Trojans Modifying Soft-Processor Instruction Sequences Embedded in FPGA Bitstreams

İsmail San, Nicole Fern, Çetin Kaya Koç and Kwang-Ting (Tim) Cheng

University of California Santa Barbara
Anadolu University

FPL 2016 — August 31, 2016

FPGA Bitstream Security

- *Soft-core processors* implemented using FPGAs are used in many critical embedded systems
  - Ubiquitous computing, e.g. IoT, Avionics, Intellectual Property
- Soft-core processor instructions stored in block memories embedded in bitstream
  - Program codes are usually *infinite loops*: they will continue to execute until the processor is turned off
  - **Usually these instructions are difficult to extract from the bitstream because memory contents are encoded**
- If attacker modifies an FPGA bitstream without disrupting normal design operation, will the modification be detected?
  - Bitstream modification occurs after place and route, so only CRC checksums have the ability to detect modifications and these can be easily disabled[1,2]

---

[1] R. S. Chakraborty et al. "Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream". In: *IEEE Design Test* 2 (2013).

[2] Tim Güneysu, Igor Markov, and André Weimerskirch. "Securely Sealing Multi-FPGA Systems". In: *Proceedings of the 8th Int. Conf. on Reconfigurable Computing: Architectures, Tools and Applications*. 2012.

Attack Scenario

- *Threat Model*
  - Program code performing critical function located in FPGA block RAM
  - Attacker can obtain the bitstream then re-introduce a modified bitsream to the FPGA but has no access to RTL code or original program code
- Our Contributions
  1. Algorithm to decode instructions residing in the FPGA bitstream allowing attacker to reverse engineer the program
  2. Methodology to identify code portions that are involved with some important process, say encryption
  3. Methodology to manipulate the code by injecting a few extra instructions leak information without changing the functionality of the original code

Case Study: Trojan Insertion in AES Instruction Sequence

```
518: 3c020000  lui    v0,0x0
51c: 8c471308  lw     a3,4872(v0)
520: 00042100  sll    a0,a0,0x4
524: 3c020000  lui    v0,0x0
528: 24850004  addiu  a1,a0,4
52c: 24421258  addiu  v0,v0,4696
530: 00452821  addu   a1,v0,a1
534: 24e80010  addiu  t0,a3,16
538: 24a3fffc  addiu  v1,a1,-4
53c: 00e01021  move   v0,a3
540: 90640000  lbu    a0,0(v1)
544: 90460000  lbu    a2,0(v0)
548: 24630001  addiu  v1,v1,1
54c: 00862026  xor    a0,a0,a2
550: 0c00012d  jal    4b4       #UARTWriteByte
554: a0440000  sb     a0,0(v0)
558: 14a3fffa  bne    a1,v1,540 #AddRoundKey+0x28
55c: 24420001  addiu  v0,v0,1
560: 24e70004  addiu  a3,a3,4
564: 14e8fff5  bne    a3,t0,538 #AddRoundKey+0x20
568: 24a50004  addiu  a1,a1,4
56c: 03e00008  jr     ra
570: 00000000  nop
```

Listing 1: AddRoundKey Code Segment

- Code segment from MIPS instruction sequence
- Corresponds to the AddRoundKey step in AES
- Compiled with MIPS cross-compiler toolchain from the C code available online [3]
- The red instruction is the injected jump-and-link instruction to the UART channel write subroutine

[3] https://github.com/kokke/tiny-AES128-C

Properties of the Trojan

- Novelty:
  - *Trojan* CPU instructions are injected by manipulating the block memory contents at the bitstream level
- Strength:
  - Powerful Trojans without extra logic
  - Not possible to trace the trojan insertion during logic synthesis and place-and-route processes
- Caveat:
  - Unencrypted bitstream is needed
  - However, there are practical side-channel attacks on bitstream encryption mechanisms

Concluding Remarks

- Motivation
  - *Cryptographic architectures* or *CPUs* have many fixed values in their design specifications embedded in bitstream
- Key Contributions
  1. **General model** for creating a covert Program code at the Bitstream level
  2. Information transmitted/leaked by *injecting existing instructions only to yield an information leakage* without changing the functionality of the original program code
  3. We avoid most of the existing verification mechanisms since it is introduced after Place & Route