Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 578363, 10 pages
doi:10.1155/2012/578363

Research Article

QoSS Hierarchical NoC-Based Architecture for MPSoC Dynamic Protection

Johanna Sepulveda,[1] Ricardo Pires,[2] Guy Gogniat,[3] Wang Jiang Chau,[1] and Marius Strum[1]

[1] *Microelectronics Laboratory, Polytechnic School, University of São Paulo, 05508900 São Paulo, SP, Brazil*
[2] *Federal Institute of Education, Science and Technology of São Paulo, 01109010 São Paulo, SP, Brazil*
[3] *Information and Communication Science and Technology Laboratory, University of South Brittanny, 56100 Lorient, France*

Correspondence should be addressed to Johanna Sepulveda, jsepulveda@lme.usp.br

Received 20 January 2012; Revised 15 May 2012; Accepted 15 May 2012

Academic Editor: Elmar Melcher

As electronic systems are pervading our lives, MPSoC (multiprocessor system-on-chip) security is becoming an important requirement. MPSoCs are able to support multiple applications on the same chip. The challenge is to provide MPSoC security that makes possible a trustworthy system that meets the performance and security requirements of all the applications. The network-on-chip (NoC) can be used to efficiently incorporate security. Our work proposes the implementation of QoSS (quality of security service) to overcome present MPSoC vulnerabilities. QoSS is a novel concept for data protection that introduces security as a dimension of QoS. QoSS takes advantage of the NoC wide system visibility and critical role in enabling system operation, exploiting the NoC components to detect and prevent a wide range of attacks. In this paper, we present the implementation of a layered dynamic security NoC architecture that integrates agile and dynamic security firewalls 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 different security policies for several MPSoC applications.

1. Introduction

SoC designers have to face up tight development times as well as the rapid evolution of current applications [1]. To be cost effective, SoCs are often programmable and integrate different applications on the same chip (i.e., cell-phone, personal digital assistant) [1]. Although sharing many of the hardware components on the SoC, different applications executed on a single chip may present very different requirements and design constraints. Such type of system is called multiapplication [2]. MPSoCs have been proposed as a promising architecture choice to overcome the new challenging application requirements. A MPSoC integrates multiple programmable processor cores, specialized memories, and other intellectual property (IP) components into a single chip [1]. MPSoC platforms allow simultaneous execution of several applications in the same structure. Current ubiquitous computing and flexibility in SoC design trends promote the resource sharing and upgrading capabilities. As MPSoCs are pervading our lives, security is emerging as an extremely important design requirement. Many of the current electronic systems embedded into an MPSoC are used to capture, store, manipulate and access sensitive data, and perform several critical functions without security guarantee [3]. Due to the increasing complexity, flexibility, intrinsic embedded constraints, and strict requirements, the implementation of security is considered as a challenging task. The security at MPSoC is specially challenging, as the flexibility offered by the platform also causes vulnerability. Each application supported by the MPSoC is characterized by a different set of security rules, called security policy. The set of applications can be mapped dynamically on 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 MPSoCs security policy must be able to supply different levels of security and be capable

of changes during operation time. A McAfee report [4] estimates an exorbitant increasing of embedded attacks in the last 5 years and loses of billions of dollars [3]. MPSoC can be attacked via hardware/software [3]. Software attacks are responsible for 80% of security incidents [4]. All software attacks start with an abnormal communication. In this paper, we address protection of the MPSoC against software attacks by implementation of security policies at the NoC-based communication structure.

In order to support the MPSoC high communication requirements, the network-on-chip (NoC) approach is employed [5–7]. An NoC is an integrated network that uses routers to allow the communication among computation components. 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 communication structure is becoming the heart of the MPSoC [7]. It has a significant impact on the overall MPSoC performance. To make feasible MPSoC protection by NoCs, the security policy must be customized in order to provide a better tradeoff between system performance and security. Our work proposes the implementation of QoSS (quality of security service) to overcome present SoC vulnerabilities. QoSS is a novel concept for data protection that introduces security as a dimension of QoS (quality-of-service). In contrast with previous works, different security levels deployment allows the best tradeoff between system security and performance requirements. In our work, QoSS is implemented at the network-on-chip (NoC) to provide predictable security levels of the system by adding functionality to the routers of the network and consequently changing some local configuration parameters or modifying the network interfaces. The goal of our work is to present a layered dynamical NoC-based architecture to efficiently meet the changeable MPSoC security requirements. Our hierarchical architecture distributes the security policy management by partitioning the NoC topology into different *security zones* (low NoC), ruled by a *local* security policy and connected through a global interconnect (high NoC), which implements a *global* security policy. Our hierarchical approach improves system's performance while effectively handling the security policy changes. Each zone integrates a set of mechanisms capable of being configured according to QoSS needs of each application. 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 penalties due to the integration of the dynamic NoC-based security architecture are limited to a fraction of time and space of the system. The different levels of security of each *security zone* 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) security mechanisms are implemented hierarchically, therefore, avoiding interruption in NoC execution; (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.

In summary, the novelties of our work are as follows:

(i) implementation of a layered dynamical MPSoC security by the use of a hierarchical NoC, we divide the MPSoC security policy into global and local ones, which is suitable for dynamic systems;

(ii) creation of security zones in the MPSoC;

(iii) implementation of QoSS for hierarchical NoCs.

2. Previous Works

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 system operation, exploiting the NoC to detect and prevent attacks. However, they implement a single-level static security, which may be inefficient as a solution 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 by using encryption techniques. The papers [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. Packets that do not satisfy these access control rules are discarded. The purpose of [9] is to prevent attacks by verifying 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. Its purpose is to prevent four common NoC attacks: denial-of-service (DoS), draining, extraction of secret information, and modification. The filter of [10, 11], called data protection unit (DPU), enables the communication only if the type of operation requested by the initiator of the communication is authorized. The work of [12] proposes an architecture composed by three modules in order to avoid code injection attacks: (1) SPU (stack protection unit), to track the transactions; (2) ITU (instruction trace unit), to track the instructions; (3) LSM (local security manager), to react upon the reception of an alert signal of SPU or ITU. 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 attacks in SoCs. However, the adoption of these previous solutions to address 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; (3) they are not appropriate for multithreaded systems. In our previous work [5], we developed a dynamical NoC-based protection for SoCs. However, it uses central control blocks that present a strong impact on the overall

area due to link overhead. The purpose of the present work is to overcome these limitations.

3. Security

Many SoCs interact with other electronic devices, in many cases wirelessly. By interacting with other digital devices, an SoC may receive viruses (or other similar malicious pieces of code). Among the motivations for someone to attack a SoC, we underline three examples: (1) economical gain by obtaining confidential information (e.g., passwords, IP bitstreams) stored in an SoC; (2) reputation: a hacker may attack a SoC by viewing this action as a personal challenge; (3) vandalism: the purpose is to cause loss or damage to a SoC. Viruses may be used for this purpose.

Attacks exploit different system vulnerabilities. An attack can be defined as any unauthorized attempt to access or to use the system resources [3]. An attack can be conducted through three different ways: (1) software: tampering with executable instructions through the communication structure [3]; (2) microprobing: an invasive technique that involves physical manipulation of the system [5]; and (3) side-channel: based on information gained from the physical implementation, including timing analysis, power analysis, electromagnetic analysis, and fault induction [6]. According to the purpose of the attacks, they can be classified into three categories: (1) extraction: unauthorized reading of critical data that is being exchanged through the network from/to a secure target; (2) modification: unauthorized change of critical data that may be done through writing actions, state modifications, data creation or removal; (3) denial of Service, whose aim is to bring down the system performance.

In order to prevent and to mitigate attacks to the NoC, security services can be implemented. The main function of security services is to protect network resources and data exchanges by means of communication management [9, 11]. There are six security services [10]: (1) confidentiality which ensures the data secrecy; (2) integrity which assures that data is kept unchanged during any operation; (3) authentication which validates the sender IP integrity; (4) access control which allows or denies the use of a particular resource; (5) availability which ensures the use of the network resources; (6) nonrepudiation that maintains evidence of NoC communication events. It has been shown that the most successful attacks are the software-based ones [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. In such cases, the vulnerable IPs fall under control of an external attacker. Thus, these IPs may become malicious. Under the attacker's control, the infected IPs may try, for example, to obtain sensitive information, like passwords or FPGA bitstreams, stored inside the SoC, and send it to the external world. An interface IP may also become a door by which viruses enter the SoC. There are known cases of SoC attacks that have succeeded [4]. In our work, we consider attacks that take advantage of the lack of MPSoC security upgrading. However, our architecture can defend against a broader range of attacks. MPSoC's characteristics demand that security policy must be upgraded as a response of the execution of a new application on it. We consider, for example, that there is an MPSoC designed to support three applications (*A1*, *A2,* and *A3*). Initially, the set of applications *A1* and *A2* is being executed in the MPSoC when, at some instant, the application *A3* must be executed. The tasks of *A3* 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 *A3* is mapped together with the tasks of *A1* and *A2*. The tasks of *A3* 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 *A1* and *A2* to expose/modify the sensitive information of *A3* or/and avoid the utilization of the MPSoC resources by the tasks of *A3* by keeping busy the component with a never-ending *A1*/*A2* task execution.

4. Quality of Security Service

4.1. Security Mechanisms. The implementation of security services is carried out through security mechanisms which increase the complexity of the NoC. Optimal NoC configuration demands a deep exploration of the wide NoC design space. Recently, the work in [13] proposes the quality-of-security-service QoSS (quality-of-security-service) for extrachip (conventional) networks. It explores the tradeoff between the system trustfulness and its performance. The traffic of a single embedded application may integrate several flows, each of which characterized by different security requirements. The QoSS concept allows differentiated treatment for the data exchange carried out through the NoC. The QoSS can be implemented either by adding functionality to NoC routers and consequently changing some local configuration parameters or by modifying network interfaces. Different security levels are implemented through security mechanisms. The security choices represent a special configuration of the *security mechanisms*. The security mechanisms use the embodied information within the packet to perform access control and authentication on the packet arriving at (1) the NoC interface or (2) the router. For comparison reasons, we addressed access control and authentication services in order to neutralize extraction and modification attacks. Additional security service can be handled with complementary cyphering techniques [8]. Access control and authentication security mechanisms are implemented as *firewalls*. Each firewall stores the security policy information of each resource in a security table. Whenever a transaction takes place, the information embodied in the packet is checked against the security table information.

4.2. Access Control. It regulates NoC traffic, allowing or denying data exchanges between a master-slave pair based upon a set of rules. The communication control flow was modified to manage packet accesses by using the data stored in the *security table* of the firewall which contains the access rights of each *MPSoC* computation component. The rights change according to the applications that are mapped on the MPSoC at that time. Our access control service implements four

Table 1: Access control levels.

Level	SV	OV	RV
L0			
L1	×		
L2	×	×	
L3	×	×	×

Table 2: Authentication levels.

Level	SV	PV	CV
L0			
L1	×		
L2	×	×	
L3	×	×	×

security levels, which arise from the combination of three security mechanisms. Note that our method can support any number of security levels. Table 1 shows the different access control levels. A higher security level compares a larger number of information embodied in the packet (source, type of operation, and master role). Unauthorized packets are discarded.

SV: it verifies the source/target of the transaction that is, the rights that the master has over the target slave component. It identifies the task and the thread authorized for each transaction.

OV: it verifies that the authorized operations are executed in the correct MPSoC resources. It also regulates the memory access through the verification of the fields: *address*, specifying the memory addresses involved in the operation; *size*, corresponding to the maximal range of memory addresses capable of being modified by the operation; *time*,stating the number of memory modifications allowed by the transaction.

RV: it verifies the role of the initiator component of the transaction.

The *SV* offers a basic access control. It verifies the existence of the message destination and that the master and slave components have not identical NoC addresses. Such characteristic avoid possible DoS (Denial of Service) attacks through *livelock*, characterized by the insertion of a packet that cannot reach its destination, and draining attacks, which is characterized by the intentional wasting of NoC resources. The *OV* can be used to detect *buffer overflow* attacks, one of the common embedded attacks [3]. It explores the rights of an authenticated component in order to perform unauthorized operations [3]. The RV mechanism includes the operation context of the master of the transaction.

4.3. Authentication. The authentication security service verifies the integrity of the source of critical data. It does this by checking if the route taken by the packet is consistent with the source IP field contents.

Our authentication service implements four security levels, which arise from the combination of 3 security mechanisms. Table 2 shows the different authentication levels. These strategies make very difficult for a malicious master to successfully send a packet as it would be another master.

SV: it is the same mechanism described at Section 4.2.

PV: in order to perform the authentication, a routing trace is embodied in the header field of the packet. This field content is modified by the routers along the communication path of each packet. In this process, when a packet traverses a router, the packet header receives this router signature. A simple strategy to do this, adopted in this work, is to use a trace field containing R bits, where R is the number of routers in the NoC. The routers are numbered from 0 to $(R-1)$. To each router, r corresponds the bit in the trace field whose position inside the field is also r. When a packet enters the NoC, all the bits of its trace field are equal to 0. Each time the packet crosses a router, this router changes the corresponding bit in the trace field to 1. Then, at the end of the route, the packet terminator reveals the complete path that has been taken by the packet, indicating with 1 which routers have been crossed by the packet and with 0 which have not. By knowing the NoC topology and routing algorithm, the slave can deduce what is the true packet sender and thus it can verify if the alleged source is in fact this sender. For this purpose, a *security table* in each destination contains the expected value for the trace field coming from each possible master. In the case of a mismatch (i.e., the trace field does not correspond to the expected one) the packet is discarded.

CV: each master-slave pair may also keep track of the sequential number of its transactions. In this case, this sequential number is also included in the header. The slave then verifies if the transactions occur according to the expected numbering. To overcome this feature, an intruder would have to know the current expected sequential number of the transaction of the master-slave pair it intends to attack. A packet whose sequential number differs from the expected one is discarded.

5. Our Approach

The goal of our work is to design a layered dynamic NoC-based architecture for the MPSoC protection. In such a scenario, all the applications that are going to be mapped on the MPSoC are known in advance. Our security architecture is based on a hierarchical NoC (HNoC), combining a *low* and *high* NoC, and on the configuration of the security mechanism embodied at the network interface of the NoCs. Such configuration is carried out through the establishment of a set of parameters (upgrade of the *security table*), avoiding the high-cost of dynamic reconfiguration. There is an IP that stores all the security policies that the MPSoC must obey. The HNoC is notated as $n(S_1)/(S_2)$, where n is the number of security zones, S_1 is the *low* NoC size, and S_2 is the *high* NoC size. Figure 1 shows an HNoC, whose size is $4(2\times 2)/(2\times 2)$. The security zones are identified as *Zone I to Zone IV*.

The IP components are mapped on the HNoC according to their communication and security characteristics by using the MAIAS algorithm [14]. The purpose is to allocate in the same security zone the IPs with higher communication frequency and similar security requirements. As a consequence,

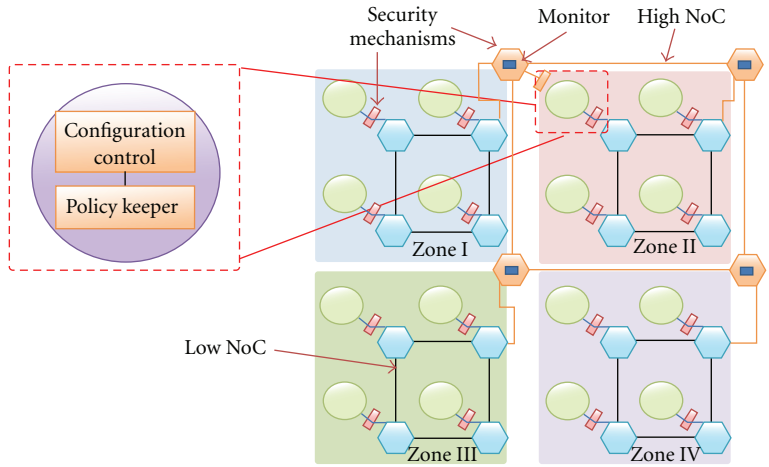

Figure 1: Layered dynamical architecture.

the inter-cluster communication is reduced. The HNoC is a scalable solution [15]. It integrates low-diameter topologies, that aided by efficient flow control and routing mechanisms, minimizes the overall power consumption and NoC latency [16].

Our architecture integrates five key components: (1) *low NoC/high NoC,* (2) *policy keeper*, (3) *configuration control,* (4) *security mechanisms,* and (5) *monitors*. Our work supposes that the task allocation of the applications has been previously defined.

5.1. Low NoC and High NoC. The HNoC is a layered communication structure composed by a set of networks. It can be divided into *low* and *high NoC* [14]. The IP cores of the MPSoC can be organized into clusters by the *low NoC*. Each cluster is completely separated from the others. The *low* 2D-mesh NoC has its own connection to a router of the *high NoC*. The *high NoC* collects the traffic coming from different clusters. Therefore, the *high NoC* must provide a larger bandwidth than the links in the *low NoC*. Higher frequency or additional links can be used at the high NoC. We use additional links. The *high NoC* links are physically longer than links at the *low NoC,* but they can be implemented on the higher metal layers with reduced RC-delay. This hierarchical structure allows the implementation of security zones, composed of the IPs at each cluster. The components at each security zone have similar security characteristics. The security policy, that rules the interaction among all the system components, is divided into local and global policies. The IPs of a security zone are ruled by the *local security policy*. The intercluster communication is ruled by the *global security policy*. Such hierarchical approach provides three advantages: (1) it facilitates the security management of the MPSoC; (2) it produces smaller security tables; (3) it improves system performance. The security policy of the MPSoC can change over time. Such behaviour is the consequence of two factors: (1) MPSoCs are used as a platform to support different applications, each one characterized by a different set of security requirements; (2) MPSoC status could influence the security policy, that is, under certain conditions, the security level of the system may be dynamically reinforced or decreased. Therefore, the security configuration of the MPSoC must be modified in order to satisfy the new requirements.

5.2. 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 policies, the *policy keeper* component stores a thread-oriented policy representation which allows the security specification tailored to the threads set of the applications being executed at each security zone of the MPSoC. The *policy keeper* is a safe component that stores the security policies (*local* and *global*). It integrates the information of the MPSoC thread scheduler, the security zone, and the access rules (rights) of each thread being executed on the MPSoC over the system resources. The local security policy configures the *low NoC* security mechanisms of each *security zone* and the global security policy configures the security mechanisms embodied at the high NoC. The security policy can express different levels of security. Data can be loaded on 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. 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.

5.3. Configuration Control. The *configuration control* 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 according to the local

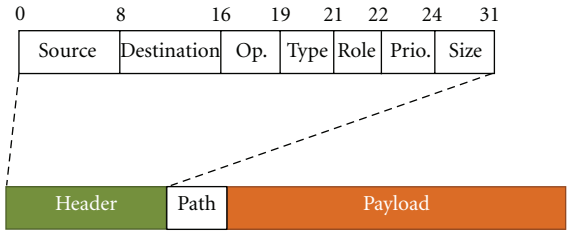

Figure 2: OCP compliant NoC packet format.

and global security policies of each application. In this work, the security mechanisms are composed of firewalls embodied in the NoC (*low* and *high*), 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.

5.4. Security Mechanisms. The main function of the *security mechanism* components is to defend the MPSoC against possible attacks. The NoC security implementation allows MPSoC protection by means of communication management. NoC firewalls are implemented at the network interface (*secure network interface*) at the *low NoC* and at the *high NoC*. The mechanisms implemented in this work are explained in Section 3. Security mechanisms at the *low NoC* are implemented at the network interface. The network interface is the point of interconnection between NoC routers and processing components of the MPSoC.

Security mechanisms at the *high NoC* are implemented at NoC routers. The security mechanism uses 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 an 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 2) is composed of 9 fields.

(1) Source: it identifies the task, the thread, and the master component. It is the initiator of the communication.

(2) Destination: it identifies the slave component. It is the target of the communication.

(3) Operation: it codes the transaction type, that is, a read, read linked, read-exclusive, write, write-non-post, conditional write, and broadcast.

(4) Type: it defines the information type that is being exchanged, that is, data, instruction, or signal types.

(5) Role: it represents the role of the initiator component. That is, user, root. The roles are defined by the security policy of the system.

(6) Priority: it allows traffic priority classification.

(7) Size: it defines the number of bytes contained in the packet payload.

(8) Path: it 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.

(9) Payload: it embodies the information generated by the master.

Our firewall differs from those proposed by [8, 10] in the *source*, *type* and *role* fields. Such characteristics allow that our security mechanism avoid a wider set of attacks as the DoS (Denial of service), by using the *source* 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 safe 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 during system operation.

5.5. Monitors. The *monitor* component is implemented at the *high NoC* and *low NoC* routers and is permanently auditing the MPSoC communication behaviour. It detects NoC activity in order to determine the completion of each communication event between different master/slave pairs of the MPSoC. A master is defined as any component that initiates a communication transaction. A 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.

5.6. Execution Mode. The functionality of our layered security system can be summarized as in Figure 3. When the MPSoC security policy must be upgraded, because, for example, a new application must be executed or the MPSoC operation mode is changed, the *configuration control* starts six procedures: *lookup policy keeper*, *block, look-up monitors, global configuration (high NoC), local configuration (low NoC)*, and *unblock*. At the *lookup policy keeper* step, the configuration control downloads the proper *local* and *global* security policies, stored in the *policy keeper* component. It uses the MPSoC tasks mapping information to identify which *security mechanism* and which *security zones* must be modified as well as the new security tables information.

The *block* procedure interrupts the injection of packets whose final destination is the processing component linked to the security zone that is going to be reconfigured (called *target interface*). Such packets are stored on the interface linked to the master processing component. The reconfiguration manager also starts a QoS (quality-of-service) mechanism that raises the priority of the communication of the packets whose final destination is the component linked to the *target interface*; modifying 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 start. In order to configure the HNoC,

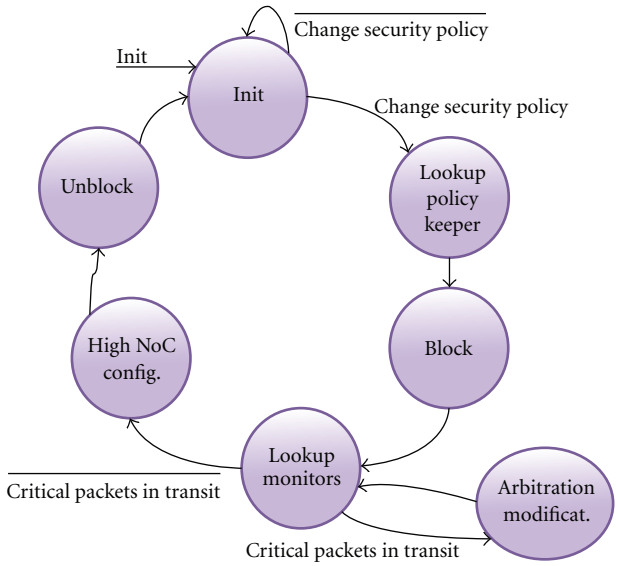

Figure 3: Functionality of the layered security system.

the *configuration control* component sends new information that must be stored at the *security tables* of the firewalls. The *high NoC configuration* modifies the security tables at the routers. The *low NoC configuration* modifies the security tables at the network interfaces of the selected security zone. 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 *unblock procedure* starts. The *configuration control* 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.

6. Results

6.1. Experimental Setup. We have developed a SystemC-TLM cycle-accurate model of our architecture. Its evaluation was performed by the SystemC-TLM framework as described in [5]. We employ a $4(2 \times 2)/(2 \times 2)$ HNoC characterized by an 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. The performance evaluation of our approach was based on the MiBench benchmark suite [17]. We select three applications: auto/industrial (*A1*), consumer electronics (*A2*), and telecommunication (*A3*), see Table 3. Each application is characterized by a security policy that establishes different levels of authentication and access control security mechanisms, as shown in Table 4.

Table 3: Benchmarks.

Auto/Industrial (*A1*)	Consumer (*A2*)	Telecomm. (*A3*)
Basicmath	Jpeg	CRC32
Bitcount	lame	FFT
Qsort	mad	IFFT
susan (edges)	tiff2bw	ADPCM enc.
susan (corners)	tiff2rgba	ADPCM dec.
susan (smoothing)	tiffdither	GSM enc.
	tiffmedian	GSM dec.
	typeset	

Table 4: Security levels.

Application	Function	Authentic.	Access control
Automotive (*A1*)	Basicmath	Level 0	Level 2
	Bitcount	Level 0	Level 0
	Qsort	Level 3	Level 3
	susan (edges)	Level 2	Level 2
	susan (corners)	Level 2	Level 2
	susan (smoothing)	Level 2	Level 2
Consummer Electronics (*A2*)	Jpeg	Level 2	Level 2
	Lame	Level 2	Level 2
	Mad	Level 0	Level 0
	tiff2bw	Level 0	Level 0
	tiff2rgba	Level 0	Level 0
	tiffdither	Level 0	Level 0
	tiffmedian	Level 0	Level 0
	typeset	Level 0	Level 0
Telecomm. (*A3*)	CRC32	Level 2	Level 2
	FFT	Level 1	Level 1
	IFFT	Level 1	Level 1
	ADPCM enc.	Level 0	Level 0
	ADPCM dec.	Level 0	Level 0
	GSM enc.	Level 3	Level 3
	GSM dec.	Level 3	Level 3

Our experimental work considers six mapping combinations resulting from the execution of these three applications on the MPSoC (*A1*, *A2*, *A3*, *A1-A2*, *A1-A3*, *A3-A2*). These patterns were used as NoC benchmarks in previous works. Our MPSoC traffic is composed of a set of heterogeneous tasks arriving at different rates in the system. The tasks are randomly allocated in the system processing components. For comparison reasons, each traffic pattern is composed of five flit size packets of three types: real-time, write or read and signaling, characterized by a different generation rate. The intercluster communication varies from 20% to 50% of the overall traffic. We compared the communication performance of the HNoC-based dynamic security architecture against the simple NoC-based dynamic security and the best-effort NoC (without security).

Table 5: Security efficacy.

Attack scenario	Authentication efficacy	Acces Control efficacy
Send critical information	87%	100%
Read critical information	83%	100%
Write not authorized areas	100%	100%
Nonexisting target	100%	100%
Repeated information	89%	100%
Communication target = source	100%	100%

6.2. Security Efficacy. Table 5 shows the results of the efficacy of our security implementation. It represents the percentage of attacks detected by the security mechanisms. The percentages of the efficacy of authentication and access control security services were the same as our previous dynamic approach. That is because both approaches use the same security mechanisms. In some cases a few attacks were detected. This means that the security level should be increased in order to achieve 100% of security.

6.3. Security Performance. For comparison reasons, the performance of the security mechanisms is expressed in this work in terms of power consumption and latency. The power consumption (P_{DYN}), given by (1) is the sum of P_{NoC} (*low NoC* and *high NoC* power), P_{Keep} (*policy keeper* power), P_{Con} (*configuration control* power), and P_{Mon} (*monitor* power):

$$P_{DYN} = P_{NoC} + P_{Keep} + P_{Con} + P_{Mon}. \quad (1)$$

The *NoC* power is given by (2). It is the sum of P_{Li} (sum of the links power of the *low NoC* and the *high NoC*), P_{int} (interfaces power), and P_{Ri} (*low NoC* and *high NoC* routers power) due to transaction completion [14]. P_{Li} and P_{Ri} are proportional to the *channel utilization rate* and *router utilization rate,* respectively,

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

We developed power models for the main components in the HNoC architecture. We integrated these models into the simulator, taking the architectural and technological parameters into account. The characterization was made under the 65 nm process constraints, 1 volt as a power supply, and a 25°C 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*. Figures 4 and 5 show the latency and power distributions over all the components of our hierarchical architecture. They show that the security interfaces and the policy keeper are the components that consume more time and power, respectively.

Figure 6 represents the average relative execution time of our architecture after completing 4, 8, 16, 32, and 64 tasks of the MiBench benchmarks. The number of processors that execute these tasks varies from 1 to 16. The comparison among the best effort NoC (without security), the simple

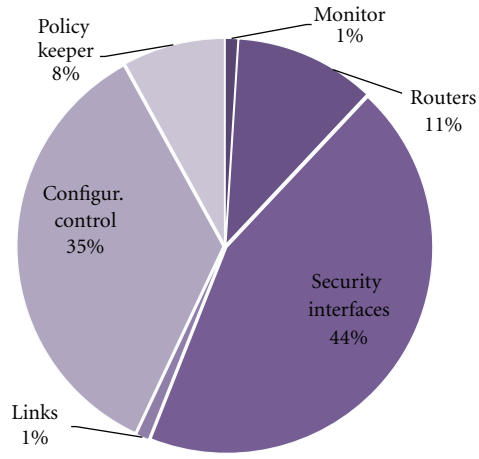

Figure 4: Latency distribution in our architecture.

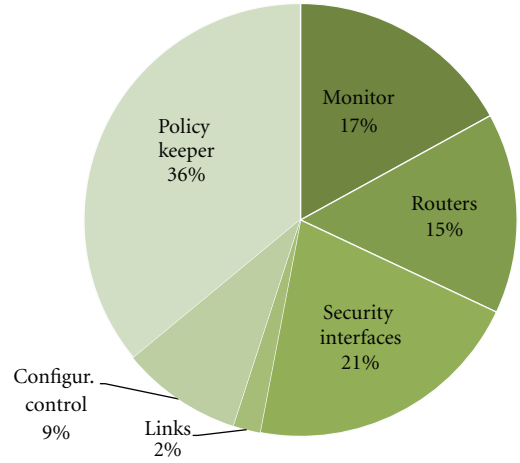

Figure 5: Power consumption distribution in our architecture.

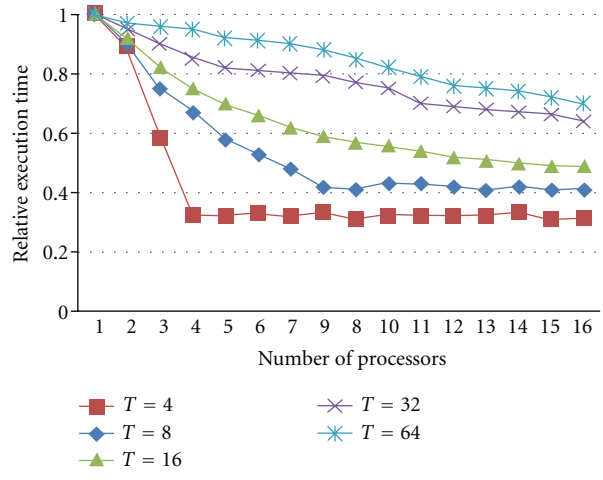

Figure 6: Relative execution time.

Table 6: Implementation penalties.

Parameter	Dynamical approach	Our HNoC approach
Latency increment	4.1%	3.8%
power overhead	19.6%	7.6%
area overhead	26.7%	5.2%

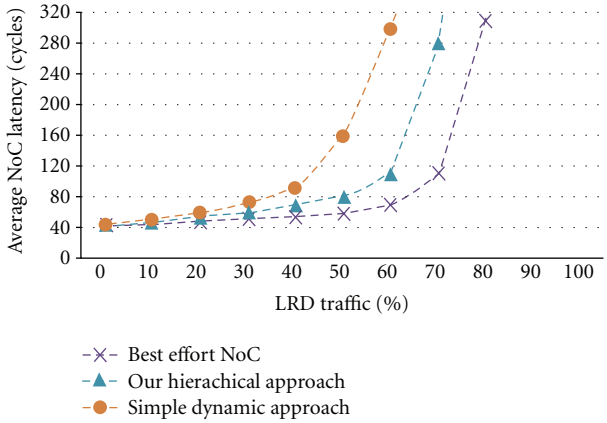

Figure 7: Average NoC latency.

dynamic security approach [5] and the layered security architecture, is shown in Figure 7 and Table 6. The results of Figure 7 are obtained after injecting a percentage of long-range-dependence (LRD) traffic in the MPSoC. Such traffic is typical for the three MiBench applications that we selected [17]. Our layered approach performs better than the simple dynamic security architecture for all percentages of LRD traffic. Table 6 shows the implementation penalties of the simple dynamic security architecture and our layered architecture when compared to the best effort NoC, stressed by uniform traffic. Our approach always achieves the best results.

7. Conclusion

In this work, we proposed a layered NoC security architecture able to support dynamic protection for MPSoC. We implement two security services: access control and authentication. We adopted the QoSS concept that allows the implementation of different security levels. Our work shows that NoC-centric security may take advantage of the distributed property of the NoC. Results show that the inclusion of security issues in the hierarchical NoC performs better that a simple NoC architecture. It reduces the penalties of the latency, power, and area up to 0.3%, 12%, and 21%, respectively, when compared to the simple dynamic solution. Inclusion of the QoSS concept allows the designer to select the most suited among different security levels in order to satisfy both security and performance requirements. In our current architecture, all the applications that are going to be mapped on the MPSoC are known in advance. As a future work, we plan to create virtual security zones, whose size will be defined by the security characteristics of the unkown applications.

References

[1] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini, and R. Zafalon, "Analyzing on-chip communication in a MPSoC environment," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, pp. 752–757, February 2004.

[2] L. Benini, "Application specific NoC design," in *Proceedings of the Design, Automation and Test in Europe (DATE '06)*, vol. 1, pp. 1–5, March 2006.

[3] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 753–760, June 2004.

[4] McAfee, "Annual report 2011," http://www.mcafee.com/.

[5] J. Sepulveda, G. Gogniat, J. C. Wang, and M. Strum, "Dynamic NoC-Based architecture for MPSoC security implementation," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems (SBCCI '11)*, pp. 197–202, 2011.

[6] C. Gebotys and Y. Zhang, "Security wrappers and power analysisforSoC technologies," in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES '03)*, pp. 162–167, 2003.

[7] U. Y. Ogras, J. Hu, and R. Marculescu, "Communication-centric SoC design for nanoscale domain," in *Proceedings of the IEEE 16th International Conference on Application-Specific Systems, Architectures, and Processors (ASAP '05)*, pp. 73–78, July 2005.

[8] L. Fiorin, C. Silvano, and M. Sami, "Security aspects in networks-on-chips: Overview and proposals for secure implementations," in *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD '07)*, pp. 539–542, August 2007.

[9] S. Evain and J. Diguet, "From NoC security analysis to design solutions," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '06)*, 2006.

[10] L. Fiorin, S. Lukovic, and G. Palermo, "Implementation of a reconfigurable data protection module for NoC-based MPSoCs," in *Proceedings od the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS '08)*, April 2008.

[11] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano, "Secure memory accesses on networks-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1216–1229, 2008.

[12] S. Lukovic and N. Christianos, "Enhancing network-on-chip components to support security of processing elements," in *Proceedings of the 5th Workshop on Embedded Systems Security, (WESS '10)*, October 2010.

[13] C. Irvine and T. Levin, "Security as a dimension of quality of service in active service environments," in *Proceedings of the 3rd Annual International Workshop on Active Middleware Services*, 2001.

[14] J. Sepulveda, G. Gogniat, R. Pires, C. Pedraza, W. Chau, and M. Strum, "Multi-objective artificial immune algorithm for security-constrained multi-application NoC mapping," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2012.

[15] B. Grot, J. Hestness, S. Keckler, and O. Multu, "Kilo-NoC: a heterogeneous network-on-chip architecture for scalability

and service guarantees," in *Proceedings of the 38th International Symposium on Computer Architecture*, 2012.

[16] M. Winter, S. Prusseit, and G. P. Fettweis, "Hierarchical routing architectures in clustered 2D-mesh networks-on-chip," in *Proceedings of the International SoC Design Conference (ISOCC '10)*, pp. 388–391, kor, November 2010.

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