INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS
DOI: 10.46300/91013.2021.15.13 Volume 15, 2021

Towards an Ontology-driven Intellectual Properties reuse for Systems On Chip design

Fateh Boutekkouk
Department of Mathematics and Computer Science
University of Oum El Bouaghi, BP 358
Oum El Bouaghi, Algeria
Fateh_boutekkouk@yahoo.fr

***Abstract*—Intellectual Properties reuse has gained widespread acceptance in System-On-Chip design to manage the complexity and shorten the time-to-market. However the need for a standard representation that permits IPs classification, characterization, and integration is still a big challenge. To address this problem, we propose to develop an IPs reuse specific ontology that facilitates IPs reuse at many levels of abstraction and independently from any design language or tool. Our ontology is built using the Protégé-OWL tool**

***Keywords*—SOC; IP; Reuse; Integration;Ontology**

I. INTRODUCTION

A System On Chip (SOC) [7] can be defined as an Integrated Circuit that can integrate in the same chip a diversity of programmable/nonprogrammable components. A typical SOC involves general purpose processors to execute embedded software and Real Time Operating System (RTOS) code (ex. ARM), DSPs (Digital Signal Processors), microcontrollers, Application Specific Integrated Circuit (ASIC) to execute specific tasks, reconfigurable parts like FPGA (Field Programmable Gate Arrays) for reconfigurable computing, memories (ex. RAM, ROM, Flash, I-cache, D-cache), and communication infrastructure. In addition to digital parts, SOC can also integrate analog, RF (Radio Frequencies) blocks, Voltage regulators, power management circuits, and Inputs/outputs interfaces.

Most embedded systems are nowadays implemented as SOCs. Such a solution can decrease the implementation cost and the power consumption but in the same time it faces a big challenge with regard to the effort of design including verification cost and time-to-market pressure. One way to manage the ever increasing complexity in SOC design is through proper Intellectual Properties (IPs) (or virtual components) reuse [2]. These design components are called "Intellectual Property" blocks because they are traded as rights to use and copy the design. IPs are predesigned, parameterized and verified components which are delivered by many vendors (third-party vendors). Normally, industry requires designing all these IPs under one platform or even within the same company. Under IP reuse platform, design efficiency is achieved by ease of plug-and-play. IP cores may include embedded processors, memory blocks, interface blocks, analog blocks, and components that handle application specific processing functions. Corresponding software components are also provided in a reusable form and may include RTOS and kernels, library functions, and device drivers. IP reuse based-design is becoming recently a standard for time-to-market driven industry.

Unfortunately, this IP based design is not free of considerable effort. Most IPs are incompatible and the cost of integration may be non reasonable. How to integrate these IPs in one chip is a tough problem. In fact, IPs may be a software component (legacy code written in assembly or a high level language such C, C++, JAVA, etc.) or a hardware component which in turn can be soft (VHDL code), hard or firm, described in Transaction Level (SystemC), algorithmic (behavioral) level (ex. VHDL, Verilog), RTL (Register Transfer Level) or logical level. Even two IPs are described in the same abstraction level and expressed in the same language; they can be incompatible in their Input/Output signals bit size, horologe frequency, voltage, timing constraints, and tools used to create these cores (ex. Simulation, performance estimation, and synthesis tools). The main steps of the IP reuse process, before the effective integration of the IP component into a design, are: IP creation, IP qualification, IP classification and search, IP transfer, and IP evaluation. The integration process involves connecting the IP cores to the communication network, implementing design-for-test (DFT) techniques and using methodologies to verify and validate the overall system-level design. A well defined reuse methodology must takes into account the reuse in various dimensions: Platform reuse, IPs reuse, verification reuse, tools and environments reuse, and expertise reuse. The purpose of this paper is to address IPs reuse and integration issues in SOC design by developing IP reuse specific ontology. Ontology is used to capture knowledge about some domain of interest. It describes the concepts and the relationships that hold between these concepts in the specific domain. In the context of SOCs domain, defining ontology may bring many advantages to IPs vendors, customers and designers who collaborate via internet to create SOCs by reusing existing IPs that satisfy the customers' requirements. A unified representation of concepts, relations, and semantics specific to SOC domain may facilitate the interoperability and integration between reusable platforms, IPs, tools and expertise. The rest of the paper is organized as follows: section two is devoted to related works on IPs reuse and integration. Section three presents the

concept of IPs. Section four discusses the IPs integration strategies for SOC design. Section five introduces the concept of ontology and some concepts that are related to IPs reuse and the IPs integration flow. Our proposed ontology is developed in section six using the *Protégé* tool for ontology editing before the conclusion and perspectives.

II. Related Work

The literature on IPs reuse is very rich. Here we try to mention some pertinent works. The VSI Alliance (VSIA) [13] specifies interface standards and specifications that facilitate the integration of IPs at both the functional level (interface protocols) and physical level from multiple sources for the development of SOCs. The OCP International Partnership Association, Inc. (OCP-IP) [12] promotes and supports the open core protocol (OCP) as the complete socket standard for rapid creation and integration of interoperable IPs. The IP-XACT standard [1] is an emerging IP reuse strategy from the Spirit Consortium. IP-XACT defines an XML schema for describing the buses, the ports, the configuration and the properties of reusable hardware cores in a vendor neutral manner and to facilitate core reuse at a high level of abstraction. Authors in [1], present CHREC XML, a XML schema that facilitate reconfigurable IPs reuse by encapsulating their details at multiple levels of abstraction namely RTL layer (the first layer), Data Type descriptions (the second layer) and Interface Operation Information (the third layer). The developed schema is independent from any design language or tool. An IP integration tool that allows a designer to select and integrate IP cores from a variety of languages/tools was also created based on this schema. The tool can automatically run the appropriate FPGA implementation tools to generate the FPGA bitstream. The work in [3], applied hardware design patterns for customizing and integrating the IP components into SOC designs. They formulate the role of design patterns in HW design, and describe their implementation using meta-programming. Finally they proposed a Wrapper design pattern for adapting the behavior of the soft IPs, and demonstrate its application to the communication interface synthesis. Other works use XML either as a specification language for IPs modeling or as an intermediate format that is generated from HDL (Hardware Description Languages) such as VHDL, SystemC using XSLT for transformation rules specification [5, 10].

Despite, all these efforts, we can state that there is no IP reuse specific ontology. Our ultimate objective is to develop an IPs reuse specific ontology that covers all IPs reuse requirements including IPs high level modeling, refinement, performance analysis, integration, and verification. In this paper, we are interested especially in IPs reuse at system level. According to our knowledge, our ontology will be the first one specific to IPs reuse domain. The contribution of this work is the definition of a novel IPs reuse specific ontology that takes into accounts both software and hardware IPs and deals with all possible kinds of incompatibilities between IPs.

III. Intellectual Property (IP)

An Intellectual Property (IP) or a Virtual Component (VC) is a reusable pre-designed and pre-verified component (core). An IP may be analog, mixed-signal, or digital. A digital IP can be programmable or not programmable. The IP programming can appear as software programming using an embedded processor or hardware programming using programmable logic cores such as FPGA. There are two essential kinds of software IP:

- Close-to-Hardware such as RTOS, drivers, hardware-dependent code, that is optimized to particular hardware platforms, often written in assembly.
- HW-independent usually in C language.

Hardware IP can be delivered as hard IP, soft IP or firm IP. A hard IP is a fully customized hardware core where all its gates and interconnects are placed and routed, generally delivered as GDS-II files. The reusability at this level is very low. Soft IP is a fully synthesizable core with an RTL representation. The reusability at this level is high because it is technology independent. A firm IP is a hybrid of both hard and soft IPs. It is a hardware core with an RTL description together with some physical floor planning or placement. Extensions of IPs are behavioral IP, Transactional Level (TL) IP, Bus Cycle Accurate (BCA) IP and functional IP (FIP). A behavioral IP is a core with algorithmic description (ex. VHDL) suited for High Level Synthesis (HLS). A transactional IP describes the core behavior and data transfers as a set of transactions (read, write) and usually described in SystemC TLM (Transaction Level Modeling). TL IPs may be timed or untimed. A BCA IP describes the IP core interface, not functionality. Timing is cycle accurate and tied to some global clock. It is used for the architecture (performance) evaluation. FIP is used at the system level design. It does not require a particular software or hardware architecture. This kind of IP offers a very high level of reusability. FIP IP may be timed or untimed. We note the existence of another class of IPs which is the verification IP. Verification IPs are designed to become a part of the testbench or verification environment and reused to ensure the correctness of the design.

IV. IP Integration Strategies

Generally, there are three main strategies for the integration of IP components [11]. This classification is based on a clear separation between communication and computation for each component:

A. Standard-based strategy

In this strategy, IPs interfaces are compliant to a given standard. In this case, IPs may be integrated in the SOC without requiring any functional adaptation.

B. Communication Synthesis Strategy

Used if components to be interconnected have heterogeneous interfaces. In this case interface adapters (wrappers) are generated and inserted between the components. For programmable components, interface adaptation requires the development of software wrappers (device drivers) to match the application software and the RTOS to the communication infrastructure.

C. IP derivation strategy

Used when the source code (using an HDL, for hardware components, or a programming language, for software ones) of the IP components is available and may be directly modified, so that a new component may be derived from the previous one.

V. ONTOLOGIES

Ontology can be defined as an explicit specification of a conceptualization [4]. It is concerned with making representational choices that capture the relevant distinctions of a domain at the highest level of abstraction while still being as clear as possible about the meanings of terms. It defines a set of representational primitives with which to model a domain of knowledge. The representational primitives are typically classes (concepts), objects (individuals or instances), attributes, and relationships among objects. The definitions of the representational primitives include information about their meaning and constraints (axioms) on their logically consistent application. Ontologies permit to:

- Share common understanding of the information structure among people or software agents.
- Enable the reuse of domain knowledge.
- Make domain assumptions explicit.
- Separate domain knowledge from operational knowledge.
- Analyze domain knowledge.

The most recent development in standard ontology languages is OWL from the World Wide Web Consortium (W3C). The impetus behind using Ontologies in IPs reuse is to:

- Facilitate the so-called distributed IPs integration at many levels of abstraction in an internet context where many vendors (IP providers), customers, and designers collaborate to design SOCs by defining reusable platforms and IPs, constraints, tools and expertise in a neutral representation.
- Expertise sharing between all persons in collaboration.
- Interoperability between different existing modelling, simulation, performances estimation, and synthesis tools.
- Effective platform derivation and IPs integration for SOC design.

The integration includes the following loop:

- IPs and platforms research, qualification and storage.
- System Level integration.
- System Level model analysis.
- Architecture candidate's selection.
- Hardware/Software Assignment.
- Communication synthesis.
- Performance estimation.

A. IPs and platform research, qualification and storage

This step is initiated by customers and according to their requirements; they launch research on existing platforms and IPs. Requests are sent to different vendors who propose their IPs. All selected IPs and platforms are then registered into an IPs database.

B. System Level Integration

In this step, designer begins to establish a system level functional model by connecting compatible functional IPs (FIPs) which are conform with the SOC specification. Functional IPs behaviours are generally described in a standard system level design language like SystemC.

Designer can resort to IP derivation strategy in order to modify source code of a FIP if it does not match the requirements of performances or interface. If IPs are not compatible (for instance, one FIP is described in SystemC and other in SpecC), designer can use some tools for automatic translation of one language to another. In the worst case, he has to modify the whole code manually. A generic FIP has a name, data ports, control ports, the configuration (the selectable candidate of functions, Master or Slave), the behavioural description, design constraints, simulation tool, and other remarks and information [8]. Each FIP port is characterized by at least the name, the data type, and the input/output direction. All FIP are supposed connected each other via abstract channels using ports (direct communication). Functional verification is performed using a simulator (ex. SystemC).

C. System Level Model Analysis

The objective of this step is to enable designer to analyze the profile of his model. The inputs of this step are the functional model defined in the first step and the test data designed at the system level. Generally profilers are used to estimate the execution time consumed by each function of the system level functional model. The result will guide the designer to Hardware/Software assignment (i.e. functions which consume more time will be implemented in Hardware).

D. Architecture candidate's selection

Based on the results of the previous step, the designer chooses an Architecture Template (AT) as an architecture candidate. After selecting an AT, the designer chooses the hardware IPs and the software IPs based on the dependencies that are registered in the AT.

E. Hardware/Software Assignment

In this step, the designer tries to tradeoff hardware and software, so software IP or hardware IP is assigned to a functional IP. There are many strategies for assignment. For instance, the designer starts to assign software IPs to functional IPs and according to profiling results, hardware IPs are assigned to the appropriate software IPs that are the

performance bottleneck of the SOC. A generic software IP has a name, a configuration (the selectable candidate of software algorithms and data structures), a set of control/data ports, a behavioral description, the IP performances including execution time, power consumption, object code size under the processor parameters such as the processor clock cycle, the compiler with optimization options, and a set of remarks. Similarly to software IP, a generic hardware IP has a name, a configuration (the selectable candidate of hardware algorithms), a set of control/data ports, a behavioral description, the IP performances including the area, the processing time, the power consumption given by synthesized circuits properties under the device parameters such as the design rule, the voltage, etc., and a set of remarks. In order to facilitate the automation of hardware/software assignment, some principles and rules have to be defined. Here are some examples:

- Functional IPs and software IPs are designed to be functionally equivalent.
- Hardware IPs are designed to be functionally equivalent to the leaf software IPs. A leaf IP is defined as an IP that cannot be further subdivided into other IPs.
- Each port of an IP must have one-on-one correspondence.
- After assigning software IPs to functional IPs, the direct communications between functional IPs are refined to a transaction level communication such as a packet transformation.
- Leaf software IPs that are the performance bottleneck are assigned by hardware IPs to accelerate the performance of the SOC.

F. Communication Synthesis

After the hardware/software assignment, interface between incompatible IPs is generated. The interface can be a hardware wrapper (transducer) or a software wrapper (RTOS, device driver) model. The choice of the interface model is based on the adopted IP integration strategy. For instance, using the core-based strategy, a transducer with two state machines (FSMD) to transform incompatible protocols and a queue to smoothen the communication data is generated. The hardware/software wrappers models descriptions are also available in the AT as upper/lower communication methods. This step passes by a series of refinement phases so finally upper communication methods are replaced by lower communication methods.

G. Performance Estimation

In this step, accurate estimation of performances (e.g. execution time and power consumption) is performed using static performance estimation tools relying on IPs characteristics or dynamic analysis with simulators.

VI. Our Ontology

As stated before, our ultimate objective is to develop a IP reuse specific ontology that permits a neutral and a standard representation of IPs, platforms, knowledge, and expertise and their eventually reuse for SOC design. The benefits of such ontology are seen especially in an internet context where customers and designers look for appropriate platforms IPs, and tools that are provided by many IPs vendors. In this work, we will deal with IPs as black boxes that communicate with each other through predefined interfaces. An interface contains several data ports, a list of attributes and constraints and a communication protocol description. To each port we, associate a contract including provided/required services (signals). We can distinct four cases for system level IPs integration:

- IPs are perfectly matched if they have the same protocol communication and data over ports have the same size. In this case, IPs can be connected directly without wrapping.
- IPs have the same communication protocol but data signals over ports have different sizes. In this case, an FSM is needed to transform the signals.
- IPs have different communication protocols but data over ports have the same size. In this case, a protocol adapter is needed.
- IPs have different communication protocols and data over ports have different sizes. In this case, a protocol adapter and an FSM are needed.
- IPs have different communication protocols, data over ports have different sizes and IPs clock frequencies are different. In this case, we can use a transducer with two FSMD to transform incompatible communication protocols and a queue to smoothen the communication data.

In order to model IPs reuse and integration, we have to define the following classes: IP, Configuration, Interface, InterfaceAdapter, Port, PortAdapter, Signal, DataType, CommunicationProtocol, FSM, FSMD, Transducer, Queue, Wrapper, Template, and IPLibrary. InterfaceAdapter and PortAdapter are subclasses of Interface and Port classes respectively and they are parts of the wrapper class. The template class shows the designer the types of available IPs, the dependencies between them, what IPs can be optional or selectable and what IPs must be required in the desired SOC architecture. The template is often abstracted in such a way the designer sees a standard API (Application Programming Interface) without detailed knowledge of the architecture. All these classes with their properties are specified in the *Protégé OWL* tool [6]. Using Protégé OWL, All concepts of IPs are defined as classes; relationships among classes are defined as objects properties; attributes of classes are defined as Datatype properties. Datatype properties link an individual to an XML Schema Datatype value. From XML format, we can exploit existing commercial or open source parsers to generate many HDL or software languages. We can also exploit existing adapters such as SystemC/OCP and SystemC/Verilog for system level integration. The links of these tools are defined as Datatype properties.

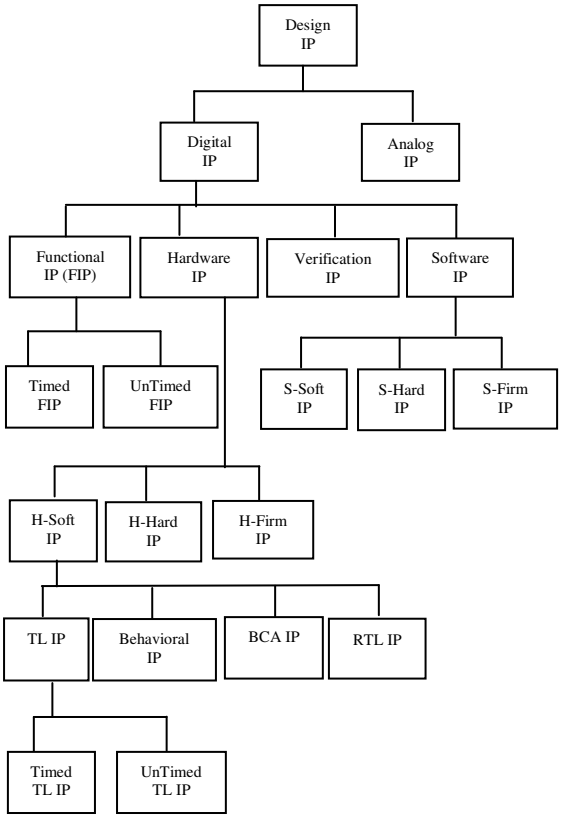

Fig. 1. IPs taxonomy

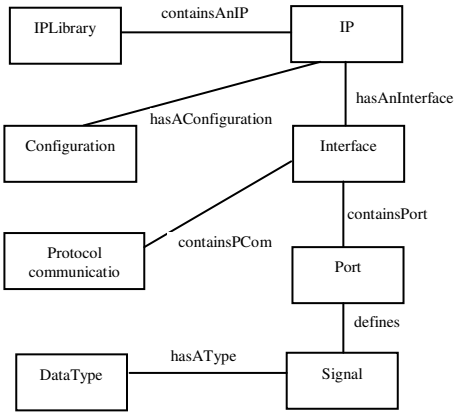

Fig. 2. IP relations

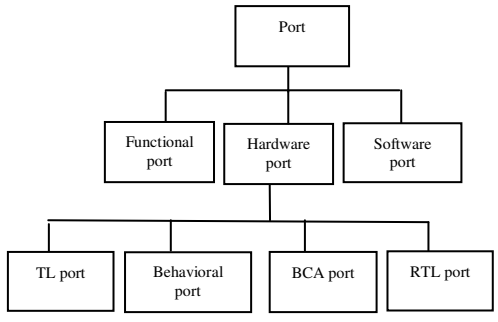

Fig. 3. Ports taxonomy

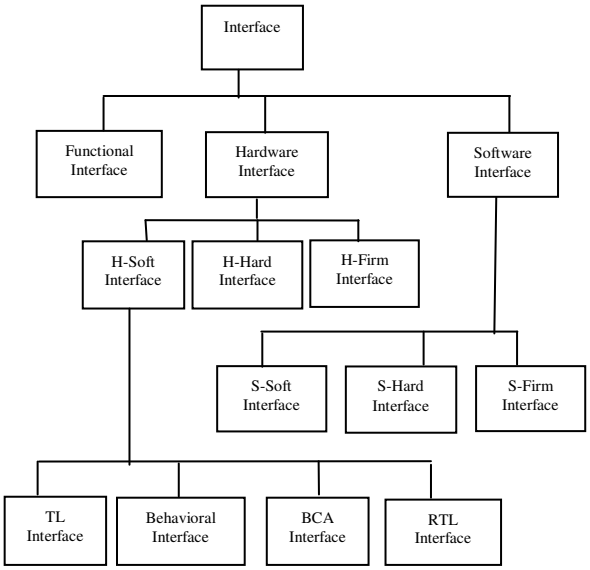

Fig. 4. Interface taxonomy

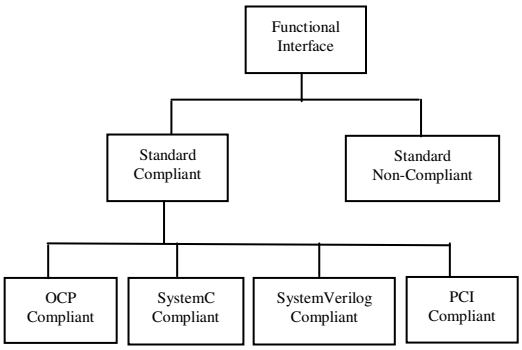

Fig. 5. Functional Interface taxonomy

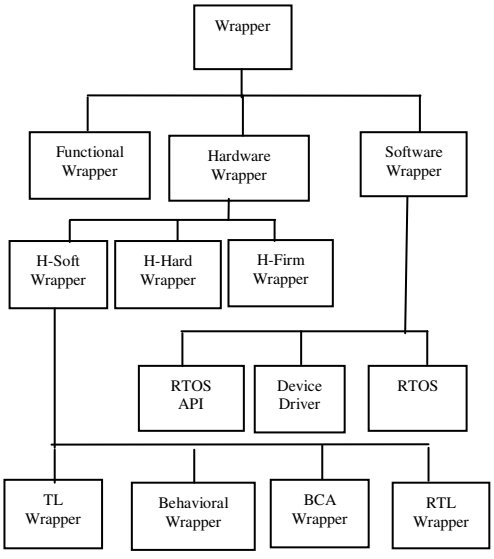

Fig. 6. Wrappers taxonomy

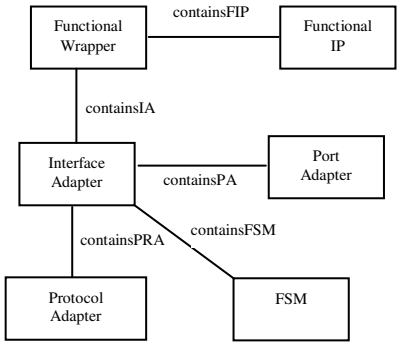

Fig. 7. Functional wrapper relations

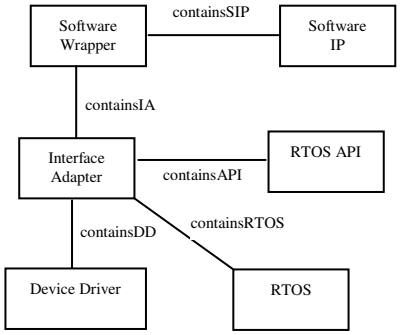

Fig. 8. Software wrapper relations

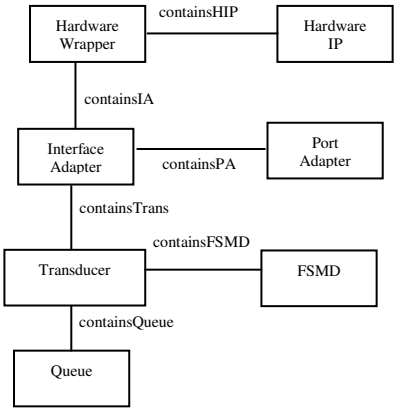

Fig. 9. Hardware wrapper relations

VII. Conclusion

In this work, we presented our ontology for IPs reuse and integration. We have exploited protégé-OWL tool to build the proposed ontology. Using Protégé, we can easily specify IP concepts, relationships and attributes, and check the coherence between classes. An XML format is generated automatically which is will be used as a front-end language to generate HDL or software languages at different levels of abstractions. As a perspective, we plan to develop our System Level wrappers to integrate SystemC and SystemVerilog non compliant IPs and to integrate an MDA (Model Driven Architecture) environment to generate SystemC and SystemVerilog code automatically.

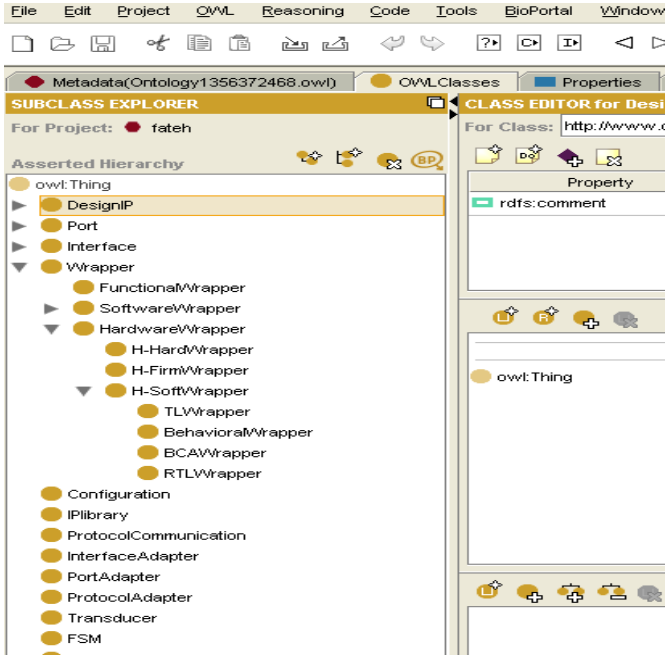

Fig. 10. Using Protégé tool to build the proposed ontology

References

[1] A. Arnesen, N. Rollins, M. Wirthlin. A multi-layered XML schema and design tool for reusing and integrating FPGA IP. In Field Programmable Logic and Applications, 2009. FPL 2009. pp.472 – 475.
[2] J. Biggs, A. Gibbon. Reference Methodology for Enabling Core Based Design. European Synopsys User Group, SNUG, March 2002.
[3] R. Damaševičius, G. Majauskas, V. Štuikys. Application of design patterns for hardware design Proceeding, DAC '03 Proceedings of the 40th annual Design Automation Conference. pp. 48-53.
[4] D. Djuric, D. Gasevic, V. Devedzic. Ontology Modeling and MDA. In Journal on Object Technology, Vol 4, N 1, January-February 2005.
[5] M. Hamdoun, A. Ghrab, P. Hernandez, G. Saucier. IP XML Encapsulation Portal. Proceedings International Workshop on IP-Based SoC Design, December 2001, Grenoble, France.
[6] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools, Edition 1.0, August 27, 2004.
[7] A.A. Jerraya, W. Wolf. Multiprocessor systems on chip, Morgan Kaufmann publishers, 2005.
[8] M. Muraoka, H. Nishi, R. K. Morizawa, H. Yokota, Y. Onishi. SoC Architecture Synthesis Methodology Based on High-Level IPs. In IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E87-A No.12 pp.3057-3067. Publication Date: 2004/12/01, Print ISSN: 0916-8508.
[9] D. Shin, D. Gajski .Interface synthesis from protocol specification. Technical Report CECS-02-13, University of California, Irvine, April 12, 2002.
[10] M. Visarius. An XML Format Based Integration Infrastructure for IP Based Design. 16th Symposium on Integrated Circuits and Systems Design (SBCCI'03), Sao Paulo, Brazil September, 2003.
[11] F. R. Wagner, W. O. Cesario, L. Carro, A.A. Jerraya. Strategies for integration of hardware and software IP components in embedded systems on chip. In VLSI journal, Elsevier, 2004.
[12] OCP-IP, http://www.ocp-ip.org.
[13] www.vsi.org