Hardware Trojan Implemented on a FPGA

Justin Cox and Tyler Travis
Department of Electrical and Computer Engineering
Utah State University
Logan, Utah 84322
email: justin.n.cox@gmail.com, tyler.travis@aggiemail.usu.edu

***Abstract*— Hardware trojans are becoming more of a threat to devices and systems as the number of ICs and chips are being manufactured and fabricated overseas. These overseas facilities often also work with other third party company and it becomes more complex to track down where each step of the design phase takes place. If an attacker was able to gain control of a step in the design phase, a hardware trojan could be inserted to corrupt functionality and reliability, or leak important or secret information. This paper discusses the method of designing a hardware trojan for an FPGA. The most effective method of defending against these hardware trojans is understanding how they are implemented.**

***Index Terms*—hardware trojan, data leaking, security, FPGA.**

I. INTRODUCTION

Hardware trojans are a great way for an attacker to leak important information such as encryption/decryption keys because, if well implemented, are hard to detect. Since most of the ICs and chips made today are important from overseas, detected these hardware trojans is critical. There are a few methods of detecting hardware trojans [1] which include power analysis or side-channel analysis. This paper will discuss the implementation of a hardware trojan on an FPGA which has been synthesized to run a 8051 Intel Microprocessor architecture. This processor has been programmed to function as a dedicated DES crypto device.

A. Previous Work

There is current and previous research on the implementation of hardware trojans and how they are used to leak information [2]. There are even competitions designed to allow competitors to design a hardware trojan on a microprocessor. This paper will focus on a FPGA hardware trojan implementation to leak the secret key used in the DES algorithm. This paper will try to design a trojan with a small profile, low power requirements, and the ability to remain hidden from detection.

II. 8051 IP CORE

The Intel 8051 architecture was developed by Intel in the 1980s. Intel has allowed other companies to design their own versions of the 8051 architecture, and as a result, can be found in many embedded system designs. Since the 8051 microcontroller is an older design, it is limited to 8-bit instructions and has a limited amount of peripherals. This is illustrated in Figure 1 and needs to be taken into consideration when designing the hardware trojan.

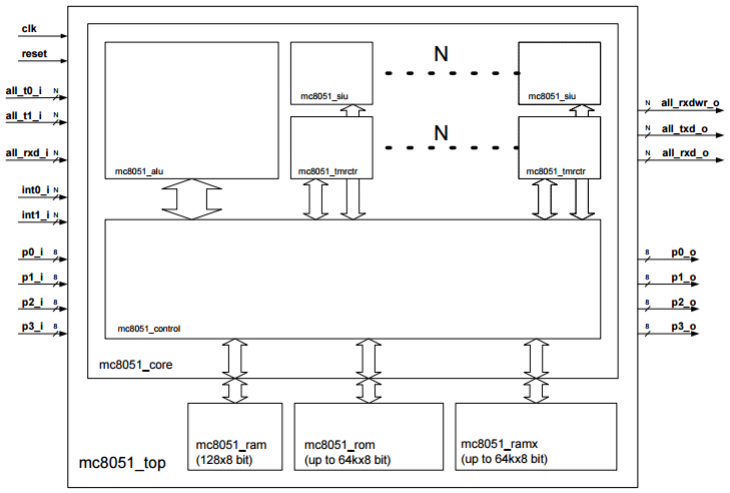

Fig. 1. System overview of the Intel 8051.

There are free 8051 IP cores available on the internet and the authors chose to use the 8051 VHDL implementation from Oregano Systems [3]. The IP core is able to have three memory modules loaded onto the FPGA design. The 8051 is limited to 128 bits of internal RAM but the ROM and extra RAM can be designed up to 64kB of data.

Once the 8051 IP core has been loaded on the FPGA, a DES program in C needs to be loaded in order to allow encryption and decryption.

III. TROJAN DESIGN

There are many different ways to characterize a trojan and this paper will use the taxonomy described in the following subsection.

A. Taxonomy

Hardware trojans will be characterized based on the following metrics:

- Insertion
- Abstraction
- Activation
- Effects
- Location

The proposed hardware trojan, once triggered, will delay the communication lines of the UART module to leak the secret key used for encryption and decryption. Tampering with

the baudrate would likely create errors in the communication. This trojan will maintain the designed baudrate while created different delays after each 8-bit data frame.

The trojan is inserted at the Design and Fabrication stage. It is assumed that the attacker would have access to the netlists and hardware description code to make the necessary changes. The trojan would be extracted at the RT level and gate level because the trojan effects the control logic as well as some gate signals. The trojan will be activated externally by a user which decreased the chance of being detected. The trojan will be triggered when a predetermined 6 byte chunk of the plaintext is entered into the crypto system. The trojan design is shown further in Figure 2.

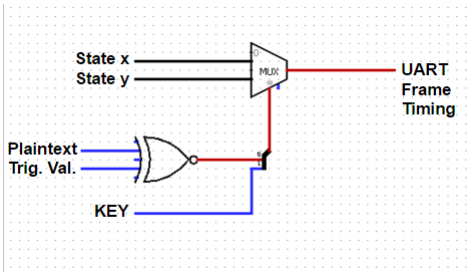

Fig. 2. Logic design of the hardware trojan

As mentioned previously, the effect of the trojan will be the leakage of secret information. The trojan will be located on parts of the processor and well as some of the I/O such as the RS-232 signals.

B. Serial Communication

A separate program was written in C to allow an interface for the user to input a desired plaintext to be encrypted. The resulting cyphertext is sent over UART to the user. The trojan is triggered when the plaintext of ”0x70776e616765... in hexadecimal or pwnage in ASCII is received. The user interface of the crypto system is shown in Figure 3.

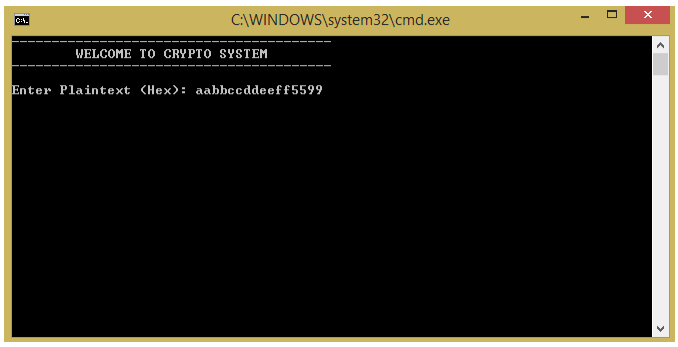

Fig. 3. The UI that allows the trigger to be sent.

IV. EXPERIMENTAL RESULTS

The trojan worked as expected once everything was loaded onto the board. The spacing between the frames from the varying stop bits caused the bits for the key to be leaked out over 8 plain text inputs, with the first one being the triggering plaintext of ”pwnage” or 0x7077.6e61.6765 in hex. The only part left to the person leaking the key is to brute force the remaining 8 bits left off by completed ciphertext outputs on the TX line of the FPGA.

Since the authors were not able to get the 8051 IP core completely working on the Spartan 3E FPGA, the trojan and DES system were implemented directly on the FPGA. The UART IP Core was obtained from www.fpga4fun.com and the DES IP Core was obtained from OpenCores. The credit of these cores are found in the source code.

A. Size

The size of this trogan compared others is realitiviy small compared to the other trojans in the class. With only a 6.2% increase in look up tables (LUTs) at the worst, there isn't a lot of extra area used. Refer to Table xx for more information.

	Original	With Trojan	Difference	Percentage
LUTs	835	887	52	6.2%
Slices	562	587	25	4.4%
Flip Flops	451	467	16	3.5%

B. Power

The power that was consumed by the trojan was a little bit more than other trojans in the class. This is probably due to the size of some of the components in the design.

	Original	With Trojan	Difference	Percentage
Volts (supply)	5.5	5.5	0	0%
Amps (supply)	0.067	0.067	0	0%
Volts (resistor)	1.72mV	1.84mV	0.12mV	6.9%
Watts	7.8mw	8.4mW	0.6mW	7.69%

C. Detectability

The detectability of the trojan is pretty low since the trojan is only activated when a certain event triggers it. Therefore, the output of the device will stay consistent as long as the plaintext trigger isn't entered into the device. Once the trigger has been triggered, the device will leak the key through the next 8 UART frames for the attacker to use.

D. Measurements

The figures for the measurements are included in the appendix of this report.

V. CONCLUSION

The trojan was successfully implemented on a Spartan 3E FPGA. The trojan does not take up a lot of extra space and would be hard to detect when analyzing the floor plan. The trojan also was able to maintain a low power profile. The key can be reliably extracted from the UART Tx communication line and it would only be detected if the signal was analyzed with an oscilloscope after activation. The probably of accidentally activating the trojan is low which would also hinder the ability to test and detect the trojan.

The authors were not able to get the 8051 IP core working with a large DES code written in C. The authors were able to

learn a lot about microcontroller implementations on FPGAs and the time spent debugging the 8051 IP core was worth while.

REFERENCES

[1] T. Machida, D. Yamamoto, M. Iwamoto, K. Sakiyama. A New Mod of Operation for Arbiter PUF to Improve Uniqueness on FPGA. The University of Electro-Communications. Tokyo, Japan. 2014.

[2] J. Orlin Grabbe. The DES Algorithm Illustrated. *Laissez Faire City Times*, 2006.

[3] T. Machida, D. Yamamoto, M. Iwamoto, K. Sakiyama. Implementation of Double Arbiter PUF and its Performance Evaluation on FPGA. The University of Electro-Communications. Tokyo, Japan. 2015.

[4] M. Rostami, M. Majzoobi, F. Koushanfar, D. Wallach, S. Devadas. Robust and Reverse-Engineering Resilient PUF Authentication and Key-Exchange by Substring Matching. Rice University. Houston, Texas. January. 2014.

[5] Martin Deutschmann. Cryptographic Applications with Physically Unclonable Functions. Alpen-Adria-Universität Klagenfurt. Klagenfurt. November. 2010.