cores are more difficult to create, they are much more reusable that static cores.

An important part of an XML schema for reusable cores is strong support for such parameterization. It should support a variety of ways of resolving core parameters. Parameter values can be resolved in one of five ways: 1) statically, 2) by another parameter, 3) by the result of a mathematical expression, 4) by a third party tool, or 5) by a designer.

4.2. Interface Descriptions

RC cores often have fine-grain custom interface protocols and do not usually rely on coarse grain bus interfaces for communication. Several additional XML elements can improve the description of interfaces used by RC cores.

Data Types: The data type of ports on a core is important information for high level tools. This data type information should include not only the low level types, such as `std_logic_vector` for VHDL, but should also include the high level type of the interface port. These high level types should include types such as integer, floating point, fixed point, character, and boolean. In addition to standard types there may be custom types such as a custom floating or fixed point format. High level types should also include information that maps the understanding of the type to the bit level representation. When interconnecting cores automatically, the tool must be able to match compatible types and convert between incompatible types to avoid data corruption.

Protocol Information: In order for a compiler or high level synthesis tool to be able to compose designs, robust interface behavior information is also required. This information should include the relative timing and sequencing of signals, the statefulness of the core, the latency of data, and the function of core control signals. Several studies have investigated methods for describing interfaces including regular expressions[4] and finite automata or finite state machine descriptions[5].

5. CHREC XML: A LAYERED XML SCHEMA FOR RECONFIGURABLE COMPUTING IP CORES

Prior to the CHREC XML presented in this paper, another XML schema was created for representing IP cores in a reconfigurable computing environment [6]. This initial schema addresses some of the limitations of IP-XACT and better supports FPGA IP used in reconfigurable computing. It also provides extensive support for parameterization, mathematical expressions, high-level data types, and tool generators. This initial schema, however, is very bulky and became too difficult to manage and manipulate.

CHREC XML was created to simplify and refine the initial attempt. It organizes the core meta-data into several distinct layers of abstraction: the RTL level, the data type level, and the interface level. Organizing IP meta-data in layers of abstraction allows IP core providers to support the integration of IP at different levels of abstraction. For example, low level tools such as a netlisting tool may require only low-level information such as port naming and bitwidths. High level synthesis tools, however, are better served with a more abstract, higher level view of the interface. Datatypes and timing information are also important to a high level tool.

CHREC XML has three levels. The first level represents the RTL level of abstraction and describes the low-level details of an IP core such as port naming, bitwidths, and low-level parameterization. The second layer describes the high level data types and their associated signals. The third layer is a behavioral description currently under development which describes the timing and high level operation of the core's interface.

5.1. Layer 1: RTL Layer

The RTL layer describes the low-level details of an IP core and is very similar to the initial schema described in [6]. It includes the naming of ports and a mapping of these ports to the actual ports listed in HDL. It further includes a list of parameters for the core and the mathematical expressions and enumerated values that these parameters may depend on. This layer also includes a list of files required to simulate or synthesize the core. This layer is especially useful to the low-level synthesis and simulation tools whose primary activity is the structural interconnection of cores and the mapping of them into the RC platform. The IP integration tool described in this paper relies heavily on this layer.

5.2. Layer 2: Data Type Descriptions

This layer adds additional information to the circuit ports defined in Layer 1 by providing high level types. The low-level bit-types used for ports are often actually interpreted as a higher level type such as string, integer, floating point, fixed point, character, and boolean. These types can be parameterizable at the bit-level (i.e., a user specific fixed-point type). This level of CHREC XML at defines several parameterizable standard types and allows user definable custom types.

This datatype is useful to a tool which reasons about the details of actually wiring cores together. The data typing of signals allows the tool to correctly match bits from one signal to another as well as do any needed conversions of data types.

5.3. Layer 3: Interface Operation Information

This layer will describe the high level operation of the interface of the core. The selection of which type of representation to use as well as the best way to implement it in an XML description is yet to be developed but may be done several ways including: regular expressions, FSM descriptions, timing diagrams, or action-based interfaces.

This layer is removed from the details of the ports and the actual implementation of the core and therefore allows a tool to reason about the relative timing of signals, data dependencies, latencies of signals, and other information required for high level interface synthesis.

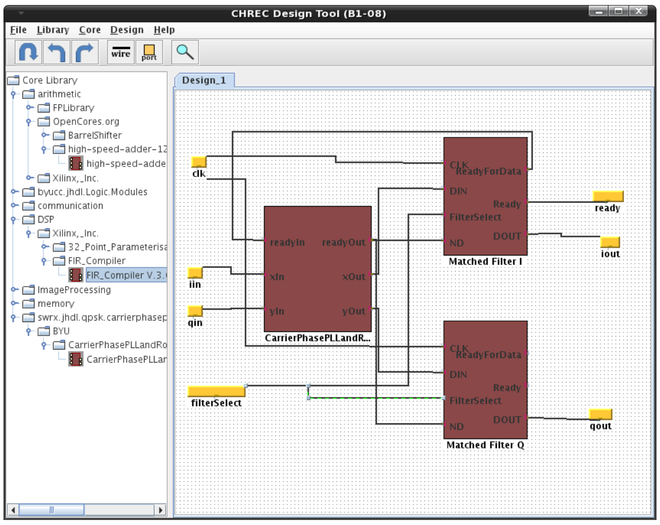


Fig. 1. The GUI tool demonstrates the ability of CHREC XML to standardize descriptions for cores from multiple environments and enable them to communicate.

6. IP INTEGRATION AND REUSE TOOL

A structural design tool was created which demonstrates the ease of integrating and reusing IP with CHREC XML. This tool allows cores from diverse environments and languages to be instanced, parameterized, and interconnected in a graphical user interface. The tool is not directly dependent on any language or specific design environment. Any IP core may be instanced and integrated as long as the core is completely described in CHREC XML. This tool generates custom wrappers for each core, integrates the cores in a top-level design, calls the appropriate core-specific tools, and finally calls the FPGA implementation tools to generate the FPGA bitstream.

6.1. Design Composition GUI

The demonstration tool provides a simple graphical environment, shown in Figure 1, in which the designer can instance and connect reusable cores. In this example, the library consists of cores generated from several tools including the Xilinx CoreGen tool [7], ImpulseC, JHDL[8], System Generator, System Verilog, Verilog, and VHDL. The designer, however, does not need to be aware of any of the language specific features of the core. The designer can use *any* core and connect it to any other core in the library as long as the core appropriately described in CHREC XML.

6.2. Core Parameter Manipulation

After an instance has been created for a core, the designer can "open" an individual instantiated core and edit the parameters that exist for the core as shown in Figure 2. This example shows the interface and parameters of a "filter" core. The component GUI and parameter input forms are generated *automatically* from the underlying XML description of the core. As the designer manipulates the parameters, the underlying CHREC XML parameter and mathematical expression representation are used to ensure that the desired combination of parameters is valid.

6.3. Wrapper Generation

Once the structure is defined in the GUI and the core parameters are correctly set, the integration tool can begin the process of creating a top-level integrated design representation. In this demonstration tool the structural representation of the top-level design is represented in VHDL (although any structural netlist could be used). Because many of the cores are not represented in VHDL, VHDL wrappers are also generated to represent the structure of these cores in the top-level design.

6.4. IP Integration

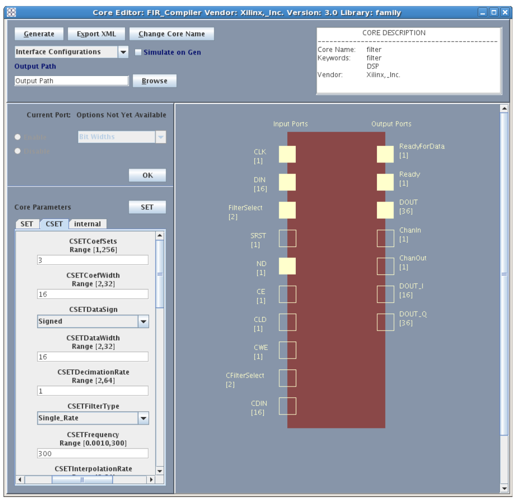
After the VHDL wrappers have been generated for each of the cores, the design is ready for final generation and implementation. A top-level VHDL file is created for the design which interconnects the cores and connects the appropriate top-level ports. The XML meta-data is queried to determine which external tools are needed to generate individual cores. After the appropriate tools have been run, the overall design is synthesized and a downloadable bitstream produced.

7. CONCLUSION

CHREC XML was created to represent important meta-data of reusable cores to improve design reuse. It is similar to the IP-XACT schema used for SoC design but it addresses some of the unique needs of reconfigurable computing. CHREC XML is organized in layers to facilitate the ability of tools to reason about the IP at different layers of abstraction.

CHREC XML facilitated the creation of a structural IP integration tool which simplifies the process of integrating cores from a variety of tools/languages. Any core described in this new XML can be instanced and integrated no matter where the core was generated. The tool developed in conjunction with CHREC XML can manipulate diverse cores in a common environment, create simple designs and generate downloadable bitstreams from these designs. A tool such as this increases design productivity by removing the need for the designer to understand the low level details of the core.

The ability of this tool to use, instance, and manipulate cores in a language and vendor independent manner is extremely useful when promoting reuse. CHREC XML provides the necessary information to facilitate reuse of cores in


Core Editor: FIR_Compiler Vendor: Xilinx,_Inc. Version: 3.0 Library: family

CORE DESCRIPTION

Core Name: filter
Keywords: filter
DSP
Vendor: Xilinx,_Inc.

Output Path: Output Path

Current Port: Options Not Yet Available

Core Parameters

CSETCoefSets Range [1,256]: 3
CSETCoefWidth Range [2,32]: 16
CSETDataSign: Signed
CSETDataWidth Range [2,32]: 16
CSETDecimationRate Range [2,64]: 1
CSETFilterType: Single_Rate
CSETFrequency Range [0.0010,300]: 300
CSETInterpolationRate

Fig. 2. Each core instance can be individually opened and its parameters modified to fit it to a particular use.

high level synthesis as well as information needed to automatically generate complex interfaces between cores. These advances will lead to greater increases in productivity and improve the ability of designers to rapidly develop and deploy systems for reconfigurable computing.

8. REFERENCES

[1] *International Technology Roadmap for Semiconductors 2005 Edition*, International Semiconductor Industry Association, 2005.

[2] R. Passerone and J. A. Rowson, "Automatic synthesis of interfaces between incompatible protocols," in *Proceedings of the 35th Design Automation Conference (DAC 1998)*, June 1998, pp. 8–13.

[3] *IP-XACT v1.4: A specification for XML meta-data and tool interfaces*, SPIRIT consortium, 2008.

[4] A. Seawright and F. Brewer, "Clairvoyant: a synthesis system for production-based specification," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 2, no. 2, pp. 172–185, 1994. [Online]. Available: http://dx.doi.org/10.1109/92.285744

[5] V. D'Silva, A. Sowmya, and S. Ramesh, "Automated interface synthesis," University of New South Wales. School of Computer Science and Engineering, Tech. Rep., September 2003. [Online]. Available: http://www.worldcat.org/oclc/224267365

[6] N. Rollins, A. Arnesen, and M. Wirthlin, "An XML schema for representing reuable IP cores for reconfigurable computing," in *Proceedings of the National Aerospace and Electronics Conference (NAECON 2008)*, July 2008.

[7] *CORE Generator Help*, Xilinx, Inc., 2007.

[8] P. Bellows and B. Hutchings, "JHDL - An HDL for reconfigurable systems," in *IEEE Symposium on FPGAs for Custom Computing Machines*, 1998, p. 175.