Securing FPGA-Based Obsolete Component Replacement for Legacy Systems

Zhiming Zhang[1], Laurent Njilla[2], Charles Kamhoua[3], Kevin Kwiat[2], and Qiaoyan Yu[1]

[1] Department of Electrical and Computer Engineering, University of New Hampshire, Durham, NH 03824, USA

[2] Cyber Assurance Branch, Air Force Research Laboratory, Rome, NY 13441, USA

[3] Army Research Laboratory, Adelphi, MD 20783, USA

Abstract—**Component-aging is unavoidable in legacy systems. Although re-designing the system typically results in a high cost, the need to replace aged components for legacy systems is an urgent priority. Unfortunately, the aged components are likely to be obsolete and not available on the current market. Obsolete component replacement with field-programmable gate array (FPGA) devices is emerging as a feasible option to extend the lifetime of legacy systems. While replacing the aged component, we traditionally only focus on matching the functionality and neglect the potential security threats from FPGA replacement. However, recent literature demonstrates that FPGA devices may contain hardware Trojans, which are induced during FPGA device fabrication or bitstream generation time. To prevent the Trojans on FPGA from receiving external inputs or leaking sensitive information, we propose a Runtime Pin Grounding (RPG) scheme to ground the unused pins and check the pin status at every clock cycle. Furthermore, we exploit the principle of moving target defense (MTD) and propose a hardware MTD (HMTD) method. In our method, the aged obsolete unit is replicated to multiple copies in the FPGA device, and two of the replicas are randomly selected for output comparison and thus Trojan detection. We successfully implemented the proposed RPG and HMTD methods on a Nexys-3 FPGA board. Our case study shows that the proposed RPG scheme increases the FPGA utilization rate by less than 0.1%. On average, our HMTD method reduces the hardware Trojan bypass rate by 61% over the existing method.**

Index Terms—**FPGA security, FPGA Trojan, Hardware security, hardware Trojan, legacy system, obsolete component replacement, pin grounding.**

I. Introduction

Due to high reliability requirements and cost on design and installation, the lifetime of electronics systems in military applications is expected to be longer than in industrial or civil use. In a legacy system, some electronic components may experience aging earlier than others. Unfortunately, the aged components may no longer be manufactured or available on the market. A straightforward solution is to re-design the entire system, but the total cost for re-designing, testing, and installation could be 10 times that of other alternatives, such as component replacement [1]. An obsolete component can be substituted by an equivalent device from gray market, application-specific integrated circuit (ASIC), field-programmable gate array (FPGA) [2], or uncommitted logic array (ULA).

Traditionally, functionality matching is the primary focus when we replace the aged module with a functional equivalent. Little or no attention is paid to the security threats originated from the component replacement. Unfortunately, the trustworthiness of the FPGA supply chain has become a serious concern now, so it is imperative to address those security threats, in particular, from untrusted FPGA manufacturers and computer-aided design (CAD) tools associated with FPGA deployment. The majority of existing efforts for FPGA security [3, 4] aim at improving the system resilience against intellectual property (IP) piracy and reverse engineering attacks. In this work, we assume that the FPGA deployment team and the legacy system user are trusted but the FPGA manufacturer and the FPGA CAD tools are not trusted.

The remainder of this work is organized as follows. Section II summarizes the related work and our main contributions. In Section III, the attack model and the legacy system model interested in this work are introduced. In Section IV, we present our Runtime Pin Grounding (RPG) scheme and Hardware Moving Target Defense (HMTD) method as an integrated countermeasure to thwart security threats from FPGA and its CAD tool. Assessment on security metric, performance, and overhead are provided in Section V. We conclude this work in Section VI.

II. Related Work and Our Contributions

A. Related Work

1) Using FPGA to Replace Obsolete Components in Legacy Systems: Advanced Microcircuit Emulation (AME) [5] has provided obsolescence solutions for over 25 years. Those solutions are at the digital component level (e.g., logic devices, ASICs, FPGAs, static memory devices, microprocessors, and micro-controllers). The Defense Logistics Agency (DLA) supports the development of Generalized Emulation of Microcircuits (GEM) technology [6] to extend the lifetime of legacy systems. The DLA assumes that AME is completely loyal and thus trusted. Unfortunately, it is difficult to hold that assumption now as the number of trusted foundries keeps decreasing due to the high cost of maintenance and upgrading to new laboratory lines.

DISTRIBUTION A. Approved for public release: distribution unlimited. Case Number: 88ABW-2017-5831. Dated 17 Nov 2017

2) Security Threats in FPGA Devices: A system that incorporates an FPGA device is vulnerable to various security threats [7, 8]. The attack surface varies from FPGA fabric [9], soft IP cores, CAD tools [10], and the download channel (in-field or wireless) for FPGA configuration bitstream [11]. Thompson [12] provides an insightful example of how the tool from an untrusted source can tamper with the original design, and he also emphasizes that some bugs in microcode will be almost impossible to detect. The purpose of attacks on FPGA-based systems include IP piracy, IP cloning, denial of service, and secret key leaking [4, 13]. Detailed examples of malicious FPGA attacks are available in [14, 10].

3) Countermeasures against FPGA Security Threats: To detect the hardware Trojans carried in the FPGA configuration bitstream, Chakraborty et al. [10] suggest the following: grounding the unused I/O pins; monitoring the temperature of the FPGA device; filling up the unused resources of the FPGA; or scrambling the bitstream file. Bloom et al. [13] propose to morph on-chip resources for moving target defense (MTD) against fabrication-time Trojans. Their method heavily utilizes encryption on the FPGA configuration for initialization boot and hardware description of functional modules. Moreover, process memory, L1 cache, and L2 cache are encrypted separately using multi-layer encryption. Although the alteration of two instances for the same CPU implementation can thwart random hardware Trojans, the multi-layer encryption is too costly for many real-time systems. The ideas proposed in [10] and [13] remain at the conceptual level, and no practical experiments have been conducted to demonstrate the method's feasibility.

To protect the FPGA configuration bitstream against piracy, reverse engineering, and tampering, Karam et al. [15] obfuscate the FPGA bitstream by inserting additional functions in the look-up tables (LUTs) that are configured for the true functionality of the design. Jyothi et al. [16] utilize ring-oscillator arrays to measure process variation among FPGA slices, which may be modified by the untrusted FPGA manufacturer. Then, the FPGA region where the process variation is below the acceptable threshold is identified as a trust zone. The authors place the hardware design only in the trusted FPGA zones. This method assumes that the malicious FPGA slices lead to significant changes on delay, and the FPGA CAD tool is trusted. Mal-Sarkar et al. [14] propose an adapted triple modular redundancy (ATMR) technique to detect the hardware Trojan inserted in one of the design replicas. To reduce the overhead on power consumption, the third replica is activated once the output mismatch is detected from the other two replicas. The limitation of this method is that the three copies of the design module are allocated by the FPGA CAD tool in a stationary manner. Because the untrusted CAD tool has the prior knowledge of the place and route rules, theoretically, the tool can insert the same Trojan in the two replicas of the design. Thus, the ATMR method may not detect the Trojan.

The aforementioned methods assume that the FPGA CAD tool is trusted. These methods do not consider the scenarios in which hardware Trojans in the bitstream configuration can be inserted during the place and route stage. Another challenge is how to prevent the countermeasure from being removed/muted by the untrusted FPGA CAD tool.

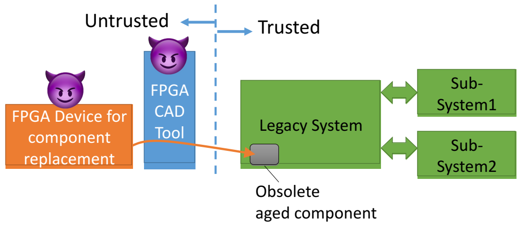


Fig. 1: System model of FPGA replacement for a legacy system.

B. Contributions of This Work

To address the security concerns discussed in the previous section, we propose a framework to detect the hardware Trojan inserted by an untrusted FPGA manufacturer or CAD tools. More specifically, our main contributions are as follows.

(1) To the best of our knowledge, this is the first work that investigates the countermeasure against the security threats that occur during the FPGA deployment for legacy systems. The primary goal of this work is to address the security attacks from the untrusted FPGA vendor and the CAD tools for FPGA configuration, rather than the IP piracy and side-channel attacks on FPGAs.

(2) We propose a RPG scheme. Compared with the conceptual proposal in [10], we implemented the pin grounding concept on a Nexys-3 Spartan-6 FPGA board that successfully prevents the communication between the external environment and the FPGA device. Moreover, our scheme additionally performs runtime checking to examine whether all user-unused I/O pins are truly grounded at every clock cycle, thus thwarting the countermeasure mutation by the FPGA CAD tool.

(3) We propose a HMTD method. In our method, the hardware description of the aged functional module in the legacy system is replicated multiple times. Two of the replicas are randomly selected by an on-chip random number generator to examine the consistency between the two groups of outputs. Furthermore, instead of leaving the FPGA CAD tool to place and route the replacement module with default settings, we propose to explicitly specify the slice physical distance between the replicas in a FPGA user constraint file. Our method is able to thwart the stationary hardware Trojan insertion by the CAD tool.

III. Preliminaries

A. Model of Legacy System in This Work

Using an FPGA to replace the aged component in a legacy system is a promising alternative to searching for a direct replacement of the original chip [2, 17]. Following this path, we further address the security threats in the process of FPGA replacement. Figure 1 shows the model of our legacy system. The FPGA device and CAD tool are not trusted, but the engineering team for the aged module replacement is trusted.

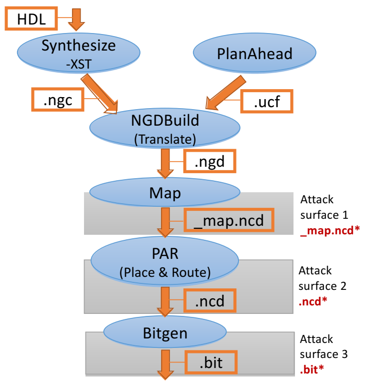


Fig. 2: Attack surfaces of interest in this work. The rectangles represent the output file from each step. The file with the symbol of * is an output file modified by the untrusted FPGA CAD tool.

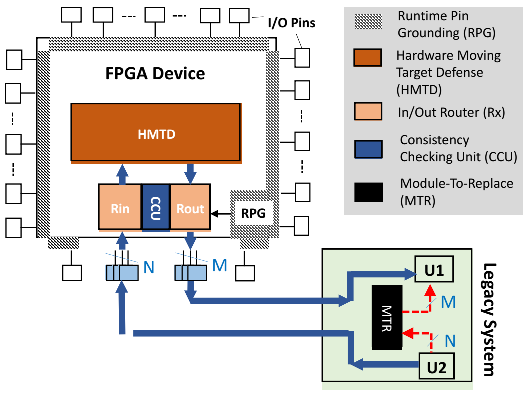


Fig. 3: Overview of proposed countermeasure to secure the FPGA replacement for a legacy system. To replace the aged module (MTR), the proposed method connects a group of FPGA modules (HMTD+Rin+Rout+CCU) to the original modules U1 and U2 in the legacy system.

B. Attack Model

Literature [10] highlights that the FPGA bitstream may be compromised by an adversary. The attacks of interest in this work are hardware Trojans on FPGA fabric that are permanently configured by untrusted FPGA manufacturers or FPGA bitstream tampering through CAD tools. For the latter attack, there are three attack surfaces, which are shown in Fig. 2. Here, we use a Xilinx FPGA implementation flow [18] as an example to explain the procedure of tampering with FPGA configuration files.

The synthesis tool **XST** converts the functional module designed in a hardware description language (HDL) to a logic gate netlist (*.ngc*). The tool **PlanAhead** facilitates functional module designers to generate a user constraint file (*.ucf*). The **NGDBuild** program translates the netlist and user constraint files to a native generic database file (*.ngd*), which describes the logic for the design and the constraints on timing, as well as the preferred slice location. Based on the *.ngd* file, the mapper program **Map** produces a native circuit description file (*_map.ncd*), which maps the input and output pins to the specific FPGA device. Next, the place & route program **PAR** continues the physical design by using LUTs, flip-flops, and SRAM blocks, and the *_map.ncd* file is updated to a new *.ncd* file. Finally, the **Bitgen** program converts the *_map.ncd* file to a FPGA bitstream, which is ready to be downloaded to the target FPGA device. There are three attack surfaces that can be exploited by the untrusted CAD tools.

(1) **Attack Surface on Map:** In the step of mapping, an attacker can introduce additional I/O pins, exchange the existing I/O pin connection, and modify the slew rate and the voltage level of I/O pins. Because the tampered *_map.ncd** is not readable (unless the FPGA CAD tool vendor provides us with program **ncd2xdl** to read back the native circuit description file), it is not easy to notice the modification performed by the malicious CAD program.

(2) **Attack Surface on PAR:** More hardware tampering can be done in this stage than in the mapping step because the original use of all the LUTs, flip-flops, SRAM blocks, and interconnects can be modified by the tool surreptitiously. The tampered *.ncd** file is not readable due to the obfuscation performed by the CAD tool to protect intellectual property.

(3) **Attack Surface on Bitgen:** To prevent reverse engineering attacks, the FPGA CAD tool writes the bitstream in binary. Except for the Virtex-II FPGA family, no public documents are available to guide FPGA users to convert the bitstream back to a readable FPGA configuration information. Although we can use a Hex editor to read the bitstream, tracking the modification on the bitstream file is almost impossible.

IV. Proposed Method

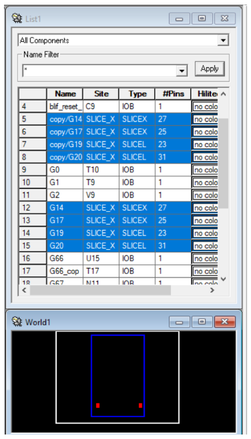
We propose a countermeasure that is composed of two parts: (1) RPG and (2) HMTD. The RPG scheme is to terminate the hardware Trojans that communicate with the external environment through unused I/O pins on the FPGA device. The HMTD method prevents the Trojan horses induced by the malicious FPGA CAD tools from interfering with the FPGA replacement in legacy systems. Figure 3 depicts the overview of the proposed countermeasure against hardware Trojans on the FPGA device.

A. Proposed Runtime Pin Grounding

Inspired by the idea proposed in [10], we apply the pin grounding scheme to the unused FPGA I/O pins by using a user constraint file. In this work, we continue to use the Nexys-3 FPGA board to introduce the procedure of our RPG scheme. This scheme is implemented on the top level of the hardware description module as shown in the black shadowed area of Fig. 3.

First, we assign every unused pin a net name in the top level of the hardware design file. Then, each NET name is linked with an unused I/O pin in the user constraint file by using the command (1).

NET "net_name" LOC = pin_name **(1)**



List1

All Components

Name Filter

*

Apply

	Name	Site	Type	#Pins	Hilite
4	blif_reset_	C9	IOB	1	no colo
5	copy/G14	SLICE_X	SLICEX	27	no colo
6	copy/G17	SLICE_X	SLICEX	25	no colo
7	copy/G19	SLICE_X	SLICEL	23	no colo
8	copy/G20	SLICE_X	SLICEL	31	no colo
9	G0	T10	IOB	1	no colo
10	G1	T9	IOB	1	no colo
11	G2	V9	IOB	1	no colo
12	G14	SLICE_X	SLICEX	27	no colo
13	G17	SLICE_X	SLICEX	25	no colo
14	G19	SLICE_X	SLICEL	23	no colo
15	G20	SLICE_X	SLICEL	31	no colo
16	G66	U15	IOB	1	no colo
17	G66_cop	T17	IOB	1	no colo
18	G67	N11	IOB	1	no colo

World1

Fig. 4: Detailed slice assignment shown in the FPGA Editor. The two red dots repre the locations for the two replicas of MTR equivalence that are specified in our method through FPGA Editor.

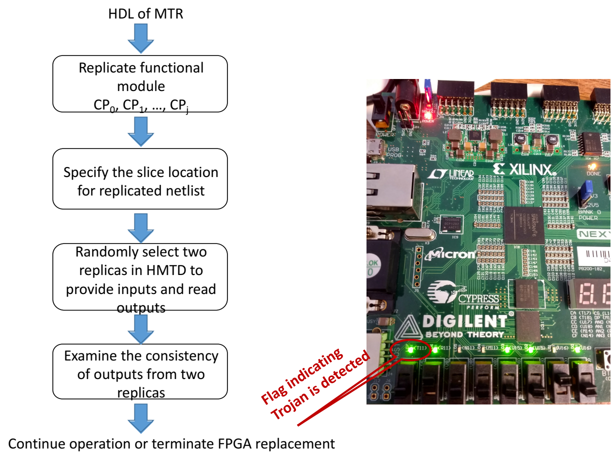

HDL of MTR
Replicate functional module CP_0, CP_1, …, CP_j
Specify the slice location for replicated netlist
Randomly select two replicas in HMTD to provide inputs and read outputs
Examine the consistency of outputs from two replicas
Continue operation or terminate FPGA replacement
Flag indicating Trojan is detected
XILINX
Micron
CYPRESS
DIGILENT BEYOND THEORY

Fig. 5: Flowchart of proposed hardware moving defense method.

After that, we proceed to ground those I/O pins through the command (2).

$$\textbf{NET "net\_name" PULLDOWN} \tag{2}$$

Even if we have grounded all of the unused pins through the user constraint file, the malicious FPGA CAD tool can alter the user-specified pin configuration by modifying the native circuit description (.ncd) file. This phenomenon has been observed in our FPGA deployment environment Xilinx ISE 14.1 [18] when we manually ground the pin reserved for the power supply. Because the .ncd file is not readable, the hardware Trojans placed by the CAD tool are stealthy. To thwart the unrevealed modification from the CAD tool, we enhance our pin grounding scheme by adding a runtime detection circuit. Since we have assigned a net name for each unused I/O pin, we can simply use the logic of *NOR* to examine the grounding status of those unused pins.

B. Proposed Hardware Moving Target Defense

To prevent the malicious CAD tool from successfully sabotaging the original FPGA configuration, we use the principle of MTD to develop our HMTD method. We assume that the functionality of the module-to-replace (MTR) in the legacy system is known by the FPGA deployment team, who is trusted. Our HMTD method replicates the MTR into multiple copies $\mathrm{CP}_0, \mathrm{CP}_1, \cdots, \mathrm{CP}_j$. We use the "RLOC" command to specify the relative physical distance between two replicas in the user constraint file. For instance, we can assign CP_0 and CP_1 to the two corners of the FPGA device by setting **RLOC = X36Y61** and **RLOC = X1Y60**, respectively. Alternatively, we can utilize the FPGA Editor tool to perform the similar operation. Figure 4 shows that two replicas of MTR equivalent are successfully placed to two FPGA corners by our method.

In the next step, we add a low-cost, random number generator in the *Rin* unit to select two replicas of the function module to feed the N-bit inputs from *U2* in the system shown in Fig. 3. This setting is essentially a power-gating technique to reduce the power consumption. For sequential circuits, state restore will be required to use the input gating technique. Note that the random number generator is implemented on the FPGA, and thus the random selection is performed at runtime. The random number generator also controls the *Rout* unit to choose which two replicas for the Trojan detection in the consistency checking unit (CCU). Once the output inconsistency is found, the M-bit output pins are grounded immediately and the flag for the Trojan detection is turned on. The flowchart of our HMTD method is summarized in Fig. 5.

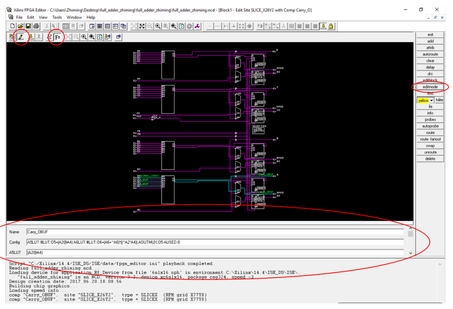
V. Experimental Results

A. Experimental Setup

The following experiments were performed on the Nexys-3 Board, which contains a Xilinx Spartan-6 XC6SLX16 CSG324C FPGA. This FPGA device includes 324 I/O pins (232 of which are user I/O pins) and 2,278 slices, each containing four 6-input LUTs and eight flip-flops. We used the Xilinx ISE 14.1 version to synthesize, place and route the Verilog HDL design files and generate bitstreams. The hardware overhead assessment was based on the ISCAS'85 benchmark circuits. We inserted the hardware Trojans on the FPGA device through two techniques: one is through the FPGA Editor, and the other is via editing the native circuit description file. Both of these techniques do not require changing the Verilog HDL file of the function module. The slice assignment shown in the FPGA Editor (see Fig. 6(a)) demonstrates that the FPGA CAD tool can successfully alter the configuration of one unused FPGA slice without disturbing the logic netlist. Although a native circuit description (.ncd) file is not readable, we can use an xdl program to translate that .ncd file to a readable file. Figure 6(b) also demonstrates that the hardware Trojan has been successfully placed in an un-occupied slice. We compared the Trojan resistance strength of our method and the ATMR approach [14] in the following subsection.

B. Hardware Trojan Bypass Rate

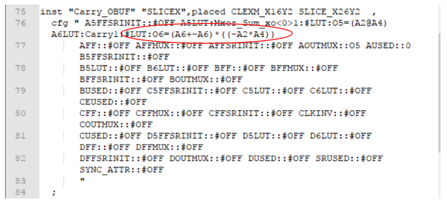
We validated the proposed HTMD method on the Nexys-3 board. Whenever a Trojan is detected, the flag light on the board will be turned on as shown in the right side of Fig. 5. To extensively assess the success rate of different FPGA hardware Trojan detection methods, we modeled the Trojan insertion and



```
Name    Carry_OBUF
Config  A5LUT:#LUT:O5=(A2@A4) A6LUT:#LUT:O6=(A6+~A6)*((~A2*A4)) AOUTMUX:O5 AUSED:0
A5LUT   (A2@A4)
```

```
Script "C:/Xilinx/14.4/ISE_DS/ISE/data/fpga_editor.ini" playback completed.
Reading full_adder_zhiming.ncd...
Loading device for application Rf_Device from file '6slx16.nph' in environment C:\Xilinx\14.4\ISE_DS\ISE\.
   "full_adder_zhiming" is an NCD, version 3.2, device xc6slx16, package csg324, speed -3
Design creation date: 2017.06.20.18.08.56
Building chip graphics...
Loading speed info...
comp "Carry_OBUF",  site "SLICE_X26Y2",  type = SLICEX  (RPM grid X77Y8)
comp "Carry_OBUF",  site "SLICE_X26Y2",  type = SLICEX  (RPM grid X77Y8)
```

(a)



```
inst "Carry_OBUF" "SLICEX",placed CLEXM_X16Y2 SLICE_X26Y2  ,
  cfg " A5FFSRINIT::#OFF A5LUT:Mxor_Sum_xo<0>1:#LUT:O5=(A2@A4)
  A6LUT:Carry1:#LUT:O6=(A6+~A6)*((~A2*A4))
        AFF::#OFF AFFMUX::#OFF AFFSRINIT::#OFF AOUTMUX::O5 AUSED::0
        B5FFSRINIT::#OFF
        B5LUT::#OFF B6LUT::#OFF BFF::#OFF BFFMUX::#OFF
        BFFSRINIT::#OFF BOUTMUX::#OFF
        BUSED::#OFF C5FFSRINIT::#OFF C5LUT::#OFF C6LUT::#OFF
        CEUSED::#OFF
        CFF::#OFF CFFMUX::#OFF CFFSRINIT::#OFF CLKINV::#OFF
        COUTMUX::#OFF
        CUSED::#OFF D5FFSRINIT::#OFF D5LUT::#OFF D6LUT::#OFF
        DFF::#OFF DFFMUX::#OFF
        DFFSRINIT::#OFF DOUTMUX::#OFF DUSED::#OFF SRUSED::#OFF
        SYNC_ATTR::#OFF
        "
  ;
```

(b)

Fig. 6: FPGA hardware Trojans inserted without disturbing the hardware description file. The modified slice can be observed from (a) the FPGA Editor and (b) the .xdl file converted from a .ncd file.

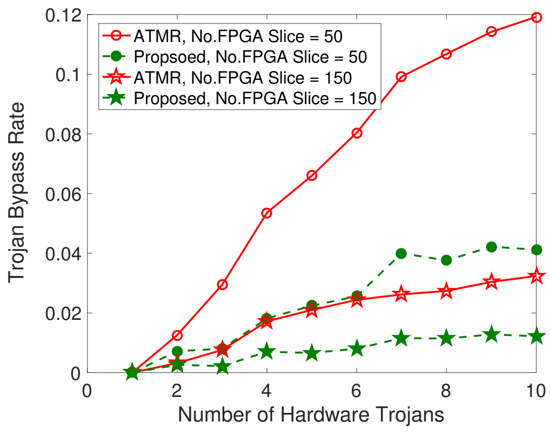


Fig. 7: Hardware Trojan bypass rate versus number of hardware Trojans inserted in the FPGA device.

the detection methods in MATLAB. We randomly selected 10 slices for hardware Trojan insertion. This operation was conducted after the slices for the original design module were configured. The hardware Trojan bypass rate is defined as the number of incorrect outputs, due to Trojans, over the number of test cases.

The impact of the number of the hardware Trojans on the Trojan bypass rate is shown in Fig. 7. As can be seen, for the range of 1 to 10 Trojans, the Trojan bypass rate almost monotonically increases with the number of injected hardware Trojans. As the number of Trojans increases, the probability for multiple replicas of the functional module simultaneously containing Trojans increases. Hence, comparison of the two copies' outputs gradually loses the Trojan detection capability, and thus the Trojan bypass rate increases.

We vary the number of FPGA slices to examine the impact of the FPGA size on the hardware Trojan bypass rate. In

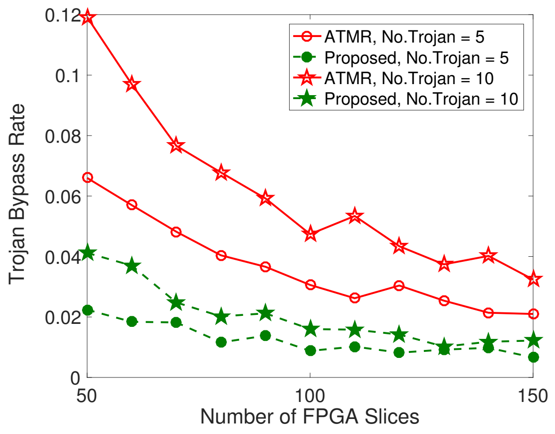


Fig. 8: Impact of the number of FPGA slices on hardware Trojan bypass rate.

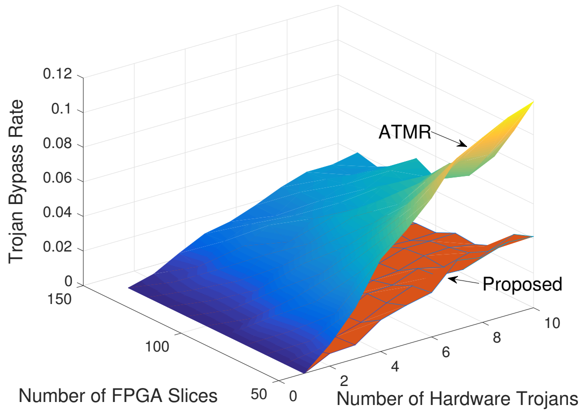


Fig. 9: Three-dimensional plot for the dependent factors for hardware Trojan bypass rate.

Fig. 8, we can observe that the Trojan bypass rate for a larger FPGA is lower than that for a smaller one. This is because the number of Trojans placed in the FPGA device is fixed per each FPGA size. The chance for a Trojan slice colliding with a design slice is higher in a smaller FPGA than in a larger one. The ATMR method compares the two of three copies for the design module using a fixed algorithm, which can only resist truly random Trojans. In contrast, our method randomly selects any two replicas for Trojan detection at runtime; moreover, our method is capable of assigning each replica to a specific location. Thus, the design location specified by our method is not predictable to the CAD tool. Hence, the randomness and unpredictability of our method strengthens the FPGA replacement resistance against the security threats from the untrusted FPGA manufacturer and CAD tool vendor.

To have a comprehensive view, we plot the Trojan bypass rate versus the FPGA size and the number of Trojans in Fig. 9. As shown, the 3D mesh sheet of our method is lower than that of the ATMR method [14]. On average, our method reduced the Trojan bypass rate by 61%.

C. Overhead on Hardware Cost and Performance

We applied the RPG scheme to the ISCAS'85 benchmark circuits. Because more unused I/O pins lead to more overhead

TABLE I: FPGA Overhead of Proposed Runtime Pin Grounding

Overhead\Circuits	s298	s344	s444	s526	s1488
Increased No. LUTs	40	30	38	40	39
Increased No. Slices	12	6	11	12	16

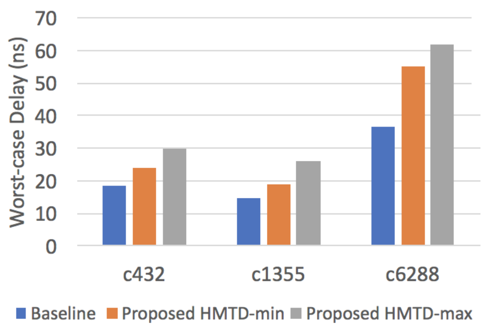


Fig. 10: The delay overhead of proposed HMTD applied on benchmark circuits.

for pin grounding, we chose the benchmark circuits with a small number of inputs/outputs. As shown in Table I, the number of utilized LUTs go as high as 40, whereas the number of occupied slices go up to 16. These hardware implementations consume 0.044% more LUTs and 0.07% more slices, respectively.

After the FPGA place and route step, we measured the worst-case delay of the c432, c1355, and c6288 benchmark circuits with and without the proposed HMTD method. As we mentioned in Section IV.B, we manually added a physical distance between the replicas of the functional module to thwart the Trojan attack from the CAD tool. The induced separation may result in longer routing interconnects than the baseline. Depending how the replicas are assigned to the FPGA slices and the amount of distance is added between two copies, the delay overhead of our method varies. We recorded the minimum and maximum delay overhead as observed in our case study. As shown in Fig. 10, the average minimum (maximum) delay overhead of HMTD is 37% (70%).

VI. Conclusions

Traditionally, FPGA replacement for legacy systems was mainly focusing on matching functionality. As the trustworthiness of FPGA devices and the CAD tools become more questionable, it compels us to address the potential security threats from the malicious FPGA manufacturer and CAD tool vendor. Existing studies provide some ideas on how to tackle the security threats on FPGA, but they do not implement their proposed methods nor perform extensive assessments. This work validates the feasibility of pin grounding and further extends it to a runtime scheme. Our method not only grounds the FPGA I/O pins at the configuration time, but it also checks the pin status during the operation time. To thwart the FPGA Trojan configured by an untrusted FPGA vendor or malicious CAD tools, we propose an HMTD method. In addition to replicating the functional module to replace the obsolete component for the legacy system, our method further specifies the relative physical distance between two replicas in the FPGA user constraint file and performs output comparison from two randomly selected replicas. Because the CAD tool cannot foresee the user constraint file, our method can effectively detect FPGA Trojans inserted by the CAD tool through implicit settings at the development time of that tool. Our experimental results show that the proposed HMTD method reduces the hardware Trojan bypass rate by 61% (on average) lower than the existing ATMR method. Our RPG scheme increases the FPGA utilization rate below 0.1%. In the future work, we will improve our method to further reduce the delay overhead.

References

[1] "Managing time and costs in legacy component upgrades." http://www.militaryaerospace.com/articles/print/volume-14/issue-12/departments/viewpoint/managing-time-and-costs-in-legacy-component-upgrades.html.

[2] D. Hallmans, K. Sandstrm, T. Nolte, and S. Larsson, "A method and industrial case: Replacement of an FPGA component in a legacy control system," in *Proc. IEEE 13th Intl. Conf. on Industrial Informatics*, pp. 208–214, July 2015.

[3] R. Karam, T. Hoque, S. Ray, M. Tehranipoor, and S. Bhunia, "MUTARCH: Architectural diversity for FPGA device and IP security," in *Proc. 22nd Asia and South Pacific Design Automation Conf.*, pp. 611–616, Jan 2017.

[4] S. Drimer, "Volatile FPGA design security a survey." Computer Laboratory, University of Cambridge, 2008.

[5] "Advanced Microcircuit Emulation (AME)." http://www.gemes.com/about_us/about_ame.

[6] L. S. Peters, "GEM: An innovative solution to the DMS problem." http://www.csl.sri.com/papers/dmc_paper1/dmc_paper1.html.

[7] I. Hadzic, S. Udani, and J. M. Smith, "FPGA Viruses," in *Proc. Intl. Workshop on Field-Programmable Logic and Applications*, pp. 291–300, 1999.

[8] S. Trimberger, "Trusted Design in FPGAs," in *Proc. 44th Annual Design Automation Conference*, pp. 5–8, 2007.

[9] Y. Pino, V. Jyothi, and M. French, "Intra-die process variation aware anomaly detection in FPGAs," in *Proc. 2014 ITC*, pp. 1–6, Oct 2014.

[10] R. S. Chakraborty, I. Saha, A. Palchaudhuri, and G. K. Naik, "Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream," *IEEE Design Test*, vol. 30, pp. 45–54, April 2013.

[11] S. Mal-sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware Trojan Attacks in FPGA Devices: Threat Analysis and Effective Countermeasures," in *Proc. ACM Great Lakes Symposium on VLSI*, pp. 287–292, May 2014.

[12] K. Thompson, "Reflections on Trusting Trust," *ACM Turing Award Lectures*, vol. 27, pp. 761–763, Aug. 1984.

[13] G. Bloom, B. Narahari, R. Simha, A. Namazi, and R. Levy, "FPGA SoC architecture and runtime to prevent hardware Trojans from leaking secrets," in *Proc. 2015 IEEE Intl. Symp. on Hardware Oriented Security and Trust*, pp. 48–51, May 2015.

[14] S. Mal-Sarkar, R. Karam, S. Narasimhan, A. Ghosh, A. Krishna, and S. Bhunia, "Design and Validation for FPGA Trust under Hardware Trojan Attacks," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, pp. 186–198, July 2016.

[15] R. Karam, T. Hoque, S. Ray, M. Tehranipoor, and S. Bhunia, "Robust bitstream protection in FPGA-based systems through low-overhead obfuscation," in *Proc. 2016 International Conference on ReConFigurable Computing and FPGAs*, pp. 1–8, Nov 2016.

[16] V. Jyothi, M. Thoonoli, R. Stern, and R. Karri, "FPGA Trust Zone: Incorporating trust and reliability into FPGA designs," in *Proc. IEEE 34th International Conference on Computer Design*, pp. 600–605, Oct 2016.

[17] C. Thompson, "Upgrading obsolete integrated circuits using Field Programmable Gate Arrays (FPGA)," in *2014 IEEE AUTOTEST*, pp. 365–371, Sept 2014.

[18] "Xilinx ise in-depth tutorial ug695 (v14.1)." https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ise_tutorial_ug695.pdf.