Evaluating the Cost to Cipher the NoC Communication

Bruno Oliveira, Rafael Reusch, Henrique Medina, Fernando Moraes
FACIN - PUCRS - Av. Ipiranga 6681, 90619-900, Porto Alegre, Brazil
{bruno.scherer, rafael.reusch, henrique.medina}@acad.pucrs.br, fernando.moraes@pucrs.br

***Abstract*—The trend in the semiconductor industry is to present more highly dense and functional MPSoC due to the increasing demand for interconnected devices (e.g., Internet of Things). In this context, these devices are holding an increasing amount of personal data from its users. With the goal of protecting users against attacks, this paper proposes a secure architecture and provides the costs of increasing the security for Networks-on-Chip (NoCs). The architecture is composed of a firewall capable of filtering incoming and outgoing network traffic and encrypting sensitive information performing end-to-end security using an AES cipher block. The Firewall plus the AES increases the router area by 193.7% and latency increases in the worst-case scenario 395.92%. Despite this performance penalty, the traffic is protected against attacks. Considering that the injection rate of applications is small (typically 5-10%), a small performance overhead at the application level is expected.**

***Index Terms*—NoC, MPSoC, Security, AES, Ciphers, Firewall.**

I. Introduction and Related Work

With the advances in manufacturing technologies of Integrated Circuits (ICs), taking into account Moore's law, implementing complete systems into a single die coined the term Systems-on-Chip (SoC). The current trend took the industry to build SoCs which present multiple multiprocessors, the so-called Multiprocessor SoC (MPSoC). MPSoCs are mostly Processing Elements (PEs) and Intellectual Property (IP) modules interconnected by a communication infrastructure. As the International Technology Roadmap for Semiconductors (ITRS) [1] foresees thousands of PEs integrated into a SoC by 2020, there is a growing need for a reliable and scalable communication architecture. The search for reliable and scalable communicating architecture, lead the community to adopt the Networks-on-Chip paradigm. The change in the approach of interconnections deeply affected the design of MPSoCs and became widely adopted.

In the case of the adoption MPSoCs for the embedded systems market, the complexity and the concern for the security that resides in such systems increases. By using such solutions in major public markets, as telecommunications, turn-out to become targets due to the amount of sensitive data available in the devices. Credit-Card, Social Security numbers and so on, are ever more intrinsic to our digital identities and are residing on our devices. Software-based attacks account for 80% of security incidents in embedded systems [2], by using abnormal communication, such as viruses and worms, exploit code structures, for example, buffer overflows. NoCs, are vulnerable to network attacks such as Denial of Services, Data Extraction, timing attacks, Hijacking Attacks. Also, there are hardware attacks that might compromise the device security (e.g., Hardware Trojans).

Reviewing the state-of-the-art, different ways to increase the security of intra-chip networks were found. Still, some points need to be addressed. In the work of Gebotys et al. [3] even though only the secure cores have the capability of decrypting the security keys, passing it through the NoC is not a secure strategy. For example, a DoS attack may be able to flood the network making it hard to allocate new keys. Sepúlveda et al. [4] [5] develop a strategy for creating networks that are secure and independent inside the NoC (i.e., secure regions). This approach may present a limitation when the MPSoC is under heavy load, with multiple applications running at the same time. If new applications need to be executed, the security zones may prevent the execution of these new applications due to the reservation of resources.

The work of Grammatikakis et al. [6] present a firewall that prevents cores from accessing memory segments that are not allowed. This implementation needs to be preconfigured, and its verification is based on the physical address of the initiators. This type of firewall will not be able to cover data extraction from a malicious task running on the same core as the communication initiator. Hu et al. [4] have shown a different method of implementing secure networks. Still, there was no demonstration on how it behaved with constraints on latency or QoS results.

Ancajas et al. [7] present a work that not only encrypts the information that travels in the NoC, it also presents a way to authenticate it. Furthermore, it presents a concern with a sophisticated attack such as Side Channel attacks. Without considering the Obfuscation strategy, the work herein proposed has similar features to [7]. The goal of not only hiding the sensitive data transmitted through the network and also being able to authenticate it, the strategy used in extra-chip networks, as the Secure Socket Layer (SSL), that present the same strategy.

Fernandes et al. [8] presented a solution using routing algorithms and security zones that might not be scalable to high-density NoC (due to the size of the routing tables). Still, for small NoC sizes, it is a fair method to provide security. On the downside, the technique needs to understand the security requirements during the development phase, to provide its heuristic to determine the weight given to the cost function of each path the algorithm will use. Furthermore, the proposed technique is not suitable for general purpose computing, since it is executed at design time.

Finally, the work of Wehbe et al. [9] exhibit a solution that is dependable on the IP provider to define private and public keys for its cores. That will have an impact on which on selecting IP providers (as not all IP designers will provide it). The work is both focused on security as well as fault tolerance. The main idea presented in the paper is built around its reconfigurable NoC.

All of these reviewed works present insights in the security field over NoCs. Still, one major gap remains. None of them present data regarding how much those strategies cost when taking the unsecure environment as the reference. The *goal* of this work is twofold. The first goal is to present a secure environment architecture, enabling to cipher the NoC flows (Section II). The second goal is to evaluate the silicon area and latency overhead, using as reference the baseline router without any security mechanism (Section III).

II. Secure Environment Architecture - SEA

This Section presents the Secured Environment Architecture (SEA) without the creation of secured zones. In the state-of-the-art, it is common to find secure environments with restricted zones, where packets are limited to run through the NoC. This type of security creates a reasonable amount of constraints. For example, when creating a secure zone, it is not possible to have more than one safe application running at each PE in that zone. Thus, limiting the number of applications running in the MPSoC. So, the question is, how to avoid these limitations and remain secure? The idea behind implementing SEA is to provide security at the application level, preventing a malicious application to interfere with the data being handled by communicating tasks of the secured application.

Many-core architectures divide workloads among multiple threads/tasks aiming to scale effectively up system performance without compromising its energy-efficiency. The multi-tasking feature does present security issues regarding messages being exchanged between threads/tasks. A message sent from one task to another task may be read/corrupted by a task in the same processor, or by a task in the path of the message.

A manager PE (a core that does not execute user applications, just management functions) generates random keys. Each key is sent to the firewalls through the Hamiltonian path [10]. Each application has a unique key. Therefore, when mapping the applications through the NoC, the manager PE also send the keys to the firewalls of the selected PEs that execute the application tasks. With the keys being the same between communicating tasks, it is possible to use symmetric cryptography (AES) to encrypt the data that flows through the NoC, and it is also possible to append digested hash information to the payload to confirm its integrity. With these functionalities, the NoC is resilient to the three common types of attacks: DoS, system hijacking, and data extraction.

A. NoC Changes

The router itself was not modified, keeping the same routing algorithm, buffer/arbitration strategy, and topology. The major addition is a firewall module. The firewall is placed between the network interface module and the local router port. Thus, being able to control the communication generated or received by the PEs.

B. Firewall

The firewall was implemented using an existing framework, similar to the firewall developed in [10]. The firewall interconnects the PE, the AES, and the router interface. The principle adopted in the development is to respect the original routers' interfaces, such a way to avoid modifications in the original modules. With this strategy, it is possible to select with routers receive the firewall or not.

Figure 1 presents the interfaces connected to the firewall. The Router and the Network Interface signals are the same. Instead of changing the interfaces, state machines in the firewall manage the flow control signal. The firewall may encrypt or not the packets according to an identifier in the packet. For sensitive applications, all traffic is encrypted, otherwise plain data is transmitted to avoid the encryption overhead.

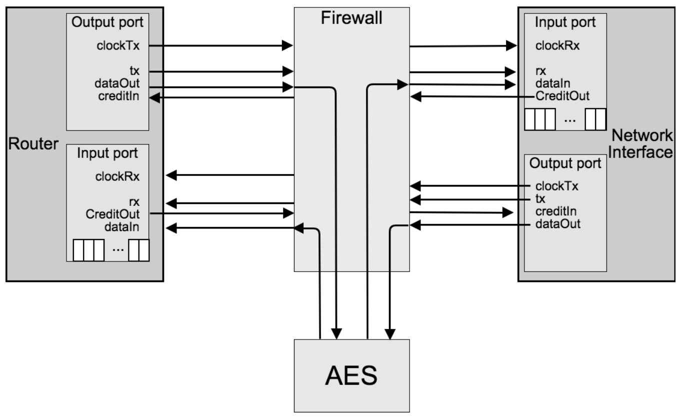

Fig. 1. Representation of the Firewall connections.

The firewall contains two separate states machines that handle the interfaces with the Router and the NI. Also, embedded in each machine there are the interconnections with the AES module. Thus, making the AES module available for one operation at a time. To provide order to the communicating states (encrypt/decrypt), an arbiter was also implemented.

C. AES Core

The AES Core used in our design is based on [11] which is a version of the FIPS-197 implemented in the ECB mode. The author understands that ECB mode presents a risk regarding the repetition of messages that could lead to plaintext attacks. The architecture works with two 64-bit blocks, loaded in consecutive clock cycles using the load signal. The load signal injects both keys (not presented in the Figure) and data. Once the data is loaded, the start signal raises, and after 13 clock

TABLE I
3x3 NoC's latency results (in clock cycles). Baseline: standard NoC. Encrypted: NoC + firewall + AES.

	Scenario A		Scenario B		Scenario C		Scenario D	
	Baseline	Encrypted	Baseline	Encrypted	Baseline	Encrypted	Baseline	Encrypted
#Flits	12.851	9.269	12.851	9.275	12.853	6.965	12.857	8.837
Average	22.02	53.96	22.08	64.47	22.02	63.72	24.04	119.22
Std. Dev	5.25	9.77	5.27	26.26	5.19	25.49	5.33	4.92
Min	13	31	13	20.50	13	31	13	108
Max	59	151	50	154	36	126	36	126

cycles, the done signal rises and the data (encrypted/decrypted) is available.

III. Results

To understand how the NoC performance is affected by the firewall with cryptography, four different test scenarios were evaluated. Each test scenario is simulated with and without encryption. The description of the modules use synthesizable VHDL, and the simulation was made using the ModelSim tool.

The baseline router has the following features [12]: (*i*) 32-bit flit; (*ii*) no virtual channels; (*iii*) centralized round-robing arbitration and XY routing; (*iv*) input buffers with 8-flit depth. This baseline router presents a small area, due to its straightforward design. The goal of using this router is to obtain a fair performance evaluation.

Results evaluate the latency to transmit the packets because this is the main performance figure affected by the encryption process. The test bench annotates the injection moment of the flit in the NI (t_{injec}) and the moment the flit arrives at the output of the NI (t_{recep}). The difference $t_{recep} - t_{injec}$ corresponds to the flit latency (i.e., the network latency). Each scenario is simulated during 1 ms (clock frequency: 100 MHz), with a producer-consumer application injecting 18-flit packets (2 for the header and 16 for the payload). The interval between the packets is 1,000 ns.

Each scenario has the following features, according to Figure 2:

- Scenario (a) - congestion in the flow 00→22, disturbed by flow 01→12, impacting in the evaluated flow due to the arbitration process.
- Scenario (b) - congestion in the target router: router 22 should decrypt two messages, one from router 00 and another from router 10.
- Scenario (c) - congestion in the source router: router 00 should encrypt the flow to 21, and simultaneously decrypt packets from router 02. As only one AES module is available per router, this scenario stresses the firewall, since it should arbitrate between encryption/decryption.
- Scenario (d) - congestion in the source and target routers: routers 00 and routers 20 simultaneously encrypt/decrypt packets.

A. Latency Results

Table I summarizes the latency results, comparing standard (non-ciphered) traffic in the network with encrypted. This first set of experiments reveals the impact to encrypt packets. The average latency increases in the best-case 126.3% (2.26x), and in the worst-case 395.92% (4.95x). This cost may be high, but it is important to mention that applications do no inject packets at a constant rate, as in these experiments. Typically, applications running in an MPSoC have an injection rate below 10% [13]: *"In real-world multi-core applications, less than 5% of channels are utilized on average."*.

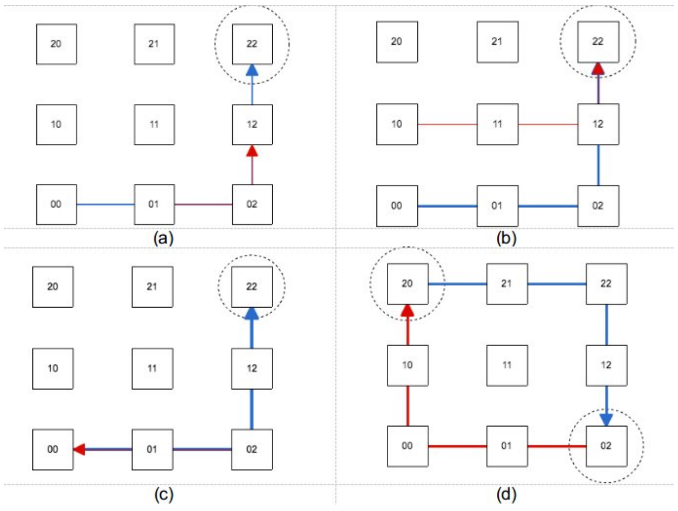

Fig. 2. Flows of the evaluated scenarios: **A**: 00→22 and 01→12; **B**: 00→22 and 10→22; **C**: 00→22 and 22→00; **D**: 02→20 and 20→02

1) Scenario A: Was developed to evaluate the impact of the disturbing traffic in the path of the ciphered traffic. Comparing this scenario with the reference one, the number of delivered flits is 27% smaller. The average latency increased because flits are injected into the west buffer of router 01, and should wait for the end of the transmission 01→12. This blocking situation also blocks the cipher block in router 00, increasing the latency values.

2) Scenario B: Evaluates the congestion at the target router. Again, the number of delivered flits is 27% smaller, but the average latency increased 145%. Since there is only one cipher block, it is necessary to wait for the decryption of one packet to start another reception.

3) Scenario C: Evaluates the congestion at the source router. In this scenario, the source router (00) must decrypt the received packet and encrypt a packet to transmit it. The number of received flits at the target router dropped 46%, and the average latency increased 189.37%

4) Scenario D: Evaluates the congestion at the source router and target routers. This scenario behaves as the previous one, with the difference that the path is not congested. Thus, flits may wait in the path's buffers, reducing the impact of the congestion in the cipher blocks. Besides the higher number of received packets at the target router, the average latency increased by 395.92%.

B. Area Consumption

Table II illustrates the required area for 3x3 mesh NoC with firewalls and AES ciphers. The NoC's area includes all connections, routers, firewall, and cipher. A single firewall including the AES represents an increase of 193.7% in the router's using STMicroelectronics standard-cells CMOS 65 nm technology. A major impact is due to the AES cipher that represents 92.44% of the additional area increase. This area overhead is a trade-off in relation the security that the AES cipher aggregates.

TABLE II
3x3 NoC's area consumption in CORE65GPSVT library (area in μm^2).

Instance	#Cells	Cell Area	Total Area
Firewall	1,019	5,522	8,609
AES	17,097	62,763	105,316
Router Buffer 16	5,926	43,188	58,795
Firewall+AES	18,116	68,275	113,925

IV. Conclusion

In this work, we proposed the implementation of firewalls that included an AES block cipher providing secrecy for the network. The firewall is decoupled from the NoC, being generic and applicable to other NoCs. The increase of concern regarding information security is presented as the major necessity for this work. Our firewall provides a larger area and latency overhead as a trade-off for secrecy. In the Authors knowledge, this is the first work presenting the real silicon cost and latency to add cipher blocks in NoCs.

The firewall presents only low-level access control. As the context protection (key exchange) is delegated to a manager processing element to handle the configuration of each firewall, keys are exchanged securely.

Future work includes the evaluation of sharing AES blocks between neighbor PEs to reduce the area overhead due to the AES module. Currently, the network latency is being evaluated in a real MPSoC, to determine the impact on the performance when encrypting the data packets.

V. Acknowledgement

Author Fernando Gehm Moraes is supported by FAPERGS (17/2551) and CNPq (302531/2016-5), Brazilian funding agencies.

References

[1] International Technology Roadmap for Semiconductors, "ITRS 2015 Report Available at:," http://www.itrs2.net/itrs-reports.html, accessed: November 02, 2017.

[2] Symantec, *Internet Security Threat ReportInternet Security Threat Report*, 2016 (accessed October 21, 2017). [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

[3] C. H. Gebotys and R. J. Gebotys, "A framework for security on NoC Technologies," in *ISVLSI*, Feb 2003, pp. 113–117.

[4] Y. Hu, D. Müller-Gritschneder, M. J. Sepúlveda, G. Gogniat, and U. Schlichtmann, "Automatic ILP-based Firewall Insertion for Secure Application-Specific Networks-on-Chip," in *INA-OCMC*, Jan 2015, pp. 9–12.

[5] J. Sepúlveda and R. Fernandes and C. Marcon and D. Florez and G. Sigl, "A security-aware routing implementation for dynamic data protection in zone-based MPSoC," in *SBCCI*, Aug 2017, pp. 59–64.

[6] M. D. Grammatikakis *et al.*, "Security Effectiveness and a Hardware Firewall for MPSoCs," in *HPCC*, 2014, pp. 1032–1039.

[7] D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-NoCs: Mitigating the threat of a compromised NoC," in *DAC*, 2014, pp. 1–6.

[8] R. Fernandes, C. Marcon, R. Cataldo, J. Silveira, G. Sigl, and J. Sepúlveda, "A security aware routing approach for NoC-based MPSoCs," in *SBCCI*, 2016, pp. 1–6.

[9] T. Wehbe and X. Wang, "Secure and Dependable NoC-Connected Systems on an FPGA Chip," *IEEE Transactions on Reliability*, pp. 1852–1863, Dec 2016.

[10] R. Fernandes, B. Oliveira, J. Sepúlveda, C. Marcon, and F. G. Moraes, "A non-intrusive and reconfigurable access control to secure NoCs," in *ICECS*, 2015, pp. 316–319.

[11] Hemanth, "aes_crypto_core," https://opencores.org/project,aes_crypto_core, accessed: November 02, 2017.

[12] F. G. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *Integration*, vol. 38, no. 1, pp. 69–93, 2004.

[13] R. Hesse, J. Nicholls, and N. E. Jerger, "Fine-Grained Bandwidth Adaptivity in Networks-on-Chip Using Bidirectional Channels," in *NOCS*, 2012, pp. 132–141.