Fault and Energy-Aware Communication Mapping with Guaranteed Latency for Applications Implemented on NoC

Sorin Manolache[*]
Linköping University, Sweden
sorma@ida.liu.se

Petru Eles
Linköping University, Sweden
petel@ida.liu.se

Zebo Peng
Linköping University, Sweden
zebpe@ida.liu.se

ABSTRACT

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. Firstly, we reduce the total amount of transmitted messages, and, secondly, 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.

Categories and Subject Descriptors

B.4.4 [**Hardware Input/Output and Data Communications**]: Performance Analysis and Design Aids—*Worst-Case Analysis*; B.4.5 [**Hardware Input/Output and Data Communications**]: Reliability, Testing, and Fault-Tolerance

General Terms

Algorithms, Performance

Keywords

Network-on-Chip

1. INTRODUCTION

Shrinking feature sizes make possible the integration of millions and soon billions of transistors on multi-core chips. At this integration level, effects such as capacitive cross-talk, power supply noise, and neutron and alpha radiation [13] lead to non-negligible rates of transient failures of interconnects and/or devices, jeopardising the correctness of applications.

Several authors [2, 5] have proposed network-on-chip (NoC) architectures as replacements to bus-based designs in order to reduce design, verification and test complexity and to ease the power management problem. One of the main problems for such highly integrated complex systems is the increasing rate of failures of the communication lines. In this paper we concentrate our discussion on how to handle transient failures of on-chip network links in the context of time and energy constrained applications implemented on NoC. The reliability of network nodes is guaranteed by specific methods which are outside the scope of this paper.

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. With shrinking feature size, the on-chip interconnects have become a performance bottleneck [4]. Thus, the selection of message routes has a significant impact on the responsiveness of applications implemented on the NoC. The energy consumption of wires has been reported to account for about 40% of the total energy consumed by the chip [9]. This is a strong incentive to consider the communication energy in addition to guaranteeing required levels of message transmission reliability.

In this paper, we address all of the three stringent problems identified above: link reliability, latency, and energy consumption. We propose a solution for the following problem: Given an NoC architecture with a specific fault model 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.

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 result-

[*] The authors thank Prof. Radu Mărculescu for inviting Sorin Manolache to visit his research group at CMU. The ideas presented in this paper originated during that visit.

DAC 2005, Anaheim, CA

ing time slack is exploited for energy minimisation by means of voltage reduction on network nodes and links.

At on-chip bus level, Bertozzi et al. [3] address the problem of energy-efficient reliable on-chip communication. 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 paper we address the problem at application level, considering time-constrained multi-hop transmission of messages sharing the links of an NoC.

At system level, Dumitraş and Mărculescu [6] 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. 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 [10].

Pirretti et al. [12] 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.

Our approach differs 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 transmission times. Additionally, by carefully balancing temporal and spatial communication redundancy, we are able to minimise the application latency and, by this, also the consumed energy.

The rest of the paper is structured as follows. The next section presents the models of architecture, communication and application and gives the problem formulation. Section 3 outlines our solution to the formulated problem, while Sections 4, 5 and 6 address different aspects of our approach. Section 7 presents experimental results and the last section draws the conclusions.

2. SYSTEM MODEL AND PROBLEM FORMULATION

2.1 Hardware model

The NoC is modelled as a 2D array of cores arranged in rows and columns, numbered from 0 to $W-1$ and to $H-1$ respectively. The core on the row x and column y is identified as $P_{x,y}$, where $0 \leq x < W$ and $0 \leq y < H$. Core $P_{x,y}$ is connected to cores $P_{x,y+1}$ (north), $P_{x+1,y}$ (east), $P_{x,y-1}$ (south), and $P_{x-1,y}$ (west) if these cores exist. The link connecting core $P_{x,y}$ to core $P_{x,y+1}$ is denoted by $L_{x,y,N}$ or by $L_{x,y+1,S}$, where the first two indexes denote one end of the link, while the third index shows the direction of the link. Each link is characterised by the time and energy it needs to transmit a bit of information.

2.2 Application model

The application is modelled as a set of task graphs. A task graph is a directed acyclic graph $(V, E \subset V \times V)$, where each vertex V_i corresponds to a task τ_i and each edge $e = (V_i, V_j) \in E$ models a data dependency between tasks τ_i and τ_j. Each task is mapped on one core and different tasks may be mapped on the same core. Let $m(\tau_i)$denote the core task τ_i is mapped onto.

Each task τ_i is characterised by its period π_i and its worst-case execution time c_i when executed on core $m(\tau_i)$. Tasks belonging to the same task graph have the same period. Task τ_i is said to *arrive* every π_i time units. Its response time (a.k.a. latency) is given by the time interval between its arrival and its completion.

Each task graph is characterised by a deadline. The task graph deadline is met if all tasks belonging to the task graph have completed their execution by the time of the graph deadline. The task graph response time is given by the maximum response time of its tasks, while the application response time (or latency) is given by the largest response time of the task graphs. In addition to task graph deadlines, every task may have its own deadline δ_i.

Each edge $e = (V_i, V_j) \in E$ is characterised by $b_{i,j}$, the largest amount of bits that is transmitted by task τ_i to task τ_j each time they are instantiated. The transmission is assumed to be ready to commence as soon as task τ_i has finished its execution. Task τ_j cannot start its execution before receiving the data items sent by *all* its predecessor tasks τ_k.

The execution of tasks mapped on the same core is scheduled based on pre-assigned priorities. This execution is preemptive.

2.3 Communication model

The time needed for the communication between two tasks mapped on the same core is assumed to be part of the worst-case execution time of the sender. Inter-core communication is packet-based, i.e. the data sent by a task is chopped at the source core in packets and then sent on the links along a predetermined route. At the destination core, the packets are assembled and the message is delivered to the receiving task. Each message is characterised by its length (bits) $b_{i,j}$, by the size of the packets it is chopped into, and by a priority for solving link contention. The packet transmission on a link is non-preemptive.

2.3.1 *Fault model*

Links may fail temporarily due to transient faults. 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 cores have the ability to detect if an incoming packet is scrambled. Scrambled packets are dropped 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 least one copy of every packet of the message has to reach the destination core unscrambled. Otherwise, the message is said to be lost. The link failure events are assumed mutually independent. The probability that a packet of b bits is conveyed successfully by a link is denoted α_b.

Let us consider an arbitrary pair of communicating tasks, $\tau_i \rightarrow \tau_j$. The two tasks have the same period π, as they belong to the same task graph. Let $S_{i,j}(t)$ denote the number of messages that were sent by task τ_i and were received unscrambled by task τ_j in the time interval $[0, t)$. The expected fraction of successfully transmitted messages for the task pair $\tau_i \rightarrow \tau_j$, called *message arrival probability* and denoted $MAP_{i,j}$, is given by

$$MAP_{i,j} = \lim_{t\to\infty} \frac{S_{i,j}(t)}{\lceil t/\pi \rceil}. \tag{1}$$

2.3.2 *Message communication support*

In order to introduce the notion of communication support we will use the example in Figure 6(a). The squares in the figure represent cores and the thick undirected lines connecting them represent the network links. The circles inside each square denote tasks that are mapped on the core represented by the square. The solid arrows connecting the circles represent the data dependence among tasks. The dashed arrows show how the data communication between the tasks is mapped on the network links. Thus, the messages between task τ_1 and τ_2 are conveyed on the link $L_{0,0,E}$. The message $\tau_1 \rightarrow \tau_3$ is sent in multiple copies. Thus, one copy circulates on the route $P_{0,0} \rightarrow P_{1,0} \rightarrow P_{1,1}$ traversing the links $L_{0,0,E}$ and $L_{1,0,N}$, while two more copies (shown as the duplicate arrow) circulate on the route $P_{0,0} \rightarrow P_{0,1} \rightarrow P_{1,1}$ traversing the links $L_{0,0,N}$ and $L_{0,1,E}$.

In general, the mapping of the communication between two tasks $\tau_i \rightarrow \tau_j$ can be formalised as a set of tuples $\mathcal{C}_{i,j} = \{(L, n) : L \text{ is a link}, n \in \mathbb{N}\}$, where n indicates how many copies of the same packet are conveyed by the corresponding link L. We will call the set $\mathcal{C}_{i,j}$ the *communication support* (CS) of $\tau_i \rightarrow \tau_j$. In our example, the two communication supports are $\mathcal{C}_{1,2} = \{(L_{0,0,E}, 1)\}$ and $\mathcal{C}_{1,3} = \{(L_{0,0,E}, 1), (L_{1,0,N}, 1), (L_{0,0,N}, 2), (L_{0,1,E}, 2)\}$.

Let us assume that a message of $b_{i,j}$ bits is supported by a particular CS. The message arrival probability (MAP) and the expected communication energy (ECE) for the message can be computed as functions of α_b, the probability that a packet is successfully conveyed across a link, and of the energy-per-bit values E_{bit} of the links of the CS, and the number $b_{i,j}$ of transmitted bits. As opposed to the message transmission time that can be affected by the interference from other transmitted messages, the MAP and ECE can be precisely computed for each message in isolation given the CS supporting it. The MAP is computed from probabilities to reach each core on the way. The latter are obtained using simple probability theory as shown by us elsewhere [10]. The ECE is a sum of expected energies consumed by each link of the CS. This energy is proportional to the probability to reach the start point of the link multiplied to the number of times the message is conveyed on the link.

For illustration, Figure 1 depicts five possible CSs for a message sent from core $P_{0,0}$ to core $P_{1,1}$. The thick arrows denote the links belonging to the CS, while the numbers that annotate them represent the number of copies to be sent on each of them. The thinly drawn links are not used by the communication. The value of α for this example is 0.97. For the CS in Figure 1(a), the probability to reach core $P_{0,1}$ is α, the MAP is α^2, and the ECE is proportional to $1 + \alpha$ (1 because the message is sent once on $L_{0,0,N}$ and α because the message is forwarded once on $L_{0,1,E}$ only if the message reaches $P_{0,1}$ successfully). Similarly, the MAP for the CS in Figures 1(b) and 1(c) is $\alpha \cdot (1 - (1 - \alpha)^2)$, and the ECE are proportional to $2 + (1 - (1 - \alpha)^2)$ and $1 + 2 \cdot (1 - (1 - \alpha)^2)$ respectively. The MAP for the CS in Figure 1(d) is $(1 - (1 - \alpha)^2)^2$ and the ECE is proportional to $2 + 2 \cdot (1 - (1 - \alpha)^2)$. The MAP for the CS in Figure 1(e) is $\alpha^2 \cdot (2 - \alpha^2)$ and the ECE is proportional to $2 + 2 \cdot \alpha$.

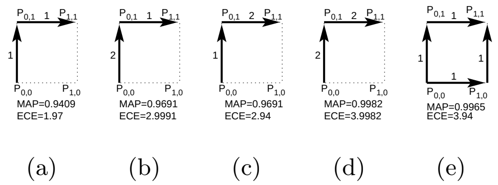


Figure 1: CSs for a 2×2 NoC

2.4 Problem formulation

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;
- The application model, i.e. the task graphs, their deadlines, the mapping of tasks to cores, the task periods, deadlines, worst-case execution times, 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 $MAP_{i,j}$, which is the expected fraction of successfully transmitted messages, for each pair of communicating tasks $\tau_i \rightarrow \tau_j$.

The problem is formulated as follows: Given the input described above, find the communication support $\mathcal{C}_{i,j}$ for each pair of communicating tasks $\tau_i \rightarrow \tau_j$ such that:

- all message arrival probabilities $MAP_{i,j}$ satisfy $MAP_{i,j} \geq B_{i,j}$,
- the communication energy is minimised, and
- all deadlines are met

3. APPROACH OUTLINE

The outline of our approach to solve the problem is shown in Figure 2. First, for each pair of communicating tasks (message), we find a set of candidate communication supports (line 2, see Section 4), 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 shortest application response time *min_rt* (line 4, see Section 6). 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 5). If no solutions are found that satisfy the response time constraint, the application is

```
(1)   for each pair of communicating tasks $\tau_i \rightarrow \tau_j$
(2)       find a set of candidate CSs that
              satisfy $MAP_{i,j} \geq B_{i,j}$ (Sec. 4)
(3)   end for
(4)   $(sol, min\_rt)$ =explore the space of candidate CSs (Sec. 6)
(5)       using response time calculation (Sec. 5)
          for driving the exploration
(6)   if $min\_rt >$ deadline then
(7)       return "no solution"
(8)   else
(9)       $sol'$ =voltage_freq_reduction($sol$) (according to [1])
(10)      return $sol'$
(11)  end if
```

Figure 2: Approach outline

deemed impossible to implement with the given resources (line 7). Otherwise, the solution with the minimum application response time among the found solutions is selected. Voltage reduction 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 5 describes how the response time calculation is performed, while Section 6 outlines how the preferred communication supports representing the final solution are selected.

4. 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 a source core with a destination core is an ordered sequence of n links, such that the end point of the i^{th} 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 core and the end point of the last link is the destination core. 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.

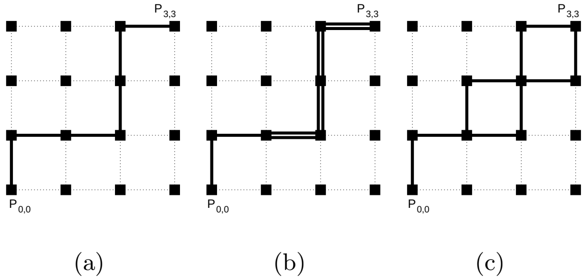
The spatial redundancy degree (SRD) of a CS is given by the minimum number of distinct paths belonging to the CS that cover all the links of the CS. For the example in Figure 6(a), the SRD of $\mathcal{C}_{1,2}$ is 1, as $\mathcal{C}_{1,2}$ contains only one path, $(L_{0,0,E})$. The SRD of $\mathcal{C}_{1,3}$ is 2, as the four links of $\mathcal{C}_{1,3}$ can be covered only by the paths $(L_{0,0,E}, L_{1,0,N})$ and $(L_{0,0,N}, L_{0,1,E})$.

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 the example in Figure 6(a), the TRD of $\mathcal{C}_{1,2}$ is 1 and the TRD of $\mathcal{C}_{1,3}$ is 2 (because two redundant copies are sent on links $L_{0,0,N}$ and $L_{0,1,E}$).

The general redundancy degree (GRD) of a CS is given by the sum of temporal redundancy degrees of all its links. For the example in Figure 6(a), the GRD of $\mathcal{C}_{1,2}$ is 1 and the GRD of $\mathcal{C}_{1,3}$ is 6.

```
for each pair of communicating tasks $\tau_i \rightarrow \tau_j$
    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
    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 $\tau_i \rightarrow \tau_j$
end for
```

Figure 3: Construction of candidate CS set



(a) (b) (c)

Figure 4: Selection of minimal GRD

It is important to use CSs of minimal GRD because the ECE of a message is strongly dependent on the GRD of the CS supporting it. To illustrate, 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 Figure 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 same conclusion is supported also by Figure 1.

The algorithm for the candidate set construction proceeds as shown in Figure 3. 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 Figure 4(a). 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 GRD of 7, and regardless to which of the 6 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 5 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 which 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 4 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 Figure 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 4 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.

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 [10]. 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 core 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 $N_2 = 10$ ($MAP = \alpha^6 \cdot (2-\alpha^2)^2 \approx 0.9793$). This CS is shown in Figure 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 negatively affect the message latency. 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 Figure 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. This can be also observed when comparing the CSs in Figures 1(d) and 1(e).

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 which belong to shortest paths between the source and the destination and whose start points coincide with the source core. Thus, if a CS consisted only of 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 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.

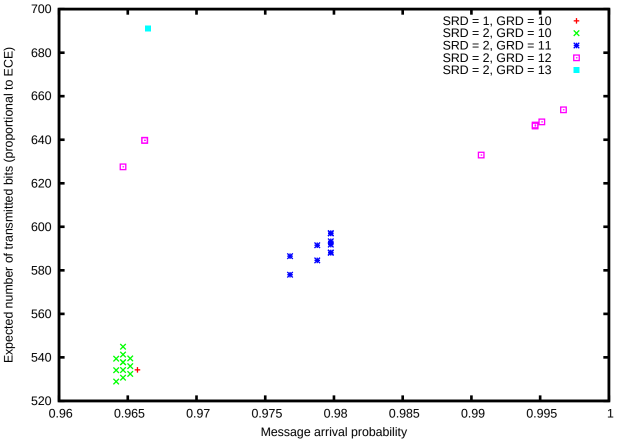
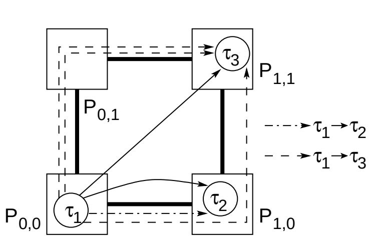
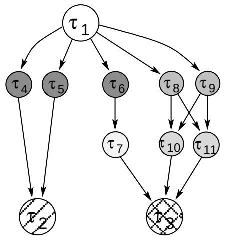


Figure 5: Energy-efficiency of CSs of SRD 1 and 2



(a)



(b)

Figure 6: Application modelling for response time analysis

5. 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 Figure 6(a) and described in Section 2.3.2. The two CSs are $\mathcal{C}_{1,2} = \{(L_{0,0,E}, 1)\}$ and $\mathcal{C}_{1,3} = \{(L_{0,0,E}, 1), (L_{1,0,N}, 1), (L_{0,0,N}, 2), (L_{0,1,E}, 2)\}$. 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 Figure 6(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 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 Figure 6(b), is analysed using the dynamic offset based schedulability analysis proposed by Palencia and Harbour [11]. The analysis calculates the worst-case response times and jitters for all tasks.

6. SELECTION OF COMMUNICATION SUPPORTS

As shown in Section 4 (see also line 2 in Figure 2), 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 last step of our approach (line 4 in Figure 2) we have to select one particular CS for each message, such that the application response time is minimised. The response time for each candidate solution is calculated as outlined in Section 5 (line 5 in Figure 2).

The solution space is explored with a Tabu Search based heuristic [7]. Given a certain solution alternative, a new one is generated by performing a "move". A move means picking a pair of communicating tasks and selecting a new communication support for it. In order to select a move, classical Tabu Search explores all solutions that can be reached by one move from the current solution. For each candidate solution, the application response time has to be calculated. Such an approach would be too time consuming for our problem. Therefore, we only explore "promising" moves. 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;
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 message from overloaded links.)

During design space exploration, the reverse of a performed move is marked as tabu, which means that it is forbidden for a number of exploration steps [7]. Thus, the best available solution at a certain step can be worse than the current best one. This could be the case if there are no moves available that improve the latency or all such moves are tabu. This feature ensures that the heuristic does not get stuck at local minima.

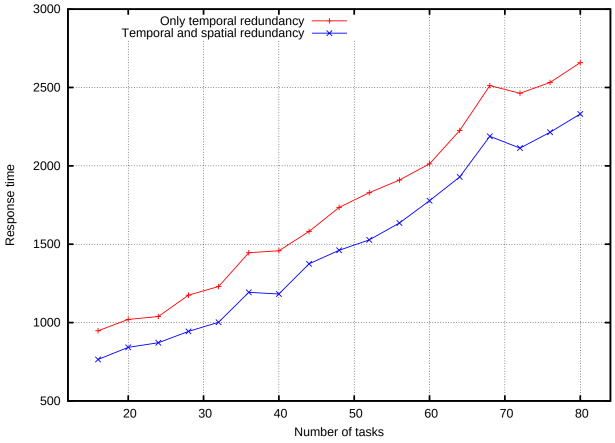


Figure 7: Application latency vs number of tasks

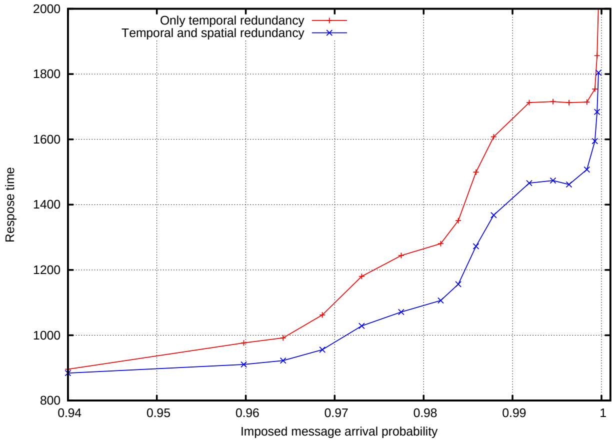


Figure 8: Application latency vs bound on MAP

7. EXPERIMENTAL RESULTS

We report on three sets of experiments that we ran in order to assess the quality of our approach. The first set investigates the application latency as a function of the number of tasks. 340 applications of 16 to 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 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 4. Figure 7 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.

The second experiment investigates the dependency of latency on the imposed message arrival probability. 20 ap-

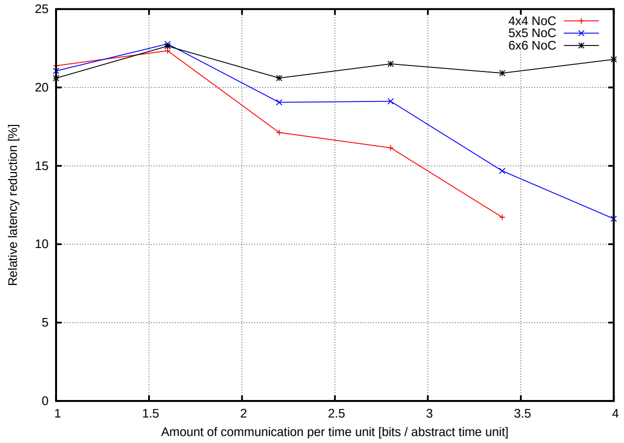


Figure 9: Application latency vs NoC size and communication load

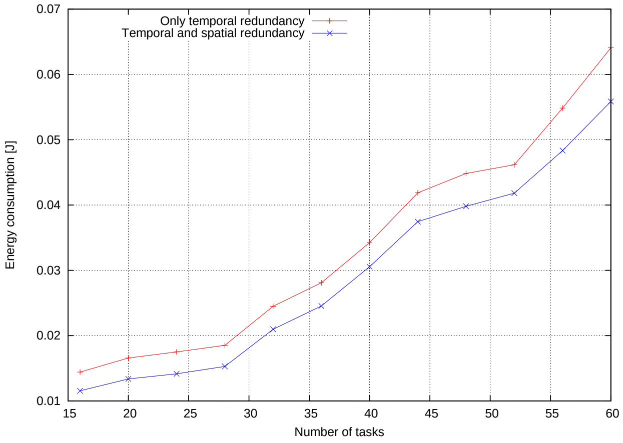


Figure 10: Energy consumption vs. number of tasks

plications, 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 Figure 8. 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 (as shown in Section 4) 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 and the approach combining spatial and temporal redundancy does not lead to significant latency reductions anymore.

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). 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 Figure 9. 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.

The presented experiments have shown that, 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 4. This means that for a class of applications using the proposed approach it will be possible to meet the imposed deadlines, which otherwise would be not 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.

Applications of 16 to 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 [1] for calculating the voltage levels for which to run the application. For our energy models, we considered a 70nm CMOS fabrication process. The resulted energy consumption is depicted in Figure 10. The energy reduction ranges from 20% to 13%.

Finally, we applied our approach to a multimedia application [8], namely an image encoder implementing the H263 algorithm. The application is composed of 24 tasks running on a platform consisting of 6 DSPs, 2 CPUs, 4 ASICs, and 2 memory cores (organised as a 4×4 NoC with one unused core). 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%.

8. CONCLUSIONS

This paper has presented an approach to reliable, low-energy on-chip communication for time-constrained applications implemented on NoC. The contribution is manifold. First, we show how to generate supports for message communication in order to meet the message arrival probability constraint and to minimise communication energy. Second, we give a heuristic for selecting most promising communication supports with respect to application responsiveness and energy. Third, we model the fault-tolerant application for response time analysis. Finally, we present experiments demonstrating the proposed approach.

9. REFERENCES

[1] 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*, 2004.

[2] L. Benini and G. De Micheli. Networks on chips: a new SoC paradigm. *IEEE Computer*, 35(1):70–78, 2002.

[3] D. Bertozzi, L. Benini, and G. De Micheli. Low power error resilient encoding for on-chip data buses. In *Proc. of DATE*, pages 102–109, 2002.

[4] W. Dally. Interconnect-limited VLSI architecture. In *IEEE Conference on Interconnect Technologies*, pages 15–17, 1999.

[5] J. Dielissen, A. Radulescu, K. Goossens, and E. Rijpkema. Concepts and implementation of the Philips network-on-chip. In *IP-Based SoC Design*, 2003.

[6] T. Dumitraş and R. Marculescu. On-chip stochastic communication. In *Proc. of DATE*, 2003.

[7] F. Glover. Tabu search—Part I. *ORSA J. Comput.*, 1989.

[8] J. Hu and R. Marculescu. DyAD–Smart routing for Network-on-Chip. In *Proc. of DAC*, 2004.

[9] D. Liu et al. Power consumption estimation in CMOS VLSI chips. *IEEE Journal of Solid-State Circuits*, (29):663–670, 1994.

[10] S. Manolache. Fault-tolerant communication on network-on-chip. Technical report, Linkoping University, 2004.

[11] J. C. Palencia Gutiérrez and M. González Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings of the* 19^{th} *IEEE Real Time Systems Symposium*, pages 26–37, December 1998.

[12] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and I. M. J. Fault tolerant algorithms for network-on-chip interconnect. In *Proc. of the ISVLSI*, 2004.

[13] K. Shepard and V. Narayanan. Noise in deep submicron digital design. In *ICCAD*, pages 524–531, 1996.