Processor Evaluation Cube: A classification and survey of processor evaluation techniques

Rickard Holsmark and Shashi Kumar

ISSN 1404 – 0018
Research Report 04:3

SCHOOL OF ENGINEERING
JÖNKÖPING UNIVERSITY

Processor Evaluation Cube: A classification and survey of processor evaluation techniques

Rickard Holsmark and Shashi Kumar

Embedded Systems Group
Department of Electronics and Computer Engineering
School of Engineering, Jönköping University
Jönköping, SWEDEN

ISSN 1404 – 0018
Research Report 04:3

Abstract

Selecting appropriate hardware resources corresponding to the application is an important task for design of an embedded system or a SoC. A large number of techniques have been proposed in literature to select a processor matching with the application requirements. In this report, we propose a framework called Processor Evaluation Cube (PEC) which helps in systematic classification and comparison of various processor evaluation techniques. The three axes of PEC are: Analysis, Architecture and Abstraction. The Analysis axis distinguishes methods employing static analysis or simulation; Architecture axis distinguishes methods evaluating single processor or multiprocessor computing platforms; Abstraction axis distinguishes methods employing clock true evaluation or higher level execution time estimation techniques. Our survey not only puts the existing techniques in proper perspective but also points to the weaknesses in the existing techniques which need to be removed if these techniques have to be used for design of multi-processor SoC (System on Chip) and Network on Chip (NoC) systems. We observe during our survey that the techniques for single processor evaluation are getting adapted for evaluation of multi-processor platforms. We also note that there are no techniques developed or proposed for low level static analysis of multi-processor platforms due to complexity of such an evaluation. Although many different evaluation approaches may fall in the same category in PEC classification they have significant differences in intermediate descriptions of application, architecture and performance parameters.

Keywords

Processor architectures, Performance, Evaluation, Estimation, Simulation, Systems on Chip, Multi-processor

Table of Contents

1 Introduction

Reuse and design at higher level of abstraction at various levels are the keys to design complex systems in short time. Various architectural platforms, like Networks on Chip (NoC) [1], are being proposed to simplify the design of Systems on Chip (SoC). In these platforms the interconnection infrastructure can be reused from one product to another. A major task in system design using NoC paradigm is to select appropriate set of cores for the required application or application area. These cores could be general purpose or special purpose processors, memory cores or application specific cores.

It seems that the trend towards more software implementations of functions in electronic devices will not only continue but also accelerate. Implementing functionality in software has many advantages compared to hardware implementation. Flexibility, cost and design time are all issues that favors a programmable architecture. It is only in the case when the available processors can not provide the required performance, that one by necessity design a hardware solution. The demand for availability of advanced multi-media applications anywhere, puts requirements of higher performance and smaller size and power consumption on the underlying computing platform.

When designing an embedded system a key issue is to get a good match between application demands and hardware resources. Excessive resources have a cost but add no value, and in the highly competitive embedded systems market, cost always has to be considered in order to survive. Evaluation of a large number of processors or processor cores is a necessary but time consuming task for selecting appropriate components. This will become even harder, since both the number of off-the-shelf processors and customizable processor cores are steadily increasing. Performance estimation techniques are important when exploring the architectural design space during the hardware/software co-design flow of embedded systems. Design Space Exploration (DSE) may involve selecting not only among processors, but also communication architectures, dedicated hardware or even programmable hardware. Fast performance evaluation tools are useful in the early stages of DSE to reduce the design space of processor/processors cores that can be used for the software implementation part of the design. But also at more concrete descriptions, there is use for efficient performance estimation techniques. Because of the amount and complexity of the possible architectures it is becoming impractical to obtain accurate performance numbers. It has become necessary to find a reasonably low number of candidate architectures that can lead to feasible design solutions.

Wolf [2] considers system level performance analysis and DSE as important research problems for SoC design. Unfortunately, there has been decrease in research and publications in the area of multi-processor performance evaluation relative to the growth of multi-processor SoCs [3]. When discussing processor performance, one must keep in mind that it is not limited to the processor core only; peripherals and memory organization also has a large performance impact when designing a computer system. Another difficulty in performance evaluation is that not only hardware determines the performance. Due to the efficiency of different compilers, one application can have different execution speed on the same processor. Many other aspects besides the measurable cost/performance have an influence on the selection of a processor. One important aspect which greatly affects selection is the existing experience of processors and its development tools in a company. There is a hard to measure cost associated with learning new tools for development and new processor architectures. For small companies, cost of the tools may be the most important factor. A new faster processor may look better on paper but there can be problems with its reliability. Design bugs will perhaps only be detected after years of usage and this may be the crucial design choice, especially important for life critical devices.

Though the focus in this paper is selection of pre-designed processor architectures, there is another area facing similar challenges. It is usually referred to as design of Application Specific Instruction set Processors (ASIPs) or Application Specific Programmable Processors (ASPP) [4], [5]. Here a processor is to be designed with resources that match a certain application or application areas demands. Typical parameters for processor specialization in this are number of registers, special instructions and various types of functional units. There is also an interesting contradiction in system design, which lies in the relationship between mapping an application to an architecture, versus selecting a suitable architecture for an application. Beck and Siewiorek denotes this as task allocation versus hardware specification and recognizes them as mutually constrained [6]. If an application is to be mapped then the hardware has to be fixed during this process. On the other hand if an architecture is to be selected then the application behavior has to be partitioned a priori.

In the research community, literature presenting evaluation methodologies address a variety of issues. Some researchers propose techniques to estimate software performance on a processor. Other proposals also include methods for selecting the processor, or set of processors, that matches best with the design requirements. The objective of this paper is to survey the ideas and techniques to evaluate, with emphasis on performance, the suitability of a computing architecture for certain applications demands and put them in proper perspective. We achieve this by classifying different techniques through a framework called Processor Evaluation Cube (PEC). The three axes of PEC are: Analysis Method, Analysis Abstraction Level and Single/Multiple Processor Evaluation.

The rest of the report is organized as follows. In section 2 we describe the steps used in processor evaluation methods. Section 3 introduces the basic idea of PEC and further discusses in detail the role and purpose of the individual axis of PEC, when distinguishing different processor evaluation techniques. In section 4 we list other work related to this report. Section 5 presents different important processor evaluation techniques and puts them in PEC framework. In section 6 we discuss important aspects of the techniques and also highlight some features that should be considered when estimating performance in packet-switched NoC systems. In section 7, we conclude the report and present some thoughts regarding different classification possibilities.

2 Tasks in Architecture Evaluation and Selection

Figure 1 show steps commonly used in evaluation and selection of an architecture appropriate for an application. Interestingly, Application Specific Instruction Processor

(ASIP) synthesis has many common steps with architecture selection. Given an application written in some high level language, e.g. C, the important behavioral requirements have to be transformed to a model suitable for evaluation. The application model is then used in the performance evaluation stage together with the architecture model.

The architectural parameters are usually extracted, specified and constrained separately before used in the evaluation stage. The models of the different processors are generally stored in a library and used for the selection purpose. The outcome of the performance estimation is a numerical description on how well the application can be executed on the architecture. If several alternatives fulfill our performance requirements we should be able to get a feasible solution based on the purpose of our design. In the case of selecting among a number of pre-designed architectures, the processor or multiprocessor architecture that can best perform the application is selected using a cost/performance ratio decided by the cost function. Analogously, in the case of ASIP synthesis, a processor configuration proposal is selected and synthesized.

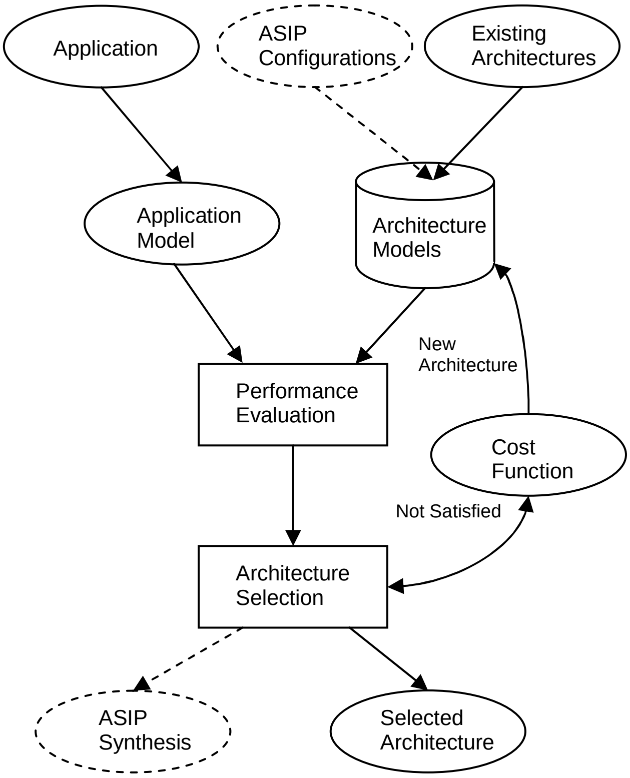


Figure 1: Processor Evaluation Flow

If the requirements are not satisfied, new architectures are chosen for evaluation in guidance of the cost function. There could also be the possibility that we have some other constraints on our design and we want to evaluate the performance that is possible to expect considering these. This survey is targeted to the evaluation of processors where we want to select the most suitable processor or processors among a number of candidates. Some of the proposals only consider techniques that stop after the performance evaluation stage in order to develop these methods. Other techniques focus on the selection procedure and use external or more simple performance estimations in the architecture selection stage.

3 Processor Evaluation Cube

Literature on processor evaluation techniques is continuously growing. Various techniques have been developed in different research and development contexts and it looks quite difficult to compare them. We have identified three important aspects by which we can distinguish various techniques. The three aspects are:

- Abstraction level of evaluation
- Analysis Axis : Static analysis vs. Simulation
- Architecture : Single processor vs. Multiprocessor

These three aspects can be graphically represented by three orthogonal axes constituting a cube as shown in Figure 2. As a result of this we call this evaluation framework as Processor Evaluation Cube (PEC). For the sake of simplicity, we restrict the number of points on each axis to only two. Each of the evaluation techniques will be a point in this discrete cube.

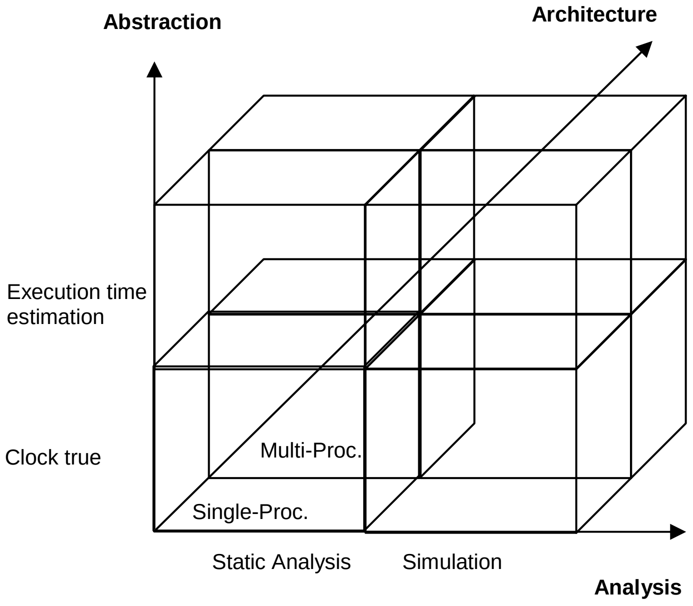


Figure 2: Processor Evaluation Cube

The vertical axis is related to the abstraction level at which the computing platform and application are modeled. The abstraction level affects the speed and accuracy of evaluation. The analysis axis distinguishes between techniques which are based on evaluation of the processor using off-line analysis of the application as compared to analysis using simulation. The architecture axis capture that some techniques are suitable for evaluation/selection of single processor computing platforms, as compared to other techniques which can evaluate/select multiprocessor structures. We believe that in the near future we will see development of techniques for multiprocessor computing platforms. In the next section, we discuss these three axes in more detail.

3.1 The Three Evaluation Axes in PEC

3.1.1 Abstraction Level

The evaluation results will be most accurate if the application is actually implemented and run on all possible architectures. However this is not easy because of a number of reasons. For example the cost of acquiring all necessary hardware and software development tools would be huge. The lowest level on which performance usually are estimated are number of clock cycles. This gives a true indication of the application performance since the latencies of the stages in the instruction cycle are accurately accounted for. A problem with this low level estimation is that it is relatively time consuming. The evaluation tools are usually proprietary and gathering these could be both time consuming and expensive. Retargetable compilers and simulators alleviate the designer from this but still the time for simulation will remain. In order to solve this, researchers have tried to raise the level of abstraction, removing details in order to get faster evaluations. Higher abstraction level may also be needed because the application description is only available in HLL code or an algorithm in a pseudo code. As a result of removing or ignoring details, the accuracy of the estimation will decrease.

3.1.2 Analysis: Static Analysis vs. Simulation

In order to evaluate the performance of a computing structure for execution of an application, one has to use models of both that can be integrated with each other. When looking at research in this field, two distinct approaches can be found; one using static analysis and the other employing simulation methods. In the static analysis based approach, requirements and characteristics are extracted from the application and statically matched with the parameters of the architecture. This analysis is carried out off-line. In the second approach the application is modeled as a simulatable description and is, dynamically, run on a model of the computing structure.

ASIP design community [5] refers to these approaches as simulation based vs. scheduler based, which is also a good description since most of the static analysis techniques, rely on applying scheduling techniques on the application to obtain characteristics. However, we use a more general name “static analysis“, since it can also cover some methods which evaluate without scheduling.

3.1.3 Architecture: Single vs. Multiprocessor Systems

An evaluation approach defines the architectural space of the computing structure. Approaches differ in whether they include effect of memory size and organization on the performance, or if they ignore its effect. Most of the approaches are restricted to evaluating computing structures with one processor. In this case the goal is to find the most suitable processor for a certain application. Some approaches attempt to support development of multiprocessor solutions. These try to find the best set of processors, meeting performance requirements and constraints. In case of multiprocessor architecture selection, inter-processor communication architecture also needs to be defined before the exploration stage. One can think about a homogenous architecture implying identical processing units or a heterogeneous architecture where the processors could be of different type. A heterogeneous multiprocessor system could also incorporate special hardware blocks and ASIPs as possible options.

4 Related Work

The PEC proposal of classifying processor evaluation methods was motivated by Flynn's famous classification of computer architectures [7] into four classes. We believe that such classifications are very useful and make the study of important topics systematic.

Other researchers have earlier distinguished techniques using one or more of these aspects. For example Hergenhan and Rosenstiel mention three methods for obtaining timing results namely, simulation, emulation and static timing analysis [8]. Giusto et al. [9] of Cadence presents an interesting and thorough description of the different approaches and techniques for performance estimation. They are however not giving any special attention to multiprocessor systems. Russel and Jacome [10] describes performance evaluation as either being static (analysis) or dynamic (simulation). Kin et al. [11] identifies a historical division of two different communities working on the architecture evaluation problem. One is the CAD community that is working from the perspective of area estimation and optimization algorithms. The other is the architecture community that uses extensive benchmarks in order to obtain performance estimates. It is observed that there is however a convergence between the two communities.

Baghdadi et al. [12] see two different classes of performance estimation proposals. First those dealing with mono-processor systems and secondly those dealing with multiprocessor systems. Zivkovic et al. [13] identify and discuss two different techniques for simulation called Trace Driven (TD) simulation and Control Data Flow Graph, (CDFG) approach. They also propose a combination of these that would solve their built-in shortcomings. In this paper they also give a thorough description of the y-chart [14] that describes the steps in performance analysis. The abstraction pyramid [14] that visualizes the relation between the reachable design space and the abstraction level of evaluation is also described.

5 Survey of Existing Methods

Figures 3 and 4 organize processor evaluation methods for single and multi-processor computing architectures respectively. To simplify readability PEC has been drawn as two planar rectangles rather than one three dimensional cube. The rectangle in figure 3 contains methods which are suitable for evaluating only single processor computing platforms. The rectangle in figure 4 covers techniques for multi-processor computing platforms. Each rectangle is further divided into two rows of two columns each. The rows show the abstraction level employed in the evaluation technique and columns tells whether the technique use static analysis or simulation. Note that all, except [15] of the papers below compare the obtained results with another, more accurate performance estimation tool, when reporting the accuracy. Though being more precise, these tools may incorporate some, not accounted for, in-accuracy.

5.1 Single Processor Evaluation/Selection

Single-Processor Architectures

Abstraction	Static Analysis	Simulation
Execution time estimation	Carro et al. [16] Gupta et al. [17] Russel and Jacome [10] Hergenhan and Rosenstiel [8] Giusto et al. [9]	Lajolo et al. [19]
Clock true	Chen et al. [18]	Desikan et al. [15]

Analysis

Figure 3: Single-Processor Evaluation Techniques

5.1.1 Static Analysis at High Abstraction Level

Methods in this category help in achieving fast estimation of performance. Here we list five important proposals in this category [16] [17] [10] [8] [9].

In Carro et al. [16], the behavior of an application is modeled using C++. The processor selection is done using a library of possible candidate processors. For evaluation an intermediate description of the code is used. The behavior is characterized into three different types, namely, Control dominated (FSM), Data intensive computation (Digital filters) and Memory intensive computation (List processing). Depending on the type of object a microcontroller, a RISC or a DSP is selected. The technique does not provide estimate of the execution time but gives relative suitability among a set of processors.

Gupta et al. [17] present a method to estimate processor performance based on the applications requirements and the architectural resources. The method extracts essential parameters of an application written in a high level language, like C, by using SUIF (Stanford University Intermediate Format) as an intermediate representation. Together with the architectural description these parameters are fed to an estimator that gives predictions on code size and execution time. The architecture representation contains features like number of registers, number of functional units and latency of operations. The application is profiled and passed to an architecture constrained scheduler, which is used to estimate the number of cycles the execution would take. Results are compared with lower level tools and accuracy where within 30 %.

Russel and Jacome [10] have proposed a method of static analysis based on a typical case rather than taking into consideration all possible paths in a program. The method is not completely automated since it relies on manual inputs from a designer. The errors in the estimates were within 25 %. More promising in this case was that relative performance accurately was captured among the evaluated architectures.

Hergenhan and Rosenstiel [8] work with static timing analysis that provides Worst Case Execution Time (WCET) estimation. In their work they have focused on the PowerPC family which have similar instructions but is built on different internal architectures. The estimation technique involves cache modeling, prediction techniques and multiple issue pipelining. In finding the worst case path they use implicit path enumeration. They use a tool called GROMIT for obtaining timing information out of an assembly program, a functional path relation description and a processor description. They observe that the prediction error is highly dependant if cache is modeled or not, but in most cases the error seems to be above 10%. The results show that prediction results of MPC750 are more sensitive to modeling cache behavior than multi-issue pipeline behavior.

Giusto et al. [9] propose a technique to estimate execution time by using virtual instructions. They do this by evaluating many applications related to the area in order to obtain correctness of statistical estimates. The cycle counts obtained for each instruction are then used to estimate the cycle count of each virtual instruction in the application. The prediction error range in the benchmarks was found to be at best -12% to 5%.

5.1.2 Static Analysis at Low Abstraction Level

In connection with the MESCAL project, Chen et al. [18] present a new method in retargetable static analysis in order to determine bounds on execution time on modern processors. They presume that the program is statically predictable and use the Cinderella tool to identify structural and functional constraints in the program. They model cache behavior, instruction pre-fetching and branch predication for increased accuracy. In order to demonstrate their technique they have run some experimental programs showing the accuracy of the method as compared to a method using compiler profiling and manual setting of input scenarios. The accuracy was extremely good and each program took less than a second to estimate.

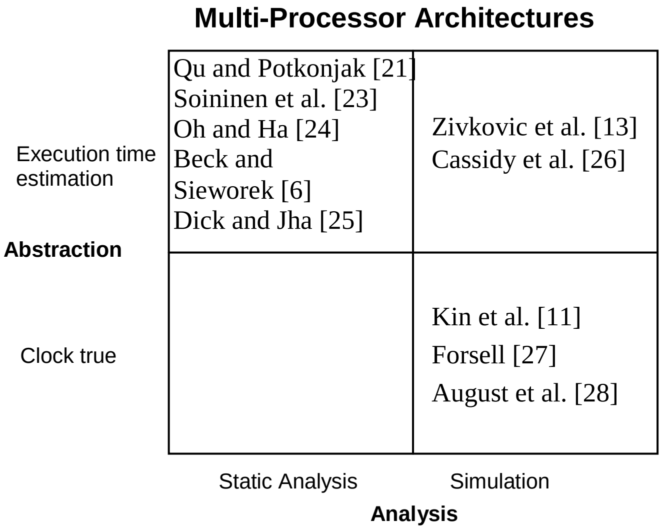
5.1.3 Simulation at High Abstraction Level

Lajolo et al. [19] propose a technique for better accuracy in co-simulation of HW/SW systems in the POLIS framework. In their proposal the application is described as a C-program and a retargetable compiler is used to optimize the code. It also generates instruction level timing analysis and a C simulation model in order to make hardware software co-simulation. They claim that the estimates are within 4% accuracy for certain case studies. It is not clear whether their technique considers performance features like pipelining etc., or can work for advanced processor architectures like VLIW processors.

5.1.4 Simulation at Low Abstraction Level

The SimpleScalar [20] is a tool-set consisting of several simulators working on different abstraction levels that provides trade-off possibilities between accuracy and speed. Included in the model are also configurable cache memory models, incorporating algorithms of some ordinary prediction techniques. Using the tool-set, Desikan et al. [15] developed a detailed micro-architecture level simulator of an Alpha 21264 processor. In an extensive hardware validation of the simulator accuracy in small benchmarks, prediction error was found to be on average 2%.

5.2 Multiprocessor Evaluation/Selection



Multi-Processor Architectures

Abstraction	Static Analysis	Simulation
Execution time estimation	Qu and Potkonjak [21] Soininen et al. [23] Oh and Ha [24] Beck and Sieworek [6] Dick and Jha [25]	Zivkovic et al. [13] Cassidy et al. [26]
Clock true		Kin et al. [11] Forsell [27] August et al. [28]

Figure 4: Multi-Processor Evaluation Techniques

5.2.1 Static Analysis at High Abstraction Level

In these multiprocessor techniques we find a mix of both evaluation and selection techniques [21] [23] [24] [6] [25].

Performance ratings, like MIPS, provided by the manufacturers have been used by Qu and Potkonjak [21] for estimation purposes. Using these numbers the applications execution time is estimated. Architecture modeling effort can in this case be considered minimal, but it is difficult to get good estimates using this technique [22].

An interesting method to match a processor core to an algorithm using a mappability estimation metric is presented by Soininen et al. in [23]. The method can be used both to find the best candidate core and to find out how the optimal core would be designed having a configurable core. Memory, cache and communication organization and requirements are not considered in this method due to complexity reasons. It is also not handling the absolute performance of the processor and thus can not tell if it is able to manage the processing task in time. A tool extracts the essential behavior characteristics of the algorithm regarding control flow and data dependencies. This is matched with the characteristics of the core such as instruction set and execution architecture. The algorithm is given as a C-file which is transformed using SUIF into an intermediate format. The tool then calculates mappability value as well as optimal values of core architectural parameters. The method has been used in order to identify suitable processor architectures for a WLAN transceiver. 10800 architectures were evaluated by calculating mappability values for the main algorithms in the code. This led to an identification of the most suitable architecture for each algorithm, the best for all algorithms and sets of architectures that were feasible. The result of the comparison was that no single processor core was feasible to run all algorithms because of the differences in the algorithms. Thus, mappability increased with the number of processors up to 7 where only small improvements were gained when adding more processors. Conclusions were that mappability analysis alone could not find a balanced system. However, when performing some of the algorithms in hardware, a two-processor solution could be reasonable. This solution was also identified in the commercially available implementations of these algorithms.

Oh and Ha [24] propose to use heterogeneous multiprocessor (HMP) scheduling algorithms for co-synthesis in SoC design They state that their method is a middle-way between SoC research and the community of researchers involved in design of distributed heterogeneous embedded (DHE) systems. The technique is divided into two parts; a scheduler and a task- PE (Processing Engine) controller. Input to the system is a task-PE profile table that contains information about task deadlines, task execution time, cost and power consumption of the different PEs. They do not consider how the execution time of tasks for different processors is obtained. They also do not consider the delay in the links connecting PEs. When comparing their technique to others published the results show both lower costs and shorter evaluation time.

Beck and Siewiorek [6] propose a method which takes both task allocation and hardware specification simultaneously into consideration. They work with application requirements represented as a task-graph where various requirements of a task are described by numbers like CPU cycles/sec and memory requirements. These values are assumed to be estimated using a reference processor or by some other method. The same parameters are used when specifying the processing resources. The arcs in the task-graph represent the inter task communication which are specified by the dataflow in Bytes/Sec. The communication resources are specified with the bit-rate of the bus, exemplified with a CAN bus. It is assumed that the tasks are both statically predictable and statically assigned to the PEs. The task allocation is formulated as a vector packing problem where several tasks can be mapped onto one PE. The communication is similarly described as a scalar packing problem. In order to solve the problems they use a heuristic algorithm.

Dick and Jha [25] present a tool called MOCSYN which is used to map task-graphs to a library of cores and ICs. It also considers floor planning of blocks for more accurate estimation. Interestingly the first task is to select the clock frequency of the system. The worst-case performance of every task is recorded for every core. The method includes also physical aspects like size and power consumption.

5.2.2 Static Analysis at Low Abstraction Level

In our survey we have not been able to find any entries that can be classified into this category. The reason for this might be that no one exists or that we have not searched thorough enough.

5.2.3 Simulation at High Abstraction Level

Some ideas on using simulation at high abstraction level for evaluation of architectures are presented by Zivkovic et al. [13]. The method is called symbolic program - trace-driven simulation approach. It is a further refinement of a previous technique that used symbolic instructions as application representation. The new technique makes it possible to consider control flow in the program since the application is transformed into a structure similar to a CDFG. The operations that can be performed are the same as the in the earlier approach. The architecture is also modeled in a similar way, with the difference that it can now handle non-deterministic behavior that occurs when using the control information. The simulation results are also similarly obtained by running the application trace on the architecture.

Cassidy et al. [26] describe the MESH tool that can be used for high level simulation of heterogeneous systems. The purpose of the tool is to be a middle-way between the not accurate functional simulation and too detailed slow ISS. The application is instrumented with statements that are used by the simulator to interpret how the computational load affects a particular resource. These statements are captured from correlating the ISS model with the high-level MESH model. A network processor SoC is used as a model example for design space exploration. Nine different changes were made and evaluated resulting in errors of about 5%. In all cases relative performance changes was indicated correctly.

5.2.4 Simulation at Low Abstraction Level

The proposals in this category deal with cycle accurate simulation methods for multi-processor computing platforms [11] [27] [28].

The focus in Kin et al. [11] is on multimedia systems and the goal is to find the architecture that maximizes the performance considering constraints on area. Performance estimation is based on benchmark programs that are compiled with the retargetable IMPACT tool individually to each architecture configuration. The architecture is described using a high level machine description language that is interpreted by the IMPACT tool. Effect of cache memory misses is also considered in order to obtain more accurate results. It took about a week to simulate a total of 20 benchmarks, using 25 different cache combinations on 175 different machine configurations. The results show that when the area constraint is tighter, more different machine configurations are needed. Having a looser area constraint implies that a more general processor can be sufficient.

A simulator considering a homogenous multiprocessor system is presented by Forsell [27]. Every processor is a parametric instance of the MTAC processor, which among other features supports multiple threads and can simulate RISC processors clock synchronously. Both shared memory and message passing are considered as options for simulation. Network topology can be varied among three types of mesh configurations. Input to the simulator is assembler level code. Simulations show that the four MTAC processors are 2.7 times faster and occupied less program memory, than four DLX processors evaluated using the same configuration.

August et al. [28] is more of a conceptual paper presented within the MESCAL framework, although the focus of their current work is on network processors. It does not contain a detailed description of methods and does not present any case study. They propose the technique of retargetable compilation to capture architecture specific features. However the architecture is limited to a network of specialized VLIW machines. The method is intended to be an interface to Ptolemy II. In order to create a simulator Liberty is used. The simulator is also retargetable and can be used at different abstraction levels to improve speed.

6 Discussion

6.1 Processor Evaluation vs. Selection

The evaluation of a processor or a computing platform may be required in different contexts. An important context when it is used is to select or configure the computing platform. Selection follows performance evaluation in these approaches. Most of the single processor techniques we have surveyed consider only up to the evaluation of processors. In some cases selection is mentioned as an explicit goal of the technique. We have also noticed that most multiprocessor evaluation proposals are focusing on the selection part of the flow. In these methods it is assumed that analysis of the application has already been completed and the extracted performance is used in the selection stage. This division of work looks quite natural; developing one of these techniques could be complex enough as a research task.

6.2 Differences among Techniques in the Same Category

Several important differences can be found among techniques which are put in the same category in PEC classification. For example in the category of single processor static analysis at higher abstraction level, we see one major distinction. Gupta et al. [17] deal with average program execution time whereas Hergenhan and Rosenstiel [8] use worst case execution time of programs. Another important difference can be found in multiprocessor static analysis at high level. Soininen et al. [23] incorporates quite a detailed application-architecture matching technique while for example Dick and Jha [25] presume that execution time on each architecture is estimated by some external technique. Other distinctions exist among methods in other categories.

6.3 Mixed or Hybrid Techniques

The PEC platform assumes that techniques can be classified into one of the eight distinct categories. Some methods do not strictly lie in one category, since they might use the best points of the two options in an axis. Depending on the interpretation of these proposals these could however also be put into a single category. As an example of such a technique, Baghdadi et al. [12] propose a solution to reduce the evaluation time of multi-processor systems by using SDL (Specification and Description Language). They identify the POLIS approach being similar to theirs and having a mixed static/dynamic technique since it uses a combination of high-level simulations and low-level estimations. In order to capture timing information they used implemented RT-level parts and back-annotated timing to the system level description. The dynamic behavior is then simulated on the system level. The results show quite accurate results with error rate of less than 10%. However, the main advantage is the reduction in time; 10^4 times decrease in simulation time compared to an RTL co-simulation of the same system.

6.4 Evaluation and Selection for Multi-Processor SoCs

An interesting observation we have from our survey is that most of the techniques for multi-processor platform evaluation and selection use performance numbers obtained from single processor methods. These evaluation methods can lead to large errors. The performance of an N processor system can not be easily estimated from the performance estimation of individual processors executing individual tasks. The job of optimally selecting and configuring processor systems to execute a set of tasks is very hard and time consuming. This has resulted in a number of research groups involved in developing algorithms and techniques considering different aspects of design exploration. Recently Dwivedi et al. [29] presented a technique for automatic synthesis of multi-processor systems out of process networks. Szymanek et al. [30] explore execution time and energy consumption of an application, while considering memory organization. NoC architectures are becoming important options for implementing multiprocessor SoCs [1]. The task of specializing NoC for an application involves selecting many processing resources and finding relative positions of these resources. For proper specialization the evaluation technique should give accurate relative performance estimates of various options. This is not easy since the design space is very large. We identify the following as important issues in NoC performance evaluation:

- Accuracy: At least relative performance numbers are important
- Speed of evaluation: Cycle accurate seems out of reach

- Multiprocessor evaluation rather than combining individual evaluations of processors
- Memory: Accuracy in evaluation is highly dependant on memory organization

We have observed one tool for evaluation of NoC architectures [27]. Even in this technique the important aspect of NoC namely its packet switched communication and heterogeneous architecture is not properly handled. In NoC evaluation one must consider not only the communication architecture but also the fact that traffic intensity can vary, not only between the communicating peers, but it can also affect the available bandwidth of others in a dynamic way. The common bus example in Beck and Siewiorek [6] use a method where the communication is aggregated depending on if placements of the tasks are in different processing units. In this way the traffic on the bus is statically increased. If a packet switched NoC should be used we would have to take into account that only parts of the traffic added in the system would affect the other communication requirements in the network.

6.5 Limitations of PEC

It is possible to consider different or more aspects for classification of evaluation techniques than those used in PEC. As we have mentioned there are many aspects which are not captured by PEC. Models of applications and architectures used are important for accuracy of evaluation. One can also think about having more than two options in some axis. For example, in the abstraction axes one could add options of measuring time as absolute units, clock cycles or number of processor instructions.

7 Conclusions

The proposed PEC framework simplifies study of evaluation approaches for computing platforms by classifying them into different categories. Giving a structure to surveyed techniques makes it easier for new researchers, to manage the diversity of proposals in the area of processor evaluation. By using the classification it helps to understand how they relate to each other regarding three fundamental aspects. The classification also points to the emerging trends for evaluation of future multi-processor SoCs. We note that most of the emerging approaches for multi-processor SoC evaluation are either adapted from single processor evaluation approaches, or the approaches use simulators for evaluation at higher level of abstraction. In our classification we found only one approach missing; low level static multi-processor analysis. The reason for this could be difficulties in handling the complexity of the models. PEC puts many techniques in the same category although there are significant differences among them. A solution to this problem may be to add more axes or by adding more points in the existing axes. But then the simplicity of PEC will be lost.

Acknowledgements

The research reported in this paper was carried under the project “Specialization and Evaluation of Network on Chip Architectures for Multi-Media applications”, funded by Swedish KK Foundation.

References

[1] Jantsch and H. Tenhunen, editors. *Networks on Chip*. Kluwer Academic Publishers, February 2003.

[2] W. Wolf, *A Decade of Hardware/Software Codesign*, IEEE Computer, V36, No4, 2003

[3] K. Skadron, M. Martonosi, D. August, M. Hill, D. Lilja, V. Pai, *Challenges in Computer Architecture Evaluation*, *IEEE Computer*, pp. 30-36, Volume 36, No. 8, (August 2003).

[4] N. Cheung, S. Parameswaran, and J. Henkel, *INSIDE: INstruction Selection/Identification & Design Exploration for Extensible Processor*, *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, November 2003.

[5] Manoj Kumar Jain, M. Balakrishnan , Anshul Kumar, *ASIP Design Methodologies: Survey and Issues*, Proceedings of the 14th International Conference on VLSI Design (VLSID '01), p.76, January 03-07, 2001.

[6] J. E. Beck and D. P. Siewiorek, *Automatic Configuration of Embedded Multicomputer Systems*, IEEE Transactions on CAD of IC and Systems, feb, 2, pp. 84-95, volume 17, 1998.

[7] Flynn M. J.: *Some Computer Organizations and Their Effectiveness*, IEEE TRANS. ON COMPUTERS C-21, 9 (1972), 948-960.

[8] Hergenhan and W. Rosenstiel, *Static timing analysis of embedded software on advanced processor architectures*, In Proceedings of Design, Automation and Test in Europe, pages 552-559, Paris March 2000

[9] P. Giusto , G. Martin , E. Harcourt, *Reliable estimation of execution time of embedded software*, Proceedings of the conference on Design, automation and test in Europe, p.580-589, March 2001, Munich, Germany.

[10] Jeffry T Russell and Margarida F Jacome, *Architecture-Level Performance Evaluation of Component-Based Embedded Systems*, in Proceedings of DAC 2003

[11] J. Kin, C. Lee, W. Mangione-Smith, M. Potkonjak, *Exploring the Diversity of Multimedia Systems*, IEEE Transactions on VLSI Systems, Vol. 9, No.3, pp. 474-485, June 2001.

[12] Baghdadi A., Zergainoh N. -E., Cesario W., Roudier T., Jerraya A. A., *Design space exploration for hardware/software codesign of multiprocessor systems,* 11th IEEE International Workshop on Rapid System Prototyping (RSP'2000), Paris, France, 21-23 June 2000.

[13] V.D. Zivkovic, E.F.Deprettere, P.van der Wolf, E.A.de Kock *From High Level Application Specification to System-level Architecture Definition: Exploration, Design and Compilation*, Proc. of The International Workshop on Compilers for Parallel Computers (CPC03), pp. 39-49, January 8-10, 2003, Amsterdam, the Netherlands.

[14] Bart Kienhuis , Ed F. Deprettere , Pieter van der Wolf , Kees Vissers, *A methodology to design programmable embedded systems: the Y-chart approach*, Embedded processor design challenges: systems, architectures, modeling, and simulation-SAMOS, Springer-Verlag New York, Inc., New York, NY, 2002

[15] R. Desikan, D. Burger and S. Keckler, *Measuring experimental error in microprocessor simulation*, Proc. 28th Ann. Int’l. Symp. on Computer Arch. (ISCA), June/July 2001, pp. 266-277.

[16] Luigi Carro, Flavio R. Wagner, Marcio Kreutz, Marcio Oyamada, *A Design Methodology For Embedded Systems Based On Multiple Processors,* Proc. Distributed and Parallel Embedded Systems, Paderborn, Germany, Kluwer Academic Publishers, 2001.

[17] T. Vinod Kumar Gupta , Purvesh Sharma , M Balakrishnan , Sharad Malik, *Processor Evaluation in an Embedded Systems Design Environment*, Proceedings of the 13th International Conference on VLSI Design, p.98, January 04-07, 2000.

[18] Kaiyu Chen , Sharad Malik , David I. August, *Retargetable static timing analysis for embedded software*, Proceedings of the 14th international symposium on Systems synthesis, September 30-October 03, 2001, Montréal, P.Q., Canada .

[19] M. Lajolo, M. Lazarescu and A. Sangiovanni-Vincentelli, **A** *Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation,* In Proceedings of the 7th IEEE International Workshop on Hardware/Software Codesign, pp. 85-89, Roma, Italy, May 3-5, 1999.

[20] Todd Austin, Eric Larson, Dan Ernst, *SimpleScalar: An Infrastructure for Computer System Modeling*, IEEE Computer, pp. 59-67, 2002.

[21] G. Qu, M. Potkonjak, *System Synthesis of Synchronous Multimedia Applications*, IEEE Transactions on Embedded Computing Systems, Special Issue on Memory Systems, Vol. 2, No. 1, pp. 74-97, February 2002.

[22] Patterson, D. A., J. L. Hennessy, *Computer Architecture, a Quantitative Approach*, Morgan Kaufmann, San Mateo, CA., 1990.

[23] J. Soininen, J. Kreku, Y. Qu and M. Forsell, *Fast Processor Core Selection for WLAN Modem using Mappability Estimation*, Proceedings of the 10th International Symposium on Hardware/Software Codesign, May 6-8, 2002, Estes Park, Colorado, USA, 61-66

[24] Hyunok Oh and Soonhoi Ha, *A Hardware-Software Cosynthesis Technique Based on Heterogeneous Multiprocessor Scheduling*, 7th International Workshop on Hardware/Software CO-Design, Rome, Italy May 1999

[25] Robert P. Dick , Niraj K. Jha, *MOCSYN: multiobjective core-based single-chip system synthesis*, Proceedings of the conference on Design, automation and test in Europe, p.55-es, January 1999, Munich, Germany.

[26] Andrew S. Cassidy, JoAnn M. Paul, and Donald E. Thomas, Layered, Multi-Threaded, High-Level Performance Design, Design and Test, Europe, March, 2003

[27] Martti Forsell, *Advanced Simulation Environment for Shared Memory Network-on-Chips*, Proceeding of 20th IEEE Norchip Conference, Copenhagen, 11 - 12 Nov 2002.

[28] David I. August, Kurt Keutzer, Sharad Malik, and A. Richard Newton, *A Disciplined Approach to the Development of Platform Architectures*, Microelectronics Journal, Volume 33, Number 11, November 2002.

[29] Basant Kumar Dwivedi, Anshul Kumar and M. Balakrishnan, *Automatic Synthesis of System on Chip Multiprocessor Architectures for Process Networks*, International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2004), Stockholm, Sweden, September

[30] Radoslaw Szymanek, Francky Catthoor, and Krzysztof Kuchcinski, *"Time-Energy Design Space Exploration for Multi-Layer Memory Architectures"*, Proc. of the Design, Automation and Test in Europe, France, Paris, February 16-22, 2004.