VHDL IP CORES ONTOLOGY

Vladimir Zdraveski Dimitar Trajanov
Faculty of Computer Science and Engineering
Skopje, Macedonia

ABSTRACT

Recently, the hardware description languages (HDL) are part of the most of hardware design processes and the HDL components are the main intellectual property (IP) of the producers of IC's. The large companies have internal databases and moreover whole code sub-versioning repositories, but however the easiness of code reuse is still quite low and there is almost no intelligence in the storage systems. Contributing to the improvement of the hardware design process, essentially based on the reuse of previously written cores, and utilizing the Semantic Web technologies, we propose a basic ontology for semantic annotation of VHDL components, that also contains the most frequently used component types and their "is-part-of"-dependencies, providing a knowledge base for classification and automated composition of predefined IP cores.

I. INTRODUCTION

The novel technologies of the Semantic Web offer completely different approach to the data storage and retrieval. The Semantic Web is defined as an extension of the World Wide Web that enables people to share content beyond the boundaries of applications and websites. It has been described in rather different ways, but the main idea is to publish on the Internet not only the pure text data, but also its semantic annotation, i.e. relations describing the semantics stored in the data [1].

The Semantic web technologies offer many tools for easy automatic or automated data annotation with its semantic in-domain relations and relations with other centralized data sets or knowledge bases. Prior to the use of the semantic tools in any scope of work, a semantic ontology should be defined for the specific scope. An ontology is a formal specification of a shared conceptualization [2].

The interconnection among data published in different domains is the main idea of the Linked Data initiative [3], which is also growing rapidly. It provides a good base for integration of the data existing on the Internet. Generally, Linked Data tries to connect/annotate the data that was published on the Internet, but not previously linked. More precisely, Wikipedia defines Linked Data as a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF [4].

Automatic semantic annotation of a pyre text data is relatively complex and although there are advanced tools doing it in a sufficient level of success, it is still difficult to rely on the tools only. On the other hand, it is more easy and reliable to do a semantic annotation of a structured or partially structured data, that is resulting with a very low error rate and non-ambiguous semantic annotation.

A good example of structured data type are the hardware description languages (HDLs). Hardware description languages importance on the everyday system on chip development seems to raise all the time. Few years ago the HDL was used only in the development process, for testing, simulating and emulating the new generations of processors and integrated circuits before the million-series were burned out. But today, it is not weird to find some PLD or FPGA on a production system, as a peripheral adapter or a mathematical co-processor to the main controller. Thus, the end target of the HDL is not only the pre-production testing, but to implement a new or existing functionality on an existing working PCB, that extremely explores the scope of work and future needs and goals.

The main problem seems to be that the VHDL code and projects are published as a pure text-file data and also the search engines index them in the same way. Since the VHDL files has a predefined structure by themselves, we propose automatic ontology based annotation. The process will require no further input by the end users, but will cause a step forward in the HDL code search engines improvement [5].

II. RELATED WORK

Although it is described as a novel technology, the Semantic Web exists quite a long time ago and many advanced and very useful tools are designed during the past. The Semantic Web core components consist of a Semantic Web statement, a Uniform Resource Identifier (URI), Semantic Web languages, an ontology, and instance data. Figure 1 illustrates the main components surrounded by the tools [6].

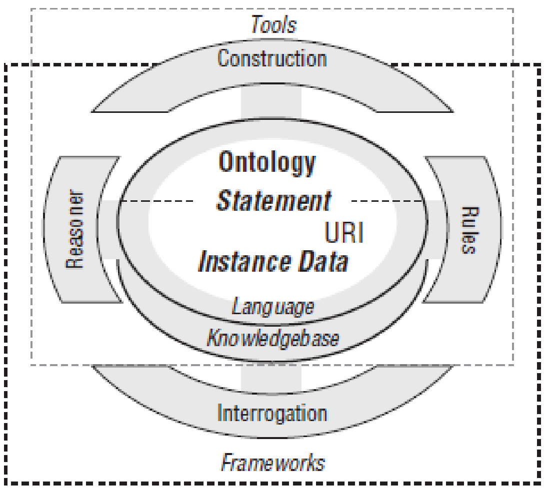

Figure 1 - Major Semantic Web Components

Besides the basic ones, very important and specific are the Reasoners, that add inference to your semantic data. Inference creates logical additions that offer classification and realization. While rules engines support inference typically beyond what can be deduced from description logic.

There are also different types of semantic systems. They can be classified, by the size of their data set, as a large scale, a medium scale or a small scale semantic systems. In terms of its architecture, semantic systems were mostly designed as a domain specific, which are intra-organizational systems, limited to the data of a company or organization [7][8]. But, today following the Linked Data initiative and taking the advantage of the available semantic data on the Internet, a more general, service oriented applications are built, that use not only their domain-specific data, but rather all the semantic data available world-wide.

On the side of the HDL code sharing and reuse has been done many attempts to design a novel approaches. Some of them has focused on a parameterized, generic components definition [9], others to code verification [10]. There are systems that use a set of rules and grammars in order to enable automated hardware design [11][12][13] and the most challenging topic nowadays seems to be the integration between the HDL, mostly SystemC, and a higher programming language, such as C or C++ [14][15][16].

The quality and achieved results of the mentioned tools and projects must not be neglected, but the biggest problem seems to be the further input required by the users, which is always an obstacle. What we propose is to use the structured format of the HDL languages and build the semantic annotation description, based on the ontology, without extra user input.

III. VHDL Ontology

A. General

Our OWL ontology [17][18] contains the basic VHDL constructs [11], such as port input/output types and data types, but also contains the properties required to represent the VHDL Entity as a logical block.

Although we would present this ontology from a VHDL perspective, the concept can be used for classification of any type of hardware units, chips, etc. There are specifications about VHDL components and many classes that enable quite original and intuitive classification of different, commonly used VHDL components, written by different authors. Furthermore, there are some predicates and relations that could be used to specify the hierarchy in the RDF description.

The VHDL ontology was designed using the Protégé editor [19], shown in Figure 2. The ontology is used to classify and annotate all of the VHDL components in order to store the details of the users' source code into the system. Further information about the ontology use cases could be found in [20].

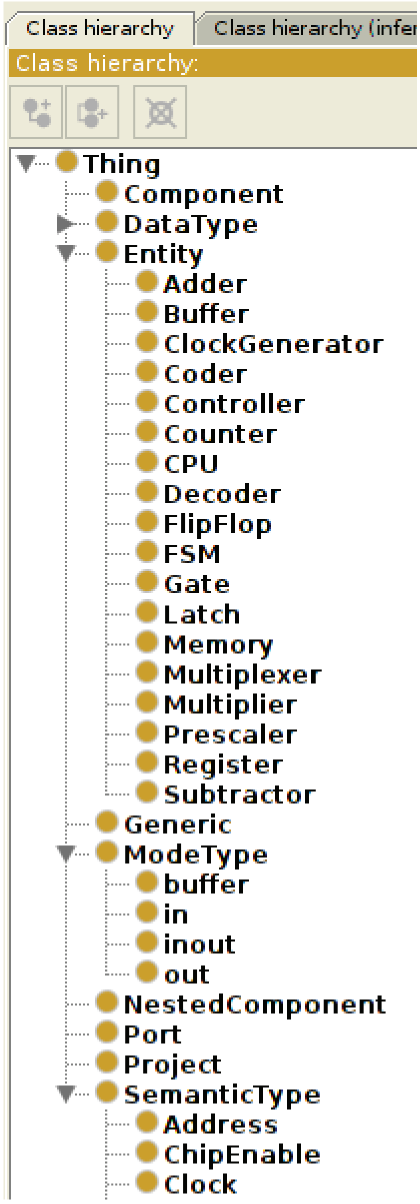

Figure 2: Entities

B. The "hasPart" sub-tree

Besides the basic VHDL constructs representation, the ontology defines types of components, starting with the basic logic gate and ending with more complex FSM and controller. The complexity levels enable easier type determination, which would be more difficult without them, since the structure must be treated as a graph.

Components are connected with property annotations, forming a complex, non-binary tree, a dag, representing the "hasPart"-dependencies among components Figure 3.

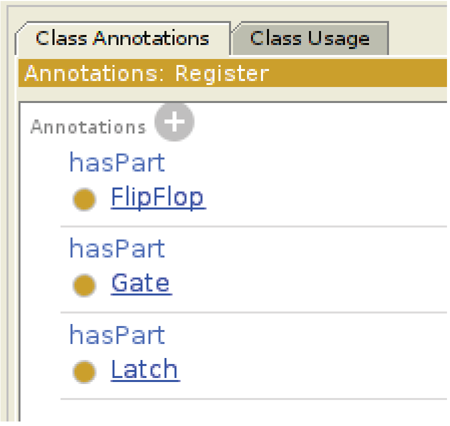

Figure 3: Annotations - "hasPart"

Actually the structure is a graph, due to the loops, but since we defined levels of complexity, it can be treated like a more complex tree structure, often called a dag, Figure 4. If a classification module of a search engine manages to generate a set of predicted component types (not exactly one), the component type could be also derived from the dag, applying some of the well-known algorithms for lowest common ancestor (LCA) in a tree/graph. Actually there are more specific algorithms which first translate the dag structure into a tree and then find the LCA or LSA (lowest single common ancestor) of a given set of nodes [21].

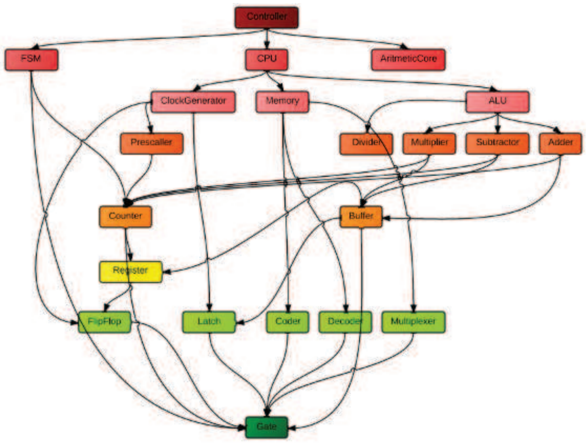

Figure 4: The "hasPart"-dag

C. Nested component - property

One of the main paradigms in the hardware design is the layer abstraction, thus the more complex components are implemented by instancing many simple ones inside them. It often happens to start searching for a simple component and end up with writing it on yourself, generating many components of the same type, inside your company's repository or even on your local machine. Our ontology makes possible to share them only by instancing them, which is a strait consequence of the ability to search and find them among the annotated source code. On the other hand, these nested components are always important in order to determinate the type of the component or other component characteristics. This the main idea of this part of the ontology is to be able to annotate the nested component of a main core, find them in the repository, retrieve their types and finally determine the component type of the main core.

D. Annotation example

Using the ontology we can generate appropriate RDF code to describe the HDL components. As we shortly mentioned above, the ontology covers the component interface and the component type semantic annotation. Furthermore, a port semantic type may be defined, thus classifying a port as data, control and etc.

```
library ieee;
use ieee.std_logic_1164.all;
--------------------------------------------------------
entity AND_ent is
--<@    Gate    @>
port(   x: in std_logic;
        y: in std_logic;
        F: out std_logic
);
end AND_ent;
--------------------------------------------------------
architecture behav1 of AND_ent is
begin
    process(x, y)
    begin
        -- compare to truth table
        if ((x='1') and (y='1')) then
            F <= '1';
        else
            F <= '0';
        end if;
    end process;
end behav1;
```

Figure 5: Simple AND_GATE annotation

A simple example of the RDF generated for a basic component is shown in Figure 5. What we propose as a final concept is an embedded RDF inside the HDL code, similar to the embedded RDF inside HTML, well known as a main idea of the Semantic Web. As shown in Figure 5, the semantic description (RDF) will be inside the source HDL file and search engines will be able to analyze its semantics, instead of doing some statistical key-word based text comparison and matching.

Since the process of semantic annotation can be done completely automatically, all the embedded RDF could be added in the client-side editors, just after the HDL code is ready to share or publish. Even more, it can be generated during the code generation process, hidden for the end user. Having this feature, the end user does not have extra input/work, but the search engines could be designed in a completely different manner, than the common ones, searching for semantic match instead of key-word match. This kind of search engine is also a stable base for an automated system composition and further system design automation tools.

IV. FUTURE WORK

The ontology is a part of our more complex system for automatic semantic annotation, classification and automated composition of VHDL IP cores. The system semantics is based on the ontology and the main task is to extend it with other HDL's, such as Verilog, System Verilog, SystemC and etc. Besides, the "hasPart" annotations should be improved and the set of entities has to be extended in order to cover all the component types appearing nowadays.

The other issue is to define, inside the ontology, configurations for the most frequently used SoC's and enable the user to chose configuration and wait for a computer to find all the required components, compatible among each other, and finish his new design in a very short time. In such manner, the designers will concentrate on their new modules and get all others automatically by the intelligent system. This concept could be also extended to the design process [22].

In order to participate in the Linked data initiative, some test data sets based on the ontology should be also published in order to enable other programmers to test and evaluate their systems and afterwards extend their knowledge bases with our ontology, leading it to the higher levels of connectivity, among the other linked data.

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] The Semantic Web initiative - http://semanticweb.org

[2] Ontology - http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[3] The Linked Data initiative - http://linkeddata.org/

[4] Resource Description Framework - http://www.w3.org/RDF/

[5] Tim Berners-Lee and Nigel Shadbolt: "There's gold to be mined from all our data.". *The Times*. 31st December, 2011.

[6] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, "Semantic Web programming", *Wiley Publishing, Inc.*, Indianapolis, Indiana, pp. 11-17 , 2009

[7] Chirita, P.-A., Costache, S., Nejdl, W. and Paiu, R. (2006) "Beagle++: Semantically enhanced searching and ranking on the desktop", *ESWC 2006*, LNCS 4011, pp 348-362.

[8] Zhang, L., Yu, Y., Zhou, J., Lin, Ch. and Yang, Y. (2005) "An enhanced model for searching in semantic portals", *In WWW 2005*, pp 453-462, 2005.

[9] Holger Lange and Andreas Koch, "Hardware/Software-Codesign by Automatic Embedding of Complex IP Cores", *Field Programmable Logic and Application, 14th International Conference ,* FPL 2004, Leuven, Belgium, September 2004.

[10] Sandeep K. Shukla, Frederic Doucet, and Rajesh K. Gupta, "Structured Component Composition Frameworks for Embedded System Design", *HiPC '02 Proceedings of the 9th International Conference on High Performance Computin, UK, pp. 663-678,* 2002.

[11] VHDL, http://www.vhdl.org/

[12] D. Mathaikutty and S. Shukla, "SoC Design Space Exploration through Automated IP Selection from SystemC IP Library", *In proceedings of IEEE International SOC Conference,* September 2006.

[13] F. Doucet and S. Shukla and M. Otsuka and R. Gupta. "BALBOA:A Component Based Design Environment for SystemModels", *IEEE Transactions on Computer Aided Design,* pp. 1597 – 1612, December 2003.

[14] Y. Xiong and E. Lee. "An extensible type system for component-based design." *In Proceedings of the 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 20–37, March 2000.

[15] D. Mathaikutty and S. Shukla. "Mining Metadata for Composability of IPs from SystemC IP Library", *In proceedings of Forum on specification and Design Languages*, September 2006.

[16] M. Baleani, A. Ferrari, A. L. Sangiovanni-Vincentelli, and C. Turchetti. "Hw/sw codesign of an engine management system", *In DATE 2000*, pages 232–237, Paris, France, March 2000.

[17] OWL, Web Ontology Language, http://www.w3.org/TR/owl-features/, 2010

[18] Custom Ontology for VHDL annotation, http://hdlipcores.mk/vhdl.owl .

[19] Protégé – semantic data editor, RDF, OWL…, Stanford Center for Biomedical Informatics Research, http://protege.stanford.edu/, 2010.

[20] 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 Volume 83, 2011,* pp 306-315, Orhid, Macedonia, September 2010.

[21] J. Fischer and D. H. Huson. "New Common Ancestor Problems in Trees and Directed Acyclic Graphs", *Information Processing Letters archive Volume 110 Issue 8-9,* pp. 331-335, Netherlands, April, 2010.

[22] H. Hu, Dayou Liu and Xiaoyong Du, "Semi-automatic Hardware Design using Ontologies", *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th,* vol. 2, pp. 792 - 797, December 2004.