Journal of Electrical Engineering, Electronics, Control and Computer Science
JEEECCS, Volume 3, Issue 7, pages 37-45, 2017

Principles, Architectures and Challenges for Ensuring the Integrity, Internal Control and Security of Embedded Systems

Fernando Georgel Bîrleanu
Faculty of Electronics, Communications and Computers
University of Pitesti
Pitesti, Romania
birleanu.fernando@gmail.com

Nicu Bizon
Faculty of Electronics, Communications and Computers, University of Pitesti, Pitesti, Romania
nicubizon@upit.ro
[2]University Politehnica of Bucharest, Doctoral School, Bucharest, Romania
nicubizon@yahoo.com

***Abstract* – This paper is a review of the main principles, architectures and challenges designed for ensuring the integrity, internal control and security of Embedded Systems based of Field-Programmable Gate Array (FPGA) technology, such as cryptographic devices and modules or monitoring devices of distant sensitive data.**

Keywords: principles, architectures, challenges, internal control, integrity, security, embedded, FPGA, attacks, hackers.

I. INTRODUCTION

Embedded systems are "information processing systems embedded into enclosing products" [1], [2].

Nowadays embedded systems represent around 90% of all commercial and research computing devices [3].

Field-Programmable Gate Arrays (FPGA) have become one of the first choices of computation in most of today embedded systems because these chips are right in the middle of fast time to market, flexibility, field programmability, performance, cost efficiency, real-time guarantees and simple design cycle which are typical specifications of most of embedded systems [4]. These systems are more and more accepted and used because a single chip combines communication interfaces, digital components, processors and digital logic design [5].

Some of the main advantages that FPGAs come with are [5]:

- Reconfigurability – designs can be reconfigured and modified whenever the developer wishes;
- High speed – implementations use fast clock rates;
- Parallelism – parallelism computation involves simultaneously execution tasks without computation loading. Developer can instantiate on the same FGPA chip multiple modules or hardware implementations;
- Reliability – no operating system to change design uptime;
- IP (Intellectual Property) protection – hard to apply reverse engineering methods on the hardware implementation.

According to [2], [6] and [7] a concept named "cyber-physical system" (CPS) has been introduced in 2006 and shows the strong synergy between embedded systems and the physical environment.

This concept represents systems that incorporate computational algorithms, physical components and networking. Figure 1 shows a concept of CPSs.

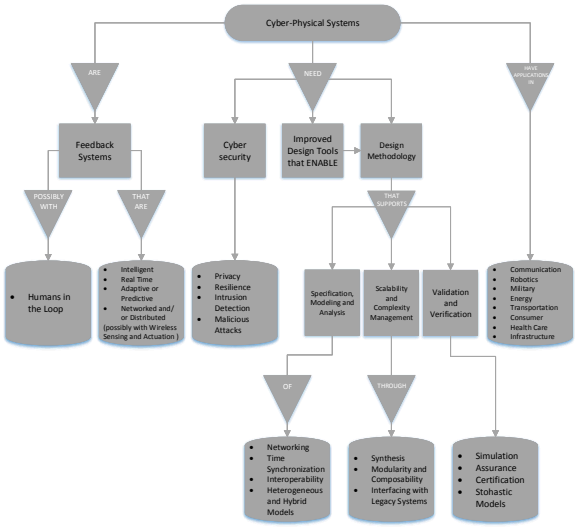

Figure 1. Concept of Cyber-Physical Systems – adapted from [8].

As we can see in the figure above this concept joins three essential parts [9]:

- The first part is a definition of these systems;

- The second part refers to methods and tools imperative to implementation;
- The third part exemplifies main areas CPSs find their applicability.

These systems focus on computational and physical processes integration and represent the next generation of embedded systems, while the old generation concentrates more on computing processes [9].

National Science Foundation believes that this new technology can and will change the way people interfere with engineered systems in the same way that Internet did between people and information [10].

Therefore, embedded systems are part of our lives because we find them in current critical areas that require high protection and security.

Thus, the structure of this study is the following: the first section is an introduction in FPGA-based embedded systems, the second one presents how an embedded system is designed and the main features that these systems should achieve, the third part is about attackers and attacks against embedded systems, the fourth part shows the main principles, architectures and challenges used to secure these systems, the fifth part reveals future research work and the last section concludes the paper.

II. EMBEDDED SYSTEMS - DESIGN AND FEATURES

In general, the process for designing embedded systems involves four steps [11]:

- Creating the architecture;
- Implementing the architecture;
- Testing the system;
- Maintaining the system.

Figure 2 shows the steps above but in detail:

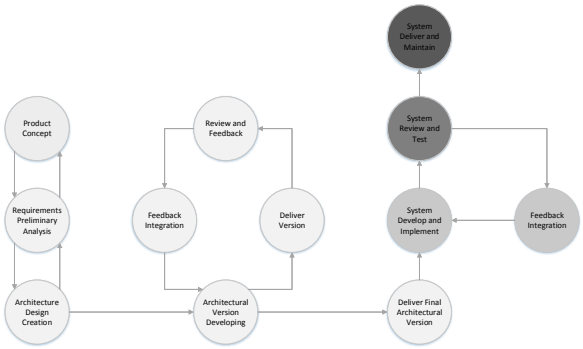

Figure 2. Designing process of embedded systems – adapted from [11].

This kind of architecture is only a structure formed with interacting elements with certain properties and it not shows implementing details such as hardware circuit design or source code, but which can solve some challenges or barriers in developing the system, such as [11]:

- Determining the integrity of the system;
- Cost limitations;
- Defining the concept of the system;
- Determining available and necessary resources;
- Salability;
- Marketability.

An embedded system or a cyber-physical system based on FPGA technology usually use a system design flow as described in Figure 3 [2].

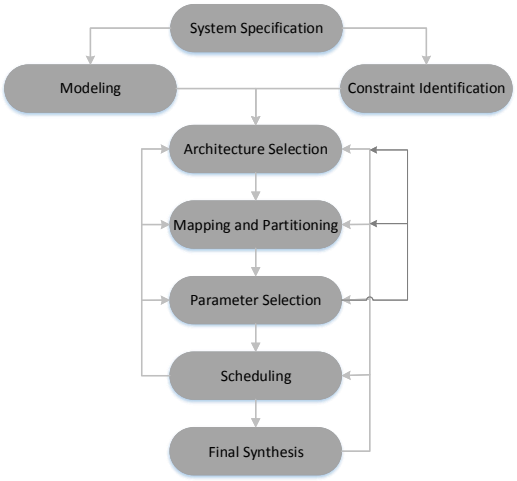

Figure 3. System design flow of FPGA-based ESs and CPSs – adapted from [2].

The System Specification step refers to general tasks and requirements the system must perform. The second step is divided in two parallel steps: Modelling that fragments step one in smaller punctual modules and Constraint Identification that provides a suite of constraints depending on the environment, performance goals, security demands or operational development setting. In the third step, Architecture Selection, the developer selects the most appropriate hardware baseboard that can support future requirements. Mapping and Partitioning refers to achieve optimum performance where the designer divides assignments among available processing units at step four, while step five, Parameter Selection, looks for high performance and efficiency by adapting system execution parameters. Step six, Scheduling, is for finding best solutions with returning to previous steps in opposite cases. Final synthesis, also the final step, reviews solutions in step six and if requirements and constraints are accomplished, the design will start with final synthesis of the low-level implementations (hardware/ software).

According to hardware and software specs, system design flow and specific requirements, main features of these embedded systems are [1]:

- An embedded system must be efficient (power consumption, run-time efficiency, design cost, code size, weight);

- An embedded system must be reliable:
  - Available – the probability that a system is available;
  - Maintainable – the probability that a broken system can be fixed in precise time;
  - Safe – the property that a broken system will not cause any injuries;
  - Secure – the property that sensitive confidential data remains confidential and original communication is guaranteed.
- These systems are related with the physical environment through sensors and actuators. Therefore, data is collected and specific environment is controlled;
- These systems are committed to a certain application;
- Many of them are hybrid systems including analog and digital modules;
- Embedded systems must meet real-time constraints;
- These systems dispose of dedicated user-interface.

After we have seen how embedded systems are designed and their features, we must find out how these systems are attacked, who are the attackers and how attacks against embedded systems are classified.

III. ATTACKERS AND ATTACKS AGAINST EMBEDDED SYSTEMS

Modern FPGA-based embedded systems control and process more and more sensitive data and information. Therefore, less or more equipped hackers are determined and motivated to steal Intellectual Property of these systems through more and more sophisticated attacks. How important to protect systems is given by how sensitive is information stored in a device inclined to attacks [12].

In the last decades attacks experienced high complexity making it very difficult for developers to build defense mechanisms. This is not thanks to high scientific knowledges of the attackers, but due to the easiness to find tools in the on-line world and free market [12], [13]. Figure 4 shows the dependencies between attacker`s knowledges and complexity level of attacks in the last decades.

Usually, an attack follows an action diagram as described in Figure 5. This diagram is also suitable for another area of systems and devices that can be attacked (such as breaking computer networks or intelligent transport systems), but in this case, is adapted for breaking FPGA-based embedded systems [14].

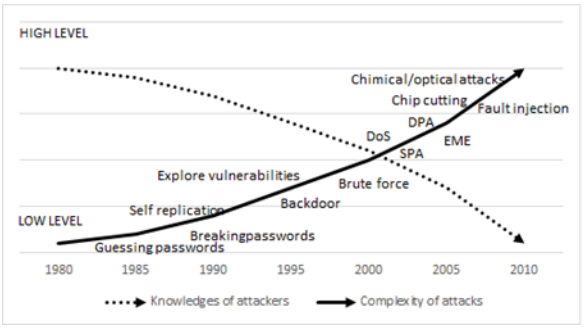

Figure 4. Dependencies between attacker`s knowledges and complexity level of attacks in the last decades – adapted from [12], [13], [29].

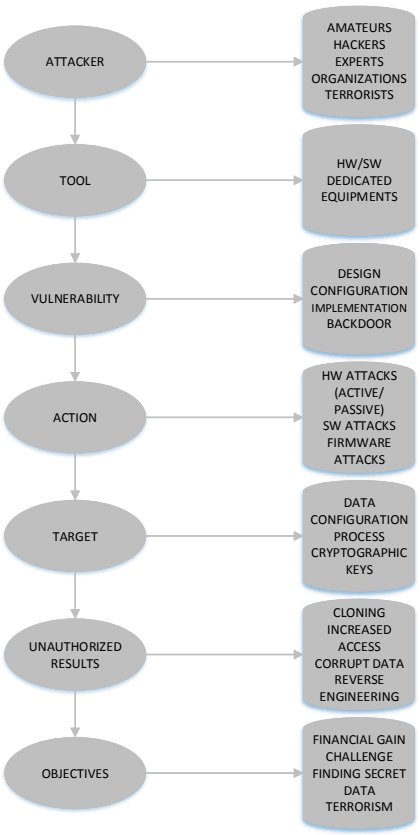

Figure 5. Diagram of an attack – adapted from [14].

Attacks variate but those which succeed run through [12]:

- Studying and exploring for vulnerabilities or any information that can be used to attack;
- Breaking all defense mechanisms and tools;
- Modifying critical security parameters;
- Disperse to other systems or much sensitive parts of the device/ system;
- Finding and assimilate sensitive data or interest information and/ or freezing the system.

In the moment that a developer has identified the benefits and possible attacks against his system he also must find potential attackers that can try to break it according to what they are looking for. Thus, those that can attempt to penetrate embedded systems are divided in four categories [15]:

- Remote or distant attackers – they don`t have physical access to the device but they can have access to a similar device in order to develop the attack. This category relies on exploring software vulnerabilities and user errors. Rising software complexity of embedded systems goes to more bugs that this type of attacker can exploit;
- Security experts – this category aims for finding and developing attacks that can be used for various systems. Technically speaking, most capable to attacks are security experts/ specialists and criminal organizations;
- Trusted developers – this type of attackers refers to the employees of companies that deliberately have stolen secret information for specific purposes. A simple way to justify this category of attacks is that it is easier to bribe someone with access to secret information than to apply reverse engineering to a piece of silicon;
- Device or system owners – this category has the objective to break own systems and to see how security mechanisms and tools handle with attacks and how they resist against attacks. Usually, this type of attackers is highly motivated but pretty low in technical and scientific knowledges, thereby, they appeal to on-line environment where they find a lot of tools to develop desired types of attacks. The risk is that they don`t know what backdoors or viruses these tools can come with.

Attackers described above use three methods to attack [15]:

- Method one – hack attack – the attacker is capable only of software attacks (malware, viruses);
- Method two – shack attack – refers to low budget attacks using free market tools where, although physical access to the device exists, the equipment they use cannot perform complex attacks;
- Method three – lab attack – in this case the attackers have access to laboratory equipment and for instance they can apply reverse engineering to a device at the level of transistor details or analyze cryptographic keys.

As we know who are the attackers and how they attack classing attacks follows, depending on three main considerations. According to [16], a new classification of embedded systems attacks is created:

- Programmability level:
  - Hardware attacks (reverse engineering, hardware Trojan, data monitoring, traffic monitoring, DoS, EME, SPA, DPA);
  - Software attacks (viruses, software Trojan, packet reply, buffer overload);
  - Firmware attacks (OS kernel).
- Integration level:
  - Intellectual Property level attacks (modifying source code);
  - Chip level attacks (cloning, bitstream reverse engineering, fault attacks);
  - Board level attacks:
    - Invasive attacks (requires physical access to the board and chemical/ mechanical/ imaging processing techniques are used);
    - Semi-invasive attacks (works only for single or doubled layered board simply through scanning top and/ or bottom of the board and using CAD – computer-aided design);
    - Non-invasive attacks:
      - Passive attacks (observing and monitoring data);
      - Active attacks (modifying clock signal and supply voltage).
- Life cycle phase:
  - Design phase attacks (cloning, adding undesired components);
  - Fabrication phase attacks (copies);
  - After-production attacks (design cloning, extracting confidential information).

The most important challenges in defending against attacks are [12]:

- Speed of the attack;
- Complexity of an attack;
- The simplicity of used tools.

Table 1 shows most common attacks to break embedded systems and mobile devices depending on several features, while Table 2 represents a map of these attacks related to domains and areas where systems can be found.

Attack`s number	Attack	Criterion
Attack 1	Static code analysis	Developer attacks
Attack 2	White-box method for finding data computation bugs	
Attack 3	White box structural testing	
Attack 4	Finding hardware-system unhandled uses in software	Control system attacks
Attack 5	Errors on HW to SW and SW to HW signal interface	
Attack 6	Long duration control attack	
Attack 7	Breaking control laws and software logic	
Attack 8	Forcing unusual bug cases	
Attack 9	Braking software with hardware	Hardware software attacks
Attack 10	Finding bugs in HW-SW communications	
Attack 11	Breaking software error recovery	
Attack 12	Testing integration and interface	
Attack 13	Finding errors in software-system fault tolerance	
Attack 14	Breaking digital software communications	Mobile and embedded software attacks
Attack 15	Finding data errors and bugs	
Attack 16	Bugs in system-software computation	
Attack 17	Using stimulation and simulation for software attacks	
Attack 18	Errors in timing interrupts	Timing attacks
Attack 19	Finding time-related bugs	
Attack 20	Time-related scenarios	
Attack 21	Performance attacks	
Attack 22	Finding support documentation errors	User interface attacks
Attack 23	Finding missing or wrong alarms	
Attack 24	Finding bugs in help documentation	
Attack 25	Finding errors in apps	Smart and mobile phone attacks
Attack 26	Testing mobile and embedded games	
Attack 27	Attacks on app-cloud dependencies	
Attack 28	Penetration authentication attack test	Mobile/embedded security
Attack 29	Stealing device data	
Attack 30	Spoofing attacks	
Attack 31	Attacking viruses on the run in factories and companies	
Attack 32	Combinatorial tests	Generic attacks
Attack 33	Attacks against functional bugs and errors	

Table 1. Attacks against embedded systems and mobile devices– adapted from [17].

Context area	Attacks to consider	Possible attacks
Telecom (general)	1, 2, 3, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 28, 33	29, 30, 31, 32
Switch	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 28, 33	11, 22, 23, 24, 29, 30, 31, 32
Cell	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 28, 33	11, 22, 23, 24, 29, 30, 31, 32
Smart device	1, 2, 3, 6, 8, 10, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33	4, 5, 7, 12, 17, 32
Cell device	1, 2, 3, 6, 8, 12, 14, 15, 18, 19, 20, 21, 22, 23, 24, 25, 26, 33	4, 5, 7, 12, 17, 28, 30, 32
Smartphone	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33	-
Tablet	1, 2, 3, 6, 9, 10, 12, 14, 15, 16, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33	-
Handheld equipment	1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33	13, 17
Medical (info system)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33	11
General info support	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 33	32
Medical (life critical)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 33	32
Patient support	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 33	11, 32
Inserted in humans	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 33	11, 32
Mission critical	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 31, 32, 33	-
Life critical	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 ,16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 32, 33	31
Robotics	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33	-
Industrial/ buildings	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 31, 32, 33	-
Machines	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 31, 32, 33	-
Lights	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 27, 28, 29, 30, 31, 32, 33	22, 23, 24
Heating, ventilation, AC	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 25, 26, 27, 28, 29, 30, 31, 32, 33	22, 23, 24
Building control	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 32, 33	31
Avionics	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 32, 33	-
Automotive (general)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33	-
Automotive (control)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 30, 32, 33	-
Rockets	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 27, 28, 30, 32, 33	-
Spacecraft	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 30, 32, 33	-
Aircraft	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,	-
Home/ office	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33	-
Control	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33	-
Gaming	1, 2, 3, 6, 11, 12, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33	18, 19, 20
Transportation (traffic control, railroad)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 30, 31, 32,	-
Utilities (energy, water, sewer, others)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 30, 31, 32, 33	-

Table 2. Map of attacks against embedded systems related to domains and areas – adapted from [17].

IV. METHODS FOR ENSURING INTEGRITY, INTERNAL CONTROL AND SECURITY OF EMBEDDED SYSTEMS

Usually, successful attacks against embedded systems give four main problems for these systems [18]:

- Modification – unauthorized tampering;
- Interruption – disrupted availability;
- Interception – unauthorized access;
- Fabrication – creation of fictitious system.

To develop and implement secure systems we must see the most used methods by bad intended people for breaking systems [19], [20], [29]:

- Cloning – when someone manages to reproduce the entire system with the certain purpose to sail it. Many components are free and identical functions of the device are not so important as they can be duplicated;
- Reverse engineering – when someone succeeds in finding out how the system works and uses this or even improves system`s functionality;
- Overbuilding – the producer makes more copies in detail and often he has even the configuration to load in the chip, so it is very easy to sell extra devices for extra financial gains.

Secure systems are dependent on security principles that protect information in computer systems first introduced in 1975 by Saltzer and Schroeder [21]:

- Economy of mechanism – this principle refers to the fact that embedded systems and cyber-physical systems are constraint (resource constraints). Due to this, complex security mechanisms increase the risk for missed configured components making it hard to verify its security;
- Least privilege – any user or program should use minimum privilege set in order to execute his tasks;
- In-depth defense – single security measure is not enough to defeat concomitant attacks;
- Isolation – this principle aims for isolating subsystems, critical resources and data from public or external access. Hardware and software exchanges between users, usually leads to security breaches;
- Minimization of attack area – the number of entry points (administrators, external access services, running services) in a system should have a minimum value with requested functionality maintained;
- Separation of privilege – efficient systems should be accessed by more than one step (two-factor authentication for example);
- Psychological acceptability – this appears when security mechanisms fail. If a particular security mechanism hinders a system's usability or accessibility, the users may reject it or look for a way to circumvent it.
- Open design – experts believe that open designs benefits of more trust than closed designs. This depends on who uses it and who owns it.

After all this above, security of FPGA-based embedded systems relies on [22], [29]:

- Device security – device physical protection;
- Data security – loaded configuration must meet critical security goals such as data authenticity or data integrity.

The security of these systems, roughly, refers to the security of FPGAs chips. So, looking at the configuration storing technology, chips can be:

- SRAM FPGA (Figure 6) – these are volatile chips and requires external memory (flash or EPROM) to store the bitstream. The route from the external memory to the chip needed to download the bitstream at power up is exposed to attacks [19], [23], [29], [30], [31];
- Flash FPGA (Figure 7) – can provide physical design security and any attempt to gain bitstream access involves opening the package but they are slower than SRAM FPGA [19], [23], [29], [30], [31];
- Antifuse FPGA (Figure 8) – can provide physical design security and they are difficult to read back but they are slower and less dense than SRAM FPGA [19], [23], [24], [29], [30], [31].

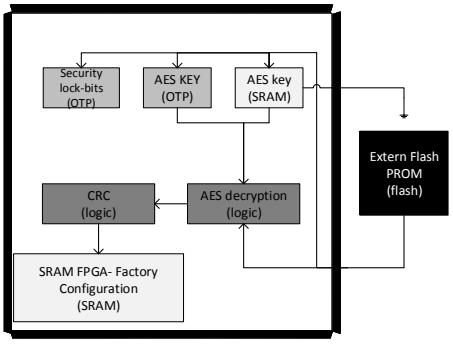

Figure 6. SRAM FPGA security architecture – adapted from [22], [29].

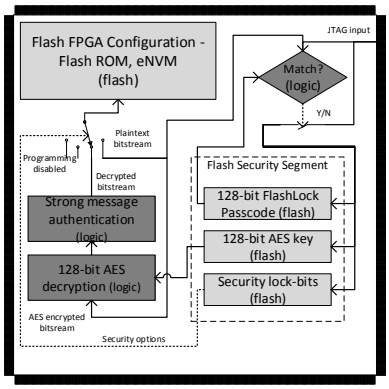

Figure 7. Flash FPGA security architecture– adapted from [22], [29].

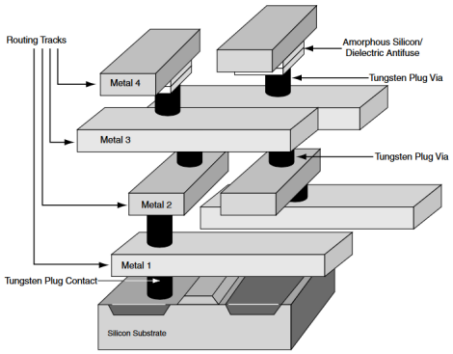

Figure 8. Cross section antifuse programmer – caption from [24].

The methods to secure data in embedded systems translate to security applications, as above [22], [25], [26], [29]:

- Data integrity – maintaining and assuring data consistency and accuracy along entire life cycle;
- Data confidentiality – implementing methods that keep away data from wrong people;
- Data availability – this method is assured through severe hardware system maintenance, correct functional state of the system and daily upgrades and updates;
- Data encryption/ decryption – this is the primary method for protecting data confidentiality, data integrity and data authenticity;
- Identification/ authentication by hardware/ software/ people – this is the first line in defense against attacks.
- Anonymity;
- Accountability;
- Non-observability, non-repudiation;
- Anti-cloning;
- Anti-overbuilding;
- Safe token;
- Cryptographic key management;
- Anti-piracy;
- Anti-reverse engineering;
- Anti-tamper.

In the last years, experts developed a security architecture (Figure 9) for embedded systems that joins the properties of security term, necessarily services and approaches methods of security elements. All starts with a strategy plan followed by [27]:

- Security policy and strategy – is one of the most important features that leads to future system requirements. It contains necessarily security properties, violations that can affect the system, risks that can show up and approaches for specific violations. Security policy and strategy helps security services;
- Services – refers to safety and security mechanisms and devices used to ensure system and data security and also security for the peoples that work on developing and implementing the system. Services are divided in security services and security support services. While the first category controls potential or actual security properties violations and depends on available physical, procedural, automated and management support mechanisms for implementing these services, the second category relies on inferior infrastructure (security policy and strategy);
- Mechanisms and implementations – these depend on commercial products and other tools used for their implementation.

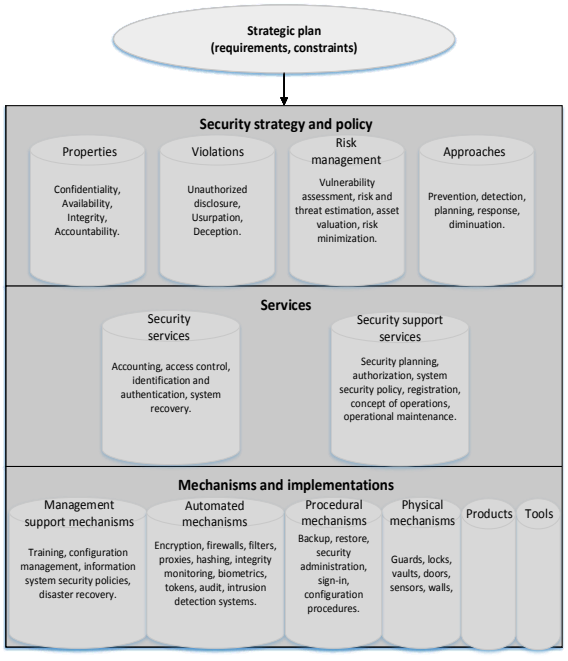

Figure 9. Security architecture for embedded systems – adapted from [24].

Approaches of potential crimes and violation against FPGA-based embedded systems are highly

important while developing the system and represent the challenges for ensuring the integrity, internal control and security of these systems. These approaches are [27]:

- Planning – refers to prevention, detections and response procedures and usually these are found in the documentation of the system;
- Prevention- protects security attributes of the system by blocking unwanted activities that can compromise the properties form security policy and strategy;
- Detection – detects and identifies undesired activities;
- Diligence – this method refers to anticipated security measures that improve the security overall;
- Response – actions and procedures used after undesired detected actions.

But, after all this above, the architecture from Figure 9 comes up with some question marks. To implement, produce, administrate, maintain and use an embedded system as described so far there is a large number of participants involved and their interest in security fluctuate which goes to conflicts. While some supports cooperation others focus only autonomy, depending on security interests (methods used to secure the system) [28].

Interest in confidentiality, anonymity, non-observability, detecting modifications and violations accentuates autonomy, while availability, accountability, non-repudiation, evidence and interception accentuates cooperation between participants [28].

If autonomy and cooperation are close to involved participants, at the level of security, integrity and internal control implemented in a system, the impact is on requirements and technical properties that the system must answer to such as performance or utility. Figure 10 shows how these properties and requirements support or block one another for proper functioning of the embedded system.

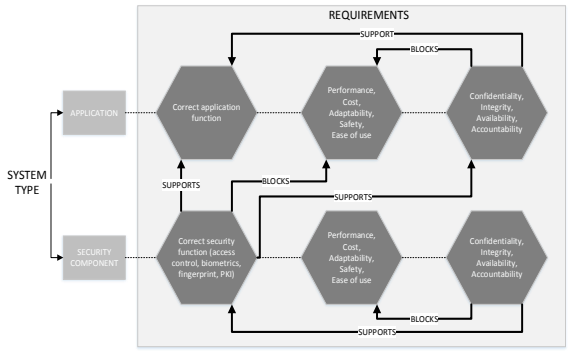

Figure 10. Impact between the requirements of embedded system – adapted from [27].

V. FUTURE WORK

Future work focuses on detailing the methods for ensuring the integrity, confidentiality, and availability of FPGA-based embedded systems with simulation and practical results, on implementing confidentiality and integrity techniques and designing a secure and cost-effective solution for a FPGA-based embedded system.

This research is the first step in designing and implementing new methods and techniques for secure hardware, design security and data security of FPGA-based embedded systems focusing on those with valuable and sensitive data passing through such as cryptographic devices.

VI. CONCLUSION

Modern FPGA - based embedded systems are processing, controlling and collecting sensitive information that motivates attackers to steal intellectual property of these systems thorough more and more sophisticated attacks.

The principles, architectures and challenges that are the base for ensuring the integrity, internal control and security of FPGA-based embedded systems relies on finding an optimal solution that can accomplish required requirements, meet constraints and that can offer imposed security principles.

Future security characteristics must keep an eye on three main features: the need to talk about new threats, fast growing of the value of intellectual property and increasing sophistication of methods and equipment that hackers use [23,32].

ACKNOWLEDGEMENTS

The research that led to the results shown here were obtained during the PhD stage # SD04/10/01.10.2016.

REFERENCES

[1] P. Marwedel, “Embedded system design: Embedded systems foundations of cyber-physical systems”, Springer Science & Business Media, 2011.
[2] K. Jiang, ”Security-Driven Design of Real-Time Embedded Systems”, LiU Tryck, 2015.
[3] K. C. Pabbuleti, “Performance Optimization of Public Key Cryptography on Embedded Platforms”, Faculty of the Virginia Polytechnic Institute and State University, 2014.
[4] T. Huffnire, B. Brotherton, T. Sherwood, R. Kastner, T. Levin, T. D. Nguyen, and Cynthia Irvine, ”Managing Security in FPGA-Based Embedded Systems”, IEEE Design & Test of Computers (Volume: 25, Issue: 6, Nov.-Dec. 2008).
[5] R. Dubey, “Introduction to Embedded System Design Using Field Programmable Gate Arrays”, Springer, 2009.
[6] E. A. Lee, ”Cyber physical systems: Design challenges”, In IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pages 363{369. IEEE, 2008.
[7] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, ”Cyber-physical systems: the next computing revolution”, In Design Automation Conference (DAC), pages 731{736. ACM, 2010.
[8] http://cyberphysicalsystems.org/

[9] http://kopustas.elen.ktu.lt/studentai/lib/exe/fetch.php?media=skaidres.pdf
[10] https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286
[11] T. Noergaard, "Embedded Systems Architecture – A Comprehensive Guide for Engineers and Programmers", Elsevier, 2005.
[12] M. Ciampa, "Security Awareness: Applying Practical Security in Your World", IEEE Design & Test of Computers 2010.
[13] P. Gregory, "CISSP guide to security essentials", Course Technology, Cengage Learning, Boston, 2010.
[14] M. Chowdhury, A. Apon, and K. Dey, "Data Analytics for Intelligent Transportation Systems", Elsevier, UK, 2017.
[15] http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/ch01s03s04.html
[16] H. Elmiligi, F. Gebali, M. Watheq El-Kharashi, "Multi-dimensional analysis of embedded systems security", Elsevier, 2017.
[17] J. Duncan Hagar, "Software Test Attacks to Break Mobile and Embedded Devices", CRC Press, 2014.
[18] R. R. Brooks, "Computer and Network Security – Navigating Shades of Gray", CRC Press, 2014.
[19] Lattice Semiconductor, "White Paper – FPGA Design Security Issues: Using the ispXPGA Family of FPGAs to Achieve High Design Security", December 2003.
[20] T. Wollinger, and C. Paar, "New Algorithms, Architectures, and Applications for Reconfigurable Computing", Kluwer, 2004.
[21] Saltzer, J. H. and Schroeder, M. D. (1975), "The protection of information in computer systems", Proceedings of the IEEE, Volume 63, No. 9, pp. 1278-1308.
[22] https://www.escrypt.com/fileadmin/escrypt/pdf/Hardware_Security_for_FPGAs_using_Cryptography_Microsemi_Huettemann.pdf
[23] S. M. Trimberger, Fellow IEEE, and J. J. Moore , "FGPA Security: Motivations, Features, and Applications", Proceedings of the IEEE (Vol. 102, No. 8, August 2014) .
[24] Microsemi, "Design Security in Nonvolatile Flash and Antifuse FPGAs", 2002.
[25] G. Gogniat, T. Wolf, W. Burleson. J. P. Diguet, L. Bossuet, and R. Vaslin, "Reconfigurable Hardware for High-Security/High-Performance Embedded Systems: The SAFES Perspective", IEEE Transactions on Very Large Scale Integration (VLSI) Systems (Volume: 16, Issue: 2, Feb. 2008).
[26] http://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA
[27] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, "Security Patterns – Integrating Security and Systems Engineering", John Wiley & Sons, Ltd, 2006.
[28] J. Biskup, "Security in Computing Systems – Challenges, Approaches and Solutions", Springer, 2009.
[29] Fernando Bîrleanu, Nicu Bizon, "Reconfigurable computing in hardware security – a brief review and application", Journal of Electrical Engineering, Electronics, Control and Computer Science (JEEECCS), volume 2, number 1, 2016, http://jeeeccs.net/index.php/journal/article/view/24
[30] https://www.design-reuse.com/articles/15105/a-security-tagging-scheme-for-asic-designs-and-intellectual-property-cores.html
[31] T. Wollinger, and C. Paar, "How Secure Are FPGAs in Cryptographic Applications", Springer, 2003.
[32] E. Pricop, and G. Stamatescu (Eds.), "Recent Advances in Systems Safety and Security", Springer, 2016.