Dynamic NoC-Based Architecture for MPSoC Security Implementation

Johanna Sepúlveda, Guy Gogniat*, Ricardo Pires+, Wang Jiang Chau, Marius Strum
Microelectronics Laboratory LME, University of São Paulo
*Information and Communication Science and Technology Laboratory Lab-STICC, Université Bretagne Sud
+Federal Institute of Education, Science and Technology of São Paulo
jsepulveda, jcwang, strum, rpires@lme.usp.br, guy.gogniat@univ-ubs.fr

ABSTRACT
MPSoCs have been proposed as a promising architecture choice to overcome the challenging embedded electronics requirements, characterized by tights development times and fast evolution of applications. The MPSoC flexibility, also represents a system vulnerability. As security requirements vary dramatically for different applications, the challenge is to provide MPSoC security that allows a trustworthy system that meets all the security requirements of such applications. NoC has become an attractive alternative to support the MPSoC communication requirements. Our work proposes the implementation of dynamic security architecture to overcome present MPSoC vulnerabilities. We integrate agile and dynamic security firewalls into the NoC in order to detect attacks based on different security rules. We evaluate the effectiveness of our approach over several MPSoCs scenarios and estimate their impact on the overall performance. We show that our architecture can perform a fast detection of a wide range of attacks and a fast configuration of the different security policies for several MPSoC applications.

Categories and Subject Descriptors
B.4.3 [**Interconnections (subsystems)**]

General Terms
Security, Performance, Design.

Keywords
Security; Network-on-Chip; MPSoC; Dynamic Systems.

1. INTRODUCTION
Electronics System design is being revolutionized by the widespread adoption of the multiprocessor system-on-chip (MPSoC) paradigm. A MPSoC integrates multiple programmable processor cores, specialized memories and other intellectual property (IP) components into a single chip [1]. MPSoCs have been proposed as a promising architecture choice to overcome the new challenging application requirements. Specially, the flexibility offered by these systems is becoming desirable and attractive for the SoC designers, which have to face up tights development times as well as the fast evolution of current applications [2]. MPSoCs allow the increase of the life cycle of a system [1]. They are able to support different applications in the same platform. As SoCs are pervading our lives, security is emerging as an extremely important design requirement. Many of the current electronic systems are used to capture, store, manipulate and access sensitive data and perform several critical functions without security guarantee [3]. The current ubiquitous computing and flexibility in SoC design trends promote the resource sharing and upgrading capabilities that integrates the SoC onto an aggressive world. Due to the increasing complexity, flexibility, intrinsic embedded constraints and strict requirements, the implementation of security is considered a challenging task. The security at MPSoC is specially challenging, as the flexibility offered by the platform also represents a vulnerability. The MPSoCs platform allows the execution of several applications in the same structure. 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 no a single and static security requirement, but a set of ever changing security policies that must be satisfied. The security of the MPSoCs must be able to supply different levels of security and capable of being changeable through the operation time.

SBCCI'11, August 30–September 2, 2011, João Pessoa, Brazil.

MPSoC can be attacked [3]. One of the most obvious threats to MPSoC security during its normal operation occurs at its interface to external devices, frequently involving reconfigurable devices or wireless communication IPs embedded onto the SoC. It is possible that during the MPSoC operation the vulnerable IPs fall under control of an external attacker. Thus, these IPs may become malicious. Under the attacker control, these IPs may try, for example, to obtain sensitive information, like passwords or FPGA bitstreams, stored inside the SoC and to send it to the external world. An interface IP may also become a door by which viruses enter the SoC. Previous SoC attacks have been succeeded. An IBM report [4] estimates an exorbitant increasing of computer attacks and foresees the embedded devices as the future targets of such attacks. Embedded attacks will cost billions of dollars [3].

Attacks exploit different system vulnerabilities. An attack can be conducted through three different ways: 1) software: tampering with executable instructions trough the communication structure [3]; 2) microprobing: an invasive technique that involves the physical manipulation of the system [5]; and 3) side-channel: based on information gained from the physical implementation. It includes timing analysis, power analysis, electromagnetic analysis and fault induction [6]. It has been shown that most successful attacks are the software-based ones [3]. In this paper we address protection of the MPSoC against the software attacks by the implementation of the security at the communication structure. The communication structure is becoming the heart of the MPSoC [7]. In order to support the MPSoC high communication requirements, the network-on-chip (NoC) is employed. A NoC is an integrated network that uses routers to allow the communication among the computation structure components. The data flows through the NoC as packets. The integration of security at the NoC is naturally advantageous [8]. The NoC may contribute to the overall security

of the system, providing the ideal mean for monitoring systems behavior and detecting specific attacks [8].
The goal of our work is to present a dynamical NoC-based architecture for meeting efficiently the ever changeable MPSoC security requirements. The security NoC-based system integrates a set of mechanisms capable of being configured according to the security policy needs. We show that our architecture can perform a fast detection of a wide range of attacks and a fast configuration of the different security policies for several MPSoC applications. We also show that the penalties due to the integration of the dynamic NoC-based security architecture are limited to a fraction of time and space of the system. For the best of our knowledge, it is the first attempt to implement a security architecture able to handle different application security policies at the MPSoC. Our architecture is not configured with dynamic or variations of the security mechanisms, but the different levels of security arise from the configuration of the parameters corresponding to the NoC security mechanisms. Two techniques are employed in order to achieve an efficient configuration: 1- the security mechanisms are implemented at the network interface and therefore avoiding the NoC interruption; and 2- QoS (Quality-of-service) mechanisms are employed to provide predictable penalties while the network interfaces are modified. The experiments were performed using a SystemC-TLM timed simulation framework. It automatically carries out performance evaluations for a wide variety of MPSoC scenarios.

The remaining text is organized as follows: Section 2 presents an overview of the previous NoC security works. Section 3 presents the security architecture. It includes the security model of the system, the attack examples, the security requirements and the dynamic security architecture. The experiments and its results are described in Section 4. Finally we present our conclusions in Section 5.

2. PREVIOUS WORKS

The security integration at the NoC level was addressed in the works of [8-9]. They take advantage of the NoC wide system visibility and critical role in enabling the system operation, exploiting the NoC to detect and prevent attacks. However, they implement a single level static security, inefficient for the highly changeable MPSoC security requirements. The work presented in [6] proposes the integration of ciphering techniques at the network interface in order to ensure the secrecy of the exchanged information through the NoC. The proposed mechanism ensures that no unencrypted data leaves the NoC. A key-keeper secure core is responsible for the key distribution in the NoC. New keys can be downloaded and stored in the key-keeper core through encryption techniques. [8-10] integrate a table at the network interface containing the access control rules of each IP. They specify how a component of the NoC can access the protected device. The packets that do not satisfy the access control rules are discarded. The purpose of [9] is to prevent attacks through the verification of the source and size of the packets. The packets that do not obey the communication rules are discarded. This work also integrates a secure network manager component to monitor the NoC behavior. The filter of [8], called Address Protection Unit (APU), verifies that the initiator has the right of access to the requested memory address. The purpose is to prevent four common NoC attacks: denial-of-service (DoS), draining, extraction of secret information and modification. The filter of [10], called Data Protection Unit (DPU), enables the communication only if the type of operation requested by the initiator of the communication is authorized. These previous works show the main role of the NoC in the security of the system. It is a powerful structure for the surveillance and detection of the SoC attacks. However, the adoption of these previous works to address the MPSoC security challenges present three main limitations. 1) they implement a single NoC security level for the entire SoC; 2) they implement static security policies; and 3) they are not appropriate for multithread systems. The purpose of our work is to overcome these limitations.

3. SECURITY ARCHITECTURE

The goal of our work is to design a dynamic NoC-based architecture for the MPSoC protection. Our architecture is shown in Fig. 1. It is based on the configuration of the security mechanisms embodied at the network interface of the NoC. Such configuration is carried out through the configuration of a set of parameters (upgrade of a table), avoiding the high-cost of dynamic reconfiguration. Before detailing our architecture, we briefly discuss the security model of our work. The security policy of the MPSoC can change over the time. Such behaviour is consequence of two factors: 1- the MPSoCs are used as a platform to support different applications, each one characterized by a different set of security requirements; and 2- the own MPSoC status could influence the security policy, that is, under certain conditions the security of the system may be reinforced or decreased. Therefore, the security configuration of the MPSoC must be modified in order to satisfy the new requirements. As the MPSoC is a multithread system, this issue is a challenging task.

To illustrate the types of attacks we consider in our work, we present an attack scenario resulting from the lack of the MPSoC security upgrading. However, our architecture can defend against a broader range of attacks. In this scenario the security policy must be upgraded as a response of the execution of a new application on the MPSoC. We consider, for example, that there is a MPSoC designed to support three applications (A_1, A_2 and A_3). Initially, the set of applications A_1 and A_2 are being executed in the MPSoC when, at some instant, the application A_3 must be executed. The tasks of A_3 are mapped onto the components of the MPSoC. The task that performs critical functions and the sensitive information (i.e. a ciphering key or personal data) of A_3 are mapped together with the tasks of A_1 and A_2. The tasks of A_3 will be unprotected, if the security policy, that defines the access control rights, is not upgraded for the new scenario. An attacker can take advantage of this threat and use the rights over A_1 and A_2 to expose/modify the sensitive information of A_3 or/and avoid the utilization of the MPSoC resources by the tasks of A_3 by keeping busy the component with a never-ending A_1/A_2 task execution.

The dynamic security architecture is shown in Figure 1. It integrates four key components: *policy keeper*, *reconfiguration manager*, *security mechanisms* and *monitor*. Our work supposes that the task allocation of the different applications has been previously defined.

3.1 Policy keeper

As mentioned earlier, secure mobile computer security systems need policies that are flexible and expressive. But traditionally, security systems are designed to enforce one particular security policy. To encompass a wide variety of policy forms the *policy keeper* component stores a thread-oriented policy representation which allows the security specification tailored to the set threads of the applications being executed at the MPSoC.

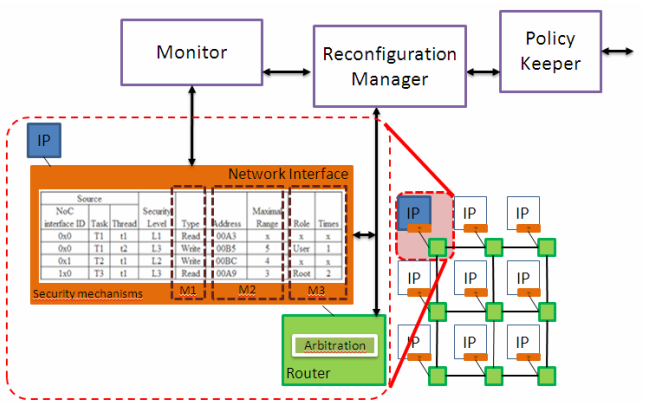

Figure 1. Dynamical security architecture

The *policy keeper* is a safe component that stores the security policies. It integrates the information of the MPSoC thread scheduler and the access rules (rights) of each thread being executed on the MPSoC over the system resources. The security police can express different levels of security. The data can be loaded to the *policy keeper* at two moments: 1) power up time, when all the applications that will be executed on the MPSoC are previously known; 2) run time, otherwise. At run time, the loaded security policy must come from a trusted third-party authority. For some applications, like military applications, the storage of the security requirements in the system is not desirable. Figure 2 shows an example of an MPSoC security policy stored by the *policy keeper* component. Each application or operation mode has a security policy capable of being described by the fields of the table of the figure 2. The size of the table stored by the policy keeper component depends on the number of applications, tasks, threads and operation modes supported by the MPSoC.

3.2 Reconfiguration Manager

The *reconfiguration manager* component focuses on establishing the conditions to guarantee the security requirements of each application and for all the operation modes throughout the operation of the MPSoC. Its main function is the coordination and configuration of the security mechanisms of the MPSoC. In this work, the security mechanisms are composed of firewalls embodied in the NoC, the communication structure of the MPSoC. It uses the NoC interface ID source/destination information embodied in the *policy keeper* component to block the communication and start the reconfiguration process of the security mechanisms.

3.3 Security Mechanisms

The main function of the *security mechanism* components is to defend the MPSoC against possible attacks. The NoC security implementation allows the MPSoC protection by means of communication management.

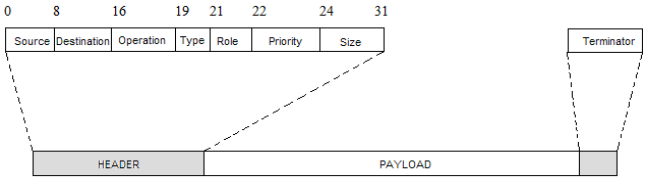

Figure 3. OCP compliant packet format

The NoC firewalls are implemented at the network interface (*secure network interface*) on the basis of a table capable of being upgraded throughout the MPSoC operation. The network interface is the point of interconnection between the NoC routers and the processing components of the MPSoC. The security mechanism uses the information embodied in the packets that flow through the NoC to enforce the different security policies. Our work assumes a network interface compliant with the specifications of the OCP/IP (Open Core Protocol) interface. Messages coming from the MPSoC processing component are translated by the interface into packets compliant to the protocol used within the NoC. The adopted OCP compliant NoC packet format (see Figure 3) is composed of 9 fields.

- ***Source:*** Identifies the task, the thread and the master component. It is the initiator of the communication.
- ***Destination:*** Identifies the slave component. It is the target of the communication.
- ***Operation:*** Codes the transaction type, i.e. a read, read-linked, read-exclusive, write, write-non-post, write-conditional, and broadcast.
- ***Type:*** Defines the information type that is being exchanged, i.e. data, instruction or signal types.
- ***Role:*** Represents the role of the initiator component. i.e. user, root. The roles are defined by the security policy of the system.
- ***Priority:*** allows traffic priority classification.
- ***Size:*** Defines the number of bytes contained in the payload of the packet.
- ***Payload:*** Embodies the information generated by the master.
- ***Terminator:*** Registers the path of the packet through the NoC and the sequential number of this packet in the current transaction between this master-slave pair.

The parameters of the security mechanisms specify the rights of each task upon the MPSoC resources. The security mechanisms implement the access control and authentication security services on the basis of a table.

Source			Destination			Security Level	Type	Address	Maximal Range	Role	Times
NoC interface ID	Task	Thread	NoC interface ID	Task	Thread						
0x0	T1	t1	1x2	T4	t1	L1	Read	00A3	x	x	x
0x0	T1	t2	1x2	T5	t1	L3	Write	00B5	5	User	1
0x1	T2	t1	0x2	T6	t1	L2	Write	00BC	4	User	5
1x0	T3	t1	3x0	T7	t1	L3	Read	00A9	3	Root	2

Figure 2. Example of the MPSoC security policy

Interface 1x2		Source			Security Level	Type	Address	Maximal Range	Role	Times	Path
Task	Thread	NoC interface ID	Task	Thread							
T4	t1	0x0	T1	t1	L1	Read	00A3	x	x	x	0x0;1x0;1x1
T5	t1	0x0	T1	t2	L3	Write	00B5	5	User	1	0x0;1x0;1x1

Figure 4. Example of the security mechanism

It specifies the characteristics of the authorized transactions according to the security policy. The security mechanisms are defined by 6 parameters: 1) *Task* and *Thread* of the interface; 2) *Source*, specifying the *NoC interface ID, task* and *thread* of the allowed transactions; 3) *Security level*, a higher level checks further information of the incoming packet; 4) *Type* of transaction that the initiator can perform; 5) *Address* to which the initiator can access; 6) Maximal range of the transaction; 7) *Role* assumed by the initiator; 8) *Times* that the initiator acceded to such component; and 9) *Path* that an authenticated transaction from the initiator to the destination must perform. Figure 4 shows the configuration of the security mechanism at the network interface located at the position 1x2 of the NoC. It follows the example presented in Figure 2.

Our firewall differs from those proposed by [8,10] in the 1, 4 and 5 fields. Such characteristics allow that our security mechanism avoid a wider set of attacks as the DoS (Denial of service), by using the *times* field, and buffer overflow by the verification of the *address*, *type* and *size* fields. Moreover, our approach supports the multithreading characteristic of the MPSoC. It allows the safely execution of multiple tasks of different applications, and thus multiple security policies implementations, at the same time. Note that our architecture is also feasible for different security mechanisms whose security characteristics are capable of being changed throughout the operation of the system.

3.4 Monitor

The *monitor* component is permanently auditing the communication behaviour of the MPSoC. It detects the NoC activity in order to determine the completion of the communication among the different master/slave pairs of the MPSoC. A master is defined as any component that initiates a communication transaction. The slave is any component that receives a request of a communication transaction of a master. The monitor also has the ability to record and report on the security mechanism configuration at any moment of time.

4. EXECUTION MODE

The functionality of our system can be summarized as follows. When the security policy of the MPSoC must be upgraded, because for example a new application must be executed by the MPSoC or the operation mode of the MPSoC changed, the *reconfiguration manager* starts three procedures: *analysis of security policy*, *mechanism configuration* and *recovery*. At the *analysis of security policy procedure* the reconfiguration manager downloads the properly security policy, stored in the *policy keeper* component. It uses the MPSoC tasks mapping information to identify which *security mechanism* must be modified and its suitable configuration parameters. The modified security mechanism corresponds to the *secure network interface* that links the processing component that handles such task to the NoC. The *security mechanism* parameters correspond to the values that must be stored in the firewall.

In order to complete the *mechanism configuration procedure*, the *reconfiguration manager* blocks the injection of packets whose final destination is the processing component linked to the security interface that is going to be reconfigured (called *target interface* for our explication purposes). Such packets are stored onto the interface linked to the master processing component. The reconfiguration manager also starts a QoS (Quality-of-Service) mechanism that rises up the priority of the communication of the packets whose final destination is the component linked to the *target interface*. That is, modify the *Priority* packet field. The QoS mechanism modifies the arbitration of the NoC routers, so that, the communication of the packets with higher priorities is performed first. Once the communication of all the packets flowing to the target interface is finished, the reconfiguration of the *target interface* can be started. The reconfiguration consists in the modification of the parameters of the security mechanisms. Note that as the *security mechanisms* are implemented at the NoC interface, the communication is not interrupted at the NoC during the reconfiguration. The NoC routers continue the communication of the remaining packets through the network. This characteristic of our architecture can reduce the latency penalties due to the reconfiguration.

When the reconfiguration is finished, the final *recovery procedure* is started. The *configuration manager* frees the injection of the packets that were being blocked during the reconfiguration. The normal NoC operation is achieved when all the target interfaces are reconfigured.

5. RESULTS

We have developed the SystemC-TLM cycle-accurate model of our architecture. The evaluation was performed by the SystemC-TLM framework developed in [BLIND]. We employ a 4x4 mesh-based NoC characterized by a XY routing scheme, round-robin (RR) arbiter and FIFO memory organization. The proposed solution has been verified against three types of attack scenarios: 1) Extraction: characterized by unauthorized attempts to access data; 2) Modification: using the buffer overflow technique; and 3) DoS: repeating several times the same transaction. It has been proved to be efficient protecting the system in all considered scenarios. Therefore, the packets that do not respect the security policy of the application are detected and discarded. The performance evaluation of our approach was based on MiBench benchmark suite [11]. These patterns were used as NoC benchmarks in previous works. Our MPSoC traffic is composed by a set of heterogeneous tasks arriving to different rates to the system. The tasks are randomly allocated at the processing components of the system. Each traffic pattern is composed of five flit size packets of three types: real-time, write or read and signalling, characterized by a different generation rate (40.000, 160.000 and 20.000 packets per second respectively). We compared the communication performance of the MPSoC dynamic NoC-based security architecture against the best-effort NoC (without security).

Figures 5-8 show the MPSoC communication results. They represent the average relative execution time of our architecture after completed 4, 8, 16, 32 and 64 tasks of the MiBench benchmarks. The number of processors that execute these tasks vary from 1 to 16. TBE corresponds to the best effort system, without security implementation, and TDS expresses the response of our dynamic NoC-Based architecture.

Figure 9 shows the power consumption of our dynamic approach. The power consumption (P_{DYN}), given by equation (1) is the sum of the *NoC* power (P_{NoC}), the *policy keeper* power (P_{Keep}), the *reconfiguration manager* power (P_{Man}) and the *monitor* power (P_{Mon}).

$$P_{DYN} = P_{NoC} + P_{Keep} + P_{iMan} \quad \textbf{(1)}$$

The NoC power is given by equation (2). It is composed by the sum of the links power (P_{Li}), interfaces power (P_{Int}) and routers power (P_{Ri}) due to transaction completion [12]. P_{Li} and P_{Ri} are proportional to the *channel utilization rate* and *router utilization rate* respectively.

$$P_{NoC} = P_{Li} + P_{Int} + P_{Ri} \quad \textbf{(2)}$$

We developed power models for the main components in the *NoC* architecture. We integrated these models into the simulator, taking the architectural and technological parameters into account. The characterization was made under the 0,25 µm process constraints, 2.5 volts as a power supply and a 25ºC temperature. Our power estimation strategy is based on identifying the activity of each component of our dynamic system. In order to fulfill this task, the monitor annotates the communication events on the *NoC*.

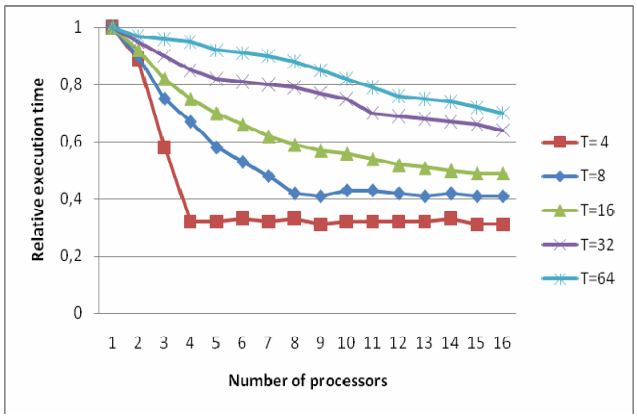

Figure 5. Relative execution time.

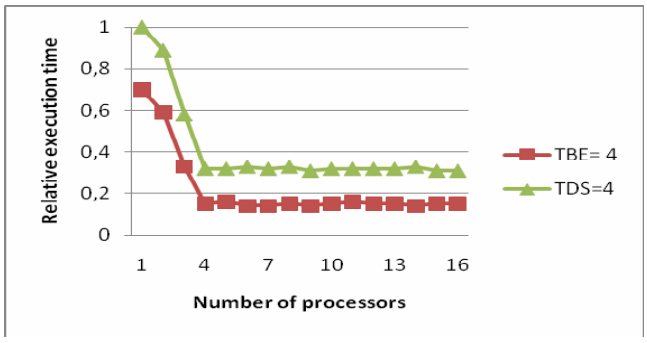

Figure 6. Best effort and our approach responses for 4 tasks.

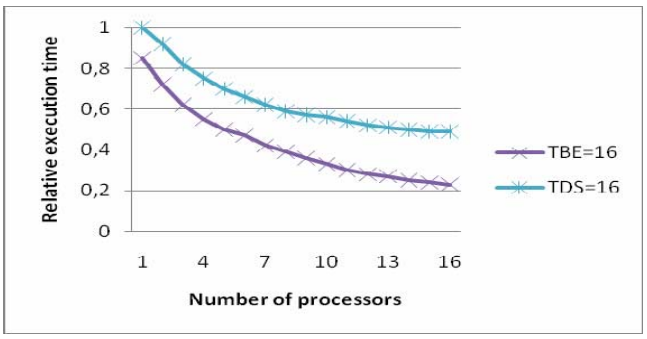

Figure 7. Best effort and our approach responses for 16 tasks.

At the end of the simulation, the number of activity occurrences is obtained for each NoC component. A power consumption cost is also evaluated for each activity. At the end of the simulation, the activity occurrences of each component are used to calculate the total power dissipation.

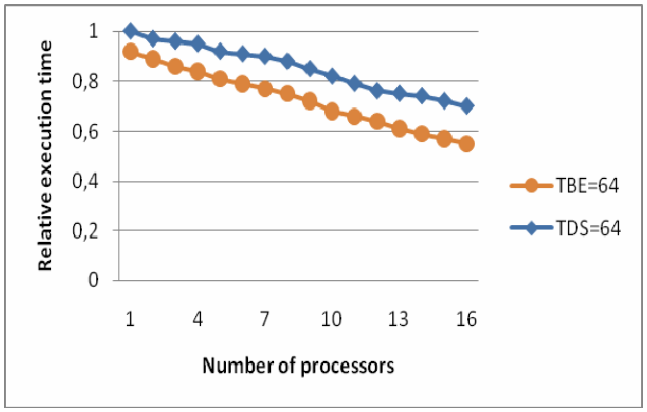

Figure 8. Best effort and our approach responses for 64 tasks.

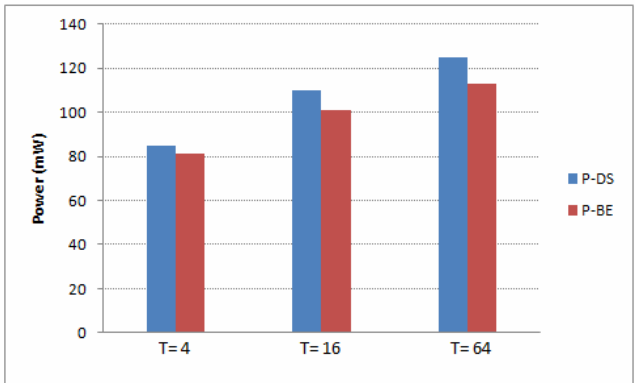

Figure 9. Power consumption results.

The attack detection is performed within a single clock cycle. Average latency penalty due to dynamic security implementation was less than 4%. When the security configuration must be performed, the NoC routers are not blocked. The communication of the MPSoC is not interrupted. Such a mechanism reduces the performance impact due to the implementation of our approach.

The results show that as the number of tasks increases, the impact on the overall system performance is greater. It can be explained by the increase of the probability of the tasks being allocated at the same processing component of the MPSoC. Therefore, the number of network interfaces that must be reconfigured decreases. The power consumption penalties achieved an overhead up to 19,6% over the implementation without security.

6. CONCLUSIONS

In this work we proposed a dynamic NoC-based security implementation for MPSoC. Such approach handles the different and ever-changing security policies, main security characteristic of current MPSoCs. The NoC-based protection is based on a set of firewalls that are implemented on the basis of a table and distributed along the NoC interfaces. These firewalls perform the access control and authentication security services. Such mechanism detects a wide set of attacks. The efficient reconfiguration is achieved through the upgrading of the firewall parameters. Results show that the inclusion of security issues in a NoC implies a tradeoff between trustworthy and performance. The inclusion of our dynamic security architecture allows the meeting of the current security and performance MPSoC requirements. As a future work, we will implement different NoC security mechanisms that allow the implementation of reconfigurable confidentiality and integrity services.

7. REFERENCES

[1] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini, R. Zafalon.: Analyzing On-ChipCommunication in a MPSoC Environment. Design, Automation and Test in Europe Conference and Exhibition (DATE 2006).

[2] Benini L.: Application Specific NoC Design. Design, Automation and Test in Europe Conference and Exhibition (DATE 2006). Vol 1, pp. 1-5. March 2006.

[3] Kocher P., Lee R., McGraw G., Raghunathan A., Ravi S.: Security as a New Dimension in Embedded System Design. Design Automation Conference (DAC2004). 2004.

[4] IBM, "*Global Business Security Index Report*", http://www-03.ibm.com/press/fr/fr/pressrelease/26379.wss.

[5] K. Elissa, "Title of paper if known," unpublished.

[6] Gebotys C., Zhang Y.: Security wrappers and power analysis for SoC technologies. CODES 2003.

[7] Ogras U., Hu J., Marculescu R.: Communication-centric SoC design for nanoscale domain. IEEE International conference on Application-specific systems 2005.

[8] Fiorin L., Silvano C., Sami M.: Security Aspects in Networks-on-Chips: Overview and Proposals for Secure Implementations. In Proc. 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools. 2007.

[9] Evain S., Diguet J.: From NoC security analysis to design solutions. Design, Automation and Test in Europe Conference and Exhibition (DATE 2006).

[10] Fiorin L., Lukovic S., Palermo G.: Implementation of a Reconfigurable Data Protection Module for NoC-based MPSoCs. In Proc. IEEE Parallel and distributed processing. 2008.

[11] MiBench Version 1.0. http://www.eecs.umich.edu/mibench/