SysCon 2008 – IEEE International Systems Conference
Montreal, Canada, April 7–10, 2008

Component selection strategies based on system requirements' dependencies on component attributes

Georgiana L. Hamza-Lup[1], Ankur Agarwal[1], Ravi Shankar[1], Cyril Iskander[2]

[1] Computer Science & Engineering, Florida Atlantic University, Boca Raton, FL, USA
[2] Hi-Tek Multisystems, Canada
{ghamza, ankur, ravi} @cse.fau.edu, cyril_iskander@hotmail.com

Abstract – *Rapid increases in systems complexity have raised the need to exploit the "design & reuse" principle to its full potential. The proposed research is targeted towards component reuse, specifically towards component selection. We assume a component specification method has been chosen and a component library has been designed and built. The problem we address in this paper is choosing a subset of components from a library of components, such that the resulting integrated system satisfies certain requirements. Our proposed approach contains two main stages. First, we address those requirements that can help us reduce our search space and secondly, we perform an intelligent search in our reduced search space. In the second stage we apply a Greedy approach for selecting components from our reduced search space. The challenge here is assessing how well a certain component satisfies the performance requirements of the target system, as these performance requirements usually refer to the system as a whole and not to individual components. To address this challenge we focused on mapping system performance requirements onto component characteristics. We will illustrate our proposed approach for component selection with a simplified example of selecting the components for a 4x4 mesh-based NOC (Network-on-Chip) architecture.*

Keywords – *Component selection, component reuse, optimal architectures*

I. INTRODUCTION

Rapid increases in systems complexity have raised the need to exploit the "design & reuse" principle to its full potential. The proposed research is targeted towards component reuse, specifically towards component selection. We assume a component specification method has been chosen and a component library has been designed and built. The problem we address in this paper is choosing a subset of components from a library of components, such that the resulting integrated system satisfies certain requirements (functional and non-functional). Depending on the system requirements, the selection process may have two outcomes: either one or more candidate combinations of components are found, or, some requirements cannot be satisfied with the existing components and as a result, either new components need to built, or the system requirements need to be relaxed. In this paper, we assume that when the second outcome occurs, the system requirements are relaxed and the selection process is restarted.

Current approaches for addressing component selection are mainly driven by functional requirements and/or by non-functional system requirements that depend directly on the attributes of the individual components. For example, in [1] and [2] the authors assume that every component satisfies one or more functional requirements and the goal is to select the minimal set of components that together satisfy a set of given system requirements. While [1] and [2] aim at minimizing the number of components of the resulting system, in [3] the cost of the final system is minimized. This non-functional system requirement is directly dependent on the attributes of the system components. Specifically, the cost of the final system is the sum of the costs of the individual components. In [4] reliability and delivery time requirements are specified in addition to cost requirements. Again, these requirements depend directly on the attributes of the system components. Specifically, system reliability, defined as the probability of the system to function without failure, is computed as the product of the reliabilities of the individual components, and system delivery time is computed as the maximum delivery time among all components.

In contrast to existing approaches, our proposed component selection technique is driven by non-functional system requirements, whose dependencies on the components attributes cannot be expressed by exact mathematical formulas. Therefore, our goal is to extract and approximate these dependencies, so that we can predict with certain accuracy the characteristics of a system, from the attributes of the individual components.

The remainder of this paper is organized as follows. In Section 2 we present our requirements categorization and our component selection algorithm. In Section 3, we briefly explain our approaches for extracting the dependencies of the system characteristics on the components attributes, and in Section 4 we illustrate them and the selection algorithm with an example. Finally, we conclude and discuss our future work in Section 5.

II. REQUIREMENTS CATEGORIZATION AND COMPONENT SELECTION ALGORITHM

In our work we assume that each component (hardware or software) is characterized by a set of functional attributes and a set of non-functional attributes that include implementation, interface and performance characteristics. For example, for a

978-1-4244-2150-3/08/$25.00 ©2008 IEEE

hardware component whose functionality is to store data (that is, a buffer), implementation characteristics could include the size of the buffer or the scheduling policies used (e.g., RR: round robin, FCFS: first-come-first-served, PB: priority-based), interface characteristics could be in the form of compatibility information, and performance characteristics could be the number of gates used (area) or power consumption.

The requirements that the integrated system of components need to satisfy can again be categorized as functional or non-functional. Functional system requirements can be easily mapped on the functional attributes of individual components. If a system requirement is that functionalities A, B and C need to be assured, the we should be looking for components with functionalities A, B, C, AB, BC, AC or ABC. Implementation & interface system requirements again translate directly into component requirements and can be mapped onto component attributes. For example, if the system needs to be compatible with platform X, than all individual components need to be compatible with platform X. System performance requirements on the other hand, might not map directly onto component performance attributes. In these situations, our goal becomes approximating the dependency of these requirements on the individual component attributes.

System requirements (functional and non-functional) represent our selection criteria in the component selection process and our approach is to categorize them based on their selectivity as follows: exact matches (e.g., functionality = buffer), range matches (e.g., system latency < 70ns) and optimal matches (e.g., number of gates used should be minimized).

Assuming a large component library, an exhaustive approach in which all potential combinations of components are evaluated against the target system requirements, is unfeasible as it has a very high computational cost [1], [2]. Therefore, our proposed approach for component selection is based on approximation and has two main stages: first, we address those requirements that can help us reduce our search space and secondly, we perform an intelligent search in our reduced search space. In stage one we propose the following order of considering requirements.

1) First, functional requirements are applied, as these are always in the form of exact matches. As a result of this filtering, all components that do not satisfy any of the target system functional requirements are eliminated.
2) Then implementation and interface requirements are considered. If these are in the form of exact or range matches, then they can be used to filter out those components that do not satisfy them. If they are in the form of optimization matches, they can only be used in searching, in stage two.
3) In the third step, performance requirements are applied. However, these requirements usually refer to an integrated system of components rather than individual components. Therefore, only those performance requirements that refer or translate to an individual component and are specified in the form of exact or range match can be applied in this first stage. For example, a requirement that the system reliability has to be greater than 50% can be used to eliminate all components whose reliability is less than that. Note that this requirement will be used again in the second stage.

In the second stage we apply a Greedy approach for selecting components from our reduced search space. Thus, we build the system incrementally, by choosing at each step the component that "seems" (at that point) to be the most helpful in achieving our objectives (the target system requirements). If at any step we detect that the system 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.

Different heuristics can be used in determining the "best" component to be chosen at each step. Our goal is to find the best order in which to evaluate/consider components such that to minimize backtracking and thus the search time. As mentioned in the previous section, the challenge here is assessing how well a certain component satisfies the performance requirements of the target system, as these performance requirements usually refer to the system as a whole and not to individual components. To address this challenge, we focused on mapping system performance requirements onto component characteristics. For example, we want to determine how the size and the scheduling criteria of a buffer influence the latency of packets through the target system. In our previous work [5], we proposed and implemented a Greedy based technique in which we have used only direct dependencies between the system requirements & component attributes. For example, the number of gates (area) used by the target system is directly dependent on the buffer sizes and the scheduling algorithms of the constituents components. In this paper we propose two approaches for extracting indirect dependencies between system performance and components characteristics, dependencies that cannot be expressed as exact mathematical formulas. The first approach uses non-linear regression analysis and the second approach uses artificial intelligent techniques, specifically decision trees and conditional probabilities. With regression analysis we are aiming at obtaining approximations of how well a combination of components satisfies certain performance requirements. Decision trees classifiers and conditional probabilities will be used to order the components based on their probability of satisfying different system performance requirements. In the next section, we will illustrate our proposed approaches with an example.

III. EXPRIMENTAL RESULTS

We illustrate our component selection algorithm with a simplified example of selecting the components for a 4x4 mesh-based NOC architecture. An NOC is a multi-core architecture that provides a communication infrastructure for different resources, such as Application Specific Processors

(ASP), Application Specific Integrated Circuit (ASIC) blocks, General Purpose Processors (GPP), Field Programmable Gate Arrays (FPGA), memory or any other hardware blocks. Communication between resources is based on a packet switching protocol [6][7]. Figure 1 illustrates a 3x3 mesh-based NOC architecture. Each resource (denoted P/C as a producer/consumer of data packets) is connected to a network switch (NS) through two buffers, an input buffer and an output buffer. Every switch is also connected to four other switches (one for each direction north, south, east, and west) through four input buffers and four output buffers.

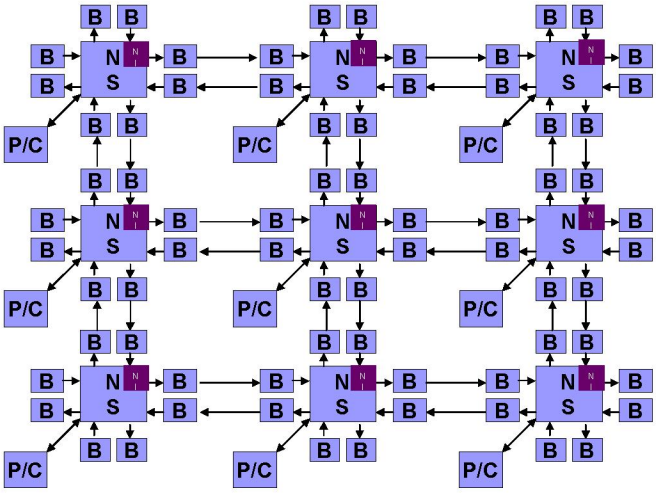

Figure 1. A 3x3 mesh-based NOC architecture

Thus, the main functional components of an NOC architecture are: resources, buffers, and switches. These components are characterized by several non-functional attributes. Resources, which are data producers, are characterized by the packet injection rate (that is, the rate at which they produce packets to be delivered through the NOC). We assume 10 possible values for the injection rate: from 0.1, 0.2,…, 0.9, 1.0. Buffers are characterize by size and scheduling criteria, and switches, are also characterized by scheduling criteria. We assume 6 possible values for the buffer size: 1, 2, 3, 4, 5 and 10, and 4 possible values for the scheduling criteria: PB (Priority Based), PBRR (Priority Based Round Robin), RR (Round Robin) and FCFS (First Come First Served). Assuming the buffers and switches use the same scheduling criteria results in 10x6x4 = 240 potential combinations of resources, buffers and switches, in an NOC architecture. Depending on the characteristics of the components integrated into the NOC architecture, the resulting system will exhibit different performances.

The performance requirements for the integrated system are expressed in terms of latency for high, medium and low priority packets and number of gates used (or area). Specifically, the system requirements for our example, are as follows: first, a resource, a buffer and a switch must be selected; second, the buffer size should be no less than 3; third, latency for priority packets flowing through the integrated system should be no greater than 30ns. In our component selection algorithm the first stage is reducing the search space, by eliminating those components that do not satisfy the system requirements. At this stage we are looking for functional requirements and implementation or interface requirements that are in the form of exact or range matches. Our first system requirement is actually a functional requirement in the form of an exact match. It allows us to eliminate from our search space all components from our library, except resources, buffers and switches. The second requirement is an implementation requirement in the form of a range match and it allows us to reduce our search space further, by eliminating the buffers whose sizes are less than 3. Our third requirement is a performance requirement in the form of range match, so it cannot be used in this stage to further reduce the search space.

In the second stage of our component selection algorithm, we perform a greedy search through our reduced search space and select at each step one more component until the resulting system is complete or until the system requirements are violated. If the second case occurs we backtrack. To minimize backtracking we have to choose at each step the component that best satisfies the requirements. For this, we need to map the performance requirements on the component attributes. Specifically, we determine the dependencies of the latency of high, medium and low priority packets on the injection ratio, buffer size, buffer scheduling criteria and switch scheduling criteria. For this purpose, we assume that using system simulation software (for example, MLDesigner) we can measure the latency and other performance characteristics of a limited number of combinations of components. Table 1 shows a subset of our measurements. Analyzing these measurements will help determining the system performance dependencies onto component attributes and the details of this analysis are presented in the next subsections.

Table 1. Latency for high, mid and low priority packets

INJECTION RATE (I)	BUFF. SIZE (B)	SCHED. CRITERIA (S)	LATENCY HIGH PR. (L1)	LATENCY MID PR. (L2)	LATENCY LOW PR. (L3)
0.1	2	PB	24	24	24
0.5	2	PB	25	25	25
1.0	2	PB	27	28.39	32.98
0.1	5	PB	21.88	22.1	63.73
0.5	5	PB	25	25	68.02
1.0	5	PB	26.7	28.73	76.16
0.1	10	PB	22.02	22.1	137.07
0.5	10	PB	25	25	139.69
1.0	10	PB	26.76	28.26	146.14

A. Regression Analysis

Regression is a mathematical measure of the average relationship between a response variable and one or more input variables. Data latency is our response variable or the dependent variable. However, we have three different types of latencies, for high priority, mid priority and low priority data packets. Thus, we have three different response variables

(L1, L2, L3). These response variables are dependent upon a number of input variables: injection rate (I), buffer size (B) and scheduling criteria (S). Injection rate is numeric and we assumed latency information is available for 3 levels of the injection rate: 0.1, 0.5 and 1.0. Buffer size is also numeric and we assume again that latency information is available for 3 buffer sizes: 2, 5, and 10. Scheduling criteria is a non-numeric input, so we have mapped each of it 4 possible values into a number: FCFS is mapped to 1, RR to 2, PB to 3, and PBRR to 4. Table 1 shows latency measurements for PB scheduling criteria. We have similar latency measurements for the other three scheduling criteria therefore, in total we have 9x3 = 27 measurements. These measurements represent our input into the regression analysis. Our goal was to identify and select the best regression model for each of our three response variables (L1, L2, L3), based on the least square method [8]. For each response variable, four different models are analyzed (quadratic model that includes main effects + interaction + square terms, model with main effects + square terms, model with only main effects, and model with main effects + interactions) and the one with the least error is selected. The quadratic model turns out to be the best for all three response variables and the resulting regression equations are as follows:

$$L1 = 49.64 - 40.96*S + 10.53*B + 8.30*I - 3.03*S*B + 0.06*B*I - 0.44*S*I + 9.28*S^2 - 0.03*B^2 - 1.63*I^2$$

$$L2 = 49.06 - 41.43*S + 10.58*B + 10.43*I - 2.97*S*B + 0.004*B*I - 1.24*S*I + 9.49*S^2 - 0.03*B^2 - 0.86*I^2$$

$$L3 = 68.82 - 54.23*S + 2.29*B - 1.77*I - 2.65*S*B + 0.11*B*I + 1.49*S*I + 9.63*S^2 + 0.04*B^2 + 5.66*I^2$$

Excel Data analysis Toolpak was used for our analysis. Using the above equations we will be able to assess how well a certain combination of components (resources, buffers and switches) satisfies a latency performance requirement. Further statistical analysis, based on Taguchi 2−step optimization technique, indicate that buffer size (B) and scheduling criteria (C) are the most important factors in the above equations, and injection ratio (I) has the least influence.

B. Decision Trees

Our second approach to determine system performance dependency on component attributes is based on decision trees. We divide our range of latency values into several classes as illustrated in Table 2. Then we use the same measurements as for the regression analysis approach, plus additional measurements for buffer sizes 1, 3 and 4. We have used XLMiner for our analysis and for building a classification tree for each of the three outputs, that is, latency for high, mid and low priority packets. The size of the training set was 58 and the size of the validation set was 14. The validation set was used for pruning the trees. The resulting minimum error trees are shown in Figures 2,3 and 4 for high, mid and low priority packets, respectively.

Table 2. Latency classes

LATENCY VALUES	CLASS
10..19	2
20..29	3
30..39	4
40..49	5
50..59	6
60..69	7
70..79	8

LATENCY VALUES	CLASS
80..89	9
90..99	10
100..109	11
110..119	12
120..129	13
130..139	14
140..149	15

Our observation from the regression analysis, that the buffer size and scheduling criteria have the most significant influence on the latency, is reinforced by our decision trees. The injection ratio is used by only two of the decision trees and it appears on the lowest levels, meaning that it has the least influence on latency.

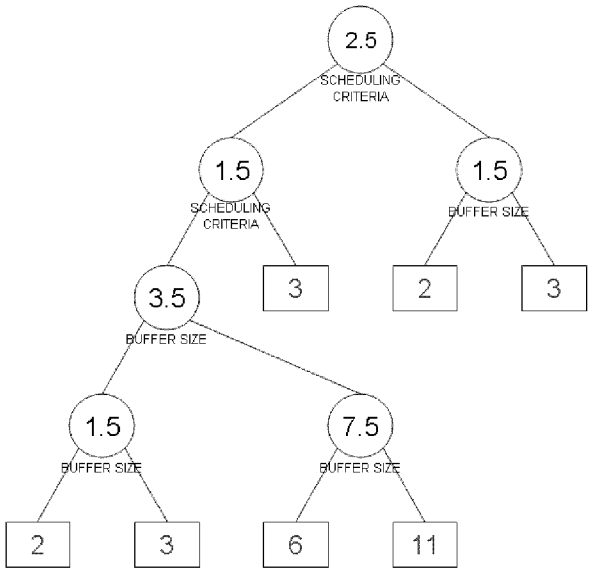

Figure 2. Classification tree for latency classes for high priority packets

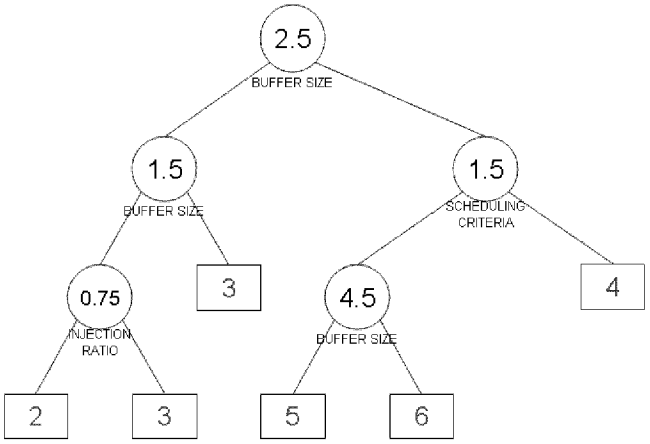

Figure 3. Classification tree for latency classes for mid priority packets

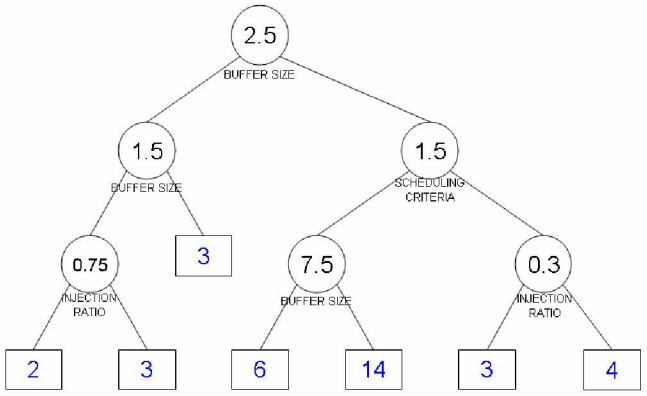

Figure 4. Classification tree for latency classes for low priority packets

Using the same tool, XLMiner, we have computed the conditional probabilities for each input variable and for each latency class. Table 3 shows a subset of these probabilities for latency classes 2 and 3, for high priority packets. For example, if one of our requirements is that the latency for high priority data packets must be less than 30ns (meaning it should fall into latency class 3 or lower), then there is a higher probability of achieving this if we select a buffer of size 1. If this is not an option, due to other system requirements, then the next best probability is obtained for a buffer of size 2, followed by a buffer of size 4, 5 or 10. Similarly, in terms of scheduling criteria, we have the best probability (0.35 + 0.28 = 0.63) of satisfying the target system requirements, if we select the scheduling criteria number 4, that is PBRR. The next best choice is scheduling criteria number 2, that is RR (with a probability of 0.25 + 0.28 = 0.53), followed by criteria number 3, that is PB (with a probability of 0.33+0.14 = 0.47).

Table 3. Conditional probabilities for latency classes 2 and 3 for high priority packets

Input Variables	Latency Classes			
	2		3	
	Value	Probability	Value	Probability
INJECTION RATE	0.1	0.5714286	0.1	0.2307692
	0.5	0.4285714	0.5	0.2820513
	1	0	1	0.4871795
BUFFER SIZE	1	1	1	0.1025641
	2	0	2	0.2051282
	3	0	3	0.1538462
	4	0	4	0.1794872
	5	0	5	0.1794872
	10	0	10	0.1794872
SCHEDULING CRITERIA	1	0.2857143	1	0.0512821
	2	0.2857143	2	0.2564103
	3	0.1428571	3	0.3333333
	4	0.2857143	4	0.3589744

Basically, using these conditional probabilities we can impose a certain ordering between components, which will be used when evaluating components during the Greedy search, in the second stage of our component selection algorithm.

For example, to satisfy our third system requirement, that latency for high priority packets should be less than 30ns, we will first select the components with the highest probability of satisfying this requirement. This results in selecting a resource with injection rate 0.1, a buffer with size 4 (because buffer sizes 1 and 2 have already been eliminated in the first stage) and scheduling criteria 4, that is PBRR. Based on these selections, we then evaluate latency for high priority packets using the first equation from our regression analysis. L1 evaluates to 28.1ns which is less than 30ns, therefore our third requirement is now satisfied and we have one potential solution to our component selection problem. If more than one solution is desired, we can simply force the algorithm to backtrack whenever a solution is found and thus, the algorithm will search for more solutions.

IV. CONCLUSION

In this paper, we have proposed a component selection technique based on system requirements dependencies on component attributes. We also proposed a requirements categorization scheme that allows us to faster reduce our search space, before we perform an intelligent search through it. The Greedy algorithm proposed for searching, evaluates components, starting with those that have a higher probability of satisfying the target system requirements. These probabilities were computed in advance and used to order components. Regression analysis was used to approximate how well a certain component or combination of components can satisfy a certain system requirement.

In our future work we propose to investigate scenarios in which a subset of components satisfying the system requirements cannot be found within the existing components and new components need to be developed. The challenge here is defining the characteristics of the components to be developed, given the unsatisfied system requirements.

REFERENCES

[1] 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.

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

[3] N. Haghpana, S. Moaven, J. Habibi, M. Kargar, S.H. Yeganeh, "Approximation Algorithms for Software Component Selection Problem", Asia-*Pacific Software Engineering Conf*, 2007, pp. 159-166.

[4] P.Potena, "Composition and Tradeoff of Non-Functional Attributes in Software Systems: Research Directions", *the 6th Joint Meeting on European software engineering conf. & the ACM SIGSOFT symp. on the foundations of software engineering*, 2007, pp. 583-586.

[5] A. Agarwal, G. Hamza-Lup, R. Shankar, J. Ansley, "An Integrated Methodology for QoS Driven Reusable Component Design and Component Selection", *1st Annual IEEE Systems Conf.*, 2007, pp.1-7.

[6] L. Benini, D. Bertozzi, "Network on chip architecture and design methods", *IEEE Computation Digital Technology*, vol. 152, no. 2, 2005, pp. 261-271.

[7] L. Benini, G. De Micheli, "Networks on chip: a new SOC paradigm", *IEEE Computer*, vol. 35, no. 1, 2002, pp. 70-78.

[8] R. Christensen, "Analysis of variance, design and regression applied statistical methods", 1st Edition, Chapman and Hall Publication, 1996.