Reconfigurable Hardware for High-Security/ High-Performance Embedded Systems: The SAFES Perspective

Guy Gogniat, *Member, IEEE*, Tilman Wolf, *Senior Member, IEEE*, Wayne Burleson, *Senior Member, IEEE*, Jean-Philippe Diguet, *Member, IEEE*, Lilian Bossuet, and Romain Vaslin

***Abstract*—Embedded systems present significant security challenges due to their limited resources and power constraints. This paper focuses on the issues of building secure embedded systems on reconfigurable hardware and proposes a security architecture for embedded systems (SAFES). SAFES leverages the capabilities of reconfigurable hardware to provide efficient and flexible architectural support for security standards and defenses against a range of hardware attacks. The SAFES architecture is based on three main ideas: 1) reconfigurable security primitives; 2) reconfigurable hardware monitors; and 3) a hierarchy of security controllers at the primitive, system and executive level. Results are presented for reconfigurable AES and RC6 security primitives and highlight the value of such an architecture. This paper also emphasizes that reconfigurable hardware is not just a technology for hardware accelerators dedicated to security primitives as has been focused on by most studies but a real solution to provide high-security and high-performance for a system.**

***Index Terms*—Cryptography, hardware monitors, performance and security policies, reconfigurable hardware, secure embedded systems, security primitive.**

I. Introduction

EMBEDDED systems are expected to play an essential role in the future and today they have already spread to many electronic devices from low-end to high-end systems [1]. Ubiquitous computing is becoming a reality, however there is still a major bottleneck to its widespread adoption. Security issues are a serious problem and attacks against these systems are becoming more critical and sophisticated [2]–[4]. Confidentiality and privacy are major concerns for users and today nearly 52% of cell phone users and 47% of PDA users feel that security is the single largest concern preventing the adoption of mobile commerce.[1] New solutions have to be proposed in order to allow for the definition of secure embedded systems. Current technologies are facing several challenges as described by Ravi *et al.* [1]. Architectures will have to meet high performance, energy efficiency, flexibility, tamper resistance, and reliability to enable the vision of ubiquitous computing [1]. From a performance point of view, processing, battery, and flexibility gaps have to be considered. The *processing gap* highlights that current embedded system architectures are not capable of keeping up with the computational demands of security processing. The *battery gap* emphasizes that the current energy consumption overhead of supporting security on battery-constrained embedded systems is very high. The *flexibility gap* shows that an embedded system is often required to execute multiple and diverse security protocols and standards. From an attack point of view, the *tamper resistance gap* emphasizes that secure embedded systems are facing an increasing number of attacks from physical to software attacks, and the *assurance gap* is related to reliability and emphasizes the fact that secure systems must continue to operate reliably despite attacks.

Designing an embedded system architecture dealing with all these requirements is a challenging task. New solutions have to be defined in order to mitigate the costs of security. Reconfigurable technologies can address these challenges and provide efficient security solutions. Their characteristics enable the system to prevent attacks or to react when attacks are detected while guaranteeing the required energy and computation efficiency. An in-depth analysis, however, must be performed to analyze the weaknesses and the strengths of reconfigurable hardware for security and performance.

To tackle this issue we propose the security architecture for embedded systems (SAFES) architectural concept that leverages the security within embedded systems. SAFES is based on reconfigurable hardware to provide high performance and flexibility and relies on hardware monitors to build intrusion detection systems (IDSs). In this paper, we focus on hardware attacks and do not consider software attacks.

The remainder of this paper is organized as follows. Section II reviews hardware attacks and countermeasures. This section also highlights the benefits of reconfigurable architectures to address both performance and security. Section III reviews previous efforts to build secure embedded systems and describes our contribution. Section IV presents the SAFES architecture and shows how the security is enforced and how attacks can be fended off. Section V deals with the AES and RC6 security primitives to illustrate our concepts and to demonstrate their efficiency. Finally, Section VI concludes this paper and draws some perspectives.

Manuscript received May 22, 2006; revised May 10, 2007. This work was supported in part by the French DGA DSP/SREA under Contract ERE 046000010.

G. Gogniat, J-P. Diguet, and R. Vaslin are with the Laboratory of Electronic and REal Time Systems (LESTER), University of South Britanny (UBS)-CNRS FRE2734, 56321 Lorient, France (e-mail: guy.gogniat@univ-ubs.fr; jean-philippe.diguet@univ-ubs.fr; vaslin@univ-ubs.fr).

T. Wolf and W. Burleson are with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003-9284 USA (e-mail: wolf@ecs.umass.edu; burleson@ecs.umass.edu).

L. Bossuet is with the IMS Laboratory, ENSEIRB, University of Bordeaux 1, CNRS UMR5818, 33405 Bordeaux, France (e-mail: bossuet@ixl.fr).

Digital Object Identifier 10.1109/TVLSI.2007.912030

[1][Online]. Available: http://www.epaynews.com/statistics/index.html

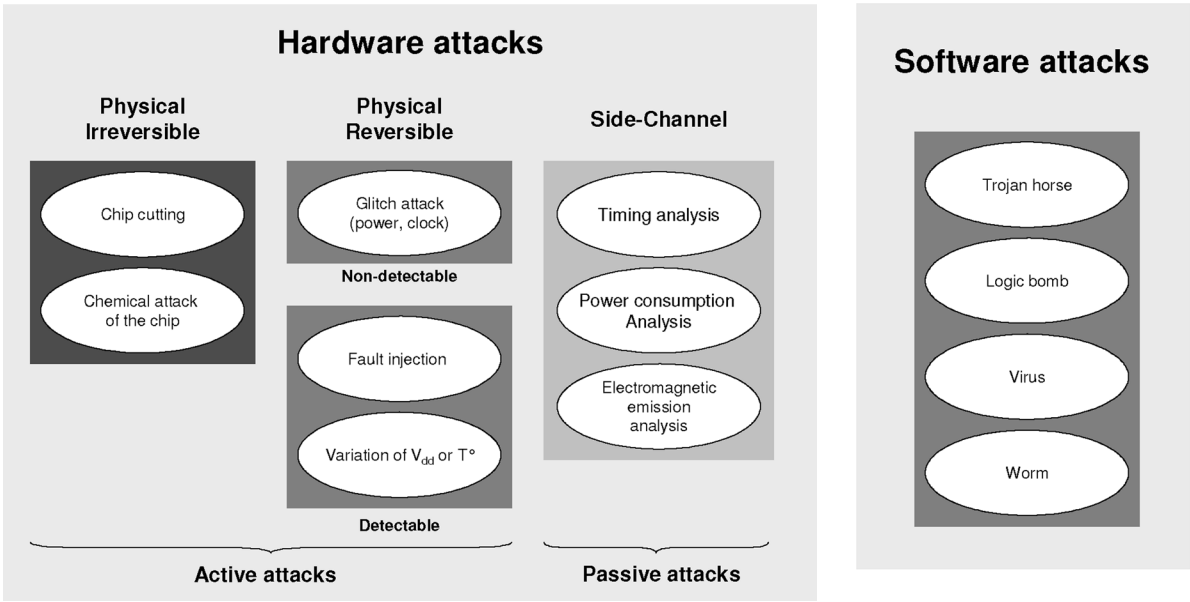

Fig. 1. Classification of attacks against embedded systems.

II. Hardware Attacks and Countermeasures

A. Hardware Attacks

Two main categories of attacks can be considered: active and passive attacks (see Fig. 1). Active attacks correspond to physical attacks. They can be refined into two subcategories: irreversible and reversible attacks. Irreversible attacks (also called invasive attacks) correspond to chip destruction or modification for reverse-engineering (e.g., through chip cutting or chemical attacks) [1]. Reversible attacks (also called noninvasive attacks) consist in punctually moving the device out of its normal modes to force it into a weak state or to gain information from a computation fault [5]. Reversible attacks are, for example, glitch on clock/power, fault injection, or power/temperature reduction. These attacks may or may not be detected at run time. It is difficult to build efficient sensors to detect attacks. If too sensitive, the sensors may not be reliable since they may detect some normal variations that do not correspond to any attacks, and if not sensitive enough, they may not be able to detect effective attacks. If an attack is detected and countermeasures have been defined, the system may react in order not to leak any information (e.g., by erasing a private key).

Passive attacks enable the extraction of secrets by observing properties of the system (i.e., current, power, electromagnetism) while it performs operations [5]. In that case, the system computes normally and the attack relies on the statistical analysis of the leaked information. Examples of passive attacks are timing, power, and electromagnetic emission analysis [6]. In practice, attackers often use a combination of various techniques to achieve their objectives [2].

B. Countermeasures

What conclusions must be drawn from these attacks to increase system security at the hardware level? To be secure a system should do as follows.

- Not provide any information (i.e., data leaks) to disable passive attacks. The system must be **symptom-free** [5].
- Be continuously aware of its states and notably of its vulnerabilities in order to react if necessary. The system must be **security-aware**.
- Analyze its states and its environments in order to detect any irregular activity. The system must embed distributed sensors and monitors to be **activity-aware**.
- Be agile enough in order to react rapidly to an attack or to anticipate an attack. Be agile enough to be able to update security mechanisms as long as attacks evolve. The system must provide **agility**.
- Be tamper resistant in order to resist physical attacks. The system must be **robust** [7], [8].

But, the system must also provide high performance to run applications. Throughput, latency, area, power, and energy are all examples of parameters that are mandatory to run applications. What is the solution; what technology provides these characteristics? Reconfigurable hardware presents several major advantages to deal with both hardware security and performance when compared to dedicated hardware components and processors. Some aspects are not specific to reconfigurable hardware but are more related to design at logic and circuit levels, such as being symptom-free and robust. But one major feature is required when dealing with security, adaptability (this term includes the notions of awareness and agility) that is not provided by dedicated hardware components. Processors provide adaptability through code update but do not meet the high performance requirements. One key feature of reconfigurable hardware that underlies all others is its dynamic property. Dynamic reconfiguration enables the system to react and adapt rapidly in order to provide efficient architecture for performance and security. This reconfiguration can be performed at run time or not, depending on the requirements. It is important to enlarge today's vision of security since reconfigurable hardware may not be a single part of the system but the whole system.

III. Related Work

Existing efforts to promote security within embedded systems mainly deal with processor-based approaches [9]–[11]. These solutions are based on cryptography mechanisms to guarantee integrity and privacy of data and applications. Such solutions are very interesting, however, as demonstrated by Ravi *et al.* [1], it is mandatory to define new alternatives to processor-based approaches as the cost of security using such solutions is very high. Other solutions can be considered using programmable hardware accelerators in order to mitigate the workload of processors. In [12] and [13], the authors propose a cryptography processor or coprocessor which can perform various execution modes and achieve high throughput. However, they do not address the attack issue and the energy efficiency metric is not considered. In [14], the authors focus on architecture support for energy-efficient security. In their work, they deal with security primitives and security protocols but they do not consider the attack issue.

Another alternative is to consider reconfigurable architecture to implement security primitives instead of using a programmable hardware accelerator. Several works have been published using such a solution [8], [15], [16]. They have demonstrated its very high efficiency but none has focused on the mechanisms required to manage the flexibility of these primitives and to detect attacks.

The concept of hardware monitoring has already been used for processor power reduction [17], [18] and recently for power-attack [4], [19]. In [4], Martin *et al.* define the energy monitoring unit (EMU) which performs energy measurements. These values are compared to a set of reference energy signatures to detect when the system is under attack.

The work presented in this paper differs from these efforts in several respects. First, the underlying concept of our approach is to dynamically adapt the security protection in order to deal with dynamic constraints (i.e., attacks, performance, power). We propose an architecture that promotes the design of secure embedded systems by targeting the challenges stated by Ravi *et al.* [1] as presented in the introduction. Our approach allows for the definition of a solution that leverages both flexibility and hardware security within embedded systems. The performance and energy issues are considered by using reconfigurable security primitives which enable the system to provide several tradeoffs depending on the requirements and the security policy. The reliability issue is managed through the use of different implementations from low to high reliability (e.g., fault detection or fault tolerance). Second, we propose a hierarchy of hardware monitors in order to track the activity of the system. In our approach, monitors provide different levels of flexibility which enable an evaluation of the right compromise between accuracy and simplicity which is mandatory to meet embedded system constraints.

IV. SAFES: Security Architecture for Embedded Systems

A. SAFES Architecture: A Proposition

Our approach to protect embedded systems is to provide an architectural support for the prevention, detection, and remediation of attacks. Most embedded systems are implemented as system-on-a-chip devices, where all important system components [processor, memory, input/output (I/O)] are implemented on a single chip. We propose to extend the functionality of such systems to include both reconfigurable hardware and a continuous monitoring system that guarantees secure operations. Through monitoring, abnormal behavior of the system can be detected and hardware defense mechanisms can be employed to fend off hardware attacks. Such an approach has several advantages since application verification and protection are provided in dedicated hardware and not directly inside the application. The security mechanisms can be updated dynamically depending on the application running on the system which guarantees the durability of the architecture.

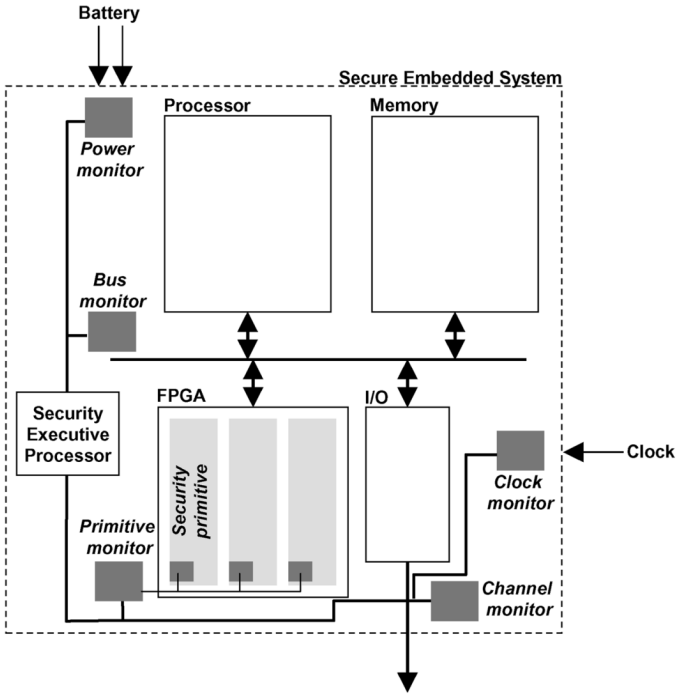

Fig. 2. SAFES. The reconfigurable architecture contains the security primitives and the monitors protect the system.

Fig. 2 presents an overview of the architecture. The FPGA contains several security primitives. These primitives can be dynamically reconfigured depending on the requirements. Several monitors are used to track specific data of the system. The number and the complexity of the monitors are important parameters as directly related to the overhead cost of the security architecture. The role of these monitors is to detect attacks against the system. To provide such a solution, the normal activity (i.e., correct or expected) of the system is characterized to detect irregular behaviors. Autonomy and adaptability have been emphasized to build an efficient security-network of monitors. The monitors should be autonomous in order to build fault tolerant system; if one monitor is attacked the others can still manage the security of the system. The monitors should be distributed to be able to analyze the different parts of the system (e.g., battery, buses, security primitives, communication channel). Each monitor should be authenticated within the network in order to prevent intrusion of malicious monitors.

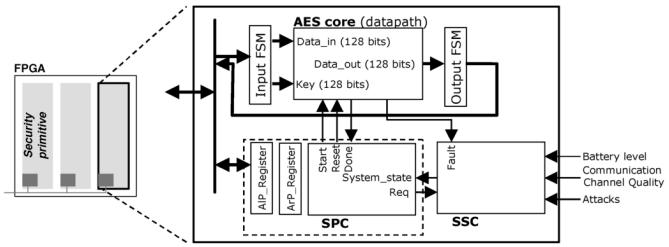

Fig. 3. Security primitive architecture. The SPC manages the flexibility of the primitive and the SSC deals with the detection of abnormal activity using specific sensors.

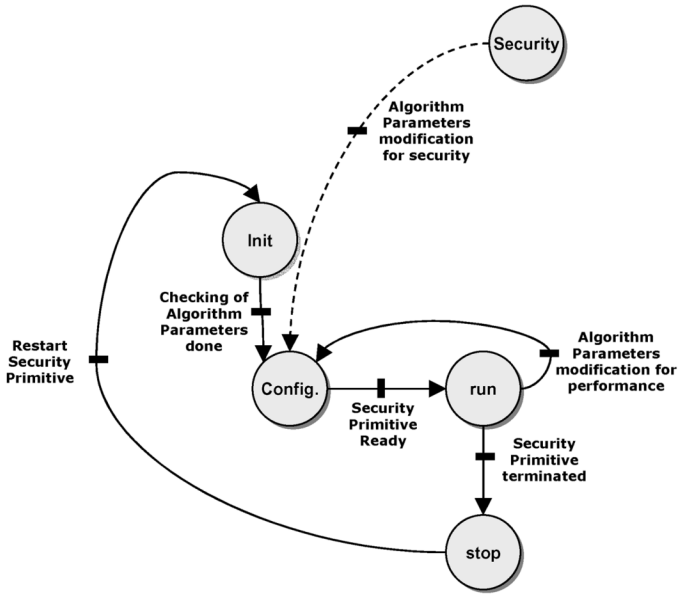

Fig. 4. SPC FSM deals with the different states of the primitive to dynamically adapt the architecture of the primitive.

Different levels of reaction (reflex or global) should be considered depending on the type of attack. Reflex reaction is performed by a single monitor; the response time is very short since no communication between the different monitors is required. Global reaction is performed when an attack involves a large modification of the system in which case the monitors need to define a new global configuration of the system which leads to a longer response time. The monitors should be linked by an on-chip security network. This network is controlled by the security executive processor (SEP) that acts as a secure gateway to the outside world. The SEP provides a software layer to map new monitoring and verification algorithms to monitors. In response to abnormal behavior, the SEP can issue commands to control the operation of the system. For example, it can override the power management or disable I/O operations.

B. Reconfigurable Architecture

The reconfigurable architecture within the system enables the implementation of several security primitives. A security primitive corresponds to an agile hardware accelerator and performs a security algorithm (e.g., cryptography, IP filtering, key management). A device generally embeds several security primitives that work independently. The main goals of these modules are as follows [20]:

- to speedup the computation of the security algorithm compared to software execution;
- to provide flexibility compared to a fixed implementation to be able to update the primitive or to switch from one primitive to another;
- to provide various tradeoffs in terms of throughput, area, latency, reliability, power, and energy in order to meet real time constraints.

Fig. 3 presents the security primitive architecture for a 128-bit AES algorithm. This primitive corresponds to one primitive of the field-programmable gate array (FPGA) and has a dedicated area within the FPGA. Three key components are used to design a security primitive: 1) the security primitive datapath; 2) the security primitive controller (SPC); and 3) the system security controller (SSC) which is a monitor. The SPC is connected to the datapath in order to manage its flexibility. The SPC control tasks are related to the reconfiguration of the datapath to change or adapt its architecture. The SPC is also connected to the system processor (master of the system) in order to define the configuration of the security primitive (a security primitive is a slave within the system). For example, in the case of cryptography it corresponds to the parameters of the algorithm (i.e., key size, mode, and key value). The SSC is connected to the security primitive to monitor the primitive and to check the state of the system to detect if some fault injections or abnormal operations have been performed. The role of the SSC is to detect attacks against the primitive. The SSC is connected to all monitors of the system to analyze the different parts of the system (e.g., battery, buses, other security primitives, communication channel).

C. Detailed Operation Modes of the Primitive

The reconfigurable security primitive is composed of the datapath and the two previous controllers (SPC and SSC) as shown in Fig. 3. The SPC is connected to the system processor through a memory mapped mechanism (i.e., hardware accelerator). Depending on the primitive, different configuration registers are used to define its configuration. These registers provide the algorithm (i.e., execution mode and key size for cryptography algorithm) and architecture parameters (i.e., throughput, area, and reliability). As previously stated, the SPC manages the flexibility of the primitive. When the processor needs a security primitive, it first configures the SPC which starts to check what execution modes can be used. Fig. 4 presents the FSM corresponding to the SPC.

During the *Initialization* state, the SPC polls, via the SSC, the state of the system (e.g., battery level and communication channel quality) in order to define what implementations can be performed within the primitive. Once the algorithm and architecture parameters are checked, the SPC provides this information to the processor. During the *Configuration* state, once the processor has selected the algorithm parameters, the SPC selects the corresponding configuration data (it corresponds to a bitstream) and starts the configuration of the datapath. On the *Run* state, the security primitive is ready to run and to handle the data. While the datapath is running, the SPC regularly checks the state of the system through the SSC to define if the primitive needs to be reconfigured. Once the computation has been

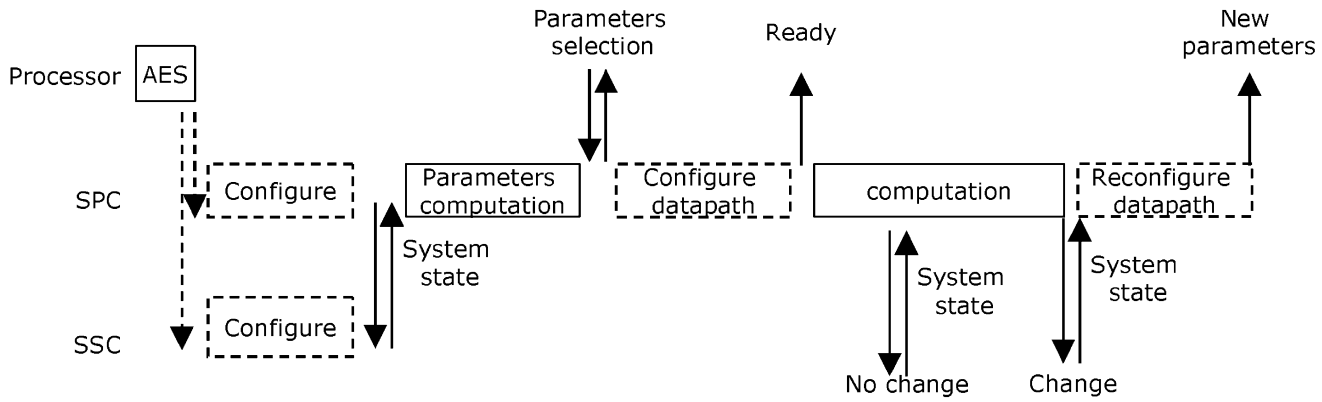

Fig. 5. Processor/security primitive schedule.

performed, the security primitive can be stopped or can be removed from the reconfigurable hardware (*Stop* state). If the security primitive remains within the reconfigurable hardware this state corresponds to an idle state before rerunning the primitive. Finally, the *Security* state is particular in the sense that it is always active. The *Security* state is SSC driven to indicate that a reconfiguration must be done in order to fend off or to anticipate a hardware attack against the primitive. Whatever the state of the SPC, the *Security* state enforces the activation of the *Configuration* state to reconfigure the security primitive with the appropriate parameters.

The execution schedule between the processor, the SPC and the SSC is described in Fig. 5 for an AES example. It highlights when the reconfigurations occur. When the processor needs a security primitive, it first configures the FPGA with the SPC and the SSC. The SPC indicates to the SSC the function it needs to perform. The SSC provides the SPC with data related to the state of the system (e.g., battery state). At the same time the SSC indicates to the SEP the type of primitive it has to monitor, so that the SEP configures it. Then, the SPC sends the processor the configurations it can perform (mode, key size) based on the sensors information. Once the configuration is completed, the SPC is no longer involved in the datapath of the security primitive. However, the SPC continues to poll, via the SSC, the state of the system to check if the mode of the security primitive needs to be changed (the aim is to change the mode if, for example, the battery is running low). At the same time, the SSC monitors the primitive and if something abnormal occurs, then some modifications can be made (for example, to provide fault detection within the security primitive or fault tolerance).

D. Performance and Security Policies

Two main scenarios are considered in our work to protect the system from being pirated and to guarantee the execution of the security protection. The first one is managed by the SSC and deals with attacks (it relies on the security policy) and the second one by the SPC and deals with the flexibility of the primitive (it relies on the performance policy).

In the first scenario, the SSC can interrupt the SPC if an irregular activity is detected within the primitive or the system. In that case the SSC indicates to the SPC the configuration to be implemented. Examples of attacks are: hijacking, denial-of-service (e.g., draining of battery or causing battery to overheat), and extraction of secret information (e.g., user's phone book). In the case of a hijacking attack, the security primitive needs to be reconfigured with a trusted configuration. In the case of a denial-of-service attack, the primitive needs to be enhanced by fault tolerance mechanisms to be able to guarantee its functionality and in the case of an extraction of secret information attack, I/Os of the primitive need to be stalled.

Once an attack has been fended off, the SPC defines a new configuration to provide the best performance tradeoff (performance policy), for example in terms of throughput versus energy when dealing with cryptography. Protected modes like fault tolerant architectures consume more area and power so it is essential to run these modes only when required and not by default to guarantee the power efficiency of the system. Embedded systems are characterized by two main parameters: the power limitation and the evolving environment. Hence, depending on both, the SPC selects which parameters have to be considered. For example, in the case of a best effort performance policy, when the level of battery is low or the channel quality decreases under some thresholds then the SPC reconfigures the primitive with a lower throughput but a better energy-efficient architecture. In the case of guaranteed throughput, the SPC keeps the same parameters even if the thresholds are crossed.

The performance and security policies are essential issues. These policies are very dependent on the primitives and have to cope with their intrinsic specificities. The definition of these policies is beyond the scope of this paper, however designers must pay particular attention to that point. In the results section, we propose a first solution to deal with the performance policy. Extensions could be imagined to face the security policy.

One last important issue is related to the added complexity due to reconfigurability which may introduce new weaknesses for the adversary. Each time the primitive and/or the system evolve, a new configuration needs to be downloaded. An attacker could replace a correct configuration by a fake one. The main solution to prevent this type of attack is to encrypt and to check the integrity of the bitstream (i.e., configuration) [21]. In that case, the attacker is not able to download his own bitstream as it will not be recognized by the system. He may still be able to disturb the system by replacing all trusted configurations by fake ones. A solution is to store all the bitstreams in a trusted memory that cannot be attacked.

TABLE I
RC6 DESIGN SPACE FOR A SYMMETRIC I/O STREAM BANDWIDTH

Archi. ID	0	1	2	3	4	5	6	7
Config.	config. 0	config. 1	config. 2	config. 3	config. 4	config. 5	config. 6	config. 7
KG	1				2			
Type	**Alternate Computation** *ciphering and deciphering sequentially*				**Parallel Computation** *ciphering and deciphering in parallel*			
Pipe. stages	1	2	4	8	1	2	4	8
Throughput (Mbits/s)	128	256	512	1024	278	556	1066	1765
Slices	2196	3003	4081	6548	2404	3186	4394	6845
Area (% of the total amount)	16%	21%	30%	45%	17%	22%	32%	47%
Power (W)	1.6	3.4	5.1	8.1	2.4	6	7.1	9.8
Energy efficiency (Gbits/J)	0.08	0.07	0.1	0.13	0.09	0.09	0.15	0.18

V. SECURITY PRIMITIVE AND MONITORS: THE AES AND RC6 CASE STUDY

To demonstrate the concepts presented in this paper, we have defined two agile security primitives and two hardware monitors. Our case study deals with the AES [22] and the RC6 [23] algorithms since both have been evaluated by the National Institute of Standards and Technology to replace the DES one. AES and RC6 are expected to be two of the major cryptography algorithms within IPSec which is a framework of different standards for ensuring secure private communications over the Internet [24]. The major advantage of IPSec is its flexibility since it allows for negotiation of algorithm choices and configurations between the communicating parties. The algorithm parameters of AES or RC6 are defined during the main mode and the quick mode security association steps of IPSec. The parameters negotiated in these phases and the current session keys are used to transmit data during the secure data transfer step.

All the experimentations have been conducted using a Xilinx Virtex-II Pro FPGA device.[2] The implementations have been performed using the Xilinx ISE Foundation 6.3i tool and the power estimations have been done using the Xilinx XPower 6.3i tool. The FPGA is connected to the processor and the memory through a bus. The different bitstreams (each bitstream corresponds to a configuration) are stored in the memory. The two registers within the SPC contain, respectively, the algorithm and architecture parameters (see Fig. 3). In our case, the algorithm parameters are related to the type of algorithm (i.e., AES, RC6), to the execution mode of the primitive (i.e., feedback, nonfeedback), and to the key and data sizes (i.e., 128 bits). The architecture parameters are focused on the reliability (i.e., no, fault detection, fault tolerance), on the throughput, the area (use rate of the device), and the energy consumption.

In the following sections different points are analyzed. First, the RC6 security primitive is analyzed from a performance point of view (see Section V-A). In that study, we have focused on the performance policy to dynamically adapt the datapath of the primitive in order to face different throughputs. Then Section V-B provides a comparison between several implementations of the AES datapath to define the performance and the cost of security. Section V-B1 describes a bus monitor that tracks the access to the keys stored in the memory in order to detect hijacking. Finally, Section V-C discusses the efficiency of the whole AES security primitive.

[2][Online]. Available: www.xilinx.com

A. RC6 Architecture Monitoring for Performance Policy

Complex encryption algorithms like RC6 require high-performance architectures when high-speed connections are needed. However, in practice bandwidth requirements fluctuate with application needs. Thus, using a reconfigurable security primitive, area and power can be adapted to application needs. In this study, we have focused on the performance policy to target a self-adaptive security primitive. The performance policy is based on a two-step strategy. The first step is performed offline, it is the exploration of efficient architectures, which ends with the selection of a small set of solutions. The second step is performed online, it consists in dynamically selecting the architecture parameters according to application and user requirements. In the next sections, we present these two steps applied to the RC6 algorithm, we consider bandwidth needs as system reference to be followed up.

1) RC6 Datapath Implementation Comparison: The RC6 algorithm needs 20 sub-keys computed by a key generator, each sub-key is applied during a round iteration on a 128-bit input data. We derive configuration options around three main architectural decisions. The first one is the number of pipeline stages, a two-stage architecture means that the data(i) is encrypted (respectively, decrypted) with the sub-key j when the data($i+1$) is encrypted (respectively, decrypted) with the sub-key $j+1$. The second one is the number of key generators (*KG*) and the third one is the period for round computation (p). With two *KG*s, encryption and decryption can be computed simultaneously during a period equal to p. With only one *KG*, the output and input streams proceed alternatively which leads to a global period equal to $2p$. In Table I, we present the architecture configurations we have implemented. With the assumption that input and output streams have the same throughput we consider eight solutions from 128 to 1765 MBits/s. The *KG* cost is not really significant from an area point of view but can still have an impact on the performance when power consumption is the most important criterion. The eight solutions provide a set of architectures available to dynamically adapt the architecture to performance requirements. Further optimizations could have been performed to improve the performances of these architectures. For this study, we have focussed more on the adaptivity of the

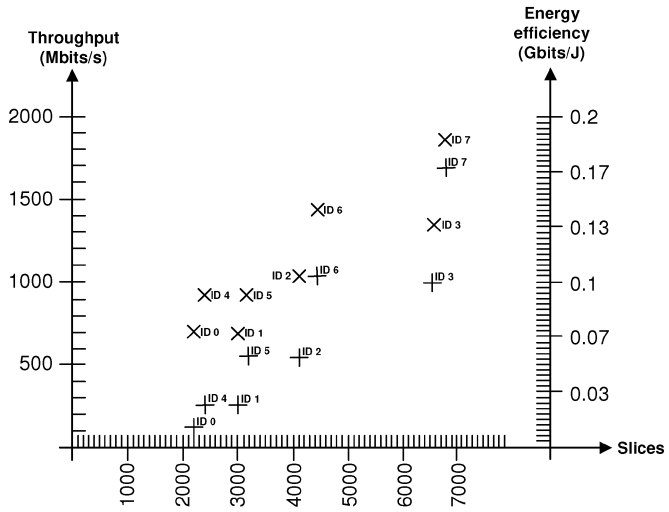

Fig. 6. RC6 design space: architectures ID0 to ID3 represent alternate computation of ciphering and deciphering and architectures ID4 to ID7 represent parallel computation of ciphering and deciphering. Chart in blue represents energy efficiency versus slices and chart in black represents throughput versus slices.

system rather than on targeting forefront performances. However, explored solutions are representative of achievable performances [25]. Solutions ID0 to ID3 correspond to an alternate computation with one *KG*. The number of pipeline stages varies from 1 (i.e., no pipeline) to 8. Each time the depth of the pipeline is multiplied by two, the throughput is also multiplied by two. Solutions ID4 to ID7 correspond to a parallel computation, thus the throughput is almost doubled compared to alternate solutions. Energy efficiency which represents the throughput per energy (Gbits/J) is presented in Table I as this metric is important for embedded systems. Even if parallel computations consume more area, they still provide better energy efficiency. Fig. 6 highlights throughput versus slices and energy efficiency versus slices charts. Each architecture provides different tradeoffs and depending on the requirements the system may switch from one solution to another. Architecture ID6 is an interesting solution as it provides both high energy efficiency and high throughput.

2) Performance Policy Based on a Close-Loop Control Approach: The control within the SPC is designed as a close-loop control (it is used in the *run* state of our FSM). In Fig. 7, we present its implementation based on four modules: the Observer which indicates the required throughput, the configuration adaptor that gives the maximum throughput for the current parameters, and a proportional (*K*)-integrator regulator that enables a smooth error fluctuation. In practice, the implementation can be greatly optimized; it is implemented as follows to meet the performance policy. The Observer provides the number of the desired configuration for two parameters: S_m and S_M (respectively, low and high threshold). When the buffer size is lower than S_m the output is -1, if it is greater than S_M the output equals $+1$ and 0 otherwise. If the input (namely the output of the PI integrator) is lower than $-T$ (which means that the S_m threshold has been crossed) then a slower configuration is selected. On the contrary, if the input is greater than T (which means that the S_M threshold has been crossed) a faster solution is selected.

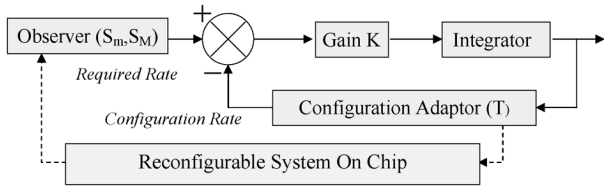

Fig. 7. Close-loop configuration control within the SPC.

In Fig. 8, we present simulation results obtained with Simulink. We consider solutions 4–6 from Table I with 1, 2, and 4 pipeline stages, respectively. In this case study, we set the regulation parameters as follows: *K* is set to $1/256$, it means that an average is computed using 256 encryption iterations. S_m and S_M are equal to 2 and 14, respectively, with a buffer length equal to 16×128 bits (the buffer is implemented with lookup table (LUT) configured as RAM16b). *T* is set to 0.75 which means that the Observer output must be equal to $+1$ (respectively, -1) more than 187 times during the 256-iteration frame to decide a faster (respectively, lower) reconfiguration. In Fig. 8, the parameters evolve from configuration 4 (no pipeline) up to configuration 6 (four pipeline stages) in order to follow the throughput requirements.

The lower curve of the upper chart in Fig. 8 (throughput violation) shows the gap between the required and available rates only when the Observer output is equal to $+1$ (respectively, -1) more than 75 of consecutive 256 iterations. Each time a threshold is crossed a reconfiguration is performed.

B. AES Datapath Implementation Comparison

In this second study, we have focused on both performance and security policies. Three different configurations have been implemented to show the flexibility provided within the primitive, feedback mode (FB), feedback mode with fault detection (FB_FD), and feedback mode with fault tolerance (FB_FT). A 128-bit key has been considered. Fault detection mechanisms enable the system to detect if a fault occurs during the computation of the AES algorithm but without correcting the result. A parity-based technique has been used to detect the fault [26]. Fault tolerance mechanisms provide a tamper resistant architecture. We have considered a TMR technique as it corresponds to a common solution [27]. Fig. 9 illustrates the architectures of these primitives.

Each solution corresponds to different levels of performance in terms of area, throughput, and power (see Table II). The fault tolerance solution is the most secure one but the area and energy overheads are very high (respectively, 6302 slices and 1673 mW). Fault detection using parity code does not lead to a significant difference in area and power consumption, respectively, $+2.1\%$ of slices and -2.7% of power consumption compared to a nonsecured implementation in feedback mode. For these implementations the throughput is almost equivalent to 400 Mbits/s.

Another metric is interesting to compare these implementations, energy efficiency (gigabits per joule). Feedback with or without fault detection provide the same efficiency. Fault tolerance guarantees the security of the primitive but has a high overhead in energy efficiency. Thus, fault detection is a good

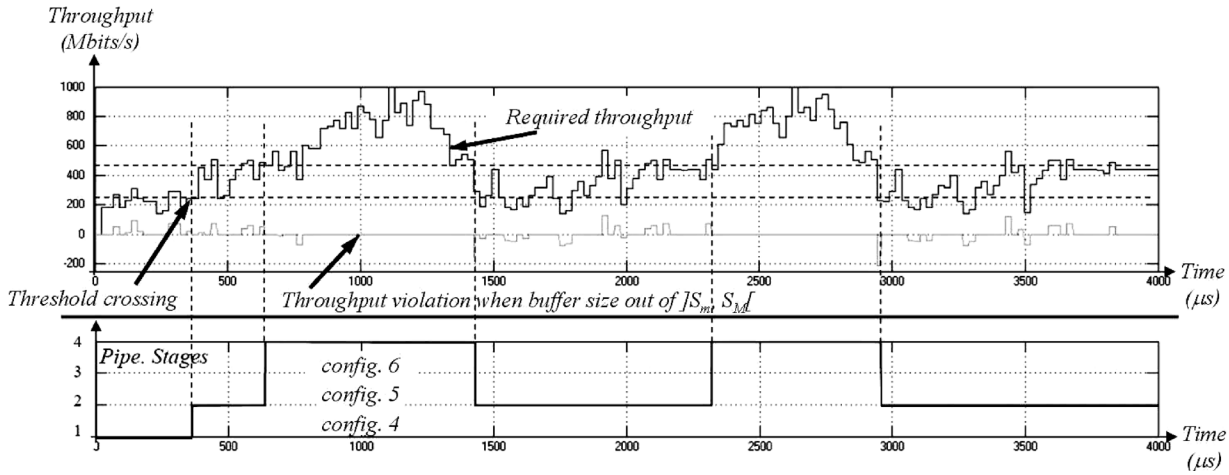

Fig. 8. Simulink simulation of the configuration control. When the input buffer use rate is not meeting the requirements more than 187 times during a 256-iteration frame (threshold crossing), a reconfiguration is performed. Upper chart of the figure represents the required throughput and when there is a threshold violation. Lower chart of the figure represents the various configurations of the RC6 primitive during the computation.

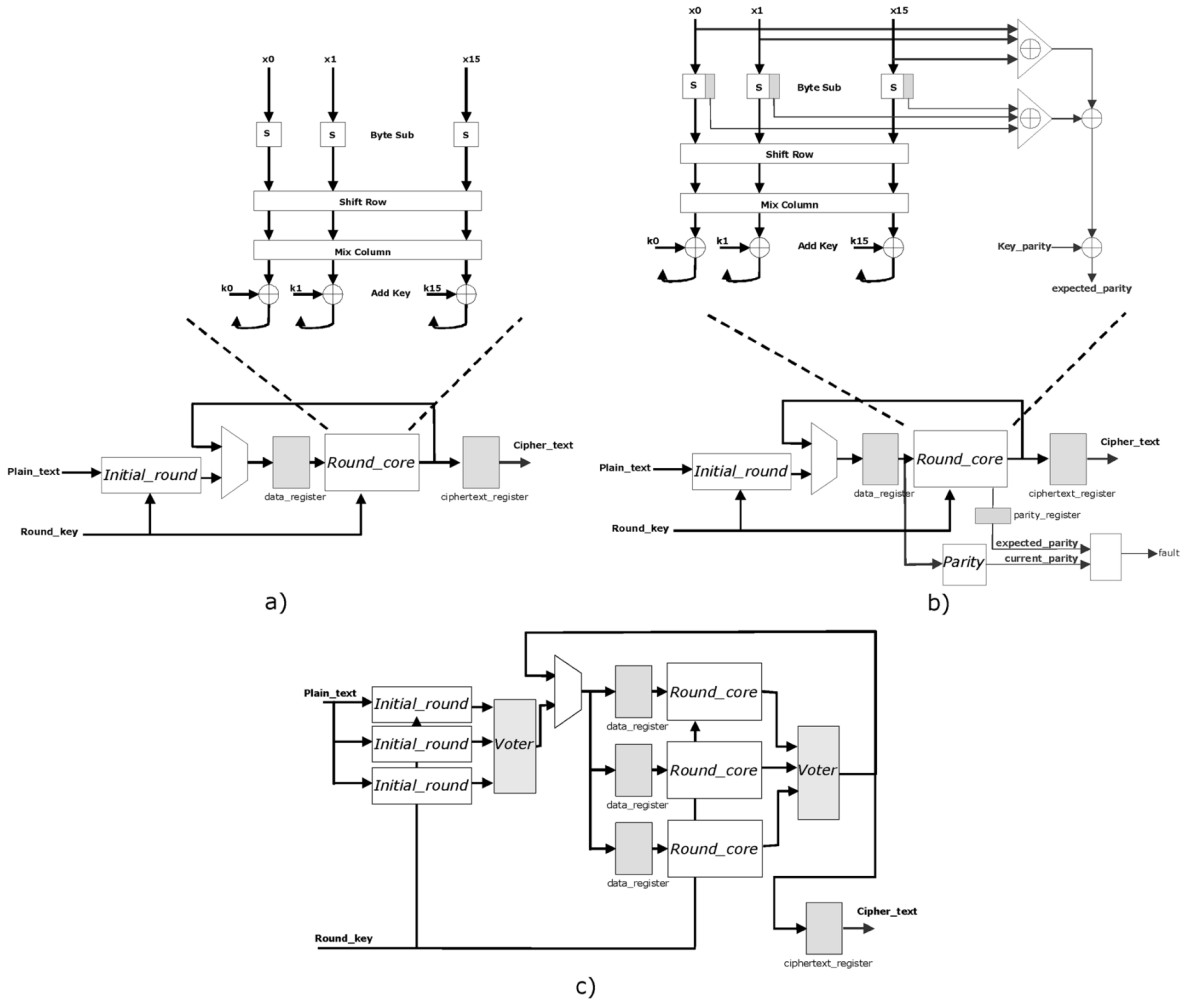

Fig. 9. AES core architecture for the three primitives: (a) AES core without protection; (b) AES core using parity-based technique (fault detection); and (c) AES core using TMR technique (fault tolerance).

compromise to guarantee the performance and to increase the security of the primitive and could be considered as an implementation by default. Further optimizations could have been performed to improve the performance of these architectures. However, explored solutions are representative of achievable performances [15], [20].

TABLE II
PERFORMANCE COMPARISON OF THE FOUR AES CONFIGURATIONS (I.E., DATAPATH). EACH CONFIGURATION CORRESPONDS TO A SPECIFIC TRADEOFF BETWEEN THE SECURITY LEVEL AND THE PERFORMANCE

AES version	Slices (% of the total amount)	Slices (% compared to FB)	Period (ns)	Frequency (MHz)	Power (mW)	Power (% compared to FB)	Energy (nJ)	Throughput (Mbits/s)	Throughput (% compared to FB)	Energy efficiency (Gbits/J)
feedback mode (FB)	2192 (16%)	-	26.4	37.8	996	-	316	403.7	-	0.4
feedback mode with fault detection	2240 (16%)	+2.1	25.3	39.4	970	-2.7	295	420.9	+4	0.4
feedback mode with fault tolerance	6302 (46%)	+65.2	25.2	39.6	1673	+40.5	507	422.2	+4.4	0.25

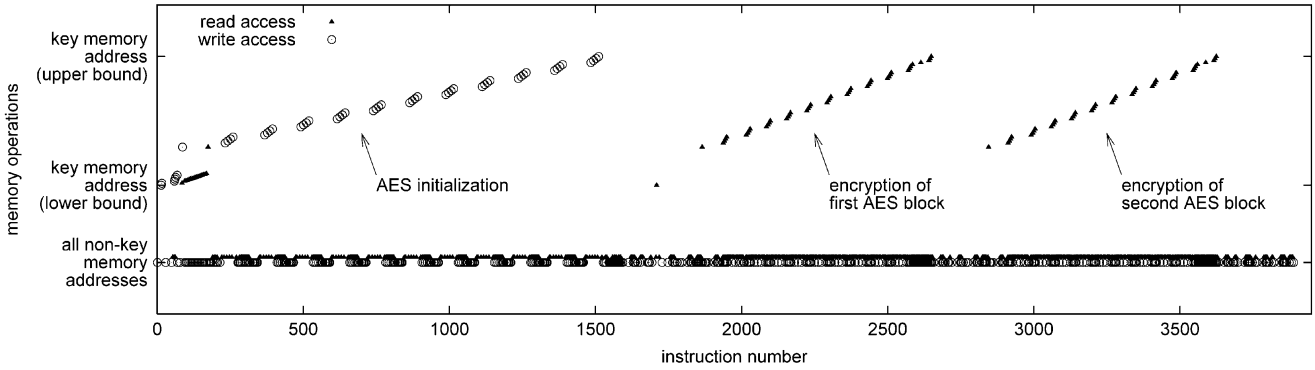

Fig. 10. Monitoring of the bus: Access to the keys are highlighted which enables the monitor to detect some abnormal activities.

1) Bus Monitoring: Tracking the activity on the bus corresponds to an interesting way to analyze the operation of the system. In our case, we have defined a monitor that spies on the address bus. Once the AES primitive starts the encryption, the access to the key memory addresses is very regular (see Fig. 10). The first sequence corresponds to the generation of the sub-keys from the cipher key. Then, each sequence represents the encryption of one block of data. The access to ten key memory addresses is carried out for each block which corresponds to the ten rounds of the AES algorithm.

Two complementary scenarios have been considered to detect abnormal activity. The first one is based on a counter which compares the offline profile with the run-time memory access. For that purpose, we have stored the offline profile in a table (using Huffman-based coding to code the data) and we have implemented a counter that counts sequences of nonkey memory access and key memory access. When the sequence matches the content of the table, no alarm is raised. When there is a mismatch then there is a problem and the monitor indicates that there is potentially an attack. The monitor is flexible in the sense that the number of blocks to be encrypted is not known statically, so we have stored only one sequence that we compare as long as some blocks need to be encrypted. The second scenario is based on the combination of different data. Indeed, if the key memory addresses are found on the bus and no encryption is running then it corresponds to hijacking of the secret keys.

The complexity of the bus monitor depends on the monitoring technique as all source and destination addresses of reads and writes to/from keys memory can be analyzed. In our case, we have considered a simpler solution as we only count the number of accesses and we do not consider the exact keys memory addresses. This solution leads to a small area overhead for the monitor but provides a less accurate approach. Dynamic reconfiguration of the monitor could be considered to adapt the accuracy of the monitor depending on the state of the system.

C. AES Reconfigurable Security Primitive Efficiency

The three previous feedback implementations, feedback mode (FB), feedback mode with fault detection (FB_FD), and feedback mode with fault tolerance (FB_FT), have been considered for the definition of the whole AES security primitive. We have defined three reconfigurable modules which are the datapath, the SPC, and the SSC. An area constraint has been associated to each module as shown in Fig. 11. In this experiment, we have considered a single primitive but there is no limitation regarding that point.

The communication between the modules have been performed through three bus macro which are predefined Xilinx hard IPs [28]. One bus macro is used to provide the *fault* signal between the datapath and the SSC (see Fig. 3). The two others are used between the datapath and the SPC and correspond to control signals (e.g., *start, reset, done*). The reconfiguration is performed by the SPC through the ICAP interface which allows for the dynamic and partial self-reconfiguration of the FPGA [29]. Fig. 11 shows the three possible configurations. The area overhead for the fault tolerant implementation is high compared to the two other solutions. The SPC and SSC modules are very small and remain constant for the three configurations. Their complexity is small compared to the datapath so that they represent a negligible area overhead. For this study, we

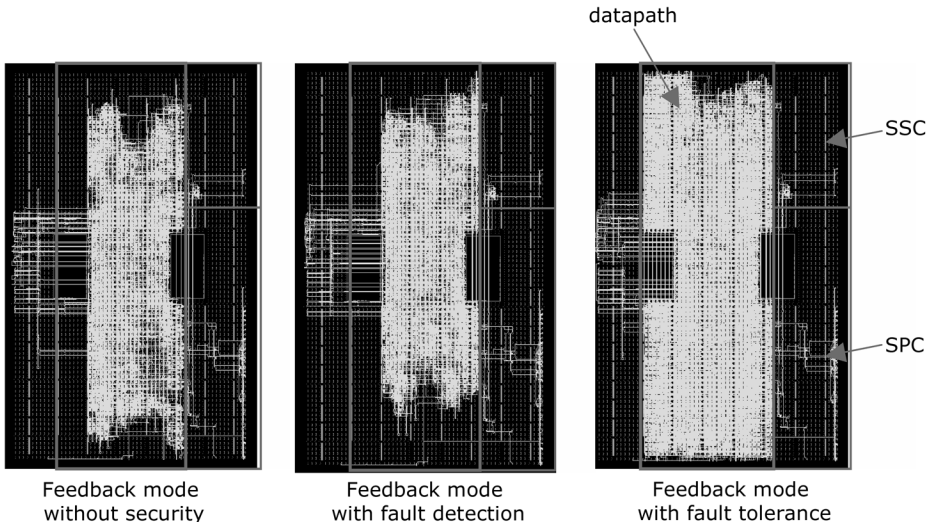

Fig. 11. Layout of the three configurations of the AES reconfigurable security primitive. Three modules are defined which are the datapath, the SPC, and the SSC.

have considered very simple performance and security policies which are basically based on a threshold crossing or on an attack or a fault detection. For real embedded systems, these policies might use more advanced techniques. However, the overhead costs should remain small compared to the datapath.

Concerning the performance of such a solution, the reconfiguration time is directly related to the size of the bitstream. The full bitstream which is used at power-up represents 1415 kB and the three partial bitstreams for the FB, FB_FD, FB_FT configurations are respectively equal to 356, 356, and 463 kB. In our case, the clock of the ICAP interface is 50 MHz which leads to an average reconfiguration time around 8 ms. Each time a reconfiguration is performed there is also an overhead cost in terms of power. However, this overhead is negligible for the FPGA power core and represents an increase of around 6% for the FPGA power supply [30].

VI. Conclusion and Future Work

Reconfigurable hardware provides important features to target high-security/high-performance embedded systems. However, today these features are only partially explored and it is important to extend the vision of security using reconfigurable hardware to where the whole system may be embedded within a reconfigurable platform. In this paper, an analysis of the major issues dealing with security at the hardware level is proposed. We have presented the SAFES architecture which relies on dynamic reconfiguration to improve the security within embedded systems. The main concepts that drive the definition of this architecture are to continuously monitor the operation of the system to detect abnormal behavior and to use reconfigurable hardware to provide various levels of protection and performance. The combination of both approaches is a novel contribution that enables the system to target both security standards and defenses against attacks. Results on the AES and RC6 algorithms show that the flexibility of our solution enables the definition of an energy-efficient solution while addressing the security issue. Future work includes the definition of other monitors to detect attacks (control flow monitor within embedded processor). This point is important as low complexity solutions have to be defined to keep the total system cost reasonable. Solutions based on signatures (using offline profiling techniques) seem promising. Performance and security policies are also main issues that need to be further investigated.

Many questions still remain open on how to make security commonplace in embedded systems. In particular, dealing with reconfigurable hardware and defining the overhead cost for security mechanisms at the hardware level are challenging questions. We believe that our work provides an important step towards a security design that uses reconfigurable hardware to meet high-security and high-performance embedded system requirements.

References

[1] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 461–491, Aug. 2004.

[2] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper resistance mechanisms for secure embedded systems," in *Proc. IEEE Int. Conf. VLSI Design*, 2004, pp. 605–611.

[3] D. Dagon, T. Martin, and T. Staner, "Mobile phones as computing devices: The viruses are coming!," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 11–15, Oct./Dec. 2004.

[4] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, "Denial-of-service attacks on battery-powered mobile computers," in *Proc. 2nd IEEE Pervasive Comput. Conf.*, 2004, pp. 309–318.

[5] S. Guilley and R. Pacalet, "SoC securiy: A war against side-channels," *Annals Telecommun., Syst. sur puce electron. pour les telcommun.*, vol. 59, no. 7–8, pp. 998–1009, Jul./Aug. 2004.

[6] F.-X. Standaert, L. Van Oldeneel tot Oldenzeel, D. Samyde, and J.-J. Quisquater, "Power analysis of FPGAs: How practical is the attack?," in *Proc. Int. Conf. Field-Program. Logic Appl. (FPL), LNCS 2778*, 2003, pp. 701–711.

[7] R. Anderson and M. Kuhn, "Tamper resistance—A cautionary note," in *Proc. 2nd USENIX Workshop Electron. Commerce*, 1996, pp. 1–11.

[8] T. Wollinger and C. Paar, , W. Rosenstiel and P. Lysaght, Eds., "Security aspects of FPGAs in cryptographic applications," in *New Algorithms, Architectures, and Applications for Reconfigurable Computing*. Norwell, MA: Kluwer, 2004.

[9] D. Lie, C. A. Thekkath, and M. Horowitz, "Implementing an untrusted operating system on trusted hardware," in *Proc. 19th ACM Symp. Operat. Syst. Principles*, 2003, pp. 178–192.

[10] E. Suh, J. Lee, S. Devadas, and D. Zhang, "Secure program execution via dynamic information flow tracking," MIT, Boston, Memo-467, 2003.
[11] X. Zhuang, T. Zhang, and S. Pande, "HIDE: An infrastructure for efficiently protecting information leakage on the address bus," in *Proc. 11th Int. Conf. Arch. Support for Program. Lang. Oper. Syst. (ASPLOS XI)*, 2004, pp. 72–84.
[12] A. Hodjat and I. Verbauwhede, "High-throughput programmable cryptocoprocessor," *IEEE Micro*, vol. 24, no. 3, pp. 34–45, May/Jun. 2004.
[13] D. Oliva, R. Buchty, and N. Heintze, "AES and the cryptonite crypt processor," in *Proc. CASES*, 2003, pp. 198–209.
[14] P. Schaumont and I. Verbauwhede, "Domain-specific codesign for embedded security," *IEEE Computer*, vol. 36, no. 4, pp. 68–74, Apr. 2003.
[15] A. J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 4, pp. 545–557, Aug. 2001.
[16] A. Dandalis and V. K. Prasanna, "An adaptive cryptography engine for internet protocol security architectures," *ACM Trans. Des. Autom. Electron. Syst. (TODAES)*, vol. 9, no. 3, pp. 333–353, Jul. 2004.
[17] E. Chi, A. M. Salem, R. I. Bahar, and R. Weiss, "Combining software and hardware monitoring for improved power and performance tuning," in *Proc. 7th Ann. Workshop Interaction Between Compilers Comput. Arch. (INTERACT-7)*, 2003, pp. 57–64.
[18] J. S. Seng, E. S. Tune, and D. M. Tullsen, "Reducing power with dynamic critical path information," in *Proc. 34th Int. Symp. Microarch.*, 2001, pp. 114–123.
[19] D. Nash, T. Martin, D. Ha, and M. Hsiao, "Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices," in *Proc. 2nd Int. Workshop Pervasive Comput. Commun. Security*, 2005, pp. 141–145.
[20] G. Gogniat, W. Burleson, and L. Bossuet, "Configurable computing for high-security/high-performance ambient systems," *Lecture Notes Comput. Sci.*, vol. 3553, pp. 72–81, Jul. 2005.
[21] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for SRAM FPGA bitstreams," in *Proc. 11th Reconfigurable Arch. Workshop (RAW)*, 2004, pp. 146–153.
[22] J. Daemen and V. Rijmen, *The Design of Rijndael AES-The Advanced Encryption Standard*. New York: Springer-Verlag, 2002.
[23] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 block cipher, Version 1.1" Aug. 20, 1998 [Online]. Available: ftp://ftp.rsasecurity.com/pub/rsalabs/rc6/rc6v11.pdf
[24] S. Kent and R. Atkinson, "RFC2401: Security architecture for the internet protocol," Nov. 1998.
[25] J.-L. Beuchat, "FPGA implementations of the RC6 block cipher," in *Proc. 13th Int. Conf. Field Program. Logic Appl. (FPL)*, 2003, pp. 101–110.
[26] K. Wu, R. Karri, G. Kuznetsov, and M. Goessel, "Parity based concurrent error detection for the advanced encryption standard," in *Proc. Int. Test Conf. (ITC)*, 2004, pp. 1242–1248.
[27] C. Carmichael, "Triple module redundancy design techniques for virtex FPGAs," Xilinx, San Jose, CA, Appl. Note 197 (XAPP197), Nov. 2001.
[28] Xilinx, San Jose, CA, "Two flows for partial reconfiguration: Module based or difference based," Appl. Note XAPP290, Sept. 2004.
[29] M. Ullmann, B. Grimm, M. Huebner, and J. Becker, "An FPGA runtime system for dynamical on-demand reconfiguration," in *Proc. 11th Reconfigurable Arch. Workshop (RAW)*, 2004, pp. 135–142.
[30] J. Becker, M. Huebner, and M. Ullmann, "Power estimation and power measurement of Xilinx virtex FPGAs: Trade-offs and limitations," in *Proc. IEEE Symp. Intregr. Circuits Syst. Des.*, 2003, pp. 283–288.

Guy Gogniat (M'04) received the B.S.E.E. degree from the FIUPSO, Orsay, France, in 1994, the M.S.E.E. degree from the University of Paris, Sud Orsay, France, in 1995, and the Ph.D. degree electrical and computer engineering from the University of Nice-Sophia, Antipolis, France, in 1997.

He is currently an Associate Professor in electrical and computer engineering with the Laboratory of Electronic and Real Time Systems (LESTER), University of South Britanny (UBS)-CNRS FRE2734, Lorient, France, where he has been since 1998. In 2004, he spent one year as an invited Researcher with the University of Massachusetts, Amherst, where he worked on embedded system security using reconfigurable technologies. His work focuses on managing the development of the EDA framework "Design Trotter" for design space exploration and combining the design space exploration with technology mapping over reconfigurable architectures. He also conducts research in high level methodologies and tools for FPGA utilization, performance estimation, and FPGA security.

Tilman Wolf (S'99–M'02–SM'07) received the Diploma in informatics from the University of Stuttgart, Stuttgart, Germany, in 1998, the M.S. degree in computer science, the M.S. degree in computer engineering, and the D.Sc. degree in computer science from Washington University, St. Louis, in 1998, 2000, and 2002.

Currently, he is an Associate Professor with the Department of Electrical and Computer Engineering, the University of Massachusetts, Amherst. He is engaged in research and teaching in the areas of computer networks, computer architecture, and embedded systems. His research interests include network processors, their application in next-generation Internet architectures, and embedded system security.

Dr. Wolf is a member of the ACM. He has been active as a program committee member and an organizing committee member of several professional conferences, including IEEE INFOCOM and ACM SIGCOMM. He is currently serving as treasurer for the ACM SIGCOMM Society.

Wayne Burleson (M'84–SM'01) is a Professor with the University of Massachusetts, Amherst, where he has been since 1990, performing research, teaching, design, and consulting in the general area of VLSI circuits and systems. He has worked in industry as a chip designer with VLSI Technology, a Consultant for startups Datafusion and Tensorcomm, and as a Consultant with the Intel Massachusetts Design Group. His work has developed new designs, analysis, methodologies, tools, and prototypes that confront the VLSI challenges typical of the era. In the 1980s, he worked on efficient systems for signal processing. In the early 1990s, he developed high-performance CMOS timing schemes, moving on to low-power techniques at several different levels in the late 1990s. Most recently he has worked on reliable signaling in long interconnects, and the challenges associated with variations and noise. His approaches typically cross multiple levels of abstraction and question conventional design assumptions, while attempting to retain a practical viewpoint. He has published over 120 papers in these areas.

Jean-Philippe Diguet (M'05) received the M.S. and the Ph.D. degrees from Rennes University, Rennes, France, in 1993 and 1996, respectively. His thesis addressed the estimation of hardware complexity and algorithmic transformations for high level synthesis.

Since 1998, he has been with the Laboratory of Electronic and REal Time Systems (LESTER), University of South Britanny (UBS)-CNRS FRE2734, Lorient, France, where he started the Design Trotter research project in design space exploration at both algorithmic and system level, and since 2004, he has been a CNRS Researcher with LESTER. He was a Postdoctoral Fellow with IMEC, Leuven, France, where he worked on memory hierarchy decisions for power optimization. From 1998 to 2002, he has been an Associated Professor with UBS University, Lorient, France. In 2003, he initiated a technology transfer and cofunded the dixip company in the domain of wireless embedded systems. His current work focuses on various topics in the domain of embedded system design: design space exploration extended to heterogeneous embedded real-time systems, environment-aware and self-adaptive HW/SW architectures under QoS, power and performance constraints, RTOS new services for managing reconfiguration, CAD tools for application aware and secured NOC design, and reconfigurable embedded systems as pervasive computing components.

Lilian Bossuet was born in France in 1975. He received the B.S. degree in electrical engineering from the ENSEA, Cergy-Pontoise, France, the M.S. degree in electrical engineering from INSA-University of Rennes, Rennes, France, and the Ph.D. degree in electrical engineering and computer sciences from the University of South Brittany, Lorient Cedex, France.

Since 2005, he has been an Associate Professor with the Department of Electrical and Computer Science, the ENSEIRB, University of Bordeaux, Bordeaux, France. His main research activities at the IMS Laboratory focus on digital systems design and hardware security for embedded systems. His research interests also include reconfigurable computing, high level methodologies and tools for SoC, and FPGA performances estimation. He also works on analog-to-digital converter characterization.

Romain Vaslin received the M.S. degrees in electronics and computer science from ESEO, Angers, France, and from IRCCyN, Nantes, France, in 2005. He is currently pursuing the Ph.D. degree with LESTER Laboratory, University of South Brittany, Lorient, France.

His research interests include electronics security in embedded systems especially with reconfigurable devices.