A FPGA-Based Scheme for Protecting Weapon System Software Technology

Minhae Jang[1], Yeonseung Ryu[1(✉)], and Hyunkyoo Park[2]

[1] Department of Security Management Engineering, Myongji University, Yongin, Gyeonggi-Do, Korea
jully6363@naver.com, ysryu@mju.ac.kr
[2] Petabi Korea, Seoul, Korea
hyunkyupark@petabi.com

Abstract. Recently the Korean government has established a law to protect the defense industrial technology for the sake of national security and is trying to apply the anti-tamper methodology to weapon systems acquisition programs. The anti-tamper refers to the system engineering activity to protect critical information in systems from tampering and reverse engineering. Adversary's malicious tampering can weaken our military advantage, shorten the expected combat life of our system, and erode our defense industrial technological competitiveness. In this paper, we introduce the overview of the anti-tampering techniques and suggest a software protection technique using FPGA which can be efficiently used for weapon systems.

Keywords: Weapon systems · Anti-tamper · FPGA · Software protection

1 Introduction

The Korean government has invested about tens of billions of dollars every year in developing the weapon systems and technologies. According to the Korea National Defense Science and Technology Survey, the Korea's defense technology level is about 81% of the United States, which is the most advanced country, and the ranking is ranked ninth in the world in 2015. In recent years, the export of military goods such as battle ships, training fighter and self-propelled guns has increased and the number of exporting countries has also increased significantly. Besides, Korea imports a large number of weapons from many countries including the United States and is required to have a security system that protects the technologies implemented in imported weapon systems. If importing countries tamper with weapon systems to acquire military technologies, military advantage of exporting country can be weaken and the expected combat life of weapon systems can be shorten. Therefore, since 1999 the United States applies anti-tamper technology to ensure that importing countries do not reverse the weapon technology when exporting their weapon systems [1–4].

In the 1970s, the United States was friendly with Iran and sold the Iranians 80 F-14 Tomcats. The United States also provided flight training to Iranian pilots and support crew training in conjunction with the sale. Overall, an estimated $4 billion dollars of

O. Gervasi et al. (Eds.): ICCSA 2018, LNCS 10964, pp. 148–157, 2018.
https://doi.org/10.1007/978-3-319-95174-4_12

hardware was purchased by Iran in order to upgrade their Air Force. However, in the late 1970s, the Islamic Revolution occurred, which led to a hostile regime change. Now the United States made F-14s in hostile regime's possession. There was no anti-tamper technology in place to counter the exploitation of weapon system and critical technologies. The weapon systems and technologies can be exposed to the risk of compromise when they are exported, stolen, lost during combat, or damaged during routine missions. When weapon technologies are compromised, it can weaken military advantage, shorten the expected combat life of a system, and erode the industrial base's technological competitiveness in the international marketplace. In an effort to protect weapons and technologies from exploitation, the United States established a policy in 1999 to implement anti-tamper techniques, which include software and hardware protective devices, when technologies are determined to be critical and vulnerable to exploitation [1].

Recently Korean government established a law to protect the defense industrial technology for national security and is making an effort to apply the anti-tamper methodology to weapon systems acquisition program. Further Korean government will invest in the development of anti-tamper technology. The weapon system is becoming the embedded system consisting of hardware and software, and thus critical technology to be protected is implemented by hardware or software. Some anti-tamper technologies include software encryption, integrated circuit protection coating, and hardware access denied systems. Use of anti-tamper protection technologies must be refined according to the technologies to be protected. For example, the latest technology in critical characteristics usually requires more sophisticated anti-tamper applications.

In this paper, we study a software protection technique using Field Programmable Gate Array (FPGA) which can be efficiently used for weapon systems. A FPGA is a semiconductor device that includes a programmable logic element and a programmable internal line [5, 6]. The logic devices can be programmed by replicating the functions of basic logic gates, and more complex decoders or combinations of computational functions. When critical technology is implemented by software code, it is inherently easy to reverse engineer. Even though the software is decoded using source-level or binary-level obfuscation techniques, it can be cracked. However, it is difficult to reverse engineer if the critical technology is implemented as a hardware circuit such as FPGA. Using this property, we propose an anti-tamper technology to protect weapons and technologies from exploitation. The proposed method isolates the parts of the source code that contain the critical technology. The code is converted to HDL and then converted to bitstream of specific FPGA circuit. When the system runs, the machine code of the target system and the code of the FPGA are executed in an integrated way.

The rest of this paper is organized as follows. In Sect. 2, we introduce the anti-tamper techniques briefly. Section 3 describes the details of FPGA-based software protection scheme and related products. Finally, Sect. 4 describes conclusion and future work.

2 Background

2.1 Anti-tamper Techniques

In military domain, anti-tamper means the system engineering activity to prevent or delay the outflow of critical technologies from the weapon system [7]. Enemies want critical information from the weapon system in order to weaken military advantage, shorten the expected combat life of weapon system, and erode the industrial technological competitiveness. Weapon system consists of hardware and software components. Attacks can be performed on weapon system by means of two type attacks: passive or active. Passive attacks include a side channel analysis to determine the timing, dynamic power consumption, or secrets from an electromagnetic leak, as well as probe circuitry or imaging components. Active attacks include not only physical intrusion and hardware modification, but also failure induction through signal corruption, protocol attack, or malicious software.

Figure 1 shows that the anti-tamper techniques are composed of hardware and software techniques [8]. Each technique can be categorized by three techniques: prevention, detection and response.

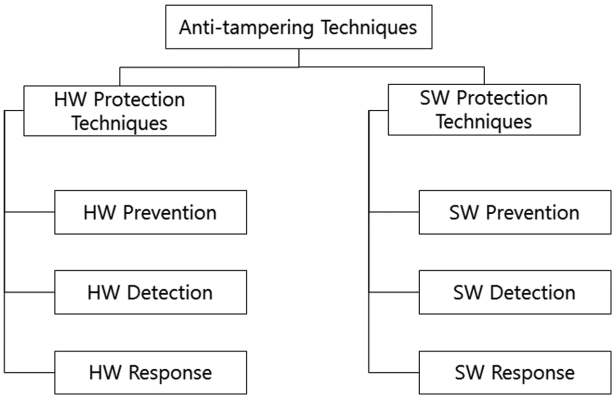

Fig. 1. Category of anti-tamper techniques

Prevention Techniques
The tamper protection techniques can prevent tampering attacks. However, when the threat of an attack is more severe than the protective strategy, it delays the collection of the least important information and makes it no longer usable. Examples of preventive safety measures for this are shielding, encapsulation, obfuscation, and encryption. The hardware protective coatings make it difficult to extract or dissect components without damaging the system. The software encryption scrambles software instructions to make them unintelligible without being reprocessed through a deciphering technique.

Detection Techniques
Tamper detection techniques may block tampering, either actively or passively, after detecting threats. Protection interlocks and low-power or non-power modulation sensors can detect and alert intrusions, and silicon's physically interruptible function (PUF) can uniquely identify a device for verification.

Response Techniques
Once tampering attack is detected, the weapon system can respond by destroying its own critical components to protect critical information. Disabling memory resources, disabling the communication interface, clearing out encryption keys, and causing explosive or high current corruption are examples of tamper response techniques (Table 1).

Table 1. Well-known 13 techniques for anti-tamper

1	Tamper indicating devices: seal and labels
2	Uniquely shaped screw heads
3	Locks for removable covers and doors
4	Coating: encapsulation materials
5	Mechanical mechanisms
6	Protective sensor mesh wall
7	Use of brittle components
8	Sensors
9	Zeroisations circuitry
10	Encryption wrappers
11	Code obfuscation
12	Guarding
13	Watermarking/fingerprinting

2.2 Weapon System Program Protection of the United States

In 1999, the United States DoD issued a policy memorandum for implementing anti-tamper protection in acquisition programs [1]. In 2000, DoD issued a policy memorandum stating that technologies should be routinely assessed during the acquisition process to determine if they are critical and if anti-tamper techniques are needed to protect these technologies [3]. In 2001, DoD designated the Air Force as the AntiTamper Executive Agent. The executive agent's office is responsible for implementing DOD's anti-tamper policy and managing anti-tamper technology development through the Air Force Research Laboratory. Acquisition program managers are responsible for ensuring anti-tamper protection is incorporated on any weapon system with critical technologies that need protection. Since it is not feasible to protect every technology, program managers are to conduct an assessment to determine if anti-tamper protection is needed.

The anti-tamper decision process is illustrated in Fig. 2 [9–14]. When assessing if anti-tamper protection is needed, program managers make several key decisions regarding the identification of critical technologies, assessment of threats and vulnerabilities, and determination of anti-tamper techniques or solutions. The process begins

with determining whether or not their system's critical program information includes any critical technologies. If it is determined that the system has no critical technologies, program managers are to document the decision and request concurrence from either the office within their component that is designated with anti-tamper responsibilities or the Anti-Tamper Executive Agent. For systems that are determined to have critical technologies, the next key steps are to identify potential threats and vulnerabilities and select anti-tamper techniques to protect those technologies. Techniques are ultimately verified and validated by a team composed of representatives from the DOD components. The program manager documents decisions in an annex of the program protection plan.

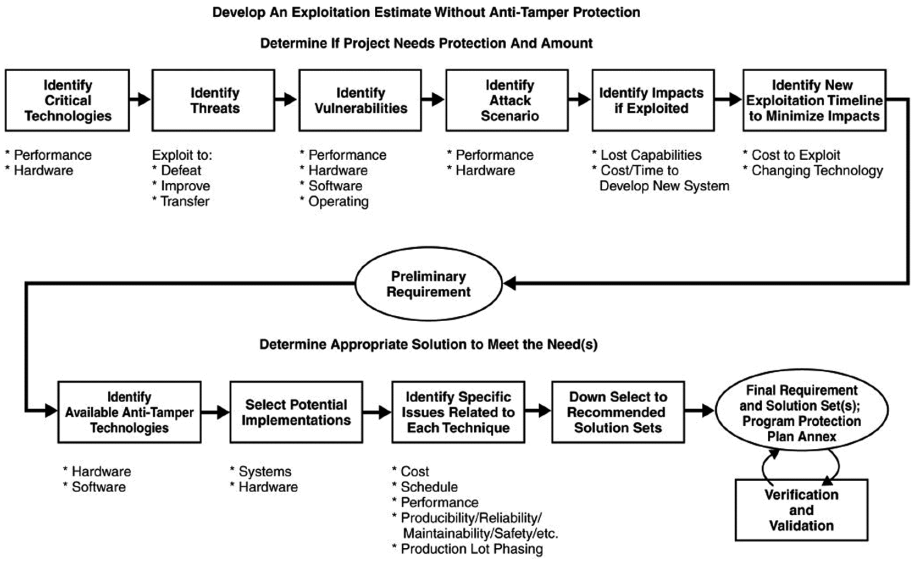

Fig. 2. US DOD's anti-tamper decision process [7]

DoD issued "Program Protection Plan Outline & Guidance" in 2011 that provides an outline, content, and formatting guidance for the Program Protection Plan (PPP) required by DoD Instruction 5000.02 and DoD Instruction 5200.39.

3 FPGA-Based Software Protection

3.1 FPGA Overview

An FPGA (Field Programmable Gate Array) is a logic device that includes a typical logic cell in a two-dimensional array and a programmable switch. Figure 3 shows the conceptual structure of the FPGA device [6]. Logical cells can be configured (programmable) to perform simple functions and programmable switches can be customized to provide interconnection between logical cells. A custom design can be implemented by specifying the functions of each logical cell and optionally setting up connections for

each programmable switch. Once designed and synthesized, a simple adapter cable can be used to change the desired logic cell to the FPGA device and obtain the custom circuitry. Because this process can be performed on site, it is known that the device enables field programming.

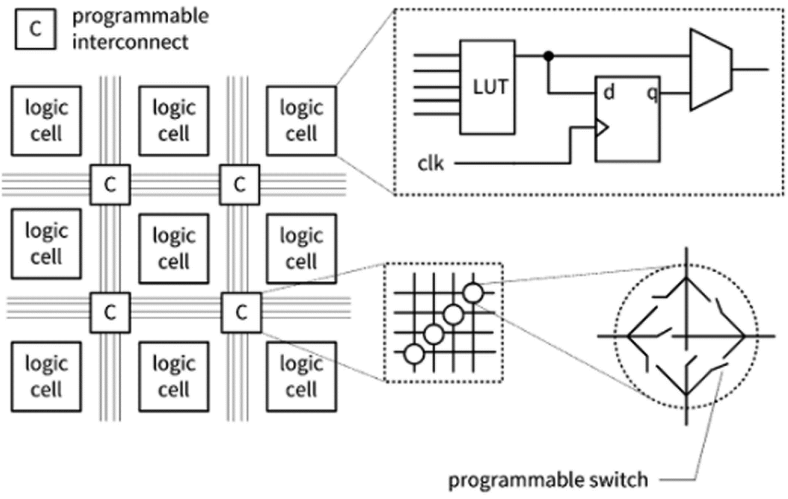

Fig. 3. Conceptual structure of a FPGA device [6]

LUT-based logic cells usually have a small configurable combination circuit with D FF (D type flip-flop). The most common method of implementing a configurable combination circuit is the LUT (lookup table). N Input LUTs can be considered small 2×1 memory. Using LUTs, n input combination functions can be implemented if memory contents are properly written. A schematic of the input LUT-based logic cell is shown in the upper right corner of Fig. 3. The LUT output can be directly used or stored on DFF. The latter can be used to implement sequential circuits.

In a macro cell, most FPGA devices contain a particular macro cell or macro block. They are designed and produced at the transistor level, which complements the common logic cells. Common macrophages include memory blocks, combination multiplier, clock management circuits, and I/O interface circuits. Advanced FPGA devices may also contain one or more pre-built processor cores.

3.2 Transformation Software Code into FPGA Code

When weapon technology is implemented by software, it is inherently easy to reverse engineering. However, it is difficult to reverse engineering if the critical technology is implemented as a hardware circuit such as FPGA. Recently some tools have been developed to translate the high-level source code into FPGA hardware circuit. Using these tools we can protect some software code by implementing as FPGA-level hardware circuit.

Figure 4 shows overall process of the proposed software protection techniques using FPGA. First of all, we identify and isolate the parts of the source code that contain the critical technology. The rest of the code is compiled in the normal way and made into the machine language of the target system. But, the code that needs to be protected is converted to HDL and then converted to bitstream to make the FPGA circuit. When the code is executed, the machine code of the target system and the code of the FPGA must be integrated and executed together. To do this, we need a well-defined call interface between the code in the target system and the code in the FPGA.

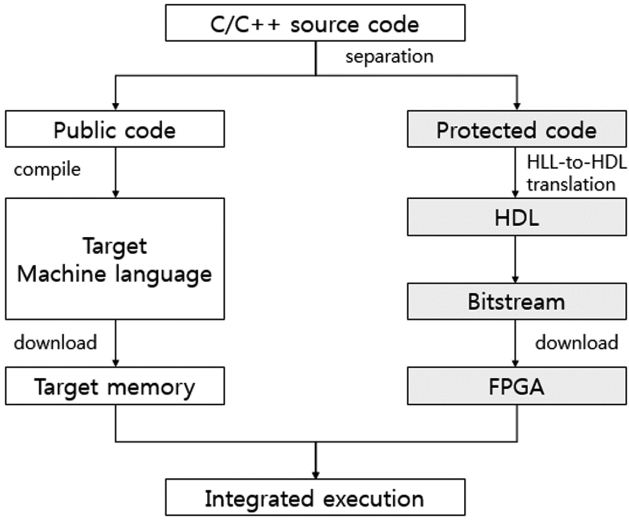

Fig. 4. Process for software code protection using FPGA

3.3 Existing Technologies

In this subsection, we introduce some products which can be used in our proposed software protection method.

Impulse C

Impulse C from Impulse Accelerate Technologies is a C-based development system for programmable hardware targets, including mixed processors and FPGA platforms [15]. This technique is based on the impulse C compiler, the related tools and the Impulse application programmer interface (API). Impulse C can process the C code block with either a Verilog Hardware Description Language (VHDL) or a Verilog Hardware Description. Impulse technique provides several FPGA platform packages which allow us to simplify C-to-hardware compilation for specific FPGA-based platform. The Impulse C compiler and optimizer supports automatic scheduling of C doors for loop growth and for the automation such as parallelism, loop pipe lining and frozen rolling, and for semi-automatic optimization. The interactive tools that come with the compilers

allow the designer to continually analyze and experiment with alternative hardware pipeline strategies.

Mitrion-C

Mitrion-C from Mitrionics provides software programmability for FPGA [16]. Further, the virtual processor is introduced as a large parallel processor for the FPGA, which runs software written in the Mitrion-C programming language. The processor architecture follows the cluster model, which places all processing nodes within the FPGA. The Mitrion-C compiler and processor unit uses the Mitrion-C source code to create processing nodes and ad-hoc network-on-a-chip.

Unlike standard C, Mitrion-C provides a fully parallel programming language to complement the fine parallel processing of the processor. In Standard C, the programmers explain the order in which programs run. This order forces a specific sequential order to run, and is therefore not suitable for parallel running. The processing model of Mitrion-C is based on data dependency and is much more appropriate for parallel processing (Fig. 5).

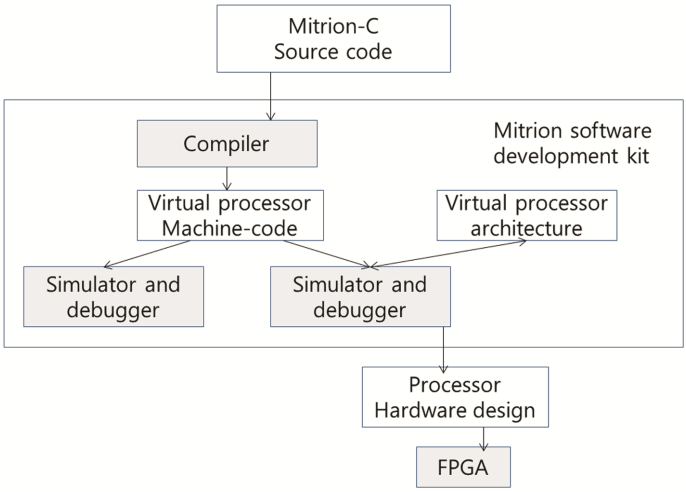

Fig. 5. Mitrion-C architecture

Carte

Carte from SRC Computers is development tools which support traditional programming development environment [17]. It allow us to write code in advanced programming languages such as C and Fortran until the implementation of the appropriate program runs correctly in a microprocessor environment. And then we can recompile the source code into the final FPGA chip configuration bitstream. Carte also provides the parallelism capability by pipelining loops, scheduling memory references, and supporting parallel code blocks and streams.

Handel-C
Handel-C from Celoxica also consolidates high-level user codes into the FPGA [18]. Handel-C was originally developed by Oxford University Computing Laboratory. By replacing the algorithm loop in the original Fortran, C, or C++ source code with API calls, the Handel-C user can derive the source code to be compiled with the FPGA. The FPGA C compiler has a runtime part to establish interactions with the hardware environment. Programmers can define the parallel process using extensions that instruct the compiler to write parallel hardware. The compiler converts the input Handel-C code into an abstract syntax tree and then optimizes the high-level netlist and then compiles it into the FPGA bitstream. Handel-C also supports a development tool that includes a GUI environment, code editing, and source-level debugging.

Trident
Trident also synthesizes circuits from a high-level language source code [19]. Unlike other products, it provides an open framework for mapping the C program's floating-point operations to hardware floating-point modules and automatically allocating floating-point arrays to off-chip memory banks. Users are free to select floating-point operators from a variety of standard libraries. It also supports various hardware platforms by defining new interface description files and producing the code to tie the design to the description interface. Unlike other products, compiler's open source code is available (http://trident.sf.net).

4 Conclusion

As defense technology level of Korea is increasing, Korean government has plan to invest in the development of anti-tamper policy and technology to counter the exploitation of weapon system and critical technologies. The weapon system is becoming the embedded system consisting of hardware and software, and thus most critical technologies to be protected are implemented by hardware or software.

In this paper, we introduce the overview of the anti-tamper techniques and propose a software protection technique using FPGA which can be efficiently used for weapon systems. Since it is difficult to reverse engineering the software code which is implemented as a hardware circuit such as FPGA, we can efficiently protect the critical technology of weapon systems. We found that there are already some products similar to the proposed techniques. For the future work, we will implement the proposed technique and validate its effectiveness.

References

1. US DoD Memorandum: Implementation of Anti-tamper (AT) Techniques in Acquisition Programs (1999)
2. Huber, I.I., Arthur, F., Scott, J.M.: The role and nature of anti-tamper techniques in U.S. defense acquisition (1999)
3. US USD (A&T) Memorandum: Guidelines for Implementation of Anti-tamper Techniques in Weapon System Acquisition Programs (2000)

4. US USD (A&T) Memorandum: Implementing Anti-Tamper (2001)
5. Pong, C.: FPGA Prototyping by VHDL Examples: Xilinx Microblaze MCS SoC, 2nd edn. Wiley, Hoboken (2017)
6. Underwood, K.: FPGAs vs. CPUs: trends in peak floating-point performance. In: 12th ACM International Symposium on Field-Programmable Gate Arrays (FPGA 2004), pp. 171–180. ACM Press (2004)
7. US General Accounting Office Report GAO-04-302: Defense acquisitions: DoD needs to better support program manager's implementation of anti-tamper protection (2004)
8. Abaco Systems White Paper: Anti-tamper technology: safeguarding today's COTS platforms (2016)
9. US DoD Directive 5000.01: The Defense Acquisition System (2003)
10. US DoD Instruction 5000.02: Operation of the Defense Acquisition System (2015)
11. US DoD Instruction 5200.39: Critical Program Information (CPI) Identification and Protection Within Research, Development, Test, and Evaluation (RDT&E) (2015)
12. US DoD Directive 5200.47E: Anti-Tamper (2015)
13. US DoD DASD: Program protection plan outline & guidance, Version 1.0 (2011)
14. Raymond, S.: Identification and protection of critical program information (CPI). In: 18th Annual NDIA Systems Engineering Conference (2016)
15. Impulse C Homepage. http://www.Impulsec.com
16. Mitrion-C Homepage. http://www.mitrionics.com
17. CARTE Homepage. http://www.src-labs.com
18. Handel-C Homepage. http://www.mentor.com
19. Justin, T., Maya, G., Kristopher, P.: Trident: from high-level language to hardware circuitry. IEEE Comput. **40**(3), 28–37 (2007)