Performance Analysis of Embedded Systems in the Virtual Component Co-Design Environment

P. Garg, A. Gupta, and J.W. Rozenblit
Department of Electrical and Computer Engineering
The University of Arizona
Tucson, AZ 85721-0104
aseemg@ece.arizona.edu

Abstract

Due to the increasing complexity of embedded systems in terms of functionality and architectural resources available to meet performance and cost criteria, there is an added responsibility on the designer to make the right choices. These choices can differ in terms of different hardware/software partitions, different types of architectural components, different communication architectures etc. and each choice meets certain performance metrics up to certain level. In this paper, we are exploring the design space to analyze different choices of design implementations by quantitative estimation of performance during simulation. Multi-Criteria Decision Making (MCDM) methods are used to rank our choices. To demonstrate the validity of the above exploration technique, a codesign tool from Cadence – Virtual Component Codesign (VCC) is used. It gives us the flexibility to create the experimental frame setup and probes to measure the performance metrics during simulations. The trade-offs between performance metrics are performed by MCDM. A safety critical example is chosen to demonstrate our approach.

1. Introduction

Board-level systems (e.g. automotive engine controllers for emissions) had used microprocessors for at least a decade before hardware/software codesign emerged as a discipline in the early 1990s. Moving the locus of CPU based design from boards to chips gave embedded microprocessors added cachet as an intellectual problem. The cost of design mistakes is also much higher in chips than on boards. Hardware/Software codesign [1] tries to increase the predictability of embedded system design by providing analysis methods that tell designers, if a system meets its performance, power and size goals, and synthesis methods that let researchers rapidly evaluate many potential design methodologies. Codesign aims at meeting system-level objectives by exploiting the synergism of hardware and software through their concurrent design [2]. The goal of codesign is to find an optimal HW/SW architecture that implements the system specification and meets the constraints with regard to real-time behavior, speed, area, memory, power consumption, flexibility, etc. This is also referred to as partitioning which is a classical combinatorial optimization problem of assigning functions to either hardware or software. In codesign, the implementation decisions for hardware, software and communication interfaces are closely related; changes in any one will immediately affect the other two.

Other codesign issues include [21]: 1) high-level architectural design-space exploration, 2) analysis of the trade-offs of implementing designs in hardware and software, 3) high-level design planning and estimation, 4) hardware/software partitioning at all design levels, and 5) analysis, verification and test issues. Among these issues, our research revolves around the second issue, i.e., the analysis of the trade-offs in implementing designs in hardware and software and partly delving into high-level architectural design-space exploration.

The variant of codesign developed at The University of Arizona has been called *Model-based Codesign* [11,12] where developers model a system specification independently of implementation and use simulation-based design to assess virtual prototypes, before the system is built. This design process uses stepwise refinement of simulatable models and offers the opportunity to abstract system components at multiple levels of representation. In this methodology, a set of requirements and constraints is obtained for the system to be modeled into an abstract model that is a combination of its structural and associated behavioral specifications. In Model-based codesign, Figure 1, we use computer simulation to increase the level of confidence that our model closely mirrors the system functionality. A simulation test setup, called an

Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04)
0-7695-2125-8/04 $ 20.00 © 2004 IEEE

IEEE COMPUTER SOCIETY

experimental frame [4,14], is associated with the system's model during simulation and specifies conditions under which the model of the system is observed. At the end of the simulation process, a virtual system prototype is obtained with the design partitioned into hardware, software and corresponding interfaces specified using a process that we call *model mapping*.

Given a particular system behavior, it is important to find the best system architecture including the right partition between hardware and software components, the right hardware components and communication protocols. Starting from the same system specification, several architectures may be produced. The exploration of all these architectures requires the ability to rapidly determine the performance resulting from a particular partitioning. We cannot afford to synthesize and simulate, at the cycle level, every single architecture to measure its performance as architecture; synthesis and low-level co-simulation may take a very long time. This explains the need for a performance estimation approach that can accomplish the complex task of architecture exploration within a reasonable amount of time. Since none of the architecture choices available is better than all others in every respect, we need to perform trade-offs.

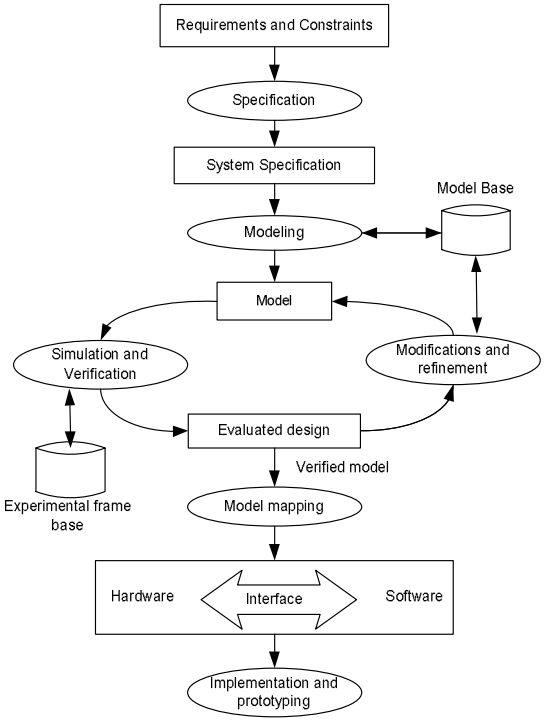

Figure 1. Model based co-design flow

We use Cadence's Cierto Virtual Component Co-design (VCC) system-level development environment to support our work. VCC [6] provides a close match to the model based codesign methodology and enables designers to rapidly mix and match software and hardware virtual components (VCs) into architectural prototypes, explore complex HW and SW trade-offs, analyze product performance, and evaluate product architectures early in the development cycle. It is found to be particularly useful for the development of embedded systems. In this paper, we begin with discussion on relevant subjects like performance evaluation, experimental frame, MCDM methods. Then, we move on to performance estimation and analysis using VCC followed by an example demonstrating our results.

2. Performance evaluation by system level estimation

The primary aspects of system evaluation are the functional features of the system such as the mode of operation, types of peripheral devices supported by the system, the size of directly addressable memory, languages supported by the system, data base management facilities, and so forth. Performance is only one point of view from which a system may be evaluated. System evaluation then involves evaluation of various trade-offs such as features vs. cost, or performance vs. cost, or performance vs. ease of use. This study concentrates on actual performance evaluation and the trade-offs involved between measures of performance. The process of evaluation starts with selecting a proper set of parameters, called performance metrics, upon which the evaluation will be based. They can be specified only with respect to the type and the purpose of the evaluated system, its workload, and the purpose of evaluation.

In our approach, metrics are the attributes of a partition that determine the partition's "goodness". We are faced with two options for computing metrics. First, by actually creating an implementation, providing us accurate metric values in large amount of time, and second, by creating a rough implementation quickly. A rough implementation contains the major real-time components of a design, but does not include many details, such as precise routing or optimized logic, that require much design time. The fidelity of estimation, [10] is defined as the percentage of correctly predicted comparisons between design implementations. The more accurate the model is, the higher the fidelity of the estimation, and the more

IEEE COMPUTER SOCIETY

likely that correct design decisions will be made based on comparing the estimates of two implementations. For quick estimation, we chose the higher fidelity option over accuracy of measurement.

3. Experimental frame

Zeigler [4] proposed the concept of an experimental frame that characterizes the circumstances under which a model or its real system counterpart is subjected to experimentation. The principle of separating the model from the experimental frame is defined in part as "any data gathering (statistics, performance measurement, etc) or behavioral control (initialization, termination, etc.) that is conceptually not carried out in the real system and should not be placed in its model but rather formulated as part of the experimental frame", [4].

A test bench (generator) is being used to generate the admissible input segments, which are applied to the model under study and observing the latter's output over the same interval. The second part of the frame is the data collection and reporting function, using the concept of probes [5], which calculate measures based on events occurring in model objects and record and/or process the values of those measures. Different types of probes are defined in a library of probe classes that predefine specific measures. We can either use one of those probes or modify the classes to create custom probes.

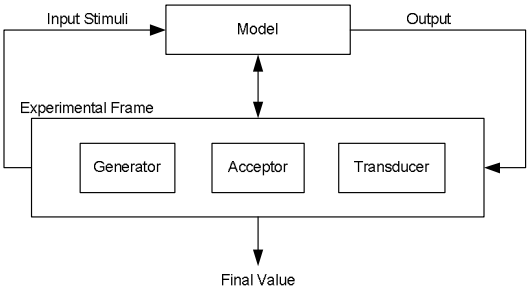

Figure 2. Experimental frame block diagram

4. Multi-Criteria Decision Making (MCDM)

The methods in MCDM focus on problems with discrete decision spaces, i.e., with countable few decision alternatives and basically use approaches from discrete mathematics. These methods do not try to compute an optimal solution; they determine through various ranking procedures either a ranking of the relevant actions (decision alternatives) that is "optimal" with respect to several criteria, or they try to find the "optimal" actions amongst the existing solutions (decision alternatives). This is achieved on the basis of the impact of the alternatives on the overall utility of the decision maker(s). The three steps [15] in utilizing any decision making technique involving numerical analysis of alternatives are: 1) determining the relevant criteria and alternatives, 2) attaching numerical measures to the relative importance of the criteria and to the impacts of the alternatives on these criteria, and 3) processing the numerical values to determine a ranking of each alternative.

Given a set of m alternatives denoted as $A_1, A_2, \ldots, A_m$ and a set of n decision criteria denoted as $C_1, C_2, \ldots, C_n$, it is assumed that the decision maker has determined (the absolute or relative) performance value a_{ij} (for i = 1,2,…,m and j = 1, 2, …, n) of each alternative in terms of each criterion. That is, we have determined the matrix A with a_{ij} values along with the criteria weights w_j and we have to rank the alternatives when all the decision criteria are considered simultaneously. The criterion represents profit and need to be maximized. MCDM methods used in this work are described briefly.

4.1. The WSM method

The Weighted Sum Model method is the most commonly used approach. If there are m alternatives and n criteria then, the best alternative is the one that satisfies (in *maximization* case) the following expression, where $A^*_{WSM\text{-}score}$ is the WSM score of the best alternative:

$$A^*_{WSM\text{-}score} = \max_i \sum_{j=1}^{n} a_{ij} w_j, \text{ for } i = 1, 2, \ldots, m. \qquad (1)$$

4.2. The WPM method

Weighted Product Model method is very similar to WSM. Here each alternative is compared with the others by multiplying a number of ratios, one for each criterion. Each ratio is raised to the power equivalent to the relative weight of the corresponding criterion. To compare two alternatives A_K and A_L, the product $R(A_K/A_L)$ has to be calculated using (2). If the product $R(A_K/A_L) \geq 1$, it implies that A_K is better than A_L (in the maximization case). The best alternative is the one

IEEE COMPUTER SOCIETY

that is better than or at least equal to all other alternatives.

$$R(A_K/A_L) = \prod_{j=1}^{n} (a_{Kj}/a_{Lj})^{wj} \quad (2)$$

4.3. The AHP Method

The analytic hierarchy process (AHP) [16,17] decomposes a complex MCDM problem into a system of hierarchies. The final step in the AHP deals with the structure of an *m* x *n* matrix constructed by using the relative importance of the alternatives in terms of each criterion. The vector ($a_{i1}, a_{i2}, a_{i3}, \ldots, a_{in}$) for each *i* is the principal eigenvector of an *n* x *n* reciprocal matrix which is determined by pair wise comparisons of the impact of the *m* alternatives on the *i-th* criterion. The entry a_{ij} in the *m* x *n* matrix, represents the relative value of alternative A_i when it is considered in terms of criterion Cj.

$$\Sigma a_{ij} = 1 \quad (3)$$

According to the AHP the best alternative (in the maximization case) is:

$$A^*_{AHP\text{-}score} = \max_i \sum_{j=1}^{n} a_{ij} w_j, \text{ for } i = 1, 2, 3, \ldots, m \quad (4)$$

4.4. The Revised AHP method

Belton and Gear [18] proposed that a ranking inconsistency can occur when the AHP is used due to the fact that the relative values for each criterion sum up to one. Instead of having the relative values of alternatives $A_1, A_2, A_3, \ldots, A_m$ sum up to one, they proposed to divide each alternative value by the maximum value of the relative values.

5. Performance estimation methodology

A behavioral model that contains the functional description of an embedded system's design can have different architectures representing it, as shown in Figure 3. The arrowhead M_i shows the mapping of the behavior to Architecture Model i and can have clearly measurable performance or cost metrics and their relative importance in the form of weights. We now need to select a suitable mapping using a solution strategy shown in Figure 4. From the input stimuli generated by test bench, we can measure values of *n* metrics using *n* probes, for a model M_i. This data forms the decision matrix for all the mappings. Element A_{ij} represents value of *j*th metric for *i*th mapping. MCDM methods described in the last section are then used to rank the different mappings.

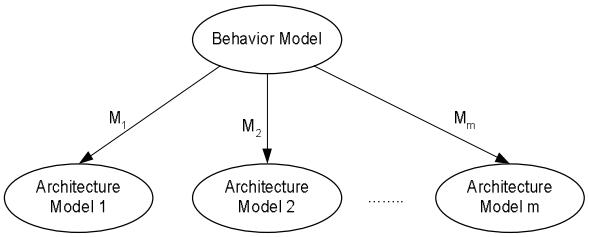

Figure 3. Behavioral model

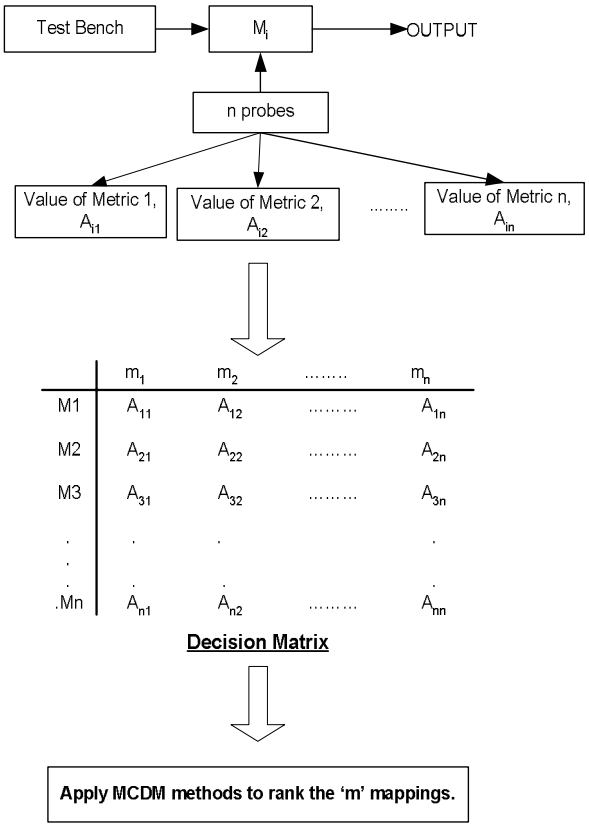

Figure 4. Design Flow

6. Performance analysis in VCC

Using the VCC environment, we can explore independent dimensions of behavior and architecture to reach optimal design performance within the given constraints. The VCC design flow is shown in Figure 5.

IEEE COMPUTER SOCIETY

To illustrate the application of MCDM trade-offs in VCC, we now present a small illustrative example. Safety Injection System (SIS) [5] of a pressurized water reactor (PWR) in a Nuclear Power Plant, which mitigates damage to the core and coolant system on the occurrence of a fault such as loss of coolant, is chosen. The block diagram is shown in Figure 6 and the details are presented in next section.

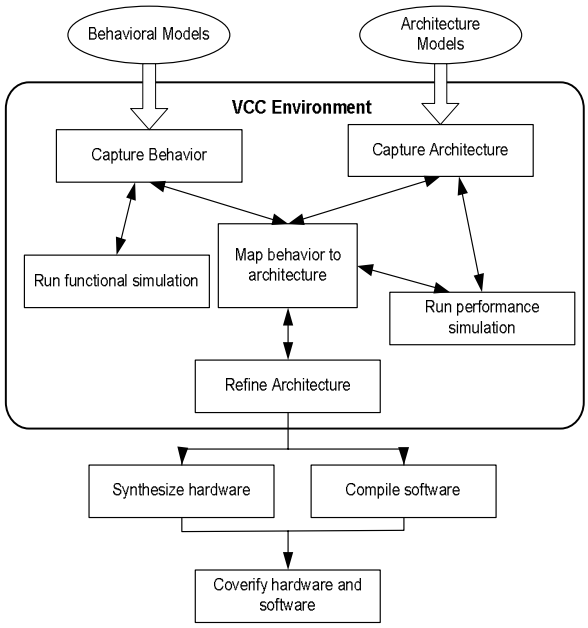

Figure 5. VCC design flow.

7. Example

The state variables include pressure in the pressurizer *Pressure*, integer type *WaterPress* with values of 0 (LOW) or 1 (PERMITTED). The values of LOW and PERMITTED are set based on whether *Pressure* is above or below the predefined threshold LOW. Other variables are *Block* and *Reset* that record the state of pushbuttons BLOCK and RESET respectively, time reference *TRef,* when the system is blocked and *TrefCnt,* an integer count of the number of events that occur on the input *TRe*f. The control variables are *SafetyInjection* and *Overridden*. *SafetyInjection* is enumerated with values of 'ON' and 'OFF' and adds water to the cooling system, when set to 'ON', which increases *Pressure* and this in turn updates *WaterPress*. *Overridden* is a boolean variable that is set when the operator asserts *Block* and reset when the operator asserts *Rese*t. *Overridden* will disable *SafetyInjection* even if the *WaterPress* indicates that *SafetyInjection* should be set to 'ON'.

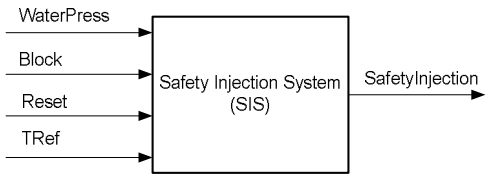

Figure 6. Block Diagram of SIS

Requirements for SIS are given in Table 1. The block diagram of the behavioral model as represented in VCC is shown in Figure 7.

Table 1. Text based requirements for SIS from [4]

[R1]	The system shall assert *SafteyInjection* when *WaterPress* falls below LOW.
[R2]	The system shall be considered blocked in response to *Block* being asserted while *Rest* is not asserted and *WaterPress* is below LOW, and shall remain blocked until either *Rest* is asserted or *WaterPress* crossed LOW from a larger to a smaller value.
[R3]	Once *SafteyInjection* is asserted, it shall remain asserted until the system becomes blocked or *WaterPress* becomes greater than or equal to LOW.
[R4]	When the system is blocked and *WaterPress* is below LOW, the system shall automatically unblock itself after the third timing reference event is sensed on input *TRef.*

Safety Injection Controller is the heart of functional description and all the requirements of Table 1, are being fulfilled by the Codesign FSM defined inside having four states (OFF, TOO LOW, BLOCK and ON) as shown in Figure 8. **Control Signal Driver** generates all the control and monitoring signals for the controller like 'overridden' signal based on the status of the Block and Reset buttons and outputs the integer count from the counter for the controller to monitor and take decisions. It also takes input from the controller about when to trigger or reset the counter. **Testbench** generates artificial inputs for the SIS model consisting of Control Signal Driver and Safety Injection Controller blocks. **Uniform Pulses** provides triggers at uniform intervals to facilitate the transitions of the CFSM. **SafetyInjection sink** and

IEEE COMPUTER SOCIETY

Integer sink complete the model specification by terminating outputs in sinks instead of keeping them open.

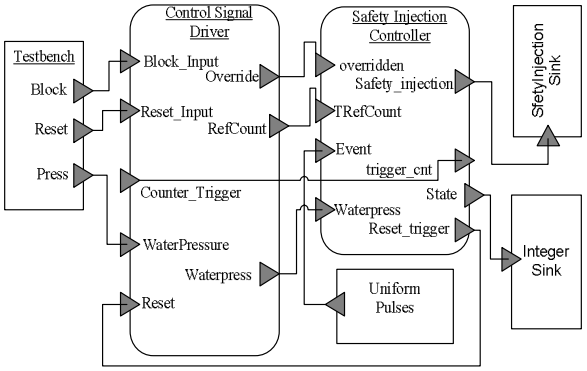

Figure 7. Behavioral Model in VCC

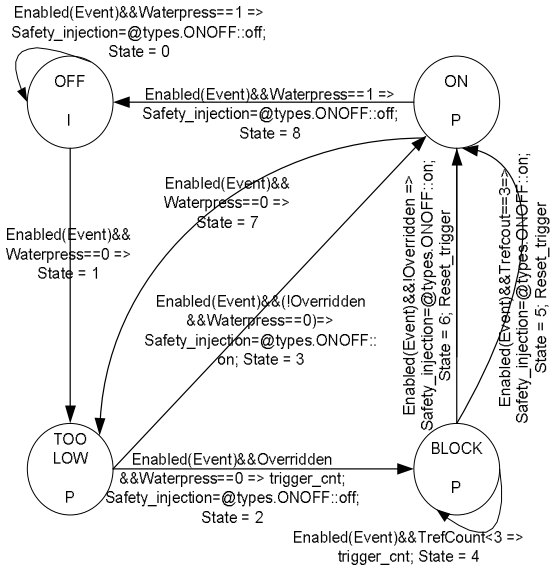

Figure 8. Safety Injection Controller FSM

Although, there can be 2^4 combinations (starting from all hardware to all software implementation) to partition the system, we included 7 different mappings in this study, shown in Table 2. The performance metrics are identified and probes built to capture the corresponding data from the system model, which in our case is the mapping diagram. The four metrics considered are: execution time, utilization, latency and response time represented by variables m1 to m4 respectively. Execution time of a behavior in the design is the average time required by the behavior from start to finish. Utilization is the fraction of time that the CPU/Bus resource is busy. Latency is the mean delay to send tokens from the source to the destination over the bus. On an occurrence of a fault or emergency, the time taken by system to change from ON state to OFF state or vice versa is called the response time.

Table 2. Different mappings

Map-ping	Safety Injection Controller	Uniform Pulses	Overridd-en signal Driver	Counter
M1	ASIC1	ASIC1	ASIC2	ASIC2
M2	ASIC1	Software	ASIC2	ASIC2
M3	Software	Software	Software	Software
M4	Software	ASIC1	Software	Software
M5	ASIC2	ASIC2	ASIC1	Software
M6	Software	ASIC1	ASIC2	Software
M7	Software	ASIC1	ASIC1	ASIC1

The performance simulator in VCC evaluates the performance and will identify missed events, estimate system-level performance and provide data on the usage of processors, buses, and other shared devices. We can remap behaviors to different architecture models, then rerun the simulation to analyze the effects on performance and compare different mappings to find out the best partition.

8. Trade-off analysis using MCDM

Based on the application of the SIS, we found that the following order of importance exists among the criteria defined above:

Response Time>Execution Time>Latency> Utilization

So, in accordance to above, arbitrary weights were assigned to the criteria such that for n metrics with weight w_i of the i-th metric.

$$\sum_{i=1}^{n} w_i = 1 \qquad (5)$$

IEEE COMPUTER SOCIETY

Also, all metrics except Utilization (m2) need minimization, so we replace m2 by inverse of utilization to maintain uniformity while calculating the aggregate of all criteria. Thus, the alternative with the smallest aggregate value will be ranked the highest. To avoid influence by order of metric, we remove powers of 10 from our calculations. Table 3 shows the decision matrix formed after data is collected and above changes are made.

Table 3. Decision matrix

	Execution Time	Utilization	1/Utilization	Latency	Response Time
w_i	m1 (e-4) 0.27	m2	1/m2 0.16	m3 (e-7) 0.22	m4 (e-5) 0.35
M1	1.40	0.123	8.092	4.481	1.022
M2	1.46	0.083	12.042	3.829	1.200
M3	1.46	0.082	12.158	5.214	1.466
M4	1.45	0.123	8.092	4.734	1.150
M5	1.40	0.123	8.092	5.814	1.045
M6	1.44	0.123	8.092	6.008	0.997
M7	1.42	0.123	8.092	5.814	1.047

Table 4. shows the scores for WPM method, where all possible combinations to compare the seven mappings are considered. The methods are then ranked on the basis of their scores. Using each of the MCDM methods and above data, the seven mappings are ranked as shown in Table 5.

Table 4. WPM results

	WPM score		WPM score
R(M1/M2)	0.90820	R(M3/M4)	1.18919
R(M1/M3)	0.79000	R(M3/M5)	1.18625
R(M1/M4)	0.93947	R(M3/M6)	1.18799
R(M1/M5)	0.93714	R(M3/M7)	1.18074
R(M1/M6)	0.93851	R(M4/M5)	0.99753
R(M1/M7)	0.93279	R(M4/M6)	0.99898
R(M2/M3)	0.86986	R(M4/M7)	0.99289
R(M2/M4)	1.03443	R(M5/M6)	1.00146
R(M2/M5)	1.03187	R(M5/M7)	0.99535
R(M2/M6)	1.03338	R(M6/M7)	0.99390
R(M2/M7)	1.02707		

Table 5. Summary of rankings

	WSM score & rank		AHP score & rank		Revised AHP score rank		WPM rank
M1	3.01	1	0.1303	1	1.046	1	1
M2	3.58	6	0.1455	6	1.160	6	6
M3	4.00	7	0.1660	7	1.335	7	7
M4	3.13	2	0.1388	2	1.114	2	2
M5	3.31	3	0.1394	3	1.130	3	4
M6	3.35	5	0.1396	4	1.132	4	3
M7	3.32	4	0.1401	5	1.135	5	5

9. Results

We observe the following findings from the rankings generated in the four MCDM methods:

(i) All the methods rank M1 at top and is the best implementation choice based on the metrics chosen, their relative importance and the given mappings.

(ii) M2, M3 and M4 also have the same standing in all the rankings.

(iii) Three methods show M5 as better than M6 and M7. Three other methods show M6 as better than M7. So, M5>M6>M7.

Thus, the final ranking is:

M1>M4>M5>M6>M7>M2>M3

10. Conclusion and future work

VCC is not exactly a partitioning tool, but it can perform an important task of partition evaluation in the process of partitioning. We demonstrated this by evaluating seven different mappings of a system. Partitioning inherently involves trade-off analysis, since each possible partition is better than the others in some respects and worse in other respects. We were able to perform this critical design step using VCC as a method to provide us with the measure of different performance metrics for the partitions. Next, we use MCDM methods to perform the trade-off analysis. The results are compiled together for overall ranking of the partitions.

The accomplishments of this work are: first is to justify the importance of taking into account the effect of metrics defining performance, i.e., than doing the tradeoff between performance (based on only one of the performance metrics) and cost. Second is the use

IEEE COMPUTER SOCIETY

of an experimental frame setup with a transducer replaced by a probe. We proposed to develop a probe for every metric of interest to gather the corresponding performance index from simulations. Third, an increase in the number of metrics will complicate the process of analyzing the trade-offs, thus, emphasis was laid on a quantitative technique for tradeoff using MCDM rather than visual or guessed estimations.

The methodology described above is scalable to even more complex systems. If the complexity of the system increases, the algorithm must be improved to reduce the exploration space.

The future work involves the inclusion of cost and power metrics as well. We can also identify dynamic metrics or custom metrics of performance for specific applications. This will help in refining the exploration and better estimates of design quality can be made. Another major scope of work is to automate the partitioning process itself. Here the sample space was generated manually and we could find the best mapping. Generation of a complete sample space for a complex system can be done using different algorithms. We identify a strong rationale to imbibe self-learning capability in the system by use of genetic algorithms. This will accurately determine weights and make the methodology more effective.

11. References

[1] Wayne Wolf, "*A Decade of Hardware/Software Codesign*", IEEE Computer, 38-43, April 2003.

[2] G. De Micheli, R.K. Gupta, "*Hardware/Software Co-Design"*, Proceedings of the IEEE, vol 85, no. 3, March 1997.

[3] Luna, Joel, "*Application of Hierarchical Modeling Concepts to a Multi-Analysis Environment"*, Proc. Of the 1991 Winter Simulation Conference.

[4] Zeigler, B.P., "*MultiFacetted Modelling and Discrete Event Simulation",* Academic Press, 1984.

[5] Courtios, P.J., Parnas, D.L., "*Documentation for Safety Critical Software*", Proceedings of the 15th International Conference on Software Engineering (ICSE'93), pp. 315-323, Baltimore, MD, 1993.

[6] Cunning, S. J. "*Automating Test Generation For Discrete Event Oriented Real-Time Embedded Systems.*" PhD Thesis for the Department of Electrical & Computer Engineering, University of Arizona, 2000.

[7] Schirrmeister, F., Krolikoski S., *"The System-level HW/SW Co-design Challenge",* http://www.cadence.com/whitepapers/vcc.html.

[8] Cadence, VCC Architecture Evaluation Guide, VCC version 2.1, March 2001.

[9] Peter M. Chen, David A. Patterson, *"Storage Performance--Metrics and Benchmarks"*, *Proceedings of the IEEE*, vol 81, no. 8, August 1993.

[10] Gajski, Daniel D., Vahid Frank, Sanjiv Narayan, Jie Gong, (1994), "*Specification and Design of Embedded Systems"*, P T R Pretence Hall, New Jersey, USA.

[11] Cunning, S.J., Ewing, T.C., Olson, J.T., Rozenblit, J.W., Schulz, S., *"Towards an Integrated, Model-Based Codesign Environment*", Proceedings of the 1999 IEEE Conference and Workshop on Engineering of Computer Based Systems (ECBS'99), pp. 136-143, Nashville, TN, March 1999.

[12] Schulz, S., Rozenblit, J.W., Mrva, M. and Buchenrieder, K., *"Model-Based Codesign*", IEEE Computer, 31(8), 60-67, August 1998.

[13] Rozenblit, J.W. and Buchenrieder, K. (Eds.), *Codesign: Computer-Aided Software/Hardware Engineering*, IEEE Press, 1994.

[14] Rozenblit, J.W., "*Experimental Frame Specification Methodology for Hierarchical Simulation Modeling,"* International Journal of General Systems, 19(3), pp. 317-336, 1991.

[15] Triantaphyllou, E., (2000), *Multi-Criteria Decision Making Methods: A Comparative Study*, Kluwer Academic Publishers.

[16] Winkels, H.-M., G. Wäscher (1981), "*Outranking approaches – an integrated survey and a bibliography",* Working Parpers on Economathematics 8107, Ruhr-Universität Bochum.

[17] Zeleny, M. (1982), *Multiple criteria decision making,*McGraw-Hill, New York.

[18] Zeleny, M. (1973), *Compromise Programming. Cochrane, J.L., M. Zeleny (Eds.): Multiple criteria decision making* ,University of South Carolina Press, Columbia. pp. 262-301.

[19] Saaty, T.L., (1980), *The Analytic Hierarchy Process,* McGraw-Hill, New York, NY, USA.

[20] Saaty, T. L. (1977), "*A Scaling Method for priorities in hierarchical structures",* Journal of Mathematical Psychology 15, pp. 234-281.

[21] Givargis, Tony D., "*System-level exploration for Pareto-optimal configurations in parameterized system-on-a-chip architectures",* PHD Dissertation, University Of California, Riverside, 2001.

IEEE COMPUTER SOCIETY