A Security Monitoring Service for NoCs

Leandro Fiorin[†] Gianluca Palermo[‡] Cristina Silvano[‡]

[†]ALaRI - Faculty of Informatics - University of Lugano
Via Buffi 13, 6904, Lugano, Switzerland

[‡]Politecnico di Milano - Dipartimento di Elettronica e Informazione
Via Ponzio 34/5, 20133, Milano, Italy

fiorin@alari.ch {gpalermo,silvano}@elet.polimi.it

ABSTRACT

As computing and communications increasingly pervade our lives, security and protection of sensitive data and systems are emerging as extremely important issues. Networks-on-Chip (NoCs) have appeared as design strategy to cope with the rapid increase in complexity of Multiprocessor Systems-on-Chip (MPSoCs), but only recently research community have addressed security on NoC-based architectures.

In this paper, we present a monitoring system for NoC based architectures, whose goal is to help detect security violations carried out against the system.Information collected are sent to a central unit for efficiently counteracting actions performed by attackers.We detail the design of the basic blocks and analyse overhead associated with the ASIC implementation of the monitoring system, discussing type of security threats that it can help detect and counteract.

Categories and Subject Descriptors

C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures (Multiprocessors)—*Interconnection architectures (e.g., common bus, multiport memory, crossbar switch)*; C.1.4 [**Processor Architectures**]: Parallel Architectures—*Distributed architectures*; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls, Authentication*

General Terms

Design, Security

Keywords

Network-on-Chip (NoC), MultiProcessor System-on-Chip (MPSoC), Security, Embedded Systems

1. INTRODUCTION

The level of integration that silicon technology has reached in the past few years allows the use of advanced design processes for enabling applications that were to date infeasible. The complexity of new systems brings the challenge of enabling reliable communication channels between cores. Traditional solutions for inter-core communication will be soon unable to guarantee sufficient levels of efficiency, both from performance and power consumption perspectives. Networks-on-Chip [2] have appeared as a strategy to connect and manage the communication between the several design elements and IP blocks required in complex Systems-on-Chip (SoCs).

Security in systems adopting the NoC paradigm has been only recently addressed by the community [9, 8, 5, 7, 6]. However, security-aware design of communication architectures is becoming a necessity in the context of the overall embedded SoC/device security. Complex communication infrastructure such as NoC may lead to new weaknesses in the system that can be critical and should be carefully studied and evaluated. On the other hand, NoCs can contribute to the overall security of the system, providing the ideal mean for monitoring system's behaviour and detecting specific attacks [8, 5].

In this work, for the best of our knowledge, we detail for the first time a security monitoring system for NoC based architectures. Aim of the proposed system is to detect security violations in the device and to help counteract them by monitoring accesses to specific addresses in memory-mapped systems and deviations from expected NoC and system behaviours.

While monitoring of NoCs has been proposed for debugging and optimal resources utilization [3, 15, 16], monitoring for security purposes has been only suggested and outlined [8, 5]. In this paper, we discuss the basic blocks composing a security monitoring system. We detail overhead associated with their implementation, and types of attacks detected by the system. Probes, collecting information about the NoC traffic, are implemented inside OCP/IP [1] compliant Network Interfaces (NIs). In fact, NIs represent the ideal position where performing analysis of incoming traffic and discard malicious requests. A central unit is in charge of collecting security alerts and decide appropriate countermeasures.

The remainder of this paper is organized as follows. Section 2 reviews related work and discusses motivations. Section 3 presents an overview of the proposed NoC monitoring architecture, while Section 4 provides implementation details of the security probes. Section 5 presents synthesis results for the discussed basic blocks. Finally, Section 6 presents conclusions and future work.

CODES+ISSS'08, October 19–24, 2008, Atlanta, Georgia, USA.

2. RELATED WORK AND MOTIVATIONS

While NoC has been subject of several studies and evaluations, security aspects related to NoC implementations has been only recently addressed. [9] presents a framework to secure the exchange of cryptographic keys within a NoC, addressing in particular the protection from power/EM attacks of a system containing not-secure cores as well as secure ones. Diguet et al. [5] propose a first solution to secure a reconfigurable SoC based on NoC. The system is composed of *Secure Network Interfaces* (SNIs) and a *Secure Configuration Manager* (SCM). The SNIs act as filter for the network and as monitor for the incoming traffic, while the SCM configures system resources and network interfaces.Fiorin at al. [6, 7] discuss data protection in NoC-based Multiprocessor systems, proposing the use of hardware modules within the NIs to check and limit access rights of processors requesting access to data locations in a shared memory. A central unit is in charge to manage their configuration.

Considering related work on monitoring of NoCs, in [3, 16] methodologies for debugging NoC-based SoCs are presented. Authors propose NoC run-time monitoring systems composed of configurable monitoring probes attached to NIs [16] or to routers and NIs [3]. They discuss associated programming models, and monitoring traffic management strategies. Several design alternatives for NoC monitoring systems are discussed in [4], in particular focusing on the interconnection of the monitoring resources, while in [15] link utilization is monitored to implement a strategy for controlling congestion in on-chip networks.

The use of monitoring for security purposes was suggested in [8, 5]. However, our work represents a first attempt to discuss in detail characteristics, implementation costs, and trade-offs of including monitoring for security in NoCs. With respect to the work above presented related to security on NoCs, the subject of our paper can be considered orthogonal and complementary, since we investigate a solution for the specific problem of monitoring attempts of attack in NoCs. While focus in previous work is on protection of specific types of attacks, the subject of this paper is a step forward towards the realization of a security solution at system level. The subject of our paper is also complementary to the previous work on monitoring. In fact, while some of the concepts developed for testing and debugging can be employed in our case, monitoring for security purposes presents unique challenges related to the implementation of "intelligent probes", able to detect and signal possible security threats for the system.

In this paper, we illustrate our solution for helping detect attacks aiming at retrieving sensitive information from the system or at causing Denial-of-Service (DoS). Protection of critical data represents a challenging task in multiprocessor Systems-on-Chip, where blocks of memory are often shared among several IPs. Unauthorized access to data and instructions in memory can compromise the execution of programs running on the systems or cause the acquisition of critical information by external entities [6].

As we will discuss in Section 4, while a hardware protection strategy can help limit unauthorized accesses to protected memory blocks [6], attempts of illegal access must be necessarily monitored to make the system aware of potentially dangerous behaviours. In fact, a compromised core executing malicious code can be used to perform a DoS attacks against the system, with the aim of reducing system performance and operative life of battery and device. Literature mainly identifies two types of DoS attacks on NoCs [8]: *Bandwidth Reduction*, where frequent and useless packets are inserted in the network in order to waste bandwidth and cause a higher latency in on-chip communications, up to the saturation of the network; *Draining Attacks*, aiming at reducing the operative life of a battery powered device by making the system executing power hungry tasks.

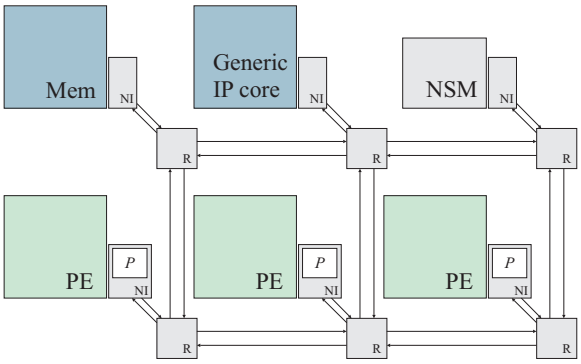

Figure 1: General NoC-based architecture including the security monitoring system

As example of DoS, a practical case that makes foreseen the possibility of performing draining attacks against NoCs (and more general SoCs) is represented by viruses for mobile phones recently appeared [12]. Currently malware are able to spread through Bluetooth connections or MMS (Multimedia Messaging Service) messages and infect recipients' mobile phones with copies of the virus or worm, hidden under the appearance of common multimedia files. If the malicious code is crafted in order to send continuous requests to the Bluetooth module for paging or scanning for devices, power consumption - if compared to the one consumed in the *Idle* state - could increase up to more than 500% [11], with a consequent significant reduction of battery lifetime. We believe that similar types of attacks, that would for instance show an unexpected traffic towards Bluetooth hardware modules, could be easily discovered observing system activities.

3. SYSTEM ARCHITECTURE

Figure 1 shows a general NoC architecture including the proposed monitoring system. We mainly refer to a NoC-based MPSoCs with shared memory, where all the cores in the system are memory-mapped. The monitoring system is mainly composed of three elements: *probes* (P), a *Network Security Manager* (NSM), and the *communication infrastructure* (NIs and Routers (R)).

Probes, whose implementation is described in Section 4, are located within NIs. The choice of embedding them inside NIs presents several advantages:

- traffic is analysed when inserted by the core, therefore there is not need of *sniffing* packets to retrieve the information necessary for threats detection.As a consequence, the logic to implement probes is less complex and expensive in term of area and power consumption than in the case of probes implemented at routers [3];
- monitoring is performed in parallel with operations performed by the NI kernel for the protocol transla-

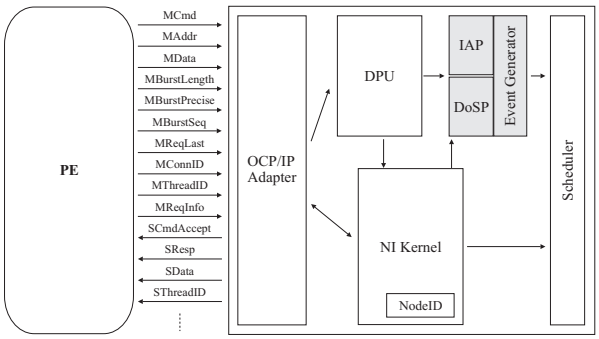

Figure 2: Architecture of the NI including the proposed probes

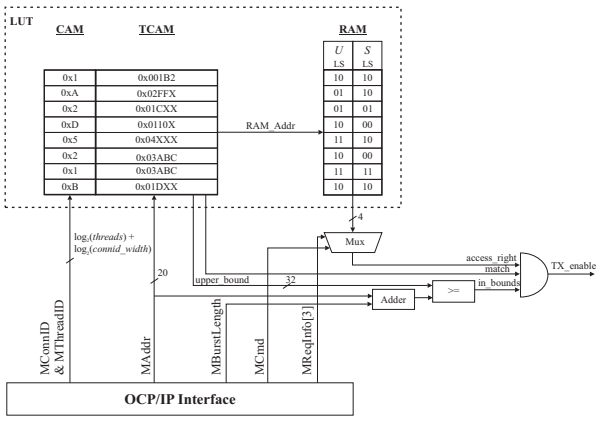

Figure 3: Architecture of the Data Protection Unit

tion from the OCP/IP interface. No overhead in performance is therefore associated to traffic analysis;

- being probes embedded in NIs, traffic detected as malicious can be stopped or limited, and the relative core considered as compromised by the system. This allows an easier identification of the source of security violations and, if necessary, the "quarantine" of the core (or the thread) by limiting its access to the NoC.

A dedicated core, the NSM, is in charge of collecting events and information coming from the several probes distributed in the system, analyse the data received, and counteract efficiently to the detected attacks. While focusing on hardware characteristics of the monitoring system, we leave to software designers the task of implementing detection strategies for security violations and appropriate countermeasures.

Traffic produced by probes is to be maintained separated from standard communication traffic inside the NoC. In order not to be sensitive to DoS attacks addressing NoC performance, the transmission of packets from probes must be guaranteed through the use of priority communication or guaranteed throughput services [14]. Moreover, differently from the case of on-chip debug, messages coming from probes should not be accessible by external entities through public interfaces, in order to avoid the exploitation of the information collected to attacks the system.

4. PROBES IMPLEMENTATION

In this Section, we first give an overview of the probes within the NI, discussing some of the concepts needed to understand their operation. In particular, we describe a hardware module for data protection (DPU), which is used in collaboration with the probes, and the concept of *Event*, useful to describe communications with the NSM. In the second part of the Section, we provide implementation details for the probes.

Figure 2 shows a NI embedding the probes we propose (in grey in the Figure). The *Illegal Access Probe* (IAP) detects attempts to illegally access restricted memory blocks or range of addresses in shared memory systems, while the *Denial of Service Probe* (DoSP) is employed to detect *Bandwidth Reduction* or *Draining Attacks*. The *Event Generator* is triggered by the two probes and generates the packets to be sent to the NSM to communicate the security violations.

4.1 DPU

Both probes rely on the presence of a Data Protection Unit[1] (DPU) [6] embedded inside the NI. The DPU is a hardware block enabling the access to a given memory space only if the initiator of the request is authorized to do the operation. Access filtering is performed by considering not only the memory address, but also type of operation requested (data *load*/*store*), and the status (*role*) of the initiator (*user* or *superuser* mode). Embedded in the initiator's NI, the DPU acts as a firewall in data networks, stopping requests not allowed in the targeted memory blocks or peripherals.

Access rights control is performed in parallel with the protocol translation, analysing OCP/IP transactions. As shown in Figure 3, information on OCP/IP signals are looked up to verify access rights. In particular, the source of the transaction is identified using a combination of the processor identifier (*MConnID*) and the thread identifier (*MThreadID*), while *MAddr* provides information about the targeted memory block. This information is looked up and access rights of the Load (L) and Store (S) operation for the two roles (User (U)) and Superuser (S)) of the initiator are provided as output of the lookup table (LUT) of the DPU. *MBurstLength*, which gives information about the length of the data to be transferred or received, is used to check if the dimension of the data are outside the block boundaries (*upper_bound*). *MCmd* identifies type of operation (*load*/*store*) on the memory address and together with information on the role of the initiator (*MReqInfo*) selects the desired signal. Therefore, in case access requirements are satisfied by the information present on the OCP/IP signals, a signal (*TX_Enable*) is risen to allow NI to send packets of the transaction through the network.

The most relevant part of the DPU is represented by the LUT, implemented in hardware by combining a Content Addressable Memory (CAM) and a Ternary CAM (TCAM) [13], and a RAM storing the access rights. The use of the TCAM allows to store range of addresses in the LUT and limit area occupied by the the module.

4.2 Events

Every probe generates events to notify the NSM security violations. As definition of event, we comply to what dis-

[1]Under patent pending - European Patent Application no. EP 07301411.0

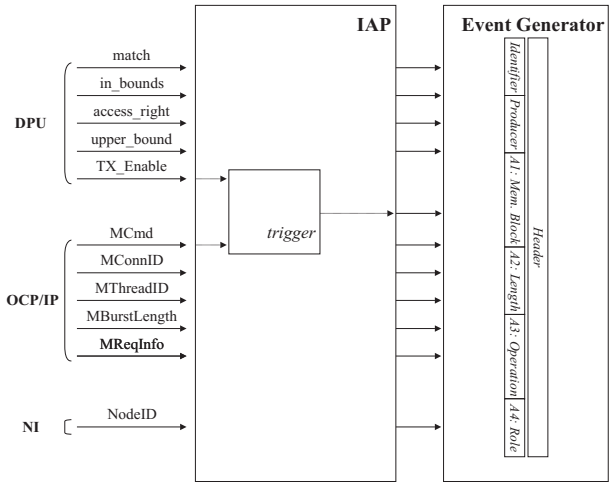

Figure 4: Illegal Access Probe: details

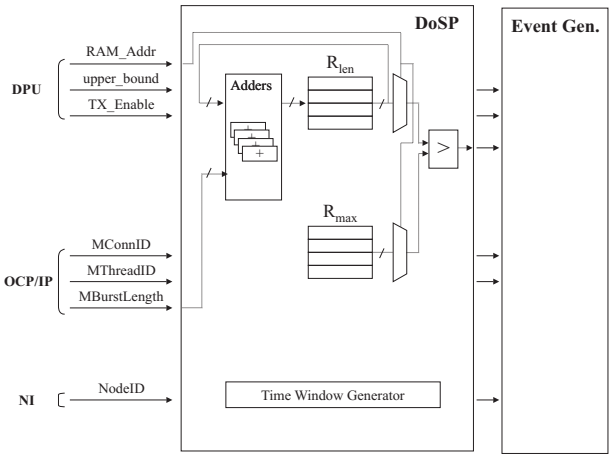

Figure 5: Architecture details of the DoSP

cussed in [3]: an event can be represented as a tuple composed of an *Identifier*, a *Timestamp*, a *Producer*, and several *Attributes*.The *Identifier* identifies events of a certain class of events, and it is unique for each class. The *Timestamp* defines the time at which the event was generated by the producer, identified by the field *Producer*. Attributes are represented in the form $Attribute = (AttributeIdentifier, Value)$, and the type of attribute and its value depend on the type of the event generated. In our case, *Identifier* specifies the type of symptom of attack detected by the probe, while *Producer* the combination of node, processing element, thread causing the illegal action. While not using *Timestamp* (we assume service packets transmitted in order and with prioritized or predictable latencies - statistics about timing are therefore generated inside the NSM), types of attributes are different for the two cases presented. Events generated are discussed in detail in next subsections.

4.3 Illegal Access Probe

As previously discussed, a data protection mechanism does not provide protection against the attacks described in Section 2. Moreover, attempts of access to unauthorized addresses should be notified to counteract efficiently to the related security violations. In fact, an access to a not allowed memory location could imply software problems, but also the presence of a compromised core.

Figure 4 presents architectural details of the *Illegal Access Probe* (IAP). The IAP is in charge of detecting the presence of attempts of unauthorized accesses to memory locations and to notify the reasons of the alert to the NSM. The IAP triggers the creation of a packet to notify the security alert, passing the necessary information to the *Event Generator*. The event is generated when a new transaction is requested to the NI (*MCmd*), and the transmission of the packet is not enabled by the DPU (*TX_enable* = '0'). The IAP module takes as input OCP/IP signals used for identifying the producer of the event and the attributes correlated, as well as some DPU signals, used for identifying the type of security alert. Three main types of event can be identified:

- *Entry input not present in DPU*: this case corresponds to a request in which the combination of the identifier of the PE and the thread ID is not present among the entry lines of the LUT of the DPU, as well as the memory address targeted. This event is identified by a DPU's *match* line equal to '0'.

- *Out of block boundaries*: in this event, requests address input combinations recorded in the LUT of the DPU, while length of data to be stored or read exceeds memory block boundaries. This event is detected when the signal *in_bounds*, which is true when *upper_bound* is higher than the sum of *MBurstLength* and the targeted memory address, is equal to '0'.

- *Wrong access rights*: while the previous two cases are satisfied, the access rights recorded in the RAM of the DPU for the input combinations are negative. The signal *access_right* is therefore equal to '0'.

The right part of Figure 4 shows also the packet generated to communicate the event. The header of the packet depends on the specific NoC implementation and it is not discussed. As previously said, the event packet is composed of several fields. *Identifier* identifies the type of security alerts detected by the IAP. *Producer* is generated by the combination of the identifier, contained in the NI, of the node in the NoC (*NodeID*), the PE identifier (*MConnID*), and the thread identifier (*MThreadID*). Attributes sent to the NSM and relevant for the analysis of the security alert are composed of the information of the unauthorized transaction, i.e., block of targeted addresses (given by the DPU's *upper_bound* signal), length of the data (*MBurstLength*), type of operation requested (given by the OCP/IP signal *MCmd*), and role of the initiator (given by the OCP/IP signal *MReqInfo*).

4.4 Denial of Service Probe

Access control solutions allow to stop unauthorized operations to restricted blocks of memory or ranges of addresses. However, they do not provide protections against attacks aiming at creating Denial-of-Service in the system for instance through the injection of useless packets. As discussed in Section 2, these attacks can be carried out in mobile and multimedia systems with the goal of reducing resources bandwidth or battery lifetime. In order to avoid

such types of attacks, our monitoring system collects statistics about traffic inserted by elements active on the NoC. This would allow discovering unnatural behaviours of the system, as done by IDSs in data networks [10]. Goals of the *Denial of Service Probe* (DoSP) presented in this section are to collect information about the traffic generated by the processing elements (PEs) interfaced to the NI, and to notify the NSM of unexpected changes in traffic conditions, interpreted as symptoms of DoS attacks.

In this work, we consider as unnatural traffic conditions deviations from the average bandwidth expected at design time. We monitor the bandwidth considering the data loaded / stored by an initiator from/to a specific memory block or range of addresses. The bandwidth - representing our statistic event - is calculated over a defined time window. The length of the time window is set by the NSM at run-time or by the designer at design time and depends on the selected security level.

Considering a generic discrete probability distribution for traffic profiles produced by a PE, with media μ and standard deviation σ, we consider unnatural traffic the one exceeding $\mu \pm m\sigma$, with m set accordingly with a desired security level. If those limits for a specific connection are exceeded, an event is generated and sent to the NSM.

Figure 5 shows architectural details of the DoSP. The probe triggers the *Event Generator*, as done already by the IAP. In this case, the DoSP monitors the amount of traffic to/from a selected memory space caused by an initiator (thread running on a PE). With respect to a complete monitoring of the lower and upper bounds of the traffic distribution, and to the monitoring of all the input combinations of the DPU, in the architecture shown in Figure 5 we monitor a limited number of combinations. In particular, we consider only transactions initiated by initiators acting as *user*. Moreover, we consider only the case of traffic exceeding the upper limit of the distribution ($\mu + m\sigma$). Therefore, without loss of generality, we assume monitored the entries combinations recorded in the first l lines of the protection unit, with $l < n$, where n is the total number of entry lines of the DPU. We believe this choice a good trade off between the security service offered and the overhead of the implementation. We implemented the generation of the time window using a programmable counter.

When an initiator loads or stores data into an address in the monitored block i, the length of the data is added to the register $R_{len,i}$. The register is selected by the signals driving the RAM of the DPU (*RAM_Addr*).The new value stored in $R_{len,i}$ is compared to the maximum value allowed for the selected block in the current time window, stored in register $R_{max,i}$. In case the new value in register $R_{len,i}$ is higher than what allowed, an event is triggered and a packet is created to communicate the security alert to the NSM. As with the IAP, the packet generated is composed of the *Identifier*, which identifies the type of security alerts detected by the DoSP, the *Producer*, generated by the same signals creating the IAP *Producer*'s field, and *Attributes*, in this case containing information about the addresses block targeted by the DoS attacks (DPU's *upper_bound* signal). Once the time window reaches its end, the value stored in register $R_{len,i}$ is reset, and statistics are collected for the following time window.

Regarding the full coverage of possible entry configurations, it is easy to see how the hardware blocks shown in Figure 5 should be replicated for every entry line of the DPU. Moreover, in case of monitoring of traffic incoming from initiators with *superuser* and *user* roles, hardware blocks described should be duplicated.

4.5 NSM and communication infrastructure

The Network Security Manager is in charge of collecting security alerts coming from probes and elaborating appropriate countermeasures to attacks and problems detected. The NSM can be implemented in ASIC as a dedicated core, as a general purpose processor running a software application, or as a mixed implementation. However, software (or reprogrammable logic based) implementations allow a higher degree of flexibility, necessary to adapt and update the system in order to be able to face threats coming from new malware. To test our concept, we opted for this solution.

Another point to consider in the implementation of the secure monitoring system is the communication infrastructure. As already mentioned, the traffic coming from probes should be kept separated (at least virtually) from traffic coming from initiators, in order to avoid DoS attacks to influence security service communication. As reported in [4], three main options can be considered: *Separate Physical Interconnect for the original NoC application and the NoC Monitoring Service*, *Common Physical Interconnect but Separate Physical NoC Resources*, *Common Physical Interconnect and Shared Physical NoC Resources*.

We have chosen in our case to implement the third option, i.e, of sharing all the NoC resources while keeping the NoC user traffic and the monitoring traffic separated, therefore creating a virtual NoC for monitoring. This solution is particularly convenient in our case, being the monitoring traffic not relevant and the overhead associated to the implementation limited.

5. SYNTHESIS RESULTS

In this Section, we present synthesis results for the implementations of the probes presented in Section 4, obtained by using the 0.13μm HCMOS9GPHS STMicroelectronics technology library. In Table 1 we show area (in μm^2) and energy consumed (pJ) of the proposed components, i.e., the IAP, the DoSP and the *Event Generator*. Value of a DPU with 16 entry lines are also shown for comparison. The value for the DoSP refers to an implementation monitoring 8 input combinations. The synthesis was optimized for a clock frequency of 500 MHz.

Figure 6 shows the area breakdown (in mm^2) for a NI including a DPU module and the two probes. The DPU has 16 entries, while the DoSP monitors 8 input configurations. The area occupied by the IAP is around the 0.02% of the overall NI considered as reference [14], impacting therefore not significantly to the overall area budget. This is mainly due to the fact that the IAP is mainly composed of combinatorial circuits, reacting to changes of the input signals to provide a trigger to the *Event Generator*. A bigger impact is given by the area consumed by the DoSP (7.63% for 8 configurations monitored). The overall security system, including the DPU and the two probes, counts for around 25.6% of the NI implementation. Compared to a NI implementation without security monitoring [14], the overhead associated is around 34.7%.

In Figure 7, we show the area occupied by several DoSP implementations, by varying the number of combinations

Table 1: Area and energy consumption of the elements composing the security monitoring system

	Area $[\mu m^2]$	**Energy** $[pJ]$
IAP	56.48	0.088
DoSP	25699.38	30.820
Event Gen.	968.26	1.123
DPU	600041.96	72.970

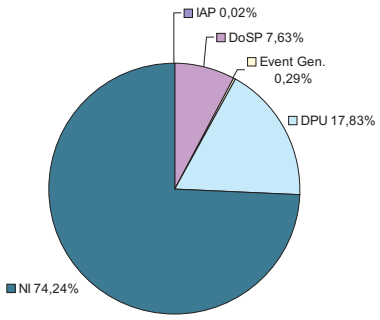

Figure 6: Area breakdown of the several elements of the security monitoring system inside the NI

monitored (4, 8, 16, 32). As expected, the area increases with the number of monitoring blocks, increasing in fact the number of registers used for storing statistics about traffic and maximum values allowed. Similar trends can be also seen for the energy consumed by the component, even if the corresponding graph is not shown for space limitation.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a monitoring system for NoC-based architectures, whose goal is to detect security violations carried out against the NoC. Information collected is provided to a central unit - the NSM - for efficiently counteracting actions performed by attackers. We detail architectural implementation of two type of hardware probes, i.e., the *Illegal Access Probe*, in charge of detecting the presence of attempts of unauthorized access to memory locations, and the *Denial-of-Service Probe*, that detects unnatural traffic behaviours, and we analysed the overhead associated with an ASIC implementation of the monitoring system.

Future work will involve the analysis of traffic behaviours of possible attacks and the integration of the monitoring system with software strategies to detect possible security threats. In particular, a challenge in this direction is to efficiently distinguish between bandwidth anomalies caused by normal system activities and those due to DoS attacks. An integration of the presented probes with other hardware monitors should also be analysed, with the aim to implement a hardware based IDS for NoCs.

Acknowledgements

This work has been carried out under the FP7-ICT-2007-1-216693-MULTICUBE project. The authors would like also to thank Prof. Mariagiovanna Sami from Politecnico di Milano for her advice.

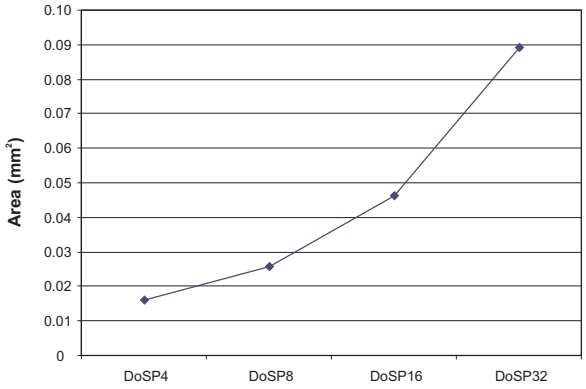

Figure 7: Area of the DoSP by varying the number of elements monitored

7. REFERENCES

[1] *Open Core Protocol Specification 2.2.*

[2] L. Benini and G. De Micheli. Networks on Chips: A New SOC Paradigm. *IEEE Computer*, 2002.

[3] C. Ciordas, T. Basten, R. Radulescu, K. Goossens, and J. Van Meerbergen. An Event-Based Monitoring Service for Networks on Chip. *ACM Trans. on Design Automation of Electronic Systems*, 10(4):702–723, Oct. 2005.

[4] C. Ciordas, K. Goossens, R. Radulescu, and T. Basten. NoC Monitoring: Impact on the Desing Flow. In *Proc. of ISCAS '06*, pages 1981–1984, May 2006.

[5] J. P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin. NoC-centric security of reconfigurable soc. In *Proc. of NOCS'07*, May 7-9 2007.

[6] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano. Secure Memory Accesses on Networks-on-Chip. *IEEE Trans. on Computers*, 57(9):1216–1229, September 2008.

[7] L. Fiorin, G. Palermo, S. Lukovic, and C. Silvano. A Data Protection Unit for NoC-based Architectures. In *Proc. of CODES+ISSS'07*, Sept. 30 - Oct. 5 2007.

[8] L. Fiorin, C. Silvano, and M. Sami. Security Aspects in Networks-on-Chips: Overview and Proposals for Secure Implementations. In *Proc. of DSD'07*, Aug. 29-31 2007.

[9] C. H. Gebotys and Y. Zhang. Security wrappers and power analysis for SoC technology. In *Proc. of CODES+ISSS'03*, pages 162–167, Oct. 1-3 2003.

[10] K. Ilgun, A. Kemmerer, and P. A. Porras. State Transition Analysis: A Rule-Based Intrusion Detection Approach. *IEEE Trans. on Software Engineering*, Mar. 1995.

[11] L. Negri. *Power Modeling and Optimization for Wireless Networks.* PhD thesis, Politecnico di Milano - Dipartimento di Elettronica e Informazione, 2006.

[12] J. Niemela. *F-Secure Virus Descriptions.* F-Secure, December 2007.

[13] K. Pagiantzis and A. Sheikholeslami. Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey. *IEEE Journal of Solid-State Circuits*, 41(3), March 2006.

[14] A. Radulescu, J. Dielissen, S. G. Pestana, O. Gangwal, E. Rijpkema, P. Wielage, and K. Goossens. An Efficient On-Chip NI Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(1), January 2005.

[15] J. van Den Brand, C. Ciordas, K. Goossens, and T. Basten. Congestion-Controlled Best-Effort Communication for Networks-on-Chip. In *Proc. of DATE '07*, March 2007.

[16] B. Vermeulen, K. Goossens, and S. Umrani. Debugging Distributed-Shared-Memory Communication at Multiple Granularities in Networks on Chip. In *Proc. of NOCS'08*, April 7-11 2008.