White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/118107/

Version: Accepted Version

Proceedings Paper:
Soares Indrusiak, Leandro, Harbin, James Robert and Sepulveda, Martha Johanna (2017) Side-Channel Attack Resilience through Route Randomisation in Secure Real-Time Networks-on-Chip. In: Proceedings of the 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC 2017).

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Side-Channel Attack Resilience through Route Randomisation in Secure Real-Time Networks-on-Chip

Leandro Soares Indrusiak
Department of Computer Science,
University of York, UK.
Email: leandro.indrusiak@york.ac.uk

James Harbin
Department of Computer Science,
University of York, UK.
Email: james.harbin@york.ac.uk

Martha Johanna Sepulveda
Institute for Security in Information Technology,
Technische Universitaet Muenchen, Germany.
Email: johanna.sepulveda@tum.de

***Abstract*—Security can be seen as an optimisation objective in NoC resource management, and as such poses trade-offs against other objectives such as real-time schedulability. In this paper, we show how to increase NoC resilience against a concrete type of security attack, named side-channel attack, which exploit the correlation between specific non-functional properties (such as packet latencies and routes, in the case of NoCs) to infer the functional behaviour of secure applications. For instance, the transmission of a packet over a given link of the NoC may hint on a cache miss, which can be used by an attacker to guess specific parts of a secret cryptographic key, effectively weakening it.**

We therefore propose packet route randomisation as a mechanism to increase NoC resilience against side-channel attacks, focusing specifically on the potential impact of such an approach upon hard real-time systems, where schedulability is a vital design requirement. Using an evolutionary optimisation approach, we show how to effectively apply route randomisation in such a way that it can increase NoC security while controlling its impact on hard real-time performance guarantees. Extensive experimental evidence based on analytical and simulation models supports our findings.

I. Introduction

The design of Network-on-Chip (NoC) interconnects for embedded systems requires the careful balance of multiple trade-offs. Over the past decades, a significant amount of work has addressed the trade-offs between performance and other secondary objectives such as energy [24], fault-tolerance [17], and chip area [16]. Less work has addressed such trade-offs in NoCs with hard real-time constraints, with some inroads towards improving energy [18] and area efficiency (by optimising buffering in virtual channels [14]) while meeting deadlines of all packets even in the worst-case scenario.

In this paper, we consider NoCs with hard real-time constraints, and address a novel trade-off that has increasing importance in embedded systems: security. Because of their key role in interconnecting the multiple components of an embedded system, NoCs can be seen as a major security vulnerability. If an attacker can extract information from the NoC interconnect, they can potentially compromise the security of the complete embedded system. Therefore, many mechanisms have been designed to improve NoC security (as reviewed in Section II) and many more will certainly be developed in the coming years. However, most of such mechanisms impose performance overheads, and therefore can potentially jeopardise the ability of the NoC to provide real-time guarantees. So we argue in this paper that, just like in the previously mentioned trade-offs, security can be seen as an optimisation objective in NoC resource management: designers must carefully consider the resources they have available to increase NoC security without sacrificing performance guarantees (which in the case of hard real-time NoCs will always be the primary objective).

The specific problem we address is a security mechanism that aims to improve the NoC resilience to side-channel attacks. Such attacks try to break a secure system by gathering information from the system's timing behaviour, power consumption, temperature or electromagnetic emissions. Just like some of the related work [25] [20], we aim to improve resilience against side channel attacks by randomising the behaviour of the NoC, aiming to make it difficult for an attacker to identify patterns and correlations between the functionality of the system and the timing, power, temperature and electromagnetic behaviour of the NoC. As expected, such an approach has a direct impact on NoC resource usage, and therefore on its real-time guarantees, so we identify techniques that support NoC designers in improving NoC resilience against side-channel attacks while still maintaining full system schedulability.

II. Related Work

Multiprocessor embedded systems are target of attacks by means of malicious hardware or software [4]. Hardware-based attacks depend on design-time access to the system, which is then modified in a way that can be exploited during operation (e.g. by adding hardware able to leak information by changing chip temperature [8]). Software-based attacks are the most common cause of security incidents in such types of systems [15], and are carried out by malicious software installed at design time or after deployment.

NoC-based systems have been shown to be vulnerable to a variety of attacks, both hardware and software-based. Active NoC attacks, such as code injection [1], malware [5] and

control hijacking [12], or passive NoC attacks, such as side-channel exploitation, can be used to read sensitive communications, modify the system behaviour or prevent correct NoC operation. NoCs are especially vulnerable to side-channel attacks that exploit traffic interference as timing channels [26] [20]. The shared nature of NoCs can be exploited by an attacker to obtain sensitive information. By forcing traffic collision with sensitive packet flows, an attacker can observe the throughput variations and infer sensitive data, as shown in [26] [20] [25].

Randomised arbitration [20], virtual channel allocation [21] and routing [25] have been investigated and evaluated as countermeasures against timing attacks. By randomising the characteristics of sensitive packet flows, it is possible to break the correlation between the traffic characteristics (e.g. volume and access patterns) and the sensitive data thus avoiding information leakage. Among those mechanisms, random routing has achieved the best levels of security enhancement with the lowest energy and area overhead [25]. By spreading sensitive traffic over the NoC, the spatial distribution makes it harder for compromised cores or external attackers to gather sufficient side-channel information to infer correlations with sensitive data.

The focus of state-of-the-art randomisation approaches is to increase security, and none of those works consider the performance requirements of the applications. In this paper, we argue that NoCs supporting real-time applications require a careful balance of a trade-off between security and performance. In most cases, we envisage that the level of security will be constrained by the NoC's ability to support attack countermeasures while at the same time ensuring performance guarantees to the application.

Thus, the main contributions of this paper are the identification of a test to evaluate whether performance guarantees can hold under a specific side-channel attack countermeasure (namely route randomisation), and a technique that uses that test to better balance the trade-off between performance guarantees, resource usage and security.

III. Problem Description

A. Network-on-Chip Architecture

While the contribution of this paper can be applied to a large variety of NoC architectures, we believe it is easier to explain it with the help of a concrete architecture. We assume a NoC architecture with a 2D-mesh topology and wormhole switching protocol, because such features are commonly used in embedded systems for their simplicity and moderate resource overheads. There is a downside to this choice, which is the difficulty in predicting packet latencies. In wormhole networks, a packet can be simultaneously occupying multiple NoC buffers and links, so there is a significant amount of competition for resources throughout the NoC at all times. The wide variety of interference patterns makes it hard to predict how long it takes for a packet to reach its destination. Different resource arbitration policies can make such predictions more or less difficult, especially in the case of hard real-time NoCs when an upper-bound worst-case latency is needed. Previous work has considered NoC arbitration based on packet priority [23], time multiplexing [19] and round robin [2], and has devised analytical models that can be used to find latency upper-bounds for packet flows transmitted over such NoCs [11]. Any of those approaches could be used in this paper, and we chose a priority-arbitrated NoC because of its ability to provide upper-bound latency guarantees that are customisable to different levels of packet urgency while allowing for high NoC link utilisation [9].

B. Threat Model

We assume that the NoC and its interfaces to the cores are secure. We also assume that secure tasks execute in secure cores (i.e. cores that do not allow the execution of unsecured tasks). For this threat model, we assume that the NoC communicates sensitive information between two secure tasks, which we refer as the sensitive communication. We then assume an adversary that has knowledge about the NoC architecture, about the mapping of secure tasks to (secure) NoC cores, and is able to gain control of at most two non-secure NoC cores.

A successfull attack happens when the adversary is able to infect two cores that can communicate over a route that intersects with that of the sensitive communication. In that case, the adversary is able to use one of the infected cores to inject low priority packets into the NoC towards the second infected core. The latency interference imposed by the sensitive communication over the malicious low priority traffic can provide the attacker with valuable information about the timing, frequency and volume of the secure communication.

This threat model is not new, and its variations have also been used in best-effort NoC-based systems by [26] and [21]. The timing nature of the threat is also the same used in hard real-time uniprocessor systems by [27].

By using a route randomisation approach, it is possible to prevent the adversary from obtaining accurate information about the sensitive communication. Since each packet of the secure communication may follow a different route, only some of them will be intercepted by the probing packets injected by the infected cores. Thus, the information about timing, frequency and volume the attacker can obtain will be less accurate: inferred frequency and total volume of sensitive packets will be lower than the real value, since not all packets will be detected by the attacker; inferred timing will deviate from the real value because the amount of blocking that a sensitive packet could suffer will have more variability due to the randomness of the routes of the packets that may block them. This, as a consequence, increases the resilience of the NoC against the threat. There are many ways to introduce route randomisation in NoCs, and we will discuss our design decisions in subsection IV-A.

Figure 1 shows an example of the described threat model. It shows an adversary controlling cores F and G, and using a malicious packet flow (shown as a purple dashed line) to infer data about a sensitive communication between secure cores C

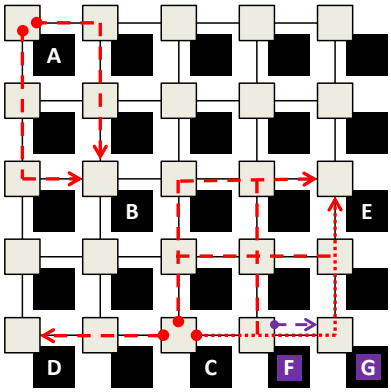

Fig. 1: Threat model, and examples of route randomisation with pseudo-adaptive XY (from A to B) and west-first (from C to D and C to E) algorithms

and E (shown as a red dotted line, representing the case of a NoC with deterministic XY routing). In the case of a NoC with randomised routing, all routes between C and E will be used (red dashed and dotted lines), preventing the adversary from inspecting the complete sensitive communication.

C. System Model

To increase NoC resilience against side-channel attacks while providing hard real-time guarantees to the application tasks running on it, we must make assumptions about the application behaviour such as upper-bounds on resource usage by every application task and packet. In this paper, we follow the well-known and widely used sporadic task model, which makes assumptions about the worst-case execution time (WCET) of all tasks and their shortest inter-arrival interval (i.e. their period). Since we are concerned about NoC communications, we follow an extension of the sporadic task model that considers that tasks inject packets to the NoC only after their execution completes, and that the maximum packet size is known [9].

Thus, a hard real-time application Γ comprises n real-time tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task τ_i is a 6-tuple $\tau_i = (C_i, T_i, D_i, J_i, P_i, \{\phi_i\})$ indicating respectively its worst case computation time, period, deadline, release jitter and priority. The sixth element of the tuple is an extension to the sporadic task model proposed by [9], and represents the communication packets sent by τ_i at the end of its execution. Each packet ϕ_i is defined as a 3-tuple $\phi_i = (\tau_d, Z_i, K_i)$ representing its destination task, size and maximum release jitter. In this paper, we assume for simplicity that a single packet is released at the end of each execution of each task, but the contributions presented here can be generalised for any number of released packets.

Such applications are executed over a NoC platform like the one described in subsection III-A above. We model such a platform as a set of cores $\Pi = \{\pi_a, \pi_b, \ldots, \pi_z\}$, a set of switches $\Xi = \{\xi_1, \xi_2, \ldots, \xi_m\}$, and a set of unidirectional links $\Lambda = \{\lambda_{a1}, \lambda_{1a}, \lambda_{12}, \lambda_{21}, \ldots, \lambda_{zm}, \lambda_{mz}\}$. We also model the mapping of tasks to cores with the function $map(\tau_i) = \pi_a$.

The routing of packets over the NoC can be modelled by the function $route(\pi_a, \pi_b) = \{\lambda_{a1}, \lambda_{12}, \ldots, \lambda_{mb}\}$, denoting the subset of Λ used to transfer packets from core π_a to core π_b. We can then extend the function map to also model the mapping of a packet to its route: $map(\phi_i) = route(map(\tau_i), map(\tau_d))$.

With the knowledge of the NoC architectural characteristics such as the latency to cross a link or to route a packet header, and with the knowledge of the length of a packet's route (i.e. its hop count, or $|route(\pi_a, \pi_b)|$ as expressed in [9]), it is possible to calculate the no-load latency L_i of every packet ϕ_i: the time it takes to completely cross the NoC from its source to destination without any interference or contention from other packets. For the NoC described in subsection III-A, and for most commercial and academic NoCs, the no-load latency of a packet can be deterministically obtained, and will not change if its route and the NoC operation frequency do not change.

IV. NoC Routing Randomisation

A. Design Choices and Constraints

The architecture of a NoC defines whether and how it can implement route randomisation. For example, some NoC architectures use deterministic routing [13], meaning that there is only one possible route between a source and a destination, effectively preventing the approach proposed here. Among NoCs supporting dynamic or adaptive routing, which are the ones we target, there is a key design choice affecting the randomisation approach: source or distributed routing.

In source-routed NoCs, the routing decision is done by the source core or its respective NI. This is usually implemented as multiple packet header flits that contain the next-hop information for each of the switches along the packet's route. Once a switch routes one of the packet headers by assigning its output port, it discards that header flit and forwards the rest of the packet through that port. The next switch will route the subsequent header flit, discard it, forward the rest of the packet, and this is repeated all the way towards the packet destination. By following this approach, it is possible to program the source core or its NI to perform full route randomisation before every packet release.

In NoCs with distributed routing, the next-hop decision is made by each switch individually. Typically, they have far less resources than the cores (and often than the NIs), so the routing decisions are based on simple rules related to the relative position of the destination core with regards to the switch holding the packet header (e.g. pseudo-adaptive XY [3], turn model [6]). In those cases, it is only possible to randomly choose from a predefined subset of all possible routes. For instance, pseudo-adaptive XY switches can only randomly choose between two routes between a source and a destination (e.g. routes between cores A and B in Figure 1). Switches implementing turn model routing may have a larger number of alternative routes to randomly choose from in most cases, but must behave deterministically for some specific cases. Figure 1 shows two routes created by a west-first turn model: packets between core C and D have only one possible

route, as the destination is located on the west of the source, while packets from core C to E can take a variety of possible routes.

In both source and distributed routing, the NoC component making random decisions must have access to a source of random data, such as a pseudo-random number generator (PRNG, generated by a deterministic algorithm) or a true random number generator (TRNG, often generated out of low level noise signals). Such sources can have significant hardware overhead, thus favouring source routing because of the low area constraints for NoC switches.

Additional issues when randomising packet routes include the potential increase of the packet route, the possibility of deadlocks, and the potential increase of packet latency (and therefore the potential violation of real-time constraints). Let us now address each of them.

All the routing approaches reviewed above are minimal: the route they choose has the smallest possible hop count between source and destination. This is because of their obvious advantages in terms of latency, network contention and energy dissipation. However, from the point of view of side-channel attack resilience, it may be interesting to exploit non-minimal randomised routing in order to decorrelate the side channels with the functional properties of the packet communication (e.g. short packet transmission between neighbouring cores would not necessarily have the shortest latency and lowest energy dissipation if they are forced to take a long route across the chip).

Deadlock-free packet communication is a critical characteristic for NoCs. This can be achieved at the link arbitration layer, e.g. with priority-preemptive virtual channels [9], or at the network layer by restricting the possible turns of the routing algorithm (either in source or in distributed routing). In NoCs that ensure deadlock-freeness at the network layer, special care must be taken by the route randomisation approach to avoid introducing turns that can lead to deadlocks.

Finally, route randomisation is likely to change the latencies of packets, both because for every release their routes may have different hop counts (leading to different no-load latencies) and because different routes may trigger different contention scenarios (leading to different blocking times). In our approach, such variability is actually desirable because it is a key aspect to increasing the NoC's resilience against side channel attacks. In the case of hard real-time systems, however, it is critical that such variability is bounded and that the worst-case latencies of all packets are always less than their deadlines. In the next subsection, we propose an extension to existing schedulability analysis to evaluate if that is the case for a given application mapped to a given NoC architecture. The proposed approach is simple, yet general enough to analyse randomised routing approaches following any of the design choices reviewed above: source or distributed, minimal or non-minimal, and with deadline-freeness ensured at the link or network layer.

B. Schedulability Analysis

Schedulability analysis for a set of sporadic packets transferred over a priority-preemptive wormhole switching NoC was presented in [22]. A set of packets is deemed schedulable if the worst-case latency of each packet is less than their deadline. By coupling that analysis with classical response time analysis for uniprocessor fixed-priority scheduling, an end-to-end schedulability analysis for that type of NoC was proposed in [9], considering the worst-case response times of tasks and the worst-case latency of the packets they generate. Both the original analysis from [22] and the end-to-end extension from [9] assume static routing, so a different formulation is needed before it can be used for the purpose of this paper. First, we review those formulations, but using the notation described in subsection III-C.

According to [22], the worst-case latency S_i of a packet ϕ_i can be obtained from Equation 1. This equation is defined recursively and iterated until a stable fixed point is discovered.

$$S_i \;=\; L_i \;+\; \sum_{\phi_j \in \mathbf{interf}(i)} \left\lceil \frac{S_i + K_j + K_j^I}{T_j} \right\rceil L_j, \qquad (1)$$

The set $\mathbf{interf}(i)$ is the set of higher priority packets ϕ_j whose route shares at least one link with the route of ϕ_i and therefore can interfere with it. Precisely, $\mathbf{interf}(i) = \{\phi_j \in \phi : map(\phi_i) \cap map(\phi_j) \neq \emptyset\}$. The two terms K_j and K_j^I denote respectively the maximum release jitter of the interfering packet ϕ_j and its maximum indirect interference jitter. As shown in [9], K_j is equal to the worst case response time R_j of task τ_j which produces ϕ_j, assuming that ϕ_j will be released immediately after the end of τ_j's execution. R_j can be calculated using uniprocessor response time analysis, considering the type of task scheduling by the operating system at each core (e.g. priority-preemptive). And as shown in [22], the indirect interference jitter K_j^I can be bound by $S_j - L_j$.

It can be seen in Equation 1 that the route of a packet affects its worst-case latency because it defines the set of packets that can add to the interference term of the equation (i.e. sum operator). Route randomisation would change the set $\mathbf{interf}(i)$ at each packet release, since different routes would produce different interference patterns. An intuitive way to find the worst-case latency of a packet with a randomised route would be to calculate the worst-case latency of each of its possible routes with Equation 1, and pick the highest value. However, that approach works only if there is a single packet with randomised route, and all others following deterministic routes.

A general analysis where all packets could potentially have randomised routes is more complex: all possible routes of a packet would have to be tested with all possible routes of all other packets before the worst case could be found. Furthermore, if one cannot make probabilistic assumptions on the randomisation approach, pathological cases must also be taken into account (e.g. the same route could be chosen again

and again for a single packet over a long period of time, even though that is very unlikely).

In this paper we assume that, in the worst case, if there is a way for a high-priority packet to interfere with a low priority packet, it would interfere with it in every possible release. This means that even though there may be routes when packets do not interfere with each other, we assume that in the worst case the random choice of route would always pick the ones where there is interference. This is perfectly reasonable when packets have similar periods, but it gets more and more pessimistic as we reduce the periods of higher priority packets. In that case, high priority packets would have a larger number of releases within a single release of a low priority packet, thus interfering more often with it, even though the larger number of releases would make less likely that an interfering route would be chosen every time.

To calculate worst-case latencies for the general problem where all packets could have randomised routes, we define the set $\mathbf{interf_r}(i)$ as the set of higher priority packets ϕ_j who could, with any of their possible routes, interfere with any of the possible routes of the packet of interest ϕ_i. To precisely define that set, we must first define a new function $route_r(\pi_a, \pi_b) = \{\lambda_{a1}, \lambda_{12}, \lambda_{13}, \lambda_{14}, \ldots, \lambda_{mb}\}$, denoting the subset of Λ that contains all the links that could be part of any of the routes that could be randomly chosen to transfer packets from core π_a to core π_b, and a new function $map_r(\phi_i) = route_r(map(\tau_i), map(\tau_d))$. Then, $\mathbf{interf_r}(i) = \{\phi_j \in \phi : map_r(\phi_i) \cap map_r(\phi_j) \neq \emptyset\}$.

By applying Equation 1 with the summation over the set $\mathbf{interf_r}(i)$ instead of the original $\mathbf{interf}(i)$, we can then find an upper bound to the packet latencies over a NoC with randomised routing.

C. Optimising the Performance-Security Trade-off

The analysis proposed above can only be used to test whether a particular randomised NoC configuration can meet all hard real-time constraints of an application, but offers no alternatives in case of negative results. In this subsection we show how such analysis can be exploited as a fitness function in a design space exploration process. Similarly to [18] and [9], we follow an evolutionary approach to navigate over a key part of the design space: task-core mapping. By changing that mapping, it is possible to achieve fine-grained improvements on schedulability of tasks over cores and packet flows over NoC infrastructure (e.g. tasks that are barely unschedulable can become schedulable by a simple remapping of one of the higher priority tasks that interfere with their computation or communication, thus changing the set $\mathbf{interf}$ in Equation 1). The same can happen in the case of route randomisation, since changes on mapping can determine which randomised routes interfere with each other and in turn affect schedulability through changes in the $\mathbf{interf_r}$ set.

Figure 2 shows the evolutionary pipeline proposed here, which start with an arbitrary population of task mappings using a given route randomisation approach and a given level of security. It then uses evolutionary operators such as mutation and crossover to improve the mapping population with regards to the percentage of schedulable tasks and packets calculated using the proposed modification of Equation 1. For every generation of the population, those with the larger number of schedulable tasks and packets are selected to the next generation, where they will be again mutated, crossed-over, evaluated and selected to the subsequent generation. The pipeline stops after a fully schedulable mapping is found, or a predefined maximum number of generations is reached.

Unlike many constructive task mapping approaches, the evolutionary pipeline proposed here does not necessarily try to map communicating tasks to the same or neighbouring cores. Its fitness function can be tuned, for instance, to keep communicating tasks as far apart as possible while keeping their communication packets schedulable over a variety of randomly-chosen routes.

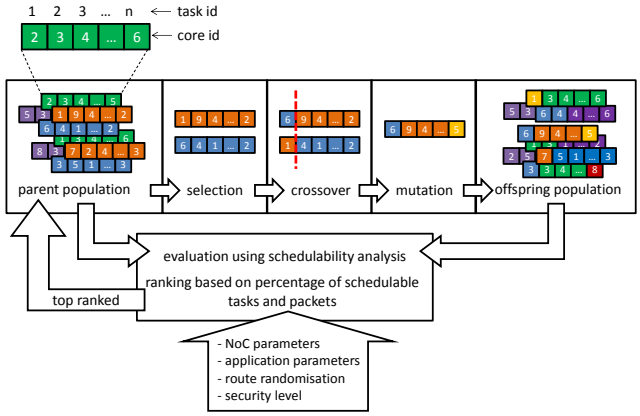

Fig. 2: Evolutionary pipeline

In this paper, we consider two types of route randomisation which can be implemented either as source or distributed routing, namely random XY/YX and random west-first. Random XY/YX is a randomised version of pseudo-adaptive XY routing used in [3], so the route of the packet to its destination is randomly chosen between the XY or the YX route prior to the injection of the packet header into the network. In random west-first, we randomise one of the turn model routing approaches [6] so that whenever a packet is allowed more than one route it randomly chooses one of them (i.e. uniform probability among all alternatives).

We then allow for multiple levels of security by changing how many packet flows are allowed to have their routes randomised. A baseline with no randomisation should have the best results regarding schedulability, given that packets suffer less interference and therefore are more likely to be schedulable. Then, increased levels of security can be achieved by randomised larger percentages of packet flows, up to a fully randomised configuration where all packets follow randomised routes on every release. In the next section, we show experimentally that the proposed schedulability test and evolutionary optimisation pipeline can produce NoC configurations able to hold hard real-time guarantees with maximised security potential.

V. Experimental Work

We evaluate the proposed approach in two distinct experimental setups. The first uses the proposed schedulability test and evolutionary pipeline to balance the trade-off between performance guarantees and security over a large set of synthetically generated applications. The second uses a cycle-accurate NoC simulator to show the effects of route randomisation upon latency with a realistic application.

A. Schedulability-driven optimisation of route randomisation

To evaluate the challenge of optimising different applications with different levels of load, we synthetically generate thousands of applications, each of them composed of tasks that communicate with each other with different numbers of packet flows. We then apply the evolutionary pipeline presented in Figure 2 to each one of those applications, aiming to optimise the mappings of tasks in such a way that the whole set of tasks and flows is schedulable at the highest possible level of security. We then plot the percentage of schedulable applications we could achieve for each level of security and each level of load. For the sake of reproducibility, we provide below more details on the whole process.

For a single experiment upon a given NoC and set of parameters (e.g. topology, operating frequency, switch and link latencies), a range of packet flow counts are identified, each of which represents a level of load upon the NoC. For each flow count, a set of tasksets and packet flowsets are generated, each containing the chosen number of flows. The number of tasks is kept roughly constant, and all of them are either source or destination of at least one packet flow. Therefore, flowsets with higher flow counts represent increasing packet contention between the same endpoints. Flows are assigned to particular source and destination tasks with uniform random probability. This implies that the average number of flows transmitted is even across all tasks, although as a result of the random assignment there may be hotspots.

An experiment is initialised by defining a population of initial mappings, and a setting the target level of security. The levels of security settings are defined as either unsecured, or 25%, 50%, 75% and 100% secured flows. The secured flows are those that will use randomised routing, providing increased potential protection against side-channel attacks. In case of a partial provision of security e.g. 50%, security is assigned to the flows in their order of priority, with the highest priority flows being randomised. The rationale is to enforce overall random interference patterns, since higher priority packets are the ones causing interference.

We then follow the evolutionary pipeline from Figure 2, using a fitness function based on the modified Equation 1 to evalute each individual mapping. This is done separately for each level of security, each of them generating a different $\mathbf{interf_r}(i)$ set representing the randomised routes of different packet flows.

By applying the modified Equation 1 for every packet flow of the application, it is possible to check whether each of them is schedulable within a given mapping, i.e. their end-to-end latency is less than the respective deadline. The overall fitness of a mapping is then assumed to be the number of schedulable packet flows it can achieve. After such evaluation, the population is culled to retain only the mappings that are at the top of the fitness ranking. The pipeline ends if a mapping can make all flows schedulable, or a maximum number of generations is reached.

To show the impact of the level of security on performance guarantees and resource usage, we have produced several experimental series:

No security (NS) Deterministic routing, fitness function incorporates schedulability calculated using Equation 1 with the original $\mathbf{interf}(i)$ set.

Percentage security (PS(%)) A given percentage of the packet flows use randomised routing, fitness function evaluated using Equation 1 with the proposed $\mathbf{interf_r}(i)$ set reflecting that percentage.

Application of security a posteriori (SAP) Evolution is performed using a fitness function that tests the schedulability without any security mechanisms (only deterministic routing), aiming to find a schedulable mapping without security considerations. Following the completion of this evolutionary process, the evolved best application mapping has 100% of its packet routes randomised, and is then evaluated with Equation 1 with the proposed $\mathbf{interf_r}(i)$ set. This experiment therefore aims to show that the optimisation of the mapping should take into account route randomisation, and that poor results can be expected from applying randomisation to a mapping that was optimised for deterministic routing.

The first plot in Figure 3 shows that, for experiments with increasing number of flows (i.e. increasing load over the NoC), the propotion of schedulable flows decreases as expected. Different levels of percent security (PS) produce similar results as the unsecured setup (NS), showing that the evolutionary pipeline was able to find mappings that randomised routes only when the additional overheads could be tolerated. In the SAP setup, route randomisation is added to a mapping that has evolved without any concern for randomisation, and its poor results serve as evidence of the superiority of the approach proposed here, where the evolution pipeline aims to jointly optimise mapping and randomisation. The second plot is based on the same data, but plots the percentage of schedulable

TABLE I: Evaluation parameters

NoC/Packet flowset parameters	Value
Maximum packet flow no-load latency	100 ms
Maximum period	500 ms
Priority assignment	Deadline monotonic
Route randomisation	Random XY/YX
Standard NoC topology	4x4
Enlarged NoC topology	8x8
Flowsets per data point	100
GA parameters	
Population size	100
Mutation individual task moving probability	0.3
Maximum generations	50

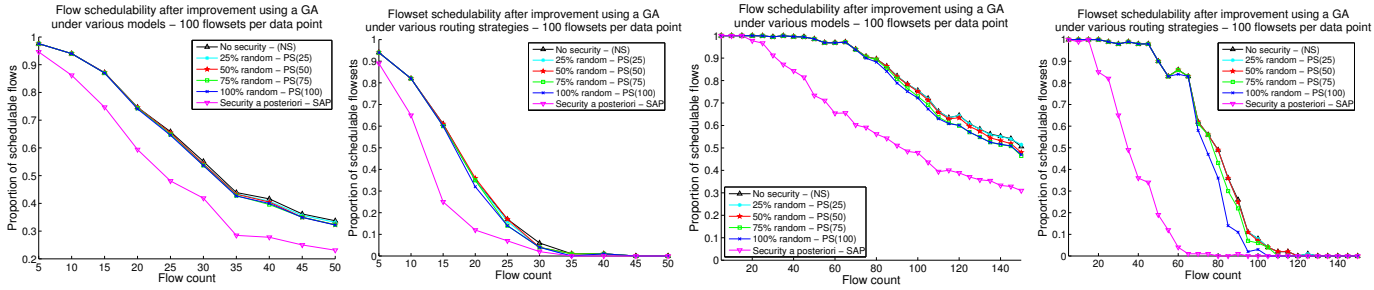

Fig. 3: Flow and flowset schedulability results for 4x4 (left) and 8x8 (right) NoCs

flowsets (i.e. a flowset is only considered schedulable if every flow within it is schedulable). The third and fourth plots also show percentage of schedulable flows and flowsets, but for larger flowsets mapped over a larger NoC (8x8). Those results show a greater separation between the NS and PS series after NoC evolution, due to the greater complexity of interference patterns over longer network paths when randomised routing is enabled.

B. Cycle-accurate simulation of route randomisation

This section uses simulation to evaluate the impact of route randomisation on the latency of an autonomous vehicle application described in [9].

The simulation framework used for this section is a cycle-accurate NoC model with support for priority preemption and virtual channels. This simulator has been extensively validated in our previous work, frequently being used as a baseline for results in latency and power analysis [10] [7].

1) Application Structure: The autonomous vehicle (AV) application consists of 38 communicating flows between a set of tasks that represent video processing, system monitoring and control for a robotic vehicle. Priorities are defined such that lower priority index values represent the highest priority transmissions. The priorities, data transmission rates, frequencies and deadlines of these application transmissions are as defined in [9], although a different mapping has been used in order to show the impact of routing protocols on a randomly selected mapping without artificial tuning to favour a particular routing protocol. The application has been mapped onto a 4x3 NoC, and the video resolution of the AV application video streams is 640x480. Since the application mapping is static and a single priority level is used per packet, a packet always travels between a fixed source-destination pair during the simulation.

2) Routing Alternatives: We compare a baseline XY routing with two randomised approaches. The first uses XY/YX, and traffic producers choose randomly upon injection whether a data packet will use XY or YX routing. Following this decision, a flag is set in the data packet to control the routing behaviour. As a result, the chosen routing algorithm (either XY or YX) is used throughout packet transmission.

The second approach is random west-first (RWF) routing. Its semi-deterministic mechanism requires the packet always to be forwarded towards the west when the destination node is

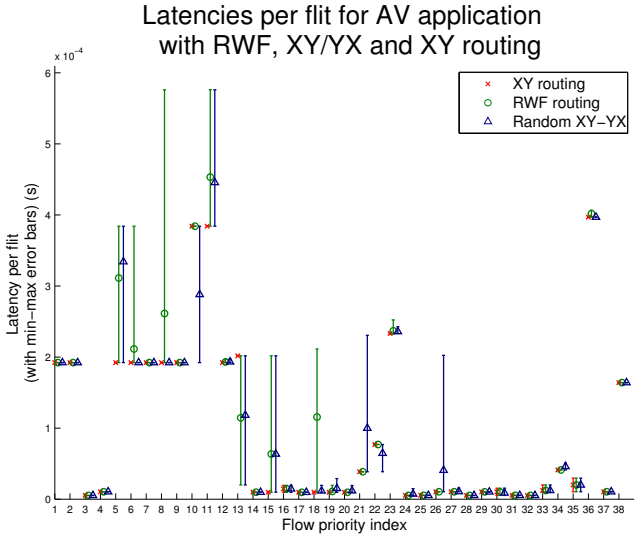

Fig. 4: Communication latency results for the randomised routing case on the AV application

west of the current node. However, any other destination port can be chosen randomly (east, north or south) as long as the direction taken is towards the destination. Therefore, the RWF approach permits a more diverse range of transmission paths than the XY/YX selection approach, providing more potential protection against side channel attacks.

3) Evaluation Results: The results are presented in Figure 4, illustrating the max-min-mean latencies for the randomised routing cases (XY/YX and RWF) versus the baseline. The results illustrate that routing randomisation typically increases the communication latencies for the majority of packets compared to fixed XY routing. This is particularly evident in the case of the packets with priority 8 under RWF routing, which experience an increased latency due to contention with other higher priority flows on some of the randomly chosen routes. In the XY/YX routing case, increased latency is also observed for the packets with priorities 21 and 26 in some cases. Interestingly, for some of the packet transmissions with priority 10 and 13, the use of randomised routing results in reduced latency in the best case, either by routing a higher priority packet so that it no longer causes interference, or routing the current packet around the interferer.

VI. Conclusions and Future Work

This paper has addressed the trade-off between security and hard real-time performance guarantees in Networks-on-Chip. It has proposed route randomisation as a way to increase NoC resilience against side-channel attacks, and has discussed a number of design alternatives for the randomisation approach. It then has proposed a schedulability test for applications running over a secure priority-preemptive NoCs using route randomisation. Finally, the paper identifies an optimisation pipeline which can be guided by the proposed schedulability test towards configurations that can achieve full schedulability while maximising the provided level of security. Extensive experimental work using 4x4 and 8x8 NoCs with random XY/YX routing running thousands of synthetically generated applications show the performance guarantees that can be achieved by the proposed approach at four different levels of security, compared against two baselines (no security, and full security applied a posteriori). Additional experiments with a realistic application running over 4x3 NoCs with random XY/YX and random west-first routing were performed with a cycle-accurate simulator, aiming to show the impact of route randomisation on latency variability, which in turn shows the increased resilience against side-channel attacks.

Since this is the first paper addressing the trade-off between security and hard real-time performance in NoCs, it had to make several assumptions to be able to attack the problem. Lifting some of those assumptions will certainly open new avenues of research, such as using different NoC arbitration mechanisms (e.g. TDM) or different route randomisation techniques (e.g. if randomised routes of subsequent releases of packets are never the same, a less pessimistic schedulability test can be used). Addressing those cases will require new schedulability tests, but could still reuse the proposed optimisation pipeline.

Acknowledgements

The research described in this paper is funded, in part, by the EPSRC grants MCC (EP/K011626/1) and MCCps (EP/P003664/1). No new primary data were created during this study.

References

[1] D. M. Ancajas, K. Chakraborty, and S. Roy. Fort-nocs: Mitigating the threat of a compromised noc. In *Proc Design Automation Conference (DAC)*, DAC '14, pages 158:1–158:6. ACM, 2014.

[2] D. Dasari, B. Nikolic, V. Nelis, and S. M. Petters. NoC Contention Analysis Using a Branch-and-prune Algorithm. *ACM Trans. Embed. Comput. Syst.*, 13(3s):113:1–113:26, March 2014.

[3] M. Dehyadgari, M. Nickray, A. Afzali-Kusha, and Z. Navabi. Evaluation of pseudo adaptive XY routing using an object oriented model for NOC. In *Int Conf on Microelectronics (ICM)*, pages 5 pp.–, December 2005.

[4] J.-P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin. Noc-centric security of reconfigurable soc. In *ACM/IEEE Int Symposium on Networks-on-Chip (NOCS)*, pages 223–232, May 2007.

[5] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano. Secure memory accesses on networks-on-chip. *IEEE Transactions on Computers*, 57(9):1216–1229, Sept 2008.

[6] C. J. Glass and L. M. Ni. The Turn Model for Adaptive Routing. In *Proc Annual International Symposium on Computer Architecture*, ISCA '92, pages 278–287, New York, NY, USA, 1992. ACM.

[7] J. Harbin and L. S. Indrusiak. Comparative performance evaluation of latency and link dynamic power consumption modelling algorithms in wormhole switching networks on chip. *Journal of Systems Architecture*, 63:33–47, February 2016.

[8] T. Iakymchuk, M. Nikodem, and K. Kepa. Temperature-based covert channel in FPGA systems. In *Int Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pages 1–7, June 2011.

[9] L. S. Indrusiak. End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration. *Journal of Systems Architecture*, 60(7):553–561, August 2014.

[10] L. S. Indrusiak, J. Harbin, and O. M. Santos. Fast Simulation of Networks-on-Chip with Priority-Preemptive Arbitration. *ACM Trans. Des. Autom. Electron. Syst.*, 20(4):56:1–56:22, September 2015.

[11] A. E. Kiasari, A. Jantsch, and Z. Lu. Mathematical Formalisms for Performance Evaluation of Networks-on-chip. *ACM Comput. Surv.*, 45(3):38:1–38:41, July 2013.

[12] S. Lukovic and N. Christianos. Enhancing network-on-chip components to support security of processing elements. In *Proc Workshop on Embedded Systems Security (WESS)*, pages 12:1–12:9. ACM, 2010.

[13] F. Moraes, N. Calazans, A. Mello, L. Moeller, and L. Ost. HERMES: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the VLSI Journal*, 38(1):69–93, October 2004.

[14] B. Nikolic, H. I. Ali, S. M. Petters, and L. M. Pinho. Are Virtual Channels the Bottleneck of Priority-aware Wormhole-switched NoC-based Many-cores? In *Proc of the 21st Int Conf on Real-Time Networks and Systems (RTNS)*, RTNS '13, pages 13–22, New York, NY, USA, 2013. ACM.

[15] D. Papp, Z. Ma, and L. Buttyan. Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In *Annual Conf on Privacy, Security and Trust (PST)*, pages 145–152. IEEE, 2015.

[16] S.G. Pestana, E. Rijpkema, A. Radulescu, K. Goossens, and O.P. Gangwal. Cost-performance trade-offs in networks on chip: a simulation-based approach. In *Proc. Design, Automation and Test in Europe Conference (DATE)*, pages 764–769, February 2004.

[17] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch. Methods for Fault Tolerance in Networks-on-chip. *ACM Comput. Surv.*, 46(1):8:1–8:38, July 2013.

[18] M. N. S. M. Sayuti and L. S. Indrusiak. Real-time low-power task mapping in networks-on-chip. In *IEEE Comp Soc Annual Symposium on VLSI (ISVLSI)*, pages 14–19, Aug 2013.

[19] M. Schoeberl. A Time-Triggered Network-on-Chip. In *Int Conf on Field Programmable Logic and Applications*, pages 377–382, 2007.

[20] J. Sepulveda, J.-P. Diguet, M. Strum, and G. Gogniat. Noc-based protection for soc time-driven attacks. *Embedded Systems Letters, IEEE*, 7(1):7–10, March 2015.

[21] J. Sepulveda, M. Soeken, D. Florez, J.-P. Diguet, and G. Gogniat. Dynamic noc buffer allocation for mpsoc timing side channel attack protection. In *IEEE Latin American Symposium on Circuits and Systems (LASCAS)*, pages 1–4. IEEE, 2016.

[22] Z. Shi and A. Burns. Real-Time Communication Analysis for On-Chip Networks with Wormhole Switching. In *ACM/IEEE Int Symposium on Networks-on-Chip (NOCS)*, pages 161–170, 2008.

[23] Z. Shi, A. Burns, and L. S. Indrusiak. Schedulability Analysis for Real Time On-Chip Communication with Wormhole Switching. *Int Journal of Embedded and Real-Time Communication Systems*, 1(2):1 – 22, June 2010.

[24] C. Silvano, M. Lajolo, and G. Palermo. *Low Power Networks-on-Chip*. Springer Science & Business Media, September 2010.

[25] H.M.G. Wassel, G. Ying, J.K. Oberg, T. Huffmire, R. Kastner, F.T. Chong, and T. Sherwood. Networks on chip with provable security properties. *IEEE Micro*, 34(3):57–68, May 2014.

[26] W. Yao and G.E. Suh. Efficient timing channel protection for on-chip networks. In *ACM/IEEE Int Symposium on Networks-on-Chip (NOCS)*, pages 142–151, May 2012.

[27] M.K. Yoon, S. Mohan, C.Y. Chen, and L. Sha. Taskshuffler: A schedule randomization protocol for obfuscation against timing inference attacks in real-time systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2016.