Secure Environment Architecture for MPSoCs

Bruno Scherer Oliveira, Henrique Medina, Anderson Sant'Ana, Fernando Gehm Moraes
PUCRS – Av. Ipiranga 6681, 90619-900, Porto Alegre, Brazil
{bruno.scherer, henrique.medina, anderson.santana.001}@acad.pucrs.br, fernando.moraes@pucrs.br

***Abstract*—The widespread use of complex MPSoCs (Multiprocessor SoCs) is a trend in the semiconductor industry due to the increasing demand for interconnected devices, thus Internet of Things (IoT). The use these devices to store, process and protect private information increases everyday, since all devices are interconnected. As a goal to create resilience against attacks on users' information, this paper proposes a secure architecture and provides the costs of increasing the security for MPSoCs. The secure architecture contains a firewall capable of filtering incoming and outgoing network traffic and ciphering sensitive information by performing end-to-end encryption using an AES cipher block. The keys are stored at each firewall using a secure channel, through a dedicated path, isolated from the NoC. The firewall and the AES block increases the router area by 95.63% and the application execution time increases in the worst-case scenario by 6% using an MPEG application running in a homogeneous MPSoC. Despite the impact in the area of the communication infrastructure and in the execution time of applications, the communication between tasks becomes safe, and the user data is protected against attacks.**

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

I. Introduction

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 industry's current trend is to build SoCs which present multiple multiprocessors, the so-called Multiprocessor SoC (MPSoC), also called many-core systems. 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 adopting 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 [2]. Software-based attacks account for 80% of security incidents in embedded systems [3], 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 Service (DoS), Data Extraction, timing attacks, Hijacking Attacks. Also, there are hardware attacks that might compromise the device security (e.g., Hardware Trojans).

978-1-5386-7431-4/18/$31.00

The literature presents proposals to create isolated environments, as disabling communications that are not certified of their security in the same PE or that can pass through that core. Others create security zones, where a whole area of the MPSoC becomes dedicated to run a single application (or multiple secure applications). Both ideas present a major tradeoff with scalability and performance. For example, a small MPSoC that needs to run multiple applications will have its cores so sectioned that it may not deliver real-time performance. Therefore, the use of these techniques might be inappropriate for to real-time systems, an important feature for most applications in the telecom and Internet of Things (IoT) domain.

The *goal* of this paper is to present a secure architecture to increase the security of applications running in MPSoCs. The secure architecture contains a firewall capable of filtering incoming and outgoing network traffic and ciphering sensitive information by performing end-to-end encryption using an AES cipher block. The keys are stored at each firewall using a secure channel, through a dedicated channel, isolated from the NoC.

This paper is organized as follows. Section II presents the State-of-the-Art research in the field of security for MPSoC. Section III details the proposed architecture. Section IV presents results of the proposed security solutions. Section V draws the conclusions and directions for future work.

II. Related Work

The state-of-the-art review contains different methods to increase the security of intra-chip networks. Still, some points need to be addressed. In the work of Gebotys et al. [4] a framework for security on NoC is presented. To control the information flow through the NoC it uses cryptography to deliver data to the secure applications. A secure PE (processing element), called Key Keeper Core, creates the public keys. The remaining of the NoC is connected to two types of PEs, the secure ones, called SCores (being able to execute one or more secure applications, such as encryption, authentication, key exchange, etc.) and a non-secure core that runs general purpose applications. 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. [5], [6] develop a strategy for creating networks that are secure and independent inside the NoC (i.e., secure regions) by using the Diffie-Hellman together with a mapping-aware key pre-distribution scheme, it is possible to create authenticated and separated secured zones. Furthermore, all components of the same security zone are considered secure among them. Thus, all transactions inside the security zone can be performed in plain text (without the need of ciphering) and without security checking. Otherwise, messages are sent in meta packets, i.e., encrypted packets, which are verified and encoded. This approach may present a limitation when the MPSoC is under heavy load, with multiple applications running at the same time. In the case of new applications having 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. [7] present a firewall that is placed between the NI and the PE (CPU, memory controller or hardware accelerator), making it possible to stop malicious injections from the applications running in the PE to the Network. Furthermore, the implementation needs to be configured (statically or dynamically) for systems or users to grant access to physical memory, and its verification is based on the physical address of the communication 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. This approach can filter underlying transactions and ensures that security rules are obeyed at the NI of each initiator by providing fast and efficient access to rules. Therefore, a compromised process cannot access data owned by a secure process, thus subverting denial-of-service (DoS) attacks, malware attacks on data or hardware/software vulnerabilities and eavesdropping.

Hu et al. [5] showed a different technique for implementing secure networks. A method of inserting firewalls between routers to avoid insecurities caused by third-party IPs is proposed. The approach is to isolate trusted initiators and targets with sensitive data in secure domains. Data flows from untrusted initiators must pass the firewall to enter a secure domain. Thus, no additional firewalls or security headers are required inside. Furthermore, in the paper, it is presented an automatic approach to satisfy a given set of security requirements, with a method based on integer linear programming (ILP). Still, there was no demonstration of how it behaved with constraints on latency or QoS results.

Ancajas et al. [8] 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 [8]. 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. [9] 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.

Wehbe et al. [10] 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 each one selected 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, two major gaps remain. The first one is that the proposals are NoC-centric, i.e., they do not consider a real MPSoC and the corresponding cost of the security with real applications. The second one is the fact that none of them present data regarding how much those strategies cost when taking the unsecure environment as the reference.

The *contribution* of this work is twofold. The first one is to present a secure environment architecture, enabling to cipher the NoC flows (Section III). The second contribution is to evaluate the silicon area and latency overhead to obtain both security and high-performance in the MPSoC, using as reference a baseline architecture without security mechanism. Cyphering the data NoC prevents eavesdropping, man-in-the-middle, and spoofing attacks.

III. Secure Environment Architecture - SEA

This Section presents the Secure Environment Architecture (SEA). Many-core architectures divide workloads among multiple threads/tasks aiming to scale efficiently up system performance without compromising its energy-efficiency. The multitasking feature does present security issues regarding messages 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.

In the state-of-the-art, it is common to find secure environments with restricted zones, where packets are limited to run through in the NoC's region. 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.

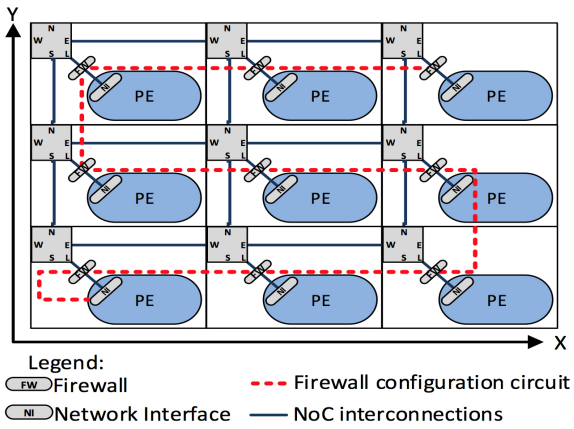

Fig. 1. MPSoC components, with a firewall (FW) between the router and processing element. A Hamiltonian Path interconnects the firewalls [11]. Only a manager PE has access to this path, resulting in a secure path to send sensitive data.

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 a Hamiltonian path [11]. The Hamiltonian path is a serial communicating channel separated from the NoC communication infrastructure. Thus, making it a secure way to deliver sensitive information such as keys. Figure 1 shows this infrastructure. Each application has its 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. Also, it is possible to append common integrity mechanisms, such as CRC or Hash functions as an additional mechanism to support data 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 (NI) module and the local router port. Thus, being able to control the communication generated or received by the PEs. The improvement over the different proposal is that any NoC with a similar interface can use a similar strategy with no significant changes in the original architecture. Thus, our approach is general and may be adopted by other NoCs.

B. Firewall

The firewall was implemented using an existing framework, similar to the firewall developed by Fernandes et al. [11]. The firewall interconnects the PE, the AES, and the router interface. The principle adopted in its development is to respect the original router's interfaces, in such a way to avoid modifications in the original modules. With this strategy, it is possible to select which routers might receive the firewall.

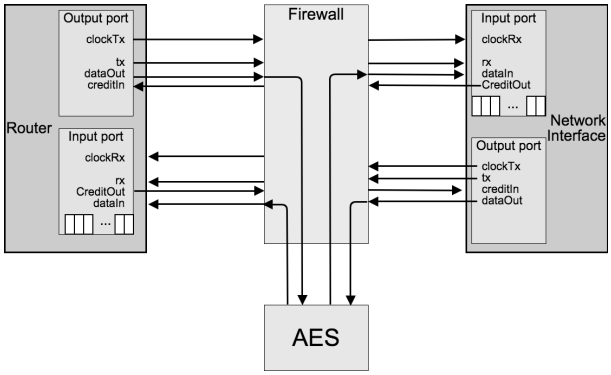

Fig. 2. Representation of the firewall connections. The Hamiltonian path is not included in the Figure for sake of simplicity.

Figure 2 presents the interfaces connected to the firewall. The router and the NI 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.

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. An arbiter manages the communication flows (encryption or decryption).

The communication through the firewall is always full-duplex. Even when both incoming and outgoing flows require encryption or decryption, the arbiter manages the flows, using time slices with four 32-bits words. The arbiter full-duplex behavior is a feature to avoid deadlocks at the communication level.

C. AES Core

The AES Core used in our design is based on [12], 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 and data. Once the data is loaded, the start signal raises, and after 13 clock cycles, the done signal rises and the data (encrypted/decrypted) is available as it can be seen in Figure 3.

IV. Results

This Section evaluates different scenarios, with and without encryption, to assess the impact of the firewall and the AES block. The description of the modules use synthesizable VHDL, and the simulation was made using the ModelSim tool.

The public available HeMPS MPSoC [13] is the baseline MPSoC architecture. It contains a set of processing elements (PEs) interconnected by the Hermes NoC [14]. Each PE

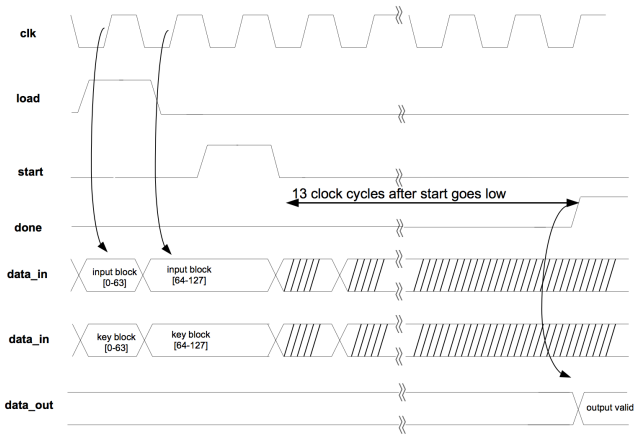

Fig. 3. Process sequence for encryption/decryption data usind the AES core [12].

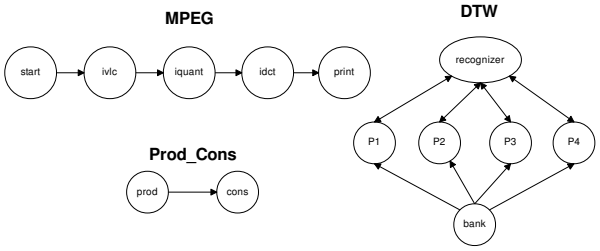

Fig. 4. Task-graphs of the benchmarks used to evaluate the impact of the messages' encryption on the execution time.

executes a small operating system, enabling multitasking and inter-task communication (message exchange). At design time the designer may include or not the structure firewall+AES. The goal of using this infrastructure is to obtain a fair performance evaluation, comparing the applications' performance when communication encryption is enabled.

Results evaluate the iteration latency of different benchmarks because this is the main performance figure affected by the encryption process. Figure 4 shows the task graphs of the applications.

A. Prod_Cons Evaluation

The first experiment evaluates the impact of the firewall with and without encryption, compared to the baseline MPSoC. This scenario uses a 3x3 instance of the MPSoC, executing the producer/consumer application (Prod_Cons in Figure 4), which has a communication intensive profile. The hop distance between the producer-consumer varies from 1 to 4 hops. Due to the software stack to create and transmit the packets, and the corresponding functions to receive them, the number of hops has a negligible impact on the latency.

Oliveira et al. [15] compared the latency of non-ciphered to ciphered flows in NoCs, using synthetic traffic. The latency overhead increased up to 4.95 times in the experiments. However, applications running on MPSoCs have an injection rate below 10% [16]: *"In real-world multi-core applications, less than 5% of channels are utilized on average."*. The experiments using the HeMPS MPSoC corroborate this statement, but as a function of the mapping, several flows share the same link, increasing the NoC utilization, leading to congested links.

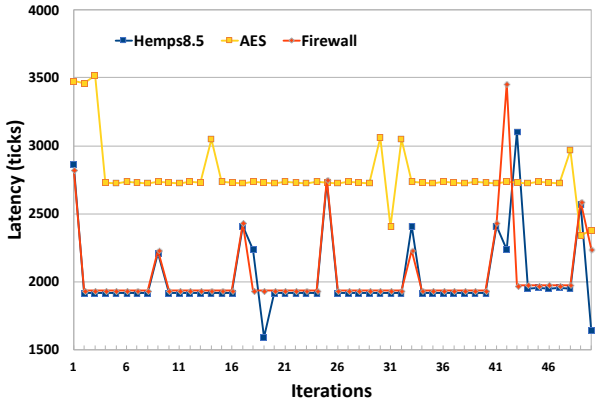

Fig. 5. Prod_Cons iteration latency (in clock cycles), considering the baseline platform (HeMPS 8.5), just the firewall, and with encryption (AES).

Figure 5 presents the iteration latency for: (*i*) baseline MPSoC; (*ii*) MPSoC with firewall and encryption disabled; (*iii*) MPSoC with firewall and encryption enabled. The latency increased in average by 1.39% by adding the firewall in the baseline MPSoC. The latency increase is due to the logic added in the firewall to manage the control flow signals. The latency increased in average by 39% when encrypting the flows. This is an important result, since the Prod_Cons is a synthetic application, with almost no computation (only loops to send/receive packets). Thus, using real traffic, generated by a CPU, the overall impact on the performance is smaller than the one observed by simulating only the NoC+AES with synthetic traffic.

B. MPEG and DTW Evaluation

The second experiment follows the same goal of the first experiment, using two real application, MPEG and DTW, which are computation intensive. Due to the structure of the applications (pipeline for MPEG and master-slave for DTW), the AES block may be a bottleneck in the communication since tasks may simultaneously send and receive data. The implementation of the firewall, with an arbiter enabling full-duplex communication, alleviates this issue. Figure 6 presents the mappings adopted in this experiment.

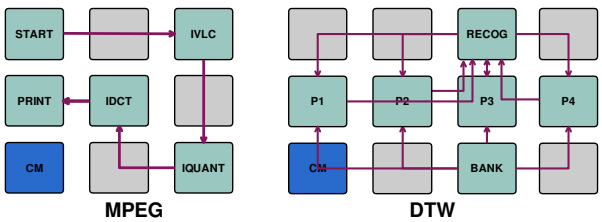

Fig. 6. MPEG and DTW applications' mapping. CM: cluster manager.

Table I presents the iteration execution time, for the baseline MPSoC and the one with encryption, as well as the overhead

TABLE I
MPEG LATENCY RESULTS (RESULTS IN CLOCK CYCLES).

frame number	Iteration Execution Time baseline	with encryption	overhead
5	213,044	218,255	5,211
6	225,470	230,852	5,382
7	224,717	229,778	5,061
8	224,798	229,861	5,063
9	224,633	229,692	5,059
10	225,040	229,778	4,738
11	224,798	230,184	5,386
12	225,532	230,015	4,483
13	225,939	230,101	4,162
14	225,697	229,861	4,164
15	225,532	229,956	4,424
16	224,717	229,778	5,061
17	225,121	229,861	4,740
18	224,956	229,692	4,736
19	225,618	230,101	4,483
20	226,022	230,184	4,162
21	225,534	230,015	4,481
22	225,618	230,679	5,061
Average	**224,599**	229,369	4,519
StdDev	2,835	2,713	1,133

due to the messages' encryption. This experiment does not have NoC congestion. The experiment decodes 26 frames, discarding the latency of first and last four frames (warm-up and NoC flushing). As shown in Figure 4, the MPEG application has a pipeline behavior. Tasks START to IDCT receive and transmit 128-flit packets. The average latency to transmit each packet increases by 880 clock cycles, explaining the overhead observed in Table I - average value of 4,519 clock cycles with a small standard deviation. The relevant result is that the iteration execution time increased, on average, only **2.12%** using an AES block implemented in hardware.

Table II presents the results for the DTW (Dynamic Time Warping) application. This application measures the similarity between two temporal sequences. Task *recognizer* sends four patterns (each one corresponding to a 128-flit message) to the worker threads (*P1* to *P4*), where each one reads the reference patterns from task *bank*. The result of the similarity returns to task *recognizer* as a small packet with the similarity rate (packet with one flit in its payload). Thus, the overhead due to the latency of the encryption/decryption is minimized due to the parallel execution of the worker threads. Each DTW latency, corresponding to the similarity between 4 patterns, increases on average 456 clock cycles. For this application, the adoption of data encryption increase on average **1%** the iteration latency.

TABLE II
DTW LATENCY RESULTS (RESULTS IN CLOCK CYCLES).

	Pattern recognition time. baseline	with encryption	overhead
Average	**47,546**	47,996	456
StdDev	674	627	364

C. Congested Scenario

In this scenario, all PEs execute two tasks simultaneously, by mapping one DTW (6 tasks) and 2 MPEG (5 tasks) applications, as shown in Figure 7. The goal is to evaluate the MPSoC with several tasks exchanging data, with one AES block per core. This scenario uses real applications, with computation-intensive applications.

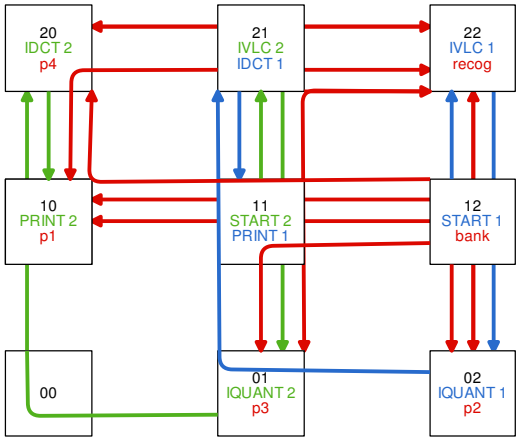

Fig. 7. Task mapping and traffic flows for scenario executing one DTW and 2 MPEG applications.

TABLE III
CONGESTED SCENARIO RESULTS (RESULTS IN CLOCK CYCLES).

	reference	Iteration Latency. without encryption	with encryption	iteration overhead
MPEG1	224,599	252,13 +12%	265,858 +18%	13,721
MPEG2	224,599	286,188 +27%	294,61 +31%	8,424
DTW	47,546	65,143 +37%	66,193 +39%	1,050

Table III evaluates the iteration latency, as in the previous tables. The percentage values corresponds to the overhead w.r.t. to the non-congested scenarios (bold values on Tables I and II). Two components affect the iteration latency: multitasking and concurrency to use the single AES block per core. The effect of the first component, multitasking, is observed in the third column (*without encryption*). The iteration latency for the applications increases from 12% up to 37%, according to the mapping adopted in the simulated scenario. Most of the computing intensive tasks from DTW share the CPU with another computing intensive tasks of the MPEG instances, explaining the higher observed overhead (37%). The effect of the second component, concurrency of the AES block, is observed in the fourth column (*with encryption*). Results of the previous section showed an impact on the iteration latency between 1% to 2.12%. Now, due to the presence of one AES block per PE, a packet encryption or decryption by one task blocks the other task to use the NoC. The impact now can reach 6%, as observed in the first instance of the MPEG application (MPEG1). The iteration overhead in the single-task evaluation was, in average, 4,519 clock cycles, while in the multitasking evaluation the average overhead reached 13,721 clock cycles.

This experiment revealed the real impact of encrypting the

messages. The communication overhead, in non-congested scenarios with one task executing per PE, has a negligible impact on the application performance. However, increasing the number of tasks executing simultaneously per PE, the crypto core may represent a communication bottleneck, even if full-duplex communication is enabled.

D. Area Consumption

Table IV illustrates the required area for 3x3 mesh NoC with firewalls and AES ciphers. The NoC area includes all connections, routers, firewall, and cipher. A single firewall including the AES represents an increase of 95.63% in the router's area using STMicroelectronics standard-cells CMOS 65 nm technology. A major impact is due to the AES cipher that represents 79.12% of the additional area increase. This area overhead is a trade-off in relation the security that the AES cipher aggregates.

TABLE IV
3X3 NOC'S AREA CONSUMPTION IN CORE65GPSVT LIBRARY (μm^2).

Instance	#Cells	Cell Area	Total Area
Firewall	1,441	7,743	9,707
AES	17,097	62,763	105,316
Router Buffer 16	5,926	43,188	58,795
Firewall+AES	18,538	70,506	115,023

Note that the area results are related to the communication infrastructure. If we consider that the router area (communication infrastructure) corresponds to no more than 10% of the PE area (which includes the processor, DMA, memory), doubling the router area means that the PE will increase, in worst-case, 10%.

V. CONCLUSION

In this work, we proposed the implementation of firewalls that included an AES block cipher providing security at the communication level. The firewall is decoupled from the MPSoC, being generic and applicable to other MPSoCs. The increase of concern regarding information security is presented as the major motivation for this work.

Results show a trade-off between silicon area, performance, and workload. The silicon area is an important overhead since the crypto core has an area equivalent to two routers. The performance versus workload is a direct function: the performance penalty increases with the workload executing in the MPSoC.

A direction to follow to diminish the area overhead is the evaluation of lightweight cryptography algorithms. These algorithms also have an impact on reducing the performance overhead. To tackle the performance degradation when increasing the workload different research directions may be followed: (1) take ino account the links' occupation during the application mapping process; (2) avoid CPU sharing between applications requiring security constraints; (3) include dedicated circuit-switching (CS) sub-NoCs (MPN - multiple physical networks) to avoid the encryption of flows using CS.

VI. ACKNOWLEDGEMENTS

Author Fernando Gehm Moraes is supported by FAPERGS (17/2551-196-1) and CNPq (302531/2016-5), Brazilian funding agencies.

REFERENCES

[1] International Technology Roadmap for Semiconductors, *ITRS 2015 Report*, 2015. [Online]. Available: http://www.itrs2.net/itrs-reports.html
[2] Massachusetts Institute of Technology, *Protecting Data*, 2017. [Online]. Available: http://web.mit.edu/infoprotect/docs/protectingdata.pdf
[3] Symantec, *Internet Security Threat Report*, 2016. [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf
[4] C. H. Gebotys and R. J. Gebotys, "A framework for security on NoC Technologies," in *ISVLSI*, 2003, pp. 113–117.
[5] Y. Hu, D. Müller-Gritschneder, M. J. Sepulveda, G. Gogniat, and U. Schlichtmann, "Automatic ILP-based Firewall Insertion for Secure Application-Specific Networks-on-Chip," in *INA-OCMC*, 2015, pp. 9–12.
[6] J. Sepulveda 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*, 2017, pp. 59–64.
[7] M. D. Grammatikakis *et al.*, "Security Effectiveness and a Hardware Firewall for MPSoCs," in *HPCC*, 2014, pp. 1032–1039.
[8] D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-NoCs: Mitigating the threat of a compromised NoC," in *DAC*, 2014, pp. 1–6.
[9] 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.
[10] T. Wehbe and X. Wang, "Secure and Dependable NoC-Connected Systems on an FPGA Chip," *IEEE Transactions on Reliability*, vol. 65, no. 4, pp. 1852–1863, Dec 2016.
[11] 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.
[12] Hemanth, *aes_crypto_core*, 2004. [Online]. Available: https://opencores.org/project,aes_crypto_core
[13] E. A. Carara, R. P. de Oliveira, N. L. V. Calazans, and F. G. Moraes, "HeMPS - a framework for NoC-based MPSoC generation," in *ISCAS*, 2009, pp. 1345–1348.
[14] 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.
[15] B. Oliveira, R. Reusch, H. Medina, and F. Moraes, "Evaluating the Cost to Cipher the NoC Communication," in *LASCAS*, 2018.
[16] 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.