Estonian Journal of Engineering, 2011, **17**, 2, 158–168 doi: 10.3176/eng.2011.2.06

System-level optimization of NoC-based timing sensitive systems

Mihkel Tagel, Peeter Ellervee, Thomas Hollstein and Gert Jervan

Department of Computer Engineering, Tallinn University of Technology, Ehitajate tee 5, 19086 Tallinn, Estonia; {mihkel.tagel, peeter.ellervee, thomas.hollstein, gert.jervan}@ati.ttu.ee

Received 7 February 2011, in revised form 29 March 2011

Abstract. Communication modelling and synthesis plays an important role in the design of complex network-on-chip based timing-sensitive systems-on-chip. Trying to guarantee the observance of timing constraints without detailed know-how of communication transactions might lead to unexpected results. In our previous work we have proposed a system level approach for communication modelling and synthesis to calculate hard communication deadlines based on communication delay models and on guidance of the scheduling process to take into account possible network conflicts. In this paper we combine our communication scheduling approach with global optimization techniques to perform design space exploration and/or improvement of the synthesized schedule.

Key words: network-on-chip, system-on-chip, communication modelling, design space exploration, system-level optimization.

1. INTRODUCTION

The trend of integrating an ever increasing number of components on the chip has led to the chip architectures where the on-chip communication infrastructure is not anymore bus-based but resembles more computer networks. Such networks-on-chip (NoC) provide to a designer a flexible, scalable, and unified layered communication platform [1]. In addition, new integration methodologies have lead to new 3D architectures, where the dies are stacked into 3-dimensional structures, thus providing even higher densities and complexity. In such NoC-based systems the communication is achieved by routing packets through the network infrastructure, rather than routing global wires. However, communication parameters (inter-task communication volume, link latency and bandwidth, buffer size) might have major impact to the performance of applications implemented on NoCs. Therefore, in order to guarantee predictable behaviour and to satisfy performance constraints of real-time systems, careful selection of application partitioning, mapping and synthesis algorithms is required. In the context of this paper, we mean by communication synthesis the mapping of data packets to the network links and the timing of the release of the packets on the links, i.e. calculating spatial and temporal properties of the application communication tasks with respect to the given system architecture at early stages of the design flow. Earlier we have proposed an approach for communication modelling and synthesis to calculate communication hard deadlines, based on communication delay models, and to guide the scheduling process to take into account possible network conflicts [2,3]. However, the quality of schedules has not been addressed so far. Nor have we taken into account the influence of the task mapping to the scheduling process.

Mapping tasks to processing cores is similar to the quadratic assignment problem (QAP) that is known to be computationally intensive [4]. To run an exhaustive search to find a global optimum is infeasible for such complex problems. Heuristics shall be used to approximate the optimum in a reasonable amount of time. Various optimization techniques have been described by several authors to solve the mapping or scheduling problem [5–13]. However, different assumptions on communication modelling do not allow the heuristics to be used one-to-one in our system-level design framework. In this paper we combine our communication modelling approach with global optimization techniques to perform design space exploration and improvement of synthesized schedules. The paper is organized as follows. In Section 2 we introduce our system-level model and the NoC platform. In Section 3 we describe the usage of two schedule optimization techniques and design space exploration in our framework. Experimental results are given in Section 4 followed by conclusions.

2. SYSTEM-LEVEL MODEL

One of the important points in the design space exploration is speed. One could simulate an application to get more accurate results, but this is usually slow. Therefore, we abstract the implementation details of the network-on-chip and perform design space exploration on a high level of abstraction. However, our approach keeps the communication modelling as precise as possible.

Input to our system-level design flow is application A, NoC architecture N and application mapping M (Fig. 1). Application is specified by a directed acyclic graph $A=(T,C)$, where $T=\{t_i \mid i=1,\dots,T\}$ is a set of vertexes representing non pre-emptive tasks, and $C=\{c_{i,j} \mid (i,j)\in \{1,\dots,V\} x \{1,\dots,V\}\}$ is a set of edges representing communication between tasks. Each task t_i is characterized by the worst case execution time (WCET) $Wcet_i$. Tasks can be completed earlier than their WCET but communication between the tasks needs to be initiated at their respective scheduled time periods to avoid network conflicts. This is controlled by the sender network interface. Therefore, no customization of routers is needed. NoC platform introduces communication latency that depends not only on the message size, but also on the resource map-

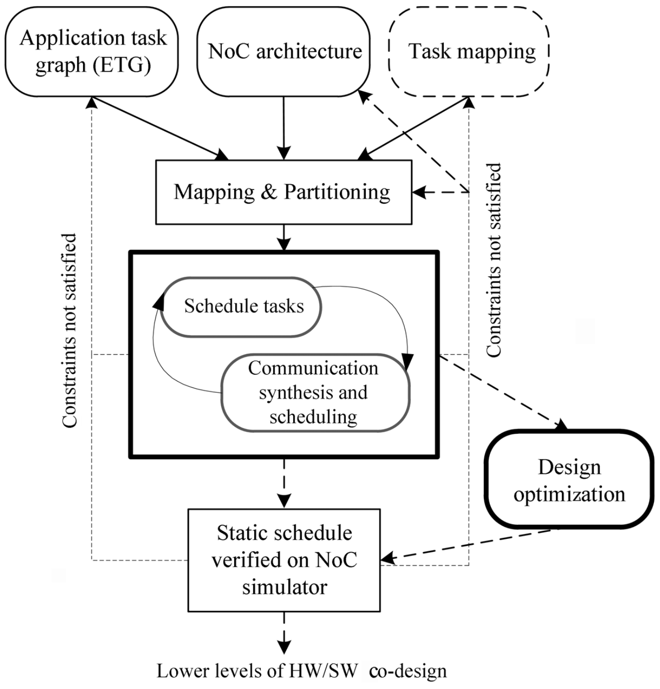
Application task graph (ETG)
NoC architecture
Task mapping
Mapping & Partitioning
Schedule tasks
Communication synthesis and scheduling
Constraints not satisfied
Constraints not satisfied
Design optimization
Static schedule verified on NoC simulator
Lower levels of HW/SW co-design

Fig. 1. System-level design flow.

ping and needs to be taken into account. Therefore, in addition to message size the edge is characterized by communication delay (CD) $Cd_{i,j}$. We assume that the application has dummy start and end vertices.

The NoC architecture can be modelled as a directed graph $N=(R,L)$, where $R=\{r_k \mid k=1,\ldots,R\}$ is a set of resources and $L=\{l_{k,l} \mid (k,l)\in \{1,\ldots,R\}x\{1,\ldots,R\}\}$ is a set of links connecting a pair of resources (k,l). The mapping M of an application A is represented by the function $M(T\rightarrow R)$. The architecture is characterized by operating frequency, topology, routing algorithm, switching method, link bit-width and the delay model of the network interface and routers.

We assume that each processing core is controlled by a scheduler that takes care of the task execution on the core and schedules the message transfers between the tasks. Otherwise a task that completes earlier than its calculated WCET and starts a message transfer could lead to an unexpected network congestion and have a fatal effect on the global execution schedule. The schedule is calculated offline and a partial schedule is stored in each local scheduler memory. We assume that the size of an input buffer is one packet in the case of virtual cut-through or store-and-forward and one flit in the case of a wormhole switching method. No output buffering is used. The input buffer of one flow control unit together with its incoming communication link can be considered as one shared resource during the communication synthesis. It also requires minimum hardware resources from a router. To have a predictable communication model, we assume the use of deterministic routing algorithms. In our experiments we are using a dimension-ordered XY routing. Our communication modelling, synthesis and scheduling approach is explained in greater detail in [2,3].

Once the tasks have been mapped to the architecture, an iterative process starts (Fig. 1). It consists of communication synthesis and task scheduling. The produced schedule is verified by running an application simulation on a NoC simulator or a design space exploration performed that will try to improve the task mapping and schedule. If the design requirements are met, the lower levels of HW/SW co-design processes continue. Otherwise changes are needed in the architecture or in the mapping.

The goal of this paper can be formulated as follows. Given application A, a NoC architecture N and a mapping $M(T \rightarrow R)$, our goal is to perform design space exploration and optimization of the initial design with respect to schedule length and to produce a set of feasible near-optimum design options. At the same time, we also measure the calculation time that is used to define the trade-off between the improvement, gained during optimization and the time spent for calculation. Our goal is not to propose any enhancements to optimization methods, but to show that our communication modelling and synthesis approach can be applied to arbitrary scheduling or optimization method.

3. DESIGN SPACE EXPLORATION AND OPTIMIZATION

Networks-on-chip are flexible communication platforms with computation being decoupled from communication. The available flexibility increases the amount of design options that need to be explored. Application mapping has a major impact on the schedule length, NoC performance and power consumption [1]. Depending on the level of freedom there are two options – to explore different application mappings or to improve a schedule, based on a given mapping. Our ultimate goal is to perform design space exploration and optimization at the same time.

First we will describe the approach to a given application with a fixed mapping on a NoC. The initial schedule is produced by applying a modified list scheduling as described in [2,3]. List scheduling is a greedy heuristics using a priority list and precedence constraints to schedule the tasks and to minimize the schedule length. The algorithm is straightforward to implement and it has a low calculation time. The algorithm could be also implemented on-chip, allowing dynamic re-configuration and resulting in improved application fault tolerance. However, to get an optimum solution, one would need to schedule all possible task sequences, reaching in worst case *n!* permutations. We are using branch-and-bound and simulated annealing global optimization techniques to explore the trade-off between improvement in schedule length and time spent for calculation. Second, we will perform a design space exploration with simulated annealing in order to compare it to the schedule optimization.

3.1. Schedule optimization with branch-and-bound

Branch-and-bound (B&B) is a general algorithm for finding optimal solutions for various computationally intensive optimization problems. Branch-and-bound consists of three main functions – branching, bounding and pruning. Branching describes the problem as a search tree, whose nodes are subsets of the given problem. Bounding calculates the upper and lower bounds that are used to evaluate a set of candidates and to prune the ones that do not lead to an optimum.

We are exploiting B&B in the following way. We maintain a partial schedule and a list of ready tasks for every B&B tree node. We have a sorted list of all partial schedule lengths and we are choosing a node to branch, based on a best-first strategy. We have also evaluated a breadth-first search, but it did not improve the calculation speed. Each ready task of a node being expanded will be added into the B&B search tree and stored also in the partial schedule list. Such a branching strategy avoids infeasible solutions and reduces the number of calculations. The partial schedule length is calculated, based on the order of tasks in the partial schedule list. The rest of the tasks are scheduled by list scheduling that gives us a tight upper bound. The lower bound B_l is calculated as in [5]:

$$B_\mathrm{l} = \frac{\sum \mathrm{WCET_t}}{\sum N_\mathrm{pc}} + S_\mathrm{p}, \qquad (1)$$

where WCET is the total of unscheduled tasks, N_pc is the number of processing cores and S_p is the partial schedule length.

After that, we evaluate all candidate solutions. We prune the nodes that have their lower bounds higher or equal to the global best upper bound. The process continues until there are no more nodes to expand. An example of the branch-and-bound calculation process is depicted in Fig. 2. The application consists of

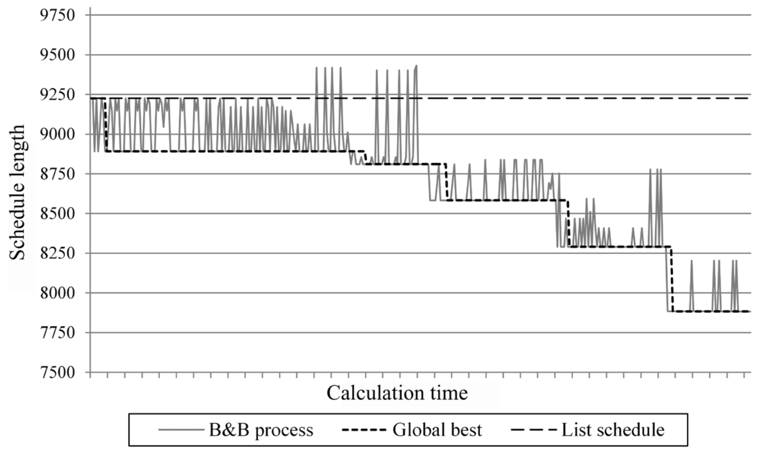

Fig. 2. An example of branch-and-bound result conversion.

100 tasks mapped on a 6×6 2D-mesh NoC. List schedule length is 9226 μs while branch-and-bound result gives almost 15% of improvement (7882 μs). However, the calculation time increases 71 times from 0.12 to 8.53 s.

3.2. Schedule optimization with simulated annealing

Simulated annealing (SA) is a probabilistic metaheuristic for the global optimization problem, locating a near-optimal solution in a feasible amount of time [14]. The simulated annealing comprises creating an initial solution that will be annealed by generating moves in the neighbourhood. Result (cost) of a move is calculated based on a target function and compared to the currently known best value. In our case the cost function is the application schedule length. The cooling schedule controls the decrease of temperature, which has an effect on the move energy. The process continues until a termination condition has been met.

In simulated annealing it is important to find feasible values for initial parameters: the initial temperature (related with initial acceptance probability) and the cooling schedule. When the initial temperature is too high, many bad uphill moves might be accepted driving SA far away from the reasonable solution space. When it is too low, SA might not reach solutions, which require more energy to cross higher hills to reach a global optimum. To estimate the initial temperature, we use the approach, described in [15]. First, we create an initial solution. Next, we generate n random moves and record the difference between the initial solution and the random move and calculate the average difference (D_{av}). The initial temperature T_{in} can be estimated as

$$T_{\text{in}} = (1/-\log(p_{\text{in}}))D_{\text{av}}^2, \tag{2}$$

where p_{in} is the initial probability to accept uphill moves. In our experiments we have used $p_{\text{in}} = 0.9$. We have used an exponential cooling schedule, described by the following equation:

$$T_{\text{new}} = \alpha T, \tag{3}$$

where α is a constant (in our experiments usually between 0.7–0.96) and T is the current temperature.

An acceptance criterion is needed to overcome local optimums and accept uphill moves. One of the most common is the Metropolis acceptance criterion:

$$p_{\text{acc}} = e^{-\Delta E/kT}, \tag{4}$$

where ΔE is the difference of the object function between the modified solution and the current one and k is the Boltzman constant [16]. The acceptance probability p_{acc} is compared to a random value, generated uniformly in the range of $[0, 1]$. The probability of accepting uphill moves decreases with the temperature and with the increase of ΔE. Therefore selection of the initial temperature, probability and cooling schedule is important for the performance of SA.

When optimizing the schedule (having fixed mapping) with simulated annealing, a neighbouring solution is created by selecting randomly a task and randomly increasing or decreasing its priority in the ready list. For this we record during scheduling the tasks that are ready at the same time. The distance M_{rt}, to which a random task can be shifted in the queue, is controlled by the following formula:

$$M_{\mathrm{rt}} = Q_{\mathrm{rl}}(T/T_{\mathrm{in}}), \tag{5}$$

where Q_{rl} is the size of the ready list.

In literature various termination conditions can be found. In our approach we specify the number of temperature levels N_{t} we are annealing at minimum, until we stop the process. The counter is incremented when we have found a better or equal solution compared to the previous one. The counter is reset to zero when a higher result is found as we might have recovered from a local optimum and it would need further examination. At each temperature we are creating at minimum N_{t} random neighbourhood solutions. The counter is incremented when an uphill move is rejected and decremented when a solution is accepted. The idea is essentially to keep the process on the same energy level until it stabilizes and only then lower the temperature. The pseudo-code of simulated annealing is depicted in Fig. 3.

```
Simlated Annealing (alpha, $T_{initial}$, $N_{allowed\_temperatures}$, $N_{allowed\_peturbations}$)
Construct initial solution $S_{current}$;
Current temperature $T = T_{initial}$;
Global best solution $S_{global} = \infty$;  $N_{temperatures} = 0$;

While $N_{temperatures} < N_{allowed\_temperatures}$ Do Begin
    Number of peturbations $N_{peturb} = 0$;
    While $N_{peturb} < N_{allowed\_peturbations}$ Do Begin
        Generate randomly a neighbouring solution $S_{neighbor}$
        Calculate $S_{neighbor}$ schedule length
        $\Delta E = S_{neighbor} - S_{current}$;

        If $\Delta E<0$ Then
            $S_{current} = S_{neighbor}$;  $N_{peturb} = 0$;
        Else
            Generate random value r = uniform_random(0, 1);
            If $r \le e^{-\Delta E/T}$ Then
                $S_{current} = S_{neighbor}$;  $N_{peturb} = N_{peturb} - 1$;
            Else
                $N_{peturb} = N_{peturb} + 1$;
            End If;
        End If;
    End;

    If $S_{current} <= S_{global}$ Then
        $S_{global} = S_{current}$;  $N_{temperatures} = N_{temperatures} + 1$;
    Else
        $N_{temperatures} = 0$;
    End If;
    Set T = alpha*T;
End;
End Simulated Annealing;
```

Fig. 3. Pseudo-code of simulated annealing.

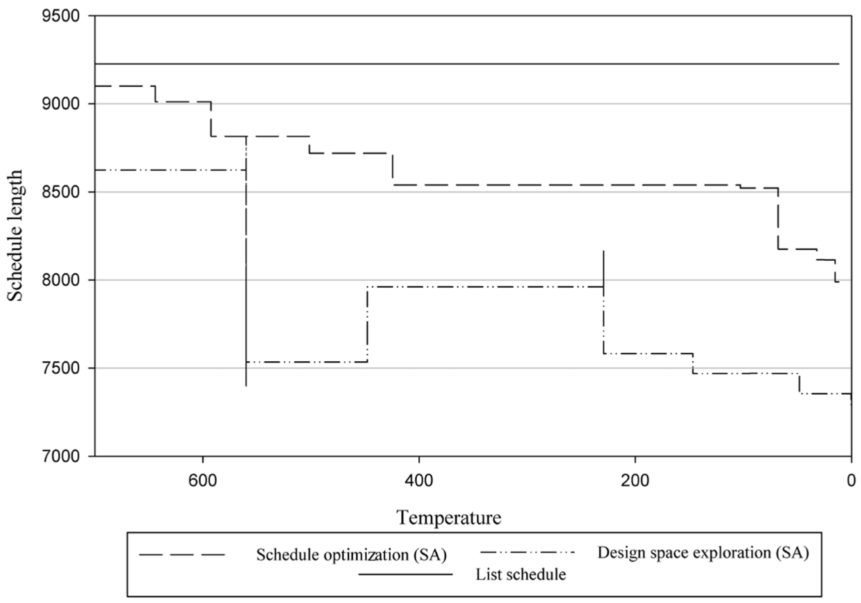

Fig. 4. An example of simulated annealing results.

Figure 4 depicts example results of simulated annealing having the same application and platform as with branch-and-bound. The simulated annealing schedule length is 7988 μs, which is 13% better than a schedule produced by list scheduling (9226 μs) and close to the branch-and-bound result (7882 μs). The SA calculation time (96 s) is 800 times higher than for list scheduling and 11 times higher compared to B&B.

3.3. Design space exploration with simulated annealing

When schedule optimization does not give the required amount of improvement we can go for design space exploration and find an alternative mapping that could give better results. We are using the simulated annealing approach for that. Design space exploration is performed by selecting randomly a task and re-mapping it to another processing core. The distance between original and re-mapped core (number of hops) is controlled by cooling. Initially, the distance can be higher while eventually reaching one hop (nearby cores). To evaluate cost of the move we need to perform each time communication synthesis and schedule the tasks and the communication. Figure 4 depicts combined results of a simulated annealing schedule optimization and a design space exploration. It can be seen that the design space exploration resulted in a mapping that gave the best schedule length 7264 μs, compared to B&B (7882 μs). However, the calculation time was around 11 times longer (96 s) compared to B&B. We have performed an experiment were application with 100 tasks and 30 arbitrary mappings were given and design exploration was run with simulated annealing. An average schedule length of 7324 μs was achieved, which is less than 1% different from the best result. It shows that general purpose heuristics can be used effectively to solve specific design space exploration problems.

4. EXPERIMENTAL RESULTS

We have built a design environment that supports the system-level model and optimization framework described in the previous sections. We have chosen synthetic task graphs, containing 100, 500, 750 and 1000 tasks, to show scaling of the global optimization on big applications. The NoC platform we have used is a 2D 6×6 mesh, having an operating frequency of 500 MHz, a link bit-width of 32 bits, a packet-size of 512 bits (flit-size 32 bits), and a wormhole switching with dimension-ordered XY routing. The computing resources are homogeneous – the task WCET is the same on all resources. The tests were performed on a computer with Intel L2400 CPU (1.66 GHz), 1 GB of available physical RAM and operating system Microsoft Windows XP. As simulated annealing depends heavily on the quality of the random number generator, we have run 20 tests in a batch varying the random number generator seed. We depict the minimum, maximum and average result. The SA initial parameters were chosen as described in Section 3.2. Task mapping was generated by an external tool for each application and the same mapping was used in all experiments.

The results in Table 1 show that all the optimization techniques have produced a better schedule than the list scheduling (LS). However, this is achieved at the cost of increased calculation time. For applications with 750 and 1000 tasks, the schedule optimization with SA was not able to produce a result in feasible amount of time. When we compare different optimization techniques it can be seen, that branch-and-bound has given from 10% to 15% of improvement with lowest calculation time. As we create only valid solution branches in the B&B, this reduces rapidly the number of iterations, reaching the solution faster than simulated annealing. The simulated annealing performance could be increased by a more efficient way to generate the neighbourhood, where a change of the existing solution is created. Design space exploration with SA reached

Table 1. Comparison of the optimization techniques

Number of tasks	Shedule length, μs								Calculation time, min			
	List shedule	Branch-and-bound	SA			Exploration with SA			LS	B&B	SA	Expl. SA
			Min	Avg	Max	Min	Avg	Max				
100	9 226	7 882	7 988	8 488	8 719	7 264	7 367	7 998	0.003	0.1	1.3	1.8
500	18 496	16 728	18 019	18 112	18 206	16 916	17 522	18 265	0.03	13.7	34.7	18.2
750	32 127	28 932	NA	NA	NA	28 150	29 217	30 193	0.05	35.5	NA	33.6
1000	36 772	33 103	NA	NA	NA	33 642	33 733	34 086	0.10	85.9	NA	267.3

a similar schedule optimization as B&B, but the calculation time explodes with the increase of the application size. It is important to note, that in the design space exploration communication must be synthesized for each modified solution. In the schedule optimization, it is needed only to re-schedule the tasks and communication that is less time consuming.

5. CONCLUSIONS

In this paper we have presented branch-and-bound and simulated annealing optimization techniques in order to estimate and improve the schedule length and to perform design space exploration in our system-level design framework. The framework models communication at the link level, using a traditional task graph based modelling technique and supporting various switching techniques. The results show, that this communication modelling technique can be used with both of the presented optimization methods, gaining improvement in schedule length by re-ordering or re-mapping the tasks.

REFERENCES

1. Marculescu, R., Ogras, U. Y., Li-Shiuan Peh, Jerger, N. E. and Hoskote, Y. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2009, **28**, 3–21.
2. Tagel, M., Ellervee, P. and Jervan, G. System-level communication synthesis and dependability improvements for network-on-chip based systems. *Estonian J. Eng.*, 2010, **16**, 23–38.
3. Tagel, M., Ellervee, P., Hollstein, T. and Jervan, G. Communication modelling and synthesis for NoC-based systems with real-time constraints. In *Proc. IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*. Cottbus, Germany, 2011, 237–242.
4. Garey, M. R. and Johson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H.Freeman Publ., New York, 1979.
5. Rahman, M. M. and Chowdhury, M. Examining branch and bound strategy on multiprocessor task scheduling. In *Proc. International Conference on Computer and Information Technology*. Dhaka, Bangladesh, 2009, 162–167.
6. Kazem, A., Rahmani, A. M. and Aghdam, H. H. A modified simulated annealing algorithm for static task scheduling in grid computing. In *Proc. International Conference on Computer Science and Information Technology*. Singapore, 2008, 623–627.
7. Orsila, H., Salminen, E., Hännikäinen, M. and Hämäläinen, T. D. Optimal subset mapping and convergence evaluation of mapping algorithms for distributing task graphs on multi-processor SoC. In *Proc. International Symposium on System-on-Chip*. Tampere, Finland, 2007, 1–6.
8. Talbi, E.-G. and Muntean, T. Hill-climbing, simulated annealing and genetic algorithms: a comparative study and application to the mapping problem. In *Proc. 26th Hawaii International Conference on System Sciences*. Wailea, Hawaii, 1993, 565–573.
9. Lee, C. and Bic, L. On the mapping problem using simulated annealing. In *Proc. 8th Annual International Phoenix Conference on Computers and Communications*. Scottsdale, Arizona, USA, 1989, 40–44.
10. Lu, Z., Xia, L. and Jantsch, A. Cluster-based simulated annealing for mapping cores onto 2D mesh networks on chip. In *Proc. 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*. Bratislava, Slovakia, 2008, 1–6.

11. Nanda, A. K., DeGroot, D. and Stenger, D. L. Scheduling directed task graphs on multiprocessors using simulated annealing. In *Proc. 12th International Conference on Distributed Computing Systems*. Yokohama, Japan, 1992, 20–27.
12. Murali, S., Benini, L. and De Micheli, G. Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees. In *Proc. Asia and South Pacific Design Automation Conference*. Shanghai, China, 2005, 27–32.
13. Ascia, G., Catania, V. and Palesi, M. An evolutionary approach to network-on-chip mapping problem. In *Proc. 2005 IEEE Congress on Evolutionary Computation*. Edinburgh, Scotland, 2005, 112–119.
14. Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. Optimization by simulated annealing. *Science*, 1983, **220**(4598), 671–680.
15. Ledesma, S., Avina, G. and Sanchez, R. Practical considerations for simulated annealing implementation. In *Simulated Annealing* (Tan, C. M., ed.). InTech, 2008.
16. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N. and Teller, A. H. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 1953, **21**, 1087–1092.

Kiipvõrkudel põhinevate keerukate kiipsüsteemide optimeerimine süsteemitasemel

Mihkel Tagel, Peeter Ellervee, Thomas Hollstein ja Gert Jervan

Kommunikatsiooni modelleerimisel ja sünteesil on oluline roll keerukate, kiipvõrkudel põhinevate kiipsüsteemide disainil. Ilma detailse arusaamiseta kiipidevahelisest kommunikatsioonist on aga raske anda hinnangut süsteemi ajalisele käitumisele ja garanteerida selle vastavust nõuetele. Artiklis on esitatud kommunikatsiooni modelleerimise ja sünteesi meetod, mis võimaldab leida andmeülekandeks kuluva aja, võttes seejuures arvesse võimalikke võrgukonflikte. On kirjeldatud selle kommunikatsiooni modelleerimise meetodi kombineerimist erinevate globaalsete optimeerimisalgoritmidega eesmärgiga leida efektiivsem ülesannete planeering ja jaotus kiipsüsteemi protsessoritel.