社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

既存設計の再利用を考慮したSoCの仕様記述手法と上位設計方法論

李　蓮福[†]　石川　悠司[†]　小島　慶久[†]　吉田　浩章[††]　余宮　尚志[†††]
小松　聡[††]　藤田　昌宏[††]

† 東京大学大学院工学系研究科電子工学専攻　〒113-8656 東京都文京区本郷 7-3-1
†† 東京大学大規模集積システム設計教育研究センター　〒113-0032 東京都文京区弥生 2-11-16
††† 株式会社東芝 ソフトウェア技術センター　〒212-8582 神奈川県川崎市幸区小向東芝町 1
E-mail: † {lee,yuji,kojima}@cad.t.u-tokyo.ac.jp,
†† {hiroaki,komatsu,}@cad.t.u-tokyo.ac.jp,fujita@ee.t.u-tokyo.ac.jp,
† † † hisashi.yomiya@toshiba.co.jp

あらまし　近年のSoCの設計は、複雑さの急激な増大と、開発期間の圧迫によりますます難しくなりつつある。さらに、記述手法のばらつきと、設計段階間のギャップの存在により起こる仕様に対する誤解は、さらなる損失を招く原因となっている。このような状況を解決するための最も議論が進んでいる手法として、設計レベル抽象化と設計再利用がある。そして、それらの手法においてUMLは最も効率的な記述手段と考えられる。本稿では、我々は要求の発生とCベース言語などで記述されたシステムレベル設計間をつなげられる設計フローを提案する。提案設計フローは設計再利用の概念も含んでいる。ケーススタディとして、デジタルカメラの設計例を用いることにより設計段階間のギャップを解消できることを示し、提案手法の有効性を述べる。

キーワード　SoC、設計仕様、UML

Specification Description and High-level Design Methodology of SoC Considering Design Reuse

Yeonbok LEE[†], Yuji ISHIKAWA[†], Yoshihisa KOJIMA[†], Hiroaki YOSHIDA[††], Hisashi YOMIYA[†††], Satoshi KOMATSU[††], and Masahiro FUJITA[††]

† Department of Electronics Engineering, School of Engineering,University of Tokyo,
Hongo 7-3-1, Bunkyo-ku, Tokyo, 113-8656,Japan
†† VLSI Design and Education Center, University of Tokyo
Yayoi 2-11-16, Bunkyo-ku, Tokyo, 113-0032,Japan
††† Corporate Software Engineering Center, Toshiba Corporation
Komukai-Toshiba-cho 1, Saiwai-ku, Kawasaki, 212-8582, Japan
E-mail: † {lee,yuji,kojima}@cad.t.u-tokyo.ac.jp,
†† {hiroaki,komatsu,}@cad.t.u-tokyo.ac.jp,fujita@ee.t.u-tokyo.ac.jp,
† † † hisashi.yomiya@toshiba.co.jp

Abstract　The development process of SoC is getting harder owing to the relentless rising complexity and time–to-market pressure. What is worse, the misunderstanding of specification due to the varied writing styles and gaps between each design level causes additional loss of time and cost. One of the most discussed methods to cope with this situation is design abstractions and design reuses. On those strategies UML is drawing a lot of notice as the most well-chosen description method. In this paper, we propose a design flow, which connects the gap between the requirements and actual design description in system level. The flow involves and activates the concept of design reuse. As a case study, we designed a digital camera which has limited functionalities in order to show the proposed flow works well and successfully closes the gap between requirement statements and actual designs.

Key words　SoC, Specification, UML

1. Introduction

In recent decades, we have seen the spectacular progress of semiconductor technology in accordance with the Moore's prediction, which has resulted in the explosive growth of the number of transistors integratable on a single chip. Today, it has even come to feasible to integrate an entire system on a single chip, called system-on-a-chip(i.e., SoC), consisting of billions of transistors.

Meanwhile, the demands for more functionality, higher performance, and smaller size upon the electronic products of has been also rising significantly. Thus, to survive the market competition, the time-to-market became enormous pressure for the product developers.

However, the increasing *complexity* of electronic designs forces the effort, design period, and cost required to a design to increase accordingly or even exponentially. It is even said that the design scale has almost reached the limit that one engineer can design in the same period by the conventional design methods. What is worse, if the developing groups try to increase just the number of engineers to save total design period, it results in the increase of not only the cost for employment but also the possibility of the miscommunication among the engineers which is come from the varied writing styles, description flaws, and varied description languages. It is apparent that the later design fault that is caused from misunderstanding of specification is detected, the bigger loss would be invited on the developing cost and time which is consumed to redesign that portion. In addition, the *gap* existing between the requirement analysis and actual design further aggravates the communication miss and the inconsistency between the requirements and implementation.

To cope with the aforementioned situation, furthermore efficient methodologies and tool support which can improve the design productivity by dealing with the rising complexity and addressing the miscommunication problem. So far, the concepts of design level abstraction and design reuse, have been believed to be the most effective solutions, and all of them are strongly related to the other.

Among them, abstraction of design level incorporated in the CAD (Computer Aided Design) technology has played an important role in improvement of design efficiency. A level of abstraction can be determined by the objects that can be manipulated and the operations that can be performed on them. Every level is deeply concerned with the description languages and the existence of the tool environments which can process it. So far, the highest design level of hardware applied practically in chip design is RTL(Register Transfer Level). The rest can be classified into gate level and physical layout. Furthermore, the higher abstraction levels such as *behavior level*, *system level* have been studied by many researchers, and hundreds of research results regarding them have been reported. As an remarkable example, the study from NEC [1] showed that the code density(in terms of line counts) can be improved by nearly 10 times when moved from RTL to *behavior level*. However, there still exists the *gap*, between the given requirement and that system level design. In other word, the method how to analyze the requirement statements and derive the consistent design description from it. Here, the key as the solution, is *specification*. Specification method for hardware is still behind different from that for software.

On the other hand, design reuse has also not been widely adopted. There are many reasons for it, but the variations that prohibit the adoptability, and the lack of the search and verification methods are regarded as the considerable reasons. To reuse existing components, designers need to decide whether the object is reusable or not by carefully examining its specification documents such as datasheets, or even its implementation data such as RTL descriptions manually, in many cases, such documents are described in various non-standardized formats with natural languages and figures. It is time consuming process, thus to establish a machine readable as well as standardized specification description method for those existing design is indispensable.

In this paper, we target to propose a solution for following problems: miscommunication problem, existence of the gaps between requirements and design, and lacking of methods facilitating efficient design reuse. To accomplish the purpose, we propose to introduce UML (Unified Modeling Language) [2] as a specification description language for current design as well as existing designs, and propose a methodology for analysis and refinement of requirement combined with the concept of reuse.

In section 2., we briefly explain our proposed high-level design flow, followed by section 3. which presents the details of description methods for each step. Then in section 4., we demonstrate the case studies to show the availability of the proposed methods, and finally conclude the discussion and present future directions in section 5..

2. High-level Design Flow

Before considering the specification, we firstly proposed a target design flow concerning the high abstraction levels. This work is largely based on the previous study conducted by Matsui [5], which aims to present an analysis and refinement method from the requirement to the system level design, and to propose an appropriate documentation method of existing design. In that work, the entire design flow mainly consists of two phases, *analysis phase* and *implementation*

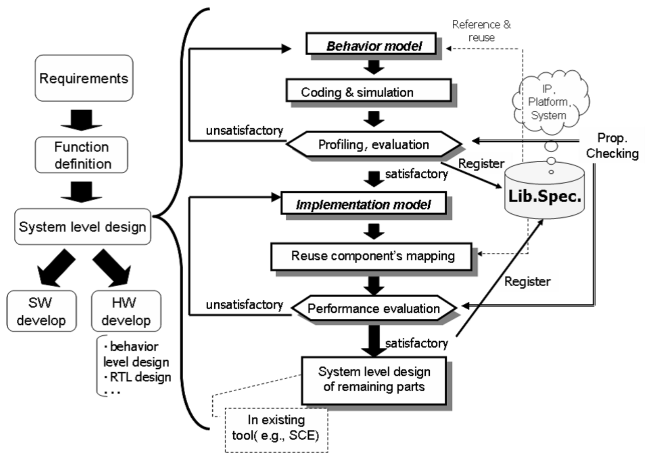


Figure 1 Overall design flow

phase. In *analysis phase*, the designer determines functional organization and behaviors on it through analysis of given requirements. In *implementation phase*, the designer establishes the template that represents physical architecture and maps physical components which are stored in PE library onto the template. The description languages adopted for those phases are UML.

UML is a visual, object-oriented, and multi purpose modeling language, standardized by OMG(Object Management Group) [3]. Due to its capability of visualization, designers can understand more intuitively about current design description, and since it is standardized, it helps the common understanding. Those facts helps us to reduce the communication miss and to improve the design efficiency.

As the output of the design flow, a system level design which is described in a system level language(*i.e.*, SpecC [4]) is generated from those structural and behavioral descriptions through manual coding.

However, in that work, the boundary among the two phases is not unclear, and it is ambiguous what process the designer should undertake in each phase. In particular, the naming, "analysis" seems to be the main factor that results in the ambiguity, since analysis can include implementation information, and implementation work requires also analysis. Actually they do, in the case study shown in the work.

Thus, in this study, inheriting the fundamental concepts of Matsui's work, we revise and improve the design flow and propose a description and refinement methodology for each design step. The overall design flow we proposed is shown in figure 1, which starts from the input of *Requirement statements*, and output the *System level design* description, thereby filling the gap between them successfully.

The flow largely consists of 3 levels, *function definition*, *behavior modeling*, and *implementation modeling*. In many cases, the terms, *function* and *behavior* are easy to be confused with each other. Thus we defined the usages of them in advance as below:

- *function*: A service that the target system provides to outside objects
- *behavior*: The interactions and interrelationship among the objects inside the target system to achieve certain purpose(*i.e.*, *function*)

Hereafter, the explanation about the work in each steps that appear in the flow is presented.

A development process starts with the genesis of requirements against the target system. Given the requirement statements, which are generally described in natural languages in an arbitrary format, each of them should be analyzed and distinguished into either a functional one or a non-functional one for analysis of functionality. Among them, only those concerning the functionality are selected and summarized as a function definition model. The remaining non-functional items are also to be kept for later use. In *function definition* model the relationships between each function as well as all the functions the system serves to outside objects must be clear.

The next step after all the functionality is defined explicitly in the *function definition model* is to model and specify the details of behaviors and structures of the target system. This stage is done throughout two modeling phases: *behavior model* and *implementation model*.

First of all, in the *behavior modeling* phase, the designer should make it clear how each function appeared in the *function definition model* can be realized throughout stepwise refinement of the internal behaviors. In this stage, the structure and performance condition are not needed to be considered. Instead, the algorithms for each function should be refined and defined as detail as it can be.

Then, *implementation modeling* would be conducted through structuration of internal objects, with setting a particular platform or mapping of IPs which are registered in a certain repository for reuse. The evaluation work should be accompanied with this process to make the target structure optimized.

The outcome obtained when all those modeling processes are finished is to be the conventional system level design description which contains architectural information as well as internal behaviors. That description can be adopted as an input design directly in the existing system-level design environments(e.g., SCE [6]).

The entire process presented here is not just straightforward but allows and needs backtracking in order to have trial-and-error processes in refinements.

Table 1 Description method used for modeling

Specification model	Description method
Function definition model	UML: Use Case Diagram State Machine Diagram
Behavior model	UML: Class Diagram UML: Sequence Diagram
Implementation model	UML: Class Diagram UML: Sequence Diagram UML: Composite Structure Diagram
Coding language	SpecC

3. Specification and Documentation

It is needless to say that a design abstraction level is deeply related to and depends on the description language and its semantics. In this work, we mainly adopt basically UML as a primary description language for each models and SpecC as a coding language corresponding to UML descriptions respectably. Table 1 presents the usage of description languages of each model.

Hereafter, the detailed explanation of the description methods for each modeling step is followed.

3. 1 Functional definition model

First of all, it is needed to define the boundary of target system, and to clarify the functions, related outer objects, and relationships. Those information are analyzed and described by depicting a Use Case Diagram of target system. Describing Use Case Diagram enables us to visualize the relationships between functions, as well as between functions and actors which (or who) has any interactions with target system. At this point, it is occasionally an arguable but important task to define the abstraction level of functions. To make it clear, we recommend following process described below for establishing function definition model.

(1) Define the system boundary and related actors. Actors can easily be found by checking the noun type expressions that play roles of subject or object in the given requirement statements.

(2) Select the functional requirement statements which are expressed usually in verbal type notations.

(3) Assign one function to one Use Case.

(4) Among them, leave only those functional notations(*i.e.,*Use Cases) which present the services that the target system provide to actors, and omit those not directly accessed from actors.

Optionally, we can also check the adequateness of the *function description model* by describing the State Machine Diagram the each state of which is an abstract states and corresponding to a function(*i.e.*, Use Case in Use Case Diagram). By describing the transitions and conditions between the functions, we can easily find the flaw of function and check the relationships between functions. In this task, all the combinations of functions which can be performed in parallel process should be expressed by a state which is the product of those states(=functions).

3. 2 Behavior model

The description of behavior model can be treated to be sheer the same level as the software program code development. Thus, introducing UML diagrams to the behavior modeling stage can be considered as a fairly plausible strategy, since the effectiveness of which on the improvement of software development efficiency combined with the object oriented methodology is proved already.

However, we must notice in advance that concept of the object oriented programming should be adopted carefully when we deal with the hardware and software heterogeneous system. That is, such concepts as *inheritance* and *polymorphism* is not appropriate for hardware design because there are not many existing tool supports for processing those descriptions of hardware design. Therefore, if there exists any description of those concepts, it must be corrected to plane description for the portions which are decided to be implemented as hardware components, at least on the later implementation modeling step. On the other hand, the concepts of *encapsulation* and *modularity* are recommended to be applied strongly to promote the reusibility of the description.

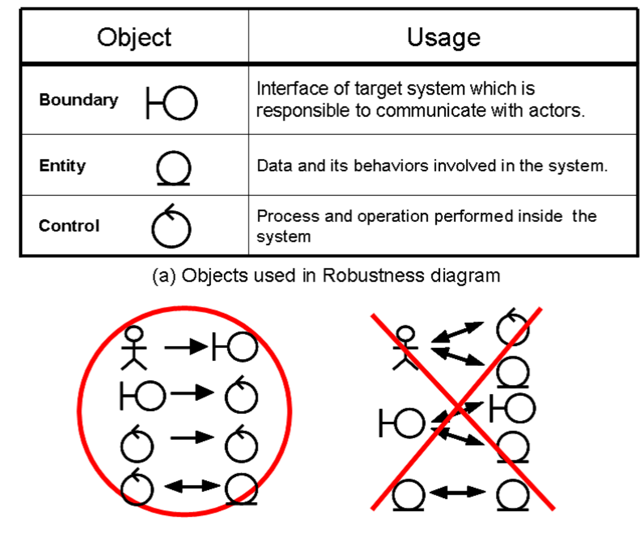
In *behavior modeling stage*, we define and refine only the behaviors not considering any physical or implementation conditions such as timing. As a matter of fact, to derive detailed behavior directly from the requirement statements is a difficult and thorny work. Therefore a method for drawing a rough sketch of the target system behavior, namely *Robustness analysis* [8] can be introduced as a intermediate model, if desired.

Robustness analysis is a method to analyze target system's behavior using also graphical expressions consisting of 3 kinds of symbols which are depicted in Figure 2 combined with actors and Use Cases. The diagram used for analysis is called Robustness Diagram, which is not regarded as a subset of UML diagrams but basically simplified version of UML Communication and Collaboration Diagrams.

The refinement work of *behavioral modeling* is done by describing UML behavioral descriptions (*i.e.*,Sequence Diagram and Class Diagrams), coding them to SpecC code. Through compilation, simulation, and evaluation of the SpecC code, the designer can judge the plausibility of the current design.

3. 3 Implementation model

The *implementation model* is constructed through the gradual structuration of the *behavior model* description and



Object	Usage
Boundary	Interface of target system which is responsible to communicate with actors.
Entity	Data and its behaviors involved in the system.
Control	Process and operation performed inside the system

(a) Objects used in Robustness diagram

(b) Relationships which are allowed (left side) and not allowed (right side)

Figure 2 Robustness Diagram

mapping of exist designs(*i.e.*, platform, Intellectual Properties(IPs)) on current design. Specifically, for this model, we modify the structures of Class diagram and Sequence diagram, and describe the interrelationship of internal architecture using Composite Structure diagram. In this stage, SpecC coding and evaluation of it is also helpful to optimize the structure.

At this point, as shown in figure 1, a repository, in which the specification description documents of exist designs are registered, is assumed (e.g., supposed to be the result of our previous work [7] which proposed an efficient specification method of reusable components based on UML and XML description and a library system which is achieved by a tool development). Since the design description methods of current design and the specification description of those reusable components are highly corresponding, reuse and mapping process become facile.

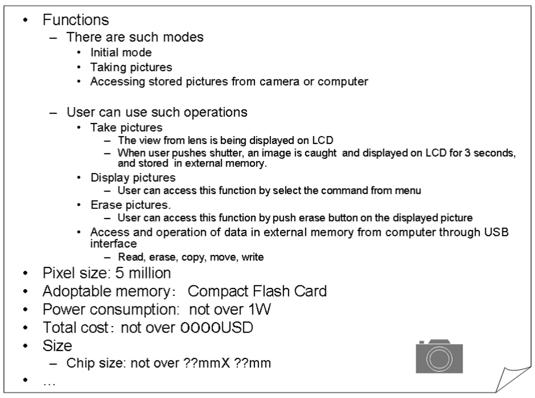
3.4 Description outcomes

The outcome of this targeting part of entire flow, namely specification analysis and refinement process is descriptions which present the behavior and implementation information for whole system and include the all UML Diagrams and SpecC code descriptions. Those description records of current design must also be registered after finished whole development process including verification and test into the reuse repository.

4. Case Studies

To show the availability and plausibility of our proposed design flow and specification description method, we demonstrate two case studies as presented below.

- **Case study1** : Modeling an SoC for a simple digital camera (ver.1)
- **Case study2** : Modeling an SoC for another digital camera (ver.2) by adding one more function onto the previous version of camera (ver.1) utilizing the specification description documents of it.



- Functions
  - There are such modes
    - Initial mode
    - Taking pictures
    - Accessing stored pictures from camera or computer
  - User can use such operations
    - Take pictures
      - The view from lens is being displayed on LCD
      - When user pushes shutter, an image is caught and displayed on LCD for 3 seconds, and stored in external memory.
    - Display pictures
      - User can access this function by select the command from menu
    - Erase pictures.
      - User can access this function by push erase button on the displayed picture
    - Access and operation of data in external memory from computer through USB interface
      - Read, erase, copy, move, write
- Pixel size: 5 million
- Adoptable memory: Compact Flash Card
- Power consumption: not over 1W
- Total cost: not over 0000USD
- Size
  - Chip size: not over ??mmX ??mm
- …

Figure 3 Requirements for digital camera

First of all, the requirements of the target camera for Case study 1 are shown in Figure 3. Those requirements were input to the proposed design flow. From the requirements, we could select the functional ones, <take picture>, <access pictures>, <displays pictures>, <erase pictures>. All of the functional requirements and the related actors were described in a Use Case diagram as shown in Figure 4. The following works in accordance with the proposed flow could be conducted without crucial problems, as a result the SpecC description which could be the input of existing system level design environment tool, namely SCE [6] was successfully generated.

Then, we tried the second design example Case study 2, with an additional function, that is, the capability of taking and processing motion pictures. This time, we utilized the existing design documents of Case study 1. In detail, we started from the descriptions of Case study 1, added to them the new function, and refined them as if required.

It is impossible put all the diagrams and SpecC codes generated in the Case Studies, thus we present the summary of total efforts to describe all the design specifications in the proposed method by table 2 and table 3.

From the results, we can discuss the effectiveness of our method as follows:

- It was possible to fill the gap between requirements and implementations
- It was possible to describe the specification and analyze and refine it in a standardized language, and to register appropriate documents
- It was possible to design new one with far less effort by activating the registered descriptions of previous designs

By conducting those case study, we could not only get

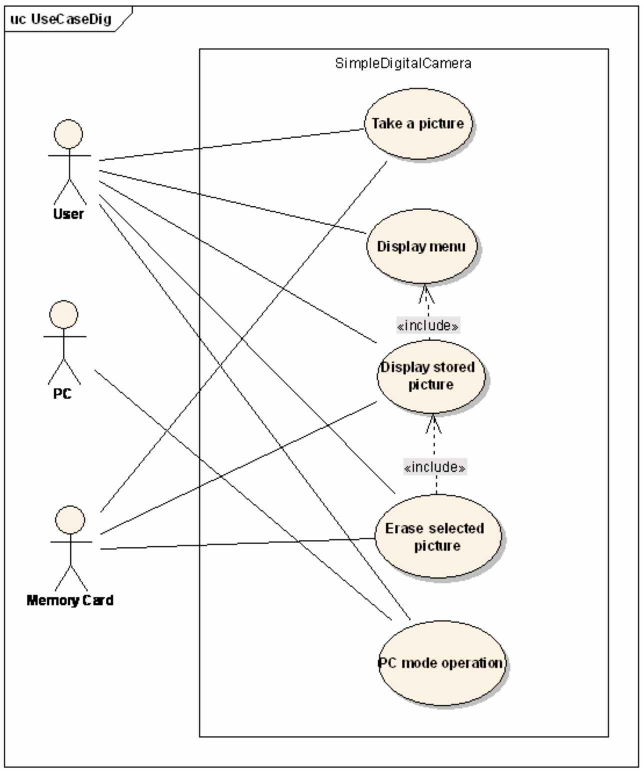


Figure 4 Use Case Diagram for camera of Case study 1

Table 2 Numbers of UML diagrams used for the Case study 1

	# of (finally employed) diagrams	Ratio of work time
Use Case Diagrams	(5)8	10%
State machine Diagrams	(4)4	5%
Robustness Diagrams	(14)20	15%
Class Diagrams for behavior model	(13)15	15%
Sequence Diagrams for behavior model	(15)24	25%
Class Diagrams for implementation model	(5)5	10%
Sequence Diagrams for implementation model	(5)13	15%
Composite Structure Diagrams for implementation model	(3)6	5%
Total	(64) 95	3 man-month

Table 3 Numbers of UML Diagrams used for the Case study 2

	# of (finally employed) diagrams	Ratio of work time
Use Case Diagrams	(2)3	5%
State machine Diagrams	(3)3	5%
Robustness Diagrams	(10)15	13%
Class Diagrams for behavior model	(4)6	7%
Sequence Diagrams for behavior model	(10)14	15%
Class Diagrams for implementation model	(2)3	15%
Sequence Diagrams for implementation model	(12)15	30%
Composite Structure Diagrams for implementation model	(3)3	10%
Total	(48) 62	1 man-month

those quantitative data, but also reach the agreement that the specification method supported by UML can greatly helpful to derive the common decision on current design. Thus, considering the growing complexity and increasing constraints of present design, we can evaluate our methodology to be able to be effective on improving the design efficiency.

5. Conclusion and Future Works

In this work, we propose a high-level design flow and specification description method to cope with the problems of the gap between the design levels and the miscommunication. The proposed design flow is supported by the proposed methodology of analysis and refinement process, which has the advantages of adopting UML to model and specify a target system. Due to its high comprehensibility and common understandability with a unified semantics, the miscommunication problem can be properly solved. Furthermore, by registering appropriate documents of the design descriptions we can enhance the efficiency of design reuse. Throughout the demonstrated case studies, not only the availability of the proposed methods but also its high efficiency of analysis, modeling, refinement ability are shown.

To show the effectiveness of our method in practical cases and to find the issues should be improved, more experiments with more complex system are needed. As another future direction, we should also consider the method to apply the IP reuse methodology in our previous work [7] which uses the compatible specification method.

References

[1] K.Wakabayashi, "C-Based Behavioral Synthesis and Verification Analysis on Industrial Design Examples ", In Proc. ASPDAC, pp. 344-348, Jan 2004.
[2] UML(Unified Modeling Language): http://www.uml.org
[3] OMG(Object Management Group): http://www.omg.org
[4] D. Gajski, J. Zhu, R. Doemer, A. Gerstlauer, and S. Zhao, *SpecC: Specification Language and Methodology*, Kluwer Academic Publisher, March 2000.
[5] Ken Matsui, Masahiro Fujita, "Object-Oriented analysis and specification for HW/SW co-design with UML diagrams, " the IASTED, International Conference on Advances in Computer Science and Technology (ACST), Puerto Vallarta, Mexico, pp.38-43, Jan, 2006.
[6] S.Abdi, J.Peng, R.Doemer, D.Shin, A.Gerstlauer, A.Gluhak, L.Cai, Q.Xie, H.Yu, P.Zhang, D.Gajski, "System-on-Chip Environment (SCE):Tutorial ", CECS, UC Irvine, Technical Report CECS-TR-02-28, Sep 2002.
[7] Y.Lee, Y.Ishikawa, S.Kang, G.Park, S.Watanabe, K.Seto, S.Komatsu, H.Hamamura, M.Fujita, "UML-based Specification Method of Hardware IPs for Efficient IP Reuse ", UML-Workshop, Design Automation Conference, pp. 23-30, Jun 2007.
[8] Doug Rosenberg, Matt Stephens, Use Case Driven Object Modeling with UML: Theory and Practice, Apress 2007.