Automated Hardware Trojan Detection in FPGAs

Nicholas Houghton, Samer Moein, and Fayez Gebali
Department of Electrical and Computer Engineering
University of Victoria
P.O. Box 1700 STN CSC
Victoria, B.C. V8W 2Y2
Email: {nhoughto, samerm, fayez}@uvic.ca

***Abstract*—The abstract goes here.**

I. Introduction

The term *Trojan Horse* or *Trojan* has become a modern metaphor for a deception where by an unsuspecting victim welcomes a foe into an otherwise safe environment [1]. Since the dawn of the computer we have dealt with software threats. We are almost as good at protecting ourselves against them as attackers are at making them. In recent years a new incarnation of electronic danger has emerged; in hardware. In this new arena of attack and defend those who seek to defend are far behind.

IC designs for Field Programmable Gate-Arrays (FPGAs) are made using a software language known as an Hardware Description Language (HDL). The design is then converted to a binary file called a configuration Bitstream which is then downloaded onto the device; this process is known as synthesizing the design. There have been many attempts to develop mechanisms and techniques to determine whether a malicious user has tampered with the design via test vectoring or side-channel analysis. As of yet there has been little effort to directly analyze the configuration Bitstream.

A method of extracting and analyzing the configuration Bitstream to determine the presence of hardware trojans has been developed. This method is able to meaningfully read the long binary file and extract modifications. Any discovered changes are located on the device using a new technique referred to as 'Component Mapping'. Further, these changes are then mapped to the user's original design. Knowing which components of the device have been modified, and the instances of the synthesized design allows for a powerful description to be built. A software tool known as FPGA Trojan Detector which implements this new method has been built. FPGA Trojan Detector is able to automatically detect and analyze trojans in FPGAs. Once analyzed a meaningful description is provided using the trojan taxonomy presented in [2].

The contributions of this paper are:

1) A new method mapping configuration Bitstream words to device components named 'Component Mapping'.
2) A systematic process of detecting and analyzing hardware trojans in FPGAs
3) A software tool which automates these new methods.

II. Methodology

Figure 1 provides a visual representation of the use-case assumed for the purposes of this work. With the exception of the fabrication process, all stages of production of an FPGA implementation are assumed to have been done "in-house". Any trojan discovered is inserted in the fabrication phase; all other stages are trusted. The method of automated trojan detection described in this work would take place in the 'testing' phase of the life-cycle. Figure 2 shows an overview

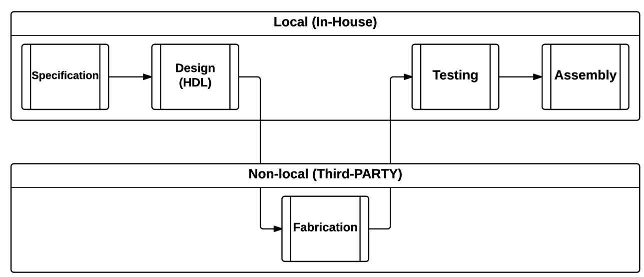

Fig. 1: FPGA Life-Cycle

of the trojan detection methodology. As mentioned, FPGA designs are written in a Hardware Description Language (HDL). *Xilinx* provides a series of User-Interface (UI) and command line tools to process the HDL known as the 'tool-chain'. The tool chain generates a series of files that are used for a variety of purposes as shown in the 'Resultant Files' box in Figure 2. The NGC file is a non-human readable semantic description of the design known as a netlist. This file can be converted into a human-readable version known as *Xilinx* Design Language (XDL) which will be described in section **??**. The Bit file is the binary representation of the design to be implemented. It is referred to as the Bitstream or 'configuration' Bitstream and is the final form that is loaded into the FPGA. This Bit file is the primary file sent to the fabrication house where it will be implemented onto the batch of devices ordered. The resultant files, produced 'in-house' are to be kept in secure storage while a copy is sent to be fabricated; these stored copies are referred to as Golden and assumed to be trojan-free. Though it is known that the fabrication houses will often attempt to make optimizations on designs, this methodology requires that no such efforts be made. When the completed batch of fabricated chips are returned the Bitstream is extracted from a sample using the *Xilinx* feature Readback. That which is extracted is referred to as the Target Bitstream. The Golden and Target Bitstreams are analyzed in conjunction to detect

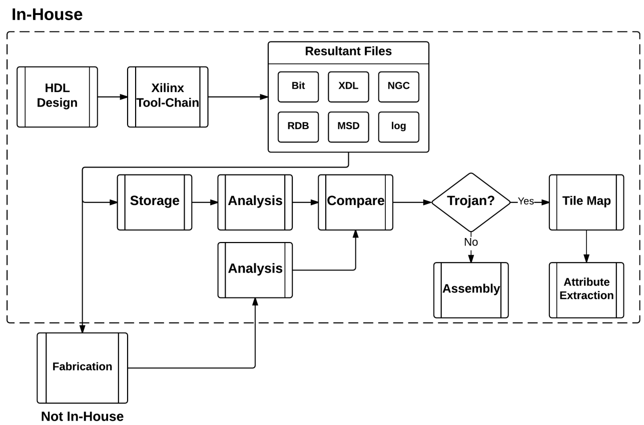

Fig. 2: Methodology Overview

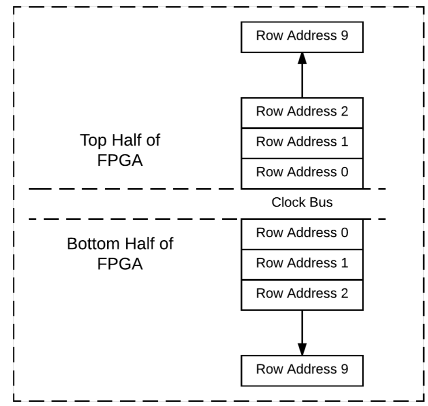

Fig. 3: Row Order of Virtex-5 Clock Region

differences. Any discovered differences are then attributed to the corresponding component in the architecture, described in section II-B. Finally, the resultant taxonomic description is returned to the user.

A. The FPGA Bitstream Analysis

The *Xilinx* Bitstream is a binary file composed of a series of 32-bit words organized into 'frames'. A frame is a string of single bits that span from the top to the bottom of a clock region of a device as seen in the top-right quadrant of Figure **??**. A frame affects every block in a column and multiple horizontally adjacent frames are required to configure an entire column. Each frame is uniquely identified by a 32-bit address and is the smallest addressable element. The composition of the frame address is fairly consistent across the *Xilinx* catalog however there are small differences between device families. The following is the structure of the Virtex-5 family frame address scheme according to [3]. The make-up of a frame address is shown in Table I.

The Block Address (BA) identifies the block type.

- BA 0: Logic type.
- BA 1: Block RAM (BRAM).
- BA 2: BRAM Interconnect.
- BA 3: BRAM non-configuration frame.

The logic block contains the columns which provides the primary configuration for the device (CLBs, IOBs... etc). The BRAM columns initialize the memory for the device while the BRAM Interconnect columns configure how the logic of the design interacts with the BRAM.

Each clock region is given a row value in its address that increments away from the center of the device starting at 0. The frame address includes a Top indicator bit in position 20 that indicates whether the specified row is above or below the center of the device [3]. The major address specifies the column within the row. These addresses are numbered from left to right and begin at 0. The minor address indicates the frame number within a column. Table II provides the number of frames per column type. A block may contain multiple tiles. In a CLB column a block consists of an interconnect tile, also

TABLE II: Number of Frames (minor addresses) per Column [3]

Block	Number Of Frames
CLB	36
DSP	28
BRAM	30
IOB	54
Clock	4

known as a Switching Matrix (SM) and a CLB. Frames are numbered from left to right, starting with 0. For each block, except in a clock column, frames numbered 0 to 25 access the interconnect tile for that column. For all blocks, except the CLB and the clock column, frames numbered 26 and 27 access the Interface for that column. All other frames are specific to that block [3]. To further understand how frames configure tiles a mapping must be made between each frame and the corresponding tile. This is described in section II-B.

B. Component Mapping

The FPGA Trojan Detector employs a method referred to as Component Mapping to create a mapping between each word in a configuration frame and the component on the device that it configures. This information is not publicly released by *Xilinx* as a means of providing security through obscurity.

1) Frame to Column Mapping: The configuration Bitstream is stored in an external memory device as described in section **??**. When powered-on the Bitstream is transmitted in frame address order to populate the dynamic memory in the tiles of the gate-array. The frame addressing scheme describes where in the gate-array the frame is destined fairly directly. Frames with a BA value of 1 are clearly destined to configure the BRAM and do not need further analysis for the purposes of this method. Frames with a BA value of 0 or 2 must be mapped more finitely. The row address specifies which row of clock-regions the frame is destined. Figure **??** shows four clock regions organized into two rows.

As an example the Virtex-5 240T has 12 rows; its row address spans from 0-5 and the Top bit in the address indicates whether it is in the top or bottom half of the device in

TABLE I: Frame Address

Unused								BA			T	Row Address					Major Address								Minor Address						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0

accordance with Figure 3. Once the correct clock region is discerned the major address is used to determine which column the frame configures. The major address begins at 0 on the left and counts up towards the number of columns in the row. Finally, the minor address is used to determine which sub-column has been modified according to Table I.

2) Word to Block Mapping: In the case of Virtex-5 devices a frame is composed of 41 words that can be thought of as a vertical stack that aligns with a column. As described in section II-A a row consists of a stack of basic blocks; there are 20 CLB blocks per column, 40 IOBs, 4 BRAM...etc for Virtex-5 devices. As can be seen in Figure 4 the central

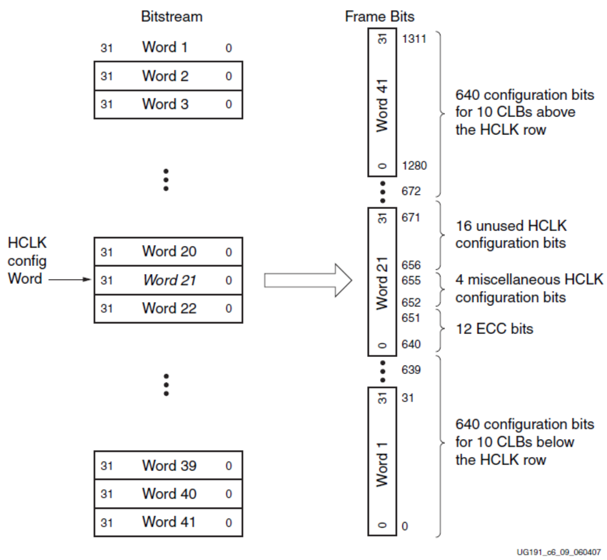

Fig. 4: Configuration Words in the Bitstream [3]

word in a frame configures the horizontally running clock bus. The remaining words are used to configure the blocks in the column. The purpose of the central word in the column is known to be mapped to the clock bus. For the purposes of the following computations it is considered removed from the frame. From this, equation 1 can be deduced which is used to compute the number of 32-bit words that span each block.

$$n = (W - C) + B \tag{1}$$

where:

n = Number of Words per Block
W = Number of 32-bit words per frame
C = Number of clock words per frame
B = Number of blocks per column

As shown in Figure 4 words are addressed from the 'top' of a device down. Equation 2 can be used to map a particular word in a frame to a block on the device.

$$i = B - \left\lfloor \frac{w}{n} \right\rfloor \tag{2}$$

where:

i = Word Number in frame
B = Number of blocks per column
w = Word number
n = Number of Words per Block

With equations 1 and 2 it is now possible attribute any modifications in the Bitstream to its corresponding block.

C. Determining Trojan Attributes

The complexity of Integrated Circuit designs and their corresponding trojans requires a more human-friendly scope. The taxonomy in [2] provides a series of thirty-three attributes which a trojan may or may not posses. Though it is desirable to be able to observe and directly extract the attributes a trojan possesses it is not always possible. In [2] a matrix, $\mathbf{R}$, is provided which describes the relationships between each of the thirty-three attributes in the taxonomy. When it is not possible to directly determine the presence of certain attributes, this relation matrix is used to infer their existence. The analysis stage of the automated trojan detection technique provided by this work begins by extracting those attributes that are directly observable then using matrix $\mathbf{R}$ to infer the existence of the remainder.

1) Observed Location Attributes: The presence of attributes in the *Location* category are directly observable from the results of the component mapping method described in section II-B. *Xilinx* tiles conform to purpose-specific groups or block types which were discussed in section II-A. These block types contain sub-types that perform actions which pertain to the *Location*, category.

1) The **Processor** attribute pertains to the core functionality of the design logic. It can be awarded for presence of a modified CLB tile or Interconnect tile.
2) The **Memory** attribute can be awarded for the presence of modified BRAM components.
3) The **IO** attribute can be awarded for presence of modified IOB tiles.
4) The **Power Supply** attribute can be awarded for the presence of modified interface or configuration tiles.
5) The **Clock Grid** attribute can be awarded for modified clock tiles.

2) Scatter Score Method: The gate-array configuration of components in *Xilinx* FPGAs allows for an analytical method of determining attributes in the *Physical Layout* category. The "Scatter Score" method uses the grid coordinates of components to derive a numerical score rating for the size, position, and augmentation of configured tiles. Tiles are assigned global coordinates that represent their horizontal and vertical positions within the gate array denoted x and y respectively. These values can then be used to strongly infer the presence of *Physical Location* attributes.

The golden chip is first analyzed. The set of all tiles which are configured in the golden design is found and a series of numerical descriptors are computed.

$$n = \sum_{x=0}^{X} \sum_{y=0}^{Y} T_{xy} \tag{3}$$

where:

n = Number of all **configured** tiles
X = The column width of the gate-array
Y = The number of rows of the gate-array
T = A configured tile

$$a_x = \frac{1}{n} \sum_{x=0}^{n} T_x \tag{4}$$

$$a_y = \frac{1}{n} \sum_{y=0}^{n} T_y \tag{5}$$

$$\sigma_x = \sqrt{\frac{1}{n} \sum_{x=0}^{X} (x_i - a_x)^2)} \tag{6}$$

$$\sigma_y = \sqrt{\frac{1}{n} \sum_{y=0}^{Y} (y_i - a_y)^2)} \tag{7}$$

where:

a_x = The average x coordinate of configured tiles
a_y = The average y coordinate of configured tiles
T_x = The x coordinate of a configured tile
T_y = The y coordinate of a configured tile
σ_x = The standard deviation of the x coordinate of configured tiles
σ_y = The standard deviation of the y coordinate of configured tiles

Equations 4 and 5 are used to create a rating known as the Position Median in Equation 8. The Position Median value provides a simple descriptor for where in the gate array the design is centralized. The Scatter Score in Equation 9 describes how spread out or, *clustered* the design is.

$$M_{xy} = (a_x,\ a_y) \tag{8}$$

$$S_{xy} = (\sigma_x,\ \sigma_y) \tag{9}$$

where:

M_{xy} = The Position Median
S_{xy} = The Scatter Score

The results of the component mapping method described in section 4 are used to generate the set of all tiles reconfigured by the trojan. The set of reconfigured tiles can be said to contain three subsets: the subset of tiles activated by the trojan, those deactivated and those modified. The results of the golden design analysis, the subsets, and the numeric descriptors can be used to discern which of the *Physical Location* attributes the trojan possesses. The *Physical Location* category contains six attributes. These six can be considered three pairs; a trojan exhibits one attribute from each pair.

1) **Large or Small** (attributes 23 or 24): According to [2], small trojans are defined as those that are nearly impossible to detect via power consumption. From this it can be said that 'small' trojans occupy minimal resources. Trojans where the number of reconfigured tiles is less than 5% of the number of tiles in the golden design are considered small. Other wise they are attributed as large.
2) **Changed Layout or Augmented** (attributes 25 or 26): A 'changed layout' trojan is such that only tiles that are configured by the golden design are reconfigured. An augmented trojan is where additional layout is added. The presence of 'activated' or 'deactivated' tiles indicates an augmented trojan.
3) **Clustered or Distributed** (attributes 27 or 28): The trojan is considered to be clustered when the standard deviation of the reconfigured tile positions is less than 15%; distributed otherwise.

3) Insertion and Abstraction Attributes: The linear nature of the manufacturing life-cycle implies a propagation of effects. For the purposes of this method it is assumed that the only non-trustworthy stage in the life-cycle is fabrication. In other words, the trojan was inserted in the third-party fabrication stage. Due to the propagating nature of the life-cycle the effects of the modifications made in the fabrication stage (attribute 3) are felt in the testing (attribute 4) and assembly (attribute 5) stages. Hence, it can be said that this trojan possesses insertion category attributes 3, 4 and 5. FPGA designs are made with a HDL. These languages dictate component arrangement in the Registry Transfer Level (RTL) abstraction level. Hence it can be said that trojans occurring in FPGAs take place in the System (attribute 6) and RTL (attribute 7).

4) Relation Matrix Use: Attributes which are not directly observable can be inferred using a systematic method of analyzing the rows and columns of the relation matrix presented in [2]. The FPGA Trojan Detector takes the attributes it is able to directly observe and uses them as input to this process. A thorough description of this method is given in [4] and [5].

III. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] Michael Wood. *"In search of the Trojan war*. 1st ed. The British Broadcasting Corporation, 1998. ISBN: 978-0-520-21599-3.
[2] S. Moein et al. "An attribute based classification of hardware trojans". In: *Computer Engineering Systems (ICCES), 2015 Tenth International Conference on*. 2015, pp. 351–356. DOI: 10.1109/ICCES.2015.7393074.
[3] *Virtex-5 FPGA Configuration User Guide*. v3.11. Xilinx. 2012.
[4] Samer Moein. "Systematic Analysis and Methodologies for Hardware Security". PhD thesis. 3800 Finnerty Rd, Victoria, BC, Canada V8P 5C2: University of Victoria, Aug. 2015.
[5] N. Houghton et al. "An Automated Web Tool for Hardware Trojan Classification". In: *ESC'16* 2 (July 2015).