Improving HDL Higher Level Logical Analysis Using Boolean Function Feature

Vladimir Zdraveski, Andrej Dimitrovski, Dimitar Trajanov

Abstract — Increased designers' interest in digital system design using hardware description languages has resulted in a huge data set of open source, available on the Web. Difficulties in discovering specific component introduce the need of automation in the process of search and reuse of already existing components. Despite the interface, a very important part required for a complete automation is the software analysis of the components' inner architecture. Applying the Semantic Web methodologies and using our existing hardware description ontology, we propose extension that will enable a semantic annotation of the inner architecture and will significantly improve the tools for automatic search and system composition of existing components. The ontology is published and can be used as a model for a standardized annotation, in order to increase the availability of the existing components and to provide easier reuse in novel designs. The concept is also applicable inside a company, to accelerate the retrieval through the local repositories of components.

Keywords — architecture, design reuse, linked data, ontology, system on chip.

I. Introduction

Hardware description languages (HDLs) are mostly used during the chip production process [1][2][3], but recently many programmable chips are embedded in production version of systems [4], as peripheral data adapters or configured as co-processors that distribute the processing power and reduce the load of the central processor [5][6]. This new approach explores a new era for the higher-level programmers to put their own logic in the programmable chip, getting them much closer to the HDL perspective, which they found very complex to do it from scratch.

We are also aware that a large bundle of HDL code is already published to many public web repositories. There are many web portals [7] containing projects available under open licenses (GPL, Apache, etc.). Although portals offer a basic classification of projects, however, finding a specific component is still very difficult and slow. Portals have no possibility for flexible search (filter by ports, type of component and so on.).

Reasons for the lack of electronic design automation tools for sharing and reusing the HDL code may be identified in the problem complexity, which mainly arises from the point where a software tool should determine the inner architecture of an HDL component.

Applying the main idea of the Semantic Web [7] [8], we explore a new approach towards a more granular (deeper) annotation of the HDL code/components and a higher-level of semantic knowledge [9] retrieved from it. Hardware description language has a certain structure, so it is possible to make automatic semantic annotation using ontology and software that will generate semantic resources for digital circuits.

Identifying the possibility of applying Semantic Web tools for the design of digital systems [10], we have implemented an early version of the HDL IP Cores system [11]. It performs automatic semantic annotation only for the interface of the components. The system consists of a web application and client plug-in [11] for Eclipse. Web application contains a Web robot that searches and downloads HDL components from the web and then each component passes through a process of automatic semantic annotation. Generated meta-data is stored in a semantic database (repository; storage). The system performs deeper analysis and mutual comparison of components and enables their ranking according to similarity and compatibility with a particular component. Client plug-in provides all functionalities of the system, directly into the designer's native development environment.

HDL IP Cores so far provide the automation in data collection and search functionality, but still there is a lack of deeper understanding of the architecture's type, which is crucial for any advanced search and component composition. Solving this problem, we have created a new module that will recursively annotate the instantiated components down to the level of basic logical blocks and thus retrieve the logical function of a given architecture. When available, the Boolean expression for a given architecture will be another property for component's matching and similarity determination process, leading to an improved EDA tool for sharing and reuse of existing HDL code.

II. Related Work

Semantic analysis of the logical circuit is extremely important in simulators and most of them provide many advanced functionalities. Among them, there is the working environment (desktop) Xilinx ISE with built in simulator ISim [12], which allows very precise and exact time execution of simulations, generating multiple ways of simulation (manual, graphically, through the terminal) and a range of formats in which you can save the result of the simulation. The simulator of Cadence [13] allows the verification, according to the well-known open methodologies OVM (en. "Open Verification Methodology") and UVM (en."Unified Verification Methodology"), enables fast and easy integration with various verification processes, as well as data level simulation (RTL), simulation of behavior, simulation with reduced consumption and etc. Some simulators [14][16] allow automated generation of test environments, using

ready generic libraries, which significantly accelerates the components testing process.

HADES [18] simulator is commonly associated with the academic environment and is considered as a tool for beginners. The capabilities of HADES [18] are quite open, as it is modular and provides an open application programming interface (API) in Java, which is a very good opportunity to exploit its internal logic for semantic execution of simulations. Additionally, it contains a module for integration with VHDL.

What is important to note is that simulators are developed primarily in order to provide time accuracy and in the precise execution of the given models than in the direction of automatic determination of the type of components and/or automatic composition and reuse systems. However, some of their functionalities is possible to be used to improve the semantic annotation of components architecture [11].

III. Boolean Function Retrieval Process

Automatic interface annotation of components allows a low level of semantic analysis. In order to enable fully automatic search and automatic composition of systems, a software must consider the internal architecture of the components.

The logical function retrieval process can be separated into two phases. The first phase is the automatic semantic annotation of the internal architecture, while the second phase is exploiting the obtained semantic resources into an unambiguous logical (Boolean) function, which describes the architecture. The two phases are actually interconnected so that the first phase is the tool for the second phase that defines the result.

In our case the first phase is done by the annotation module of the HDL IP Cores system and the internal architecture of components is modeled as interlinked instances of simpler architectures. This methodology of developing new architectures has been accepted in the paradigm of structured representation of architecture within the hardware description languages (HDL), such as VHDL.

The HDL IP Cores annotator is based on ontology [12] which represents a semantic structure of hardware. In the ontology we provide the necessary classes and relations for the full annotation of architecture which continues to be the main tool for semantic search, comparison and simulation of multiple digital components that are found and distributed all over the web. The annotator will also provide the architecture type (such as Gate, Multiplexer, Coder and etc.).

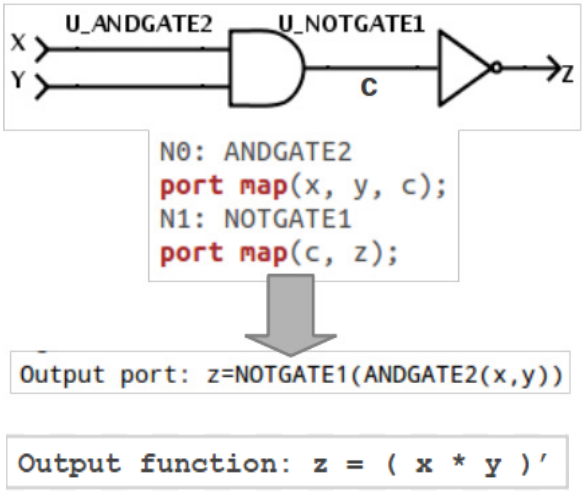

Fig 1: Annotation of logical function

The second phase will extract the logical function using semantic resources (RDF) which architecture is annotated with. The process of extracting the logical function of the architecture defined in VHDL is illustrated in Fig. 1. Architecture has an interface with two inputs (X and Y) and one output (Z). It contains an instance of "AND"-gate ("U_ANDGATE2") and an instance of a logical inverter ("U_NOTGATE1"), mutually interconnected with a signal "C". The part of the code that is instantiating the components are shown in Fig. 1, while the full content of the file is displayed in Appendix.

The final goal is to obtain the logical function of the output port "Z", i.e. to find the logical functions for all output ports of the architecture. The proposed algorithm for the logical function determination of the output port aims to find the input signals that function depends on, connecting recursively the inner instances and associated signals. The result of the algorithm is the architecture's logical function in Prolog syntax. The algorithm starts from each output port and is recursively finds component instances and its inputs, until the architecture inputs are found, Fig. 2.

The architecture description in the form of a logical function will be a good base to improve the algorithms and systems for automatic classification and composition of components, since it will allow unambiguous comparison of two architectures, according to the logical search function and appropriate ranking results according to mutual similarity and compatibility.

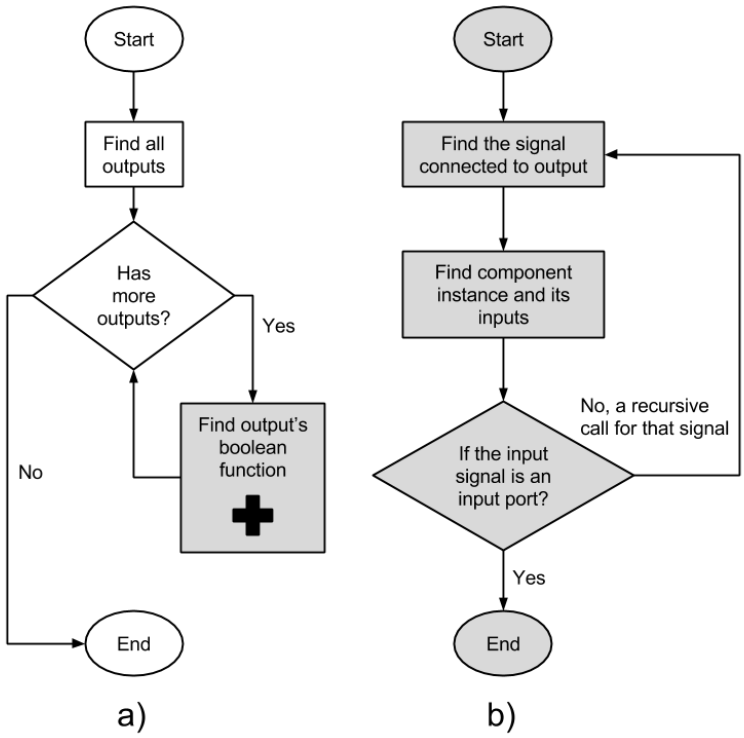

Fig. 2. Algorithm's flowchart. The algorithms are called for each output port (a) and for each output recursively finds component instances and its inputs, until the architecture inputs are found (b).

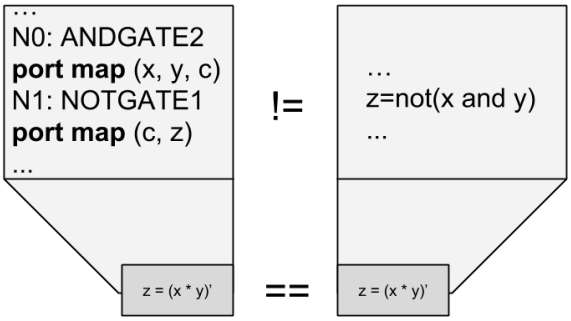

Fig. 3. Syntax abstraction layer. The VHDL codes of both components are different, but the Boolean functions are equal.

Thus, a level of syntax abstraction is introduced, as shown in Fig. 3, and different HDL IP Cores (even in different hardware description languages) may be compared and matched.

As already noted, the proposed algorithm works only for combinational circuits, whose architectures are described by structural paradigm. In case when using processes (functions) for definning the architecture of systems, the procedure for determining the logical function of architecture is significantly more complicated because the process would have to be modeled as a black box with a custom interface, and then to determine the type of architecture.

This procedure, is not possible without automatic execution of the simulation, except in the simplest scenarios. The procedure is also difficult to perform for sequential logic circuits, even the use of the simulator can not always be expected to lead to an unambiguous determination of architecture.

From the previous elaboration, the process determination of the logical function of combinational digital circuits described in hardware description languages is obvious that as the output receives the logical function output port "Z", from which could be easily obtained a mathematical function of the output "Z". That is equal to the negation of the product of inputs "x" and "y". This operation certainly belongs to the class of reverse engineering processes, since from an existing HDL code it obtains the mathematical function (expression), which might be used, elaborated, analyzed in different directions.

I. Future work

This paper illustrates the process of obtaining a logical function for combinational logical circuits described with structural paradigm, which is the first step in automatic semantic annotation of the internal architecture of digital systems.

Further step is to expand the procedure in order to cover the logical circuits defined by processes (functions). For this purpose, we will use the programming interface of the simulator HADES. Additionally, there is provided a library with classified components (gates, multiplexers, counters, etc.), which would be a good basis for algorithm training and automatic generation of rules for determining the architecture type of the new components.

Afterwards, unambiguous or using heuristic algorithms would be necessary to describe the architectures of complex sequential circuits and digital systems. This concept can encompass the most of the components available today and provide advanced functionalities within the HDL IP Cores system.

APPENDIX

```
library  ieee;
use  ieee.std_logic_1164.all;

entity NANDGATE2 is
    port(
       x : in STD_LOGIC;
       y : in STD_LOGIC;
       z : out STD_LOGIC
       );
end NANDGATE2;

architecture NANDGATE2 of NANDGATE2 is

signal c, d: std_logic;
   component NOTGATE1
       port(
          n_in : in STD_LOGIC;
          n_out : out STD_LOGIC
          );
   end component;

   component ANDGATE2
      port(
          a_in1, a_in2 : in STD_LOGIC;
          a_out        : out STD_LOGIC
          );
   end component;

   begin
   N0: ANDGATE2
      port map(x, y, c);
   N1: NOTGATE1
   port map(c, z);
end NANDGATE2;
```


ACKNOWLEDGEMENT

The work in this paper was partially financed by the Faculty of Computer Science and Engineering, at the "Ss. Cyril and Methodius" University in Skopje.

REFERENCES

[1] G. Martin and G. Smith, "High-level synthesis: Past, present, and future," Design Test of Computers, IEEE, vol. 26, no. 4, pp. 18–25, 2009.

[2] H. Dibowski, J. Ploennigs, and K. Kabitzsch, "Automated design of building automation systems," Industrial Electronics, IEEE Transactions on, vol. 57, no. 11, pp. 3606–3613, 2010.

[3] J. Teich, "Hardware/software codesign: The past, the present, and predicting the future," Proceedings of the IEEE, vol. 100, no. Special Centennial Issue, pp. 1411–1430, 2012.

[4] W. Silva, E. Bezerra, M. Winterholer, and D. Lettnin, "Automatic property generation for formal verification applied to hdl-based design of an on-board computer for space applications," in Test Workshop (LATW), 2013 14th Latin American, pp. 1–6, 2013.

[5] G. De Micheli, "An outlook on design technologies for future integrated systems," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 28, no. 6, pp. 777–790, 2009.

[6] S. Hansen, D. Koch, and J. Torresen, "Simulation framework for cycle-accurate rtl modeling of partial run-time reconfiguration in vhdl," in Re-configurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2013 8th International Workshop on, pp. 1–8, 2013.

[7] Open Cores, http://opencores.org;

[8] T. Berners-Lee, J. Hendler, O. Lassila, et al., "The semantic web," Scientific american, 2001.

[9] T. Berners-Lee and N. Shadbolt, "Theres gold to be mined from all our data," The Times, 2011.

[10] T. Berners-Lee, "Long live the web," Scientific American, Dec. 2010.

[11] HDL IP Cores, http://hdlipcores.finki.ukim.mk

[12] Eclipse plug-in, http://hdlipcores.finki.ukim.mk/plugin

[13] Xilinx ISE Simulator (Isim), http://www.xilinx.com/tools/isim.htm

[14] Cadence's Enterprise Simulator, http://www.cadence.com

[15] V. Zdraveski, M. Jovanovik, R. Stojanov, and D. Trajanov, "HDL IP Cores search engine based on semantic web technologies," ICT Innovations 2010, Communications in Computer and Information Science, vol. 83, no. 2, pp. 306 – 315, 2011.

[16] Verissimo System Verilog Test bench Linter, http://www.dvteclipse.com/Verissimo_SystemVerilog_Testbench_Linter.html

[17] VCS' Native Test bench (NTB), http://www.synopsys.com/Tools/Verification/FunctionalVerification

[18] HADES, http://tams-www.informatik.uni-hamburg.de/applets/hades

[19] V. Zdraveski, A. Dimitrovski, D. Trajanov. "HDL IP Cores System as an Online Testbench Provider". Small Systems Simulation Symposium, Volume: 5th, Nis, Serbia, February 2014.

[20] V. Zdraveski, M. Jovanovik, R. Stojanov, D. Trajanov. "HDL IP Cores Searh Engine Based on Semantic Web Technologies".