Enabling Secure MPSoC Dynamic Operation through Protected Communication

Siavoosh Payandeh Azad*, Behrad Niazmand*, Gert Jervan*, Johanna Sepulveda†
**Tallinn University of Technology, Tallinn, Estonia* †*Technical University of Munich, Munich, Germany*
siavoosh.azad@ttu.ee, behrad.niazmand@ttu.ee, gert.jervan@ttu.ee, johanna.sepulveda@tum.de

Abstract—**High parallelism and flexibility have turned the Multi-Processors System-on-Chip (MPSoCs) into one of the key enabling technologies for new computational paradigms such as Internet-of-Things (IoT) and Machine Learning. Given the continuous and wide distribution of MPSoC devices together with the MPSoC utilization in critical and sensitive applications, data confidentiality and privacy are of utmost importance. However, with the growing complexity of MPSoCs, the risk of Malware infections and code injection at booting and operation time increases significantly. A persistent threat to MPSoCs security is the configuration of critical MPSoC infrastructures, such as security frameworks (on-chip firewalls and routers). In this work, we propose an infrastructure able to ensure secure booting and dynamic MPSoC operation. To this end, we present two contributions. First, we integrate lightweight security infrastructure based on firewalls and lightweight authentication. Second, we explore the design space of the security infrastructure and evaluate the impact on the overall performance and cost of the system. In contrast with previous works we explore the fine and coarse grained reconfiguration and show the impact of the location of the security manager in the overall system performance.**

I. Introduction

Multi-Processor System-on-Chips (MPSoCs) are considered as a driving force behind the digitalization in many critical applications and key industrial markets. MPSoCs integrate a set of heterogeneous IP hardware cores (e.g. single processor, hardware accelerators, peripherals or memory) which communicate through a Network-on-Chip (NoC). A NoC is composed by a set of routers and links to allow the data exchange. However, the use of MPSoCs in critical applications and key markets has turned them into targets of attacks. A compromised MPSoC operating in the context of Internet-of-Things (IoT) may reveal secret information (e.g., passwords, secret keys) or grant to an attacker the control over a critical infrastructure, thus, jeopardizing the entire IoT environment. In order to enable the wide adoption of digitalization, security of MPSoC is mandatory.

One of the most used techniques to attack MPSoCs is the code injection. This attack exploits two facts: i) MPSoC applications are stored in external non-volatile memories which are prone to malicious modifications; and ii) MPSoCs are able to support several applications, usually being executed simultaneously while sharing several resources, such as processors, memories and communication structure. By storing a malicious code (Malware) and executing into the MPSoC, an adversary may execute illegal operations to overwrite or read sensitive data. As a result an attacker may obtain complete or partial control over the system. Code injection attacks may compromise sensitive code and data or install new firmware. To avoid this kind of attacks, MPSoC should be able to spatially and temporally isolate sensitive applications. This isolation should be compliant with the security policy of the applications, a set of rules that specify the different access permissions. Security policies are implemented through a security infrastructure which is configured by the security manager, a secure IP core in the MPSoC.

Previous works have proposed the use of firewalls for enforcing the security policies. These firewalls allow the construction of continuous and disruptive security zones, thus, becoming a key building block for MPSoC security. However, most of the works deal with static firewalls. But security is dynamic and due to the data migration among the cores and remapping of applications on the MPSoC, firewalls should also be updated during MPSoC run-time. Moreover, the configuration of this security infrastructure, initiated by the security manager IP located inside the MPSoC, should be performed in a protected way. The configuration should include authentication for the security manager IP core's messages through fast and lightweight techniques. Only a few works have addressed the study of the performance impact on update-able firewalls. In contrast with the work of [1], the firewall updating due to security zone reconfiguration requires low granularity. To the best of authors' knowledge, none of the previous works have scrutinized the design space explorations considering different network distances between the security manager IP core location and the reconfigurable firewalls and the different reconfiguration packet lengths while evaluating the impact on the overall system performance. The novelties of this work are two-fold: i) integration of an infrastructure able to support the protected dynamically update of the security infrastructure of the MPSoC; and ii) the impact evaluation of the security manager location on the MPSoC performance.

The rest of the paper is organized as follows. Section II presents the previous work on firewall-based MPSoC protection and evaluation. Section III describes the MPSoC security through the use of firewalls. Section IV describes the security infrastructure and Section V presents the design space exploration and evaluation. Conclusions are presented in Section VI.

II. Related Work

Security in MPSoCs can be achieved by integrating firewalls. The works in [1]–[6] have implemented hardware firewalls in a NoC-based system in order to improve the security of the system. The MPSoC platform allows the parallel execution of several applications in the same system. The dynamic nature of applications during run-time, necessitates, in turn, the MPSoC firewalls to be dynamic in order to handle the different sets of security rules for different applications. Works such as [7]–[9] have implemented the firewalls with static security policies, which cannot address dynamic nature of current MPSoCs with different mapped applications and they only implement a single NoC security level for the entire system. On the other hand, works such as [3] take into account the dynamic mapping of application tasks to NoC nodes and dynamic reconfiguration of firewalls, or the run-time reshaping of security zones (for isolating sensitive communication) [10]. The work in [11] has introduced insertion of multi-level firewalls into NoC-based systems to satisfy security requirements. Despite the advantages the approach has, the work in [11] has only focused on the placement of hardware firewalls in application-specific NoCs. Moreover, the utilized model does not address dynamic configuration of firewalls. In the works such as [12]–[16], dynamic configuration of firewalls is explored, where a Secure Manager (SM) is in charge of (re-)configuring the firewalls and controlling the recovery mechanism under a possible attack. In [15], a secure hypervisor (or OS) is in charge of dynamic reconfiguration of firewall rules. However, the impact of the location and communication characteristics of the security manager on the overall performance of the system is not explored.

III. MPSoC SECURITY

Isolation at MPSoCs can be enforced into two structures: i) computation structure, by including secure scheduling techniques, using exclusive data-path for sensitive applications or by restricting the access of memory addresses that contain sensitive information (memory isolation); and ii) communication structure, at the router-level (avoiding communication collision) or at the network interface (enforcing the security policy through firewalls). It should be noted that these techniques are not exclusive, but they are complementary. A very common approach is to use the memory isolation in combination with firewalls.

Firewalls enforce the access control policy. These modules regulate admission to the IP cores and the external memory blocks by the different processes that are executed on the MPSoC. Each of the tasks are able to be reallocated during MPSoC run-time. This means that the contents that are stored in the firewalls are changing.

Each application supported by the MPSoC is characterized by different sets of security rules, called security policy. The set of applications can be mapped dynamically at the MPSoC. Therefore, there is not a single and static security requirement, but a set of ever changing security policies that must be satisfied. The challenge is to provide MPSoC security that allows a trustworthy system that meets all the security requirements of such applications. MPSoC can be attacked via hardware/software.

IV. SECURITY INFRASTRUCTURE

The code injection attack includes malicious pieces of code into the application to compromise the integrity or confidentiality of the MPSoC. In order to avoid code injection attacks SipHash-2-4 can be used [17] [18]. It is a fast and lightweight Message Authentication Code (MAC) generator to authenticate and guarantee the integrity of the packet. SipHash is recommended for short inputs. Each authorized application has a pre-shared key with the MPSoC which is used to generate the MACs (ret_MAC) for the different memory blocks. At the security manager, before executing an application, the data blocks are retrieved from the external memory and the MACs (comp_MACs) are computed. A match between ret_MAC and comp_MAC guarantees that the original code was not modified. In addition, as the key is only shared with the application, the authentication can be also performed.

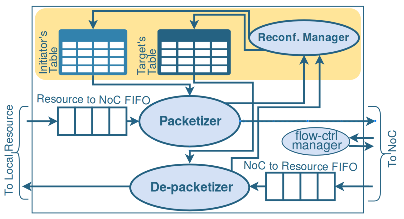


Fig. 1. Proposed firewall architecture

At the MPSoC, the security infrastructure in this work is composed of a security manager and a set of security interfaces that integrates a firewall. The security manager is a secure IP core which is in charge of implementing the security policy of the system. It generates the access control rules stored in the firewalls, distributed into the MPSoC. Firewalls check the information embedded into the packet and compare it with a set of access rules. A match packet will be able to access the requested resources. Otherwise, it is discarded and an alarm is triggered. Our firewall is implemented as two tables: i) *initiator table*, checking the rights of the communication initiator (at the source node); and ii) *target table*, checking the rights of the received packet at the destination node. Each entry in initiator table contains the following information:

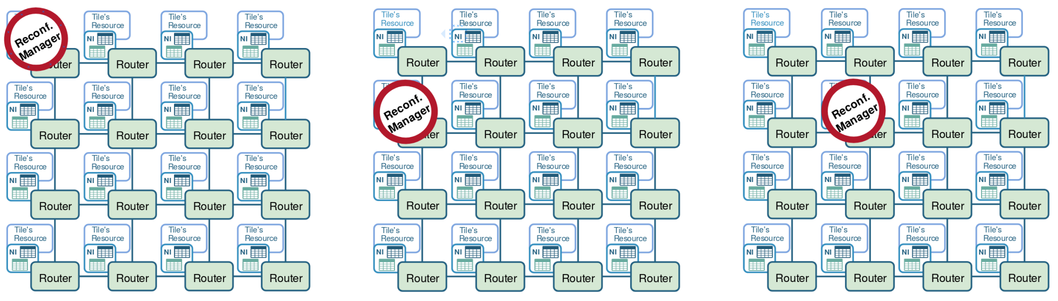
- **Destination address:** ID of allowed destinations for current node.
- **Base-Memory address:** base address in the destination node that can be accessed by current node.
- **Memory address range:** allowed memory range to be accessed by the current node.
- **Permission:** contains the access rights of the current node to the specified memory addresses in the destination node.

Similarly, the target table entries contain the following information:

- **Source address:** ID of the allowed sources which can access the current node.
- **Base-Memory address:** base address in the current node that can be accessed by the source node.
- **Memory address range:** allowed memory range to be accessed by the source node.
- **Permission:** contains the access rights of the source to the specified memory addresses in the current node.

Figure 1 shows the block diagram of the proposed architecture for the Network Interface (NI) firewalls (additional blocks integrated to the baseline NI are highlighted in orange). The security manager is in charge of issuing the reconfiguration commands to the network interfaces. Upon receiving the reconfiguration command from the security manager, the NI checks its authenticity and rights, and initiates a reconfiguration process (storing new values in the firewall table). SipHash [17] can be used for generating 64-bit MAC for such short messages such as in [19] [18]. Once all the firewalls in the network are configured, the system can start its normal operation. The security manager can initiate a full or partial dynamic reconfiguration of the firewalls during the system's operation.

The implementation of dynamic security has a strong impact on the system performance. Thus, the exploration of the design



(a) Avg. Distance = 3 (b) Avg. Distance = 2.5 (c) Avg. Distance = 1.875

Fig. 2. Different mapping scenarios for system's security manager

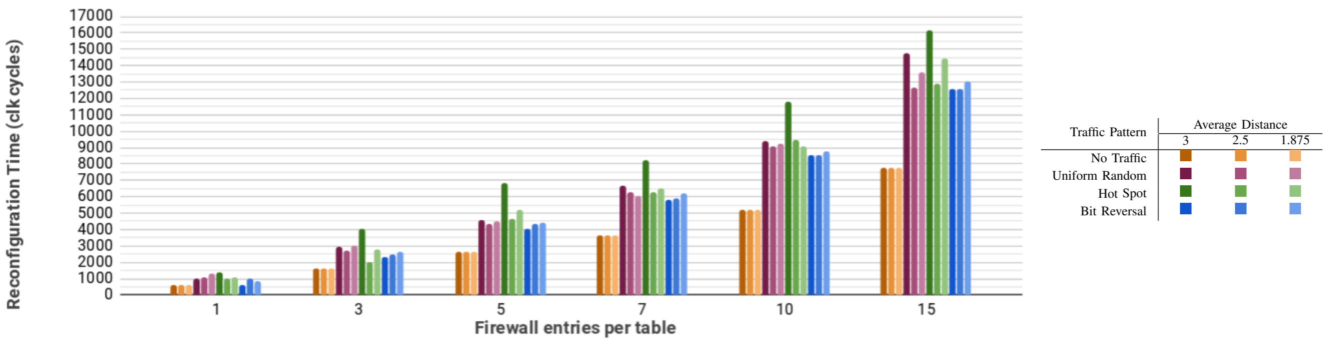


Fig. 3. Reconfiguration time evaluation for different number of locations in the firewall and different security manager placement

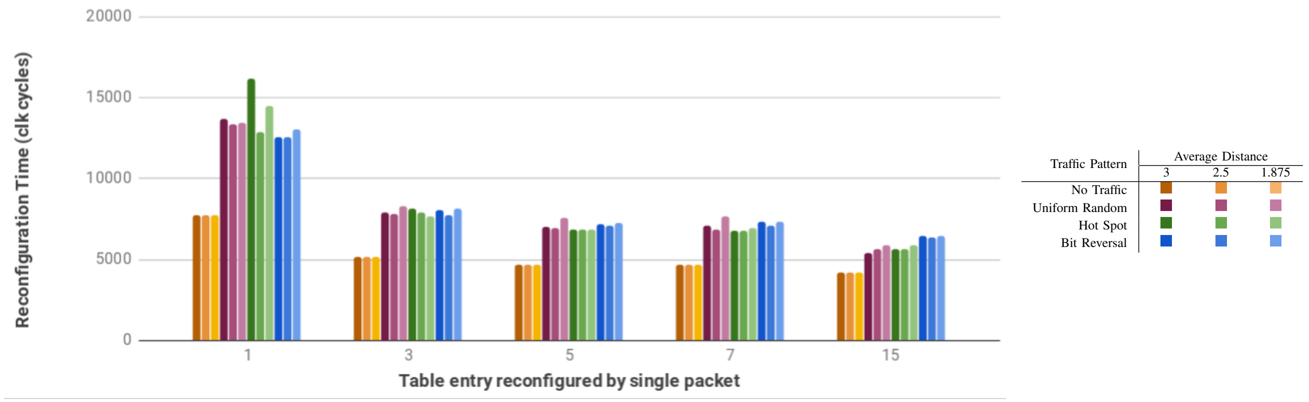


Fig. 4. Reconfiguration time for different reconfiguration packet size and different security manager placement

TABLE I: EXPERIMENTAL SETUP

Network Size	4X4
Routing algorithm	XY
Network FIFO depth	4 flits
Flow control	Credit-based, Wormhole switching
Network interface FIFO depth	32 flits
Network traffic packet size	8 flits
Network Traffic packet injection rate	0.03

space is mandatory. The design choices include: number of entries in the firewall, location of the security manager, number of firewall entries to be dynamically reconfigured and the number of firewall entries to be reconfigured by a single packet. The next section provides experimental results for each of the above mentioned design choices.

V. Experimental Work

In this section we evaluate the reconfiguration latency of the firewalls under different scenarios and the overhead of the proposed firewall mechanism on the overall system. Our experiments are based on the Bonfire NoC-based MPSoC [20] which uses a configurable NoC and NIs together with configurable traffic emulators. The configuration of Bonfire is summarized in Table I. Three different mapping scenarios for the security manager has been considered. They explore the average distance of the manager to other nodes in the network. The average distance of the manager can be defined as in (1).

$$Avg_{distance} = \frac{\sum_{i=0}^{N-1} shortest\_path(ID_M, ID_i)}{N} \quad (1)$$

Where N is the number of nodes in the network, ID_i designates the node identifier for the network tile i and M is id of the tile where the security manager tasks are mapped. Figure 2 shows the different mapping scenarios for the manager.

Figure 3 depicts the reconfiguration latency (in clock cycles) of different number of firewall entries in the network under different traffic scenarios (traffic-free NoC, uniform

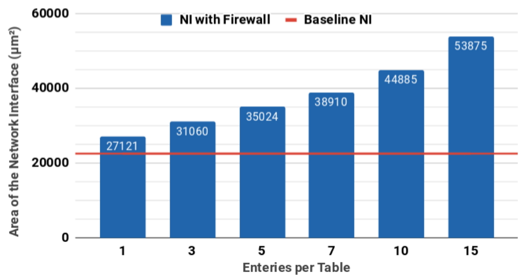


Fig. 5. Area of the Firewall equipped NI with number of firewall entries

random, hot-spot and bit-reversal). As expected, the required reconfiguration time increases with the growing number of firewall entries to be reconfigured in each table. However, results show that the placement of the security manager has a profound impact on the firewall reconfiguration time regardless of the size. For larger reconfiguration table sizes, the average distance of 2.5 presents the best reconfiguration time. While higher distances (3) causes a global contention on the MPSoC, extremely short distances (1.8) causes the communication isolation of a portion of the MPSoC during the reconfiguration. Both approaches (high and extremely low) impact heavily the system performance. For bigger MPSoCs, the integration of several security managers is recommended in order to moderate the impact on the system performance. For networks with size of 8×8, 12×12 and 20×20 network, 4, 9 and 25 managers are required to keep the 2.5 average distance, respectively.

Figure 4 explores the reconfiguration time required for the entire firewall table with regards to the number of the table entries that a single packet can reconfigure. Results show that the larger reconfiguration packets can in fact reconfigure the entire network with lower latency.

Overhead analysis: The design for baseline NI (without any security mechanism) and the secure NI augmented with dynamic re-configurable firewall, are implemented at Register Transfer Level (RTL) in VHDL. The designs are synthesized using TSMC $40nm$ CMOS technology standard cell library in Synopsys Design Compiler. Figure 5 depicts the increase in the secure NI with firewall's area based on the number of firewall entries (139% for firewall with 15 entries). By adding the firewall, the critical path delay of the baseline NI will increase by 6.46% from 2.32 ns to 2.47 ns. However for a $400mm^2$ chip, the integration of Firewall equipped NIs only imposes 0.2% overhead. Integration of SipHash [17] for providing MAC for reconfiguration packets, imposes an estimated 17105 μm^2 (normalizing results reported in [19] to 40 nm technology) to the Network interface. Addition of MAC using SipHash technique imposes an additional 0.06% overhead to a $400mm^2$ chip with 16 firewall equipped NIs.

VI. Conclusions

NoC firewalling has emerged as a practical security measure for the MPSoCS in the age of IoT, where the systems are subject to code injection attacks. This paper proposes a lightweight security infrastructure able to avoid code injection attacks, which can compromise the booting and dynamic operation of MPSoC. Our work provides a design space exploration for the placement of the system's security manager, along with different reconfiguration schemes of the firewall entries under different network traffic patterns. Our results show that the average distance between the security manager and the firewalls have a huge impact on the overall system performance. While shorter distances are recommended for small reconfiguration messages, for big messages extremely small average distances may isolate a portion of the MPSoC during the reconfiguration time, thus, heavily compromising the system performance. For bigger MPSoCs, the integration of several security managers distributed on the systems is recommended.

References

[1] M. D. Grammatikakis *et al.*, "Security in MPSoCs: A NoC firewall and an evaluation framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1344–1357, 2015.
[2] A. B. Achballah *et al.*, "Fw_ip: A flexible and lightweight hardware firewall for NoC-based systems," in *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, 2018.
[3] J. Sepúlveda *et al.*, "Dynamic NoC-based architecture for MPSoC security implementation," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design*. New York, NY, USA: ACM, 2011, pp. 197–202.
[4] R. Fernandes *et al.*, "A non-intrusive and reconfigurable access control to secure nocs," in *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2015, pp. 316–319.
[5] L. Fiorin *et al.*, "A security monitoring service for NoCs," in *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '08. New York, NY, USA: ACM, 2008, pp. 197–202.
[6] J. P. Diguet *et al.*, "NoC-centric security of reconfigurable SoC," in *First International Symposium on Networks-on-Chip (NOCS'07)*, 2007.
[7] L. Fiorin *et al.*, "Security aspects in networks-on-chips: Overview and proposals for secure implementations," in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, 2007, pp. 539–542.
[8] S. Evain *et al.*, "From NoC security analysis to design solutions," in *IEEE Workshop on Signal Processing Systems Design and Implementation, 2005.*, 2005, pp. 166–171.
[9] L. Fiorin *et al.*, "Implementation of a reconfigurable data protection module for NoC-based MPSoCs," in *2008 IEEE International Symposium on Parallel and Distributed Processing*, 2008, pp. 1–8.
[10] J. Sepúlveda *et al.*, "Efficient and flexible NoC-based group communication for secure MPSoCs," in *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2015, pp. 1–6.
[11] Y. Hu, *et al.*, "Automatic ilp-based firewall insertion for secure application-specific networks-on-chip," in *2015 Ninth International Workshop on Interconnection Network Architectures: On-Chip, Multi-Chip*, 2015, pp. 9–12.
[12] J. Sepúlveda *et al.*, "A security-aware routing implementation for dynamic data protection in zone-based MPSoC," in *2017 30th Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2017, pp. 59–64.
[13] G. G. J. Sepúlveda *et al.*, "Hierarchical NoC-based security for MPSoC dynamic protection," in *2012 IEEE 3rd Latin American Symposium on Circuits and Systems (LASCAS)*, 2012, pp. 1–4.
[14] L. Fiorin *et al.*, "Secure memory accesses on networks-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1216–1229, 2008.
[15] M. D. Grammatikakis *et al.*, "High-level security services based on a hardware NoC firewall module," in *2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2015, pp. 73–78.
[16] J. Sepúlveda *et al.*, "QoSS hierarchical NoC-based architecture for MPSoC dynamic protection," *Int. J. Reconfig. Comput.*, vol. 2012, pp. 3:3–3:3, 2012.
[17] J.-P. Aumasson *et al.*, "SipHash: A fast short-input PRF," in *Progress in Cryptology - INDOCRYPT 2012*, S. Galbraith and M. Nandi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
[18] J. Seplveda, A. Zankl, D. Flrez, and G. Sigl, "Towards protected mpsoc communication for information protection against a malicious noc," *Procedia Computer Science*, vol. 108, pp. 1103 – 1112, 2017, international Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.
[19] J. Sepulveda *et al.*, "SEPUFSoC: Using PUFs for memory integrity and authentication in multi-processors system-on-chip," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '18. New York, NY, USA: ACM, 2018, pp. 39–44.
[20] "Project Bonfire," https://github.com/Project-Bonfire, 2016.