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

A. Background

A *Xilinx* Field Programmable Gate-Array (FPGA) is comprised of a matrix of tiles. A device can contain anywhere from a couple hundred to a few thousand. Tiles are arranged into columns by their type. An FPGA may have over one-hundred different types of tiles however each column is comprised almost entirely of a single type. Columns are separated by clock regions. Each clock region is an independent array of tiles which uses a dedicated clock resource; this minimizes clock skew from timing delays. The functionality of each tile is dictated by a binary file referred to as a bitstream. When an FPGA is turned on the bitstream is loaded from an external memory into the device, implementing the user's design.

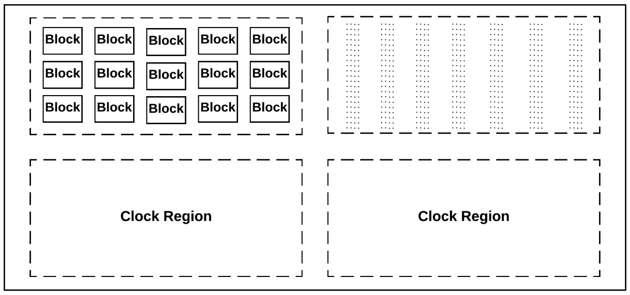

Fig. 1: Rudimentary Layout of a Virtex FPGA

The *Xilinx* bitstream is 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. In other words, each frame affects every tile in a column. Multiple adjacent frames are required to configure an entire column.

B. Implementation

A software tool has been developed to automate the detection and analysis of trojans. This tool was written in Java and employs the *Swing* toolkit to provide a simple-to-use User-Interface (UI). The *RapidSmith* Application Programming Interface (API) was developed at the Brigham Young University (BYU) in Utah. *RapidSmith* is a research-based, open source FPGA CAD tool written in Java. It uses the *Xilinx* Design Language (XDL) to generate a series of database files which describe the layout and architecture of modern FPGAs. These files are then references by a series of tools and methods that serve as a rapid prototyping platform for 'low-level' research ideas and algorithms. Employing *RapidSmith* has made the *Automated Trojan Detection System* much more efficient and universal.

II. Methodology

Before a design is sent to be fabricated the finalized design must be used to generate the 'golden' configuration bitstream via the *Xilinx* Synthesis Tools (XST). This file is to be preserved in order to be used as a reference during detection. After fabrication a 'target' device must be selected from the batch returned. This 'target' device must then have the configuration bitstream extracted from the SRAM. The 'golden' and 'target' bitstream will be compared to determine modification.

A. Bitstream Parsing and Analysis

At the beginning of a configuration file there are a series of instructions. These instructions are used to perform a variety of tasks to prepare the gate-array for configuration. Each configuration manual, example [1] for the Virtex-5 family, provides the definition of the instructions used during configuration. These instructions can be used to gain insight into the structure, size and format of configuration frames. Extracting details such as frame length, size, packet organization and more allows for the unnecessary information to be discarded and the frames to be organized and stored as a list. The 'golden' and 'target' configuration files are for the exact same device; in consequence the bitstreams have the same length and organization. The list of frames from the 'golden' and 'target' bit files are compared at the binary level. Frames containing a difference between the two files are stored for analysis.

B. Tile Mapping

Any bits which differ between the 'golden' and 'target' files result in undesired behavior. To understand the effect of such differences the affected component must be identified. *RapidSmith* provides a database file for every device in the Virtex-4, 5 and 6 families. Using the API the definition and organization of each tile, wire and primitive can be extracted. Configuration frames are loaded sequentially according to an addressing scheme. Each device is slightly different however addresses increment from left to right across a clock region.

C. Attribute Extraction

III. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] *Xilinx* , “Virtex-5 FPGA Configuration User Guide,” Version 3.11, Oct. 2012.