# Hardware Trojan Attacks in FPGA Devices: Threat Analysis and Effective Countermeasures

Sanchita Mal-Sarkar
Cleveland State University
Department of CIS
Cleveland, Ohio, 44115
s.malsarkar@csuohio.edu

Aswin Krishna, Anandaroop Ghosh and Swarup Bhunia
Case Western Reserve University
Dept. of EECS, Cleveland, Ohio, 44106
{ark70, axg468, skb21}@case.edu

## ABSTRACT

Reconfigurable hardware including Field programmable gate arrays (FPGAs) are being used in a wide range of embedded applications including signal processing, multimedia, and security. FPGA device production is often outsourced to off-shore facilities for economic reasons. This opens up the opportunities for insertion of malicious design alterations in the foundry, referred to as hardware Trojan attacks, to cause logical and physical malfunction. The vulnerability of these devices to hardware attacks raises security concerns regarding hardware and design assurance. In this paper, we analyze hardware Trojan attacks in FPGA considering diverse activation and payload characteristics and derive a taxonomy of Trojan attacks in FPGA. To our knowledge, this is the first effort to analyze Trojan threats in FPGA hardware. Next, we propose a novel redundancy-based protection approach based on Trojan tolerance that modifies the application mapping process to provide high-level of protection against Trojans of varying forms and sizes. We show that the proposed approach incurs significantly higher security at lower overhead than conventional fault-tolerance schemes by exploiting the nature of Trojans and reconfiguration of FPGA resources.

## Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Authentication

## General Terms

Design, Security

## Keywords

Hardware security, FPGA, Trojan, Trust

## 1. INTRODUCTION

Reconfigurable hardware platforms are integrated circuits (ICs), consisting of an array of logic blocks and distributed interconnect structure, which can be programmed and, in many cases, re-programmed to implement logic functions. FPGAs dominate the space of reconfigurable hardware. They are increasingly used in diverse embedded applications for improving performance and/or energy efficiency compared to software-based execution. Additionally, designs implemented on FPGAs do not suffer from the increasing non-recurring engineering (NRE) costs of ASIC production. Figure 1(a) shows a high-level block diagram of an FPGA and illustrates the structure of a programmable interconnect. They are typically designed as an interleaved array of configurable logic blocks and programmable interconnects.

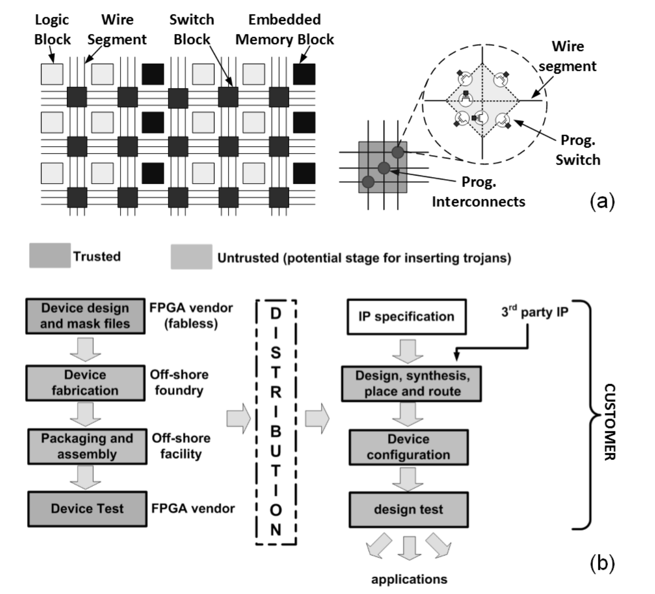


**Figure 1: (a) Conventional island-style FPGA architecture with programmable interconnects; (b) FPGA design flow from device design to deployment showing potential stages for malicious alterations.**

The growing use of FPGAs in diverse and critical applications has urged designers to think about security. In this context, security refers to protecting the intellectual property (IP) of the design mapped to a FPGA device from being stolen. However, little attention has been directed towards security and assurance of the FPGA device itself. Malicious alteration to the device is possible at several stages of design/fabrication flow of FPGA as shown in Fig. 1(b). Security in every stage of the design flow is of growing impor-

*GLSVLSI'14,* May 21–23, 2014, Houston, Texas, USA.
Copyright 2014 ACM 978-1-4503-2816-6/14/05 ...$15.00.
http://dx.doi.org/10.1145/2591513.2591561.

tance. After all, the security of a design mapped to FPGA is only as good as the security of the system itself.

FPGA system protection has been investigated earlier in different contexts [1, 3, 2, 4]. In [2], the authors describe the various logical and electrical attacks possible to cause malfunction and physical destruction of a FPGA device caused by creating internal conflicts in the device by inserting malicious code in the configuration files. Earlier works also present attack models such as replay attacks, cloning, power analysis attacks, invasive and semi-invasive attacks, and radiation attacks as related to the security of FPGA design [3]. None of these existing works, however, discusses hardware attacks in the foundry to cause malfunction and leak IP.

Malicious modifications of ICs, referred to as Hardware Trojans, have emerged as a major security threat due widespread outsourcing of IC manufacturing to untrusted foundries [7, 8, 9, 10, 11]. An adversary can potentially tamper with a design in these fabrication facilities by inserting malicious circuitry, intended to cause malfunction or leak secret information from inside a chip during field operation. Conventional post-manufacturing testing often fail to detect hardware Trojans due to their stealthy nature, complex structure, and inordinately large number of possible instances [10]. The condition of Trojan activation is referred as the triggering condition and the node(s) affected by a Trojan is referred to as its payload. Fig. 2(a) illustrates the general scenario of a Trojan attack in a design. Fig. 2(b) shows example of a combinational Trojan, which activates on simultaneous occurrence of a set of node conditions and a sequential Trojan, which activates on a sequence of rare events.

The issue of foundry trust related to the fabrication of FPGAs has been discussed in [1]. It argues that hardware attacks on FPGA devices in the foundry to tamper with the functionality of the final design are slim due to: (1) the foundry does not know about the design being implemented; (2) exhaustive testing of the bitstream security features to ensure that any attacks on them are detected during testing; and (3) possible destructive testing of a large number of chips to identify any extraneous logic.

In this paper, we present comprehensive analysis of hardware Trojan attacks in FPGA devices and propose effective protection approaches. To our knowledge, this is the first effort to analyze Trojan attacks in FPGA hardware and develop countermeasures against them. We show that even with the above-mentioned security features to ensure the integrity of FPGA devices, a variety of attacks are possible in the foundry. Firstly, we show that not all attacks have to depend on the final design and it is possible to insert malicious logic which are "independent" of the design and can also be used to leak the intellectual property implemented in the device. Moreover, an attacker in the foundry can distribute Trojans dependent on the internal logic values all over the chip. Secondly, even though bitstream security functions can be fully tested to ensure that no attacks are made on the FPGA's security, an attack can be made to steal a key and not cause malfunction. Thus, thoroughly testing the security functions may not help in protecting the IP from being copied by an attacker. Finally, even though number of FPGAs can be exhaustively (possibly through destructive process) tested after production, a Trojan may exist in only a subset of chips which may not be fully tested. In particular, the paper makes the following major contributions:

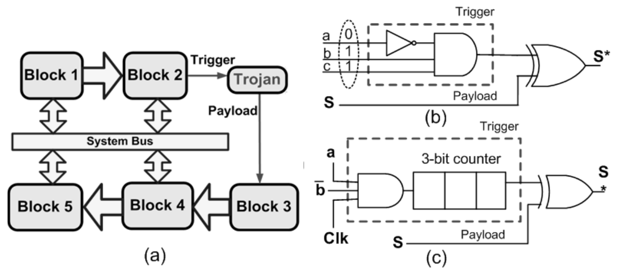


**Figure 2: (a) General model of a hardware Trojan realized through malicious design modification; example of (b) combinational, (c) sequential Trojan.**

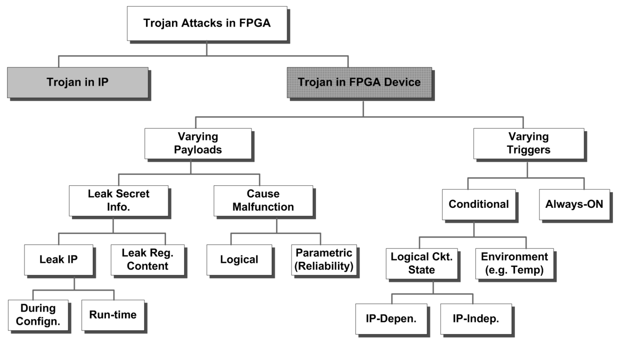


**Figure 3: Taxonomy of hardware Trojan attacks in FPGA devices.**

- It identifies the vulnerabilities; analyzes the spectrum of Trojan attacks on FPGA hardware; and provides Trojan models along with examples. It presents a taxonomy of Trojan attacks in FPGA which categorizes possible Trojan attacks in FPGA into different classes based on activation and payload characteristics.

- It presents a novel low-overhead redundancy based Trojan tolerance scheme for effective protection against diverse Trojan attacks in FPGA. It takes inspiration from existing run-time fault tolerance approaches [12, 13]. However, compared to these approaches (in particular, Triple Modular Redundancy or TMR), the proposed scheme has the following distinctions: (1) it leverages on dynamic reconfiguration capability of FPGA to minimize the overhead; and (2) it uses a novel variant-based redundancy scheme to maximize protection against Trojan attacks. The later enables detection of similar Trojan instances in multiple regular structures (e.g. CLBs or clusters).

## 2. HARDWARE TROJANS IN FPGA

Reconfigurable hardware consists of a regular array of programmable cells and other modules (e.g. decryptor, clock manager, DSP cores, block RAMs) connected through a distributed programmable interconnect structure. Since most of the chip is occupied by the regular structure of logic blocks and interconnect, it is relatively easy (compared to ASICs) for an attacker to reverse engineer the device and identify the components. For example, a DSP core or clock manager can be easily identified from the layout of the FPGA and can be potential target for Trojan attacks.

We have developed a taxonomy of FPGA-specific hardware Trojans that alter the programmed state of its logic, memory, interconnect and I/O blocks. We classify the FPGA hardware Trojans into two main categories as shown in Fig. 3, according to their activation and payload characteristics. Activation characteristics refer to the triggers or conditions that make a Trojan active while payload characteristics refer to the signal(s) that the Trojans affect. While it may be possible to classify FPGA Trojans based on other characteristics such as size, distribution, we believe that the proposed classification comprehensively covers FPGA Trojans and is adequate to evaluate the capabilities and limitations of detection methods.

## 2.1 Activation Characteristics

Based on the activation characteristics, Trojans can fall into two subcategories marked as *condition-based* and *always-on* in Fig. 3. Always-on Trojans are always active while condition-based FPGA Trojans wait until a particular condition is met before causing malfunction or leaking information. At this level, Trojans can be further classified as *logic-based* and *sensor-based* (e.g. temperature, delay). At the lowest level, logic-based FPGA Trojans can be further divided into *IP dependent* and *IP independent* categories as explained next.

*IP-dependent Trojans:* IP dependent Trojans represent ones whose trigger signals depend on the design implemented in the device. As shown in Fig. 4, an adversary can insert malicious circuit which monitors the logic values of several nodes such as configuration cells in the logic modules and interconnect structures, outputs of logic modules, look-up table (LUT) values. When triggered, such a Trojan can cause malfunction in many different ways, e.g. by altering the values stored in LUTs or configuration cells to cause incorrect routing between logic blocks or writing wrong values into block-RAMs (BRAM). These Trojans are highly likely to evade conventional FPGA testing, which cannot sensitize all possible trigger conditions of a Trojan.

Since the IP to be mapped is not available to the foundry during device fabrication, an attacker who plans to insert design-dependent hardware Trojans must do so without assuming any characteristic of the IP. Even though the probability of such a Trojan becoming active is very low, an attacker may distribute many such Trojans over the entire chip to increase chances of causing malfunction. A possible goal for the attacker in this case can be giving competitive edge to one FPGA vendor by creating bad reputation to another. Given the countless designs that can be mapped and the growing field of applications of FPGAs, IP dependent Trojans can be a practical threat that should be considered for hardware assurance.

*IP-independent Trojans:* An intelligent attacker can also insert Trojans whose activation conditions do not depend on the final design. Such Trojans can be inserted to alter the functionality of critical modules of the device. For example, Xilinx Spartan-3, Virtex-II, Virtex-II Pro FPGAs contain a separate module for clock management known as digital clock manager (DCM) as shown in Fig. 5. It contains a frequency synthesizer for producing a multiple or division of the input clock. The amount of phase shifting required and the amount of clock division or multiplication required

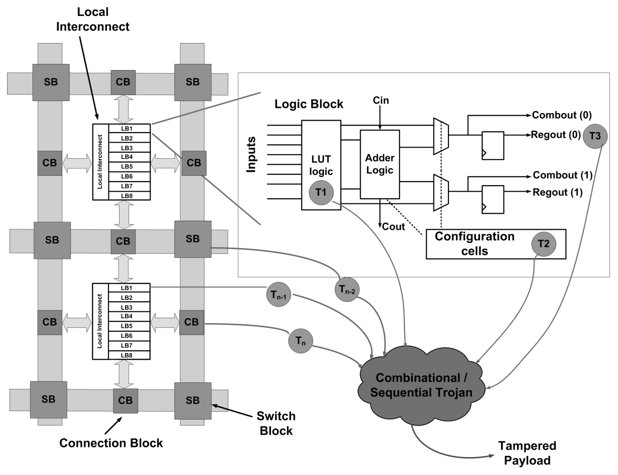


Figure 4: Simplified architecture of an FPGA showing the trigger points that a Trojan may use.

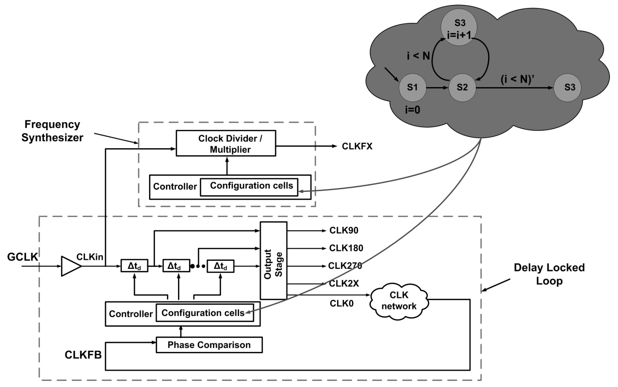


Figure 5: Schematic of digital clock manager (DCM) which can be affected by Trojan attack.

are stored in SRAM cells in the DCM. A possible Trojan design could be a counter that counts the clock edges to a specific number and then modifies the values of the SRAM cells to increase clock rate. A faster clock can cause delay failure in a sequential circuit. For a $N$-bit counter, since the final output occurs once in every $2^N$ clock cycles, even a reasonably large counter could evade conventional logic test methods.

## 2.2 Payload Characteristics

Hardware Trojans can also be classified into two categories based on their intended behavior.

*Trojans for malfunction:* Trojans in this class can be further classified into two subcategories based on whether they cause *logical* malfunction or *physical* malfunction. Trojans presented in the previous sections cause logical malfunction by modifying the values in the LUTs, causing undesired routing between two logic modules, etc. Fig. 6 shows additional examples of payloads affected by Trojans. Trojans intended to cause physical damage can create electrical conflicts at the I/O ports or at the programmable interconnects. Consider a typical programmable I/O block in FPGA. When a I/O port is configured to be an input by a design, the configuration cells in the I/O block should disable the output

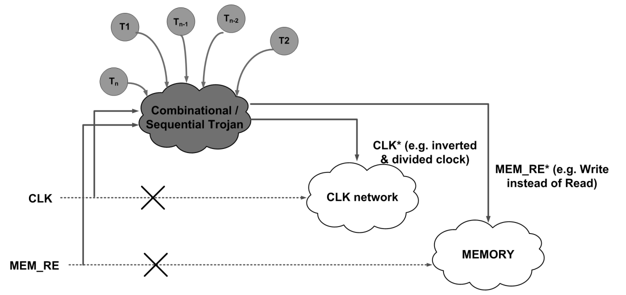
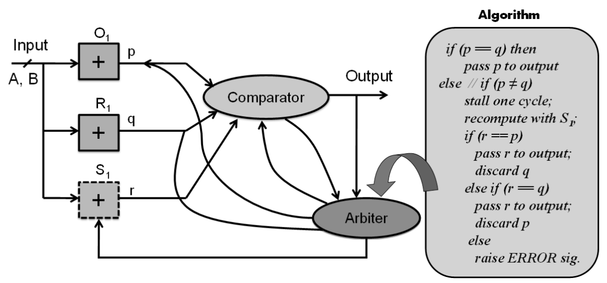

T1
$T_{n-1}$
$T_{n-2}$
T2
$T_n$
Combinational / Sequential Trojan
CLK* (e.g. inverted & divided clock)
MEM_RE* (e.g. Write instead of Read)
CLK
CLK network
MEM_RE
MEMORY

**Figure 6: Diagram showing examples of payloads that can be altered by Trojans.**



```
if (p == q) then
    pass p to output
else // if (p ≠ q)
    stall one cycle;
    recompute with $S_1$;
    if (r == p)
      pass r to output;
      discard q
    else if (r == q)
      pass r to output;
      discard p
    else
      raise ERROR sig.
```

**Figure 8: Proposed Trojan-tolerant application mapping scheme using redundant computing blocks and the output routing algorithm in arbiter.**

block to prevent internal conflicts. A counter-based Trojan can be inserted in the foundry which detects the state of the I/O port (i.e. I or O) and begins counting. When the counter counts to the final value, the Trojan may enable the output logic when the port is configured as an input. This would cause a high short-circuit current to flow between the FPGA and the external device, thus damaging the system.

*IP-leak Trojans:* Since IP designs involve a high development cost and contain sensitive information, security of IP is of critical importance against theft and reverse engineering. Many high-end FPGAs such as Xilinx's Virtex5 and Altera's StratixIII offer bitstream encryption to prevent unauthorized cloning of the bitstream. Fig. 7 shows the security features in a generic FPGA device which contains the programmable logic array (bottom right in the figure), configuration logic which controls the programming of the SRAM cells in the logic array, interconnect network and additional modules in the device [6, 5]. The device also contains a decryptor module for decrypting the bitstream using a key stored in a non-volatile memory. Security measures in the device (1) prevents the key from being read and sent to a port by clearing the configuration data and keys when an attempt is made, (2) prevents readback of the configuration data, and (3) restricts decryptor access after configuration [1]. However, all these measures only prevent malicious code in an IP from accessing the key or configuration data.

Hardware Trojans can leak an IP in two ways by leaking either (1) the decryption key, or (2) the design itself. An attacker in the foundry can insert extraneous circuit as shown in the Fig. 7 to tap the wires connecting the non-volatile memory and decryptor module. Even if the decryptor module is implemented in the logic array by using a decryptor bitstream as mentioned in [5], such an instantiated module must have access to the non-volatile key for decryption.

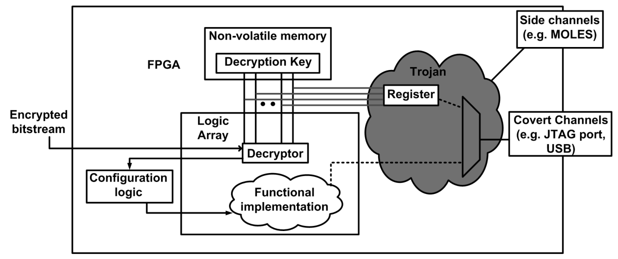


**Figure 7: FPGA device with security features for bitstream decryption.**

A copy of the key can be stored in the Trojan which may then leak it through side-channels or covert-channels. For example, the MOLES Trojan presented in [9] uses a spread-spectrum technique to leak the key in the power traces over several clock cycles. Alternatively, a Trojan may also mux the JTAG port, USB port, or any ports to leak the key through I/O channels when they are not being used.

## 3. HARDWARE TROJAN TOLERANCE

In this section we introduce a novel protection approach against Trojan attacks in FPGA devices through tolerance of Trojan effects during operation. It works by either containing the Trojan effect or bypassing the effect of Trojan using spare units. We propose a hybrid scheme of hardware Trojan tolerance that combines an efficient redundancy based mapping approach with dynamic reconfiguration.

### 3.1 Adapted TMR (ATMR)

TMR is a well-known fault mitigation technique that masks circuit faults by using redundant hardware [12, 13]. A TMR circuit has three redundant copies of the original circuit and a majority voter. A single fault in one of the redundant hardware modules will not produce an error at the output as the majority voter selects the result from the two working modules. Use of TMR has been explored earlier in FPGA in the context of tolerating run-time failures e.g. soft error. However, it typically comes at the cost of about three times the size and power of the original circuit.

We note that appropriate adaptations in TMR for Trojan tolerance can make significant improvement on redundant hardware usage and power consumption by employing two modules at a time, instead of three. The outputs of the modules are compared using a comparator circuit. The third module is used on demand when the comparator circuit detects a mismatch in outputs. With the help of an arbiter, the comparator determines which module is in error; discards it; and then creates correct output using the third one. It, however, requires to halt the circuit for few cycles (typically 1-2) in order to transfer control to the arbiter on a mismatch and to include the third spare component in computing.

Fig. 8 shows three adders ($O_1$, $R_1$, and $S_1$) that are mapped to the Trojan infected region of an FPGA and their corresponding outputs ($p$, $q$, and $r$). The comparator will compare the outputs ($p$, $q$) of the first two adders ($O_1$, $R_1$). The third adder ($S_1$) will not be used unless there is a mismatch between the two outputs $p$ and $q$. In case of a mismatch,

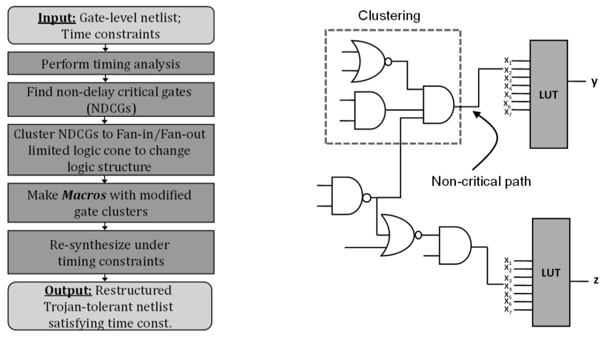

Input: Gate-level netlist; Time constraints
Perform timing analysis
Find non-delay critical gates (NDCGs)
Cluster NDCGs to Fan-in/Fan-out limited logic cone to change logic structure
Make Macros with modified gate clusters
Re-synthesize under timing constraints
Output: Restructured Trojan-tolerant netlist satisfying time const.
Clustering
LUT
y
Non-critical path
LUT
z

**Figure 9: Flowchart showing a variant generation approach from a gate-level design and an example.**

the comparator with the help of the arbiter continues comparing the output $r$ with the outputs ($p$, $q$) until it finds a match and it determines which adder is in error. Then the comparator outputs the correct result and prevents the propagation of erroneous data. Note that although the third replica is required to determine which one is in error, only two are enough to flag that one of them is infected with a Trojan, prevent the propagation of the Trojan's payload in the system and/or stop leakage of potentially sensitive data. Therefore the scheme we have proposed, even without the third replica, will be of particular interest to many mission-critical applications.

## 3.2 Improving Protection through Variants

The Trojan tolerance scheme proposed in Section 3.1 cannot protect against simultaneous activation of identical Trojan instances in $O_i$ and $R_i$. An adversary can incorporate the same Trojans in two or more clusters or CLBs in an FPGA, so that both $O_i$ and $R_i$ can be similarly affected. For example, this can happen if $O_i$ and $R_i$ are identically mapped in two different clusters, both of which has a combinational Trojan triggered by specific combination of LUT content. In such a scenario, two Trojans in $O_i$ and $R_i$ would trigger at the same cycle with identical malicious effect and the proposed tolerance approach would fail to detect it. To address this scenario, we enhance the proposed scheme by implementing ***variants*** of a module e.g. the adders for $O_i$ and $R_i$ to ensure that $O_i$ and $R_i$ functionally compute the same result, but the adders are implemented in a structurally different manner. It is highly unlikely that both adders ($O_i$ and $R_i$) will simultaneously trigger the same Trojan because different implementations involve different logic blocks, storage elements, logic structures, and memory locations. Thus we can assume that both $O_i$ and $R_i$ cannot simultaneously trigger the same Trojan.

A judicious design of structural variants can tolerate simultaneous activation of Trojans in both original and replica modules. The variants need to differ in both LUT and interconnect structures while maintaining the functional behavior and parametric specifications (e.g. critical path delay). Fig. 9 shows the flow chart and an example for implementing the variants. It starts with finding non-delay critical gates (NDCGs) and regrouping them into clusters. The clusters are derived using a hypergraph partitioning approach with specific fan-in/fan-out limits. It ensures that the content/type/number of the LUTs, and interconnections among them are changed while critical path delay remains unchanged. Next, we convert the regrouped LUTs as "hard macros" (to avoid re-optimization during synthesis). The last step is to re-synthesize the circuit including the macros with the original time constraints. Hence, even when both replicas are infected with a Trojan, the Trojan is very unlikely to be simultaneously activated during operation in both original and replica modules. Hence, the proposed scheme can overcome the limitation of TMR since it can protect against multiple Trojan effects in functional units.

## 3.3 Trojan Tolerance in the controller/arbiter

If the arbiter or the comparator is compromised i.e. has Trojans, we can: (a) use majority voting also for comparator and arbiter, or (b) exhaustively test the them to validate their trustworthiness. The first approach can improve the integrity of the system but hardware overhead can be considerable. The second one, however, can be more attractive to designers since the small comparator and arbiter circuits are amenable for exhaustive testing at modest test cost.

## 4. SIMULATION RESULTS

In this section we present case studies with a commercial FPGA device (Altera Cyclone IV GX) and a large benchmark design (32-bit pipelined DLX processor) to analyze the effectiveness of the proposed Trojan tolerance approach. The processor pipeline is broken into five stages (Fetch, Decode, Execute, Memory Access, and WriteBack). We applied both TMR and ATMR to the combinational logic for each sequential boundary to mitigate the effect of hardware Trojans in them. First, we present the mapping results for the processor execution unit followed by results for the entire processor. Area, delay and power overhead for the proposed scheme are compared with those for conventional TMR.

Figure 10 show the variation in area requirement, power consumption and delay of the mapped ALU in original mapping scheme (using area constraint) and variant mapping scheme (using delay constraint). Table 1 compares the overhead, in terms of power, resources, and performance of TMR and our hybrid tolerance scheme (ATMR). It illustrates that ATMR requires 1.5X less power than TMR to achieve the same level of security while maintaining equivalent resource and performance requirements as in TMR. This result is expected given that ATMR, unlike TMR, uses the third spare resources/replicas only when Trojans are activated and a mismatch occurs in the outputs of the initial two replicas. Similar performance (delay) of TMR and ATMR indicates that ATMR does not require additional processing cycle.

Table 2 presents the overhead of TMR and ATMR when a judicious design of structural variants is introduced in FPGAs. As shown in Table 2, ATMR with variants requires 1.5X less power than TMR with variants with similar performance and resource usage. Moreover, ATMR with variants provides higher level of security since it can protect against simultaneous activation of identical Trojan instances in two or more clusters or CLBs. The improvement in power consumption in ATMR is due to the use of the third spare resources/replicas only in case of Trojan activation.

## 5. CONCLUSION

We have presented possible malicious changes i.e. hardware Trojan attacks in FPGA that can be inserted during

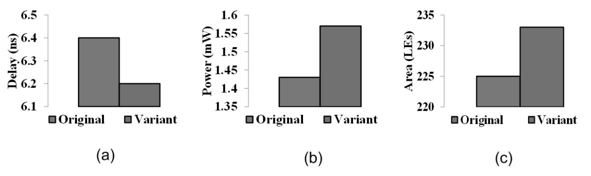


**Figure 10: Single ALU (a) delay (b) power and (c) area results in conventional and the proposed variant-based mapping scheme.**

**Table 1: Comparison of design overhead between TMR and ATMR**

| | TMR | ATMR | Times Impr. |
|---|---|---|---|
| Power | 4.70 mW | 3.15 mW | 1.5X |
| Resource | 850 LEs | 856 LEs | 1X |
| Performance | 6.7 ns | 6.7 ns | 1X |

*LEs = Logic Elements in FPGA

**Table 2: Comparison of overhead between TMR and ATMR with variants**

| | TMR with Variants | ATMR with Variants | Times Impr. |
|---|---|---|---|
| Power | 4.95 mW | 3.26 mW | 1.5X |
| Resource | 860 LEs | 872 LEs | 1X |
| Performance | 6.4 ns | 6.4 ns | 1X |

device production. As FPGAs are being increasingly used in wide array of applications including many defense and other security-critical applications, vulnerability of FPGA devices against hardware Trojan attacks pose a major security threat. We have presented a taxonomy of hardware Trojan attacks in FPGAs, including models and specific examples of Trojans that cause logical malfunctions and/or physical damage. We have also shown the possibility of Trojan attacks targeting information leakage from inside an FPGA during operation. Next, we have proposed a novel Trojan tolerant application mapping scheme that can effectively protect against Trojan attacks of varying sizes and functionalities. We compared our scheme with the conventional redundancy-based error tolerance approach. The proposed Trojan tolerance scheme incurs significantly less power overhead, while providing higher level of security. It is easily scalable to larger designs. Future work will include efficient detection of hardware Trojans in FPGA and developing a metric for estimating FPGA trust.

## 6. ACKNOWLEDGMENTS

This work is funded in part by National Science Foundation (NSF) Grant #1245756 and #1054744. The authors acknowledge valuable suggestions by Dr. Seetharam Narasimhan.

## 7. REFERENCES

[1] S. Trimberger, "Trusted design in FPGAs," *Design Automation Conference*, 2007.

[2] I. Hadzic, S. Udani, and J. Smith, "FPGA viruses," *International Workshop on Field Programmable Logic and Applications*, 1999.

[3] S. Drimer, "Volatile FPGA design security: a survey," Cambridge University, 2008.

[4] T. Huffmire, "Handbook of FPGA design security," *Design Automation Conference*, 2007.

[5] S. Trimberger, "Method and apparatus for protecting proprietary decryption keys for programmable logic devices," US Patent 6654889, 2003.

[6] Aletar: Military Anti-Tampering Solutions Using Programmable Logic. [Online]. Available: http://www.altera.com/literature/cp/CP-01007.pdf.

[7] D. Du, S. Narasimhan, R. S. Chakraborty, and S. Bhunia, "Self-referencing: a scalable side-channel approach for hardware Trojan detection,", *Workshop on Cryptographic Hardware and Embedded Systems*, 2010.

[8] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Treats and Emerging Solutions," *International High Level Design Validation and Test Workshop*, pp. 166-171, 2009.

[9] L. Lin, W. Burleson, and C. Paar, "MOLES: malicious off-chip leakage enabled by side-channels," *International Conference on Computer-Aided Design*, 2009.

[10] R.S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: A Statistical Approach for Hardware Trojan Detection," *Workshop on Cryptographic Hardware and Embedded Systems*, 2009.

[11] F. Wolff, C. Papachristou, S. Bhunia, and R.S. Chakraborty, "Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme," *Design Automation and Test in Europe*, 2008.

[12] H. Kubatova and P. Kubalik, "Fault-Tolerant and Fail-Safe Design Based on Reconfiguration," *Design and Test Technology for Dependable Systems-on-Chip*, pp. 175-194, 2011.

[13] N. Gaitanis, "The Design of Totally Self-Checking TMR Fault-Tolerant Systems," *IEEE Transaction on Computers*, vol. 37, no. 11, 1988.