UML based hierarchical state diagram approach for protocol designs

Sreejith Sudhakaran, Wah Man Cheung,
Klaus D. McDonald-Maier
School of Computer Science and Electronic Engineering
University of Essex
Colchester, United Kingdom
ssudha@essex.ac.uk, wmcheu@essex.ac.uk,
kdm@essex.ac.uk

Gareth Howells
School of Engineering and Digital Arts
University of Kent
Kent, United Kingdom
w.g.j.howells@kent.ac.uk

Abstract— **The increasing complexity in the design of protocol based sequential digital systems such as USB3.0 and PCI express (PCIe), is leading to an increased time to market constraint. This paper introduces a UML based visual design approach to address this increased complexity in the design of IP as well as System-on-Chips (SoC). A hardware development method for USB3.0 device using the Unified Modeling Language is explored. It focuses on the conversion of UML structure diagrams and hierarchical state machines into synthesizable hardware description language. A Model Driven Development (MDD) method using UML state diagrams and hierarchical design breakdown approach is used for the development the synthesizable HDL for USB3.0 device IP with more than 20 sequential states.**

Keywords-Unified Modelling Language (UML); USB3.0; Model Driven Development (MDD); State machine based design; Intellectual Property(IP); Link Training and Status State Machine (LTSSM); Verilog; Hardware Development Language (HDL)

I. INTRODUCTION

System-on-Chip (SoC) design has become a time consuming process because of the increased complexity of protocols such as USB3.0, PCI Express (PCIe) and other digital logics. Recent studies have underlined the relevance of the Unified Modeling Language (UML) for the effective modeling and development of electronic systems and handling the exponential increase in the complexity of the systems [1, 2, 3, 4, 5, 6, 7, 8]. After UML emerged in the early 90's, quite a few flavors of UML have been derived, in different domains such as System-on-Chip and real-time embedded systems [9]. Majority of this research where generally concentrated at high level modeling, documentation and system software developments [5]. UML is a multidirectional tool, the application of UML modeling can be extended in the development of complex sequential electronic systems. The understanding and implementation of digital systems, where there are more than 20 states can be rather challenging. Abstract UML diagrams, combined with a model driven development method, can significantly reduce this design complexity, leading to a short design turn-around time. This paper is an attempt towards a UML based visual design approach to address this increased complexity in the design of various IP blocks as well as System-on-Chips (SoC).

The paper presents an implementation method using UML diagrams and their properties in the design of Link Training and Status State Machine (LTSSM) of USB3.0 protocol. This paper is a continuation of a preliminary paper [10].

A feature subset of UML2.0 and different flavors of UML are selected for the efficient modeling of the design and the major focus is to experiment the usage of UML diagram based hierarchical design of complex protocol based digital circuits. The preliminary UML diagram is converted to a detailed UML model, by including most of the minor design details and logics to the diagram itself. This UML model is then to be converted to a synthesizable hardware description language (HDL) making use of Modeling Driven Development (MDD) methods [1, 2, 4]. Thus all aspects of a complex digital design can be successfully modeled in the UML structure for automatic conversion and development of the final circuit.

After the introduction, the subsequent part gives an overview of the related work with applications of UML diagrams in electronic logic development and the approach of Model Driven Development. The UML based visual design method is explained in section III. A case study of the Link Training and Status State Machine (LTSSM) of USB 3.0 device IP is explained in section IV. Finally, the paper summarizes with the advantages of the UML design method and a description of future studies and implementations for the automated design of complex digital systems.

II. RELATED WORK

A. *UML in Electronic System Development*

The Unified Modeling Language (UML) can be described as a set of graphical symbols that help in representing and designing systems, mainly focused on software systems made with the help of object oriented concepts [11]. Graphical symbols characterize the language rules for UML. UML was originally developed in 1990's and maintained by the Object Management Group (OMG) [11]. It is a tool for enumerating, representing, building and documenting the objectives of a

system [12]. UML has wide range syntax to deal with modern complex systems and enhance the multi-domain interactions [5]. UML has been improved from its origin in basic software designs to address the advanced modeling issues in various domains during the last decade. These include, the UML flavor for System Modeling, SysML (OMG 04-2006) [13], the modified version for System-on-Chip (SoC) (OMG 08-2006) [7, 11], and the extended version for real-time embedded system (Modeling and Analysis of Real-time Embedded System, MARTE, OMG 2008) [9].

SysML improved UML by including tools for the well-organized modeling of composite designs that contain a wide variety of hardware and software particularly in the documentation of non-functional design features [5]. UML for SoC addresses timing aspects of digital designs, helping to model hardware oriented complex digital designs in an efficient way. Alternatively, the UML version for SoC also offers improvement in modeling of the architecture of systems with the help of the Structure Diagrams for the graphical modeling of hierarchical components, ports and interfaces [5].

Furthermore, the wide range of UML diagrams materializes an efficient documentation method for design verification and component reuse.

B. Model Driven Development (MDD)

Modeling is the abstract or simplified representation of a system. As the complexity of systems increases, the process of Model Driven Development becomes important to achieve an increased productivity. Recently, there have been immense study efforts converging on the relevance of Model Driven Development for the conversion of the UML models into domain specific language (DSL) such as Very High Speed Hardware Description Language (VHDL) and Verilog [1, 2, 4, 8, 14]. In the MDD method, hybrid models of UML and HDL are used for the conversion of the high-level model into the synthesizable HDL code [2]. At present, the complete automation of the conversion is the focal point in the improvement phase of latest versions of the UML [1, 2, 5, 8, 15].

III. UML BASED DEVELOPMENT APPROACH

This section gives an overview of the UML based design approach adopted in this paper. As a visualization tool, UML diagrams are sufficient to express the depth of current design, to show the future works and for the reusability of already implemented part of the design. The topology of a digital system can be represented using the UML Structure diagram as shown in Fig. 1. This is equivalent to the block diagram for traditional hardware circuit design approach. The high level diagrams can be enriched using the unique features of UML, by representing different relationships such as dependency, association, generalization and extensibility [12] between various modules. Since UML diagrams are very abstract with ability to contain a lot of information, other design features like clock frequency, timing parameters and pseudo codes can be updated in the diagram itself (Fig. 1 and Fig. 5).

Another practice applied in this project is to use the hierarchical states in UML to represent complex state machines. In complex state machines states are rearranged, categorized and merged together to form hierarchical state machines. This will make the design easy to understand and implement. As the design becomes simpler, the lesser will be the chance of an error in the design [16]. Realization of this state diagram with even 14 numbers of states can be really demanding [16]. Consider an example diagram show in the Fig.2.

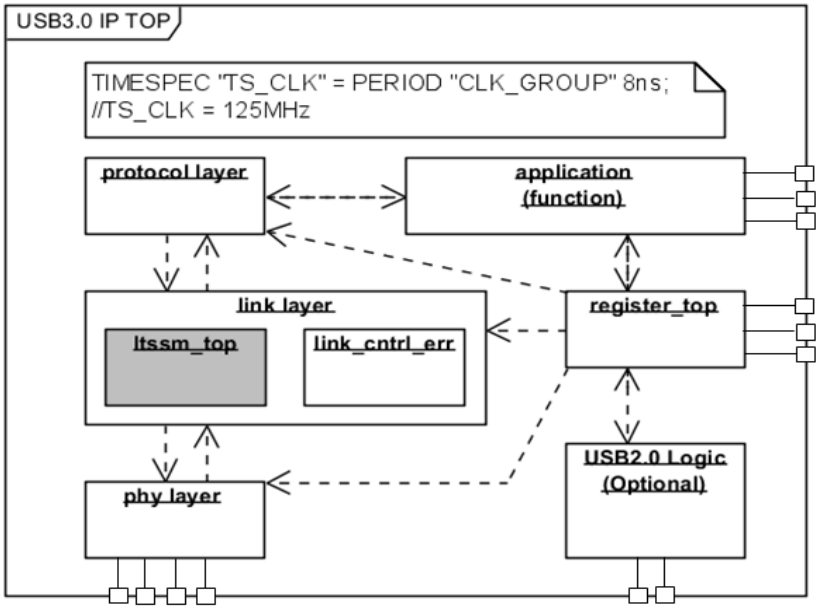

Figure 1. UML Structure Diagram for a USB3.0 design

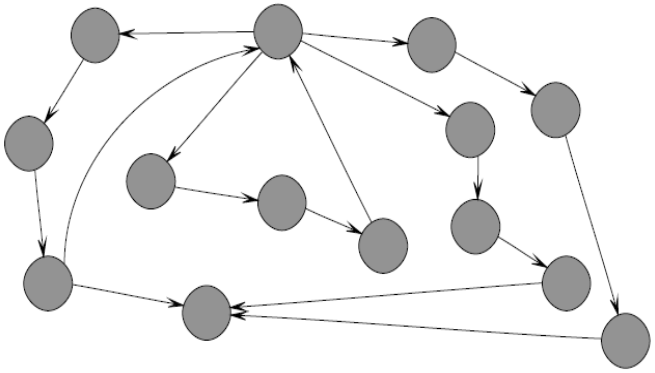

Figure 2. State Machine example

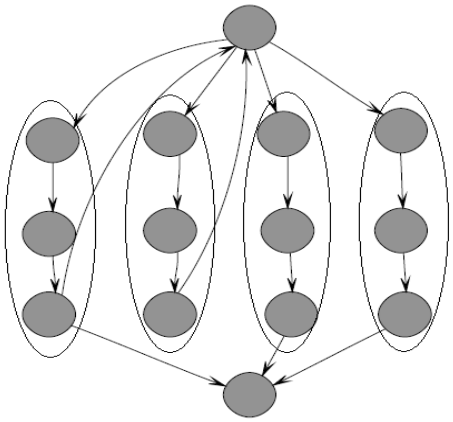

Figure 3. Hierarchical simplification

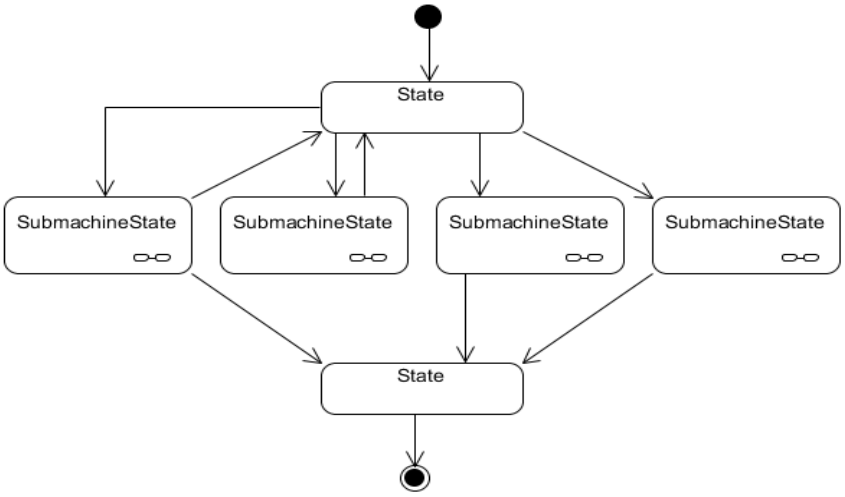

Figure 4. Hierarchical State UML Diagram

As the number of states in the state machine increases, the implementation becomes complex. Realization of this state diagram with even 14 numbers of states can be really demanding [16]. However, a simple reordering can make the state machine more readable. The reordered from is shown in the Fig. 3 below.

After the reordering, the state diagram in state 3 can be represented as a hierarchical state machine as shown in Fig. 4. This is achieved by merging the similar category states into one super-state. Current UML tools are allowing support for drawing and managing sub-state machines with different hierarchies [17].

IV. CASE STUDY: USB3.0 LTSSM IMPLEMENTATION

Very fast peripheral components such as storage devices with Terabytes capacity and other fast video equipments demands very fast data transfer schemes. Thus, the peripheral device are also needs to be fast enough, leading to the expansion of USB 3.0 over USB 2.0. USB3.0 implements a very high speed mode with a speed of 5Gbps called super speed mode [18].

The Link Layer of USB3.0 is accountable for maintaining the link interconnection; to perform an efficient data transfer between the two ends of a USB link [18]. The most important state machine in the link layer is the Link Training and Status State Machine (LTSSM), which is responsible for the initial sync up between the host and device parts. The LTSSM is a state machine with 23 different major link states for link initialization and power management [18]. LTSSM is chosen an idle sample to experiment UML, because of its complexity and hierarchical sub-state machine dependencies.

In the UML modeling of LTSSM, the requirement specification of USB3.0 is studied to evaluate the different kinds of UML building blocks necessary to realize all the features as listed in the documentation. Initially the high level UML structure diagram as given in the Fig. 1 is developed. This diagram is helpful in describing the overall hierarchy of the system and the interconnection between different sub-modules. UML diagrams are useful when re-using the sub

modules for integrating to a new system. Later this high level diagram is enriched by developing separate and detailed UML diagrams for different building blocks. A high level hierarchical state diagram of LTSSM module is given in Fig. 5 [17]. These diagrams can be more detailed by including pseudo codes for the activities in a state, conditions for state transitions and other timing parameters.

After the development of top level state machine, each sub-state is further expanded. An example of a second level hierarchical state with sub-state rx_detect of LTSSM is given in Fig. 6 [17]. The State Diagram is organized and developed in a manner to allow the mapping to Verilog in the model transformation phase [1, 2, 19]. The corresponding Verilog pseudo code representing the lower level rx_detect sub-state is presented in Fig. 7. In the similar manner, UML can be useful for describing the complete SoC or an IP in a pictorial manner, including the timing features for the synthesis phase.

The final LTSSM logic developed by the conversion of UML models in to HDL, can then be tested using a FPGA development tool with functional and timing simulations in a suitable testbench environment. After this, the synthesized logic can be downloaded into a high-speed FPGA for prototype emulation testing.

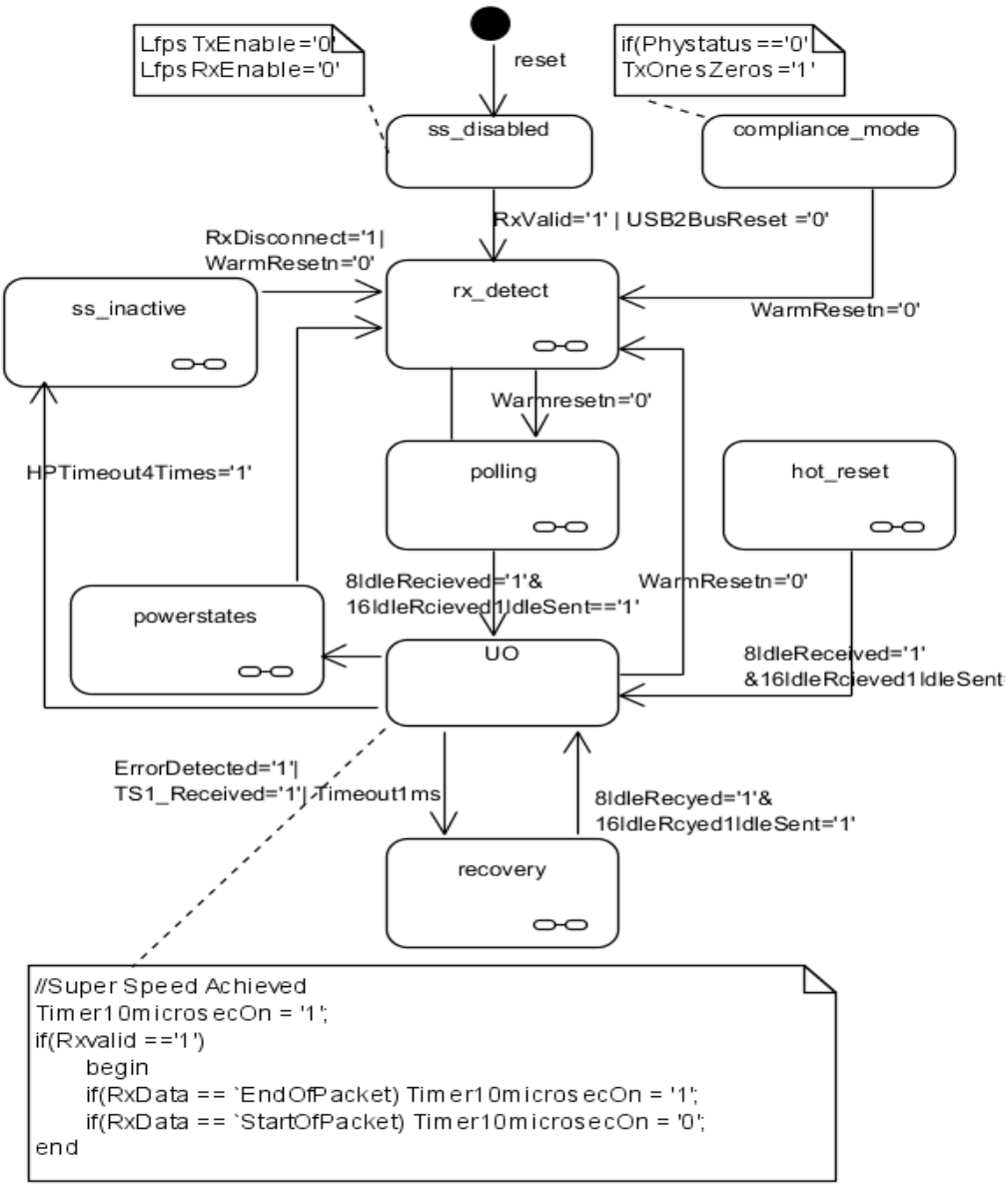

Figure 5. State Diagram for USB3.0 LTSSM (Device implementation)

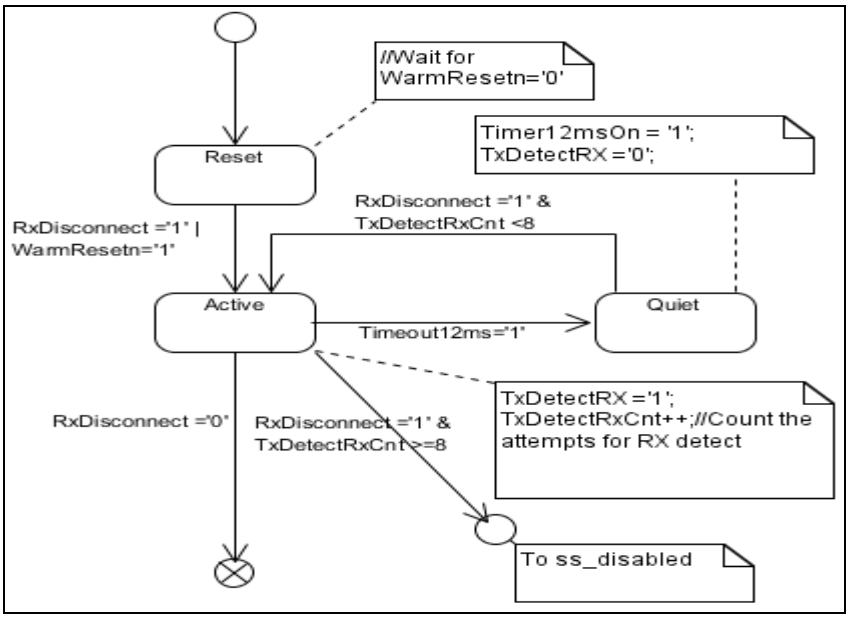

Figure 6. rx_detect sub-state machine

```
// Current State Logic (sequential)
always @ (posedge clk)
begin : Ltssm_CurrentState
if (reset = '0')
    CurrState_Ltssm <= `ss_disabled;
else
    CurrState_Ltssm <= NextState_Ltssm;
End
// Next State Logic (combinatorial)
always @ (CurrState_Ltssm)
begin : Ltssm_NextState
    NextState_Ltssm <= CurrState_Ltssm;
    // Set default values for outputs and signals
    // ...
    case (CurrState_Ltssm) // synopsys parallel_case full_case
        //...code for other states...
        `rx_detect_Quiet:
            begin
                Timer12msOn = '1';
                TxDetectRX = '0';
                if (Timeout12ms='1')
                    NextState_Ltssm <= `rx_detect_Active;
            end
        `rx_detect_Reset:
            begin
                //Wait for WarmResetn='0'
                if (RxDisconnect ='1' |
                    WarmResetn='1')
                    NextState_Ltssm <= `rx_detect_Active;
            end
        `rx_detect_Active:
            begin
                TxDetectRX = '1';
                TxDetectRxCnt++; //Count the attempts for RXdetect
                if (RxDisconnect ='0')
                    NextState_Ltssm <= `polling_LFPS;
                else if (RxDisconnect ='1' & TxDetectRxCnt <8)
                    NextState_Ltssm <= `rx_detect_Quiet;
                else if (RxDisconnect ='1' &  TxDetectRxCnt >=8)
                    NextState_Ltssm <= `ss_disabled;
            end
        //...code for other states...
    endcase
end
```

Figure 7. LTSSM Verilog code (Partial)

V. CONCLUSION AND FUTURE WORK

The paper presents a UML based and Model Driven Development scheme for complex sequential protocol design with USB 3.0 device as a case study. The Link Training and Status State Machine (LTSSM) of the USB3.0 link layer is chosen as a typical sample to start with for the development of complex state machines in UML.

This experiment strengths the observation that usage of UML can simplify the design process. This has advantages such as simple and abstract representation of the system, easy reusability, future enhancement, early verification planning and better usage of Model Driven Design (MDD). Also it can be seen that, the use of UML hierarchical states will make the design simpler and easier to implement. In future, Model Driven Development approach along with Logic synthesis can help in optimization and reduction of state machine logics.

The major drawbacks with this method are lack of features in UML for VLSI design as well as the lack of an automatic diagram to HDL conversion tool. In future, the Model Driven Development approach with fully automatic model conversion of diagrams to HDL enhances the effectiveness of design process, providing more robust design and simple design.

The major focus of this work is to set up a complete design UML library for sequential digital designs to increase component reuse, cut down design turn-around time, enhance production and improve design quality. There are some studies in the area of automatic conversion of UML diagrams into Verilog/VHDL, at the time of writing this paper. The full automation of UML to HDL conversion is in the development phase with space for improvement [5, 10].

REFERENCES

[1]. S. K. Wood, D. H. Akehurst, O. Uzenkov, W. G. J. Howells, and K.D. McDonald-Maier, "A Model Driven Development approach to mapping UML State Diagrams to synthesizable VHDL", IEEE Transactions on Computers, Vol.57, Issue 10, pp. 1357 – 1371, October 2008

[2]. D. H. Akehurst, W. G. J. Howells, K. D. McDonald-Maier, B. Bordbar, An Experiment in Using Model Driven Development: Compiling UML State Diagrams in VHDL, Forum on Specification and Design Languages (FDL 07), 2007

[3]. D. H. Akehurst, S. K. Wood, W. G. J. Howells, K. D. McDonald-Maier, Towards a Twofold Approach to the Verification of Generated VHDL Systems, International Workshop ModEasy'07. Barcelona Spain, September 2007

[4]. S. Le Beux, P. Marquet, A. Honore, Jean-Luc Dekeyser, "A Model Driven Engineering Design Flow to Generate VHDL", International Workshop ModEasy'07, Barcelona Spain, September 2007

[5]. Y. Vanderperren, W. Mueller, W. Dehaene, "UML for Electronic Systems Design: A Comprehensive Overview" Springer Netherlands, Design Automation for Embedded Systems Journal, Vol. 12, No. 4, pp. 261 – 292, December 2008

[6]. T. Schattkowsky, J. H. Hausmann and G. Engels, "Using UML Activities for System-on-Chip Design and Synthesis", MoDELS 2006, LNCS 4199, pp. 737 – 752, 2006

[7]. E. Riccobene, P. Scandurra, A. Rosti, S. Bocchio, "A SoC Design Methodology Involving a UML 2.0 Profile for System C", 1530-1591/05, IEEE, 2005

[8]. D. Bjorklund and J. Lilius, "From UML Behavioral Descriptions to Efficient Synthesizable VHDL", Proceedings of the 20th IEEE NORCHIP Conference, 2002

[9]. MARTE (November 2009), "UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems", http://www.omgmarte.org/node/4

[10]. W. Cheung, S. Sudhakaran, G. Howells, K. McDonald-Maier, University of Essex, University of Kent , "UML and MDD Design Approach for High Speed Digital Systems with USB3.0 as Case Study", Proceedings of The UK Electronics Forum ,June,2010.

[11]. OMG (August 2006), "UML Profile for System on a Chip (SoC)", http://www.omg.org/technology/documents/formal/profile_soc.htm

[12]. UML, (February 2009), "UML superstructure Specification Version 2.2", http://www.omg.org

[13]. SysML(June 2010), "OMG Systems Modeling Language, (OMG SysML™) Version 1.2", http://www.omg.org/spec/SysML/1.2/

[14]. D. H. Akehurst, B. Bordbar, M. Evans, W. G. Howells, and K. D. McDonald-Maier, "SiTra: Simple Transformations in Java", ACM/IEEE 9th Int. Conf. on Model Driven Engineering Languages and Systems, Genova, Italy, 2006

[15]. J. S. Cuadrado and J. G. Molina, "Build Domain-Specific Languages for Model-Driven Development", IEEE Computer Society, IEEE Software, , pp. 48-55, Sep/Oct 2007

[16]. Keating, M., "Measuring design quality by measuring design complexity", Proceedings. IEEE 2000 First International Symposium on Quality Electronic Design, 2000

[17]. Visual Paradigm for UML 7.2 Enterprise Edition , http://www.visual-paradigm.com

[18]. Universal Serial Bus 3.0 Specification Revision 1.0, November 2008, http://www.usb.org/developers/docs/

[19]. Aldec (2010), "Active-HDL 8.3" http://www.aldec.com/Products/default.aspx