

Hardware Trojan Designs on BASYS FPGA Board

Zhimin Chen, Xu Guo, Raghunandan Nagesh, Anand Reddy, Michael Gora, and Abhranil Maiti

Virginia Polytechnic Institute and State University

Blacksburg, VA 24060

Email: {chenzm, xuguo, rnagesh, anandrdy, gora, abhranil}@vt.edu

Abstract

A hardware Trojan is now considered as a possible threat to integrated circuits when they are produced in an un-trusted foundry. The published detection methods search for Trojans that are inserted during the manufacture step. In our Trojan implementations, we design Trojans that can not only be added in foundries, but also can be inserted into the RTL or netlist code in design houses. *Most importantly, these Trojans will not be activated during the simulation analysis even if the tester provides the correct trigger pattern. Only after implementation, the Trojans can go active when the trigger condition is satisfied.*

We investigate a ‘content & timing’ based Trojan trigger and two methods of obfuscated communication of the trigger signal to the Trojan: ‘thermal communication’, and ‘feedback’. The trigger method makes it very difficult for both simulation analysis and detection after implementation. Next, the two obfuscation techniques can only be detected after implementation which provides a more powerful attack mechanism since they do not cause abnormal behavior during simulation analysis. By combining these methodologies together, we designed hardware Trojans.

1. Introduction

To save cost, many chips are manufactured abroad. However, if the foundry abroad cannot be trusted, some malicious modifications, called hardware Trojans [1], may be made to the chip, which can cause abnormal function once a certain trigger condition is satisfied. To detect a Trojan, the most expensive way is to exhaust the input patterns and see whether the outputs are as expected. If the trigger condition is of very low probability, it is hard to verify whether a chip is secure or not. Some other more efficient methods use parametric testing techniques. They do not check the output, but turn to parametric characteristics of the chip, such as the power consumption [2], path delay [3] and so on. Commonly, these methods require a correct reference netlist for comparison. If a difference can be found between the susceptible design and the reference netlist, probably some additional circuits are inserted.

The aforementioned detection methods aim to protect circuits from modifications in the implementation phase. However, a hardware Trojan can also be inserted to the RTL or netlist code. In this design phase, modifications are easier to make. As the chip becomes more and more complex, it becomes more and more common that companies try to buy IP blocks from other companies. If there is a Trojan in the IP block the chips are still insecure, even if the

foundry can be trusted. What's more, compared with the foundries, there are many more design companies offering different kinds of IP modules. This means the probability to add Trojan here is substantially higher than that at foundries.

Compared to the physical chip coming back from manufacturers, the RTL or netlist codes are no longer black boxes. It is much easier to analyze whether there are some unexpected functions. However, logic analysis and simulation do not reveal everything. Some physical features of the chip or some information on the PCB are not taken into account in stand-alone logic analysis or simulation. If the Trojan can make use of this, their destructive function will not appear during analysis and simulation. This idea is illustrated in Figure 1, where the physical features and PCB information are represented by a 'covert communication channel'. They are not visible by the logic analysis and simulation.

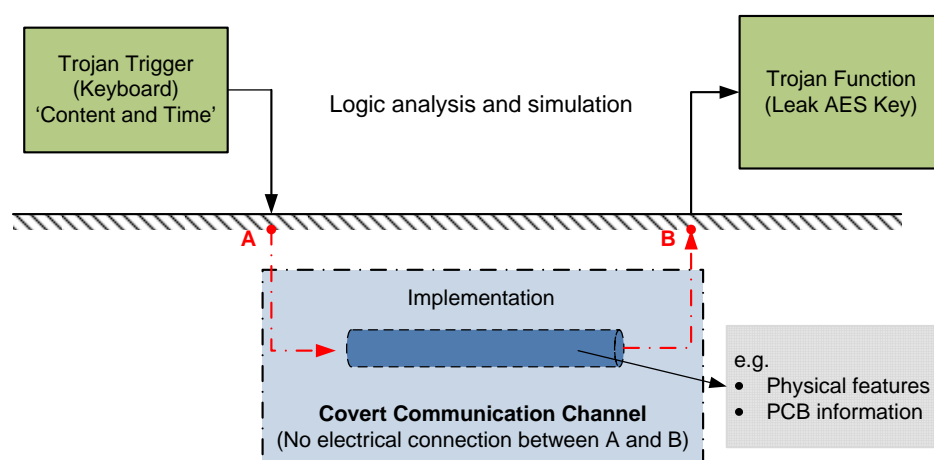


Figure 1: Idea of covert communication channel

In this paper, we successfully explore a technique for triggering Trojans called 'content & timing' and two additional methods for obfuscating the trigger signal to the Trojan: 'thermal communication', and 'feedback'. We begin by investigating a trigger mechanism that decreases the probability of activating the Trojan – 'content & timing'. With this approach the Trojan can be triggered only when the correct input pattern is entered at the correct time. Compared with methods that solely rely on content, the inclusion of timing as a trigger renders it impractical to detect Trojans by exhaustive testing. 'Thermal communication' method makes use of the temperature on the chip to activate the Trojan. Since logic analysis and simulation do not consider the temperature, Trojan will never be activated within the design phase. Finally, the 'feedback' method makes use of the existing resources or peripherals on the BASYS board [4]. Off-chip peripherals are typically not included in a logic simulation of the design. However, the signals feedback from off-chip peripherals can be used to transport a trigger signal that activates the Trojan.

We will illustrate the Trojan implementations in Section 2. In Section 3, we present the result of our Trojans. Finally, Section 4 concludes this paper.

2. Techniques for Hardware Trojan Design

2.1 Content & Timing

The main idea behind the triggering method is to reduce the probability of finding the trigger pattern. Generally speaking, the more inputs that are required, the lower the probability will be to find the Trojan. Our method to increase the inputs for the pattern is to use a two-dimension keyboard input pattern – ‘content & time’. Trojan can be triggered only when the correct patterns are input at the correct time. For example, suppose the trigger pattern is F1 + F2 and we have 100 choices of keys including not input anything. If we do not including timing, then the probability to find the pattern is

$$1/100 * 1/100 = 10^{-4}$$

After including timing, say F2 can only be pressed 10 – 11 seconds after F1 is pressed. Within these 10 seconds, pressing any key will fail the trigger. Suppose we can press the keyboard for 2 times per second, then the probability to find the pattern is

$$(1/100)^{21} + (1/100)^{22} = 10^{-42}$$

which means trying 10^{42} times can give the tester a correct pattern. That is

$$10^{42} * 10 \text{ seconds} = 3 * 10^{35} \text{ years.}$$

Therefore, we can see the ‘content & timing’ approach makes it even impossible to find the correct pattern by exhaustive testing.

In our Trojan design:

- 1) A Finite State Machine (FSM) is used to produce the trigger only when the right patterns are entered separately within a specified time period.
- 2) A very low frequency internal clock (2.98Hz compared with the 50MHz system clock) is used to sample the input patterns. Compared with most of the other parts of the circuit, which run at 50MHz, power consumed by the FSM can be neglected.
- 3) Unused keys (e.g. F1 and F2) are used as trigger patterns to increase the testing space.

In detail, the working mechanism of the FSM used in this trigger method is as follows (see Figure 2): when the first right pattern, **F1**, is entered, the state jumps from **INIT** (initial state) to **WAIT** (wait state); during a specified wait period (e.g. 7 seconds), which is controlled by a counter with a 2.98Hz clock, if any key is pressed, the state will go back to **INIT**, otherwise it will jump to **GEN** (generate trigger state); then, within a specified time period (e.g. 7 seconds), if the second right pattern, **F2**, is entered, the trigger signal is generated, otherwise it will go back to the **INIT**.

2.2 On-Chip Thermal Communication

The first covert Trojan trigger communication method exploits a method that is invisible during the logic analysis and simulation. For example, we utilize the temperature of the chip to activate the Trojan. We refer to this as on-chip thermal communication. Once an input pattern that generates a lot of heat is entered, then the Trojan becomes active.

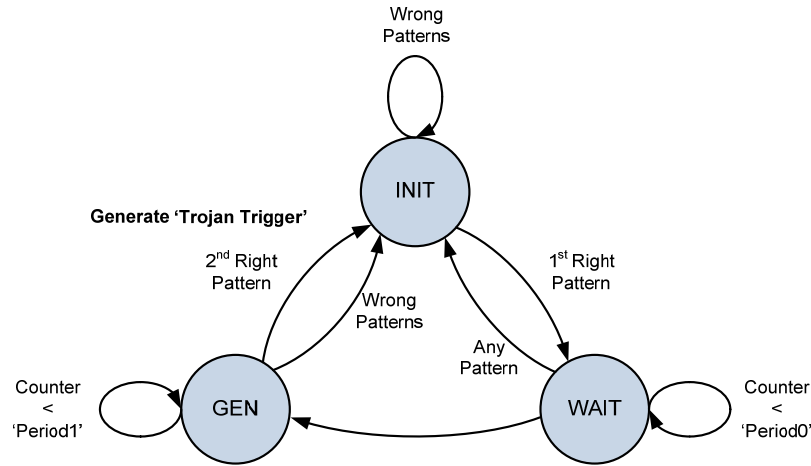


Figure 2: the FSM of the Input Pattern trigger generator

The advantage of this kind of communication method is that it cannot be simulated at the design phase. If such a Trojan trigger is inserted into a hardware IP block, it can hide until the chip is manufactured and the Trojan can escape from detection even if the tester is able to do thorough logic analysis on the RTL code. This makes it possible to add the Trojan at the design house.

The basis of on-chip thermal communication is that the delay of a circuit varies according to its temperature. The heat of the chip is mainly caused by the circuit activity: more activity, more heat. We exploit this by designing two small circuit modules: the heat generator and the temperature sensor. The heat generator is used to generate different amounts of heat while the temperature sensor is sensitive to the delay related to the temperature. In such way, we can build a communication channel between two parts of the circuit without wire connection.

In our Trojan design, we use inverter-based ring oscillators to both generate the heat and sense the temperature. The design structure is shown in Figure 3 as follows.

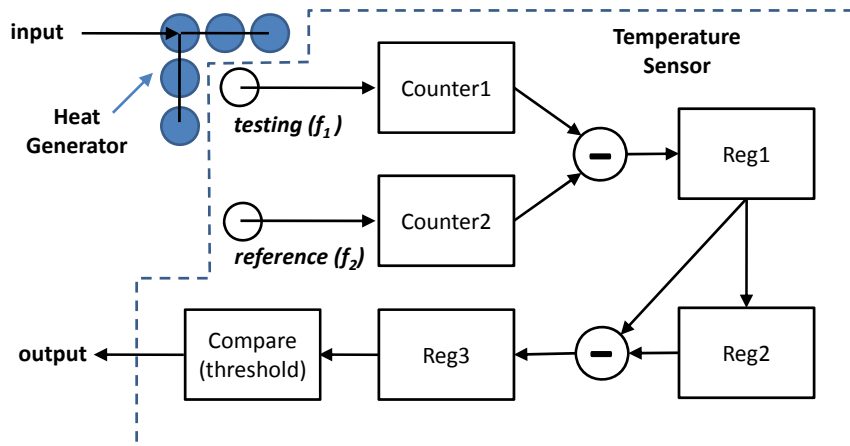


Figure 3: On-chip thermal communication structure

As we can see from figure 3, the on-chip thermal communication consists of two parts in different colors. The shaded part represents the heat generation part and the temperature sensor denoted by dotted region. The heat generator is implemented by a set of inverter-based ring oscillators (called surrounding ring oscillators) which are included with a shared enable signal. When enabled, all the ring oscillators run at high frequency and generate more heat than when disabled. The temperature sensor utilizes two ring oscillators; one is placed close to the heat generation part, called testing ring oscillator. While the other is placed far away from the first part, called reference ring oscillator. Outputs of both ring oscillators are fed to 2 counters as clock signals. These two counters accumulate the delay difference between those 2 ring oscillators. By subtracting these 2 counters, we can derive the difference. We use the low frequency clock called 'slowclk' in the reference design to sample the difference and store it into 'reg1'. Then the difference in 'reg1' is delayed by one clock cycle in 'reg2'. Again, we do the subtraction to get the difference of 'reg1' and 'reg2' and store the new difference in 'reg3'. Finally, value in 'reg3' is compared with a 'threshold'. If the 'reg3' > 'threshold', then the trigger is '1', otherwise, the trigger remains '0'.

The operating mechanism of the above design can be explained as follows.

- 1) When the input is low, all the surrounding ring oscillators are disabled. No heat is generated. Suppose the frequencies of the testing ring oscillator and the reference ring oscillator are f_1 and f_2 . Then the value in 'reg1' and 'reg2' should be $N*(f_1 - f_2)$, where N is a constant. Value in 'reg3' should be near zero because frequency f_1 and f_2 are not changing much. The threshold is not reached in this case, and the trigger remains '0'.
- 2) When the input goes from low to high, f_1 will decrease to f_1' . In the next cycle the 'reg1' value will be $N*(f_1' - f_2)$ while the value in 'reg2' remains $N*(f_1 - f_2)$. At this time, the difference between 'reg1' and 'reg2' should not be near 0 any longer, say bigger than the 'threshold'. Finally, the trigger turns to be '1'.

From this analysis, we can find that there is no electrical path between 'input' and the 'output'. However, every time when 'input' goes from '0' to '1', the 'output' will go from '0' to '1' for one or a few cycles.

2.3 Feedback

The second communication obfuscation method is to divide a Trojan into several small parts. Each part is an incomplete Trojan and harmless. However, when we combine them together, the Trojan is there. For example, the FPGA contains half of a Trojan and a peripheral contains the other half. Each half will not insert malicious functions to the chip or system individually. However, when we plug the FPGA and the peripheral on the same board, the Trojan channel exists on the board system.

To detect this kind of Trojan, one should gather every small part of the Trojan together and for testing or analysis. However, in most cases, combining the chip and peripherals for analysis is not practical. For example, some of the peripherals are analog devices. Even if the tester has a way to analyze mixed signals, it is infeasible to use mixed-mode simulation throughout the entire design testbench.

After analyzing the system, we find that the VGA interface can be used as a peripheral which acts as a part of the Trojan communication channel. As shown in Figure 4, each bit of the color signals (R, G, B) are wired together outside of the FPGA on the PCB board resulting in an existing path from one port to another port (for example from RED0 to RED2). Our idea is to output a signal to port RED2, and then read it back from RED1. In this case, the VGA peripheral acts as a **'feedback'**. Suppose the input signal on port RED1 is used to trigger the Trojan, with the VGA peripheral, we output '1' on port RED2. After it is captured by RED1, the Trojan is active. However, without the VGA peripheral, there is no signal for port RED1 and hence the Trojan will never be triggered.

What's more, to make the Trojan channel more difficult to detect, we configure the port RED2 as an 'input' and output '1' on port RED1, only when we want to trigger the Trojan. In any other time, they act exactly the same as they were in the reference design.

The operation mechanism of the 'feedback' Trojan trigger can be explained as follows: For the VGA interface part, after changing the of 'RED1' to be a tri-state signal and hence has the 'input' function (original design only uses it as 'output'), the feedback signal is available if the 'sample_en' is high, which may be then used to trigger the Trojan function inside the FPGA; for the FPGA part, the 'sample_en' signal is generated to configure the 'inout' port, RED1, to behave as 'input' and simultaneously sample the input feedback signal and make the RED2 output constant '1'. To improve the design, this 'sample_en' signal can directly use the 'Input Pattern trigger' signal. In this case, even the 'Input Pattern trigger' is detected in the simulation of standalone FPGA, the Trojan function will not triggered until the FPGA receives the input feedback from the VGA, which can only be achieved when running the whole system on board. Using the 'inout' port as the second-stage Trojan trigger is almost impossible to be detected in simulation since it behaves as 'output' as usual under normal working mode and changes to 'input' only when the first-stage trigger is enabled.

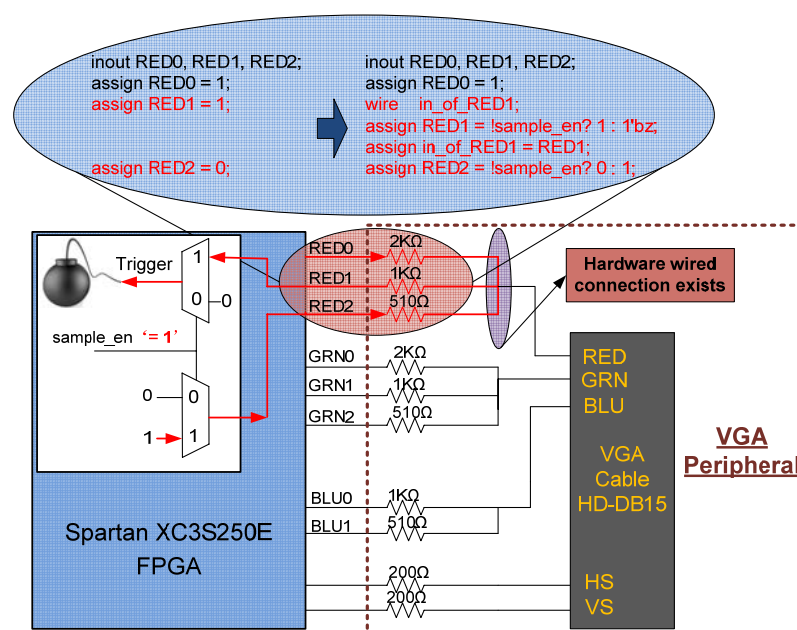


Figure 4: VGA definitions and BASYS board circuits

3. Hardware Trojans Results

We implement 4 Trojan trigger methods according to the above 3 Trojan techniques.

- 1) Trojan1 with only the 'content & timing' technique;
- 2) Trojan2 with 'content & timing' and 'thermal communication' techniques;
- 3) Trojan3 with 'content & timing' and 'feedback' techniques;
- 4) Trojan4 with all the 3 techniques combined together.

The URL for the source codes is:

http://filebox.vt.edu/users/chenzm/Embedded_System_Challenge/

The malicious Trojan function is to output the secret key instead of the encrypted message through RS232.

3.1 Trojan1 ('Content & Timing')

We changed the following files: alphatop.v, kb2ascii.v, kbttop.v, video_ctrl.v, vga80x40.vhd.

The folder name in the submission is 'trojan1_keyboard'.

Operation steps and results:

- 1) Press the 'F1' key on the keyboard.(You can press 'F1' before or after pressing the 'INI System button'.)
- 2) Wait 7 seconds. (Do not press any key during this time. If you do you will need to start again from step 1.)
- 3) Press the 'F2' key on the keyboard between the 7th and 14th second to trigger the Trojan. (If you do not press 'F2' during this period, the state machine will timeout and you will need to start from step 1)
- 4) The 'ALPHA' logo will flash blue and green to indicate that the Trojan has been triggered.
- 5) The 'AES key' will be available to the public on the RS232 port if you press the 'Start Encryption' and then the 'Transmit' button. (A hyperterminal will show this.)
- 6) Pressing the F3 key will make the trojan inactive and get you get you back to the initial state. To trigger the Trojan again you will need to start from step 1). However, once the Trojan has been triggered the damage is irreversible.(The 'AES' key has already been leaked)

Device utilization summary:

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	1,486	4,896	30%
Number of 4 input LUTs	4,230	4,896	86%
Logic Distribution			
Number of occupied Slices	2,404	2,448	98%
Number of Slices containing only related logic	2,404	2,404	100%
Number of Slices containing unrelated logic	0	2,404	0%
Total Number of 4 input LUTs	4,384	4,896	89%
Number used as logic	4,230		
Number used as a route-thru	154		
Number of bonded IOBs	46	108	42%
Number of RAMB16s	8	12	66%
Number of BUFGMUXs	4	24	16%
Number of DCMs	3	4	75%
Number of MULT18x18SIOs	2	12	16%


3.2 Trojan2 ('Content & Timing' and 'thermal communication')

We changed the following files: alphetop.v, kb2ascii.v, kbttop.v, video_ctrl.v, vga80x40.vhd. Also, we added ro_counter_ref.v, ringosc_3_v3.0.v, latchfb.v for the thermal communication. The folder name in the submission is 'trojan2_keyboard_ring'.

Operation steps and results:

- 1) Press the 'F1' key on the keyboard.(You can press this before or after pressing the 'INI System button'.)
- 2) Wait 7 seconds. (Do not press any key during this time. If you do you will need to start again from step 1.)
- 3) Press the 'F2' key on the keyboard between the 7th and 14th second to trigger the Trojan. (If you do not press 'F2' during this period, the state machine will timeout and you will need to start from step 1)
- 4) The 'ALPHA' logo will flash blue and green to indicate that the Trojan has been triggered.
- 5) The 'AES key' will be available to the public on the RS232 port if you press the 'Start Encryption' and then the 'Transmit' button. (A hyperterminal will show this.)
- 6) Pressing the F3 key will make the trojan inactive and get you get you back to the initial state. To trigger the Trojan again you will need to start from step 1). However, once the Trojan has been triggered the damage is irreversible.(The 'AES' key has already been leaked)

Device utilization summary:

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Total Number Slice Registers	1,609	4,896	32%	
Number used as Flip Flops	1,601			
Number used as Latches	8			
Number of 4 input LUTs	4,278	4,896	87%	
Logic Distribution				
Number of occupied Slices	2,445	2,448	99%	
Number of Slices containing only related logic	2,445	2,445	100%	
Number of Slices containing unrelated logic	0	2,445	0%	
Total Number of 4 input LUTs	4,469	4,896	91%	
Number used as logic	4,278			
Number used as a route-thru	191			
Number of bonded IOBs	46	108	42%	
Number of RAMB16s	8	12	66%	
Number of BUFGMUXs	7	24	29%	
Number of DCMs	3	4	75%	
Number of MULT18x18SIOs	2	12	16%	

3.3 Trojan3 ('Content & Timing' and 'feedback')

We changed the following files: alphetop.v, kb2ascii.v, kbttop.v, video_ctrl.v, vga80x40.vhd. The folder name in the submission is 'trojan3_keyboard_vga'.

Operation steps and results:

- 1) Press the 'F1' key on the keyboard.(You can press this before or after pressing the 'INI System button'.)
- 2) Wait 7 seconds. (Do not press any key during this time. If you do you will need to start again from step 1.)
- 3) Press the 'F2' key on the keyboard between the 7th and 14th second to trigger the Trojan. (If you do not press 'F2' during this period, the state machine will timeout and

you will need to start from step 1)

- 4) Once the thermal trojan is active, the characters on the screen will turn blue to indicate that the trojan has been triggered. (Note: You might need to wait a couple of seconds for the thermal trojan to become active)
- 5) The 'AES key' will be available to the public on the RS232 port if you press the 'Start Encryption' and then the 'Transmit' button. (A hyperterminal will show this).
- 6) Pressing the F3 key will make the trojan inactive and get you back to the initial state. To trigger the Trojan again you will need to start from step 1). However, once the Trojan has been triggered the damage is irreversible.(The 'AES' key has already been leaked)

Device utilization summary:

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	1,488	4,896	30%
Number of 4 input LUTs	4,203	4,896	85%
Logic Distribution			
Number of occupied Slices	2,397	2,448	97%
Number of Slices containing only related logic	2,397	2,397	100%
Number of Slices containing unrelated logic	0	2,397	0%
Total Number of 4 input LUTs	4,356	4,896	88%
Number used as logic	4,203		
Number used as a route-thru	153		
Number of bonded IOBs	46	108	42%
Number of RAMB16s	8	12	66%
Number of BUFGMUXs	4	24	16%
Number of DCMs	3	4	75%
Number of MULT18x18SIOs	2	12	16%

3.4 Trojan4 (all 3 techniques)

We changed the following files: alphetop.v, kb2ascii.v, kbttop.v, video_ctrl.v, vga80x40.vhd. Also, we added ro_counter_ref.v, ringosc_3_v3.0.v, latchfb.v.

The folder name in the submission is 'trojan4_keyboard_vga_ring'.

Operation steps and results:

- 1) Press the 'F1' key on the keyboard.(You can press this before or after pressing the 'INI System button'.)
- 2) Wait 7 seconds. (Do not press any key during this time. If you do you will need to start again from step 1.)
- 3) Press the 'F2' key on the keyboard between the 7th and 14th second to trigger the Trojan. (If you do not press 'F2' during this period, the state machine will timeout and you will need to start from step 1)
- 4) The characters on the screen will first turn 'yellow' and then turn 'white' to indicate that the 'VGA Trojan' and the 'THERMAL SENSOR' trojan respectively have been triggered.
- 5) The 'AES key' will be available to the public on the RS232 port if you press the 'Start Encryption' and then the 'Transmit' button. (A hyperterminal will show this.)
- 6) Pressing the F3 key will make the trojan inactive and get you back to the initial state. To trigger the Trojan again you will need to start from step 1). However, once the Trojan has been triggered the damage is irreversible.(The 'AES' key has already been leaked)

Device utilization summary:

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Total Number Slice Registers	1,613	4,896	32%
Number used as Flip Flops	1,605		
Number used as Latches	8		
Number of 4 input LUTs	4,279	4,896	87%
Logic Distribution			
Number of occupied Slices	2,433	2,448	99%
Number of Slices containing only related logic	2,433	2,433	100%
Number of Slices containing unrelated logic	0	2,433	0%
Total Number of 4 input LUTs	4,470	4,896	91%
Number used as logic	4,279		
Number used as a route-thru	191		
Number of bonded IOBs	46	108	42%
Number of RAMB16s	8	12	66%
Number of BUFGMUXs	7	24	29%
Number of DCMs	3	4	75%
Number of MULT18x18SIOs	2	12	16%

4. Conclusion

In this paper, we successfully explore a technique for triggering Trojans called ‘content & timing’ and two additional methods for obfuscating the enable signaled: ‘thermal communication’, and ‘feedback’. Triggering method prevents detection of the Trojan by adding an additional dimension of time. The first covert communication technique hides the Trojan enable signal from logic analysis and simulation by transmitting the signal with temperature variation. The second covert communication technique divides the Trojan into harmless small parts and implements them into different chips or peripherals. The difficulty to analyze different chips and peripherals together makes it even impossible to detect such kind of Trojan enable channel; These Trojan techniques allow us to insert the Trojan not only in foundries but also in hardware IP design houses, which largely increase the risk for chips.

Based on these 3 techniques, we designed 4 different Trojan schemes for demonstration. All of them work reliably and correctly. The modifications to the reference design are also very small, with minimal impact on area and power consumption.

References

- [1] R. Rad, J. Plusquellic and M. Tehranipoor, “Sensitivity Analysis to Hardware Trojans using Power Supply Transient Signals,” In Proc. of Workshop on Hardware-Oriented Security and Trust 2008, pp 3 – 7.
- [2] M. Banga and M. S. Hsiao, “A Region Based Approach for the Identification of Hardware Trojans,” In Proc. of Workshop on Hardware-Oriented Security and Trust 2008, pp 43 – 50.
- [3] Y. Jin and Y. Makris, “Hardware Trojan Detection Using Path Delay Fingerprint,” In Proc. of Workshop on Hardware-Oriented Security and Trust 2008, pp 62 – 67.
- [4] http://www.digilentinc.com/Data/Products/BASYS/BASYS_E_RM.pdf