A Multi-Faceted Approach to FPGA-Based Trojan Circuit Detection

Michael Patterson, Aaron Mills, Ryan Scheel, Julie Tillman, Evan Dye, Joseph Zambreno
Electrical and Computer Engineering, Iowa State University, Ames, Iowa, USA
Email: {mjpatter, ajmills, rascheel, tillmanj, emdye, zambreno}@iastate.edu

***Abstract*—Three general approaches to detecting Trojans embedded in FPGA circuits were explored in the context of the 2012 CSAW Embededed Systems Challenge: functional testing, power analysis, and direct analysis of the bitfile. These tests were used to classify a set of 32 bitfiles which include Trojans of an unknown nature. The project is a step towards developing a framework for Trojan-detection which leverages the strengths of a variety of testing techniques.**

I. Motivation and Background

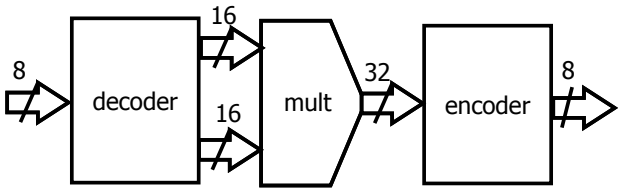
In recent years, the vulnerability of chips to hardware Trojans has garnered a great deal of attention [1], [2], [3], [4]. Many types of Trojans have been detected, and work has even been done to classify such Trojans [5]. It has been realized that malicious attackers can secretly insert Trojans into chips during the manufacturing process. This vulnerability is especially troublesome because of the degree to which chip manufacturing is outsourced to foreign countries, which may not share the interests of those ordering the manufacturing [6]. The customers send their designs off to be manufactured, but have no way of ensuring the chips they receive back contain only their original design and not a Trojan.

There are a few options to mitigate this vulnerability. Trusted foundries could be used during manufacturing to ensure a reliable final product. However, this is usually not economically feasible or desirable. Another option is to perform destructive testing on the final product to detect the presence of a Trojan, but there are some obvious drawbacks to this approach. It destroys the chip under test, it requires specialized and often expensive equipment, and it doesn't guarantee that the untested chips are also Trojan-free. What is more, the technique is not useful for FPGAs, since the end-application is not a physical construct on the chip. Non-destructive testing methods need not suffer from these limitations, and are therefore the focus of this paper.

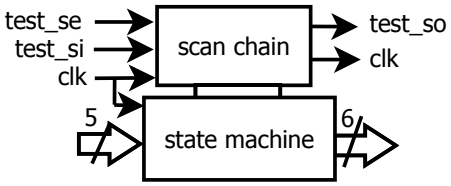
This paper discusses three testing methods we explored in order to detect the Trojans presented in the 2012 CSAW Embedded Systems Challenge. It details our testing setup as well as listing our classification for each of the provided FPGA bitfiles and our evidence for each classification. Our testing methods are compared to similar published methods, and various possibilities for future work are discussed.

II. Testing Techniques

The goal of the 2012 CSAW Embedded Systems Challenge was to correctly identify which among the 32 provided bitfiles contained Trojans. As shown in Figure 1, there were two circuits provided; the ISCAS85 benchmark `C6288` consisting of only combinational logic (circuit type A), and the ISCAS89 benchmark `S9234` which includes sequential logic (circuit type B) and a 147 flipflop scan chain. Only rudimentary documentation was



(a) Design A, the `C6288` ISCAS85 benchmark



(b) Design B, the `S9234` ISCAS85 benchmark

Fig. 1: Circuit designs for the 2012 CSAW Embedded Systems Challenge.

provided for these circuits, and nothing was known in advance of the Trojans they contained. Although numerous testing strategies were considered, we decided to focus on the following three. Our results indicate that these techniques can also be generalized to non-FPGA circuits. Bitfile analysis might appear to be the exception, but in general chip design flows will produce a netlist before fabrication, so the process is comparable.

A. Functional Testing

In functional testing, the correct output for each input over a given range is first calculated. Then, the chip being tested is given the same range of inputs, and the corresponding outputs are recorded. These output sets are compared, and if any inconsistencies are found, it can be concluded that the chip has a Trojan. However, even if the outputs are all the same, it cannot be definitively said that the chip does not have a Trojan, as the Trojan may not modify the functionality.

Functional testing is an effective way to test for Trojans, but it does have several limitations. It is most effective for combinational circuits. Sequential circuits have an intractably large input space, making exhaustive functional testing impossible. Additionally, the Trojan could have a trigger that activates only at a certain time or under other circumstances completely independent of input. It is also not useful for detecting Trojans that don't affect the output. Even for combinational circuits, functional testing becomes less effective as the possible number of inputs increases. For each extra bit of input, the time to completely test a combinational circuit increases by a factor of two, so testing the entire input space quickly becomes impractical. However, even with all these limitations, functional testing is often the best way to begin testing for Trojans. Under the correct circumstances, it can provide a relatively easy method to conclusively prove that a chip does have a Trojan.

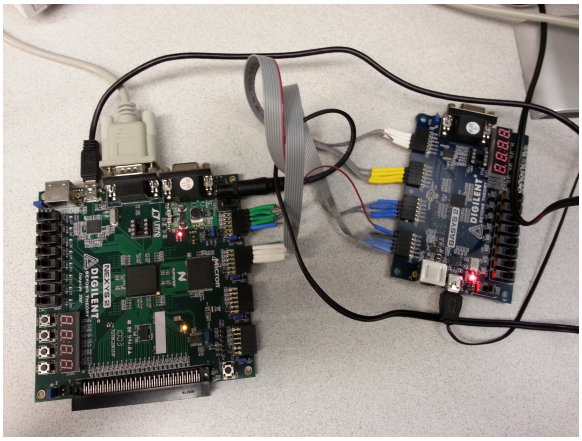


Fig. 2: Functional test setup using Digilent Nexsys2 and Basys2 FPGA boards.

B. Power Analysis

As a signal inside an integrated circuit switches between logical 1 and 0, the total power consumed by the circuit will increase and decrease in a correlated fashion–this is known as dynamic power. Thus, one way to indirectly observe changes inside a circuit is to measure the current draw.

This principle can be leveraged to detect whether or not a chip contains a Trojan [7], [8], [9]. First, the golden design is observed while operating, and the power characteristics of the chip are recorded. Next, each potentially infected chip is tested, and the power characteristics are compared to those originally measured. By analyzing these measurements, a Trojan can often be detected without directly observing the functionality of the Trojan itself.

However, accurate power analysis can be challenging. Many non-obvious factors influence the power consumption of a chip, and some Trojans can only be detected by this method when they are triggered. Both of these factors come into play when deciding how to configure a test, specifically what input to provide to the chip. Two general strategies are often used: static input and dynamic input. Static input testing, in which the same input is applied to the chip during the test, can quickly find Trojans which are always operating regardless of the input. However, dynamic testing, in which many different inputs are applied to the chip during testing, is often needed to detect Trojans that are only triggered by certain input conditions.

C. Bitfile Analysis

The third technique that we considered involved analyzing the bitfile itself. Even though vendors work to prevent the possibility of reverse engineering the bitfiles generated for their FPGAs, it has been shown that such a technique is possible [10], [11]. In fact, tools have even been developed that attempt to automate this process [12], [13], [14]. By analyzing the logic elements behind each bit file's generation, much can be learned about the actual components in each design, and whether or not a bit file contains a Trojan. While we did not primarily rely on this technique for the CSAW competition, it did provide some helpful validation of our other methods.

III. Experimentation

A. Functional Testing

For all of our functional testing, we used the setup shown in Figure 2. We used a Digilent Nexys2 board to send and receive signals through the "pmod" connectors on the Basys2 board. This physical setup was used for testing both design A and design B. The overall results are presented in Table I.

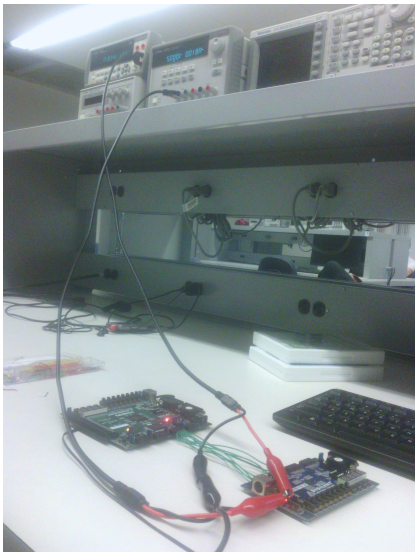


Fig. 3: Setup for circuit power analysis.

1) Design A: Design A was fairly easy to functionally test. The Design A input is only 8 bits, so the Nexys2 testbench only had to test 256 possible inputs to complete a functional test. The testbench simply went through all of these inputs while collecting the output from the Basys2. As each test was running, the testbench would send the output over a serial connection to a PC to be recorded. After we captured the output, we then ran a script to parse it and compare it to the expected golden circuit behavior. If there was a Trojan running on the board that caused any discrepancy between input and output, then this functional test would guarantee that we caught it.

2) Design B: Design B was much more challenging due to the large input range and the use of sequential logic. Furthermore, we were provided no information on the overall function, so we could not strategically pick our inputs. Again our testbench ran on the Nexys2 board while the bitfile under test was run on the Basys2. The LSB of a pseudo-random number (based on a linear feedback shift register with period 2^{32} - 1) was shifted into the scan chain, providing a random initial state. Subsequently each of the 32 possible inputs were sent into the primary input. These two steps were then repeated endlessly.

For each input, the output from the Basys2 was compared to the output of a golden circuit running inside the testbench and being presented the same inputs. Whenever a discrepancy between the output of the Basys2 and the output of the golden circuit was found, an LED would light on the board and stay lit, signaling that a Trojan had been detected.

Unfortunately the testbench implementation was overly sensitive to the noisy channel between the boards. If even a single bit was sent incorrectly to the Basys2 over the span of thousands of inputs, a false positive would be generated. A more reliable solution would have been to include at least a basic level of error detection such as a parity bit.

B. Power Analysis

The power analysis technique required using the following tools:

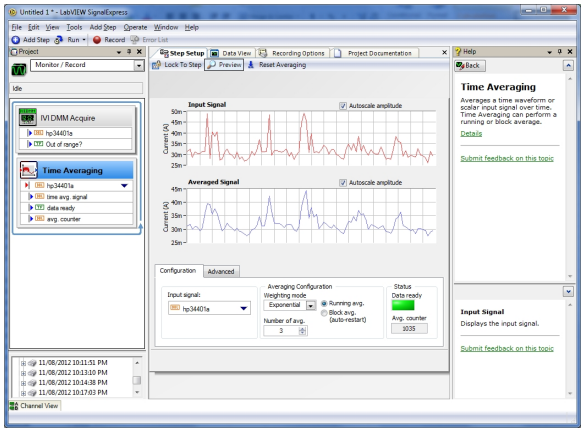


Fig. 4: LabView Signal Express used to capture and process DMM data

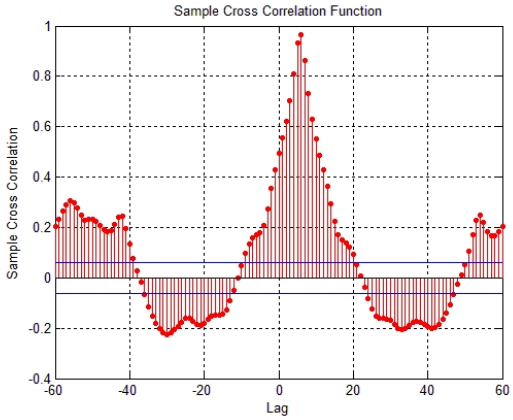


Fig. 5: Matlab's *crosscorr* shows a strong correlation between an unverified waveform and the golden waveform, with a small positive time lag, corresponding to the reset delay.

- Agilent 34410A high performance digital multimeter (DMM)
- Agilent E3631A DC power supply
- 1 Digilent Nexys2 board, running the testbench
- 1 Digilent Basys2 board, running the unknown bitfile
- LabView Signal Express, attached to DMM

1) Design A: The physical setup of the power analysis testing for Design A is shown in Figure 3. For the static input test, the DMM was connected in series with the power supply and the Basys2 "battery" pin header. Each bitfile was programmed onto the Basys2, and the DMM was used to find the average current across 500 samples. Averaging is required because even with a static input applied to the asynchronous Circuit A, the current is not constant due to other ICs running on the Basys2 board as well as ambient electromagnetic interference, power supply ripple, and other factors. However, the measurement standard deviation was around 400μA, indicating a precise reading.

For the dynamic input test, current waveforms are first generated for each bitfile. These are then cross-correlated with the waveform of a "golden" bitfile. Cross-correlation is a mathematical process which produces a value between -1 and 1, where the two limits correspond to strong correlation and 0 corresponds to no correlation. Here we expect that clean bitfiles will correlate with a value close to 1, and the tainted bitfiles will have a lower correlation. Ideally the two classes are cleanly delineated.

The Nexys2 board was programmed with the same testbench used in the functional test. The LEDs were removed from the Basys2 while it was slaved to try to reduce the associated current switching overhead. From Signal Express, the DMM was set for a 0.03s sample period, and data was recorded for 30 seconds. At the same time, a continuous averaging filter was applied to the signal to reduce the noise, with a 3 sample window. This data capture setup can be seen in Figure 4. Finally, Matlab's *crosscorr* function was used to compare the waveform data from the unverified bitfile and the golden waveform. Simple classification was performed based on the two highest and the two lowest correlations.

Table I shows the measurements for power testing. The measurements that seem to suggest the presence of a Trojan are highlighted. It is generally assumed that the circuit with the higher current contains a Trojan. However, if components are removed, the reverse could be true, which is why cross correlation can be a more valuable metric.

One limitation of this setup is that the Nexys2 needs to be reset manually, creating a synchronization problem between the waveforms. A benefit of using the *crosscorr* function is that it iterates through all possible "time lags" so that the one with the greatest positive correlation can be identified for comparison purposes. Figure 5 shows the *crosscorr* iteration over one bitfile's waveform. If the reset on the board and record function on the PC were perfectly synchronized it could be assumed that selecting the correlation at the zero time lag is sufficient for comparison.

Another limiting factor is the number of external variables involved in this process, such as sample rate and averaging window. These would need to be properly tuned for a production environment. Process variation would also become a significant concern in this area as it would introduce additional noise.

2) Design B: Power analysis was performed on Design B with a similar setup as Design A, except the inputs were static. As expected, the measurements did not reveal a substantial difference between any bitfile. The challenge for Design B is that an inordinate amount of data may need to be recorded before an input activates a hidden network– a more sophisticated approach is needed in the future.

C. Bitfile Analysis

The Spartan 3 version of the *debit* tool [13] was used to analyze the given bitfiles. In doing so, we were able to reverse engineer the provided bitfiles to provide a netlist of the main combinational circuit elements. While the output netlists could not be trivially correlated with the original golden designs, the tool still provided useful information for Design A. Many of the sub-designs had clearly differentiated numbers of nets. Unfortunately, the Spartan 3 version of the tool cannot currently handle registers and other sequential elements, so the results were inconclusive when run on Design B.

Table I shows the counts extracted from *debit*. Based on their elevated net count, the highlighted cells are those which appear to contain a Trojan.

IV. Conclusion and Future Work

Although the functionalities of these Trojans were not provided, they can be placed within the taxonomy in [5]. Trojans 1 and 2 must be user-input triggered, and 3-6 must be always on, since there is elevated current draw regardless of input. Design B could either be internally triggered (e.g. time-based),

TABLE I: Classification of bitfiles based on three features. Highlighted measurements suggest the presence of Trojans.

Design	Trojan Type	Bitfile	Functional Mismatches	Power Analysis		Net Count	Final Decision	
				I_{ave}	Cross Correlation		Trojan?	Deciding Test
A	1	1	80	0.035625	N/A	1265	Yes	Functional
		2	0	0.035598		1244	No	
		3	0	0.035601		1247	No	
		4	80	0.035602		1237	Yes	
	2	1	0	0.035603	N/A	1217	No	Functional
		2	1	0.035674		1119	Yes	
		3	0	0.035621		1244	No	
		4	1	0.035646		1187	Yes	
	3	1	0	0.035720	0.9652	1306	Yes	Power, Bitfile
		2	0	0.035717	0.9652	1306	Yes	
		3	0	0.035606	0.9814	1162	No	
		4	0	0.035631	0.9730	1129	No	
	4	1	0	0.035600	0.9720	1120	No	Power
		2	0	0.035607	0.9679	1113	No	
		3	0	0.035684	0.9603	1367	Yes	
		4	0	0.035687	0.9591	1267	Yes	
	5	1	0	0.041586	N/A	2242	Yes	Power, Bitfile
		2	0	0.035599		1079	No	
		3	0	0.041586		2242	Yes	
		4	0	0.035598		1079	No	
	6	1	0	0.036928	N/A	1404	Yes	Power, Bitfile
		2	0	0.036934		1404	Yes	
		3	0	0.035640		1053	No	
		4	0	0.035636		1053	No	
B	7	1	0 after 4hr	0.036105	N/A	N/A	No	Functional
		2	1 immediately	0.036112			Yes	
		3	1 immediately	0.036113			Yes	
		4	0 after 4hr	0.036114			No	
	8	1	1 after 8hr	0.036120	N/A	N/A	Yes	Functional
		2	0 after 4hr	0.036100			No	
		3	1 after 3.5hr	0.036140			Yes	
		4	0 after 4hr	0.036119			No	

or based on inputs. Given this diversity, the best defense is multi-faceted and layered. Thus the presented ways of detecting Trojans, when used in conjunction, offer a broader coverage than any single approach. As discussed in Section II, individual tests used were identified based on their relevancy within the 2012 CSAW Embedded Systems Challenge, and our testing does not fundamentally deviate from those already published, but instead attempts to combine a variety of testing methods to achieve more reliable Trojan identification.

Future work would involve the integration of these approaches into a formalized framework which includes a well-defined process and provides a numerical confidence level for the classifications. It would also involve an increased focus on robustness, repeatability, and flexibility.

V. Acknowledgement

We would like to acknowledge the travel grant to participate in the Embedded Systems Challenge, which was provided in part by the National Science Foundation (0958510,1059328), Army (W911NF-11-1-0470), Air Force Research Labs and Intel. The FPGA platforms for the contest were donated by Xilinx.

References

[1] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," in *Proc. of the Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[2] R. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: threats and emerging solutions," in *Proc. of the International High Level Design Validation and Test Workshop (HLDVT)*, Nov. 2009.

[3] Homeland Security News Wire, "Dell warns of Hardware Trojan," Jul. 2010. [Online]. Available: http://www.homelandsecuritynewswire.com/dell-warns-hardware-trojan

[4] Dangerous Prototypes, "Defcon 16: Hardware Trojans using FPGA," Jan. 2011. [Online]. Available: http://dangerousprototypes.com/2011/01/25/defcon-16-hardware-trojans-using-fpga/

[5] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39 –46, oct. 2010.

[6] C. Brown and G. Linden, "Semiconductor capabilities in the U.S. and industrializing Asia," in *Industry Studies*, 2008.

[7] R. Rad, J. Plusquellic, and M. Tehranipoor, "A sensitivity analysis of power signal methods for detecting hardware Trojans under real process and environmental conditions," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 18, no. 12, pp. 1735–1744, Dec. 2010.

[8] C. Lamech, R. Rad, M. Tehranipoor, and J. Plusquellic, "An experimental analysis of power and delay signal-to-noise requirements for detecting Trojans and methods for achieving the required detection sensitivities," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 6, no. 3, pp. 1170–1179, Sep. 2011.

[9] L.-W. Wang and H.-W. Luo, "A power analysis based approach to detect Trojan circuits," in *Proceedings of the International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE)*, Jun. 2011.

[10] D. Ziener, S. Assmus, and J. Teich, "Identifying FPGA IP-cores based on lookup table content analysis," in *Proc. of the International Symposium on Field Programmable Logic and Applications (FPL)*, 2006.

[11] S. Drimer, "Volatile FPGA design security a survey," 2007.

[12] A. Megacz, "A library and platform for FPGA bitstream manipulation," in *Proc. of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2007.

[13] J.-B. Note and E. Rannaud, "From the bitstream to the netlist," in *Proc. of the International Symposium on Field Programmable Gate Arrays (FPGA)*, 2008.

[14] F. Benz, A. Seffrin, and S. A. Huss, "Bil: A tool-chain for bitstream reverse-engineering," in *Proc. of the International Symposium on Field Programmable Logic and Applications (FPL)*, 2012.