Experiment 5
Hardware Trojan Attack II

We describe an experiment on the Hardware Trojan attack which requires you to insert your own Trojan triggered by temperature sensor measurement signal

Instructor: Dr. Swarup Bhunia
Co-Instructors/TAs: Shuo Yang and Reiner Dizon-Paradis

Case Study

We describe an experiment on hardware Trojan attacks which requires you to insert your own Trojan triggered by temperature sensor measurement signal.

Temperature Sensor in the FPGA

MAX 10 FPGA on the HAHA Board features one analog-to-digital converter (ADC). The ADC provides the MAX 10 with built-in capability for on-die temperature monitoring and external analog signal conversion. In the temperature sensing mode, it monitors external temperature data input with a sampling rate of up to 50-kilo samples per second.

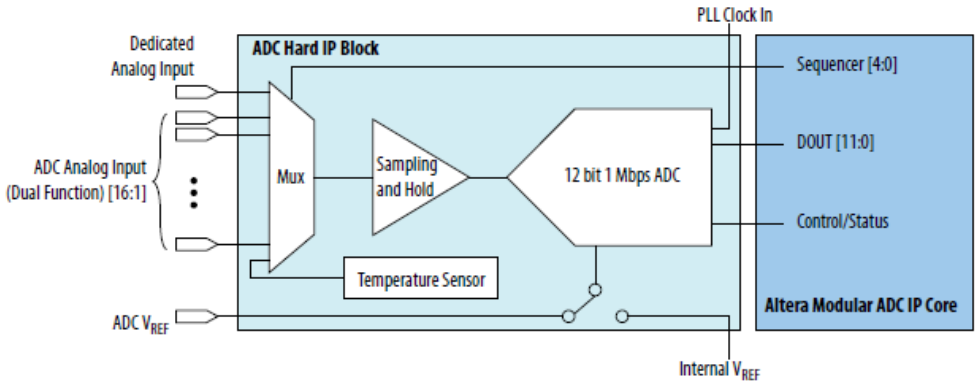

Figure 1 ADC Hard IP Block in the MAX 10 FPGA

While using the temperature sensing mode, the ADC sampling rate is 50-kilo samples per second during temperature measurement. After the temperature measurement completes, if the next conversion in the sequence is normal sampling mode, the Altera Modular ADC IP core automatically switches the ADC back to normal sampling mode. The maximum cumulative sampling rate in normal sampling mode is 1 MSPS. When the ADC switches from normal sensing mode to temperature sensing mode, and vice versa, calibration is run automatically for the changed clock frequency. The calibration incurs at least six clock calibration cycles from the new sampling rate. The ADC TSD measurement uses a 64-samples running average method.

Theory Background

The taxonomy of Trojan circuits has been presented in various forms, and it continues to evolve as newer attacks and Trojan types are discovered. Figure 2 shows a high-level classification based on variations in activation mechanism and Trojan effect.

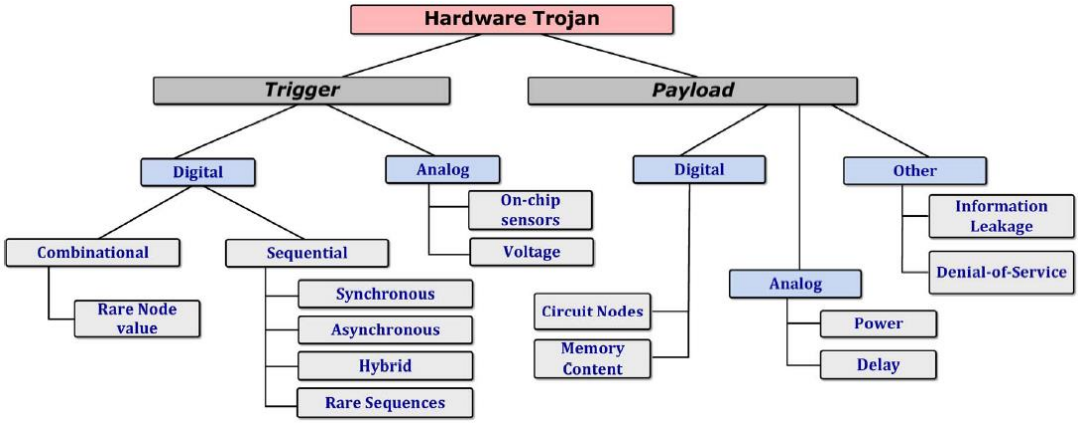

Figure 2 Trojan taxonomy based on trigger and payload mechanisms.

Based on the trigger condition, the hardware Trojans can be classified into analog or digital Trojans. The former is activated by analog conditions such as temperature, delay, or device aging effect, whereas the latter are triggered by some Boolean logic function. Digitally triggered Trojans can again be classified into combinational and sequential types.

In terms of the payload, the Trojan can cause functional failure upon triggering or have a passive effect such as heating of the die or leaking of information. A Trojan can cause an ‘‘information leakage’’ attack, where secret information is leaked by a Trojan via a transmitted radio signal or serial data port. It could also involve a side-channel attack where the information is leaked through the power trace or through thermal radiation or through optical modulation of an output LED. Another type of Trojan payload would be an unauthorized alteration in system behavior.

Experiment Set-up: Configuration

The instruments needed for this experiment are the HAHA Board, a USB Blaster a computer and a Heater, e.g., a hairdryer.

To use the ADC Hard IP (Figure 1) in the FPGA, it should be configured in Quartus from IP Catalog→Altera Modular ADC core. An IP parameter Editor window will come up for you to configure the parameters. Among the four core variants, the "Standard sequencer with external sample storage" is recommended. ADC input clock should use lower than 50MHz ones, such as 10MHz. In the Channels tab, choose the TSD (Temperature Sensing Diode) and use the on-chip TSD. In the Sequencer tab, create a sequencer of 1 slot of TSD. More details configuring the ADC IP can be found in [5].

After generating the HDL files, you should include those files in the project. Then the ADC IP core can be instantiated in the design with input/output ports as shown in Figure 3. Refer to [5] to instantiate the module.

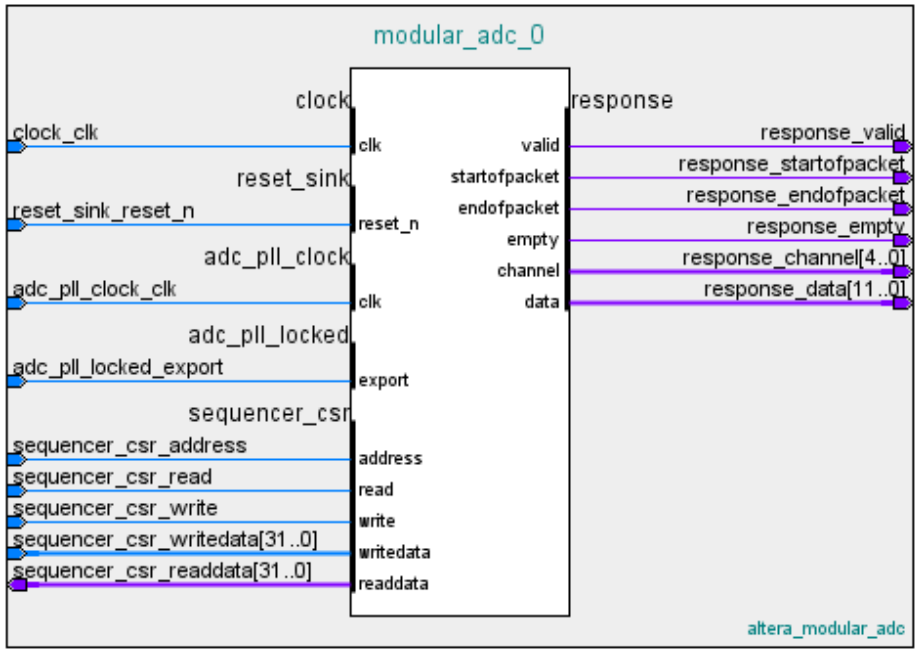

Figure 3 Block symbol for ADC core

Experiment Set-up: Instructions

Part I: Use the Temperature Sensor

In this part, you will use the temperature sensor in the FPGA on the HAHA Board.

Instantiate an ADC core, a PLL and a RAM in your design. Connect them properly. Use the single cycle mode of the sequencer of the ADC. The design should send the temperature data from the ADC to the RAM so that you can read the number. Refer to the Appendix for more details.

You can heat the board to be higher than 40℃, but strictly below 80℃ with a heater, i.e., a hairdryer, and see how the value changes.

Part II: Design a system of your own

Use the FPGA to design a circuit of your own. It can have any functions as you want. The only requirement is that it should receive the temperature data from the ADC (or the RAM). Your design can be a counter which shows its value through LEDs or 7 segment display, or your design can show the temperature on the LEDs in binary, or your design can be an encryption machine that encrypts information, etc.

Part III: Insert a temperature triggered Trojan

Modify your design by inserting a Hardware Trojan into it. The Trojan should be activated by the temperature signal: The Trojan will be triggered when the temperature becomes 40℃ or higher.

Its payload could be any kinds: it could totally halt the original function; or it could induce delays to the design; or it could make the design consume more power, or it could cause information leakage; or it could make the design have other unexpected functions. Whatever the Trojan does, you should be able to observe the malicious effects the Trojan caused.

Your design should come back to normal functions when the devices cool down to be under 40℃.

Alternative Arrangements:

For those who do not have access to a hairdryer:

1. You can try triggering the Trojan at a slightly higher temperature than room temperature, i.e., lowering the triggering temperature. The board should automatically heat-up as it runs, and you can try safely touching the FPGA chip to alter the temperature slightly.
2. You can also try to trigger the Trojan at a lower temperature with respect to room temperature, i.e., reverse the trigger condition (lower temperature triggers the Trojan). You can try using a moisture-isolated ice bag or a cold object to decrease the temperature.

Measurement, Calculation, and Question

Part I: Use the Temperature Sensor

1) Turn in your code for using the ADC. Your code should be able to update the temperature measurement result, i.e., repeating outputting values time after time.
2) What is the output value of the temperature sensor under room temperature? Attach a screenshot showing the content of the RAM.
3) What is the output value of the temperature sensor under 40℃? Attach a screenshot showing the content of the RAM.

Part II: Design a system of your own

1) Describe your design. What is the function? Turn in your Verilog or VHDL code.
2) How does your design receive the temperature data and read it? If you read it from the In-System Memory Content Editor, submit a screenshot. If you read it from LEDs or 7-segment display, submit a photo.
3) There are 49760 LEs (Logic Elements) in the FPGA. How many LEs does your circuit use?

Part III: Insert a temperature triggered Trojan

1) Describe your Trojan. How is it triggered?
2) What is the payload? How do you observe the phenomenon when the Trojan is activated? Attach pictures as needed.
3) After inserting the Trojan, how many LEs does your design use now? How much is the hardware overhead?
4) Turn in your Verilog (or VHDL) description for Part III.
5) What classification does your Trojan belong to? Refer to Figure 2 to answer this question.

Optional Follow-up

Part IV: Modify your Trojan

Modify your Trojan in Part III, its trigger condition will become two successive heating over 10 seconds. It's to say that the Trojan will not be triggered immediately when the temperature becomes higher than 40℃ (or lower if you are performing the alternative arrangement). It will only be triggered after being higher than 40℃ for more than 10 seconds for the ***second time***. Figure 4 shows the trigger condition. At all other times, your design should work normally with the Trojan not triggered. This trigger should occur every time these heating conditions are met.

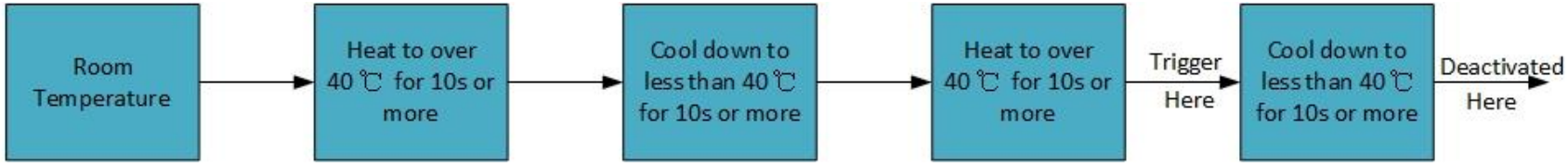

Figure 4 Trojan trigger condition.

1) Is this Trojan a combinational Trojan or a sequential Trojan?
2) What is the hardware overhead for this Trojan?
3) Turn in your Verilog (or VHDL) description file.

☞ Lab Report Guidelines and Demonstration

Deliverables:

1. In your report, answer ALL the questions.
2. Prove you can use the temperature sensor in the FPGA by giving the code and the photos/screenshots when you are using it.
3. Prove your own design works well by giving the code and the photos/screenshots when your system is working.
4. Prove your Trojan can be triggered and affects the system by giving code and photos/screenshots.

Demonstration:

Please take a video as the demo. If you don't do this, you will lose points.

1. Show that you can read the temperature data continuously from the FPGA.
2. Show that the normal function of your design and how "bad things" happen when the board gets heated.

References and Further Reading

[1] http://securityaffairs.co/wordpress/17875/hacking/undetectable-hardware-trojan-reality.html

[2] Bhunia, Swarup, et al. "Hardware Trojan attacks: threat analysis and countermeasures." Proceedings of the IEEE 102.8 (2014): 1229-1247.

[3] Roy, Debapriya Basu, et al. "Reconfigurable LUT: A Double Edged Sword for Security-Critical Applications." International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer International Publishing, 2015.

[4] Majzoub, Sohaib, and Hassan Diab. "Mapping and performance analysis of lookup table implementations on reconfigurable platform." 2007 IEEE/ACS International Conference on Computer Systems and Applications. IEEE, 2007.

[5] https://www.altera.com/en_US/pdfs/literature/hb/max-10/ug_m10_adc.pdf

[6] https://www.youtube.com/watch?v=0oO1RFa-4Xk

[7] https://www.youtube.com/watch?v=3i2LV9SsyPU

[8] https://www.youtube.com/watch?v=qZppbzTgbSI

Appendix

Here is an example of how you connect all the ADC pins. Some pins should be used and connected; some pins don't need to.

```
tsd3 u0 (
      .adc_pll_clock_clk        (c0),                          //  adc_pll_clock.clk
      .adc_pll_locked_export    (c_lock),                      //  adc_pll_locked.export
      .clock_clk                (clk),                         //  clock.clk
      .reset_sink_reset_n       (1'b1),                        //  reset_sink.reset_n
      .response_valid           (response_valid),              //  response.valid
      .response_channel         (),                            //          .channel
      .response_data            (response_data),               //          .data
      .response_startofpacket   (),                            //          .startofpacket
      .response_endofpacket     (),                            //          .endofpacket
      .sequencer_csr_address    (0),                           //  sequencer_csr.address
      .sequencer_csr_read       (1'b0),                        //               .read
      .sequencer_csr_write      (1'b1),                        //               .write
      .sequencer_csr_writedata  (sequencer_csr_writedata),     //               .writedata
      .sequencer_csr_readdata   ()                             //               .readdata
);
```

Explanation for the signal names:

- `c0` is the 10 MHz clock from the PLL
- `c_lock` is one output from the PLL
- `clk` is the 50 MHz clock
- `response_valid`, `response_data` feed to the RAM
- `sequencer_csr_writedata` can be set to 32'h00000003

Instructions for instantiating PLL module:

1) Navigate to **Tools** > **IP Catalog.**
2) Search for ALTPLL and double click on **"ALTPLL"** (not ALTPLL_RECONFIG).
3) Set the inclk0 frequency to **50 MHz** as this is the FPGA system clock. Click **Next**.
4) Uncheck the boxes for all the Optional Inputs. Keep the **Create 'locked' output** box ticked. Click **Next** multiple times until you reach the page titled **c0 – Core/External Clock.**
5) Select **Enter output clock frequency** and set the Requested Setting to **10 MHz**. Click **Finish**.

An example of instantiating the RAM:

```
ram1 U3(
      .address(0),
      .clock(clk),
      .data(response_data),
      .wren(response_valid),
      .q());
```