This is the author's version of the work. The definitive work was published in *Proceedings of Design, Automation and Test in Europe (DATE 2009), pp. 472-477, 2009.* The work is supported in part by the German Science Foundation (DFG), SFB 694.

Combined System Synthesis and Communication Architecture Exploration for MPSoCs

Martin Lukasiewycz, Martin Streubühr, Michael Glaß, Christian Haubelt, and Jürgen Teich
University of Erlangen-Nuremberg, Germany
{martin.lukasiewycz,streubuehr,glass,haubelt,teich}@cs.fau.de

Abstract

In this paper, a novel design space exploration approach is proposed that enables a concurrent optimization of the topology, the process binding, and the communication routing of a system. Given an application model written in SystemC TLM 2.0, the proposed approach performs a fully automatic optimization by a simultaneous resource allocation, task binding, data mapping, and transaction routing for MPSoC platforms. To cope with the huge complexity of the design space, a transformation of the transaction level model to a graph-based model and symbolic representation that allows multi-objective optimization is presented. Results from optimizing a Motion-JPEG decoder illustrate the effectiveness of the proposed approach.

1. Introduction and Related Work

The increasing computational demands of modern applications often require their implementation on heterogeneous multiprocessor platforms. As a consequence, selecting the optimal platform and optimal mapping of the application to this platform is a challenging task. As these decisions are taken in early design phases, they have a huge impact on the quality of the resulting implementation. To substantiate these decisions, *design space exploration* is used to identify promising implementation options.

Many different approaches to *ESL* (*Electronic System-Level*) design space exploration have been published in recent years. They are either (1) platform-based, where resources are allocated from an architecture template and, subsequently, an application model is mapped to the resulting architecture obeying additional mapping constraints, or (2) multi-phased, assuming task partitioning is done before communication architecture exploration. Both approaches are shown in Fig. 1.

Platform-based design space exploration as, e.g., presented in [4, 5, 17, 18], performs the task of system synthesis, i.e., resource allocation and task binding. Nearly all automatic approaches follow the Y-chart approach [8] where the implementation platform is selected from an architecture template by allocating computational as well as communication resources. Data mapping is often not performed during design space exploration but postponed to the code generation phase. This, however, might result in an ad hoc and suboptimal communication synthesis step. As a consequence, an overestimated number of communication resources (buses, point-to-point connections, etc.) or oversized memories might be allocated deteriorating the overall chip area or performance (see Fig. 1(a)).

On the other hand, multi-phased approaches assume the task binding and (computational) resource allocation to be performed before running a second optimization, i.e., the communication architecture exploration. These approaches as, e.g., presented in [1, 3, 11, 13, 14], start with a so called *Virtual Architectural Model* [7]. During exploration, the topology is selected, i.e., communication resources and their interconnection to computational resources are determined. Simultaneously, the data mapping is performed. That way, the resulting communication architecture is adjusted to the task binding and data mapping (see Fig. 1(b)).

Most of the proposed approaches to communication architecture exploration use *TLMs* (*Transaction Level Models*) [2] with different degrees of time accuracy to do the necessary refinements. TLMs have shown to be particularly useful in virtual prototyping for performance estimation [6, 15]. However, separating system synthesis, i.e., resource allocation and task binding, from communication architecture exploration, might lead to suboptimal designs in multi-phase approaches. To overcome these drawbacks, a new platform-based ESL design space exploration for *MPSoCs* (*Multiprocessor System-on-Chip*) is proposed by combining system synthesis and communication architecture exploration. An overview of the proposed approach is shown in Figure 2.

For this purpose, the application model is located at the transaction level with threads communicating using transactions. The application model is mapped to an architecture template by simultaneously (1) allocating computational and communication resources, (2) binding tasks to computational resources, (3) mapping channels to memories, and (4) routing transaction in the resulting communication ar-

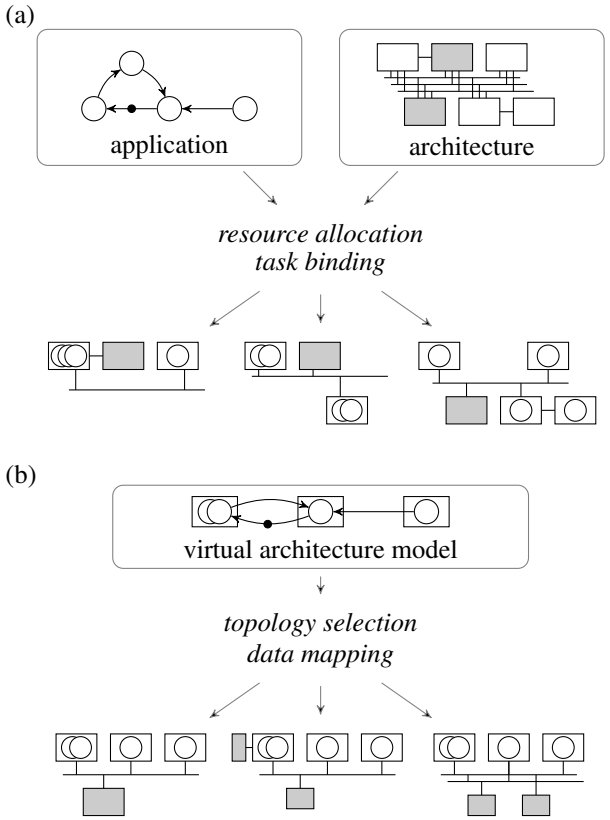

Figure 1. Two different approaches in ESL design space exploration can be distinguished: (a) The platform-based approach basically performs resource allocation and task binding. (b) The multi-phased approach basically performs bus selection and data mapping.

chitecture. In this novel one-phased approach, given mapping constraints imposed by, e.g., maximal computational load, maximal bus load, or maximal memory size, etc. are respected inherently. Former approaches rely on straightforward heuristic optimization algorithms, i.e., *Evolutionary Algorithms* and *Simulated Annealing*, and do in general not perform well for such hard constrained problems. In order to solve this problem appropriately, the design space is encoded symbolically by linear constraints with binary variables, and a state-of-the-art hybrid optimization approach [9, 10] is used to enable an adequate optimization of complex systems.

The remainder of the paper is organized as follows: Section 2 introduces the graph-based exploration model and its symbolic encoding. A transformation scheme from SystemC TLM 2.0 to the proposed exploration model is given in Section 3. In Section 4, a case study is presented that performs the proposed design space exploration for a Motion-JPEG decoder application on a multimedia architecture before the paper is concluded in Section 5.

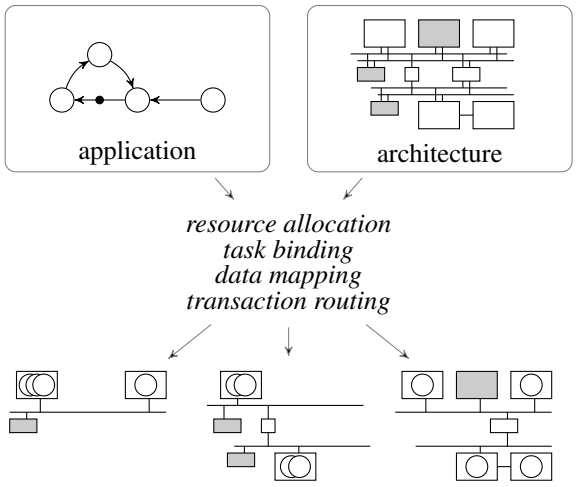

Figure 2. Proposed design space exploration: The application model is mapped to an architecture template by simultaneously allocating resources, binding tasks, mapping channels, and routing transactions.

2. Exploration

From a system synthesis point of view, the problem of mapping a dataflow model specified at transaction level to an MPSoC implementation is defined by the following steps:

1. Select the resources to set up the implementation platform, i.e., allocate processors, hardware accelerators, buses, memories, etc.
2. Bind each SystemC module (task) with its process to exactly one computational resource (processor or hardware accelerator) of the selected platform.
3. Bind each channel to exactly one memory of the selected platform.[1]
4. Route all transactions according to the computed binding and the topology of the selected platform.

In order to perform these steps automatically, we follow the commonly accepted Y-chart approach [8] and use a symbolic decoding strategy to efficiently derive implementations from the resulting design space.

In the following, the graph-based exploration model is presented. A symbolic encoding as linear constraints is introduced such that each solution equals a feasible implementation. This symbolic representation in combination with state-of-the-art optimization approach allows an optimization of multiple conflicting and even non-linear objectives.

[1] It is possible to map the communication to computational resources if the resource can establish the communication internally.

2.1. Exploration Model

The exploration model is defined by a *specification* that consists of an *application* and an *architecture*. From this specification, various *implementations* can be derived by defining the *allocation* of the architecture and the *mapping* of the application.

The specification consists of an architecture graph G_R and an application graph G_T :

- The architecture is given by a directed graph $G_R(R, E_R)$. The vertices R represent resources such as processors, memories, or buses. The directed edges E_R indicate available communication connections between two resources.
- The application is given by a bipartite directed graph $G_T(T, E_T)$ with $T = P \cup C$. The vertices T are either process tasks $p \in P$ or communication tasks $c \in C$. Each edge $e \in E_T$ connects a vertex in P to one in C, or vice versa. Each process task can have multiple incoming edges that indicate the data-dependencies to communication information of the predecessor communication tasks. On the other hand, each communication task has exactly one predecessor process task as the sender, but a process task can of course have multiple successor communication tasks. To allow multicasts, each communication task can have multiple successor process tasks.

  Each process task $p \in P$ can be implemented on a resource from R_p with $R_p \subseteq R$. Each communication task $c \in C$ can be routed on a subset of resources from R_c with $R_c \subseteq R$.

One implementation consists of the allocation graph G_A that is deduced from the architecture graph and a function i that maps the application onto the allocation graph.

- The allocation is a directed graph $G_A(A, E_A)$ that is an induced subgraph of the architecture graph G_R. The allocation contains all resources that are available in the current implementation and the edges are induced from the graph G_R such that G_A is aware of the communication connections.
- Each process task $p \in P$ is bound to exactly one allocated resource $i(p)$ such that $i(p) \in (A \cap R_p)$. Each communication task in $c \in C$ is routed on a tree that is a subgraph of the allocation such that $i(c) \subseteq G_A$ with all vertices in R_c. These bindings and routings have to be performed such that all data-dependencies given by the following two conditions are satisfied:

  1. For each communication task $c \in C$, the root of the routing has to equal the binding of the predecessor sender process task $p \in P$. It holds:
  $$\forall (p, c) \in E_T : root(i(c)) = i(p)$$

  2. For each process task $p \in P$ the routings of the predecessor communication tasks $c \in C$ have to be routed on the same resource as the binding of process p. It holds:
  $$\forall (c, p) \in E_T : i(p) \in i(c)$$

An implementation is *feasible* if all requirements regarding the process and communication mapping as well as the data-dependencies are fulfilled.

With the definition of a feasible implementation, the task of the *design space exploration* can be formulated as the following multi-objective optimization problem:

Definition 1 (Design Space Exploration)

optimize $f(\mathbf{x})$
subject to:
$\mathbf{x}$ is a feasible *implementation*

In real-world problems, the objective function f consists of multiple functions including also non-linear calculations. In single-objective optimization, the feasible set of networks is totally ordered, whereas in multi–objective optimization problems, the feasible set is only partially ordered and, thus, there is generally not only one global optimum, but a set of *Pareto solutions*. A Pareto-optimal solution is better in at least one objective when compared to any other feasible solution.

2.2. Encoding

In the following, a binary search problem is defined such that a solution $\mathbf{x}$ corresponds to a *feasible* implementation x. The symbolic encoding consists of the following binary variables:

- $\mathbf{r}$ - one variable for each resource $r \in R$ indicating whether this resource is in the allocation (1) or not (0).
- $\mathbf{p_r}$ - a set of variables for each process task $p \in P$ and the available resources $r \in R_p$ indicating whether the process task is bound on the resource (1) or not (0).
- $\mathbf{c_r}$ - a set of variables for each communication task $c \in C$ and the available resources $r \in R_c$ indicating whether the communication task is routed over the resource (1) or not (0).
- $\mathbf{c_{r,n}}$ - additional variables for each communication and resource pair indicating on which communication step $n \in \mathbf{N}$ (communication tasks are propagated in steps) a communication is routed over the resource.

The linear constraints are formulated as follows:
$\forall p \in P$:

$$\sum_{r \in R_p} \mathbf{p_r} = 1 \tag{1a}$$

$\forall c \in C$:

$$\sum_{r \in R_c} \mathbf{c_{r,0}} = 1 \tag{1b}$$

$$\forall c \in C, p \in \{\tilde{p} | (\tilde{p}, c) \in E_T\}, r \in R_p \cap R_c :$$

$$\mathbf{p_r} - \mathbf{c_{r,0}} = 0 \tag{1c}$$

$$\forall p \in P, c \in \{\tilde{c} | (\tilde{c}, p) \in E_T\}, r \in R_p \cap R_c :$$

$$\mathbf{c_r} - \mathbf{p_r} \geq 0 \tag{1d}$$

$$\forall c \in C, r \in R_c :$$

$$\mathbf{c_{r,1}} + \mathbf{c_{r,2}} + ... + \mathbf{c_{r,n}} \leq 1 \tag{1e}$$

$$\mathbf{c_{r,1}} + \mathbf{c_{r,2}} + ... + \mathbf{c_{r,n}} - \mathbf{c_r} \geq 0 \tag{1f}$$

$$\forall c \in C, r \in R_c, i = \{1, ..., n\} :$$

$$\mathbf{c_r} - \mathbf{c_{r,i}} \geq 0 \tag{1g}$$

$$\forall c \in C, r \in R_c, i = \{1, .., n-1\} :$$

$$-\mathbf{c_{r,i+1}} + \sum_{\tilde{r} \in R_c \wedge e = (\tilde{r}, r) \in E_R} \mathbf{c_{\tilde{r},i}} \geq 0 \tag{1h}$$

$$\forall p \in P, r \in R_p :$$

$$\mathbf{r} - \mathbf{p_r} \geq 0 \tag{1i}$$

$$\forall c \in C, r \in R_c :$$

$$\mathbf{r} - \mathbf{c_r} \geq 0 \tag{1j}$$

$$\forall r \in R :$$

$$-\mathbf{r} + \sum_{c \in C \wedge r \in R_c} \mathbf{c_r} + \sum_{p \in P \wedge r \in R_p} \mathbf{p_r} \geq 0 \tag{1k}$$

Equation (1a) ensures that each process task is bound exactly once. The Equations (1b) and (1c) imply that each communication task has exactly one root that equals the used resource of the predecessor process task. Analogously, for each process task the predecessor communication tasks have to be routed on the corresponding resources as stated in Equation (1d). Equation (1e) ensures that a communication task can pass a resource at most once such that no loops occur. A communication task has to be existent in one communication step on a resource in order to be correctly routed on this resource as implied by the Equations (1f) and (1g). Equation (1h) states that a communication is only possible between adjacent resources. The Equations (1i) and (1j) imply that a process or communication task, respectively, is bound or routed on an allocated resource only. On the other hand, Equation (1k) states that a resource is only allocated if at least one process is bound or a communication is routed on this resource. Moreover, this representation allows additional linear or linearizable constraints like, e.g., on maximal computational load, maximal bus load, or maximal memory size for each resource.

Given a single solution $\mathbf{x}$ of this linear search problem, the corresponding implementation x is deduced by constructing the allocation from the $\mathbf{r}$ variables, the binding for each process task from the $\mathbf{p_r}$ variables, and the routing of the communication task from the $\mathbf{c_r}$ and $\mathbf{c_{r,n}}$ variables.

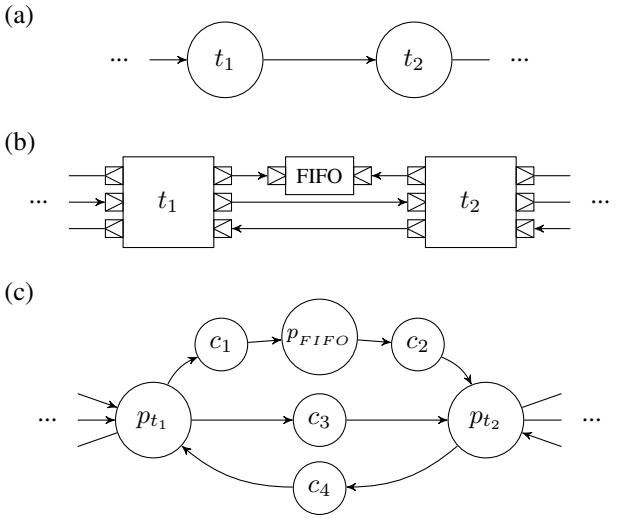

Figure 3. Example of a transformation from a (a) dataflow graph to the corresponding (b) TLM and (c) exploration model with $p_{t_1}, p_{t_2}, p_{FIFO} \in P$ and $c_1, c_2, c_3, c_4 \in C$.

2.3. Optimization

Common optimization approaches that are based on *Integer Linear Programs* or *Evolutionary Algorithm* only are either restricted to a single linear objective function or do not perform well on optimization problems with many constraints and few feasible solutions. With the binary encoding and linear constraint from the previous section, the design space exploration problem as stated in Def. 1 can be carried out efficiently by using the heuristic *SAT decoding* optimization approach [9]. This hybrid optimization approach based on an *Evolutionary Algorithm* and a *PB (pseudo boolean) solver* allows the optimization of multiple conflicting and non-linear objectives under linear constraints in a binary search space. The optimization approach iteratively improves found implementations such that with a higher runtime and more evaluated implementations, respectively, the quality of the results increases.

3. Transformation

In order to apply the design space exploration automatically, the specification has to be given. While the architecture graph and mapping constraints provided by a system engineer can be directly embedded in the specification, most multimedia applications are represented by dataflow models and have to be transformed into an application graph. Dataflow models are typically already represented as directed graphs, where vertices represent actors and edges represent queues with FIFO semantics.

In order to support an efficient simulation, an intermediate representation of these dataflow models at the transaction level, i.e., SystemC TLM 2.0 is aspired. For that purpose, automatic approaches exist [3]. Otherwise, a straightforward reimplementation following the scheme shown in Figure 3 is possible as well: Each actor of the dataflow model corresponds to a single thread encapsulated in a single SystemC module. Each communication queue is modeled by a SystemC channel. The sending and receiving threads are connected to the channel via socket bindings. Moreover, the sending and receiving threads are directly connected to each other in order to allow the exchange of synchronization information, e.g., availability of new data on the channel.

To transform the transaction level model into an application graph, the following steps are mandatory: Each thread and each channel is represented by exactly one process task. Using the mapping constraints, threads are restricted to be mapped to computational resources, e.g., processors and hardware accelerators, while channels are commonly restricted to be mapped to memory resources like, e.g, on-chip or off-chip RAM. Each socket binding is represented by exactly one communication task connecting the process task representing the sending thread and the process task respresenting the receiving thread. This transformation can be performed automatically, cf. Figure 3.

4. Case Study

In this case study, the proposed design space exploration is applied to a multimedia system. In particular, a Motion-JPEG decoder application is mapped onto a multimedia architecture template. The objectives of the optimization are *throughput* in terms of frames per second (fps), *latency* in terms of milliseconds, and *chip area* in terms of the number of gates. The performance evaluation of each candidate implementation is carried out by a fast high-level simulation that dynamically annotates the performance characteristics from a selected allocation to the application [16]. The chip area is approximated by a linear cost function. The task execution times are derived from another case study investigating the Codesign of a Motion-JPEG decoder on Xilinx FPGA platforms including a MicroBlaze soft-core processor [5]. The task execution times in software are taken from the MicroBlaze processor software implementation and are scaled using the typical MIPS values for the ARM9 and ARM11 processors (e.g. 35MIPS@50MHz for a MicroBlaze processor and 220MIPS@200MHz for an ARM9 processor). Additionally, artificial area values are annotated to the architecture template reflecting the relative ratio of the resources in terms of chip area. These area values are inspired by typical chip size values obtained from processor specifications and gate count values for the hardware modules from the mentioned case study. Although these performance and area values are not accurate enough to make irrevocable decisions from the exploration results, these inputs are suitable to explore the complex design space and to find possible high quality implementation alternatives.

	area (gates)	throughput (fps)	avg. latency (ms)
1	33,090	29.5	34.5
2	25,285	15.1	68.7
3	17,063	10.3	100.4

Table 1. Objective values for the implementations with highest throughput, lowest area consumption as well as a balanced trade-off implementation.

The application model of the decoder is specified at transaction-level using 21 SystemC single-threaded modules and 56 SystemC channels for data storage communication between modules. An architecture template incorporates 12 multi-purpose processing elements and communication resources, as well as 10 dedicated hardware accelerators connected via 36 point to point communication resources including storage memory. Here, the set of multi-purpose processing elements consists of two ARM9 and two ARM11 processors-cores, one DSP, three memories, two buses, and one gateway connecting these buses.

All application tasks can be mapped to any of the ARM processors or the DSP. Additionally, some tasks known to be critical like, e.g., the *discrete cosine transformation*, *Huffman decoding* can be mapped to application-specific hardware accelerators. The communication queues can be mapped to memories or to hardware communication resources.

A sequence of images in SQCIF-format (128×96 pixels) is used as input stimuli for performance simulation. The exploration was performed on a 2.66 GHz quad-core Intel Xeon Server with 8GB of RAM using the optimization framework OPT4J [12] that includes the suggested *SAT decoding* optimization approach. In 17 hours total runtime[2], 7600 candidate architectures were evaluated, resulting in 27 high quality implementations from which a designer can choose the most suitable implementation. A projection of the area and throughput for these implementations is shown in Fig. 4. Note that three implementations provide a throughput of more than 24fps and, thus, fulfill current real-time video requirements.

Exemplarily, all objective values for the implementations with highest throughput, lowest area consumption as well as a balanced trade-off implementation are given in Table 1.

Discussion

The case study shows that the proposed methodology is applicable to a typical multimedia multiprocessor system. Although the application and the architecture template seem

[2]More than $99,9\%$ of the runtime is spent on the simulative evaluation of the performance values.

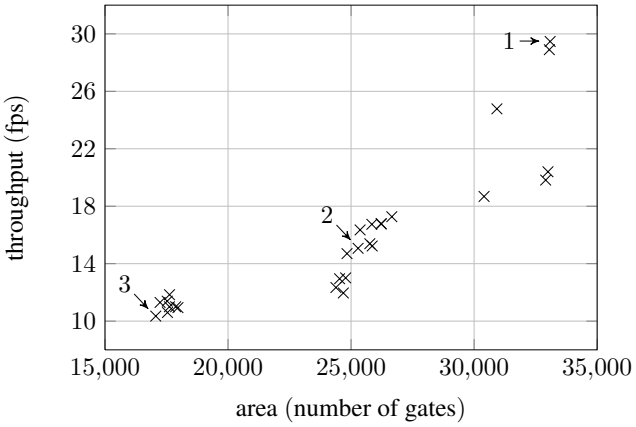

Figure 4. The two dimensional projection of the best found implementations showing the trade-off between area and throughput.

manageable at first glance, the design space exploration reveals a quite large design space and, thus, a great potential for an optimization. Moreover, a huge diversity among the delivered high quality implementations can be observed. This allows the designer to make an accurate decision on which implementation should be the base for further design refinements.

Compared to the runtime of the simulative evaluation of the implementations, the optimization algorithm requires only 26 seconds in total. With the known scalability analysis for the applied optimization approach [9], this shows that the model and encoding for the design space exploration is capable of handling even considerably larger applications and architectures than provided in this case study.

5. Conclusion

In this paper, a novel ESL design space exploration on the basis of the Y-chart approach is presented that combines common system synthesis and communication architecture exploration approaches. The model explicitly distinguishes between process and communication tasks for the application and, therefore, also multihop and multicast communication for the resulting implementations is enabled. The presented binary encoding into linear constraints allows an efficient multi-objective exploration with a novel optimization approach. Moreover, a transformation from dataflow models and TLMs to this exploration model is presented. A case study of a Motion-JPEG decoder for an MPSoC multimedia system shows the efficiency of the proposed approach by automatically delivering several high quality implementations in a reasonable amount of time.

References

[1] N. Bombieri, F. Fummi, and D. Quaglia. TLM/Network Design Space Exploration for Networked Embedded Systems. In *Proc. of CODES+ISSS '06*, pages 58–63, 2006.

[2] L. Cai and D. Gajski. Transaction level modeling: an overview. In *Proc. of CODES+ISSS '03*, pages 19–24, 2003.

[3] A. Gerstlauer, J. Peng, D. Shin, D. Gajski, A. Nakamura, D. Araki, and Y. Nishihara. Specify-Explore-Refine (SER): From Specification to Implementation. In *Proc. of DAC '08*, pages 586–591, 2008.

[4] T. Givargis, F. Vahid, and J. Henkel. System-Level Exploration for Pareto-Optimal Configurations in Parameterized Systems-on-a-Chip. In *Proc. of ICCAD '01*, pages 25–30, 2001.

[5] C. Haubelt, M. Meredith, T. Schlichter, and J. Keinert. SystemCoDesigner: Automatic Design Space Exploration and Rapid Prototyping from Behavioral Models. In *Proc. of DAC '08*, pages 580–585, June 2008.

[6] Y. Hwang, S. Abdi, and D. Gajski. Cycle-approximate Retargetable Performance Estimation at the Transaction Level. In *Proc. of DATE '08*, pages 3–8, 2008.

[7] K. Huang, et al. Simulink-Based MPSoC Design Flow: Case Study of Motion-JPEG and H.264. In *Proc. of DAC '07*, pages 39–42, June 2007.

[8] B. Kienhuis, E. Deprettere, K. Vissers, and P. van der Wolf. An Approach for Quantitative Analysis of Application-Specific Dataflow Architectures. In *Proc. of ASAP '97*, pages 338–349, July 1997.

[9] M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich. SAT-Decoding in Evolutionary Algorithms for Discrete Constrained Optimization Problems. In *Proc. of CEC '07*, pages 935–942, 2007.

[10] M. Lukasiewycz, M. Glaß, C. Haubelt, J. Teich, R. Regler, and B. Lang. Concurrent Topology and Routing Optimization in Automotive Network Integration. In *Proc. of DAC '08*, pages 626–629, June 2008.

[11] B. H. Meyer and D. E. Thomas. Simultaneous Synthesis of Buses, Data Mapping and Memory Allocation for MPSoC. In *Proc. of CODES+ISSS '07*, pages 3–8, 2007.

[12] Opt4J. The optimization framework for java. http://www.opt4j.org/, Version 1.3.

[13] S. Pasricha, N. Dutt, and M. Ben-Romdhane. Extending the Transaction Level Modeling Approach for Fast Communication Architecture Exploration. In *Proc. of DAC '04*, pages 113–118, 2004.

[14] S. Pasricha, N. Dutt, and M. Ben-Romdhane. Using TLM for Exploring Bus-based SoC Communication Architectures. In *Proc. of ASAP '05*, pages 79–85, 2005.

[15] J. Schnerr, O. Bringmann, A. Viehl, and W. Rosenstiel. High-Performance Timing Simulation of Embedded Software. In *Proc. of DAC '08*, pages 290–295, 2008.

[16] M. Streubühr, J. Falk, C. Haubelt, J. Teich, R. Dorsch, and T. Schlipf. Task-Accurate Performance Modeling in SystemC for Real-Time Multi-Processor Architectures. In *Proc. of DATE '06*, pages 480–481, Munich, Germany, Mar. 2006.

[17] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. Design Space Exploration of Network Processor Architectures. In *In Network Processor Design: Issues and Practices, Volume 1*, pages 30–41, 2002.

[18] M. Thompson, H. Nikolov, T. Stefanov, A. D. Pimentel, C. Erbas, S. Polstra, and E. F. Deprettere. A Framework for Rapid System-Level Exploration, Synthesis, and Programming of Multimedia MP-SoCs. In *Proc. of CODES+ISSS '07*, pages 9–14, 2007.