FPGA Reconfiguration Using Physical Unclonable Functions (PUF)

Yangpingqing Hu, Yuqiu Jiang
Weizhong Wang
Electrical Engineering

BACKGROUND

One of the key requirements for securing communications through public open networks is the ability to authenticate the recipients. In order to block malicious network elements, a network node must validate the identity of the recipient. One of such authentication methods is using unique hardware-dependent keys provided by physical unclonable functions (PUF). The basic idea is to exploit non-reproducible manufacturing variations to provide a device-specific query. Such manufacturing variations are effectively impossible to predict or replicate. A PUF that can be implemented using general-purpose, re-configurable hardware is extremely attractive.

Re-configurable hardware allows us make system more flexible, smaller size, less cost and less power consumption. Partial Reconfiguration is the ability to dynamically modify blocks of logic by downloading partial bit files while the remaining logic continues to operate without interruption. Our design is a secured approach to do partial reconfiguration with minimum use of processor, avoiding potential security problem.

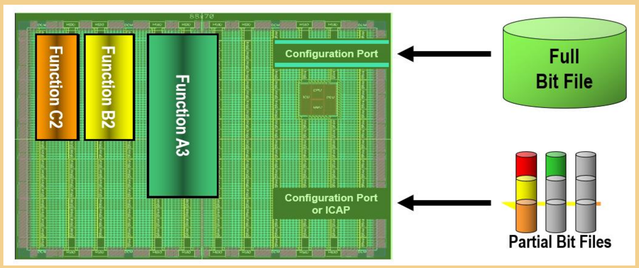

Fig. 1 Partial reconfiguration [1]

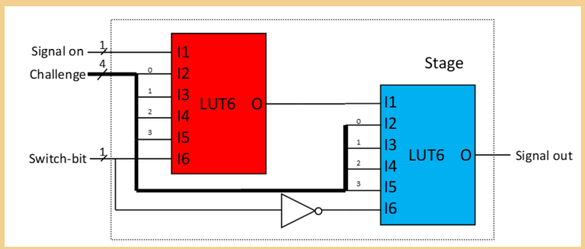

Fig. 2 Structure of LUT pair with two LUT6

METHODOLOGY

Description for our design divides into two parts, the design of RO-based PUF and the implementation of our PUF for PR.

a. PUF

The Physical Unclonable Features (PUFs) provide a strong secure root source for identification and authentication applications. Here, a compact FPGA based PUFs extraction circuits is used for Partial Reconfiguration. The PUFs extraction circuits are solely based on variation inside FPGA LUT (lookup Table) cells.

LUT6 consist RO, which is the main element in the PUF. It has 6 input, 4 of them connected to challenge, 1 to another LUT6 output and 1 to switch-bit. Two such units together is a LUT pair (Fig. 2), whose structure could decrease systematic bias. For a 32-bit RO, 8 LUT pairs are connected in series. For one challenge, RO run two times, with 4 input to the same challenge and the 1-bit switch-bit flip. Counter returns the counts for each run and the result is compared. Response of RO is determined by this.

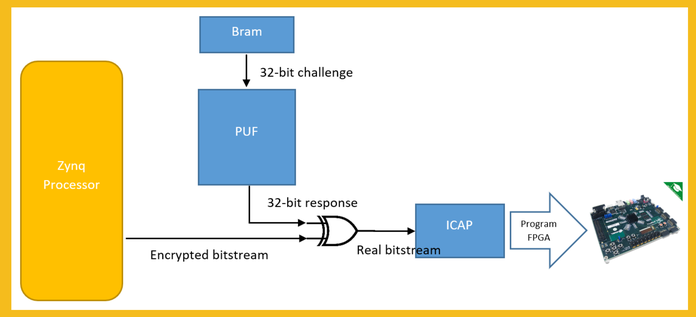

Fig. 3 Structure of the PUF based secure partial reconfiguration

b. Implementation for PR

A prototype circuit is presented in Fig. 3. Our Partial Reconfiguration method specifically uses ICAP (Internal Configuration Access Port) to program the hardware. A FPGA would typically do partial reconfiguration with PCAP (Processor Configuration Access Port), which is directly transmitting bitstream through processor. However, this would reduce the resistance to attack as hacker would more easily steal the data by hacking the processor. More security comes from that challenge for PUF is hidden from processor because it is stored in stand-alone BRAM that cannot be touched by processor.

Each 32-bit challenge is sent to PUF and PUF would generate a 1-bit response, as described before. 32 predetermined challenge are used, thus 32-bit response is fetched. In this prototype, this 32-bit response is the key for decryption, which is just a simple XOR operation. The key could decrypt the encrypted bitstream that is transferred by processor from SD card. And then, the original bitstream is sent to ICAP who is in charge of programming FPGA hardware.

RESULTS

Two aspects of the implementation are most concerned by us: resource utilization and timing.

a. Resource utilization

The PUF part and corresponding support peripherals should not take too many hardware resources, otherwise we are limited to what we can implement for the reconfigurable modules. Our implementation is on Zedboard and the utilization is quite small. So, this design could work on even smaller FPGA chip.

Resources	Utilization	Available	Utilization %
LUT	3496	53176	6.57
LUTRAM	238	17400	1.37
FF	3801	106352	3.57
BRAM	6	140	4.29
IO	0	200	0

b. Timing

The bottleneck for the timing happens to be in the ring oscillator. In previous research on solely ring oscillator, we did comprehensive test respect to timing and other performance of PUF. We have found that the longer it runs, the more reliable the ring is. This is to say, the easier it is to find usable challenge for ring oscillator. With this knowledge, each run for ring oscillator is set as 0.00656 sec long, thus about 0.4198 sec for finish running for 32 challenges. This allows around 88% 32-bit numbers could be used as the predetermined challenges.

CONCLUSION

This design utilizes a RO-based PUF that eliminates systematic bias for FPGA partial reconfiguration. The reliability of the PUF allows us choose challenge easily. Hardware resource utilization is pretty low that it could be applicable to small FGPA hardware. And more importantly, the security is ensured by the use of PUF, as well as the use of ICAP and stand-alone Bram.

Reference

[1] Xilinx, Zynq-7000 SoC Technical Reference Manual, 2018.

[2] Xilinx, Partial Reconfiguration Flow on Zynq using Vivado, 2016.

[3] Filip Kodytek, Robert Lorencz, “A design of ring oscillator based PUF on FPGA”, IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, 2015.

[4] Linus Feiten, Jonathan Oesterle, Tobias Martin, Matthias Sauer, and Bernd Becker, “Systemic Frequency Biases in Ring Oscillator PUFs on FPGAs”, IEEE Transactions on Multi-Scale Computing Systems, Vol. 2, No. 3, July-September 2016.