2007 1st Annual IEEE Systems Conference
Waikiki Beach, Honolulu, Hawaii, USA April 9-12, 2007

AN INTEGRATED METHODOLOGY FOR QOS DRIVEN REUSABLE COMPONENT DESIGN AND COMPONENT SELECTION

Ankur Agarwal, Georgiana Hamza-Lup, Ravi Shankar, James Ansley
Department of Computer Science & Engineering
Florida Atlantic University
777 Glades Road, Boca Raton, FL 33431
{ankur, ghamza, ravi} @ cse.fau.edu, jansley@fau.edu

Abstract - Embedded Systems complexity has been increasing drastically in recent years. This had a negative impact on the product development cycle (PDC). The increasing software and hardware concurrency, and the need for architectural optimization, have also complicated the PDC. One way of enhancing system productivity is to exploit the principle of "design-and-reuse" to its full potential. Component-based design is one effective way of implementing "design-and-reuse". However, given a library of components with varying performance parameters and quality-of-service metrics, it is often difficult to select the right components that will satisfy the system requirements. In this paper, we develop an automated process of component selection. We further demonstrate the proposed methodology by applying it for component selection for a network-on-chip (NOC) architecture.

INTRODUCTION

A system design process is inherently complex. The design involves multiple representations, multiple (design) groups working on different design phases, and a complex hierarchy of data and applications [1]. The different groups bring different perspectives towards system design. System or product inconsistencies arise primarily due to the lack of appropriate communication among various design teams. For example, the concerns of a hardware (HW) design engineer are different from those of a software (SW) designer, and very often, they are unable to understand and address each others problems [2]. Such issues lead to an increased PDC length and product development cost, thereby reducing system design productivity [3].

To enhance this declining productivity, one will have to exploit the principle of "design-and-reuse" to its full potential [4]. This will require us to design and develop a component library. Then, a system (subsystem) would be composed of reusable sub-systems (components). For example, a Network-on-Chip (NOC) architecture can serve as a reusable communication sub-system for an embedded device [5]. This NOC architecture may comprise of components such as, routers, input and output buffers, network interfaces, switches, virtual channel allocators, schedulers and switch allocators. To develop a system from such reusable components, one will have to design and develop variants of each component. For example, buffer size is a customizable parameter for an input buffer. Similarly, scheduling criteria provides customizability for a scheduler. For these components to be reusable, we should properly define their attributes, methods, concurrency signals, customizable parameters, and performance characteristics [3]. Attributes are the inputs and outputs of a component, methods represent functions of a component, customizable parameters refer to possible customization that can bring a change in component performance, concurrency signals refer to interfacing signals and conditions for composing systems (subsystems), and performance characteristics refer to QoS metrics such as, area, power consumption and latency. The performance parameters will have to be abstracted using either a bottom-up annotation process such as, SW estimation, SW emulation, and rapid-prototyping, or a top-down annotation process such as subsystem modeling or literature survey [6]. Thus, for components to be used as reusable building blocks, we must properly define their interfaces and encapsulate their performance metrics for various useful parameter combinations, in order to

1-4244-1041-X/07/$25.00 ©2007 IEEE

help the architect make informed decisions. One should use a middle-out design philosophy, where the system is modeled using the top-down approach from a system specification document, and performance parameters are abstracted using the bottom-up or top-down approach.

We need to address this problem in two parallel phases. The first phase will model a reusable component library. The second phase will model an automatic process for selecting, from the component library, a subset of components that satisfy a set of given functional and QoS performance requirements. Such a model would allow architects to make informed decisions that could reduce the scope of the multidimensional search in the design space. Such decisions (like, choosing the number of processors, HW/SW partitioning or reusing specific software or hardware components) have the potential to significantly enhance the productivity of the system design.

In this paper, we present an integrated methodology for system design that addresses three main concerns: (1) designing reusable component(s); (2) developing a middle-out philosophy for abstracting performance parameters for annotation; (3) developing an automated process of component selection.

INTEGRATED SYSTEM MODELING METHODOLOGY

Figure 1, shows the basic overview of this integrated methodology. As seen in this figure, we propose to develop a concurrent system model from system requirements and specifications. Such a model will define component(s) interfaces and will also help in separating the design concerns, by properly partitioning the system into sub-systems and components. We have discussed this aspect in detail in [3], [7], [8]. Once the system components and interfaces are identified, a system must be modeled in a system level modeling environment such as MLDesigner (MLD). MLD will allow making system level trade-offs among cache size, number of processor, HW-SW partitioning, etc. [9]. Performance numbers can be annotated and fed back into MLD modeling for making these key design decisions [6]. We refer to this part of the flow as the component modeling flow. Component selection flow is another important aspect. We must be able to select the right components from the component library, such that to satisfy system requirements and specifications. In Figure 1, we show two different techniques (Semantic Web, and a Greedy Approach) for the component selection process. We propose to use non-linear regression analysis to construct dependency relationships between various component parameters and performance characteristics. This dependency can be used to impose a certain order among components, in terms of their performance characteristics, and allows a more informed search through the component library. One can initially develop a non-linear multiple regression model. These equations must be able to specify performance parameter as a function of various component parameters. For example, for a NOC architecture, we can define latency as a function of buffer size, routing algorithm, and scheduling criteria. Similarly, area and power consumption can also be defined as a function of different component parameters. Such a regression model will provide one with a rough estimate of the performance dependency with respect to the component parameters. Therefore, a system architect can observe the parameter which impacts the most the system performance and use this information when selecting components. For example, we might see that increasing the buffer size from 1 to 5 decreases the system latency and increases the area linearly; but increasing the buffer size from 5 to 10 does not impact the latency, although it still increases the area linearly. Based on this information we can narrow down our component search for buffers of size up to 5. In this paper, we briefly discuss both proposed techniques for component selection but limit our detailed discussions to the Greedy approach for component selection.

Component Selection using Semantic Web

Semantic Web utilizes document layering to provide differing levels of context to information. Using these layers to provide context to database information allows for more efficient and intelligent database queries. One of the applications of these queries is to search component databases for the reuse of components in new designs. Two possible approaches are as follows. The first approach uses web based user interfaces with Java's JENA framework, it applies Semantic Web to an XML document for querying, OWL for component definitions and a super site that

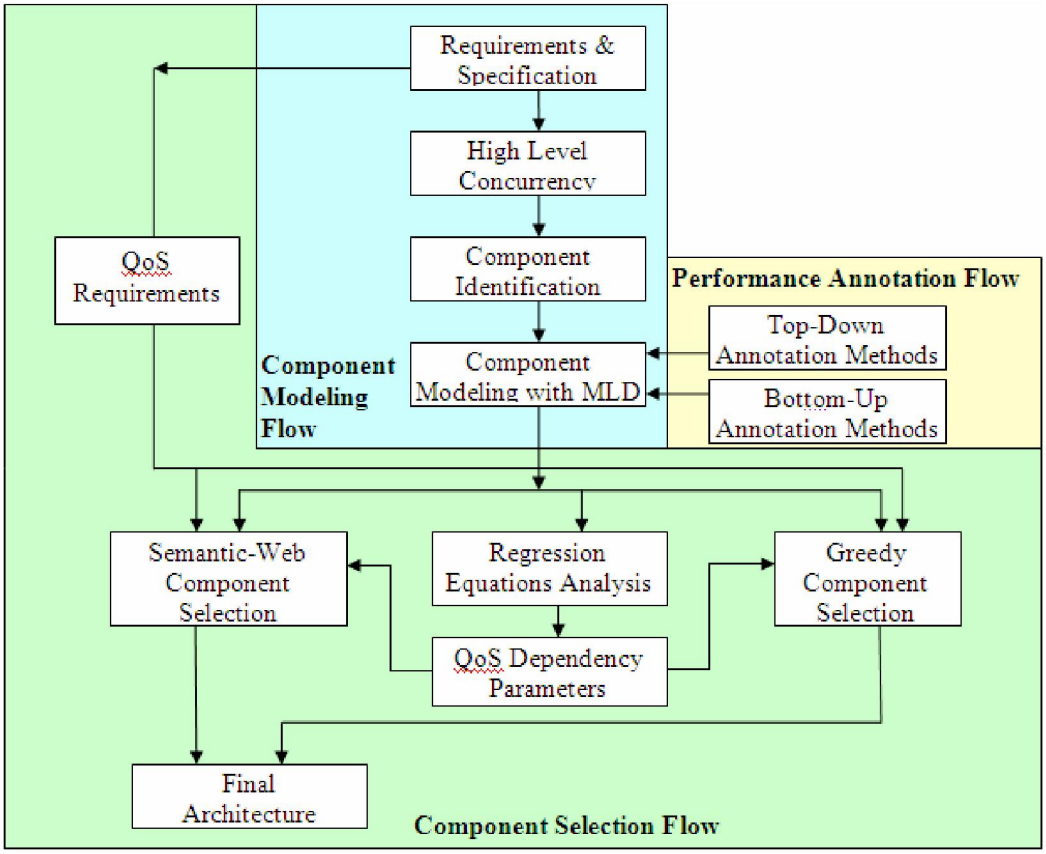


Figure 1. An integrated methodology for designing a reusable component library and for component selection

combines the query and the definitions to create multiple configurations based on available components, query criteria and local restrictions. The second approach uses Eclipse's SWeDE (Semantic Web Development Environment) and IODT (Integrated Ontology Development Toolkit) plug-ins to interact with the same files. In the first approach, the super site uses an OWL document to define a configuration as a grouping of one or more components that meet a query. It then uses the definitions of a component and a query, from the two other sub-sites, to clarify the definition of a configuration. Then the site is presented with a query and a list of components to select from, and configurations are produced. The query and the component sub-sites are there own separate entities. The query is based off of an XML document that contains the search criteria and an ontology that defines what a legal query is. The component site is based off of component ontology, obtained by either querying or iteratively filtering a database of components, and allows for the addition and removal of new components. This approach shows the power of Semantic Web technologies in several key areas. First the component site uses OWL ontology to create its component references demonstrating the creating and storage ability. The super site displays knowledge integration by utilizing the definitions stored in the query and component sites to define it. This also shows limited inference capability.

A Greedy Approach for Component Selection

Component selection is the problem of choosing a subset of components from a set of components, such that the resulting system satisfies certain functional requirements and QoS performance requirements. In this paper we assume that each component satisfies exactly one functional requirement and is characterized by a set of customizable component parameters specific to the functionality assured. For example, a buffer is characterized by buffer size. The values of these component parameters determine the performance (the values of the QoS parameters) of the integrated system of

components. Therefore, our component selection problem becomes selecting one component from each set of components satisfying a certain functional requirement, such that when integrating the selected components the resulting system satisfies the given QoS requirements.

Formally, we can express our component selection problem as follows. Let F be a set of n functional requirements: $F = \{f_1, f_2, \ldots, f_n\}$. Let S_1, S_2, ... S_n be distinct sets of components, each set S_i containing all components that satisfy the functional requirement f_i. Components in a set S_i are assumed to be distinct, that is, they have distinct component parameter values. Let R be a set of given QoS requirements. Our component selection problem can be expressed as finding n components, $c_1 \in S_1$, $c_2 \in S_2, \ldots, c_n \in S_n$, such that the system resulting by combining c_1, c_2, .., c_n satisfies all the QoS requirements in R.

Finding the optimal combination of components requires an exhaustive approach in which all potential combinations are evaluated. However, due to the very large size of the search space, such an exhaustive approach is unfeasible, as it has a very high computational cost [11]. Greedy algorithms can be used for approximation [12], as they have lower computational complexity. For our problem, a Greedy approach would involve choosing at each step the component that "seems" (at that point) to be the most helpful in achieving our performance objectives. If at any step we detect that the objectives (QoS requirements) can no longer be satisfied based on the choices of components made so far, backtracking is required in order to allow us to follow a different path in our search space. Backtracking also enables finding more than one solution. Different heuristics can be used in determining the "best" component to be chosen at each step. Naïve approaches will result in longer searches as they would require a lot of backtracking. In our initial work we propose an approach in which we construct a heuristic based on the dependency between the component parameters and the QoS parameters. The purpose of this dependency is two fold: first, it can be used to eliminate components that do not satisfy certain QoS requirements, thus reducing our search space, and secondly, it can be used to impose a certain order among components, in terms of their performance, that would allow a more informed search through the candidate solutions space. The Greedy algorithm proposed in this paper is given in Figure 2.

In our approach, we classify our requirements based on their dependencies on the component parameters in two categories: component-level and system-level. The component-level category includes requirements directly related to the parameters of a component, and the system-level category includes requirements related to a combination of components. Component-level requirements are used in the Greedy search to select the "best" component to be chosen at each step. Specifically, a component-level requirement can be used to eliminate some of the components

1. Divide the set of requirements R into two sets: L, containing component-level (low-level) requirements and H, containing system-level (high-level) requirements
2. For each set of components S_i corresponding to a functional requirement f_i do:
   2.1. For all $r \in L$, if r is directly related to a component parameter do:
      2.1.1. Eliminate those components from S_i that do not satisfy r
      2.1.1. Order the remaining components in S_i according to the magnitude by which they satisfy r and mark them as *unvisited*
3. Set i = 1;
4. While (i <= n) do:
   4.1. Select the first *unvisited* component in the ordered set S_i
   4.2. For all $r \in H$, if the components selected so far do not satisfy r do the following:
      4.2.1. Mark the component selected from S_i as visited
      4.2.2. If there are unvisited components in S_i goto step 4.1.
         Else mark all components of S_i unvisited and i = i-1 and goto step 4.1.
   4.3. i = i+1;
5. If i < n then no combination of components can satisfy the given requirements
   Else a combination of components satisfying the given requirements has been found

Figure 2. Greedy algorithm for component selection

(those that do not satisfy that specific requirements) and/or to impose a certain order on the remaining components according to the magnitude by which they satisfy that requirement. On the other hand, system-level requirements are used to determine the backtracking points in our Greedy algorithm. Specifically, system-level requirements can be used to determine if a certain combination of components, that does not satisfy these requirements, has been reached.

EXPERIMENTAL RESULTS

We illustrate our Greedy approach with a simplified example of selecting the components for a 4x4 mesh based NOC architecture. A 4×4 mesh based NOC is a multi-core architecture with 16 producers and 16 consumers. NOC is based on packet switching protocol [10] and can be used for designing complex embedded systems. NOC can improve design productivity by supporting modularity and reuse of complex cores, thus enabling a higher level of abstraction in the architectural modeling of future systems. Some of the NOC components are: Input and Output Buffer (B), Producer (P), Consumer (C), and Scheduler (S). Each of these components can be modeled either as hardware or as a software process. This NOC architecture offers various performance tradeoffs. For example, one would be able to analyze the impact of a particular routing strategy or scheduling criteria in terms of latency, power consumption and area. In practice, about 60% of the area of the communication backbone will be occupied by the buffers. For a 4×4 mesh based NOC, there will be 160 buffers. Thus, the buffer size is one of the key parameter of NOC architecture. Other key performance parameters are leakage power consumption and system latency for high, medium (mid) and low priority data traffic.

In our simplified example, the two sets of components each satisfying a certain required functionality are buffers and schedulers. The parameters for these components are as follows: size and scheduling criteria for buffers and scheduling criteria for schedulers. We assume 6 possible values for the buffer size: 1, 2, 3, 4, 5 and 10, and 4 possible values for the scheduling algorithm: PB (Priority Based), PBRR (Priority Based Round Robin), RR (Round Robin) and FCFS (First Come First Served). We also assume that the scheduling criteria used by the buffer and scheduler are the same. This results in 6*4 = 24 potential combinations of buffers & schedulers, out of which we are searching for the optimal combination that satisfies certain performance requirements. Our requirements are as follows: first, the buffer size should be no less than 3; second, latency for high, medium and low priority packets should be no greater than 30ns, 35ns and 70ns respectively; and third, the number of gates (or area) used by the selected components should be minimized.

The first requirement, that the buffer size should be no less than 3, is a component-level requirement. It has a direct relationship with the component parameter size of the buffers set, and therefore, it can be used to eliminate immediately those components that do not satisfy it. Our second requirement, that latency for high, medium and low priority packets should be no greater than 30ns, 35ns and 70ns respectively, is related to a combination of components and therefore, it is a system-level requirement. The third requirement, that the number of gates should be minimized, is directly related to the buffer size, as well as, the scheduling criteria. Therefore, it is a component-level requirement and can be used to impose an order within the components in the buffers set and schedulers set. The components in the schedulers set are ordered based on the number of gates required for each scheduling criteria. The component in the buffers set can be ordered either based on the number of gates required for each scheduling criteria or based on the number of gates required for each buffer size. The later ordering is chosen at random for our example. The ordering of components in the buffers and schedulers sets is shown in Figure 3. The red bars indicate that buffers of size 1 and 2 have been eliminated (based on the first requirement).

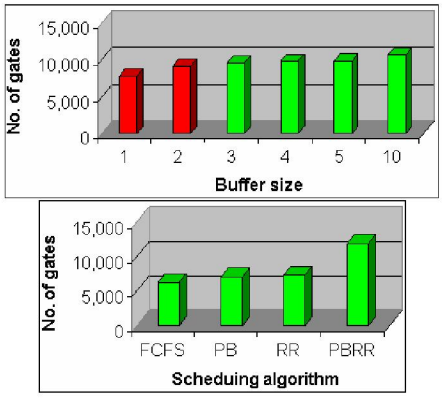
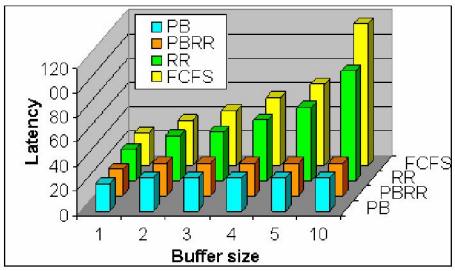
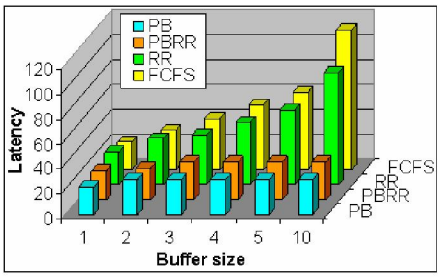


Figure 3. Ordering of components in the buffers & schedulers sets

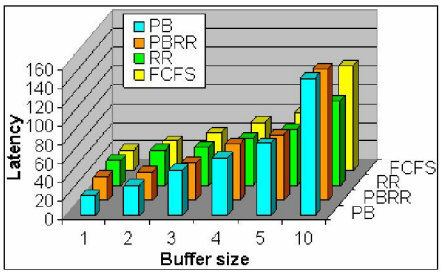
The ordering established based on requirements one and three is used in the Greedy algorithm. Based on that ordering, a buffer with size 3 is selected first from the buffers set and than a scheduler with a FCFS algorithm is selected from the schedulers set. The resulting combination is checked against the system-level requirement. The latency for all combinations of components is presented in Figures 4a, 4b and 4c for high, medium and low priority packets respectively.



(a) Latency for high priority packets



(b) Latency for medium priority packets

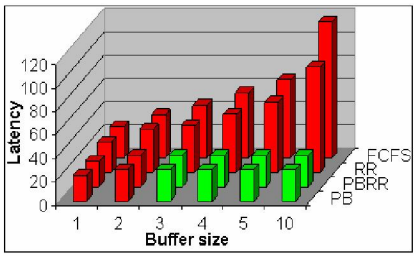


(c) Latency for low priority packets

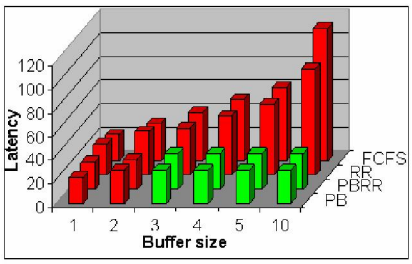
Figure 4. Latency for all combinations of buffers and schedulers

According to the data from Figure 4, the latency requirement is not satisfied by our first combination of components and therefore, our algorithm will backtrack, and the next scheduler in our ordered set of schedulers will be chosen. A scheduler with a PB scheduling algorithm is selected and the new combination of components is checked against the latency requirement. According to the data from Figure 4, this combination does satisfy the latency requirement. Therefore, at this point we have a potential solution. However, this might not be the optimal solution that minimizes the number of gates. The reason is that the components in the buffers set are only ordered based on one dimension, the area occupied by the buffer size and not based on the area occupied by the buffer's scheduling criteria. It is possible that the area of a larger size buffer with a different scheduler is smaller than our potential solution.

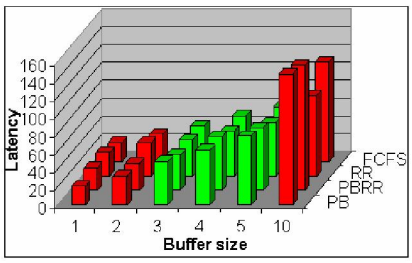
An exhaustive approach would require determining all combinations of components and than checking all combinations against all the requirements. The result of eliminating the components and combinations of components that do not satisfy the buffer size and latency requirements is illustrated in Figures 5a, 5b and 5c for high, medium and low priority packets. The combinations that satisfy the requirements are marked green, those that do not are marked red.



(a) High priority packets



(b) Medium priority packets



(c) Low priority packets

Figure 5. Combinations satisfying the first two requirements (green bars)

The combinations of components that satisfy the first two requirements for all types of packet priorities are shown in Figure 6 as the green bars. Figure 6 also shows the number of gates required by each combination of components. Based on our third requirement, to minimize number of gates used, we can determine our optimal combination that satisfies all our three performance requirements. It is illustrated in Figure 6 as the hashed green bar.

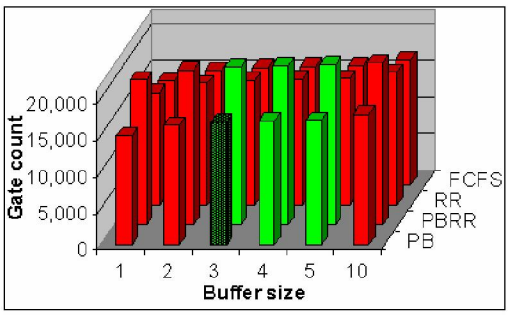


Figure 6. Number of gates used by each potential combination of components

Therefore, in our example the solution obtained with our Greedy approach was indeed the optimal solution obtained with the exact, exhaustive approach. In general, if the given performance requirements result in different potential orderings of a component set, then one ordering will be chosen and used in our Greedy approach, and in that case the obtained solution might not be the optimal one.

CONCLUSION

In this paper, we have proposed an integrated methodology for modeling a system and configuring a system by selecting its subsystems and components from component library. Such an integrated methodology can be used to enhance system design productivity. It will help in analyzing system requirements/specifications and the impact of system parameters in the beginning of system design phase. Therefore, a right decision on component selection can be taken well during system modeling phase. Right selection of component and its parameters along with key design decisions can further avoid design re-spins. Thus, it will further enhance the component re-use capability, thus enhancing system productivity. We demonstrate this methodology with component selection process for a NOC architecture.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge Motorola's support for this research, through its OPP (One Pass to Production) Grant to CSI at FAU.

REFERENCES

[1] G. Desoli, E. Filippi, "An outlook on the evolution of mobile terminals: from monolithic to modular multi-radio, multi-application platforms", *IEEE magazine on Circuits and Systems*, vol. 6(2), pp. 17-29, 2006.

[2] W.C. Rhines, "Sociology of design and EDA", *IEEE Transaction of Design and Test*, vol. 23(4), pp. 304-310, Apr. 2006.

[3] A. Agarwal, "QoS Driven Communication Backbone Design for Embedded Systems", Ph.D. Dissertation, Florida Atlantic University, 2006.

[4] Y. Xiong and E. A. Lee, "An extensible type system for component-based design", *Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, 2000.

[5] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor SoC", *IEEE Trans. on Parallel and Distributed Systems*, vol. 16(2), pp. 113-129, 2005.

[6] R. Shankar, H. Kalva, A. Agarwal, A. Jain, "Annotation Methods and Application Abstraction", to appear in the Proc. of the *IEEE Int'l Conf. on Portable Information Devices*, Orlando, Mar. 2007.

[7] A. Agarwal, R. Shankar, "A Concurrency Model for NOC Design Methodology", *IEEE Conference of High Performance Computing*, Massachusetts Institute of Technology, 2006.

[8] A. Agarwal, R. Shankar, F. Kovalski, "Modeling Concurrency on NOC Architecture with Symbolic language: FSP", *IEEE Int'l Conf. on Symbolic Methods and Applications to Circuit Design*, 2006.

[9] A. Agarwal, R. Shankar, "A Layered Architecture for NOC Design Methodology", *IASTED Int'l Conf. on Parallel and Distributed Computing and Systems*, pp. 659-666, 2005

[10] L. Benini and G. De Micheli, "Networks on chip: a new SOC paradigm", *IEEE Computer*, vol. 35(1), pp. 70-78, 2002.

[11] R.G. Bartholet, D.C. Brogan, P.F. Reynolds Jr., "The Computational Complexity of Component Selection in Simulation Reuse", *2005 Winter Simulation Conf.*, pp. 2472-2481.

[12] M.R. Fox, D.C. Brogan, P.F. Reynolds, "Approximating Component Selection", *2004 Winter Simulation Conf.*, vol.1 , pp. 429-434.