A Non-intrusive and Reconfigurable Access Control to Secure NoCs

Ramon Fernandes[1], Bruno Oliveira[1], Johanna Sepúlveda[2], Cesar Marcon[1], Fernando G. Moraes[1]
[1]FACIN - PUCRS – Av. Ipiranga 6681, 90619-900, Porto Alegre, Brazil
[2]Institute for Security in Information Technology, Technical University of Munich, Munich, Germany
{ramon.fernandes, bruno.scherer}@acad.pucrs.br , johanna.sepulveda@tum.de,{cesar.marcon, fernando.moraes}@pucrs.br

***Abstract*—Following the current trend in the semiconductor industry (MPSoC) and the massive advances presented by all things interconnected (Internet of Things), a massive quantity of private and metadata is being transferred through insecure channels. In the industry, almost no attention is given to the amount of data that can be collected from different individuals, just by getting access to their house appliances. With that in mind, this paper proposes a non-intrusive and reconfigurable access control architecture for Networks-on-Chip (NoCs). This architecture comprises firewalls, which are capable of filtering both incoming and outgoing network traffic by analyzing packet information and verifying a traffic initiator's access permission, providing a secure environment that is capable of protecting the user data. The firewalls have approximately 12% of the router area. When the number of routers increases, the firewall area overhead grows slightly, up to only 16% in NoCs with 64 routers.**

***Keywords*—NoC; MPSoC; Information Security; Non-Repudiation; Authenticity; Availability.**

I. Introduction

Advances in manufacturing technology of Integrated Circuits (ICs) and the growth in the number of transistors per area allows implementing complete system functionality into a single IC, which is called a System-on-Chip (SoC). A SoC may also be called a Multiprocessor SoC (MPSoC) when multiple Processing Elements (PEs) and Intellectual Property (IP) modules are integrated employing a communication infrastructure. The International Technology Roadmap for Semiconductors (ITRS) foresees thousands of PEs integrated into a SoC by 2020 [1]. Therefore, there is a growing need for a reliable and scalable communication architecture. The Network-on-Chip (NoC) paradigm provides high scalability and communication parallelism [2][3] and has been widely adopted [4] as a viable interconnection mechanism for MPSoCs.

In the early 2010's, MPSoC became a trend in the industry as a major design solution for the increasing demands in low power needs and functional requirements [5]. The MPSoC became a power player in the mobile computing world. A new challenge is to thrive the cell phone as the central hub for all connections. The Internet of things [6] is a new design trend for the industry. Hence, to keep secure the data that navigates through these devices, a new challenge is created.

As the adoption and complexity of such systems increases for embedded systems, so does the concern for security [7]. An MPSoC system may be used in scenarios where availability is a critical factor and downtimes must be avoided. These systems also handle sensitive information and as such, it is important to protect this data from unauthorized access. While software-based attacks account for 80% of security incidents in embedded systems [8], often using abnormal communication, the system may also be prone to hardware attacks that compromise its security.

Software attacks include elements such as viruses, worms, and Trojan horses, often-exploiting weaknesses in code structure, such as buffer overflow attacks [9]. NoCs are vulnerable to attacks that exploit its specific structure, consisting of three general types [10]:

- *Denial of Service (DoS)*, which degrade the performance of the system by overloading the communication's architecture, regularly generating useless packets in the system. These attacks can be also classified as *Bandwidth Reduction attacks*, or if an embedded system contains a limited power source, as a *Draining* or *Sleep Deprivation attack*;
- *Extraction of secret information* that aims to access sensitive data, often through buffer overflow attacks or similar methods that exploit software weaknesses;
- *Hijacking attacks*, which attempt to alter the system configuration to perform a set of foreign tasks along with normal system operation.

As most SoCs are customized for different application requirements and development constraints, the applicability of security measures may vary greatly, and concrete considerations should be taken. In [11], the authors propose the integration of a security module in the Network Interface (NI) of a shared memory MPSoC, where hardware firewalls use lookup tables to grant the CPUs access to memory related operations, protecting the NoC from malicious code. The work of [12] uses reprogrammable hardware Data Protection Units (DPUs) managed by a centralized unit for protecting data from unauthorized access. The DPUs are reconfigured at runtime, and traffic initiators are filtered based on the specified memory address and requested operation.

Hardware and software elements are used in [13] to split the system while using Secure Channels and a Secure Kernel for managing keys and access rights of tasks to system resources, forming a Trusted Computing Base (TCB). Their claim is that as long as the TCB is not compromised, the overall system security is guaranteed. The authors in [14] propose a hybrid-on-chip communication structure, which monitors system behavior and uses heuristics for blocking malicious traffic while providing secure data exchange among tasks with cryptographic mechanisms.

This work *proposes* a hardware-based firewall unit for NoC-based SoCs enabling two properties of secure systems: *confidentiality*, as only authenticated sources may access restrict resources in the system, and *nonrepudiation*, by guaranteeing that a traffic initiator cannot forge a traffic origin. The reconfigurable firewall, placed between the NI and the router, monitors the incoming network traffic, authorizing it to enter the designated NI,

978-1-5090-0246-7/15/$31.00 ©2015 IEEE

and verifies if the traffic initiator does not forge its source address, which would compromise the authentication process.

II. Secure NoC Architecture

Our proposed firewall architecture monitors both incoming and outgoing traffic in NoC-based SoCs by filtering the packet header information. We assume that communication in a NoC requires packet headers with both source and destination addresses, used for validating communication events by the firewalls, as shown in Fig. 1.

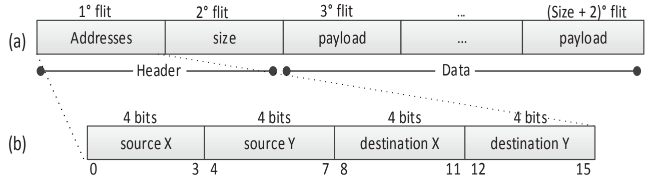

Fig. 1 – Example of a NoC's packet composition for a 16-bit flit.

Consider an $m \times n$ mesh NoC, with N_s routers, where $N_s = m \times n$. Each firewall contains an access register A_r of size N_s for determining which addresses in the NoC can access a given PE. A single bit A_b indicates the access permissions for each source PE in the NoC, which can be either *allowed* (the bit is set) or *disallowed* (the bit is not set). The firewall is placed between the NI and the local port of the routers (the local port is the one that connects the router with the local PE). Once a packet arrives at the local port of a given router, the firewall checks the *(x,y)* coordinates of the source address and determines access permissions of the packet by verifying $A_r(x,y)$. If the source address has permission to access the PE, then the firewall directs the packet to the destination NI. Otherwise, the firewall discards the incoming packet and denies the communication.

Another function executed by the firewall is checking packet validity upon injection by a source PE. Each firewall contains an internal register F_{addr} with its respective address in the NoC. Upon packet injection, the firewall verifies if the header source information matches the address of the source PE. If a malicious application forges the source information, the firewall is capable of denying the packet from entering the NoC, therefore offering protection from impersonation attacks (*nonrepudiation*).

Since each firewall monitors both directions of the communication, their placement requires manipulation of the router and NI control signals to filter the communication events. However, it is not necessary for the firewalls to store and forward packets, thus reducing the implementation complexity. This approach turns the implementation *non-intrusive*, as no elements of the NoC, other than signal mapping, need to be modified to accommodate the firewalls.

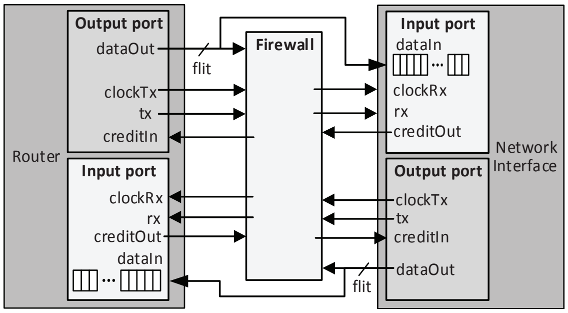

Fig. 2 – NoC port interface and firewall interconnection.

Fig. 2 shows the interface for a credit-based protocol NoC with bidirectional links. Each input port has three control and one data signals: (*i*) *clockRx* for data synchronization; (*ii*) *rx* for data availability signaling; (iii) *creditOut* for indicating buffer availability; and (*iv*) *dataIn*, which carries the receiving data. The output port employs the counterpart control/data signals: (*i*) *clockTx*; (*ii*) *tx*; (*iii*) *creditIn*; and (*iv*) *dataOut*. The firewall has access to all control signals involved in the communication, as well as the data signals since it requires checking the headers of incoming and outgoing packets.

Configuring the firewall requires an independent and presumed secure circuit separated from the NoC communication infrastructure. We use a serialized Hamiltonian path for interconnecting the firewalls to a safe PE responsible for their configuration, as Fig. 3 illustrates.

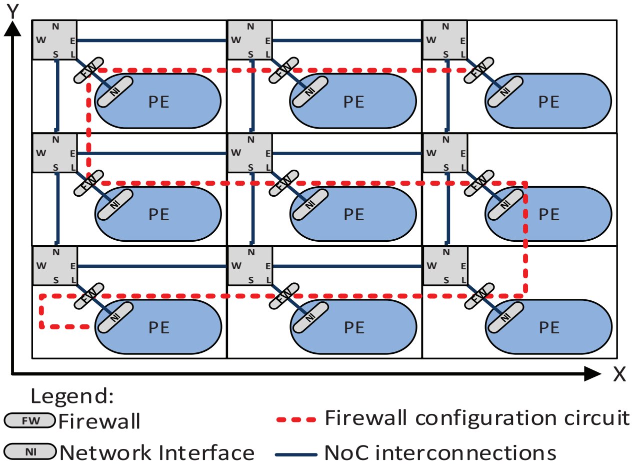

Fig. 3 – Firewall placement in a mesh 3×3 NoC.

Firewall rules navigate the configuration circuit until arrival at the designated firewall module. Configuration information requires three clock cycles for propagation from one firewall to another, where each cycle corresponds to: (*i*) *x* coordinate of target firewall; (*ii*) *y* coordinate of the target firewall; (*iii*) the $A_r(x,y)$ index of the register information at the targeted firewall for configuration. The A_b value uses a dedicated signal for setting the A_r value on the target firewall.

Forwarding incoming and outgoing packets require three additional clock cycles, as the firewall unit must check the packet header (1st flit) and packet size (2nd flit), and verify the value of A_b in A_r or F_{addr}, depending on the traffic direction. Only after these steps, the control signals are set for handling the packet and the remainder of the communication occurs in a pipelined manner. Discarding a packet requires only two clock cycles, as synchronization between the router and the NI is not necessary. In these situations, the firewall sets the *creditIn* signal for the output ports while not setting the *rx* signal at the input ports, consuming the entire packet. This step is necessary to prevent deadlock situations where an unauthorized packet would otherwise remain in the routers or NIs indefinitely. This flow control mechanism requires an additional register in the firewall for storing the packet size.

The firewall area is a function of the NoC size. Each new router in the NoC adds one bit in A_r, which is a small cost in terms of area, leading to a scalable and secure solution. Assuming a 16-bit flit size and a single flit containing the entire header information, we can address up to 256 PEs (4 bits for the *X* coordinate and 4 bits for the *Y* coordinate). Since each entry in A_r corresponds to a single PE address, then $0 \leq |A_r| \leq 256$ bits.

III. Experimental Setup

The firewall is validated by simulating its behavior under synthetic traffic scenarios using the public available Hermes NoC [15], a 2D packet-switching credit-based mesh NoC. SystemC modules attached to the NoC local ports simulate PEs communicating by injecting and receiving traffic from different source and destination pairs. The correct behavior of the firewall is verified by analyzing different application traces with both allowed and disallowed communication events for various system configurations.

Fig. 4 illustrates traffic flows for evaluating the firewall operation. Allowed flows represent communication where the initiator has access permissions at the target PE. The forbidden flows generate traffic from a PE that has no access rules defined at the designated target, therefore violating security permissions set on the firewall. Malicious traffic combines allowed traffic flows, but we assume a compromised initiator PE that attempts to impersonate another, which is known to have access permissions at the target PE.

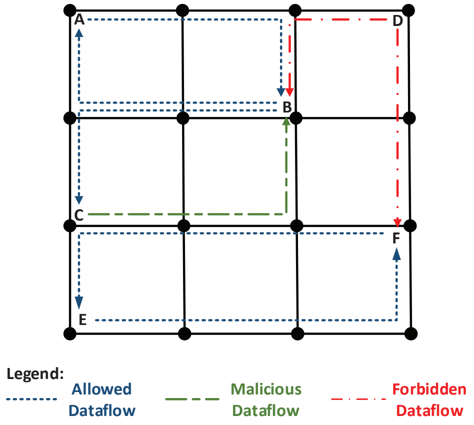

Fig. 4 – A 4×4 mesh NoC with allowed, malicious and forbidden traffic flows.

IV. Results

The NoC and the firewalls are implemented in VHDL. All results were obtained by simulation using Mentor ModelSim. For area consumption estimation, we use Encounter RLT Compiler, where we synthesize the VHDL hardware description for the NoC and firewalls, using STMicroelectronics standard-cells CMOS 65 nm technology (CORE65GPSVT library).

A. Access Control

The tested application contains six communicating nodes in a 4×4 mesh NoC with different access permissions. Table 1 illustrates the communication pairs used in the test application, as well as the allowed communication flows. We list only the explicitly allowed and forbidden communication pair situations and assume that any other traffic condition is denied.

In this application, node C generates malicious traffic as it attempts to impersonate node A, which has access permissions at target B. Source node C originates legitimate traffic since it contains access permissions at node B. This traffic situation exemplifies *Hijacking attack*, where a PE executes a malicious code along with legitimate tasks to evaluate if the firewall blocks the outgoing traffic with incorrect source information. Node D attempts to generate a *Denial of Service attack* by flooding nodes B and F with useless packets.

Table 1 – Application access permissions.

Source	Destination	Communication
A	B	Allowed
B	A	Allowed
B	C	Allowed
C	B	Allowed
D	B	Forbidden
D	F	Forbidden
E	F	Allowed
F	E	Allowed

All communication events consist of packet exchanges among each PE. We compared the output traces of each PE and analyzed the communication traces to detect which packets traversed the NoC and arrived at its intended destination. Our approach was able to identify and discard all unauthorized packet exchanges for the tested application. Regarding node C, the legitimate traffic correctly reached node B during our tests, while the firewall discarded the malicious packets preventing it from entering the NoC. The traffic of node D traversed the NoC but never reached its intended destination PEs, because the firewalls discarded all of its packets.

B. Firewall Configuration Time

We consider node E (Fig. 4) as the secure element in this configuration and, therefore, this PE sets all access permissions of other firewalls during system initialization. Considering the Hamiltonian path used for transferring configuration data among firewalls, node A represents the worst case, requiring the longest configuration time. Equation 1 enables to estimate the setup time of node A (T_A) – remarks that the experiments use $m \times n$ mesh NoC.

$$T_A = 3 \times m \times n \times (m \times n - 1) \qquad (1)$$

T_A considers that propagating a configuration information requires three clock cycles, as discussed in Section II, and that each access permission for each other PE must be set individually. Node A also does not need to configure access permissions for itself. We can optimize this process if each firewall starts with an initial configuration of either enabling or disabling traffic for other PEs and then treating each configuration case individually. In this case, we can consider the set of PEs to be configured in node A as C_A, and Equation 2 enables to estimate the new configuration time of node A (T_A^*).

$$T_A^* = 3 \times m \times n \times |C_A| \qquad (2)$$

To configure the firewall of node A completely, in a 4×4 mesh NoC, T_A corresponds to 720 clock cycles. Consider that each firewall starts with all access permissions disabled for other PEs, and that node A communicates only with node B. In this case, configuring node A with the approach that uses T_A^* requires 48 clock cycles, a 93% reduction in the configuration time compared to a complete configuration.

C. Area Consumption Evaluation

Table 2 illustrates the required area for a 4×4 NoC with firewalls. The NoC area includes all connections, routers, and firewalls. Router contains both total and average area consumption among all routing elements since each router is optimized according to its position on the NoC (buffers in border routers are suppressed). The firewall area consumption represents the total area of a firewall and its average consumption since the synthesis may optimize each module differently.

Routing units represent 88.8% of the NoC's area, considering that there is 16 routers in a 4×4 mesh NoC. A single firewall represents 13.91% of a router area while all firewalls combined account for 12.4% of the entire NoC area consumption.

Table 2 – 4×4 NoC area consumption.

	# Cells	Cell Area (µm²)	Total Area (µm²)
NoC (Total)	**47889**	**314450**	**546613**
Router (Total)	**41086**	**284309**	**485423**
Firewall (Total)	**6803**	**30140**	**67559**
Router (Average)	**2568**	**17769**	**30338**
Firewall (Average)	**428**	**1893**	**4222**

D. Scalability Evaluation

Table 3 evaluates the scalability of the proposal for different NoC sizes. The firewall area is dominated by its control logic (a finite state machine), representing an area overhead around of 13% compared to the baseline router. It is important to note that increasing the NoC size also increases the A_r register size. As shown in Table 3, the area overhead grows 3.5% from a NoC with 9 routers compared to a NoC with 64 routers. *Therefore, we conclude the method is scalable, and may be adopted in large NoCs*.

Table 3 – Scalability evaluation.

NoC Size	Original NoC area (µm²)	NoC with firewall area (µm²)	Area overhead
3×3	**250353**	**283579**	**13.27%**
4×4	**485420**	**546613**	**12.61%**
5×5	**798188**	**912382**	**14.31%**
6×6	**1185973**	**1367764**	**15.33%**
7×7	**1650422**	**1918890**	**16.27%**
8×8	**2191409**	**2559044**	**16.78%**

Fig. 5 shows the firewall configuration delay for the worst case (T_A), as discussed in Section IV.B. The time required for fully configuring the firewalls increases with the NoC size. In practice, each PE communicates with few other PEs. Therefore, the configuration time can be easily optimized adopting the proposal to start the system with all communications denied, configuring only the flows specified by the applications that will execute in the system.

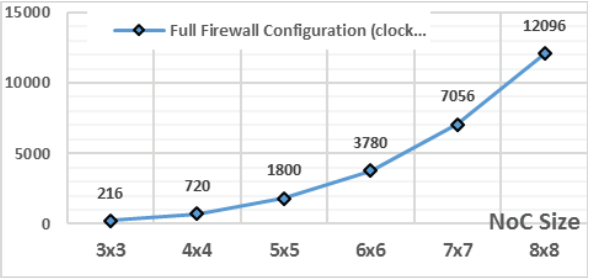

Fig. 5 – Firewall configuration time (worst case) evaluation.

V. CONCLUSIONS

In this work, we proposed the implementation of firewalls for providing secure access to resources of NoC based systems. As information security and availability become an ever-growing concern for embedded systems, so does the need for optimized mechanisms capable of protecting critical data and systems. Our firewalls provide a reasonable tradeoff in terms of area consumption for mesh-based NoCs while correctly filtering low-level flows for increased data access security.

Besides, the firewalls present only low-level access control because application context protection is delegated to an MPSoC operating system, as the firewalls only consider PE addresses for packet filtering. We consider that the MPSoC operating system handles the configuration of each firewall according to the current application requirements.

Further improvements could be made in the firewalls for evaluating communication at the application level and considering multiple applications running on the same PE; however, at the cost of implementation complexity.

ACKNOWLEDGMENTS

The Author Fernando Moraes is supported by CNPq - projects 472126/2013-0 and 302625/2012-7, and FAPERGS - project 2242-2551/14-8.

REFERENCES

[1] International Technology Roadmap for Semiconductors (ITRS). ITRS 2013 Edition. Available in: www.itrs.net/reports.html (Captured in Jun. 2015).
[2] L. Benini, G. De Micheli. "Networks on Chips: A New SoC Paradigm". Computer, vol.35(1), pp. 70-78, 2002.
[3] A. Ahmed, A. Abdallag, K. Kuroda. "Architecture and Design of Efficient 3D Network-on-Chip (3D NoC) for Custom Multicore SoC". In: International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA), pp. 67-73, 2010.
[4] E. Salminen, A. Kulmala, T. Hämälänien. "On network-on-chip comparison". Euromicro Conference on Digital System Design (DSD), pp. 503-510, 2007.
[5] P. Greenhalgh. "big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7". ARM, 2011.
[6] ITU-T Y.2060 – 06/2012: Overview of the Internet of things.
[7] S. Baron, M. Wangham, C. Zeferino. "Security Mechanisms to Improve the Availability of a Network-on-Chip". In: ICECS, pp. 609-612, 2013.
[8] J. Sepúlveda, G. Gogniat, D. Flórez, J. Diguet, C. Zeferino, M. Strum. "Elastic Security Zones for NoC-Based 3D-MPSoCs". In: ICECS, pp. 506-509, 2014.
[9] L. Fiorin, C. Silvano, M. Sami. "Security Aspects in Networks-on-Chips: Overview and Proposals for Secure Implementations". In: DSD, pp. 539-542, 2007.
[10] S. Evain, J. Diguet. "From NoC security analysis to design solutions". In: IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS), pp. 166-171, 2005.
[11] M. Grammatikakis, K. Papadimitriou, P. Petrakis, A. Papagrigoriou, G. Kornaros, I. Christoforakis, M. Coppola. "Security Effectiveness and a Hardware Firewall for MPSoCs". In: IEEE International Conference on High Performance Computing and Communications (HPCC), pp. 1032-1039, 2014.
[12] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, C. Silvano. "Secure Memory Accesses on Networks-on-Chip". IEEE Transactions on Computers, vol.57(9), pp. 1216-1229, 2008.
[13] H. Isakovic, A Wasicek. "Secure Channels in an Integrated MPSoC Architecture". In: IEEE Industrial Electronics Society (IECON), pp. 4488-4493, 2013.
[14] J. Sepulveda, G. Gogniat, R. Pires, W. Chau, M. Strum. "Hybrid-on-Chip communication architecture for dynamic MP-SoC protection". In: SBCCI, 6p, 2012.
[15] F. Moraes, N. Calazanz, A. Mello, L. Möller, L. Ost. "Hermes: an infrastructure for low area overhead packet-switching networks on chip." Integration, the VLSI Journal, vol.38(1), pp. 69-93, 2004.