International Journal of Parallel Programming, Vol. 35, No. 2, April 2007 (© 2006)
DOI: 10.1007/s10766-006-0029-7

Fault-aware Communication Mapping for NoCs with Guaranteed Latency

Sorin Manolache,[1,2] Petru Eles,[1] and Zebo Peng[1]

Received August 15, 2006; revised October 18, 2006; accepted November 1, 2006

As feature sizes shrink, transient failures of on-chip network links become a critical problem. At the same time, many applications require guarantees on both message arrival probability and response time. We address the problem of transient link failures by means of temporally and spatially redundant transmission of messages, such that designer-imposed message arrival probabilities are guaranteed. Response time minimisation is achieved by a heuristic that statically assigns multiple copies of each message to network links, intelligently combining temporal and spatial redundancy. Concerns regarding energy consumption are addressed in two ways. First, we reduce the total amount of transmitted messages, and, second, we minimise the application response time such that the resulted time slack can be exploited for energy savings through voltage reduction. The advantages of the proposed approach are guaranteed message arrival probability and guaranteed worst case application response time.

KEY WORDS: Networks-on-chip; communication synthesis; transient link failures.

1. INTRODUCTION

Shrinking feature sizes make possible the integration of millions and soon billions of transistors on multi-core chips. For example, new technologies such as Extreme Ultraviolet Lithography promise to deliver feature sizes of 20 nm.(1) This allows for single-chip implementations of extremely complex, computation-intensive applications, such as advanced signal

[1]Department of Computer and Information Science, Linköping University, S-581 83, Linköping, Sweden.
[2]To whom correspondence should be addressed. E-mail: sorma@ida.liu.se

processing in, for example, the military or medical domains, high-quality multimedia processing, high-throughput network routing, and high-traffic web services.

However, these technological capabilities do not come without unprecedented challenges to the design community. These challenges include increased design and verification complexity and high-power density. At this high-integration level, effects such as capacitive cross-talk, power supply noise, and neutron and alpha radiation [2,3] lead to non-negligible rates of transient failures of interconnects and/or devices, jeopardising the correctness of applications.

Several authors[4–6] have proposed network-on-chip (NoC) architectures as replacements to bus-based designs in order to improve scalability, reduce design, verification and test complexity and to ease the power management problem.

With shrinking feature size, the on-chip interconnects have become a performance bottleneck.[7] Thus, a first concern, which we address in this article, is *application latency*.

The energy consumption of wires has been reported to account for about 40% of the total energy consumed by the chip.[8] This is a strong incentive to consider the communication energy reduction by means of efficient utilisation of the on-chip communication channels. Thus, a second concern is communication energy.

A third problem arising from shrinking feature size is the increasing rate of transient failures of the communication lines. The reliability of network nodes is guaranteed by specific methods, which are outside the scope of this work. In general, 100% reliable communication cannot be achieved in the presence of transient failures, except under assumptions such as no multiple simultaneous faults or at most n bit flips, which are unrealistic in the context of complex NoC. Hence, we are forced to tolerate occasional errors, provided that they occur with a rate below an imposed threshold. Thus, a third concern is to ensure an imposed communication reliability degree under constraints on application latency, while keeping energy consumption as low as possible.

We address the three identified problems, namely energy reduction and satisfaction of timeliness and communication reliability constraints, by means of *communication synthesis*. In this context, synthesising the communication means mapping data packets to network links. The selection of message routes has a significant impact on the responsiveness of applications implemented on the NoC. The communication reliability is ensured by deploying a combination of spatially and temporally redundant communication. This however renders the communication mapping problem particularly difficult.

The article is structured as follows. The next section surveys related work. Section 3 introduces the notation used throughout the article and presents the system model. Section 4 contains the precise formulation of the problem that we solve in this article. Section 5 outlines our solution approach. Sections 6, 7 and 8 dive into various specifics of our approach, while Section 9 presents the experimental results that validate our approach. The last section draws the conclusions.

2. RELATED WORK

Communication synthesis greatly affects performance and energy consumption. Closest to our approach, which maps data packets to network links in an off-line manner, is deterministic routing.[9,10] One of its advantages is that it may guarantee deadlock-free communication and the communication latency and energy consumption are easier to predict. Nevertheless, deterministic routing can be efficiently applied only if traffic patterns are known in more detail at design time. Under the assumptions that we make in this article, the communication mapping (and the deterministic routing that results from it) is complicated by the fact that we deploy redundant communication.

Wormhole routing[11] is a popular switching technique among NoC designs. However, an analysis that would provide bounds on its latency and/or energy consumption has yet to be devised. Therefore, we will assume virtual cut-through switching,[12] whose analysis we present in Section 7.

As opposed to deterministic routing, Dumitraş and Mărculescu[13] have proposed stochastic communication as a way to deal with permanent and transient faults of network links and nodes. Their method has the advantage of simplicity, low-implementation overhead, and high robustness w.r.t. faults. However, their method suffers the disadvantages of non-deterministic routing. Thus, the selection of links and of the number of redundant copies to be sent on the links is stochastically done at runtime by the network routers. Therefore, the transmission latency is unpredictable and, hence, it cannot be guaranteed. More importantly, stochastic communication is very wasteful in terms of energy.[14]

Pirretti et al.[15] report significant energy savings relative to Dumitraş' and Mărculescu's approach, while still keeping the low-implementation overhead of non-deterministic routing. An incoming packet is forwarded to exactly one outgoing link. This link is randomly chosen according to pre-assigned probabilities that depend on the message source and destination. However, due to the stochastic character of transmission paths

and link congestion, neither Dumitraş and Mărculescu, nor Pirretti et al. can provide guarantees on the transmission latency.

As opposed to Dumitraş and Mărculescu and Pirretti et al., who address the problem of reliable communication at system-level, Bertozzi et al.[16] address the problem at on-chip bus level. Bertozzi's approach is based on low-swing signals carrying data encoded with error resilient codes. They analyse the trade-off between consumed energy, transmission latency and error codes, while considering the energy and the chip area of the encoders/decoders. While Bertozzi et al. address the problem at link level, in this article we address the problem at application level, considering time-constrained multi-hop transmission of messages sharing the links of an NoC.

In this article, we address all of the three identified stringent problems: link reliability, latency and energy consumption. We propose a solution for the following problem: Given an NoC architecture with a failure probability for its network links and given an application with required message arrival probabilities and imposed deadlines, find a mapping of messages to network links such that the imposed message arrival probability and deadline constraints are satisfied at reduced energy costs.

Our approach differs from the approaches of Dumitraş and Mărculescu[13] and of Pirretti et al.[15] in the sense that we *deterministically* select *at design time* the links to be used by each message and the number of copies to be sent on each link.

Thus, we are able to guarantee not only message arrival probabilities, but also worst-case message arrival times. In order to cope with the unreliability of on-chip network links, we propose a way to combine spatially and temporally redundant message transmission. Our approach to communication energy reduction is to minimise the application latency at almost no energy overhead by intelligently mapping the redundant message copies to network links. The resulting time slack can be exploited for energy minimisation by means of voltage reduction on network nodes and links.

3. SYSTEM MODEL

This section introduces the notation used throughout the article and presents the system model.

3.1. Hardware Model

We describe the system model and introduce the notations based on the example in Fig. 1. The hardware platform consists of a 2D array of $W \times H$ cores, depicted as squares in the figure, where W and H denote

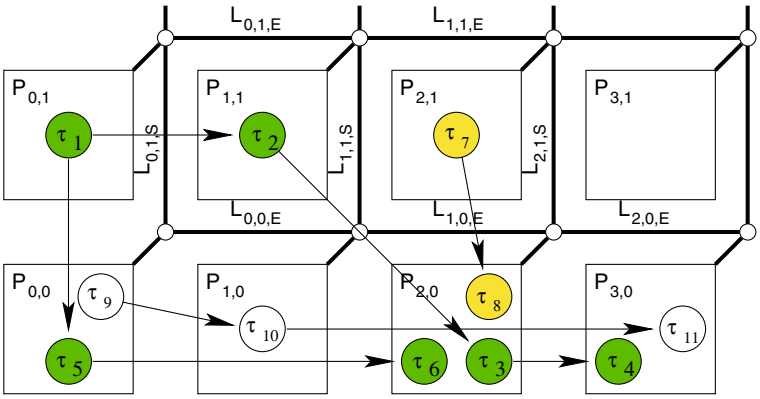

Fig. 1. Application example.

the number of columns and rows of the array, respectively. The cores are denoted with $P_{x,y}$, where x is the 0-based column index and y is the 0-based row index of the core in the array. The inter-core communication infrastructure consists of a 2D mesh network. The small circles in Fig. 1 depict the switches, denoted $S_{x,y}$, $0 \leq x < W$, $0 \leq y < H$. Core $P_{x,y}$ is connected to switch $S_{x,y}$, $\forall 0 \leq x < W$, $0 \leq y < H$. The thick lines connecting the switches denote the communication links. Each switch, except those on the borders of the 2D mesh, contains five input buffers: one for the link connecting the switch to the core with the same index as the switch, and the rest corresponding to the links conveying traffic from the four neighbouring switches.

The link connecting switch $S_{x,y}$ to switch $S_{x,y+1}$ is denoted with $L_{x,y,N}$ while the link connecting switch $S_{x,y+1}$ to switch $S_{x,y}$ is denoted with $L_{x,y+1,S}$. The link connecting switch $S_{x,y}$ to switch $S_{x+1,y}$ is denoted with $L_{x,y,E}$ while the link connecting switch $S_{x+1,y}$ to switch $S_{x,y}$ is denoted with $L_{x+1,y,W}$. Each link is characterised by the time and energy it needs to transmit a bit of information.

3.2. Application Model

The functionality of an application is modelled as a set of tasks, denoted with $T = \{\tau_1, \tau_2, \ldots, \tau_N\}$. Tasks are graphically represented as circles, as shown in Fig. 1.

Data dependencies among tasks are graphically depicted as arrows from the sender task to the receiver task, as shown in Fig. 1.

The task that sends the message is the *predecessor* of the receiving task, while the receiving task is the *successor* of the sender. The set of predecessors of task τ is denoted with $^{\circ}\tau$, while the set of successors of task τ with τ°. For illustration (Fig. 1), task τ_2 has one predecessor, namely task τ_1, and one successor, namely task τ_3.

Let us consider a sequence of tasks $(\tau_1, \tau_2, \ldots, \tau_k)$, $k > 1$. If there exists a data dependency between tasks τ_i and τ_{i+1}, $\forall 1 \leq i < k$, then the sequence $(\tau_1, \tau_2, \ldots, \tau_k)$ forms a *computation path* of length k. We say that the computation path leads from task τ_1 to task τ_k. Task τ_i is an *ancestor task* of task τ_j if there exists a computation path from task τ_i to task τ_j. Complementarily, we say that task τ_i is a *descendant task* of task τ_j if there exists a computation path from task τ_j to task τ_i. We do not allow circular dependencies, i.e. no task can be both the ancestor and the descendant of another task. In Fig. 1, $(\tau_1, \tau_2, \tau_3, \tau_4)$ is an example of a computation path of length 4, and task τ_2 is an ancestor of tasks τ_3 and τ_4.

The set of tasks of an application is partitioned into g partitions. Any two tasks within the same partition have a common ancestor or a common descendant or they are in a predecessor–successor relationship. Two tasks belonging to different partitions have no common ancestor, nor any common descendant, neither are they in a predecessor–successor relationship.

The task partitions are denoted V_1, V_2, ..., V_g. An application consists of a set $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_g\}$ of g *task graphs*, $\Gamma_i = (V_i, E_i \subset V_i \times V_i)$, $1 \leq i \leq g$. A directed edge $(\tau_a, \tau_b) \in E_i$, $\tau_a, \tau_b \in V_i$, represents the data dependency between tasks τ_a and τ_b, denoted $\tau_a \rightarrow \tau_b$.

The application in Fig. 1 consists of three task graphs, namely $\Gamma_1 = (\{\tau_1, \tau_2, \ldots, \tau_6\}, \{(\tau_1, \tau_2), (\tau_2, \tau_3), (\tau_3, \tau_4), (\tau_1, \tau_5), (\tau_5, \tau_6)\})$, $\Gamma_2 = (\{\tau_7, \tau_8\}, \{(\tau_7, \tau_8)\})$, and $\Gamma_3 = (\{\tau_9, \tau_{10}, \tau_{11}\}, \{(\tau_9, \tau_{10}), (\tau_{10}, \tau_{11})\})$.

Task *instantiations* (also known as *jobs*) *arrive* periodically.

Let $\Pi = \{\pi_i \in \mathbb{R} : \tau_i \in T\}$ denote the set of *task periods*, or job inter-arrival times, where π_i is the period of task τ_i. Tasks belonging to the same task graph have the same period. For any task $\tau_j \in T$, the fact that the kth instantiations of tasks τ_i, for all tasks $\tau_i \in {}^{\circ}\tau_j$, have completed their execution is a necessary condition for the kth instantiation of task τ_j to start its execution.

Tasks are *mapped* on cores. All instances of a task are executed by the same core on which the task is mapped.

Let Map: $T \rightarrow P$ be a surjective function that maps tasks on the cores. $\text{Map}(\tau_i) = P_{x,y}$ indicates that task t_i is executed on the core $P_{x,y}$.

Let $\mathcal{T}_{P_{x,y}} = \{\tau \in T : \text{Map}(\tau) = P_{x,y} \in P\}$ denote the set of tasks that are mapped on core $P_{x,y}$.

The mapping is graphically indicated by placing the circle that represents the task inside the box that represents the core on which the task is mapped. In Fig. 1, task τ_1 is mapped on core $P_{0,1}$, task τ_2 on core $P_{1,1}$, task τ_7 on core $P_{2,1}$, tasks τ_5 and τ_9 on core $P_{0,0}$, task τ_{10} on core $P_{1,0}$, tasks τ_3, τ_6, and τ_8 on core $P_{2,0}$, and tasks τ_4 and τ_{11} on core $P_{3,0}$.

For any task τ_i, $\forall 1 \leq i \leq N$, let $BCET_i$ and $WCET_i$ denote its best-case and, respectively, worst-case execution times on core $\text{Map}(\tau_i)$.

The real-time requirements are expressed in terms of deadlines. Let $\Delta_T = \{\delta_1, \delta_2, \ldots, \delta_N\}$ denote the set of *task deadlines*, where δ_i is the deadline of task τ_i. If the kth job of task τ_i has not completed its execution at time $k \cdot \pi_i + \delta_i$, then the job is said to have missed its deadline. Let $\Delta_\Gamma = \{\delta_{\Gamma_1}, \delta_{\Gamma_2}, \ldots, \delta_{\Gamma_g}\}$ denote the set of *task graph deadlines*, where δ_{Γ_i} is the deadline of task graph Γ_i. If there exists a task $\tau \in V_j$ such that its kth instantiation has not completed its execution at time $k \cdot \pi + \delta_{\Gamma_j}$, where π is the period of any task $\tau \in V_j$, then the kth instantiation of task graph Γ_j has missed its deadline.

In the common case of more than one task mapped on the same core, a scheduler selects the next task to run based on the priority associated to the task off-line. Job execution is preemptible, i.e. a higher priority job may preempt the execution of a lower priority job.

3.3. Communication Model

Communication between pairs of tasks mapped on different cores is performed by message passing. Their transmission on network links is done packet-wise, i.e. the message is chopped into packets, which are sent on links and reassembled at the destination core. Messages are characterised by their priority, length (number of bits), and the size of the packets they are chopped into.

If an output link of a switch is busy sending a packet while another packet arrives at the switch and demands forwarding on the busy link, the newly arrived packet is stored in the input buffer corresponding to the input link on which it arrived. When the output link becomes available, the switch picks the highest priority packet that demands forwarding on the output link. If an output link of a switch is not busy while a packet arrives at the switch and demands forwarding on the free link, then the packet is forwarded immediately, without buffering. This scheme is called virtual cut-through routing.[12]

Packet transmission on a link is modelled as a task, called communication task. The worst-case execution time of a communication task is

given by the packet length divided by the link bandwidth. The execution of communication tasks is non-preemptible. The priority of a communication task modelling the transmission of a packet is equal to the priority the packet. The packet priority is equal to the message priority which in turn is assumed to be given by the designer.

3.4. Fault Model

Communication links may temporarily malfunction, with a given probability. If a data packet is sent on the link during the time the link is in the failed state, the data is scrambled. We assume that the switches have the ability to detect if an incoming packet is scrambled. Scrambled packets are dropped as soon as they are detected and are not forwarded further. Several copies of the same packet may be sent on the network links. In order for a message to be successfully received, at the destination core, at least one copy of every packet of the message has to reach the destination core unscrambled. Otherwise, the message is said to be lost.

We define the message arrival probability of the message $\tau_i \rightarrow \tau_j$ as the long term ratio $\text{MAP}_{i,j} = \lim_{t\to\infty} \frac{S_{i,j}(t)}{\lfloor t/\pi_i \rfloor + 1}$, where $S_{i,j}$ is the number of messages between tasks τ_i and τ_j that are successfully received at the destination in the time interval $[0, t)$, and π_i denotes the period of the sender task. For each pair of communicating tasks $\tau_i \rightarrow \tau_j$, the designer may require lower bounds $B_{i,j}$ on the ratio of messages that are received unscrambled at the destination.

Let us assume that switches have the capability to detect erroneous (scrambled) packets, but they do not have the capacity to correct these errors. We let α denote the probability of a packet to traverse a network link unscrambled. A strategy to satisfy the constraints on the message arrival probability ($\text{MAP}_{i,j} \geq B_{i,j}$) is to make use of spatially and/or temporally redundant packet transmission, i.e. several copies of the same packet are simultaneously transmitted on different paths and/or they are resent several times on the same path. This strategy is discussed in this article.

An alternative strategy to cope with transient faults on the network links is to add redundant bits to the packets for error correction. Additionally, extra circuitry has to be deployed at the switches such that they can correct some of the erroneous packets. In this case, let α denote the probability of a packet successfully traversing a link, which means that

- the packet traverses the network link unscrambled, *or*
- the packet is scrambled during its transmission along the link, but the error correction circuitry at the end of the link is able to correct the scrambled bits.

Note that the two strategies are orthogonal, in the sense that the redundant transmission can be deployed even when error correction capability is present in the network. In this case, redundant transmission attempts to cope with the errors that are detected but cannot be corrected by the correction circuitry. An analysis of the trade-off between the two strategies is beyond the scope of the article.

In the sequel, we will make use of α, the probability of a packet to successfully traverse a link, abstracting away whether the successful transmission is due to error correction or not.

3.5. Message Communication Support

In order to satisfy message arrival probabilities imposed by the designer, temporally and/or spatially redundant communication is deployed. We introduce the notion of *communication supports (CS)* for defining the mapping of redundant messages to network links. For this purpose, we use the example in Fig. 1. A possible mapping of messages to network links is depicted in Fig. 2. The directed lines depicted parallel to a particular link denote that the message represented by the directed line is mapped on that link. Thus, message $\tau_1 \rightarrow \tau_2$ is conveyed by link $L_{0,1,E}$, message $\tau_1 \rightarrow \tau_5$ by link $L_{0,1,S}$, message $\tau_7 \rightarrow \tau_8$ by link $L_{2,1,S}$, message $\tau_9 \rightarrow \tau_{10}$ by link $L_{0,0,E}$, message $\tau_5 \rightarrow \tau_6$ by links $L_{0,0,E}$ and $L_{1,0,E}$, message $\tau_{10} \rightarrow \tau_{11}$ by links $L_{1,0,E}$ and $L_{2,0,E}$.

Of particular interest are messages $\tau_3 \rightarrow \tau_4$ and $\tau_2 \rightarrow \tau_3$. *Two* identical copies of the former are sent on the same link, namely link $L_{2,0,E}$, as indicated by the double arrow between task τ_3 and τ_4 in the figure.

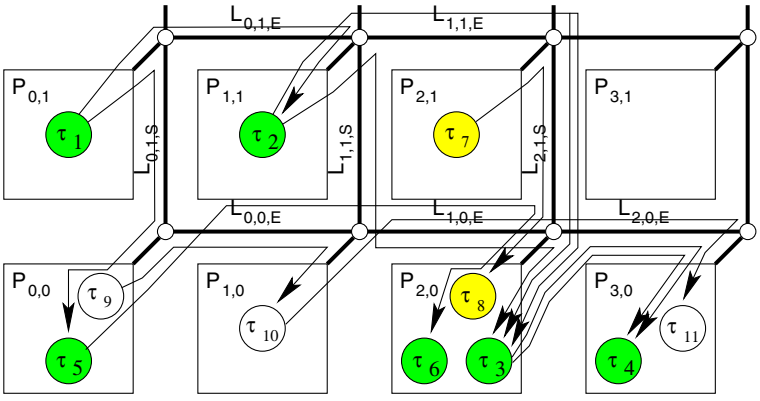

Fig. 2. Message mapping for the application in Fig. 1.

Therefore, the transmission of message $\tau_3 \rightarrow \tau_4$ is temporally redundant. A more complex case is exemplified by message $\tau_2 \rightarrow \tau_3$. Identical copies of the message take different routes. Therefore, the transmission of message $\tau_2 \rightarrow \tau_3$ is spatially redundant. One copy is conveyed by links $L_{1,1,E}$ and $L_{2,1,S}$, while the second copy is conveyed by links $L_{1,1,S}$ and $L_{1,0,E}$. Moreover, the copy travelling along the first route is in its turn replicated once it reaches switch $S_{2,3}$ and sent twice on link $L_{2,1,S}$, as shown by the double arrow in the figure.

In general, the mapping of the communication between two tasks $\tau_i \rightarrow \tau_j$ can be formalised as a set of tuples $\mathrm{CS}_{i,j} = \{(L, n) : L \text{ is a link}, n \in \mathbb{N}\}$, where n indicates the number of copies of the same message that are conveyed by the corresponding link L. We will call the set $\mathrm{CS}_{i,j}$ the *communication support* (CS) of $\tau_i \rightarrow \tau_j$.

Let $\mathcal{M} \subset T \times T$ be the set of all pairs of communicating tasks that are mapped on different cores ($(\tau_a, \tau_b) \in \mathcal{M}$ if $\exists \Gamma_i = (V_i, E_i \subset V_i \times V_i)$ such that $(\tau_a, \tau_b) \in E_i$ and $\mathrm{Map}(\tau_i) \neq \mathrm{Map}(\tau_j)$.) A *communication mapping*, denoted $\mathcal{CM}$, is a function defined on $\mathcal{M}$ that maps each pair of communicating tasks to one communication support.

In our example, the communication mapping is the following: $\tau_1 \rightarrow \tau_2$ is mapped on $\mathrm{CS}_{1,2} = \{(L_{0,1,E}, 1)\}$, $\tau_1 \rightarrow \tau_5$ on $\mathrm{CS}_{1,5} = \{(L_{0,1,S}, 1)\}$, $\tau_7 \rightarrow \tau_8$ on $\mathrm{CS}_{7,8} = \{(L_{2,1,S}, 1)\}$, $\tau_9 \rightarrow \tau_{10}$ on $\mathrm{CS}_{9,10} = \{(L_{0,0,E}, 1)\}$, $\tau_{10} \rightarrow \tau_{11}$ on $\mathrm{CS}_{10,11} = \{(L_{1,0,E}, 1), (L_{2,0,E}, 1)\}$, $\tau_5 \rightarrow \tau_6$ on $\mathrm{CS}_{5,6} = \{(L_{0,0,E}, 1), (L_{1,0,E}, 1)\}$, $\tau_3 \rightarrow \tau_4$ on $\mathrm{CS}_{3,4} = \{(L_{2,0,E}, 2)\}$, and $\tau_2 \rightarrow \tau_3$ on $\mathrm{CS}_{2,3} = \{(L_{1,1,E}, 1), (L_{2,1,S}, 2), (L_{1,1,S}, 1), (L_{1,0,E}, 1)\}$.

Two properties of a communication support are of interest:

- The arrival probability of a message that is mapped on that communication support, called message arrival probability of the CS, denoted MAP, and
- The expected energy consumed by the transmission of the message on that support, called expected communication energy of the CS, denoted ECE.

The values of MAP and ECE can be computed by means of simple probability theory. We will illustrate their computation using the CS supporting message $\tau_2 \rightarrow \tau_3$, $\mathrm{CS}_{2,3}$. For simplicity, in this example we assume that the message consists of a single packet and the energy consumed by the transmission of the packet on any link is 1.

The MAP of $\mathrm{CS}_{2,3}$ is given by $\mathsf{P}(V \cup W)$, where V is the event that the copy sent along the path $L_{1,1,S} \rightarrow L_{1,0,E}$ successfully reaches core $P_{2,0}$, and W is the event that the copy sent along the path $L_{1,1,E} \rightarrow L_{2,1,S}$ successfully reaches core $P_{2,0}$.

$$\mathsf{P}(V) = \alpha^2.$$

The probability that both temporally redundant copies that are sent on link $L_{2,1,S}$ get scrambled is $(1-\alpha)^2$. Thus, the probability that the packet successfully reaches core $P_{2,0}$ if sent on path $L_{1,1,E} \rightarrow L_{2,1,S}$ is

$$\mathsf{P}(W) = \alpha \cdot (1 - (1-\alpha)^2).$$

Thus, the MAP of $\mathrm{CS}_{2,3}$ is

$$\begin{aligned}\mathsf{P}(V \cup W) &= \mathsf{P}(V) + \mathsf{P}(W) - \mathsf{P}(V \cap W) \\ &= \alpha^2 + \alpha \cdot (1 - (1-\alpha)^2) - \alpha^3 \cdot (1 - (1-\alpha)^2).\end{aligned}$$

The expected communication energy is the expected number of sent bits multiplied by the average energy per bit. The energy per bit, denoted E_{bit}, can be computed as shown by Ye et al.[17]

The ECE of $\mathrm{CS}_{2,3}$ is proportional to

$$E[\mathrm{Sent}_{S_{1,1}}] + E[\mathrm{Sent}_{S_{2,1}}] + E[\mathrm{Sent}_{S_{1,0}}],$$

where Sent_S denotes the number of packets sent from switch S and $E[\mathrm{Sent}_S]$ denotes its expected value. We assume that no packet is lost en-route between the source core and the switch associated to it. In our example, $E[\mathrm{Sent}_{1,1}] = 2$, i.e. two packets are sent by switch $S_{1,1}$, one along link $L_{1,1,S}$ and the other along link $L_{1,1,E}$. Hence, the expected number of packets sent by the switch associated to the source core is deterministically known and given by the communication support. Obviously, a switch that does not receive any valid copy of a packet, does not forward any. Hence, the expected number of packets forwarded by a switch is

$$E[\mathrm{Sent}_S] = E[\mathrm{Sent}_S | R_S] \cdot \mathsf{P}(R_S),$$

where R_S is the event that at least one copy of the packet successfully reaches switch S, and $E[\mathrm{Sent}_S|R_S]$ is the expected number of copies of the packet that are forwarded from switch S given that at least one copy successfully reaches switch S. In our example, $E[\mathrm{Sent}_{2,1}|R_{2,1}] = 2$ and $E[\mathrm{Sent}_{1,0}|R_{1,0}] = 1$. The probability that a packet successfully reaches switch $S_{2,1}$, $\mathsf{P}(R_{2,1})$, is equal to α, the probability that the packet traverses link $L_{1,1,E}$ unscrambled. Analogously, $\mathsf{P}(R_{1,0}) = \alpha$. Hence,

$$\mathrm{ECE}_{2,3} \sim 2 + 2 \cdot \alpha + 1 \cdot \alpha = 2 + 3\alpha.$$

The proportionality constant is $E_{\text{bit}} \cdot b$, where E_{bit} is the energy per bit and b is the number of bits of the packet.

4. PROBLEM FORMULATION

This section gives the formulation of the problem that we solve in this article.

4.1. Input

The input to the problem consists of:

- The hardware model, i.e. the size of the NoC, and, for each link, the energy-per-bit, the bandwidth, and the probability of a packet to be successfully conveyed by the link; and
- The application model, i.e. the set of task graphs Γ, the set of task and task graph deadlines Δ_T and Δ_Γ, respectively, the mapping of tasks to cores, the set of task periods Π, the best-case and worst-case execution times of all tasks on the cores on which they are mapped, the task priorities and the amounts of data to be transmitted between communicating tasks;
- The communication model, i.e. the packet size and message priority for each message (alternatively, our approach can automatically assign message priorities according to the message criticality);
- The lower bounds $B_{i,j}$ imposed on the message arrival probability $\text{MAP}_{i,j}$, which is the expected fraction of successfully transmitted messages, for each pair of communicating tasks $\tau_i \rightarrow \tau_j$.

4.2. Output

The output of the problem consists of the communication mapping $\mathcal{CM}$, such that the total communication energy is minimised.

4.3. Constraints

The application has to satisfy the following constraints:

- For each pair of communicating tasks $\tau_i \rightarrow \tau_j$, such that tasks τ_i and τ_j are mapped on different cores, the message arrival probability $\text{MAP}_{i,j}$ is greater than or equal to the imposed lower bound $B_{i,j}$ ($\text{MAP}_{i,j} \geq B_{i,j}$, $\forall \tau_i, \tau_j \in T : \tau_i \in {}^\circ\tau_j \wedge \text{Map}(\tau_i) \neq \text{Map}(\tau_j)$).
- All deadlines are met.

(1) **for each** pair of communicating tasks $\tau_i \rightarrow \tau_j$
(2) *find* as et of candidate CSs that satisfies $MAP_{i,j} \geq B_{i,j}$ (Section6)
(3) **end for**
(4) $(sol.\, min\_cost) = explore$ the space of candidate CSs (Section 8)
(5) using *response time calculation* (Section 7) for driving the exploration
(6) **if** $min\_cost = \infty$ **then**
(7) **return** "no solution"
(8) **else**
(9) sol' =voltage_freq_selection(sol) (according to [18])
(10) **return** sol'
(11) **end if**

Fig. 3. Approach outline.

5. APPROACH OUTLINE

The outline of our approach to solve the problem is shown in Fig. 3. First, for each pair of communicating tasks (message), we find a set of candidate communication supports (line 2, see Section 6), such that the lower bound constraint on the message arrival probability is satisfied. Second, the space of candidate communication supports is explored in order to find *sol*, the selection of communication supports that result in the minimum cost *min_cost* (line 4).[1] The worst-case response time of each explored solution is determined by the response time calculation function that drives the design space exploration (line 5, see Section 7). If no solutions are found that satisfy the response time constraints (min_cost = ∞), the application is deemed impossible to implement with the given resources (line 7). Otherwise, the solution with the minimum cost among the found solutions is selected. Voltage selection is performed on the selected solution in order to decrease the overall system energy consumption (line 9), and the modified solution is returned (line 10).

The next section discusses the construction of the set of candidate communication supports for an arbitrary pair of communicating tasks. Section 7 describes how the response time calculation is performed, while Section 8 outlines how the preferred communication supports representing the final solution are selected.

[1]See Section 8 for a precise definition of the cost of a solution. Intuitively, a low cost corresponds to a solution characterised by large time slack (long intervals between the finishing time of a task and its deadline).

6. COMMUNICATION SUPPORT CANDIDATES

This section describes how to construct a set of candidate communication supports for a pair of communicating tasks. First, we introduce the notions of path, coverage, and spatial, temporal, and general redundancy degree of a CS.

A path of length n connecting the switch corresponding to a source core to a switch corresponding to a destination core is an ordered sequence of n links, such that the end point of the ith link in the sequence coincides with the start point of the $i + 1$th link, $\forall 1 \leq i < n$, and the start point of the first link is the source switch and the end point of the last link is the destination switch. We consider only loop-free paths. A path *belongs* to a CS if all its links belong to the CS. A link of a CS is *covered* by a path if it belongs to the path.

The spatial redundancy degree (SRD) of a CS is given by the minimum number of distinct paths belonging to the CS that cover all the links of the CS. For example, the CSs depicted in Figs. 4(a), (b) both have a SRD of 1, as they contain only one path, namely path ($L_{0,0,N}$, $L_{0,1,E}$, $L_{1,1,E}$, $L_{2,1,N}$, $L_{2,2,N}$, $L_{2,3,E}$). The CS shown in Fig. 4(c) has spatial redundancy degree 2, as at least two paths are necessary in order to cover links $L_{1,1,N}$ and $L_{1,1,E}$, for example paths ($L_{0,0,N}$, $L_{0,1,E}$, $L_{1,1,E}$, $L_{2,1,N}$, $L_{2,2,N}$, $L_{2,3,E}$) and ($L_{0,0,N}$, $L_{0,1,E}$, $L_{1,1,N}$, $L_{1,2,E}$, $L_{2,2,N}$, $L_{2,3,E}$).

The temporal redundancy degree (TRD) of a link is given by the number of redundant copies to be sent on the link. The TRD of a CS is given by the maximum TRD of its links. For example, the TRD of the CS shown in Fig. 4(b) is 2 as two redundant copies are sent on links $L_{1,1,E}$, $L_{2,1,N}$, $L_{2,2,N}$, and $L_{2,3,E}$. The TRD of the CSs shown in Fig. 4(a), (c) is 1.

The general redundancy degree (GRD) of a CS is given by the sum of temporal redundancy degrees of all its links. For example, the GRD of the

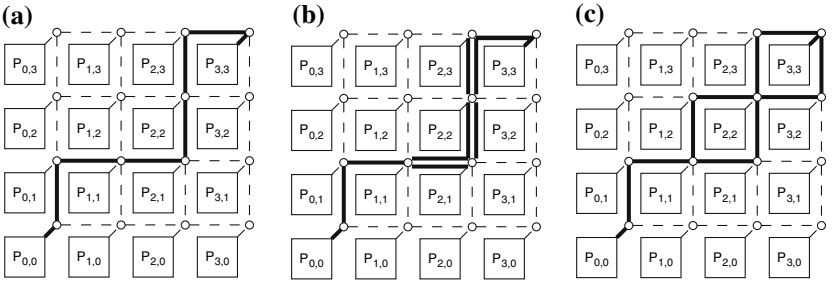

Fig. 4. Communication supports.

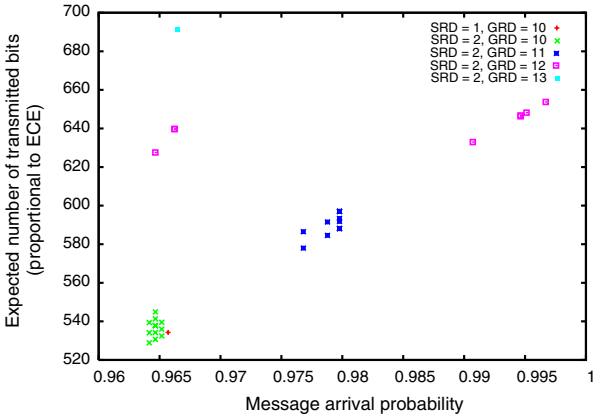

Fig. 5. Energy-efficiency of CSs of SRD 1 and 2.

CS shown in Fig. 4(a) is 6, the GRD of the CSs shown in Fig. 4(b), (c) is 10.

It is important to use CSs of minimal GRD because the expected communication energy (ECE) of a message is strongly dependent on the GRD of the CS supporting it. To illustrate this, we constructed all CSs of SRD 2 and GRD 10–13 for a message sent from the lower-left core to the upper-right core of a 4×4 NoC. We also constructed all CSs of SRD 1 and GRD 10. For each of the constructed CS, we computed their MAP and ECE. In Fig. 5, we plotted all resulting (MAP, ECE) pairs. Note that several different CSs may have the same MAP and ECE and therefore one dot in the figure may correspond to many CSs. We observe that the ECE of CSs of the same GRD do not differ significantly among them, while the ECE difference may account to more than 10% for CSs of different GRD.

The algorithm for the candidate set construction proceeds as shown in Fig. 6. Candidate CSs with SRD of only 1 and 2 are used. The justification for this choice is given later in the section.

We illustrate how to find the minimal GRD for a message based on the example depicted in Fig. 4. We consider a 4×4 NoC, and a message sent from core $P_{0,0}$ to core $P_{3,3}$. The message consists of just one packet, the probability that the packet successfully traverses any of the links is $\alpha = 0.99$, and the imposed lower bound on the MAP is $B = 0.975$.

We look first at CSs with SRD of 1, i.e. consisting of a single path. We consider only shortest paths, that is of length 6. Obviously, a lower bound on GRD is 6. If we assign just one copy per link, the message arrival probability would be $\alpha^6 \approx 0.941 < 0.975 = B$. We try with a

(1) **for each** pair of communicating tasks $\tau_i \rightarrow \tau_j$
(2) Determine N_1 and N_2, the minimum general redundancy degrees of CSs of SRD 1 and 2 respectively, such that the MAP constraint on $\tau_i \rightarrow \tau_j$ is satisfied
(3) Add all CSs with SRD 1 and with GRD N_1 and all CSs with SRD 2 and with GRD N_2 to the set of CS candidates of $\tau_i \rightarrow \tau_j$
(4) **end for**

Fig. 6. Construction of candidate CS set.

GRD of 7, and regardless to which of the six links we assign the redundant copy, we get a MAP of $\alpha^5 \cdot (1-(1-\alpha)^2) \approx 0.95 < 0.975 = B$. Hence, we are forced to increase the GRD once more. We observe that there are five links left with a TRD of 1. The probability to traverse them is $\alpha^5 \approx 0.95$, less than the required lower bound. Therefore it is useless to assign one more redundant copy to the link that now has a TRD of 2 because anyway the resulting MAP would not exceed α^5. Thus, the new redundant copy has to be assigned to a different link of the CS of GRD 8. In this case, we get a MAP of $\alpha^4 \cdot (1-(1-\alpha)^2)^2 \approx 0.96$, still less than the required bound. We continue the procedure of increasing the GRD and distributing the redundant copies to different links until we satisfy the MAP constraint. In our example, this happens after adding four redundant copies (MAP$= \alpha^2 \cdot (1-(1-\alpha^2))^4 \approx 0.9797$). The resulting CS of SRD 1 and GRD 10 is shown in Fig. 4(b), where the double lines represent links that convey two copies of the same packet. Thus, the minimal GRD for CSs of SRD 1 is $N_1 = 10$. There are 20 distinct paths between core $P_{0,0}$ and core $P_{3,3}$ and there are 15 ways of distributing the four redundant copies to each path. Thus, $15 \cdot 20 = 300$ distinct candidate CSs of SRD 1 and GRD 10 can be constructed for the considered message. They all have the same message arrival probability, but different expected communication energies. The ECEs among them vary 1.61%.

Similarly, we obtain N_2, the minimal GRD for CSs of SRD 2. In this case, it can be mathematically shown that larger message arrival probabilities can be obtained with the same GRD if the two paths of the CS intersect as often as possible and the distances between the intersection points are as short as possible. [14] Intuitively, intersection points are important because even if a copy is lost on one incoming path, the arrival of another copy will trigger a regeneration of two packets in the switch where the two paths intersect. The closer to each other the intersection points are, the shorter the packet transmission time between the two points is. Thus, the probability to lose a message between the two intersection points is lower. Therefore, in order to obtain N_2, we will consider

CSs with many intersection points that are close to each other. For our example, the lowest GRD that lets the CS satisfy the MAP constraint is $N_2 = 10$ ($\text{MAP} = \alpha^6 \cdot (2 - \alpha^2)^2 \approx 0.9793$). This CS is shown in Fig. 4(c). The minimum number of needed redundant copies in order to satisfy the MAP constraint is strongly dependent on α and the imposed lower bound on the MAP, and only weakly dependent on the geometric configuration of the CS. Therefore, typically $N_2 = N_1$ or it is very close to N_1.

In conclusion, N_1 and N_2 are obtained by progressively increasing the GRD until the CS satisfies the MAP constraint. The redundant copies must be uniformly distributed over the links of the CS. Additionally, in the case of CSs with SRD 2, when increasing the GRD, links should be added to the CS such that many path intersection points are obtained and that they are close to each other.

The following reasoning lies behind the decision to use CSs with SRD of only 1 and 2. First, we give the motivation for using CSs with SRD larger than 1. While, given a GRD of N, it is possible to obtain the maximum achievable message arrival probability with CSs of SRD 1, concurrent transmission of redundant message copies would be impossible if we used CSs with SRD of only 1. This could severely affect the message latency or, even worse, lead to link overload. CSs with SRD 2 are only marginally more energy hungry, as can be seen from the cluster of points in the lower-left corner of Fig. 5. Usually, the same MAP can be obtained by a CS of SRD 2 with only 1–2% more energy than a CS of SRD 1.

While the previous consideration supports the use of CSs with SRD greater than 1, there is no reason to go with the SRD beyond 2. Because of the two-dimensional structure of the NoC, there are at most 2 different links that belong to the shortest paths between the source and the destination and whose start points coincide with the source core. Thus, if a CS consisted only of the shortest paths, the message transmission would be vulnerable to a double fault of the two initial links. Therefore, CSs with SRD greater than 2, while consuming more energy for communication, would still be vulnerable to a double fault on the initial links and hence can only marginally improve the MAP. If we did not restrict the CS to the shortest paths, while overcoming the limitation on the MAP, we would consume extra energy because of the longer paths. At the same time, latency would be negatively affected. Thus, for two-dimensional NoC, we consider CSs of SRD of only 1 and 2.

7. RESPONSE TIME CALCULATION

In order to guarantee that tasks meet their deadlines, in case no message is lost, response times have to be determined in the worst case.

Let us consider the example depicted in Fig. 7(a). Solid lines depict data dependencies among the tasks, while the dotted lines show the actual communication mapping to the on-chip links. The two CSs are $CS_{1,2} = \{(L_{0,0,E}, 1)\}$ and $CS_{1,3} = \{(L_{0,0,E}, 1), (L_{1,0,N}, 1), (L_{0,0,N}, 2), (L_{0,1,E}, 2)\}$. Packet sizes are such that message $\tau_1 \rightarrow \tau_2$ is chopped into 2 packets, while message $\tau_1 \rightarrow \tau_3$ fits into a single packet.

Based on the application graph, its mapping and the communication supports, we construct a task graph as shown in Fig. 7(b). Each link L is regarded as a processor P_L, and each packet transmission on link L is regarded as a non-preemptive task executed on processor P_L. The shadings of the circles denote the processors (links) on which the tasks (packets) are mapped. Tasks τ_4 and τ_5 represent the first and the second packet of the message $\tau_1 \rightarrow \tau_2$. They are both dependent on task τ_1, as the two packets are generated when task τ_1 completes its execution, while task τ_2 is dependent on both task τ_4 and τ_5 as it can start only after it has received the entire message, i.e. both packets, from task τ_1. Both tasks τ_4 and τ_5 are mapped on the "processor" corresponding to the link $L_{0,0,E}$. Task τ_6 represents the packet of the message $\tau_1 \rightarrow \tau_3$ that is sent on link $L_{0,0,E}$ and task τ_7 represents the same packet once it reaches link $L_{1,0,N}$. Tasks τ_8 and τ_9 are the two copies of the packet of the message $\tau_1 \rightarrow \tau_3$ that are sent on link $L_{0,0,N}$.

We are interested in the worst-case scenario w.r.t. response times. In the worst case, all copies of packets get scrambled except the latest packet. Therefore, the copies to be sent by a core on its outgoing links have to wait until the last of the copies arriving on incoming links of the core has reached the core. For example, tasks τ_{10} and τ_{11}, modelling the two copies of the message $\tau_1 \rightarrow \tau_3$ that are sent on the link $L_{0,1,E}$, depend on both τ_8 and τ_9, the two copies on link $L_{0,0,N}$. Also, task τ_3 depends

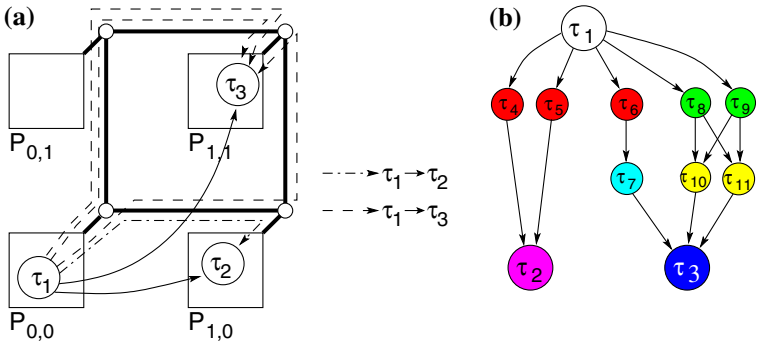

Fig. 7. Application modelling for response time analysis.

on all three copies, τ_7, arriving on link $L_{1,0,N}$, and τ_{10} and τ_{11}, arriving on link $L_{0,1,E}$.

The modified model, as shown in Fig. 7(b), is analysed using the dynamic offset based schedulability analysis proposed by Palencia and Harbour.[19] The analysis calculates the worst-case response times and jitters for all tasks.

8. SELECTION OF COMMUNICATION SUPPORTS

As shown in Section 6 (see also line 2 in Fig. 3), we have determined the most promising (low energy, low number of messages) set of CSs for each transmitted message in the application. All those CSs guarantee the requested MAP. As the next step of our approach (line 4 in Fig. 3) we have to select one particular CS for each message, such that the solution cost is minimised, which corresponds to maximising the smallest time slack. The response time for each candidate solution is calculated as outlined in Section 7 (line 5 in Fig. 3).

The design space is explored with a Tabu Search based heuristic. Tabu Search is a heuristic introduced by Glover[19]. It has been successfully used for various system level design problems, such as yield maximisation problems,[20] FIR filter design,[21] mapping and scheduling of task graphs on SoCs,[22] and latency-area trade-offs for NoCs.[23] Several researchers have compared Tabu Search with other optimisation heuristics, such as Simulated Annealing and Genetic Algorithms and shown the superiority of Tabu Search with regard to optimisation time and quality of results [22,24–27].

We use an extended variant of Tabu Search, which is described in this section. The variant is not specific to a particular problem. After explaining the heuristic in general, we will become more specific at the end of the section where we illustrate the heuristic in the context of communication mapping.

We define the design space $\mathcal{S}$ as a set of points (also called solutions), where each point represents a configuration that satisfies the imposed constraints. A *move* from one solution in the design space to another solution is equivalent to assigning a new value to one or more of the parameters that characterise the system. We say that we obtain solution s_2 by applying the move m on solution s_1, and we write $s_2 = m(s_1)$. Solution s_1 can be obtained back from solution s_2 by applying the *negated move* m, denoted $\overline{m}$ ($s_1 = \overline{m}(s_2)$).

Solution s' is a *neighbour* of solution s if there exists a move m such that solution s' can be obtained from solution s by applying move m. All

```
(1)   crt_sol = init_sol
(2)   global_best_sol = crt_sol
(3)   global_best = cost(crt_sol)
(4)   TM = ∅
(5)   since_last_improvement = 0
(6)   iteration_count = 1
(7)   CM = set_of_candidate_moves(crt_sol)
(8)   (chosen_move, next_sol_cost) = choose_move(CM)
(9)   while iteration_count < max_iterations do
(10)    while since_last_improvement < W do
(11)      next_sol = move(crt_sol, chosen_move)
(12)      TM = TM ∪ {chosen_move}
(13)      since_last_improvement + +
(14)      iteration_count + +
(15)      crt_sol = next_sol
(16)      if next_sol_cost < global_best_cost then
(17)        global_best_cost = next_sol_cost
(18)        global_best_sol = crt_sol
(19)        since_last_improvement = 0
(20)      end if
(21)      CM = set_of_candidate_moves(TM, crt_sol)
(22)      (chosen_move, next_sol_cost) = choose_move(CM)
(23)    end while
(24)    since_last_improvement = 0
(25)    (chosen_move, next_sol_cost) = diversify(TM, crt_sol)
(26)    iteration_count + +
(27)  end while
(28)  return global_best_sol
```

Fig. 8. Design space exploration algorithm.

neighbours of a solution s form the neighbourhood $V(s)$ of that solution ($V(s) = \{q : \exists m$ such that $q = m(s)\}$).

The exploration algorithm is shown in Fig. 8. The exploration starts from an initial solution, labelled also as the current solution (line 1) considered as the globally best solution so far (line 2).

The cost function is evaluated for the current solution (line 3). We keep track of a list of moves TM that are marked as *tabu*. A solution will leave the tabu list after a certain number of iterations (the tabu tenure). Initially the list is empty (line 4).

We construct CM, a subset of the set of all moves that are possible from the current solution point (line 7). Let N(CM) be the set of solutions that can be reached from the current solution by means of a move in CM.[2] The cost function is evaluated for each solution in N(CM).

[2] If CM is the set of *all* possible moves from crt_sol then N(CM) = V(crt_sol).

Let $m \leq m'$ if $\text{cost}(m(\text{crt\_sol})) \leq \text{cost}(m'(\text{crt\_sol}))$. A move in CM is selected (line 8) as follows.

If $\exists m \in \text{CM}$ such that $m \leq m' \forall m' \in \text{CM} \wedge \text{cost}(m(\text{crt\_sol})) \leq$ global_best, then move m is selected. Else, if $\exists m \in \text{CM} \setminus \text{TM}$ such that $m \leq m' \forall m' \in \text{CM} \setminus \text{TM}$, then move m is selected. Else, $m \in \text{TM}$ such that $m \leq m' \forall m' \in \text{TM}$ is selected.

The new solution is obtained by applying the chosen move m on the current solution (line 11). The reverse of move $\overline{m}$ is marked as tabu such that m will not be reversed in the next few iterations (line 12). The new solution becomes the current solution (line 15). If it is the case (line 16), the new solution becomes also the globally best solution reached so far (lines 17–18). However, it should be noted that the new solution could have a larger cost than the current solution. This could happen if there are no moves that would improve on the current solution or all such moves would be tabu. By this, the heuristic has the potential to escape from local minima. Placing the reverse of the most recent moves into the list TM of tabu moves also avoids cycling that could occur if the move returns to a recently visited solution. The procedure of building the set of candidate moves and then choosing one according to the criteria listed above is repeated. If no global improvement has been noted for the past W iterations, the loop (lines 10–23) is interrupted (line 10). In this case, a diversification phase follows (line 25) in which a rarely used move is performed in order to force the heuristic to explore different regions in the design space. The whole procedure is repeated until the heuristic iterated for a specified maximum number of iterations (line 9). The procedure returns the solution characterised by the lowest cost function value that it found during the design space exploration (line 28).

Two issues are important when tailoring the general tabu search based heuristic described above for particular problems.

First, there is the definition of what is a legal move. On one hand, the transformation of a solution must result in another solution, i.e. the resulting parameter assignment must satisfy the set of constraints. On the other hand, because of complexity reasons, certain restrictions must be imposed on what constitutes a legal move. For example, if any transformation were a legal move, the neighbourhood of a solution would comprise the entire solution space. In this case, it is sufficient to run the heuristic for just one iteration (max_iterations = 1) but that iteration would require an unreasonably long time, as the whole solution space would be probed. Nevertheless, if moves were too restricted, a solution could be reached from another solution only after applying a long sequence of moves. This makes the reaching of the far-away solution unlikely. In this case, the heuristic

would be inefficient as it would circle in the same region of the solution space until a diversification step would force it out.

The second issue is the construction of the subset of candidate moves. One solution would be to include all possible moves from the current solution in the set of candidate moves. In this case, the cost function, which sometimes can be computationally expensive, has to be calculated for all neighbours. Thus, we would run the risk to render the exploration slow. If we had the possibility to quickly assess which are promising moves, we could include only those in the subset of candidate moves.

For our problem, the design space is the Cartesian product of the sets of CS candidates for each message (constructed as shown in Section 6). Because all CS candidates guarantee the requested MAP, all points in the solution space satisfy the MAP constraint of the problem (Section 4.3). A point in the design space is an assignment of communication supports to messages (see Section 3.5). A move means picking one pair of communicating tasks and selecting a new communication support for the message sent between them.

In order to speed up the exploration, we pick only "promising" moves to be included in the set of candidate moves CM. Thus,

1. we look at messages with large jitters as they have a higher chance to improve their transmission latency by having assigned a new CS; and
2. for a certain message $\tau_i \rightarrow \tau_j$, we consider only those candidate CSs that would decrease the amount of interference of messages of higher priority than $\tau_i \rightarrow \tau_j$. (By this we remove messages from overloaded links.)

The value of the cost function that drives the response-time minimisation, evaluated for an assignment of communication supports to messages $\mathcal{CM}$, is:

$$\text{cost}(\mathcal{CM}) = \begin{cases} \infty, & \exists \tau \in T : \text{WCRT}_\tau > \delta_\tau \vee, \\ & \vee \exists \Gamma_i \in \Gamma : \text{WCRT}_{\Gamma_i} > \delta_{\Gamma_i}, \\ \max_{\tau \in T} \frac{\text{WCRT}_\tau}{\delta_\tau}, & \text{otherwise,} \end{cases} \quad (1)$$

where T is the set of tasks and WCRT and δ denote worst-case response times and deadlines, respectively. The worst-case response time of a task is obtained as shown in Section 7.

In the case of the cost function in Eq. (1), we make the conservative assumption that voltage and frequency are set system-wide for the whole NoC. This means that, at design time, an optimal voltage and/or frequency is determined for the whole NoC. The determined values for

voltage and frequency do not change during the entire execution time of the application. For such a scenario, if we decrease the system-wide voltage (or frequency), the worst-case response times of all tasks would scale with the same factor. Therefore, in the definition of the cost function (Eq. (1)) we use the *max* operator, since the width of the interval in which the response time is allowed to increase is limited by the smallest slack (largest $\frac{\text{WCRT}_{\tau_i}}{\delta_{\tau_i}}$) among the tasks.

In a second scenario, we can assume that voltage and/or frequency may be set core-wise. This means that, at design time, an optimal voltage and/or frequency is calculated for each core. These voltages and frequencies do not change during the whole execution time of the application. The cost function would become

$$\text{cost}(\mathcal{CM}) = \begin{cases} \infty, & \exists \tau \in T : \text{WCRT}_\tau > \delta \vee, \\ & \exists \Gamma_i \in \Gamma : \text{WCRT}_{\Gamma_i} > \delta_{\Gamma_i}, \\ \sum_{p \in P} \max_{\tau \in \mathcal{T}_p} \frac{\text{WCRT}_\tau}{\delta_\tau}, & \text{otherwise}, \end{cases} \tag{2}$$

where p is a core in P, the set of cores of the NoC, and $\mathcal{T}_p$ is the set of tasks mapped on core p.

In a third scenario, we assume that voltage and/or frequency may change for each core during operation. Then they may be set task-wise and the cost function becomes

$$\text{cost}(\mathcal{CM}) = \begin{cases} \infty, & \exists \tau \in T : \text{WCRT}_\tau > \delta \vee, \\ & \vee \exists \Gamma_i \in \Gamma : \text{WCRT}_{\Gamma_i} > \delta_{\Gamma_i}, \\ \sum_{\tau \in T} \frac{\text{WCRT}_\tau}{\delta_\tau}, & \text{otherwise}. \end{cases} \tag{3}$$

9. EXPERIMENTAL RESULTS

We report on three sets of experiments that we ran in order to assess the quality of our approach.

9.1. Latency as a Function of the Number of Tasks

The first set investigates the application latency as a function of the number of tasks. About 340 applications of 16–80 tasks were randomly generated. The applications are executed by a 4×4 NoC. The probability that a link successfully conveys a data packet is 0.97, and the imposed

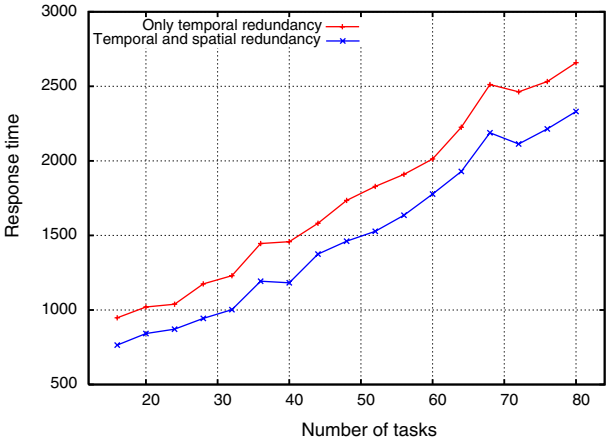

Fig. 9. Application latency versus number of tasks.

lower bound on the message arrival probability is 0.99. For each application, we ran our communication mapping tool twice. In the first run, we consider CSs of SRD 1, i.e. packets are retransmitted on the same, unique path. In the second run, we consider CSs of SRD 1 and 2, as described in Section 6. Figure 9 depicts the averaged results. The approach that uses both spatially and temporally redundant CSs leads to shorter application latencies than the approach that just re-sends on the same path.

9.2. Latency as a Function of the Imposed Message Arrival Probability

The second experiment investigates the dependency of latency on the imposed message arrival probability. About 20 applications, each of 40 tasks, were randomly generated. We considered the same hardware platform as in the first experiment. For each application, we considered 17 different lower bounds on MAP, ranging from 0.94 to 0.9966. The averaged results are shown in Fig. 10. For low bounds on MAP, such as 0.94, almost no transmission redundancy is required to satisfy the MAP constraint. Therefore, the approach combining spatially and temporally redundant communication fares only marginally better than the approach that uses only temporal redundancy. However, for higher bounds on the MAP, the approach that combines spatially and temporally redundant transmission has the edge. In the case of bounds on the MAP larger than 0.9992, spatial redundancy cannot satisfy the constraint anymore, and therefore the temporally redundant transmission becomes dominant

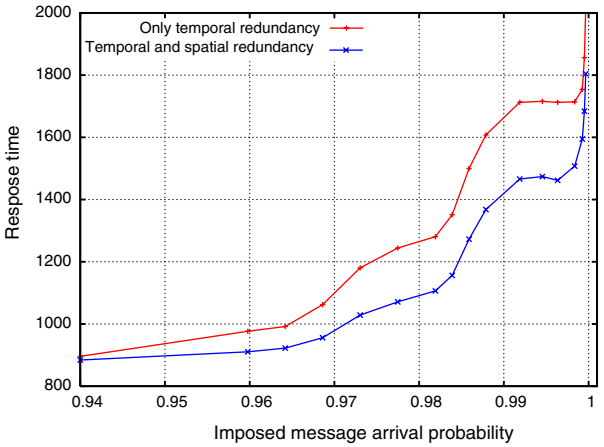

Fig. 10. Application latency versus bound on MAP.

and the approach combining spatial and temporal redundancy does not lead to significant latency reductions anymore.

9.3. Latency as a Function of the Size of the NoC and Communication Load

The third experiment has a double purpose. First, it investigates the dependency of latency reduction on the size of the NoC. Second, it investigates latency reduction as a function of the communication load (bits/time unit). About 20 applications of 40, 62 and 90 tasks were randomly generated. The applications with 40 tasks run on a 4×4 NoC, those with 62 tasks run on a 5×5 NoC and those with 90 tasks run on a 6×6 NoC. For each application, we considered communication loads of 1–4 bits/time unit. The averaged latency reductions when using the optimal combination of spatial and temporal redundancy, compared to purely temporal redundancy, are depicted in Fig. 11. We observe that for low-communication loads, the latency reduction is similar for all three architectures, around 22%. However, at loads higher than 3.4 the relatively small number of links of the 4×4 NoC get congested and response times grow unboundedly. This, however, is not the case with the larger NoCs. Latency reduction for a load of 4 is 22% for a NoC of 6×6 and 12% for 5×5.

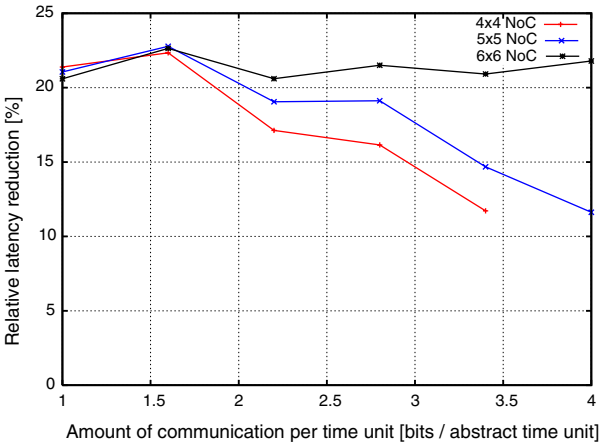

Fig. 11. Application latency versus NoC size and communication load.

9.4. Optimisation Time

Figure 12 depicts the histogram of the optimisation time for all benchmarks that are used in this section, as measured on a desktop PC with an AMD Athlon processor clocked at 1533 MHz. On average, the optimisation time is 912 s. We note that the optimisation time for a large majority of benchmarks is smaller than 1000 s, while 5.6% of all benchmarks took between 1000 and 2000 s to optimise, and the optimisation of 8.7% of benchmarks took longer than 2000 s.

9.5. Exploiting the Time Slack for Energy Reduction

The presented experiments have shown that, by using an optimal combination of temporal and spatial redundancy for message mapping, significant reduction of latency can be obtained while guaranteeing message arrival probability at the same time. It is important to notice that the latency reduction is obtained without energy penalty, as shown in Section 6. This means that for a class of applications using the proposed approach it will be possible to meet the imposed deadlines, which otherwise would not be possible without changing the underlying NoC architecture. However, the proposed approach gives also the opportunity to further reduce the energy consumed by the application. If the obtained application response time is smaller than the imposed one, the resulting slack can be exploited by running the application at reduced voltage. In order to illustrate this, we have performed another set of experiments.

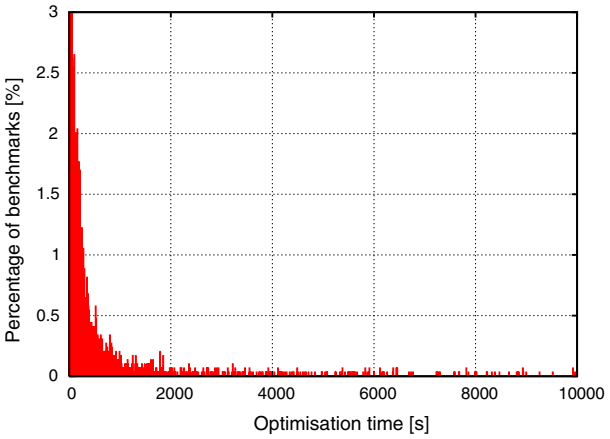

Fig. 12. Histogram of the optimisation time as measured on an AMD Athlon@1533 MHz desktop PC.

Applications of 16–60 tasks running on a 4×4 NoC were randomly generated. For each application we ran our message mapping approach twice, once using CSs with SRD of only 1, and second using CSs with SRD of 1 and 2. The slack that resulted in the second case was exploited for energy reduction. We have used the algorithm published in[28] for calculating the voltage levels for which to run the application. For our energy models, we considered a 70 nm CMOS fabrication process. The resulted energy consumption is depicted in Fig. 13. The energy reduction ranges from 20% to 13%. For this experiment, we considered the conservative scenario in which, at design time, an optimal voltage and/or frequency is computed for the whole NoC (see Eq. (1) in Section 8). We do not assume the availability of a dynamic voltage scaling capability in the NoC. If such capability existed, even larger energy savings could be achieved.

9.6. Real-life Example: An Audio/Video Encoder

Finally, we applied our approach to a multimedia application, namely an audio/video encoder implementing the H.263 recommendation[29] of the International Telecommunication Union (ITU) for video encoding and the MPEG-1 Audio Layer 3 standard for audio encoding (ISO/IEC 11172-3 Layer 3[30]).

Figure 14 depicts the task graph that models the application, while Fig. 15 shows the application mapping to the NoC cores. The task partitioning, mapping, and profiling was done by Hu and Mărculescu.[31]

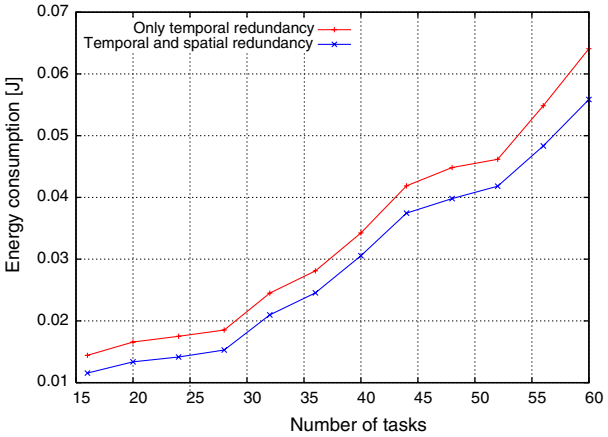

Fig. 13. Energy consumption versus number of tasks.

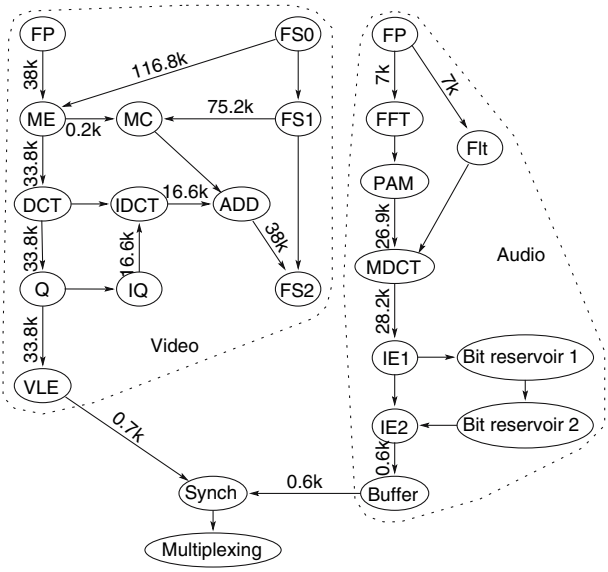

Fig. 14. H.263 and MP3 encoding application.

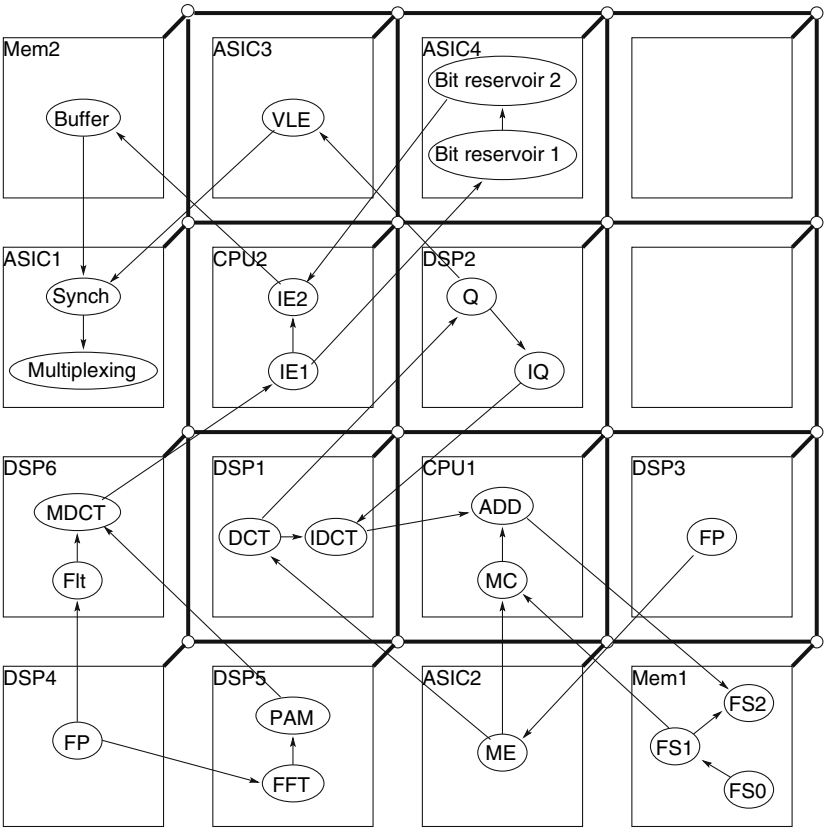

Fig. 15. NoC implementation of the H.263 and MP3 encoding application.

The video encoding part of the application consists of nine tasks: frame prediction (FP), motion estimation (ME), discrete cosine transform (DCT), quantisation (Q), inverse quantisation (IQ), inverse discrete cosine transform (IDCT), motion compensation (MC), addition (ADD), and variable length encoding (VLE). Three memory regions are used for frame stores FS0, FS1, and FS2. The audio encoding part consists of seven tasks: frame prediction (FP), fast Fourier transform (FFT), psycho-acoustic model (PAM), filter (Flt), modified discrete cosine transform (MDCT), and two iterative encoding tasks (IE1 and IE2). The numbers that annotate arcs in Fig. 14 denote the communication amount of the message represented by the corresponding arc. The period of the task graph depends on the imposed frame rate, which depends on the video clip. We use

periods of 41.6 ms, corresponding to 24 frames per second. The deadlines are equal to the periods.

The application is executed by an NoC with 6 DSPs, 2 CPUs, 4 ASICs, and 2 memory cores, organised as a 4×4 NoC with two unused tiles, as shown in Fig. 15. The probability that a packet successfully traverses a network link is assumed to be 0.99. The approach combining spatially and temporally redundant message transmission obtained a 25% response time reduction relative to the approach deploying only temporal redundancy. The energy savings after voltage reduction amounted to 20%. Because of the relatively small design space of this example, the optimisation took only 3 s when combining spatially and temporally redundant communication supports.

10. CONCLUSIONS

In this article, we addressed the problem of communication energy minimisation under task response time and message arrival probability constraints. The total communication energy is reduced by means of two strategies. On one hand, we intelligently select the communication supports of messages such that we reduce application response time with negligible energy penalty while satisfying message arrival probability constraints. On the other hand, the execution time slack can be exploited by deploying voltage and/or frequency scaling on the cores and communication links. The approach is efficient as it results in energy reductions up to 20%.

REFERENCES

1. K. Diefenderhoff, Extreme Lithography, *Microprocessor Rep.*, **6**(19): 1–10 (2000).
2. K. Shepard and V. Narayanan, Noise in Deep Submicron Digital Design, in *Proc. of the ICCAD*, pp. 524–531 (1996).
3. K. Aingaran, F. Klass, C. M. Kim, C. Amir, J. Mitra, E. You, J. Mohd, and S. K. Dong, Coupling Noise Analysis for VLSI and ULSI Circuits, in *Proc. of IEEE ISQED*, pp. 485–489 (2000).
4. L. Benini and G. De Micheli, Networks on Chips: A New SoC Paradigm, *IEEE Comput.*, **35**(1):70–78 (2002).
5. K. Goossens, J. Dielissen, and A. Radulescu, AEthereal Network on Chip: Concepts, Architectures and Implementations, *IEEE. Des. Test. Comput.*, **22**(5):414–421 (2005).
6. S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, and A. Hemani, A Network on Chip Architecture and Design Methodology, in *Proc. of the IEEE Computer Society Annual Symposium on VLSI*, pp. 105–112 (2002).
7. W. Dally, Interconnect-limited VLSI Architecture, in *Proc. of the IEEE Conference on Interconnect Technologies*, pp. 15–17 (1999).

8. D. Liu *et al.*, Power Consumption Estimation in CMOS VLSI chips, *IEEE J. Solid-State Circ.*, (29):663–670 (1994).
9. J. Hu and R. Mărculescu, Energy and Performance-aware Mapping for Regular NoC Architectures, *IEEE Trans. CAD Integr. Circ. Syst.*, **24**(4):551–562 (2005).
10. S. Murali and G. De Micheli, Bandwidth-constrained Mapping of Cores onto NoC architectures, in *Proc. of the Conference on Design Automation and Test in Europe*, pp. 896–901 (2004).
11. E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, QNoC: QoS architecture and Design Process for Networks-on-chip, *J. Syst. Arch.*, **50**:105–128 (2004)
12. P. Kermani and L. Kleinrock, Virtual Cut-Through: A New Computer Communication Switching Technique, *Comput. Netw.*, **3**(4):267–286 (1979).
13. T. Dumitraş and R. Mărculescu, On-chip Stochastic Communication, in *Proc. of DATE*, pp. 790–795 (2003).
14. S. Manolache, *Fault-tolerant Communication on Network-on-chip*, Technical report, Linköping University (2004).
15. M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, Fault Tolerant Algorithms for Network-on-chip Interconnect, in *Proc. of the ISVLSI*, pp. 46–51 (2004).
16. D. Bertozzi, L. Benini, and G. De Micheli, Low Power Error Resilient Encoding for On-chip Data Buses, in *Proc. of DATE*, pp. 102–109 (2002).
17. T. T. Ye, L. Benini, and G. De Micheli, Analysis of Power Consumption on Switch Fabrics in Network routers, in *Proc. of DAC*, pp. 524–529 (2002).
18. J. C. Palencia Gutiérrez and M. González Harbour, Schedulability Analysis for Tasks with Static and Dynamic Offsets, in *Proc. of the 19th IEEE Real Time Systems Symposium*, pp. 26–37 (December 1998).
19. F. Glover, Tabu Search—Part I., *ORSA J. Comput.*, **1**(3):190–206 (1989).
20. C. Alippi, A. Galbusera, and M. Stellini, An Application-level Synthesis Methodology for Multidimensional Embedded Processing Systems, *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst.*, **22**(11):1457–1470 (2003).
21. S. Traferro, F. Capparelli, F. Piazza, and A. Uncini, Efficient Allocation of Power of Two Terms in FIR Digital Filter Design using Tabu Search, *IEEE Trans. Comput. Aided Des. Integr. Circ. and Syst.*, **3**:411–414 (1999).
22. T. Wild, J. Foag, N. Pazos, and W. Brunnbauer, Mapping and Scheduling for Architecture Exploration of Networking SoCs, in *Proc. of the 16th International Conference on VLSI Design,* pp. 376–381 (2003).
23. R. S. Cardoso, M. E. Kreutz, L. Carro, and A. A. Susin, Design Space Exploration on Heterogeneous Network-on-chip, in *Proc. of the IEEE International Symposium on Circuits and Systems*, Vol. 1, pp. 428–431 (2005).
24. O. Hajji, S. Brisset, and P. Brochet, Comparing Stochastic Optimization Methods used in Electrical Engineering, *IEEE Trans. Syst. Man Cybern*, **7**(6–9):6 (2002).
25. S. Pierre and F. Houeto, Assigning Cells to Switches in Cellular Mobile Networks Using Taboo Search, *IEEE Trans. Syst. Man Cybern,* **32**(3):351–356 (2002).
26. B. Krishnamachari and S. B. Wicker, Optimization of Fixed Network Design in Cellular Systems using Local Search Algorithms, in *Proc. of the 52nd Vehicular Technology Conference*, Vol. 4, pp. 1632–1638 (2000).
27. P. Eles, Z. Peng, K. Kuchcinsky, and A. Doboli, System-level Hardware/software Partitioning Based on Simulated Annealing and Tabu Search, *J. Des. Autom. Embed. Sys.,* **2**:5–32 (1997).

28. A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. Al-Hashimi, Simultaneous Communication and Processor Voltage Scaling for Dynamic and Leakage Energy Reduction in Time-constrained Systems, in *Proc. of ICCAD*, pp. 362–369 (2004).
29. International Telecommunication Union (ITU). *H.263 – Video coding for low bit rate communication*, 2005. http://www.itu.int/publications/itu-t/.
30. International Organization for Standardization (ISO). *ISO/IEC 11172-3:1993 – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio*, 1993. http://www.iso.org/.
31. J. Hu and R. Mărculescu, Energy-aware Communication and Task Scheduling for Network-on-chip Architectures Under Real-time Constraints, in *Proc. of the Design Automation and Test in Europe Conference*, p. 10234 (2004).