Extensible IP-core library for High-Level Synthesis

Ivan Grigorov[1,2]
[1] Ivannikov Institute for System Programming of the Russian Academy of Sciences (ISP RAS)
[2] National Research University – Higher School of Economics (HSE)
Email: grigorovia@ispras.ru

***Abstract*—There is a problem of hardware component integration designed by different vendors. To address this problem, IP-XACT standard has been introduced. It suggests representation of the components' key features by means of XML descriptions, and is applicable for different components types. This paper aimed to solve the problem of the components integration during the process of the High-Level Synthesis (HLS) of hardware descriptions. We propose to annotate the HLS library components with the corresponding IP-XACT specifications (partially, automatically), which keep information about the components interfaces and, in some cases, the way of how to gather the given component by means of the external sources. In general, the proposed technique allows structuring the HLS library, also making it independent from the current HLS tool, and, as a result, improving the HLS results.**

Keywords*—*High-Level Synthesis, IP-XACT, IP, integration, XML, Electronic Design Automation, hardware development, hardware design.

I. Introduction

Hardware designs are continuously increasing in complexity. The conventional manual development of the designs is a difficult and error-prone process, which requires the knowledge of hardware description languages (HDL). There comes the concept of High-Level Synthesis (HLS). HLS is an automated design process that takes as input an algorithmic description in order to create the hardware design that implements the desired function. Typically, the algorithmic description is written in a high-level programming language such as C, C++, Python, etc. The process of HLS for a complex design includes using of sub-components of the target design. For instance, matrix multiplier uses adders and multipliers for the matrix elements. Therefore, there is a need for a library with such sub-components, which are, in fact, IP-cores (i.e., Intellectual Properties). Each IP-block represents the behavior, properties and/or description of some part of the complete future design. However, to integrate these IP-cores is not an easy task as they may have complex interfaces without machine-readable semantics. Besides, the IP-cores are provided by different vendors. So, there is a need for a standard solution, supported by HLS tool as well.

The purpose of this paper is to present a way to integrate IP-cores in HLS process. The first task is to identify a way to describe interface of IP-cores. The second task is to implement support of this description. The third task is to use information from the description to bind all these IP-cores together. The final task is to optimize the structure and to translate the result into HDL.

The rest of the paper is organized as follows. The second section touches upon similar approaches. The third section describes implementation of the proposed technique. The forth section summarizes the results.

II. Related work

There are a number of standards for IP-cores delivery. The most prominent of them are SystemRDL [1], Standard Universal Verification Methodology (UVM) [2], IP-XACT [3] and Accellera Portable Test and Stimulus Standard (PSS) [4].

SystemRDL has been developed since 2013. It specifies both hardware and software behavior. However, it is limited in scope — the main purpose of the standard is to describe registers only. Moreover, SystemRDL standard working group is currently inactive.

The Portable Test and Stimulus Standard (PSS) defines a specification to create a unified representation of stimuli and test scenarios to supply IP-cores with verification means. It has been developed since 2018. As its main purpose is verification, it is limited in scope.

UVM has been developed since 2011. As well as PSS, it is aimed to supply IP-cores with the verification means.

IP-XACT standard was introduced in 2014. It proposes description the meta-data of IP-cores, including information about their interfaces, in a standard way. IP-XACT also supports so-called *vendor extensions* to describe user-defined vendor features. It should be noticed, that a number of commercial Electronic Design Automation (EDA) tools use IP-XACT in their infrastructure, for instance Xilinx's EDAs [5], Magillem [6], Agnisys [7]. So, the only way for development of standard solution for IP-cores is seen, and it is the usage of IP-XACT.

There are some open-source EDA tools that use IP-XACT standard. Among these instruments Kactus2 [8] toolset should be mentioned. It is an open-source toolset for designing embedded products. The project is under active development. Kactus2 has various applications such as IP-packaging, creation of hierarchical hardware designs and integration of hardware and software.

III. Implementation

This work is part of an HLS-tool development, which is currently in the prototype stage. The implementation is written in the C++ language. It takes an algorithmic description as input and produces a Register Transfer Level

(RTL) model. Fig.1 shows architecture of the tool. In the first stage *Parser* analyzes and parses a program written on prototype *HIL* language to get the Intermediate Representation (IR) in the form of a graph where nodes represent atomic operations such as addition, multiplication, subtraction, etc. and arcs represents data dependencies between these operations.

In the second stage, the graph is scheduled and optimized by *Scheduler*. This component is using data from *Library*. *Mapper* implements the following steps. First it creates so-called *circuit* from the scheduled IR. *Circuit* consists of *main module*, *external modules* and their interconnection. *Main module* interconnects *external modules*, which correspond to atomic operations mentioned earlier. Then it creates CIRCT [9] FIRRTL [10] dialect model for the main module and Verilog [11] RTL models for the nodes in the aforementioned graph from the *circuit* using *Library*. CIRCT stands for Circuit IR Compilers and Tools. CIRCT is an experimental effort to apply LLVM [12] and MLIR [13] development methodology for the development of hardware design tools. Finally, mapper optimizes the result using *circt-opt* utility provided by CIRCT infrastructure and translates the result to Verilog.

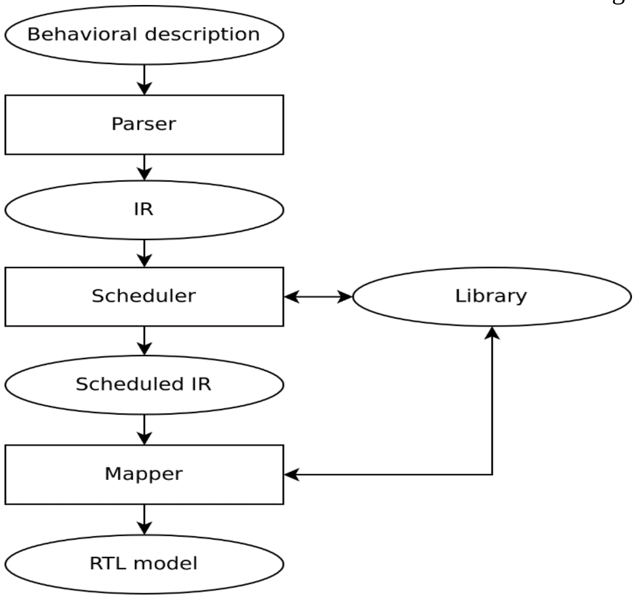

Fig. 1. HLS-tool architecture

Fig.2 shows the proposed architecture. In the first step of the improvement *IP library* is created. The library consists of RTL models of standard components such as adders, multipliers, etc. Then these components are provided with IP-XACT specification. An IP-block with the corresponding IP-XACT specification is called IP-XACT enabled IP [14]. These components form IP-XACT library. IP-XACT specifications can be written manually or generated using specific EDA tools. The manual process is tedious and error-prone. So, the automation is preferable. For this purpose, Kactus2 toolset has been chosen. The part of the project under consideration begins after producing an IR of an input program. In proposed architecture *Scheduler* and *Mapper* get component data by parsing the corresponding IP-XACT specifications. The parsing is done using open-source xerces-c++ parser [15]. For the present, only a subset of IP-XACT is supported, including the following IP-XACT structures: *component* and *catalog*. *Component* structure describes interfaces of an IP such as parameters, registers and ports. *Catalog* documents the location of IP-XACT files. The result is used to create aforementioned *circuit*.

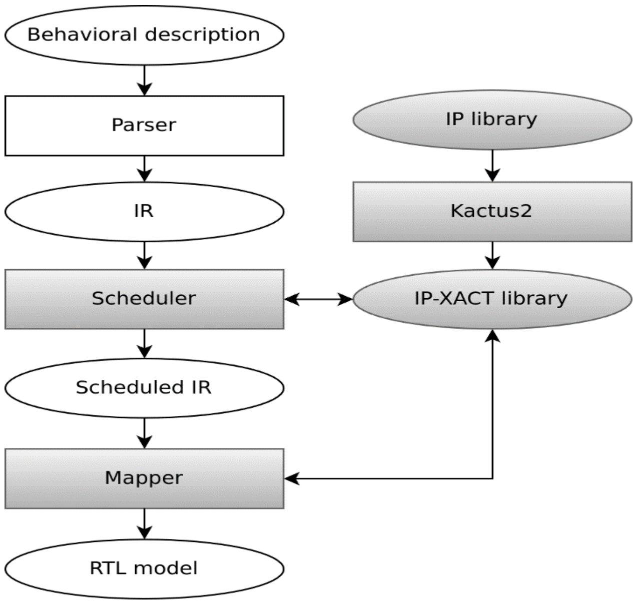

Fig. 2. The proposed HLS architecture

CONCLUSION

This paper presents a way to integrate IP-XACT infrastructure in development of complex IP-cores propagating IP-XACT standard for further use in EDA community.

The next step is to support of all IP-XACT structures. It is needed for the development of more complex designs. Also, IP-XACT library is intended to be dynamic, i.e. both an IP and the corresponding IP-XACT specification will be created by the query in completely automated fashion. Considering the fact that CIRCT is an open-source project, some specific optimizations may be added.

REFERENCES

[1] SystemRDL 2.0 Register Description Language. [Online] Available: http://www.accellera.org

[2] Universal Verification Methodology (UVM). [Online] Available: http://www.accellera.org

[3] IP-XACT IEEE 1685-2009 Standard. [Online] Available: http://www.accellera.org

[4] Portable Test and Stimulus Standard (PSS). [Online] Available: http://www.accellera.org

[5] Xilinx official site. https://www.xilinx.com

[6] Magillem tool. https://www.magillem.com/ip-xact-as-a-way-to-orchestrate-your-hw-design-flow

[7] Agnisys tool. https://www.agnisys.com

[8] Kactus2 toolset. https://github.com/kactus2/kactus2dev

[9] "CIRCT" / Circuit IR Compilers and Tools. https://circt.llvm.org

[10] Specification for the FIRRTL Language. [Online] Available: https://aspire.eecs.berkeley.edu

[11] IEEE Standard for Verilog Hardware Description Language. https://www.eg.bucknell.edu/~csci320/2016-fall/wp-content/uploads/2015/08/verilog-std-1364-2005.pdf

[12] The LLVM Compiler Infrastructure. https://llvm.org

[13] Multi-Level IR Compiler Framework. https://mlir.llvm.org

[14] IP-XACT IEEE 1685-2009 Standard. [Online] Available: http://www.accellera.org

[15] Xerces-C++ XML Parser. https://xerces.apache.org/xerces-c