

This is a repository copy of *Side-Channel Protected MPSoC through Secure Real-Time Networks-on-Chip*.

White Rose Research Online URL for this paper:
<https://eprints.whiterose.ac.uk/145894/>

Version: Accepted Version

Article:

Soares Indrusiak, Leandro orcid.org/0000-0002-9938-2920, Harbin, James Robert orcid.org/0000-0002-6479-8600, Reinbrecht, Cezar et al. (1 more author) (2019) Side-Channel Protected MPSoC through Secure Real-Time Networks-on-Chip. *Microprocessors and Microsystems*. pp. 34-46. ISSN 0141-9331

<https://doi.org/10.1016/j.micpro.2019.04.004>

Reuse

This article is distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) licence. This licence only allows you to download this work and share it with others as long as you credit the authors, but you can't change the article in any way or use it commercially. More information and the full terms of the licence here: <https://creativecommons.org/licenses/>

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 Protected MPSoC through Secure Real-Time Networks-on-Chip

Leandro Soares Indrusiak^{a,*}, James Harbin^a, Cezar Reinbrecht^b, Johanna Sepúlveda^c

^a*Department of Computer Science, University of York, UK*

^b*Institute of Informatics - Federal University of Rio Grande do Sul, Brazil*

^c*Institute for Security in Information Technology, Technical University Munich, Germany*

Abstract

The integration of Multi-Processors System-on-Chip (MPSoCs) into the Internet-of-Things (IoT) context brings new opportunities, but also represent risks. Tight real-time constraints and security requirements should be considered simultaneously when designing MPSoCs. Network-on-Chip (NoCs) are specially critical when meeting these two conflicting characteristics. For instance the NoC design has a huge influence in the security of the system. A vital threat to system security are so-called side-channel attacks based on the NoC communication observations. To this end, we propose a NoC security mechanism suitable for hard real-time systems, in which schedulability is a vital design requirement. We present three contributions. First, we show the impact of the NoC routing in the security of the system. Second, we propose a packet route randomisation mechanism to increase NoC resilience against side-channel attacks. Third, using an evolutionary optimisation approach, we effectively apply route randomisation while controlling its impact on hard real-time performance guarantees. Extensive experimental evidence based on analytical and simulation models supports our findings.

Keywords: Side Channel, MPSoC, NoC, Routing

*I am corresponding author

Email address: leandro.indrusiak@york.ac.uk (Leandro Soares Indrusiak)

1. Introduction

The comprehensive use of Internet-of-Things (IoT) will be the driver of digitization in all domains, e.g. industry automation, automotive, avionics, and healthcare. Increasingly complex and powerful Multi-processor Systems-on-Chips (MSoCs) connected through a 5G network, form the basis of the IoT. The semiconductor industry has been challenged to meet the tight and demanding requirements of such applications. These requirements include low power, tight latencies and high throughput. When developing systems for these hyper-connected environments, real-time constraints and security are necessary considerations.

Network-on-Chips (NoCs) are the heart of the MPSoC. NoCs are shared by different communication flows characterized by a wide set of requirements, which include performance, reliability or security. Their key role in the MPSoC operation turns the NoC design into a critical task. Over the past decades, a significant amount of work has addressed the trade-offs between performance and other secondary objectives such as energy [1], fault-tolerance [2], and chip area [3]. Less work has addressed such trade-offs in NoCs with hard real-time constraints, with some inroads towards improving energy [4] and area efficiency (by optimising buffering in virtual channels [5]) while meeting deadlines of all packets even in the worst-case scenario. While hard real-time applications impose strict latency requirements on the NoC, the impact on security has been not addressed before. Hard real-time mechanisms may impact the MPSoC security.

MPSoCs operating in the context of IoT usually integrate cryptographic hardware cores for confidentiality and authentication security services. However, these components are prone to implementation attacks. During the operation of a cryptographic core, the secret key may passively be revealed through so-called side-channels. Classical side-channels include the measurement of the execution time, power-consumption and electromagnetic (EM) radiation of the cryptographic IP core [6]. The interconnection of MPSoCs operating in the Internet-of-Things permits possible timing side-channel attacks that emerge from sharing resources on the MPSoC.

Cache hierarchies and NoC are a common target in timing side-channel attacks. In general, NoC communication can be exploited to optimize cache attacks, as demonstrated in [7] and [8]. By detecting the communication patterns of the sensitive traffic (e.g., volume and communication rate) an attacker is able to trigger cache attacks in the most vulnerable point of the

38 encryption process. Thus the NoC communication collision of malicious and
39 sensitive traffic can potentially compromise the security of the complete em-
40 bedded system. Many mechanisms have been proposed to improve NoC secu-
41 rity and many more will certainly be developed in the coming years. However,
42 most of such mechanisms impose performance overheads, and therefore can
43 potentially jeopardise the ability of the NoC to provide real-time guarantees.
44 In this paper, security is used as a driver to optimise hard real-time NoC
45 design. The hard real-time NoCs constraints must be always guaranteed.
46 Our approach is based on the randomisation of packet routes. By randomly
47 changing the route of every packet injected into the NoC, we can introduce
48 random effects to all side-channels of interest, such as packet timing, energy
49 dissipation, temperature and electromagnetic emissions. In this paper, we
50 concentrate on a threat model based on packet timing.

51 This paper extends our earlier conference paper work upon security through
52 routing randomisation in NoCs [9]. In summary, the contributions of our to-
53 tal work upon this idea are:

- 54 • Provide a realistic motivation for our work by specifying case studies;
55 a side-channel attack on AES encryption and how it may arise in an
56 IoT context due to the interaction between secure and malicious down-
57 loaded code communicating over a shared NoC. A novel case study
58 involving an autonomous vehicle is added in this paper, over that pre-
59 sented in [9].
- 60 • Present an experiment performed on a NoC hardware platform in order
61 to motivate route randomisation as a viable approach for improving
62 security - the current publication adds this upon the earlier work in [9]
- 63 • Define a schedulability analysis for determining the worst-case end-to-
64 end latency in the case of randomised routing
- 65 • Present a GA optimisation process which uses task mapping to main-
66 tain schedulability assessed by this analysis, while permitting improv-
67 ing security by allowing flows to use randomised routing
- 68 • Assess via simulation the impact of randomised routing strategies upon
69 empirically measured latency in a real application case study

70 The rest of the paper is organized as follows. Section 2 presents the
71 description of the MPSoC and the security requirements. It includes the NoC

72 timing attack and the threat model. Section 3 presents the most relevant NoC
73 security mechanisms and the types of security mechanisms to prevent the
74 MPSoC attacks. Performance overheads and resource usage are discussed,
75 highlighting the need for the contributions of this paper. Section 4 we identify
76 techniques that support NoC designers in improving NoC resilience against
77 side-channel attacks while still maintaining full system schedulability. The
78 paper is closed with extensive experimental work based on schedulability
79 analysis and simulation in Section 5, and with a summary of our findings.

80 **2. Multi-Processor System-on-Chip and security requirements**

81 MPSoCs are prone to attacks. In this section the MPSoC architecture
82 and operation are described. These concepts will be used to understand the
83 threat model for the NoC-based communication side-channel vulnerability.

84 *2.1. MPSoC / NoC Architectural Description*

85 While the contribution of this paper can be applied to a large variety of
86 NoC architectures, we believe it is easier to explain it with the help of a con-
87 crete architecture. We assume a NoC architecture with a 2D-mesh topology
88 and wormhole switching protocol, because such features are commonly used
89 in embedded systems for their simplicity and moderate resource overheads.

- 90 • In a 2D-mesh topology, every core is connected to a NoC switch via a
91 network interface (NI), which is responsible for packetising and depack-
92 etising data, and controlling the injection of packets into the network.
93 The regularity of such a topology is attractive because it simplifies
94 packet routing, and facilitates chip floorplanning, placement and rout-
95 ing.
- 96 • The use of wormhole switching protocols allows packets to be gradually
97 sent over the NoC in smaller units called flits. Once a flit is received
98 by a switch, it can be forwarded to the next switch down the packet
99 route as long as that switch has sufficient buffering to hold it. This
100 means that at any given time a packet could have its flits temporarily
101 stored by multiple switches, so each of them are not required to hold
102 a complete packet, thus reducing the overall buffering requirements of
103 the NoC.

- 104 • There is a downside to this choice of topology and switching protocol,
 105 which is the difficulty in predicting packet latencies. Since a packet
 106 can be simultaneously occupying multiple NoC buffers and links, there
 107 is a significant amount of competition for resources throughout the
 108 NoC at all times. The wide variety of interference patterns makes it
 109 hard to predict how long it takes for a packet to reach its destination.
 110 Different resource arbitration policies can make such predictions more
 111 or less difficult, especially in the case of hard real-time NoCs when an
 112 upper-bound worst-case latency is needed.

- 113 • Previous work has considered NoC arbitration based on packet prior-
 114 ity [10], time multiplexing [11] and round robin [12], and has devised
 115 analytical models that can be used to find latency upper-bounds for
 116 packet flows transmitted over such NoCs [13]. Any of those approaches
 117 could be used in this paper, and we chose a priority-arbitrated NoC
 118 because of its ability to provide upper-bound latency guarantees that
 119 are customisable to different levels of packet urgency while allowing for
 120 high NoC link utilisation [14].

- 121 • The general architecture of the network on chip described in previous
 122 bullet points explains the data communications. However, when con-
 123 sidering security implications it is important to describe the enclosing
 124 context of the MPSoC in which the NoC exists. MPSoCs are tile-based
 125 structures which are flexible enough to meet a variety of application re-
 126 quirements. Each tile is either composed of a single IP core or a cluster
 127 of IP cores. Data is exchanged over a NoC between tiles. In order to
 128 increase the performance, current MPSoCs employ two main strategies:
 129 i) memory hierarchies, where several levels of cache (e.g. L1 to L3) and
 130 a set of DRAMs are integrated; and ii) resource sharing, where different
 131 applications are split and mapped onto the MPSoC resources.

132 *2.2. Threat Model and Timing Side Channel Attacks*

133 In this paper, we assume that the NoC and its interfaces to the cores
 134 are secure. We also assume that secure tasks execute in secure cores (i.e.
 135 cores that do not allow the execution of unsecured tasks). For this threat
 136 model, we assume that the NoC communicates sensitive information between
 137 two secure tasks, which we refer as the sensitive communication. We then
 138 assume an adversary that has knowledge about the NoC architecture, about

139 the mapping of secure tasks to (secure) NoC cores, and is able to gain control
140 of at most two non-secure NoC cores.

141 A successful attack happens when the adversary, which has taken control
142 of two non-secured processors, is able to obtain information about the sensi-
143 tive traffic. In such attack, the adversary injects packets to the NoC in order
144 to collide with the sensitive traffic. These two types of traffic (malicious and
145 sensitive) collide inside the router, that is, they compete for the same output
146 port resource. As a consequence of the malicious traffic, delays in the com-
147 munication are caused and thus the malicious packets transmission is also
148 delayed. At an endpoint at the other non-secured core, the adversary is able
149 to measure the latency of their malicious traffic and infer how many collisions
150 with the sensitive traffic occurred. The resulting collisions leak information
151 regarding sensitive communication flows. Note that the router is not nec-
152 essarily malicious and that no any information embedded into the sensitive
153 packet is required to perform the attack. The latency interference imposed
154 by the sensitive communication over the malicious low priority traffic can
155 provide the attacker with valuable information about the timing, frequency
156 and volume of the secure communication.

157 This threat model is not new, and its variations have also been used in
158 best-effort NoC-based systems by [15], [16], [7]. The timing nature of the
159 threat is also the same used in hard real-time uniprocessor systems by [17].

160 *2.3. Security of MPSoCs Case Studies*

161 In order to motivate the work and provide a concrete example of the
162 security consequences of a timing attack, we now present two illustrative
163 interrelated case studies. In both of them, timing attacks focused upon NoC
164 communication can lead to negative consequences for a trusted application,
165 even if the endpoint cores are fully secured against intrusion. The first
166 focuses upon an AES encryption scenario, and the second focuses upon an
167 autonomous vehicle. Note that these case studies are illustrative and apply
168 to an example system; while expected to be representative of real security
169 concerns in a system, they are not strictly based upon a particular hardware
170 implementation or the simulation case study evaluated later in the paper.

171 *2.3.1. AES NoC Timing Attack Case Study*

172 MPSoCs operating in the context of IoT usually integrate security fea-
173 tures such as cryptographic hardware cores for providing security services

174 (confidentiality, authentication and integrity). The symmetric key encryption
 175 algorithm Advanced Encryption Standard (AES) is widely used to im-
 176 plement security functions in several MPSoCs. AES encrypts 128 bits of
 177 data with key lengths of 128 bits using 10 rounds. AES operates iteratively
 178 data organized as a 4x4 state matrix. Each round is composed of four round
 179 operations: AddRoundKey (XORing the state with the current round key),
 180 SubByte (byte substitution), ShiftRow (byte transposition) and MixColumn
 181 (matrix multiplication). In order to speedup the execution of AES, trans-
 182 formation tables (T-tables) are used. T-table AES reduces the SubByte,
 183 ShiftRow and MixColumn operations to four lookup tables whose entries
 184 are simply XOR'ed [7]. The AES functionality is integrated in the MPSoC
 185 through a security co-processor, an IP core with a private L1 cache. In such
 186 scenario T-tables are stored along the different cache hierarchies of the MP-
 187 SoC. The vulnerability exploited by attackers is that T-tables are accessed
 188 depending on the secret key. Such attacks are know as cache attacks. A
 189 deeper description of the access-based cache attacks for MPSoCs is given in
 190 [7] and [6]. The weakest point of the AES operation is at the end of the
 191 first round. The NoC timing attack detects the end of the first round of
 192 the AES, thus allowing an attacker to trigger in the cache attack in the best
 193 moment, when noise generated from other cache accesses performed during
 194 the encryption are avoided.

195 Fig. 1 presents the NoC-based MPSoC architecture. It integrates 16 IP
 196 cores (IP_0 to IP_{15}) which exchange communication through a mesh-based
 197 NoC. The integration of MPSoCs into an IoT context may permit remote
 198 applications downloaded from the Internet to be stored into external mem-
 199 ories and mapped into the MPSoCs. These applications are vulnerable to
 200 attacks and can be tampered with by an attacker. When mapped into the
 201 system resources, attackers are able to control packet injection into the NoC.
 202 In Fig. 1, the attacker has infected the IP_1 and controls the traffic injection.
 203 IP_3 represents the AES cryptoprocessor, where the T-Tables are stored in
 204 the shared cache IP_0 .

205 The goal of the NoC timing attack is to identify the end of the first AES
 206 round. The attacks is performed in 7 steps, as shown in Fig. 1. In step
 207 (1) and (2), the attacker triggers first an AES encryption, then continues
 208 to frequently inject packets into the NoC. The throughput of the infected
 209 core is monitored by the attacker. Step (3) shows the execution of the AES
 210 encryption by the IP_{13} . During the AES encryption, the value stored in the
 211 T-tables should be retrieved, thus a read request to IP_0 is performed in step

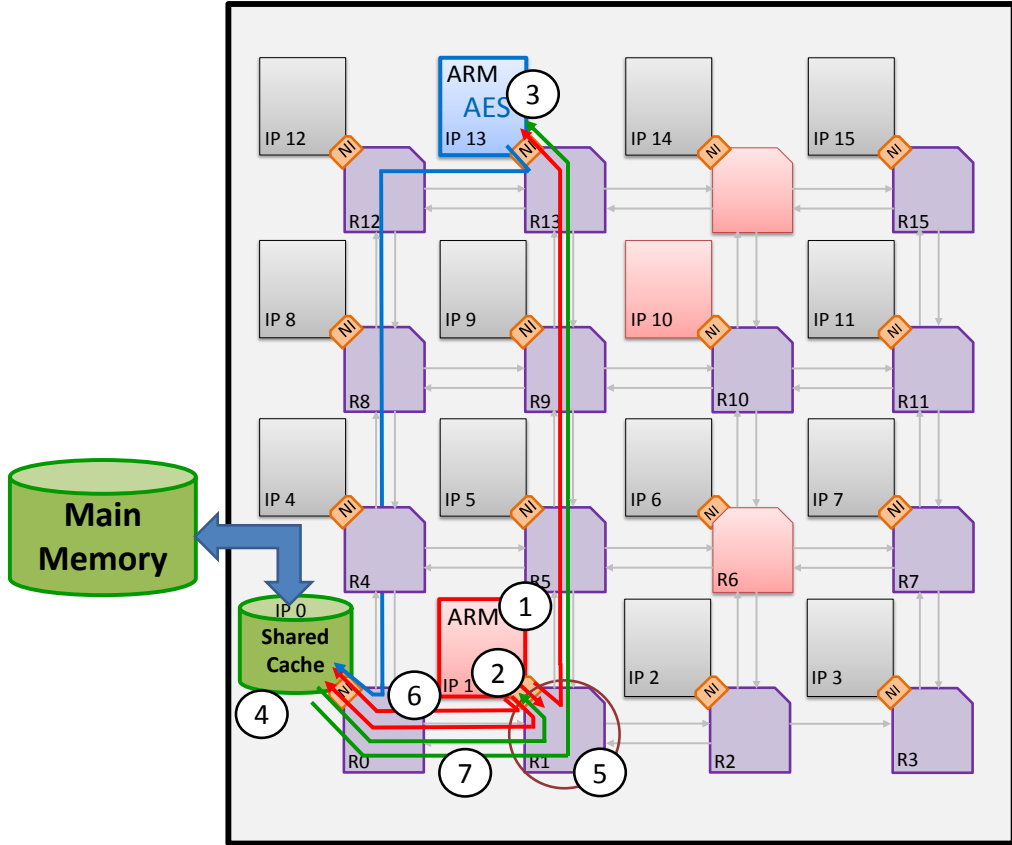


Figure 1: Description of the NoC timing attack to NoC-based MPSoCs

212 (4). As a result, a big packet is retrieved in step (5). The communication
 213 collision in R_1 between the infected traffic and the sensitive traffic causes a
 214 degradation in the throughput of the attacker. This is illustrated in Figure
 215 2 which illustrates the timing behaviour at the router R_1 , for an attacker
 216 injecting a packet coincident with IP_0 responding. The attacker can measure
 217 the time taken between injecting its malicious packets and its completion.
 218 Since it knows its basic latency; the time taken to deliver this packet without
 219 load, it can determine the excess latency by subtraction. This provides an
 220 estimate of the response size.

221 This triggers a cache attack, where the attacker perform a read request to
 222 the shared memory in IP_0 in step (6). As is [7] and [6], by reading the shared
 223 cache in step (7), the attacker can identify the memory sets accessed due the

224 AES encryption. As a result a key candidate is obtained. This process is
 225 performed multiple times until the key is found.

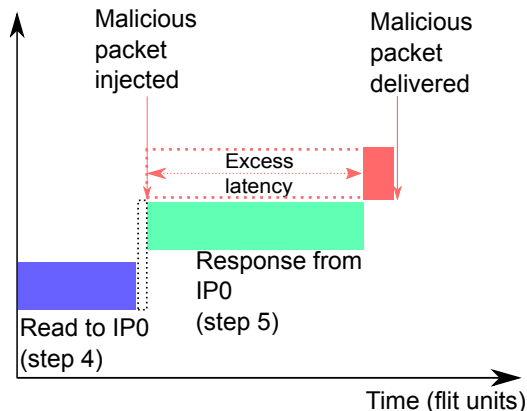


Figure 2: Timing diagram demonstrating the attacker measuring the latency

226 *2.3.2. Autonomous Vehicle Case Study*

227 A similar attack case study may apply in the case of an autonomous
 228 vehicle. The integration of heterogeneous software and MPSoCs for an au-
 229 tonomous vehicle could incorporate components derived independantly by
 230 different manufacturers. As mechanical systems and engine control units
 231 (ECUs) become more complex and integrated, the timing of messages for
 232 tasks such as engine control and emissions control becomes more critical [18].
 233 Simple attacks on timing in an AV system could include using malicious cores
 234 to inject additional traffic that delays sensor readings from reaching external
 235 buses such as CAN at the expected times [18]. If the MPSoCs in the AV
 236 uses AES encryption, then this would be vulnerable to the attack described
 237 in the previous section 2.3.1.

238 However, it is possible to imagine a more subtle attack. Take for exam-
 239 ple the requirement in autonomous vehicles for computer vision algorithms
 240 to analyse camera data and identify particular targets. It is possible that
 241 manufacturers of these systems would not wish to reveal their algorithm op-
 242 eration, either from competitors, or to not reveal what targets their system
 243 is scanning for. In the case of a potential detection of an object requiring
 244 additional processing, the secured tasks may need to transmit more data
 245 amongsts themselves, or send requests for additional data from sensors. The

246 potential motivation of an attacker would be to detect these communica-
247 tions occurring, and thus infer information about the AV system’s goal or
248 techniques of operation.

249 If the computer vision tasks were located upon secured cores, then the
250 attacker would not be able to access these tasks directly. However, by inject-
251 ing low priority traffic into the onboard NoC and observing the delays these
252 low priority communications experience, the attacker would be able to infer
253 increased communication lengths or frequencies by the secured tasks, leading
254 to potential leakage of the AV system purpose or operation.

255 **3. Related Work**

256 Multiprocessor embedded systems are target of attacks by means of ma-
257 licious hardware or software [19]. Hardware-based attacks depend on design-
258 time access to the system, which is then modified in a way that can be
259 exploited during operation (e.g. by adding hardware able to leak informa-
260 tion by changing chip temperature [20]). Software-based attacks are the most
261 common cause of security incidents in such types of systems [21], and are car-
262 ried out by malicious software installed at design time or after deployment.

263 NoC-based systems have been shown to be vulnerable to a variety of
264 attacks, both hardware and software-based. Active NoC attacks, such as
265 code injection [22], malware [23] and control hijacking [24], or passive NoC
266 attacks, such as side-channel exploitation, can be used to read sensitive com-
267 munications, modify the system behaviour or prevent correct NoC operation.
268 NoCs are especially vulnerable to side-channel attacks that exploit traffic in-
269 terference as timing channels [15] [25]. The shared nature of NoCs can be
270 exploited by an attacker to obtain sensitive information. By forcing traffic
271 collision with sensitive packet flows, an attacker can observe the throughput
272 variations and infer sensitive data, as shown in [15] [25] [26].

273 Security-enhancing mechanisms have been added to NoC platforms to
274 provide authentication [27], access control [23], integrity [28], and confiden-
275 tiality services [29]. By monitoring and controlling the data exchange inside
276 the chip, NoCs can detect and avoid attacks.

277 Firewall-based and crypto-based techniques integrated at the network in-
278 terface are the most commonly used approaches against active NoC attacks
279 over the past decade [23] [30]. Firewalls implement authentication, access
280 control and integrity services by means of traffic matching with a security

281 table. Authorized transactions are allowed and injected to the NoC, other-
282 wise they are denied and thus dropped. Crypto-based NoCs implement the
283 confidentiality service by creating a shared secret among the sensitive cores
284 and perform the encoded data exchange. While achieving desirable secu-
285 rity enhancements, such approaches have an unpredictable impact upon the
286 performance of the NoC and thus the overall system.

287 PhaseNoC [31] focuses upon traffic isolation, which provides separation
288 of traffic in adjacent domains and therefore potential reductions in the attack
289 surface for timing attacks. However, such TDM (time-division multiplexing)
290 static techniques reduce performance in the case of dynamic traffic arrival,
291 so the authors provide a scheme which can opportunistically steal bandwidth
292 between traffic classes. This scheme does permit potential timing attacks via
293 leakage between the traffic classes.

294 Firewalls and crypto-based NoCs are the state-of-the-art in NoC security,
295 but they are not able to protect the system against passive NoC attacks.
296 Randomised arbitration [25], virtual channel allocation [16] and routing [26]
297 have been investigated and evaluated as countermeasures against timing at-
298 tacks. By randomising the characteristics of sensitive packet flows, it is
299 possible to break the correlation between the traffic characteristics (e.g. vol-
300 ume and access patterns) and the sensitive data thus avoiding information
301 leakage. Among those mechanisms, random routing has achieved the best
302 levels of security enhancement with the lowest energy and area overhead [26].
303 By spreading sensitive traffic over the NoC, the spatial distribution makes
304 it harder for compromised cores or external attackers to gather sufficient
305 side-channel information to infer correlations with sensitive data.

306 Similarly to firewalls and crypto-based approaches, the focus of randomi-
307 sation approaches is to increase security and none of the works in the state-
308 of-the-art consider the performance requirements of the applications. In this
309 paper, we argue that NoCs supporting real-time applications require a care-
310 ful balance of a trade-off between security and performance. In most cases,
311 we envisage that the level of security will be constrained by the NoC's ability
312 to support attack countermeasures while at the same time ensuring perfor-
313 mance guarantees to the application. By providing a test to evaluate whether
314 performance guarantees can hold under a specific side-channel attack coun-
315 termeasure (namely route randomisation) we aim for a better balance of
316 performance guarantees, resource usage and security trade-offs.

317 *3.1. System Model*

318 To increase NoC resilience against side-channel attacks while providing
 319 hard real-time guarantees to the application tasks running on it, we must
 320 make assumptions about the application behaviour such as upper-bounds
 321 on resource usage by every application task and packet. In this paper, we
 322 follow the well-known and widely used sporadic task model, which makes
 323 assumptions about the worst-case execution time (WCET) of all tasks and
 324 their shortest inter-arrival interval (i.e. their period). Since we are concerned
 325 about NoC communications, we follow an extension of the sporadic task
 326 model that considers that tasks inject packets to the NoC only after their
 327 execution completes, and that the maximum packet size is known [14].

328 Thus, a hard real-time application Γ comprises n real-time tasks such as
 329 $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$. Each task τ_i is a 6-tuple $\tau_i = (C_i, T_i, D_i, J_i, P_i, \{\phi_i\})$
 330 indicating respectively its worst case computation time, period, deadline,
 331 release jitter and priority. The sixth element of the tuple is an extension to
 332 the sporadic task model proposed by [14], and represents the communication
 333 packets sent by τ_i at the end of its execution. Each packet ϕ_i is defined as a
 334 3-tuple $\phi_i = (\tau_d, Z_i, K_i)$ representing its destination task, size and maximum
 335 release jitter. In this paper, we assume for simplicity that a single packet
 336 is released at the end of each execution of each task, but the contributions
 337 presented here can be generalised for any number of released packets.

338 Such applications are executed over a NoC platform like the one described
 339 in subsection 2.1 above. We model such a platform as a set of cores Π
 340 $= \{\pi_a, \pi_b, \dots, \pi_z\}$, a set of switches $\Xi = \{\xi_1, \xi_2, \dots, \xi_m\}$, and a set of unidirec-
 341 tional links $\Lambda = \{\lambda_{a1}, \lambda_{1a}, \lambda_{12}, \lambda_{21}, \dots, \lambda_{zm}, \lambda_{mz}\}$. We also model the mapping
 342 of tasks to cores with the function $map(\tau_i) = \pi_a$.

343 The routing of packets over the NoC can be modelled by the function
 344 $route(\pi_a, \pi_b) = \{\lambda_{a1}, \lambda_{12}, \dots, \lambda_{mb}\}$, denoting the subset of Λ used to transfer
 345 packets from core π_a to core π_b . We can then extend the function map to also
 346 model the mapping of a packet to its route: $map(\phi_i) = route(map(\tau_i), map(\tau_d))$.

347 With the knowledge of the NoC architectural characteristics such as the
 348 latency to cross a link or to route a packet header, and with the knowledge
 349 of the length of a packet's route (i.e. its hop count, or $|route(\pi_a, \pi_b)|$ as
 350 expressed in [14]), it is possible to calculate the no-load latency L_i of every
 351 packet ϕ_i : the time it takes to completely cross the NoC from its source to
 352 destination without any interference or contention from other packets. For
 353 the NoC described in subsection 2.1, and for most commercial and academic

354 NoCs, the no-load latency of a packet can be deterministically obtained, and
355 will not change if its route and the NoC operation frequency do not change.

356 4. NoC Routing Randomisation

357 4.1. Overview Of Route Randomisation

358 By using a route randomisation approach, it is possible to prevent the
359 adversary from obtaining accurate information about the sensitive commu-
360 nication. Because not every packet of the secure communication will interfere
361 on the malicious flows injected by the attacker, the information about tim-
362 ing, frequency and volume they can obtain will be less accurate, which as a
363 consequence increases the resilience of the NoC against the threat. There are
364 many ways to introduce route randomisation in NoCs, and we will discuss
365 our design decisions in subsection 4.3.

366 Figure 1 and Fig. 3 show examples of the described threat model in
367 Section 2.2. Fig. 3 shows an adversary controlling cores F and G, and using
368 a malicious packet flow (shown as a purple dashed line) to infer data about
369 a sensitive communication between secure cores C and E (shown as a red
370 dotted line, representing the case of a NoC with deterministic XY routing).
371 In the case of a NoC with randomised routing, all routes between C and E
372 will be used (red dashed and dotted lines), preventing the adversary from
373 inspecting the complete sensitive communication.

374 4.2. Motivation Experiment

375 Different NoC parameters impact the security of the system. Routing
376 may have a huge impact on the success of the attack. In order to show this
377 statement, we performed an experiment on a FPGA-based prototype of an
378 MPSoC as shown in [8]. It is composed of 16 NIOS II IP cores, each with
379 a 32 kB private L1 cache. The shared L2 cache is of 256 kB size and it is
380 inclusive of L1. All of them have a cache line size of 16 bytes. The MPSoC
381 structure is similar to Fig. 1. However the position of the AES, shared caches
382 and attackers are modified. In the experiment the infected IP could be placed
383 in six location of the MPSoC: i) linked to the east port of the router 1 (R1
384 E); ii) linked to the north port of the router 1 (R1 N); iii) linked to the east
385 port of the router 4 (R4 E); iv) linked to the north port of the router 4 (R4
386 N); v) linked to the north port of the router 6 (R6 N); and vi) linked to the
387 east port of the router 9 (R9 E).

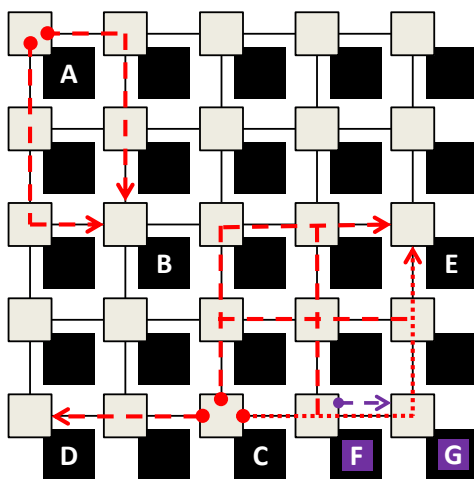


Figure 3: Threat model, and examples of route randomisation with pseudo-adaptive XY (from A to B) and west-first (from C to D and C to E) algorithms

388 The sensitive path is defined by the communication channel between the
389 IP 0 and IP 10. Three different dimension ordered routing strategies were
390 used to commute packets in the NoC: i) XY, which limits all turns to y-
391 dimension until the x-dimension is reached; ii) XY-YX, which alternates
392 randomly the XY and YX routing algorithms; and iii) West First (WF),
393 which restricts turns to the west. The detection rate of the sensitive pack-
394 ets for each configuration was evaluated. The observation points were the
395 output ports shared with the sensitive traffic. Results are shown in Fig. 4).
396 For the deterministic XY, the attackers that intersected the sensitive path
397 were able to detect all the packets. However, when XY and YX were used
398 randomly, the effectiveness of the attacker varies according to the amount of
399 traffic that collides with the attacker traffic. Since only two paths were pos-
400 sible, an attacker was not able to detect all sensitive traffic. The best results
401 were achieved for attacks on the east port of the router 9 (R9 E) and the north
402 port of the router 4 (R4 N). However, such results are highly dependent on
403 the routing algorithm. In the last scenario, the West-first algorithm has six
404 route possibilities. Hence, the efficiency of the attack was very low, since
405 the messages became spread in the NoC through different routes. This moti-
406 vates work on improving the security of the MPSoC via route randomisation.
407 However, in viable real-time systems, security must be considered alongside

408 end-to-end latency constraints.

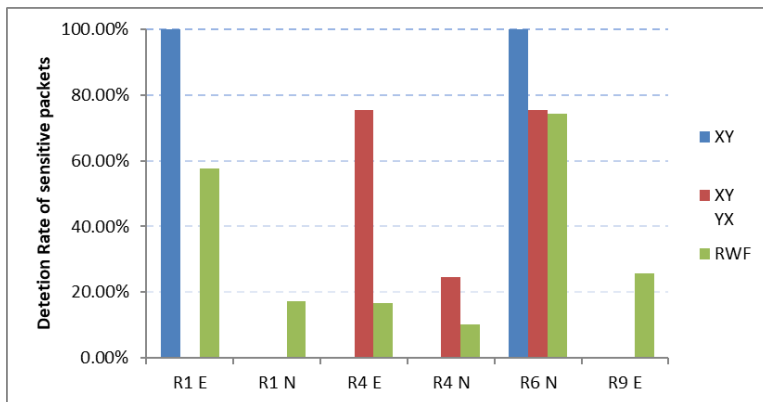


Figure 4: The detection rate of sensitive packets under different attacker IP locations and routing strategies

409 *4.3. Design Choices and Constraints*

410 There are many design choices related to packet routing in different NoC
 411 architectures [32]. As expected, those choices also define whether and how
 412 route randomisation can be achieved. For example, some NoC architectures
 413 use deterministic routing [33], meaning that there is only one possible route
 414 between a source and a destination, effectively preventing the approach pro-
 415 posed here. Among NoCs supporting dynamic or adaptive routing, which are
 416 the ones we target, there is a key design choice affecting the randomisation
 417 approach: source or distributed routing.

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

428 In NoCs with distributed routing, the next-hop decision is made by each
 429 switch individually. Typically, they have far less resources than the cores (and

430 often than the NIs), so the routing decisions are based on simple rules related
431 to the relative position of the destination core with regards to the switch
432 holding the packet header (e.g. pseudo-adaptive XY [34], turn model [35]).
433 In those cases, it is only possible to randomly choose from a predefined
434 subset of all possible routes. For instance, pseudo-adaptive XY switches can
435 only randomly choose between two routes between a source and a destination
436 (e.g. routes between cores A and B in Figure 3). Switches implementing turn
437 model routing may have a larger number of alternative routes to randomly
438 choose from in most cases, but must behave deterministically for some specific
439 cases. Figure 3 shows two routes created by a west-first turn model: packets
440 between core C and D have only one possible route, as the destination is
441 located on the west of the source, while packets from core C to E can take a
442 variety of possible routes.

443 In both source and distributed routing, the NoC component making ran-
444 dom decisions must have access to a source of random data, such as a pseudo-
445 random number generator (PRNG, generated by a deterministic algorithm)
446 or a true random number generator (TRNG, often generated out of low level
447 noise signals). Such sources can have significant hardware overhead, thus
448 favouring source routing because of the low area constraints for NoC switches.
449 For the route randomisation approaches reviewed above, however, overheads
450 should be minimal in either case as they only require random sources with
451 one-bit output.

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

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

466 Deadlock-free packet communication is a critical characteristic for NoCs.
467 This can be achieved at the link arbitration layer, e.g. with priority-preemptive

468 virtual channels [14], or at the network layer by restricting the possible turns
 469 of the routing algorithm (either in source or in distributed routing). In NoCs
 470 that ensure deadlock-freeness at the network layer, special care must be taken
 471 by the route randomisation approach to avoid introducing turns that can lead
 472 to deadlocks.

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

487 4.4. *Schedulability Analysis*

488 Schedulability analysis for a set of sporadic packets transferred over a
 489 priority-preemptive wormhole switching NoC was presented in [36]. A set of
 490 packets is deemed schedulable if the worst-case latency of each packet is less
 491 than their deadline. By coupling that analysis with classical response time
 492 analysis for uniprocessor fixed-priority scheduling, an end-to-end schedula-
 493 bility analysis for that type of NoC was proposed in [14], considering the
 494 worst-case response times of tasks and the worst-case latency of the packets
 495 they generate. Both the original analysis from [36] and the end-to-end ex-
 496 tension from [14] assume static routing, so a different formulation is needed
 497 before it can be used for the purpose of this paper. First, we review those
 498 formulations, but using the notation described in subsection 3.1.

499 According to [36], the worst-case latency S_i of a packet ϕ_i can be obtained
 500 from Equation 1. This equation is defined recursively and iterated until a
 501 stable fixed point is discovered.

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

502 The set **interf**(i) is the set of higher priority packets ϕ_j whose route shares
 503 at least one link with the route of ϕ_i and therefore can interfere with it.
 504 Precisely, **interf**(i) = $\{\phi_j \in \phi : \text{map}(\phi_i) \cap \text{map}(\phi_j) \neq \emptyset\}$. The two terms
 505 K_j and K_j^I denote respectively the maximum release jitter of the interfering
 506 packet ϕ_j and its maximum indirect interference jitter. As shown in [14],
 507 K_j is equal to the worst case response time R_j of task τ_j which produces ϕ_j ,
 508 assuming that ϕ_j will be released immediately after the end of τ_j 's execution.
 509 R_j can be calculated using uniprocessor response time analysis, considering
 510 the type of task scheduling by the operating system at each core (e.g. priority-
 511 preemptive). And as shown in [36], the indirect interference jitter K_j^I can be
 512 bound by $S_j - L_j$.

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

522 A general analysis where all packets could potentially have randomised
 523 routes is more complex: all possible routes of a packet would have to be tested
 524 with all possible routes of all other packets before the worst case could be
 525 found. Furthermore, if one cannot make probabilistic assumptions on the
 526 randomisation approach, pathological cases must also be taken into account
 527 (e.g. the same route could be chosen again and again for a single packet over
 528 a long period of time, even though that is very unlikely).

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

540 To calculate worst-case latencies for the general problem where all pack-
 541 ets could have randomised routes, we define the set $\mathbf{interf}_r(i)$ as the set
 542 of higher priority packets ϕ_j who could, with any of their possible routes,
 543 interfere with any of the possible routes of the packet of interest ϕ_i . To
 544 precisely define that set, we must first define a new function $route_r(\pi_a, \pi_b)$
 545 $= \{\lambda_{a1}, \lambda_{12}, \lambda_{13}, \lambda_{14}, \dots, \lambda_{mb}\}$, denoting the subset of Λ that contains all the
 546 links that could be part of any of the routes that could be randomly chosen
 547 to transfer packets from core π_a to core π_b , and a new function $map_r(\phi_i)$
 548 $= route_r(map(\tau_i), map(\tau_d))$. Then, $\mathbf{interf}_r(i) = \{\phi_j \in \phi : map_r(\phi_i) \cap$
 549 $map_r(\phi_j) \neq \emptyset\}$.

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

553 4.5. Optimising the Performance-Security Trade-off

554 The schedulability analysis proposed in the previous subsection can only
 555 be used to test whether a particular randomised NoC configuration can meet
 556 the hard real-time constraints of an application. It offers no alternatives
 557 in case of negative results, i.e. when performance constraints are not met.
 558 In this subsection we show how the schedulability test can be exploited as
 559 a fitness function in a design space exploration process. Similarly to [4]
 560 and [14], we follow an evolutionary approach to navigate over a key part
 561 of the design space: task-core mapping. By changing that mapping, it is
 562 possible to achieve fine-grained improvements on schedulability of tasks over
 563 cores and packet flows over NoC infrastructure (e.g. tasks that are barely
 564 unschedulable can become schedulable by a simple remapping of one of the
 565 higher priority tasks that interfere with their computation or communication,
 566 thus changing the set \mathbf{interf} in Equation 1). The same can happen in the
 567 case of route randomisation, since changes on mapping can determine which
 568 randomised routes interfere with each other and in turn affect schedulability
 569 through changes in the \mathbf{interf}_r set.

570 Figure 5 shows the evolutionary pipeline proposed here, which starts with
 571 an arbitrary population of task mappings using a given route randomisation
 572 approach and a given level of security. It then uses evolutionary operators
 573 such as mutation and crossover to improve the mapping population with re-
 574 gards to the percentage of schedulable tasks and packets calculated using the
 575 proposed modification of Equation 1. For every generation of the population,
 576 those with the larger number of schedulable tasks and packets are selected

577 to the next generation, where they will be again mutated, crossed-over, eval-
 578 uated and selected to the subsequent generation. The pipeline stops after
 579 a fully schedulable mapping is found, or a predefined maximum number of
 580 generations is reached.

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

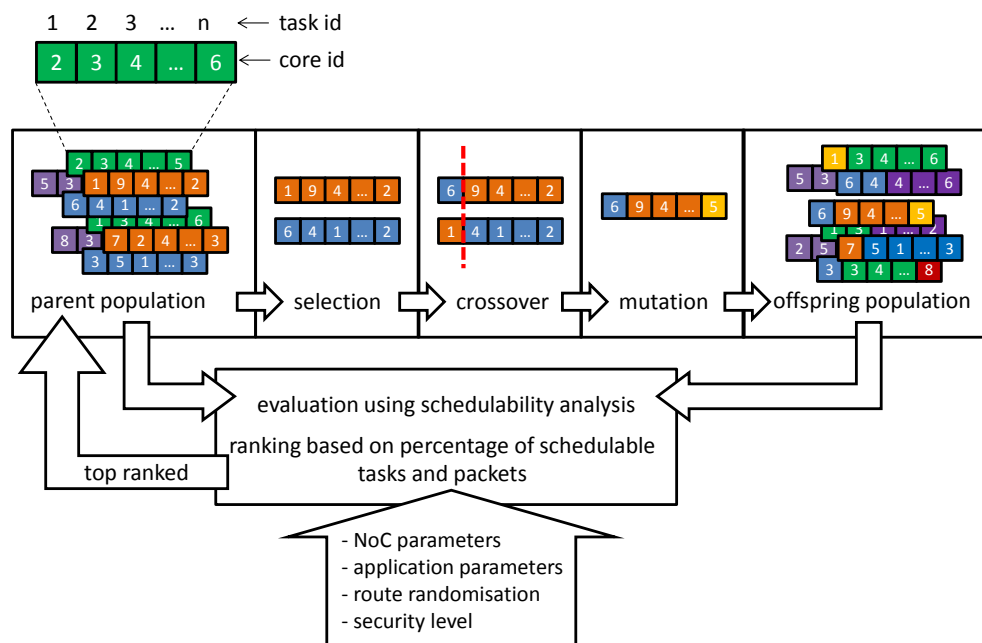


Figure 5: Evolutionary pipeline to optimise performance-security trade-off

587 In this paper, we consider two types of route randomisation which can be
 588 implemented either as source or distributed routing, namely random XY/YX
 589 and random west-first. Random XY/YX is a randomised version of pseudo-
 590 adaptive XY routing used in [34], so the route of the packet to its destination
 591 is randomly chosen between the XY or the YX route prior to the injection
 592 of the packet header into the network. In random west-first, we randomise
 593 one of the turn model routing approaches [35] so that whenever a packet is

594 allowed more than one route it randomly chooses one of them (i.e. uniform
595 probability among all alternatives).

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

606 5. Experimental Work

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

613 5.1. *Schedulability-driven optimisation of route randomisation*

614 This section presents the workflow for analytic schedulability evaluation,
615 and evolution with an evolutionary pipeline based on a genetic algorithm
616 (GA). It follows the pipeline presented in Figure 5. To evaluate the challenge
617 of optimising different applications with different levels of load, we synthet-
618 ically generate thousands of applications, each of them composed of tasks
619 that communicate with each other with different numbers of packet flows.
620 We then apply the evolutionary pipeline to each one of those applications,
621 aiming to optimise the mappings of tasks in such a way that the whole set
622 of tasks and flows is schedulable at different levels of security. We then plot
623 the percentage of schedulable applications we could achieve for each level of
624 security and each level of load. For the sake of reproducibility, we provide
625 below more details on the whole process.

626 For a single experiment upon a given NoC and set of parameters (e.g.
627 topology, operating frequency, switch and link latencies), a range of packet
628 flow counts are identified, each of which represents a level of communication

629 within the application, and therefore a utilisation load upon the NoC. For
630 each flow count chosen for experimental evaluation, a set of tasksets and
631 packet flowsets are generated, each containing the chosen number of flows.
632 The number of tasks is kept roughly constant, and all of them are either
633 source or destination of at least one packet flow. Therefore, flowsets with
634 higher flow counts represent increasing packet contention between the same
635 endpoints. Flows are assigned to particular source and destination tasks
636 with uniform random probability. This implies that the average number of
637 flows transmitted is even across all tasks, although as a result of the random
638 assignment there may be unique hotspots.

639 Following this, an experiment is initialised by defining a population of
640 initial mappings, and a setting for the target level of security case setting.
641 The levels of security settings are defined as either unsecured, or 25%, 50%,
642 75% and 100% secured flows. The secured flows are those that will use
643 randomised routing, providing increased potential protection against side-
644 channel attacks. In case of a partial provision of security e.g. 50%, security is
645 assigned to the flows in their order of priority, with the highest priority flows
646 being randomised. The rationale is to enforce overall random interference
647 patterns, since higher priority packets are the ones causing interference.

648 A population of chromosomes (each representing of a mapping of tasks to
649 cores upon the NoC, as shown in the upper-left corner of Figure 5) is specified
650 for each level of load (i.e. synthetically generated taskset and flowset with a
651 specific flow count). A genetic algorithm is then used to evolve these chro-
652 mosomes, performing mutation, crossover and evaluation of the population
653 according to a fitness function based on the modified Equation 1. This is
654 done separately for each level of security, each of them generating a different
655 $\mathbf{interf}_r(i)$ set representing the randomised routes of different packet flows.

656 By applying the modified Equation 1 for every packet flow of the appli-
657 cation, it is possible to check whether each of them is schedulable, i.e. their
658 end-to-end latency is less than the respective deadline. The overall fitness
659 of an application is then assumed to be the number of schedulable packet
660 flows. Following the fitness function evaluation, the population is culled to
661 retain only the chromosomes that are at the top of the fitness ranking. If
662 the fitness function indicates that the top-ranked chromosome represents a
663 mapping where all flows are schedulable, then the GA terminates early. Oth-
664 erwise, following the completion of the chromosome improvement process at
665 a fixed number of generations, the best chromosome (output mapping) and
666 schedulability obtained (both aggregate flows and flowsets) is output for dis-

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

Table 1: Evaluation parameters

667 play.

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

670 **No security (NS)** Deterministic routing, fitness function incorporates schedu-
671 lability calculated using Equation 1 with the original $\mathbf{interf}(i)$ set.

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

675 **Application of security a posteriori (SAP)** Evolution is performed us-
676 ing a fitness function that tests the schedulability without any security
677 mechanisms (only deterministic routing), aiming to find a schedulable
678 mapping without security considerations. Following the completion
679 of this evolutionary process, the evolved best application mapping has
680 100% of its packet routes randomised, and is then evaluated with Equa-
681 tion 1 with the proposed $\mathbf{interf}_r(i)$ set. This experiment therefore aims
682 to show that the optimisation of the mapping should take into account
683 route randomisation, and that poor results can be expected from apply-
684 ing randomisation to a mapping that was optimised for deterministic
685 routing.

686 5.1.1. Results

687 Figure 6a shows the aggregate schedulability of flows after improvement
 688 with the GA, as a mean proportion across all flowsets generated for that
 689 data point. It is clear that the ordering of the results series in the illustrated
 690 plot follows the proportion of security provided, with an increasing number
 691 of flows in the flowsets (and therefore an increasing load upon the NoC) pro-
 692 viding a slight reduction in schedulability of the evolved cases. This is as
 693 anticipated, in that the worst-case schedulability analysis would be affected
 694 by the increased interference present from the optional random routes. How-
 695 ever, since each GA run is an independent evolutionary process, the ordering
 696 of the series does not always follow the anticipated order. In the SAP se-
 697 ries (security a posteriori), evolution is performed using a fitness function
 698 that tested schedulability under the no security case (XY routing). How-
 699 ever, following the completion of the GA the evolved mapping schedulability
 700 was evaluated with all flows using randomised routing. As anticipated, the
 701 schedulability of SAP is considerably worse than the NS or PS series, since
 702 the evolution was performed using a routing strategy that assumes lower in-
 703 terference than the final evaluation case. Figure 6b shows the schedulability
 704 of flowsets. A flowset is only considered schedulable if every flow within it
 705 is schedulable. The results follow the same general trend as in Figure 6a,
 706 although they reach zero earlier since flowset schedulability requires every
 707 component flow to be schedulable.

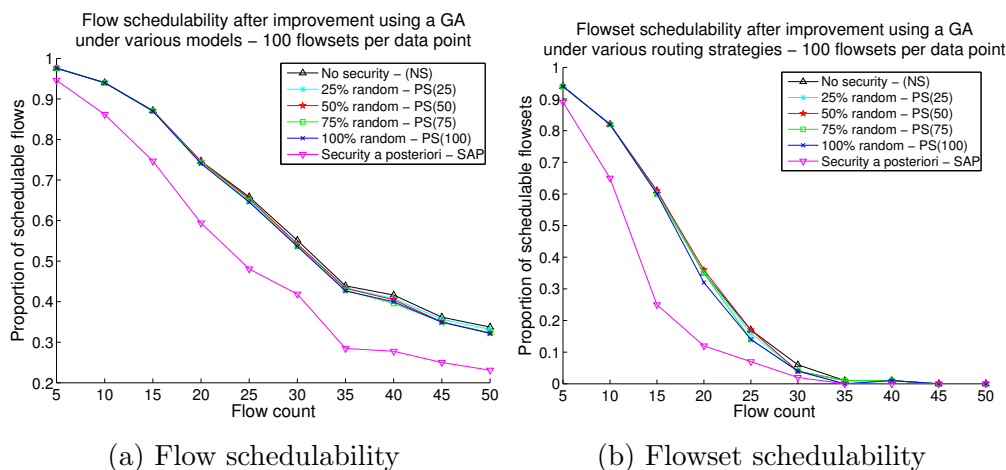


Figure 6: Schedulability under various security models in the 4x4 case

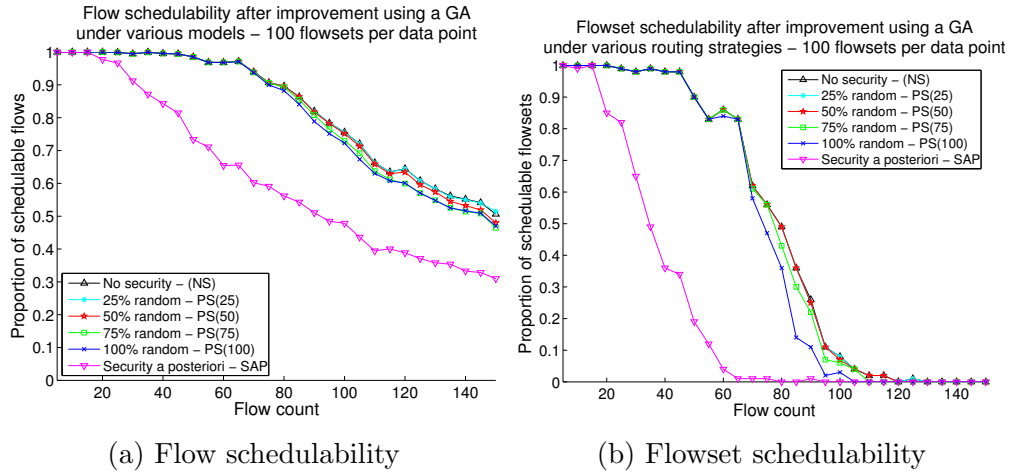


Figure 7: Schedulability under various security models in the 8x8 case

708 For the 8x8 example evaluation case, the results are presented in Figures
709 7a and 7b. The results show a greater separation between the NS and PS
710 series after NoC evolution, due to the increased NoC size and number of flows
711 allowing a greater complexity of interference graphs when randomised routing
712 is enabled. The SAP case also has significantly lower schedulability, since its
713 evolved mapping was obtained without routing randomisation and imposing
714 randomisation later affects schedulability. In the schedulability of flowsets
715 in Figure 7b, it is clear there is a wider difference in schedulability between
716 the PS(100) secured case and NS (no security) particularly in flowsets with
717 70 to 85 flows. This illustrates that as the interference graph becomes more
718 complex it is harder for the GA to find schedulable mappings.

719 5.2. Cycle-accurate simulation of route randomisation

720 One of the key concerns in altering network routing is the impact that
721 it will have upon latency for packet transmission, particularly in latency-
722 sensitive real time applications. This section considers via simulation the
723 impact of randomising of the routing protocol on the latency of a previously
724 published real-time application case, the autonomous vehicle application [14].

725 The simulation framework used for this section is a cycle-accurate NoC
726 model with support for priority preemption and virtual channels. This sim-
727 ulator has been extensively validated in our previous work, frequently being
728 used as a baseline for results in latency and power analysis [37] [38].

729 *5.2.1. Application Structure*

730 The application used in this application is an autonomous vehicle (AV)
731 application [14]. This application consists of 38 communicating flows be-
732 tween a set of tasks that represent video processing, system monitoring and
733 control for a robotic vehicle. As is the convention throughout this paper, pri-
734 orities are defined such that lower priority index values represent the highest
735 priority transmissions. The priorities, data transmission rates, frequencies
736 and deadlines of these application transmissions are as defined in [14], al-
737 though a different mapping has been used in order to show the impact of
738 routing protocols on a randomly selected mapping without artificial tuning
739 to favour a particular routing protocol. The application has been mapped
740 onto a 4x3 NoC, and the video resolution of the AV application video streams
741 is 640x480. Since the application mapping is static and a single priority level
742 is used per packet, a packet always travels between a fixed source-destination
743 pair during the simulation.

744 *5.2.2. Routing Alternatives*

745 In this simulation evaluation, two routing alternatives incorporating ran-
746 domisation are used, in addition to the baseline comparison of XY routing.
747 The first routing alternative uses the XY/YX approach. In this approach,
748 traffic producers determine uniformly randomly on injection whether a data
749 packet will use XY or YX routing, and following this decision a flag is set
750 in the data packet to control the routing behaviour. As a result, the chosen
751 routing algorithm (either XY or YX) is used throughout packet transmission.

752 In addition, an alternative routing structure known as random west first
753 (RWF) routing is also implemented, which allows randomised routing deci-
754 sions to be taken by individual arbiters during data transmission. RWF
755 requires the packet always be forwarded towards the west when the desti-
756 nation node is west of the current arbiter. However, any other destination
757 port can be chosen uniformly randomly (east, north or south) as long as the
758 direction taken is towards the destination. Therefore, the RWF approach
759 permits a more diverse range of transmission paths than the XY/YX se-
760 lection approach, providing more potential protection against side channel
761 attacks.

762 *5.2.3. Evaluation Results*

763 The results are presented in Figures 8 and 9, illustrating the max-min-
764 mean latencies and normalised latencies for the randomised routing cases

765 (XY/YX and RWF) versus the baseline. Normalised latency is calculated
766 by dividing the end-to-end latency of the packets by the packet size, which
767 provides a metric of latency per flit. This metric is therefore more sensitive
768 to delays in the transmission of short packets.

769 The latency results presented in Figure 8 illustrate that routing randomi-
770 sation typically increases the communication latencies for the majority of
771 packets compared to fixed XY routing. This is particularly evident in the
772 case of the packets with priority 8 under RWF routing, which experience
773 an increased latency due to contention with other higher priority flows on
774 some of the randomly chosen routes. In the XY/YX routing case, increased
775 latency is also observed for the packets with priorities 21 and 26 in some
776 cases. Interestingly, for some of the packet transmissions with priority 10
777 and 13, the use of randomised routing is also to reduce latency in the best
778 case, either by routing a higher priority packet so that it no longer causes
779 interference, or routing the current packet around the interferer.

780 Considering the normalised latency results in Figure 9, it is clear that
781 the relative impact of route randomisation is most significant upon packets
782 with priorities 13, 15, 18 and 26. These transmissions represent some of the
783 shortest packets in the system, which are therefore more greatly impacted on
784 a relative basis by contention with other packets. As depicted in the previous
785 figure, some priority 13 packets encounter a large reduction in latency during
786 some transmissions as a result of avoiding interference.

787 6. Conclusions and Future Work

788 This paper has addressed the trade-off between security and hard real-
789 time performance guarantees in Networks-on-Chip. It has proposed route
790 randomisation as a way to increase NoC resilience against side-channel at-
791 tacks, and has discussed a number of design alternatives for the randomi-
792 sation approach. It then has proposed a schedulability test for applications
793 running over a secure priority-preemptive NoCs using route randomisation.
794 Finally, the paper identifies an optimisation pipeline which can be guided
795 by the proposed schedulability test towards configurations that can achieve
796 full schedulability while maximising the provided level of security. Extensive
797 experimental work using 4x4 and 8x8 NoCs with random XY/YX routing
798 running thousands of synthetically generated applications show the perfor-
799 mance guarantees that can be achieved by the proposed approach at four
800 different levels of security, compared against two baselines (no security, and

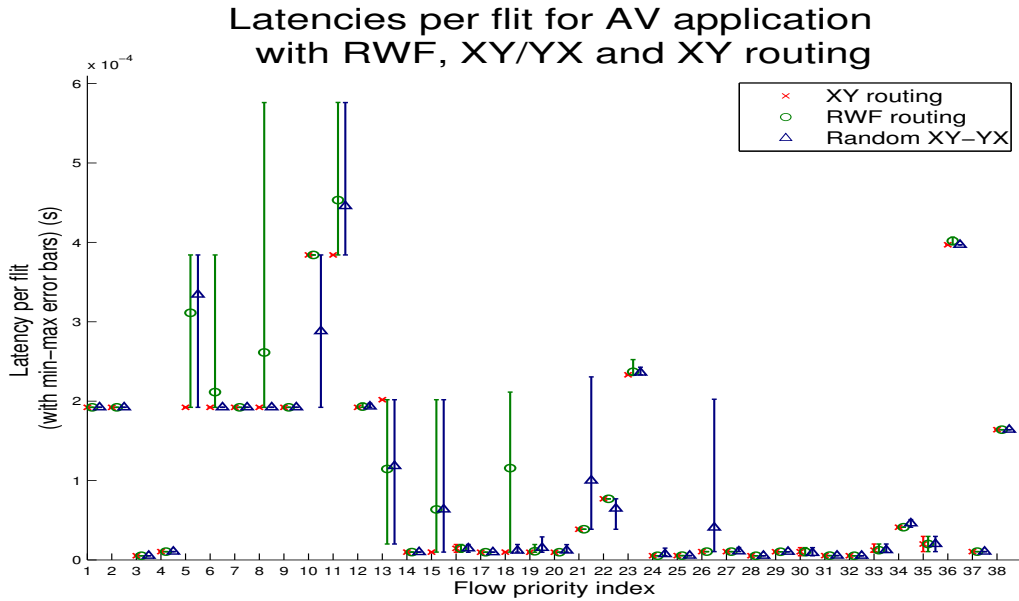


Figure 8: Communication latency results for the randomised routing case on the AV application

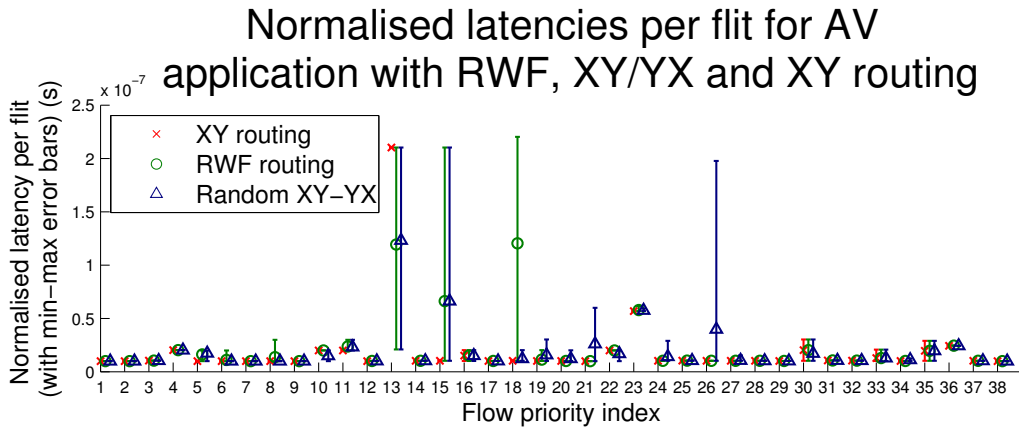


Figure 9: Communication latency results (normalised) for the randomised routing case on the AV application

801 full security applied a posteriori). Additional experiments with a realistic
 802 application running over 4x3 NoCs with random XY/YX and random west-
 803 first routing were performed with a cycle-accurate simulator, aiming to show

804 the impact of route randomisation on latency variability, which in turn shows
805 the increased resilience against side-channel attacks.

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

815 *Acknowledgements*

816 The research described in this paper is funded, in part, by the EPSRC
817 grant, MCC (EP/K011626/1). No new primary data were created during
818 this study. This work was partly funded by the German Federal Ministry of
819 Education and Research (BMBF), grant number 01IS160253 (ARAMiS II).

820 **References**

- 821 [1] C. Silvano, M. Lajolo, G. Palermo, Low Power Networks-on-Chip,
822 Springer Science & Business Media, 2010.
- 823 [2] M. Radetzki, C. Feng, X. Zhao, A. Jantsch, Methods for Fault Tolerance
824 in Networks-on-chip, *ACM Comput. Surv.* 46 (2013) 8:1–8:38.
- 825 [3] S. Pestana, E. Rijpkema, A. Radulescu, K. Goossens, O. Gangwal,
826 Cost-performance trade-offs in networks on chip: a simulation-based
827 approach, in: *Design, Automation and Test in Europe Conference and
828 Exhibition*, 2004. *Proceedings*, pp. 764–769.
- 829 [4] M. N. S. M. Sayuti, L. S. Indrusiak, Real-time low-power task mapping
830 in networks-on-chip, in: *VLSI (ISVLSI)*, 2013 IEEE Computer Society
831 Annual Symposium on, pp. 14–19.
- 832 [5] B. Nikolic, H. I. Ali, S. M. Petters, L. M. Pinho, Are Virtual Channels
833 the Bottleneck of Priority-aware Wormhole-switched NoC-based Many-
834 cores?, in: *Proceedings of the 21st International Conference on Real-
835 Time Networks and Systems, RTNS '13*, ACM, New York, NY, USA,
836 2013, pp. 13–22.

- 837 [6] J. Sepulveda, M. Gross, A. Zankl, G. Sigl, Exploiting bus communi-
838 cation to improve cache attacks on systems-on-chips, in: 2017 IEEE
839 Computer Society Annual Symposium on VLSI (ISVLSI), pp. 284–289.
- 840 [7] C. e. a. Reinbrecht, Gossip noc - avoiding timing side-channel attacks
841 through traffic management, in: ISVLSI 16, Pittsburgh, USA, pp. 601–
842 606.
- 843 [8] C. Reinbrecht, B. Forlin, A. Zankl, J. Sepulveda, Earthquake - a noc-
844 based optimized differential cache-collision attack for mpsoCs, in: 2018
845 Design, Automation Test in Europe Conference Exhibition (DATE), pp.
846 648–653.
- 847 [9] L. S. Indrusiak, J. Harbin, M. J. Sepulveda, Side-channel attack
848 resilience through route randomisation in secure real-time networks-
849 on-chip, in: 2017 12th International Symposium on Reconfigurable
850 Communication-centric Systems-on-Chip (ReCoSoC), pp. 1–8.
- 851 [10] Z. Shi, A. Burns, L. S. Indrusiak, Schedulability Analysis for Real Time
852 On-Chip Communication with Wormhole Switching, *International Journal of Embedded and Real-Time Communication Systems* 1 (2010) 1 –
853 22.
854
- 855 [11] M. Schoeberl, A Time-Triggered Network-on-Chip, in: *Field Pro-*
856 *grammable Logic and Applications, 2007. FPL 2007. International Con-*
857 *ference on*, pp. 377–382.
- 858 [12] D. Dasari, B. Nikolic, V. Nelis, S. M. Petters, NoC Contention Analysis
859 Using a Branch-and-prune Algorithm, *ACM Trans. Embed. Comput. Syst.* 13 (2014) 113:1–113:26.
860
- 861 [13] A. E. Kiasari, A. Jantsch, Z. Lu, Mathematical Formalisms for Perform-
862 ance Evaluation of Networks-on-chip, *ACM Comput. Surv.* 45 (2013)
863 38:1–38:41.
- 864 [14] L. S. Indrusiak, End-to-end schedulability tests for multiprocessor em-
865 bedded systems based on networks-on-chip with priority-preemptive ar-
866 bitration, *Journal of Systems Architecture* 60 (2014) 553–561.

- 867 [15] W. Yao, G. Suh, Efficient timing channel protection for on-chip net-
868 works, in: Networks on Chip (NoCS), 2012 Sixth IEEE/ACM Interna-
869 tional Symposium on, pp. 142–151.
- 870 [16] J. Sepulveda, M. Soeken, D. Florez, J.-P. Diguët, G. Gogniat, Dynamic
871 noc buffer allocation for mpsoC timing side channel attack protection, in:
872 Circuits and Systems (LASCAS), 2016 IEEE Seventh Latin American
873 Symposium on, IEEE, pp. 1–4.
- 874 [17] M. Yoon, S. Mohan, C. Chen, L. Sha, Taskshuffler: A schedule ran-
875 domization protocol for obfuscation against timing inference attacks in
876 real-time systems, in: 22nd IEEE Real-Time and Embedded Technology
877 and Applications Symposium (RTAS 2016), IEEE, pp. 1–12.
- 878 [18] A. Lima, F. Rocha, M. Vlp, P. Esteves-Verissimo, Towards safe and
879 secure autonomous and cooperative vehicle ecosystems, in: Proceedings
880 of the Second ACM Workshop on Cyber-Physical Systems Security and
881 PrivaCy, ACM, pp. 59–70.
- 882 [19] J.-P. Diguët, S. Evain, R. Vaslin, G. Gogniat, E. Juin, Noc-centric
883 security of reconfigurable soc, in: Networks-on-Chip, 2007. NOCS 2007.
884 First International Symposium on, pp. 223–232.
- 885 [20] T. Iakymchuk, M. Nikodem, K. Kepa, Temperature-based covert chan-
886 nel in FPGA systems, in: 2011 6th International Workshop on Re-
887 configurable Communication-centric Systems-on-Chip (ReCoSoC), pp.
888 1–7.
- 889 [21] D. Papp, Z. Ma, L. Buttyan, Embedded systems security: Threats,
890 vulnerabilities, and attack taxonomy, in: Privacy, Security and Trust
891 (PST), 2015 13th Annual Conference on, IEEE, pp. 145–152.
- 892 [22] D. M. Ancajas, K. Chakraborty, S. Roy, Fort-nocs: Mitigating the
893 threat of a compromised noc, in: Proceedings of the 51st Annual Design
894 Automation Conference, DAC '14, ACM, New York, NY, USA, 2014,
895 pp. 158:1–158:6.
- 896 [23] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, C. Silvano, Secure
897 memory accesses on networks-on-chip, Computers, IEEE Transactions
898 on 57 (2008) 1216–1229.

- 899 [24] S. Lukovic, N. Christianos, Enhancing network-on-chip components to
900 support security of processing elements, in: Proceedings of the 5th
901 Workshop on Embedded Systems Security, WESS '10, ACM, New York,
902 NY, USA, 2010, pp. 12:1–12:9.
- 903 [25] J. Sepulveda, J.-P. Diguët, M. Strum, G. Gogniat, Noc-based protection
904 for soc time-driven attacks, *Embedded Systems Letters, IEEE* 7 (2015)
905 7–10.
- 906 [26] H. Wassel, G. Ying, J. Oberg, T. Huffmire, R. Kastner, F. Chong,
907 T. Sherwood, Networks on chip with provable security properties, *Micro,*
908 *IEEE* 34 (2014) 57–68.
- 909 [27] J. Sepulveda, R. Pires, G. Gogniat, W. J. Chau, M. Strum, Qoss hier-
910 archical noc-based architecture for mpsoc dynamic protection, *Internation-*
911 *al Journal of Reconfigurable Computing* 2012 (2012) 3.
- 912 [28] J. Sepulveda, G. Gogniat, D. Florez, J.-P. Diguët, C. Zeferino, M. Strum,
913 Elastic security zones for noc-based 3d-mpsocs, in: *Electronics, Circuits*
914 *and Systems (ICECS), 2014 21st IEEE International Conference on,*
915 *IEEE*, pp. 506–509.
- 916 [29] J. Sepulveda, D. Florez, G. Gogniat, Reconfigurable security architec-
917 ture for disrupted protection zones in noc-based mpsocs, in: *Reconfig-*
918 *urable Communication-centric Systems-on-Chip (ReCoSoC), 2015 10th*
919 *International Symposium on, IEEE*, pp. 1–8.
- 920 [30] P. Cotret, G. Gogniat, J. Sepulveda, Protection of heterogeneous archi-
921 tectures on fpgas: An approach based on hardware firewalls, *Micropro-*
922 *cessors and Microsystems* (2016) 1–31.
- 923 [31] A. Psarras, J. Lee, I. Seitanidis, C. Nicopoulos, G. Dimitrakopoulos,
924 Phasenoc: Versatile network traffic isolation through tdm-scheduled vir-
925 tual channels, *IEEE Transactions on Computer-Aided Design of Inte-*
926 *grated Circuits and Systems* 35 (2016) 844–857.
- 927 [32] S. Pasricha, N. Dutt, *On-chip communication architectures: system on*
928 *chip interconnect*, Morgan Kaufmann, 2010.

- 929 [33] F. Moraes, N. Calazans, A. Mello, L. Moeller, L. Ost, HERMES: an
930 infrastructure for low area overhead packet-switching networks on chip,
931 *Integration, the VLSI Journal* 38 (2004) 69–93.
- 932 [34] M. Dehyadgari, M. Nickray, A. Afzali-Kusha, Z. Navabi, Evaluation of
933 pseudo adaptive XY routing using an object oriented model for NOC,
934 in: *The 17th International Conference on Microelectronics, 2005. ICM*
935 *2005*, pp. 5 pp.–.
- 936 [35] C. J. Glass, L. M. Ni, The Turn Model for Adaptive Routing, in:
937 *Proceedings of the 19th Annual International Symposium on Computer*
938 *Architecture, ISCA '92*, ACM, New York, NY, USA, 1992, pp. 278–287.
- 939 [36] Z. Shi, A. Burns, Real-Time Communication Analysis for On-Chip Net-
940 works with Wormhole Switching, in: *ACM/IEEE Int Symposium on*
941 *Networks-on-Chip (NOCS)*, pp. 161–170.
- 942 [37] L. S. Indrusiak, J. Harbin, O. M. Santos, Fast Simulation of Networks-
943 on-Chip with Priority-Preemptive Arbitration, *ACM Trans. Des. Au-*
944 *tom. Electron. Syst.* 20 (2015) 56:1–56:22.
- 945 [38] J. Harbin, L. S. Indrusiak, Comparative performance evaluation of la-
946 tency and link dynamic power consumption modelling algorithms in
947 wormhole switching networks on chip, *Journal of Systems Architecture*
948 63 (2016) 33–47.