A New Characterization of Hardware Trojan Attributes

Samer Moein, *Student Member, IEEE,* T. Aaron Gulliver, *Senior Member, IEEE,* and Fayez Gebali, *Life Senior Member, IEEE*

***Abstract*—This work examines hardware trojan threats for semiconductor chips intended for critical infrastructure and applications. Several real world examples are discussed. The phases of the chip production life-cycle are reviewed and opportunities for trojan insertion are examined. Trojans are classified using a comprehensive attribute taxonomy based on eight categories. A matrix identifying the causal relationships between these attributes is defined. This matrix is used to characterize hardware trojans from both the attacker and defender perspectives.**

***Index Terms*—Hardware trojans, chip life cycle, hardware trojan taxonomy, hardware trojan attributes.**

I. Introduction

We begin with several real world examples to motivate this work. In 2009, several countries participated in a multinational air manoeuvre. One of the exercises stipulated that airplanes from country B were to launch missiles against airplanes from country A. The aircraft used by country B were manufactured in country A. The missiles failed to launch even after numerous attempts. A thorough inspection of the airplanes from country B was unsuccessful in determining the reason for the failure of the weapons control systems. After the manoeuvre was over, the airplanes from country B used the missile systems and no problems were encountered. A possible reason for the failure is that a hardware backdoor was used to disable the missile systems [1].

Between 2011 and 2012, many owners of a certain car make and model observed a particular error message displayed on their dashboards. It was suspicious that this error appeared after the expiration of the car warranties. The car dealers informed the owners that their computer system had to be reprogrammed and if the problem appeared again, the computers should be replaced. From the timing of this event, it could be concluded that a timed failure was inserted into the computer systems of these cars [2].

In September 2007, Israeli jets bombed a suspected nuclear installation in northeastern Syria. Among the mysteries surrounding this airstrike is the failure of the Syrian radar systems. It has been suggested that the commercial off-the-shelf microprocessors used in these systems may have been fabricated with a hardware backdoor which was used to temporarily disable them [3].

S. Moein is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, Canada.

T. Gulliver and F. Gebali are with University of Victoria.

Manuscript received April 19, 2005; revised September 17, 2014.

There are many documented cases of similar incidents which have caused significant government, industry, and consumer concerns. In response, the Defence Advanced Research Projects Agency (DARPA) initiated the trust in ICs program to develop techniques for trojan detection [3], [4]. This highlights the fact that hardware designers and researchers must be vigilant to the insertion of hardware trojans during all phases of the chip production life-cycle. Examination of these trojans is the main goal of this paper.

Several researchers have proposed taxonomies for hardware trojans based on their attributes [5]–[8]. In [7], hardware trojans were classified based on two categories: trigger and payload. These are in fact activation mechanisms for trojans. In [6] and [8], the classification was based on three categories: physical, activation, and action. Although this adds two categories to the previous taxonomy, the classification is not related to the chip life-cycle. In [5], a more detailed classification was developed based on five categories: insertion phase, abstraction level, activation mechanism, effect, and location. This classification considers the chip life-cycle and the targeted location, but not the physical characteristics of trojans. The taxonomy in [5] was tested using a set of trojans to verify the classification, and the attributes corresponding to each trojan was identified. However, the relationship between the attributes associated with a hardware trojan has not been investigated in the literature. In this work, a comprehensive hardware taxonomy is developed in Section III, and this is used as an example to illustrate the new methodology presented in Section IV.

The proposed methodology is flexible and can be used with any hardware trojan classification. Further, new attributes based on technology or chip manufacturing developments can easily be accommodated. As with any circuit, a hardware trojan goes through several production phases as it becomes embedded into the target system. Therefore, studying the life cycle along with other attributes will provide a better insight into the insertion phase, functionality, logic type, physical characteristics, and location of a trojan.

The approach presented in this paper has two major advantages over existing results. First, the relationships between the hardware trojan attributes are clearly defined based on a comprehensive taxonomy with eight categories. Second, the production life-cycle of a chip is used to determine how and where a trojan can be inserted into a chip. The relationships between the attributes can be used to identify missing trojan attributes. This information can be employed to improve chip security during manufacturing, develop trojan detection

techniques, and assess chips when covert attacks occur [9].

The remainder of this paper is organized as follows. Section II reviews the chip production life-cycle. Section III provides a comprehensive hardware trojan taxonomy based on eight categories: insertion, abstraction, effect, logic type, functionality, activation, physical layout, and location. Based on these categories, an algebraic approach to investigating hardware trojans is given in Section IV. Two case studies are presented in Section 5. Finally, Section VI provides some concluding remarks.

II. Chip Life-cycle

Integrated circuits (ICs) are becoming increasingly vulnerable to hardware trojan attacks due to design outsourcing, overseas fabrication and increasing reliance on third-party intellectual property (IP). In addition, hardware attacks can originate from the use of unverified electronic design automation tools. Figure 1 shows the modern IC production life-cycle phases: ***specification***, ***design***, ***fabrication*** (including interfacing, masking, wafer fabrication and assembly, and packaging), ***testing*** and ***deployment*** [10]–[13].

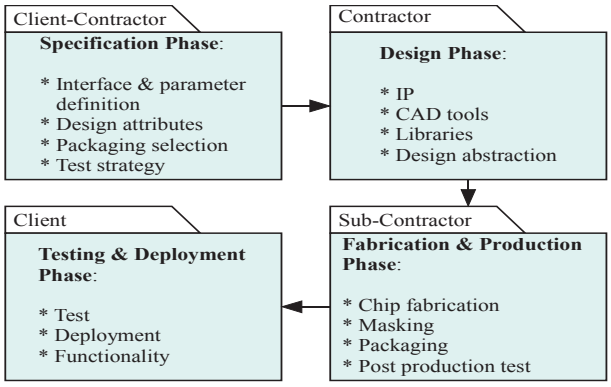


Fig. 1: The integrated circuit (IC) design life-cycle phases.

Complete protection of a chip during its life-cycle is only possible if it is produced by a trusted source. The stages in the production process have been classified as ***trusted***, ***untrusted***, or ***either*** [10]. This is based on the level of control over a given stage. A ***trusted*** stage indicates that the designer has complete control, whereas an ***untrusted*** stage means that the designer has only limited control. ***Either*** indicates that control may be complete or limited. This classification assumes that the specification, testing and assembly phases are trusted. However, the increasing complexity of chip design and fabrication requires state-of-the-art infrastructure and very specialized processes at the production facilities. Often the *design* and *fabrication* must be carried out simultaneously, requiring support for multiple capabilities and mixed technologies. This creates significant economic challenges in meeting production quality and demand which has caused increased outsourcing to *contractors* and *subcontractors* during the design and manufacturing phases. Unfortunately, outsourcing significantly reduces trust [14]. In fact, if the design phase is outsourced, it also reduces trust in the fabrication phase. Further, test vectors can be modified during the design phase, which leads to untrusted testing. Even if the testing phase is trusted, it does not necessarily mean that the assembly phase is trusted. This can only be ensured if it is done under client control.

There is typically a high level of control over the ***specification phase*** since the *client* and *contractor* must work closely together to define the system blocks, design specifications, packaging, and test requirements. These are reviewed and validated to confirm the target die size and fabrication process [15]. However, deficiencies in the specification or incomplete validation can lead to vulnerabilities.

There is less control over the ***design phase*** primarily because many manufacturers outsource their designs. Further, using standard cell design abstractions, design tools, design kits and design libraries can reduce trust [16]. The ***fabrication and production phase*** of an IC typically involves hundreds of steps. CAD tools are used to define the doping, metallization and glass region masks to transfer the design onto a silicon wafer. This process is repeated to embed the masks for the IC layers on the silicon. Each layer is tested for functionality and defective chips are discarded after the wafer is cut into individual ICs. The chips are then bonded to a mounting package and contact leads are attached. The mounting package is encapsulated with a plastic coating for protection and identifying part numbers and other data are added. This is followed by the ***testing and deployment phase*** in which ICs undergo final testing before they are distributed [16].

If the production life-cycle is not completely controlled by the designer, reverse engineering, malicious circuit insertion, circuit modification, and intellectual property (IP) piracy are possible. Of particular concern is an untrusted IC production facility inserting hardware trojans into chips unknown to the designer.

III. Trojan Taxonomy

The growing application of semiconductor chips in critical applications such as military infrastructure, industrial automation, supervisory control and data acquisition (SCADA) systems and the smart grid, has made security a serious concern. Tampering during the IC life-cycle can cause it to respond in an unexpected manner or worse in a manner determined by an attacker.

A *hardware trojan* is defined as a malicious component embedded in an integrated circuit which causes abnormal behaviour [17]. Hardware trojans can be implemented in microprocessors, microcontrollers, network and digital signal processors, field programmable gate array (FPGAs), application specific integrated circuits (ASICs), and other integrated circuits. Any attempt to address the growing hardware security concerns should begin with an investigation of the threats based on the numerous processes involved in the IC production life-cycle. However, it is extremely challenging to characterize, classify, and detect hardware trojans. Trojans have previously been classified by their *physical* characteristics, by the *effect* they have on a system, and by their *activation* mechanisms [18] [8]. These approaches are based on the assumption that the trojans are inserted only during *fabrication* [5]. Typically

the specification and design phases are considered trusted [10] [14], but here the vulnerability of these phases is also considered. In addition, a comprehensive model for trojan classification is presented which is based on eight key categories: insertion (*when was it inserted?*), abstraction (*where was it inserted?*), effect (*what can it do?*), logic type (*what logic does it employ?*), functionality (*what capabilities does it have?*), activation (*how is it activated?*), physical layout (*how does it appear?*), and location (*where is it located?*). This classification is illustrated in Figure 2.

The **insertion** category represents the hardware manufacturing process. This process has many levels and each level is vulnerable to malicious insertions that can cause abnormal functionality [3], [17]. Modifications can be as early as the *specification phase* where parameters such as the size, structure, type, intended function, power, timing and delay can be targeted. Design constraints and deficiencies can be exploited by an attacker. Protocols and functions can also be modified to insert trojans [3], [17]. Alterations to the specification lead to changes in the design which result in the production of a modified IC. In the *design phase*, factors concerning the logical, functional, timing and physical constraints of realizing the design on the target hardware technology are considered. In this phase, designers typically use different IP blocks, design tools, standard cell libraries and templates. These can contain malicious components if they originate from untrusted or commercial off-the-shelf sources. The *fabrication phase* involves preparation of the silicon wafers and processes such as masking, photolithography, doping, etching and interconnections to complete the manufacturing and packaging. An attacker can introduce trojans by altering parameters in these processes, modifying the mask geometry and layout, or changing chemical compositions.

In the *testing phase*, hardware trojans can be inserted, or hardware trojans inserted earlier can be concealed from detection. Editing during the design and prototype stages can result in large blocks of unused circuitry from previous iterations. This latent circuitry can be exploited by attackers to reroute signals and trigger trojans [3], [16], [19]. Testing is critical as it is the easiest phase to detect trojans. However, testing at an untrusted facility can lead to test vectors being revealed and detection deficiencies [18]. An attacker can exploit this information to omit or mask test vectors that will reveal hardware trojans. Trojans may also have been inserted in the design which will return acceptable values for test vectors [5].

In the *assembly phase*, hardware components are interfaced on the chip. Every interface between components is a possible trojan insertion point. At an interface, improper termination or improper shielding against phenomena such as electromagnetic coupling makes the chip susceptible to exploitation by an attacker. For example, deficiencies can be identified to inject faults or gather sensitive information.

The **abstraction** category represents the level at which a hardware trojan is introduced [3], [17]. The sophistication required to inject a trojan at a given abstraction level varies depending on the level.

A trojan can be introduced in the *system level* by altering interconnections, hardware modules, or communication protocols. This does not require a high level of sophistication. At the *register-transfer level (RTL)*, signals, registers and boolean functions are described in the function modules. Trojan insertion opportunities are very high in this level since an attacker can gain control over the hardware functionality [3], [5]. The *development environment* of an IC involves the simulation, verification, validation and synthesis. Software can be inserted into this abstraction level through CAD tools and scripts to hide the effects of hardware trojans [20]. The *logic (gate level)* is also prone to trojan insertion, but it is considered relatively secure since tampering requires a high level of sophistication. Logic design alteration is possible at the gate level or the netlist design [19].

The *transistor* level provides attackers with the opportunity to control circuit parameters such as power and timing, and trojans can be inserted by resizing or deleting existing transistors, or inserting new transistors, to modify the circuit functionality and characteristics. Modifications at the *physical* level involve the transistors and/or layout and are typically achieved by changing circuit parameters that affect the reliability and functionality.

The **effect** category describes the disruption or effect a trojan can have on the system. These effects are: *change in functionality, information leakage, reduced reliability, and denial of service*. A *change in functionality* is caused by trojans that introduce new logic or bypass existing logic to produce unexpected results. This can also be achieved by deleting logic [18]. A trojan can also cause *information leakage* through a covert or existing channel. These channels may be radio frequency based or via JTAG or RS232 interfaces, and provide backdoor access to assist in compromising the chip. Information such as encryption keys can also be leaked through thermal or optical patterns created by the hardware [9], [19]. Hardware trojans can also cause *reduced reliability* by altering interface, functional or circuit characteristics such as path delay and power consumption. Increased power consumption may cause the ambient temperature of the circuit to rise above normal operating levels and/or cause quicker battery depletion. Finally, a *denial of service (DOS)* trojan modifies device parameters to exhaust on-board resources such as power or memory, or introduces computational delays to degrade performance or create malfunctions.

The **logic type** category considers the circuit logic that triggers the trojan. A *combinational* trojan uses a particular logic value at one or more circuit locations as the trigger. A *sequential* trojan is triggered by a sequence of conditions after a given period of operation [21] [22].

The **functionality** category includes hardware trojans which are *functional* or *parametric*. A *functional* trojan introduces a change in the functionality of the device [23]. A *parametric* trojan exploits the parametric effects of the device circuitry such as power consumption, thermal and delay profiles [24]. This is achieved by weakening transistors, modifying the length and/or thickness of wires, or changing physical geometries [25].

The **activation** category is based on whether a trojan is always on or triggered. An *always on* trojan is always active. A triggered trojan will cause some unintended or malicious

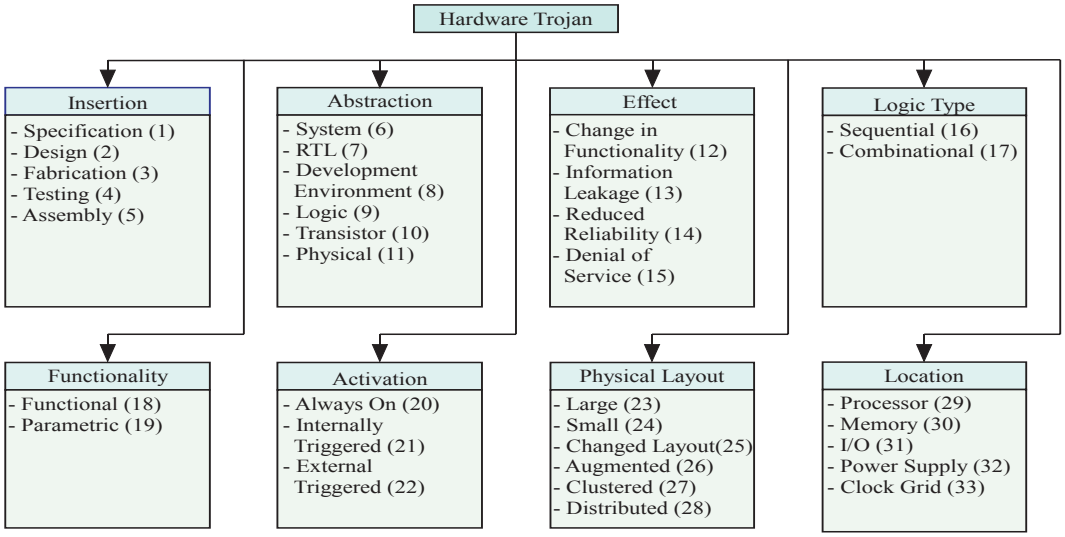


Fig. 2: The hardware trojan taxonomy.

effect only after it is activated [19]. The trigger can be either *external* or *internal*. An *externally triggered* trojan is activated via an external signal received by an antenna or sensor that interacts with the outside world. An attacker can then be at a geographically distant location. An *internally triggered* trojan waits for an internal condition which can be a sequence of one or more events that occur in the system. This condition is typically an internal logic state or a pattern of input/output signals [18].

The **physical layout** category is based on the physical characteristics of hardware trojans. Hardware trojans have been developed with various sizes and layouts to evade detection. Some trojans are very *small* and thus are virtually undetectable by inspection of the power consumption or heat dissipation [26]. Other trojan are *large* and often employ unused circuitry in the device to avoid detection. The size of trojans is determined by the number of deleted, added or compromised components in the chip. The distribution of a trojan is determined by the layout of the trojan on the chip. A *clustered* trojan has a topology in which the components are close to each other. A *distributed* trojan has a sporadic topology and can be dispersed throughout the chip. Some trojans employ a *changed layout* structure where an existing layout is modified. If the layout is added to it is known as an *augmented* circuit [25].

The **location** category is based on where the trojan is located. As mentioned above, trojans can be *distributed* or *clustered* among IC components. They can be located in components such as the *processor, memory, input/output, power supply, or clock grid*. A trojan injected into the *processor* can be located in the logical units so that it can change the instruction or execution cycles. Trojans in the *memory* units or interfaces can create incorrect addresses, modify memory contents, or enable/disable read/write instructions. The *input/output* peripherals have shared interfaces with external devices through communication and data buses such as serial ports. Trojans located here can modify the data or alter the way these devices communicate with the IC components. The *power supply* is a location where trojans can be placed to create effects such as denial of service or reduced reliability. These effects are produced by varying the current and/or voltage supply to the chip to cause malfunctions or abnormal behaviour. Finally, a trojan in the *clock grid* can cause variations in the clock frequency, or skip or freeze the clock signals supplied to chip modules [5].

IV. Algebraic Approach to Hardware Trojan Attributes

In this section, an algebraic approach to hardware trojan attributes is presented. Figure 2 provides a comprehensive classification of these attributes. There are many connections between these attributes which indicate their relationships. The attributes belong to one of four trojan levels: chip life-cycle, abstraction, properties, and location, as shown in Figure 3.

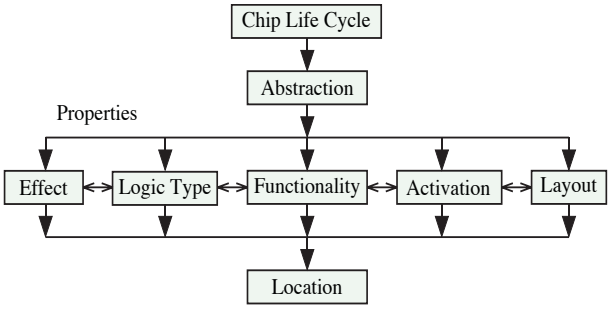


Fig. 3: The four hardware trojan levels.

The chip life-cycle level in Figure 3 is equivalent to the insertion category in Figure 2. In this level, the attacker decides at which phase of the chip life-cycle the trojan will be inserted. According to this decision, the abstraction of the trojan insertion will be determined. Therefore, the abstraction category in Figure 2 is the same as the abstraction level in Figure 3. The trojan properties level contains the properties

a trojan may have according to the chip life-cycle and abstraction levels. The possible location for a trojan is based on the chip life-cycle, abstraction, and properties levels. Each trojan has a particular combination of attributes which define its characteristics. These are represented using an $n \times n$ matrix defined as

$$\mathbf{R} = \begin{bmatrix} R_1 & R_{12} & 0 & 0 \\ 0 & R_2 & R_{23} & 0 \\ 0 & 0 & R_3 & R_{34} \\ 0 & 0 & 0 & R_4 \end{bmatrix}$$

where n is the number of attributes.

A. Hardware Trojan Matrix

The hardware trojan matrix $\mathbf{R}$ has four square submatrices: $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, and $\mathbf{R_4}$. These matrices represent the levels in Figure 3. The transfer matrices $\mathbf{R_{12}}$, $\mathbf{R_{23}}$, and $\mathbf{R_{34}}$ represent the connections between the levels.

1) Single Element: In the matrix $\mathbf{R}$, $r(i,j) = 1$ indicates that attribute i can lead to attribute j

$$\forall i,j: \quad r(i,j) = 1 \Rightarrow \text{attribute}(i) \rightarrow \text{attribute}(j). \tag{1}$$

By definition $r(i,i) = 0$.

2) Row: In the matrix $\mathbf{R}$, $r(i,j) = 1$ in row i indicates that attribute i can lead to these attributes in column j

$$\forall i: \quad r(i,j) = 1 \Rightarrow \text{attribute}(i) \rightarrow \text{attribute}(j) \tag{2}$$

3) Column: In the matrix $\mathbf{R}$, $r(i,j) = 1$ in column j indicates the attributes that can lead to attribute j

$$\forall j: \quad r(i,j) = 1 \Rightarrow \text{attribute}(i) \rightarrow \text{attribute}(j) \tag{3}$$

4) Fan In: The fan in is the number of connections to an attribute from other attributes. This indicates how many other attributes can lead to this attribute. The fan in can be expressed as

$$F_{in}(j) = \sum_{i=1}^{n} r(i,j). \tag{4}$$

5) Fan Out: The fan out is the number of connections from an attribute to other attributes. This indicates how many attributes this attribute can lead to. The fan out can be expressed as

$$F_{out}(i) = \sum_{j=i}^{n} r(i,j). \tag{5}$$

6) Root Attribute: An attribute with $F_{in} = 0$ is called a root attribute. It is an attribute that is not the result of any other attribute so that

$$F_{in}(j) = \sum_{i=1}^{n} r(i,j) = 0. \tag{6}$$

7) Terminal Attribute: A terminal attribute is an attribute with $F_{out} = 0$. It is the location of a trojan inserted in a system. It does not lead to any attributes so that

$$F_{out}(i) = \sum_{j=i}^{n} r(i,j) = 0. \tag{7}$$

8) Intermediate Attribute: An intermediate attribute has $F_{in} \neq 0$ and $F_{out} \neq 0$. Thus it is an attribute that can be a consequence of other attributes, and also lead to other attributes.

B. Submatrix $\mathbf{R_1}$

Submatrix $\mathbf{R_1}$ represents the trojan chip life-cycle level shown in Figure 3. It also shows the relationships between attributes in the insertion phase in Figure 2. These start with specification (attribute 1), and end with assembly (attribute 5). For example, $\mathbf{R_1}(1,2) \equiv \mathbf{R_1}$(Specification, Design) = 1 indicates that if there is an incomplete or incorrect specification, it can lead to an improper or defective design.

$$\mathbf{R_1} = \left[\begin{array}{c|ccccc} A & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

C. Submatrix $\mathbf{R_2}$

Submatrix $\mathbf{R_2}$ represents the trojan design level in Figure 3. It also shows the relationships between the attributes in the abstraction level in Figure 2. The abstraction level has six sublevels beginning with system (attribute 6) and ending with physical (attribute 11). A modification in a sublevel can propagate to the following sublevels. For example, $\mathbf{R_2}(6,7) \equiv \mathbf{R_2}$(System, RTL) = 1 indicates that if there is an alteration to the interconnections, hardware modules, or communication protocols, it can lead to the RTL signals, registers, or boolean functions being compromised.

$$\mathbf{R_2} = \left[\begin{array}{c|cccccc} A & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 6 & 0 & 1 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 1 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 1 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 1 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

D. Submatrix $\mathbf{R_{12}}$

Submatrix $\mathbf{R_{12}}$ describes the connections between $\mathbf{R_1}$ and $\mathbf{R_2}$. This shows the relationships between the insertion attributes and the abstraction attributes.

For example, $\mathbf{R_{12}}(2,7) \equiv \mathbf{R_{12}}$(Design, RTL) = 1 indicates that if there is an attempt to insert a trojan in the design level, a modification of the RTL can be used to accommodate this alteration.

It is clear that the development environment (attribute 8) and transistor (attribute 10) are not directly connected to the insertion phase, but both are indirectly connected through paths in $\mathbf{R}$. The system level (attribute 6) has the highest *fan in* as it is connected to specification (attribute 1), testing (attribute 4), and assembly (attribute 5) from the insertion category. These three attributes are connected to the system level as control over these is required at the abstraction category.

$$\mathbf{R_{12}} = \left[\begin{array}{c|cccccc} A & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

E. Submatrix $\mathbf{R_3}$

Submatrix $\mathbf{R_3}$ represents the properties of a trojan given in Figure 3 and contains attributes 12 to 28. This shows the relationships between the effect, logic type, functionality, activation, and physical layout attributes. For example, $\mathbf{R_3}(14, 19) \equiv \mathbf{R_3}$(Reduced Reliability, Parametric) = 1 indicates that if a trojan reduces chip reliability (i.e. quicker battery drain), it can lead to changes in the power consumption, thermal and delay profiles.

$\mathbf{R_3} =$

A	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
12	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1
14	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1
15	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1
16	1	0	0	1	0	0	1	0	0	1	1	1	0	1	1	1	1
17	1	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1
18	1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1
19	0	1	1	0	0	0	0	0	1	0	0	0	1	0	1	0	1
20	1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1
21	1	0	0	1	1	1	1	0	0	0	0	1	1	0	1	1	1
22	1	1	0	1	1	1	1	0	0	0	0	0	1	0	1	1	0
23	1	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1
24	1	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	0
25	1	0	0	1	1	1	1	0	1	0	0	1	0	0	1	1	0
26	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
27	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0
28	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0	0

F. Submatrix $\mathbf{R_{23}}$

Submatrix $\mathbf{R_{23}}$ is the transfer matrix that describes the connections between $\mathbf{R_2}$ and $\mathbf{R_3}$. Thus it shows the connections between the abstraction attributes and the trojan property attributes.

$\mathbf{R_{23}} =$

A	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
6	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0
7	1	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0
8	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0
10	1	0	1	0	0	1	1	1	1	0	0	0	1	0	1	1	0
11	1	1	1	0	0	0	1	1	1	0	0	1	1	1	1	1	1

1) Single Element Example: $\mathbf{R_{23}}(7, 12) \equiv \mathbf{R_{23}}$(RTL, Change Function) = 1 indicates that if a trojan has been inserted in the RTL abstraction level, which changes the function description, it can also change the function behaviour.

2) Row Example: For row = 7, $\mathbf{R_{23}}(7, j) \equiv \mathbf{R_{23}}(\text{RTL}, j) = 1$: $\mathbf{R_{23}}(7, 12) \equiv \mathbf{R_{23}}$(RTL, Change Function), $\mathbf{R_{23}}(7, 15) \equiv \mathbf{R_{23}}$(RTL, DoS), $\mathbf{R_{23}}(7, 16) \equiv \mathbf{R_{23}}$(RTL, Sequential), $\mathbf{R_{23}}(7, 17) \equiv \mathbf{R_{23}}$(RTL, Combinational), $\mathbf{R_{23}}(7, 18) \equiv \mathbf{R_{23}}$(RTL, Functional), $\mathbf{R_{23}}(7, 20) \equiv \mathbf{R_{23}}$(RTL, Always On), $\mathbf{R_{23}}(7, 21) \equiv \mathbf{R_{23}}$(RTL, Internally Triggered), $\mathbf{R_{23}}(7, 22) \equiv \mathbf{R_{23}}$(RTL, Externally Triggered), $\mathbf{R_{23}}(7, 23) \equiv \mathbf{R_{23}}$(RTL, Big), and $\mathbf{R_{23}}(7, 24) \equiv \mathbf{R_{23}}$(RTL, Small) = 1, gives the trojan property attributes that may result if a trojan is inserted in the RTL abstraction level.

3) Column Example: For column = 14, $\mathbf{R_{23}}(i, 14) \equiv \mathbf{R_{23}}$(i, Reduced Reliability) = 1: $\mathbf{R_{23}}(10, 14) \equiv \mathbf{R_{23}}$(Transistor, Reduce Reliability), and $\mathbf{R_{23}}(11, 14) \equiv \mathbf{R_{23}}$(Physical, Reduce Reliability) = 1 indicates that if a trojan reduces the reliability of a chip, it may be because of a modification in the transistor and/or physical abstraction levels.

4) Fan In and Fan Out Example: Table I shows the fan in and fan out for each attribute in $\mathbf{R_{23}}$. The maximum fan out is 14 for development environment (attribute 8). Thus if a CAD tool is exploited by an attacker, there can be many chip vulnerabilities. The maximum fan in is 6 for change functionality, functional, and always on (attributes 12, 18, and 20, respectively). This means that many trojans will have these attributes.

Attribute	F_{in}	F_{out}	Attribute	F_{in}	F_{out}
6	-	6	18	6	-
7	-	10	19	3	-
8	-	14	20	6	-
9	-	8	21	3	-
10	-	9	22	3	-
11	-	12	23	3	-
12	6	-	24	4	-
13	2	-	25	2	-
14	2	-	26	3	-
15	4	-	27	3	-
16	3	-	28	2	-
17	4	-	-	-	-

TABLE I: Matrix $\mathbf{R_{23}}$ Attributes Fan In and Fan Out

G. Submatrix $\mathbf{R_{34}}$

Submatrix $\mathbf{R_{34}}$ describes the connections between $\mathbf{R_3}$ and the trojan location category. Thus it shows the connections between the trojan properties and the hardware locations. For example, $\mathbf{R_{34}}(15, 29) \equiv \mathbf{R_{34}}$(DoS, Processor) = 1 indicates that if a denial of service trojan has been inserted, it may be located in the processor to prevent the chip from executing properly. It is clear from this matrix that all locations are almost equally vulnerable to trojans.

$\mathbf{R_{34}} =$

A	29	30	31	32	33
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	0	1	1	1
16	1	1	1	1	1
17	1	1	1	1	1
18	1	1	1	1	1
19	0	0	1	1	1
20	1	1	1	1	1
21	1	1	1	1	1
22	1	1	1	1	1
23	1	1	0	0	0
24	1	1	1	1	1
25	1	1	1	1	1
26	1	1	1	1	1
27	1	1	1	1	1
28	1	1	1	1	1

H. Matrix $\mathbf{R}$

Matrix $\mathbf{R}$ represents the complete set of attributes in Figure 2, and the hardware trojan levels in Figure 3. The characteristics of a trojan can be inferred from the paths in $\mathbf{R}$. Note that an empty submatrix indicates that all entries are zero. Submatrix $\mathbf{R_4} = 0$ since the location attributes are terminal attributes and are not interconnected.

1) Single Element Example: $\mathbf{R}(17, 29) \equiv \mathbf{R}$(Combinational, Processor) = 1 indicates that if there is a trojan inserted in a processor, it may be a combinational circuit.

2) Row Example: For row = 6, $\mathbf{R}(6, j) \equiv \mathbf{R}$(System, j) = 1: $\mathbf{R}(6, 7) \equiv \mathbf{R}$(System, RTL), $\mathbf{R}(6, 12) \equiv \mathbf{R}$(System, Change Function), $\mathbf{R}(6, 13) \equiv \mathbf{R}$(System, Leak Information), $\mathbf{R}(6, 15) \equiv \mathbf{R}$(System, DoS), $\mathbf{R}(6, 18) \equiv \mathbf{R}$(System, Functional), $\mathbf{R}(6, 19) \equiv \mathbf{R}$(System, Parametric), and $\mathbf{R}(6, 20) \equiv \mathbf{R}$(System, Always On) = 1, which implies that if a trojan is introduced at the system abstraction level, it may lead to a combination of RTL, changed function, leaked information, denial-of-service, functional, parametric, or always on.

3) Column Example: For column = 6, $\mathbf{R}(i, 6) \equiv \mathbf{R}$(i, System) = 1: $\mathbf{R}(1, 6) \equiv \mathbf{R}$(Specification, System), $\mathbf{R}(4, 6) \equiv \mathbf{R}$(Testing, System), and $\mathbf{R}(5, 6) \equiv \mathbf{R}$(Assembly, System) = 1, which implies that if a trojan is inserted at the system abstraction level, it may be because of an incomplete or modified specification, control of the testing to evade trojan detection, or modification in the assembly phase.

4) Root Attribute Example: From (6), specification is a root attribute of any trojan as it is at the start of the chip life-cycle. Thus it does not depend on any other attributes.

5) Terminal Attribute Example: From (7), the location attributes are terminal attributes since they do not lead to other attributes.

6) Fan In and Fan Out Example: Table II shows the fan in and fan out for the attributes in $\mathbf{R}$. The maximum fan in of 19 occurs for functional, always on, and augmented (attributes 18, 20, and 26, respectively). This means that these three attributes occur most frequently due to other attributes. Adding an inverter gate to a chip to insert a trojan is a simple example which combines all of these attributes. The maximum fan out of 21 occurs only for augmented (attribute 26). This means that this attribute most frequently leads to other attributes.

Table II, shows the sums of the fan in and fan out for the attributes in $\mathbf{R}$. These values reflect the connectivity of an attribute with other attributes. A critical attribute is an attribute with the highest sum, since it has the greatest number of connections with other attributes. For the analysis presented here, augmented (attribute 26) is the critical attribute. Thus inserting a trojan without modifying the chip layout makes detecting it a very difficult task.

V. Case Studies

In this section, two examples of hardware trojan classification are presented based on the algebraic approach in Section IV.

A. Combinational Trojan

Assume that an attacker works as a design engineer in a chip manufacturing facility. He decides to use the trojan shown in Figure 4 to change the functionality of a chip when a particular event occurs. This is a combinational logic triggered trojan where $A = B = 0$ causes the output C to have an incorrect value (C modified). This circuit reflects an attacker inserting a trojan based on a rare event which is unlikely to be detected during the testing phase. From the figure, the trojan characteristics are change in functionality, combinational, functional, internally triggered, small, clustered, and augmented (attributes 12, 17, 18, 21, 24, 26, and 27 in $\mathbf{R_3}$). Examining the corresponding columns in $\mathbf{R_{23}}$, these attributes are all directly connected to the development environment in the abstraction category (attribute 8).

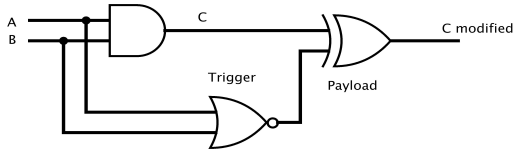


Fig. 4: A combinational logic triggered trojan [27].

From $\mathbf{R_{12}}$, there is no direct connection between attribute 8 and an attribute in the insertion category (attributes 1 to 5). However, $\mathbf{R_2}$ provides a connection between the development environment (attribute 8) and RTL (attribute 7). Then attribute 7 is connected to design (attribute 2) and system (attribute 6). Attribute 6 is connected to specification, testing, and assembly (attributes 1, 4, and 5). In addition, $\mathbf{R_1}$ provides connections between the attributes in the insertion category. $\mathbf{R1}$ shows all candidates paths that can be used to embed this trojan. In $\mathbf{R1}$, the light blue colour shows the horizontal connections, the light red colour shows the vertical connections, and the attribute connections are shown in dark red. Now assume that attributes 12, 17, 18, 21, 24, 26, and 27 have been identified for the trojan. These attributes all lie in $\mathbf{R_3}$. Moving forward (increasing attribute numbers), the expected trojan locations can be identified, and moving backward (decreasing attribute numbers), the untrusted attributes in the insertion and abstraction phases can be identified. It is clear that the row attributes (i.e. 12, 17, 18, 21, 24, 26, and 27), share can lead to all trojan locations (columns 29, 30, 31, 32, and 33). In $\mathbf{R1}$, attributes 12, 17, 18, 21, 24, 26, and 27 all have value 1 in row 8 only. Thus a trojan with these seven attributes should be inserted using attribute 8 (development environment) in the abstraction phase. $\mathbf{R_{12}}$ can then be used to reach the life-cycle phase. Then using $\mathbf{R_1}$, the complete paths from the insertion phase to the location phase. These paths are represented by the directed graph in Figure 5, which provides a complete characterization of the hardware trojan in Figure 4. In this figure, S indicates a possible trojan insertion point.

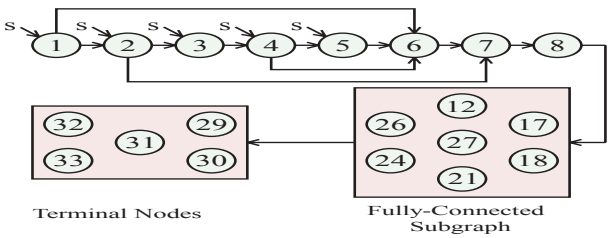


Fig. 5: A directed graph characterization of the hardware trojan in Figure 4.

An attacker will choose a particular point in the chip life-cycle for insertion. Assuming the design phase is selected, the graph shown in Figure 6 represents the inserted trojan. Conversely, a defender has no information about the attacker decisions, so it necessary to consider all paths in Figure 5.

Attribute	F_{in}	F_{out}	$F_{in}+F_{out}$	Attribute	F_{in}	F_{out}	$F_{in}+F_{out}$	Attribute	F_{in}	F_{out}	$F_{in}+F_{out}$
1	0	3	3	12	18	17	35	23	14	13	27
2	1	2	3	13	10	13	23	24	16	17	33
3	1	2	3	14	8	11	19	25	11	14	25
4	1	3	4	15	16	16	32	26	19	21	40
5	1	1	2	16	13	15	28	27	17	19	36
6	3	7	10	17	17	18	35	28	14	17	31
7	2	11	13	18	19	18	37	29	16	0	16
8	1	15	16	19	9	9	18	30	15	0	15
9	2	9	11	20	19	18	37	31	16	0	16
10	1	10	11	21	13	15	28	32	16	0	16
11	2	12	14	22	12	14	26	33	16	0	16

TABLE II: Sum of the Fan In and Fan Out for the Attributes in $\mathbf{R}$

However, if the specification and assembly phases are trusted, then the graph in Figure 5 will reduce to the graph in Figure 7. Then if a defender is able to verify that attribute 6 is not a trojan characteristic, the trojan in Figure 6 is identified exactly.

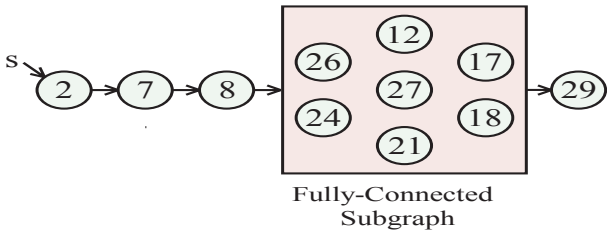


Fig. 6: The trojan graph assuming it is inserted in the design phase.

According to this discussion, a defender can detect such a trojan using hardware trojan detection techniques [28], [29], and [30].

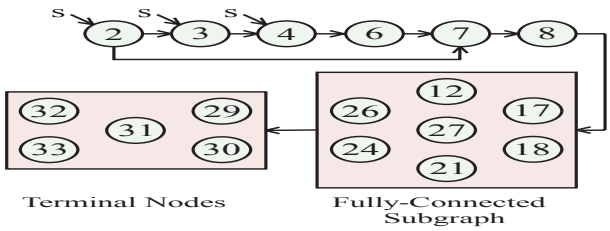


Fig. 7: The defender graph assuming the trojan is inserted by the attacker during the chip life cycle.

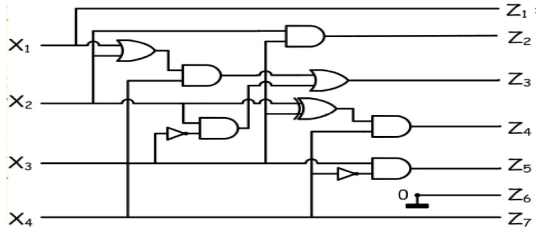
B. Backdoor Trojan

Consider a chip manufacturer that designs a circuit to compute a function $F(X)$ for a system to authenticate user-password pairs X and $F(X)$. The design specifies ten users I_0 to I_9 with $F(X) = X^2$. Since there are ten users, the designer uses four bits, X_1 to X_4, to encode them. Since $F(X) = X^2$, the largest function output is 81, so seven bits are required, Z_1 to Z_7. The resulting circuit is shown in Figure 8a. Testing

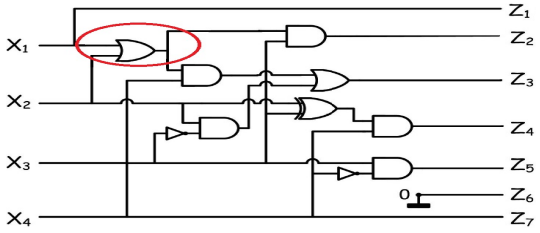
$\mathbf{R} =$

A	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
1	0	1	0	0	0	1	0	0	0	0	0																						
2	0	0	1	0	0	0	1	0	0	0	0																						
3	0	0	0	1	0	0	0	0	0	0	1																						
4	0	0	0	0	1	1	0	0	1	0	0																						
5	0	0	0	0	0	1	0	0	0	0	0																						
6						0	1	0	0	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0					
7						0	0	1	0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0					
8						0	0	0	1	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1					
9						0	0	0	0	1	0	1	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0					
10						0	0	0	0	0	1	1	0	1	0	0	1	1	1	1	0	0	0	1	0	1	1	0					
11						0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	1	1	1	1					
12												0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13												0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1
14												0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1
15												0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1
16												1	0	0	1	0	0	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1
17												1	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18												1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19												0	1	1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	1	1	1
20												1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
21												1	0	0	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1
22												1	1	0	1	1	1	1	0	0	0	0	1	0	1	0	1	1	0	1	1	1	1
23												1	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	0	0	0
24												1	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	0	1	1	1	1	1
25												1	0	0	1	1	1	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1
26												1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
27												1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1
28												1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	1
29																																	
30																																	
31																																	
32																																	
33																																	

using these ten input values shows that the outputs are correct and so the function works properly.



(a) Trojan free circuit



(b) Circuit with a backdoor trojan

Fig. 8: The circuits with and without the trojan.

With four bits, there are 16 input combinations, but in this case the six inputs 10 to 15 are not used so they are don't care conditions. With this incomplete specification, the designer asserts that the circuit performs the function $F(X) = X^2$ correctly. If an attacker makes the small modification indicated in Figure 8b, the output for the ten valid inputs is the same as with the original circuit. However, it also provides two additional correct outputs for inputs 10 and 11. Therefore, the modified circuit contains a backdoor trojan.

Table III shows the outputs from both circuits for all possible inputs. With an input of 10, circuit (a) produces 68, which is not the square of 10, while an input of 11 produces 89. Thus, circuit (a) provides outputs for the don't care conditions which are not the correct values for $F(X)$. However, circuit (b) outputs the correct values of $F(X)$ for inputs 10 and 11, so the user-password pairs $(10, 100)$ and $(11, 121)$ are valid. Thus, circuit (b) contains a back door trojan. It can be used to allow users who do not have permission to access the system. Note that both circuits have the same type and number of gates, the only difference is a wire connection. Further, the layouts are almost identical.

Assuming the trojan in Figure 8b is used to gain control of the system to create a denial of service attack, the characteristics are DoS, combinational logic, functional, and externally triggered. (attributes 15, 17, 18, and 22 in $\mathbf{R_3}$). Examining the corresponding columns in $\mathbf{R_{23}}$, this combination of attributes can occur if the trojan is inserted at the RTL, development environment, or logic type in the abstraction phase (attributes 7, 8, or 9). If this attack occurred due to an incomplete specification, the attacker needs to control the testing so that the trojan is not discovered. $\mathbf{R_2}$ provides a connection between logic type (attribute 9) and development environment (attribute 8), and between development environment (attribute 8)and RTL (attribute 7).

There are two steps the attacker takes. First, the trojan is injected during the design phase, and second, the testing phase is controlled to evade detection during testing. From $\mathbf{R_{12}}$, attribute 7 is connected to design (attribute 2) system (attribute 6), and attribute 6 is connected to specification, testing, and assembly (attributes 1, 4, and 5). Attribute 9 is directly connected to testing (attribute 4), as the trojan must evade detection. Incomplete specification as attribute 1 leads to this trojan, so $\mathbf{R_1}$ provides the connection between design (attribute 2) and specification (attribute 1). These paths are

$\mathbf{R1} =$

A	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
1	0	1	0	0	0	1	0	0	0	0	0																						
2	0	0	1	0	0	0	1	0	0	0	0																						
3	0	0	0	1	0	0	0	0	0	0	1																						
4	0	0	0	0	1	1	0	0	1	0	0																						
5	0	0	0	0	0	1	0	0	0	0	0																						
6						0	1	0	0	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0					
7						0	0	1	0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0					
8						0	0	0	1	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1					
9						0	0	0	0	1	0	1	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0					
10						0	0	0	0	0	1	1	0	1	0	0	1	1	1	1	0	0	0	1	0	1	1	0					
11						0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	1	1	1	1					
12												0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13												0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1
14												0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1
15												0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1
16												1	0	0	1	0	0	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1
17												1	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18												1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19												0	1	1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	1	1	1
20												1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
21												1	0	0	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1
22												1	1	0	1	1	1	1	0	0	0	0	0	1	0	1	1	0	1	1	1	1	1
23												1	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	0	0	0
24												1	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	0	1	1	1	1	1
25												1	0	0	1	1	1	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1
26												1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
27												1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1
28												1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	1
29																																	
30																																	
31																																	
32																																	
33																																	

TABLE III: The Outputs for the Circuits in Figure 8

		Inputs					Circuit (a)								Circuit (b)							
		X_1	X_2	X_3	X_4	X	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	$F(X)$	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	$F(X)$
Inputs	I_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	I_1	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
	I_2	0	0	1	0	2	0	0	0	0	1	0	0	4	0	0	0	0	1	0	0	4
	I_3	0	0	1	1	3	0	0	0	1	0	0	1	9	0	0	0	1	0	0	1	9
	I_4	0	1	0	0	4	0	0	1	0	0	0	0	16	0	0	1	0	0	0	0	16
	I_5	0	1	0	1	5	0	0	1	1	0	0	1	25	0	0	1	1	0	0	1	25
	I_6	0	1	1	0	6	0	1	0	0	1	0	0	36	0	1	0	0	1	0	0	36
	I_7	0	1	1	1	7	0	1	1	0	0	0	1	49	0	1	1	0	0	0	1	49
	I_8	1	0	0	0	8	1	0	0	0	0	0	0	64	1	0	0	0	0	0	0	64
	I_9	1	0	0	1	9	1	0	1	0	0	0	1	81	1	0	1	0	0	0	1	81
Undefined	I_{10}	1	0	1	0	10	1	0	0	0	1	0	0	68	1	1	0	0	1	0	0	100
	I_{11}	1	0	1	1	11	1	0	1	1	0	0	1	89	1	1	1	1	0	0	1	121
	I_{12}	1	1	0	0	12	1	1	1	0	0	0	0	112	1	0	1	0	0	0	0	80
	I_{13}	1	1	0	1	13	1	1	1	1	0	0	1	121	1	0	1	1	0	0	1	89
	I_{14}	1	1	1	0	14	1	1	0	0	1	0	0	100	1	1	0	0	1	0	0	100
	I_{15}	1	1	1	1	15	1	1	1	0	0	0	1	113	1	1	1	0	0	0	1	113

represented by the directed graph in Figure 9, which is a complete characterization of the hardware trojan in Figure 8b. In this figure, S indicates a possible trojan insertion point.

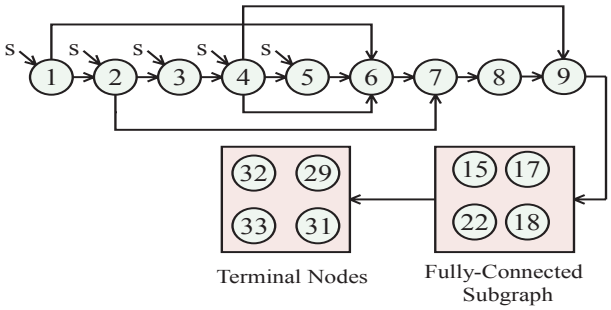


Fig. 9: A directed graph characterization of the hardware trojan in Figure 8b.

From an attacker respective, he will choose a particular point in the chip life-cycle for insertion. Assuming the design or testing phase is chosen, the graph shown in Figure 10 represents the trojan. Conversely, a defender has no information about the attacker decisions, so it necessary to check all possible paths in Figure 9. According to this discussion,

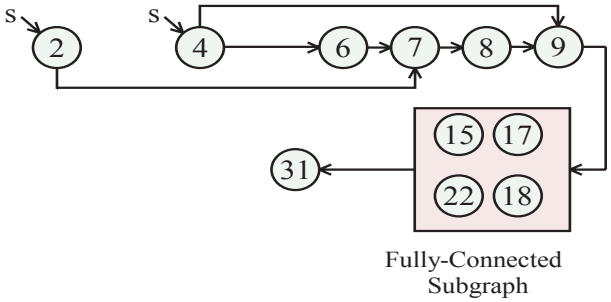


Fig. 10: The trojan graph assuming it is inserted by the attacker during the design or testing phase.

a defender can detect such a trojan using hardware trojan detection techniques [31].

VI. Conclusion

Hardware trojans are a growing concern in modern integrated circuit (IC) development and production. This is particularly true for chips intended for critical infrastructure and critical applications. This paper considered the attributes of hardware trojans for every phase of the chip life-cycle. Previous results in the literature assume that some phases of this life-cycle can be trusted, while others may be untrusted. It was shown in this paper that any phase can be vulnerable to hardware trojan insertion. The attributes of hardware trojans were characterized, and a hardware trojan matrix was developed to show their connections. This matrix provides many insights into the characteristics of trojans, and thus is a valuable tool for their detection. The results presented here can be used to study hardware trojan and their attributes, and determine the untrusted phases in the chip manufacturing process.

References

[1] "Anonymous fighter of army b."
[2] A. experience and announced later related problem in CNN, "http://money.cnn.com/2015/02/04/autos/toyota-camry-lawsuit/."
[3] S. Adee, "The hunt for the kill switch," *IEEE Spectrum*, pp. 34–39, 2008.
[4] D. S. Board, "Defense science board task force on high performance microchip supply."
[5] R. Karri, J. Rajendra, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. Computer 43.10, no. 10, pp. 39–46, 2010.
[6] M. T. RM. Rad, X. Wang and J. Plusquellic, "Power supply signal calibration techniques for improving detection resolution to hardware trojans," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 632–639, 2008.
[7] F. Wolff, C. Papachristou, S. Bhunia, and R. Chakraborty, "Towards trojan-free trusted ics: Problem analysis and detection scheme," *In Design, Automation and Test in Europe,DATE'08, IEEE*, pp. 1362–1365, 2008.
[8] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," *Hardware-Oriented Security and Trust*, pp. 15–19, 2008.
[9] S. Moein, F. Gebali, and I. Traore, "Analysis of covert hardware attacks," *Journal of Convergence*, 2014.
[10] B. Sharkey, "Trust in integrated circuit program - briefing to industry," 2007. [Online]. Available: http://cryptocomb.org/DARPA%20-%20Trust%20in%20Integrated%20Circuits%20Program.pdf
[11] T. Zhou and T. Tarim, "An efficient and well-controlled ic system development flow: Design approved specification and design guided test plan," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. IEEE, 2005, pp. 2775–2778.

[12] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. Springer, 2013.
[13] S. Padmanabhan. Discover a better way to go from c-level to synthesis for soc designs. [Online]. Available: http://electronicdesign.com/technologies/discover-better-way-go-c-level-synthesis-soc-designs
[14] D. Collins, "Trust, a proposed plan for trusted integrated circuits," DTIC Document, Tech. Rep., 2006.
[15] STARCHIPÂ®. Product development - starchipâ® development flow. [Online]. Available: http://www.starchip-ic.com/Company/Product_DP.html
[16] K. Goertzel, "Integrated circuit security threats and hardware assurance countermeasures," *The Journal of Defense Software Engineering*, pp. 33–38, 2013.
[17] M. Banga and M. Hsiao, "A region based approach for the identification of hardware trojans," *Hardware-Oriented Security and Trust, 2008*, 2008.
[18] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design and Test*, 2010.
[19] A. Iqbal. Security threats in integrated circuits. [Online]. Available: https://sdm.mit.edu/news/news_articles/iqbal-threats-solutions-ic-security-threats/iqbal-threats-solutions-ic-security-threats.html
[20] J. Roy, F. Koushanfar, and I. Markov, "Extended abstract: Circuit cad tools as a security threat," *Proc. IEEE Workshop Hardware-Oriented Security and Trust (HOST 08)*, pp. 65–66, 2008.
[21] N. Lin, S. Li, J. Chen, P. Wei, and Z. Zhao, "The influence on sensitivity of hardware trojans detection by test vector," in *Communications Security Conference*, 2014, pp. 22–24.
[22] S. Narasimhan, X. Wang, D. Dongdong, R.Chakraborty, and S. Bhunia, "Tesr: A robust temporal self-referencing approach for hardware trojan detection," *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2011.
[23] G. Becker, A. Lakshminarasimhan, L. Lin, S. Srivathsa, V. Suresh, and W. Burelson, "Implementing hardware trojans: Experiences from a hardware trojan challenge," in *Computer Design (ICCD)*, 2011.
[24] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric trojans for fault-injection attacks on cryptographic hardwar," *Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2014.
[25] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic, "Hardware trojan detection and isolation using current integration and localized current analysis," *Defect and Fault Tolerance of VLSI Systems*, pp. 87–95, 2008.
[26] W. Danesh, J. Dofe, and Q. Yu, "Efficient hardware trojan detection with differential cascade voltage switch logic," *VLSI Design*, 2014.
[27] R. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," *IEEE International in High Level Design Validation and Test Workshop*, pp. 166–171, 2009.
[28] Y. Liu, K. Huang, and Y. Makris, "Hardware trojan detection through golden chip-free statistical side-channel fingerprinting," in *Proceedings of the 51st Annual Design Automation Conference*. ACM, 2014, pp. 1–6.
[29] Y. Liu, Y. Jin, and Y. Makris, "Hardware trojans in wireless cryptographic ics: silicon demonstration & detection method evaluation," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2013, pp. 399–404.
[30] X. Mingfu, H. Aiqun, and L. Guyue, "Detecting hardware trojan through heuristic partition and activity driven test pattern generation," *IET*, 2014.
[31] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware trojan detection," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2013, pp. 532–539.

Samer Moein received the B.Sc. degree in computer engineering from Kuwait University, Kuwait, M.Sc. degree in computer engineering from Kuwait University, Kuwait. In 2013 he received the University of Victoria Fellowship and started his PhD in the Department of Electrical and Computer Engineering of the University of Victoria, Canada. His research interests include hardware security, encryption algorithms, encryption processors, and networks-on-chip.

T. Aaron Gulliver received the Ph.D. degree in Electrical Engineering from the University of Victoria, Victoria, BC, Canada in 1989. From 1989 to 1991 he was employed as a Defence Scientist at Defence Research Establishment Ottawa, Ottawa, ON, Canada. He joined the University of Victoria in 1999 and is a Professor in the Department of Electrical and Computer Engineering. In 2002, he became a Fellow of the Engineering Institute of Canada. In 2012, he was elected Fellow of the Canadian Academy of Engineering. From 2000 to 2003 he was Secretary and a member of the Board of Governors of the IEEE Information Theory Society. He is currently an Area Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. His research interests include information theory and communication theory, algebraic coding theory, OFDM systems, smart grid and ultra-wideband communications.

Fayez Gebali received his B.Sc. in Electrical Engineering (first class honors) from Cairo University, his B.Sc. in Mathematics (first class honors) from Ain Shams University, and his Ph.D. degree in Electrical Engineering form the University of British Columbia where he was a holder of an NSERC postgraduate scholarship. Dr. Gebali is a Professor and Chair of the Department of Electrical and Computer Engineering at the University of Victoria. His research interests include parallel algorithms, networks-on-chip, three-dimensional integrated circuits, digital communications, and computer arithmetic.