Experiment on Replication of Side Channel Attack via Cache of RISC-V Berkeley Out-of-Order Machine (BOOM) Implemented on FPGA

CARRV 2020

Anh-Tien Le, Ba-Anh Dao, Kuniyasu Suzaki, Cong-Kha Pham

leanhtien@vlsilab.ee.uec.ac.jp

The University of Electro-Communications (UEC), Tokyo, Japan

Contents

1. Introduction
2. Implementation of BOOM on FPGA and benchmarks
3. Replication of side channel attack via cache
4. Conclusion

Introduction

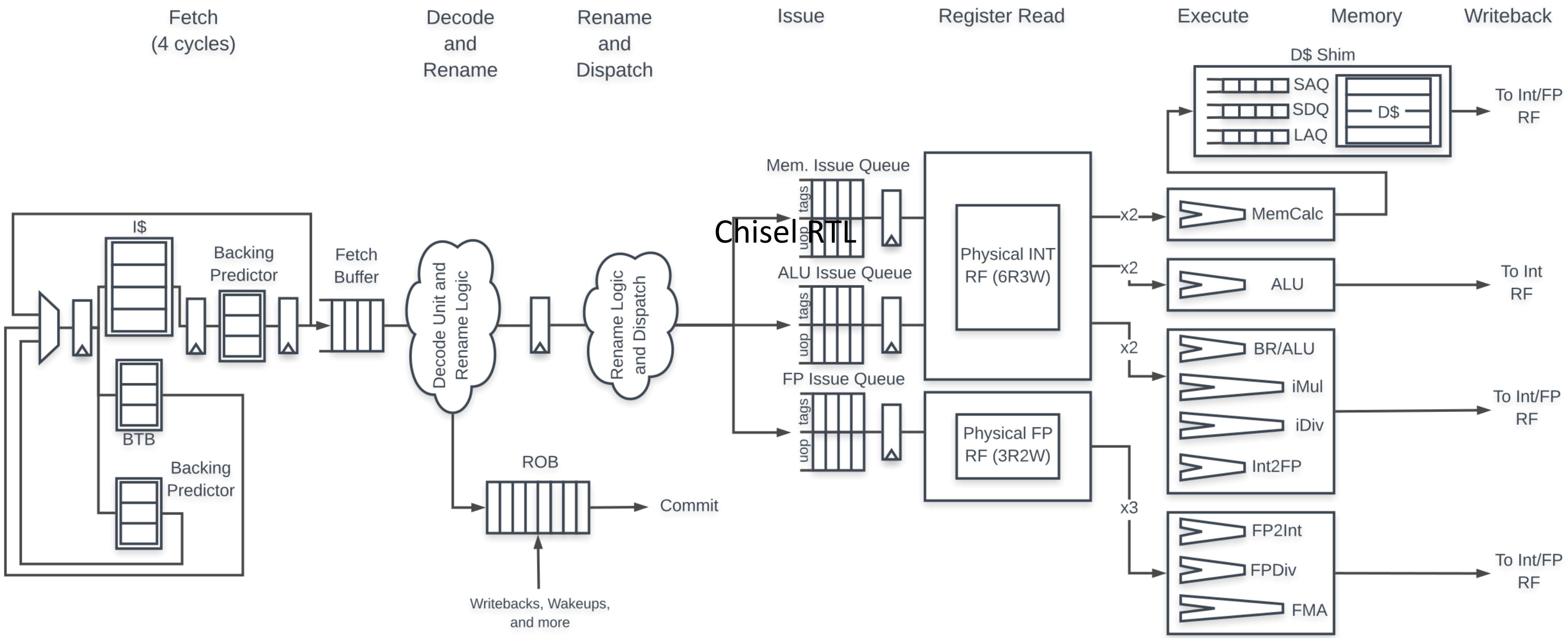
Motivation

- RISC-V Open source approach:
  - Open, free and non-restrictive licenses.
  - Both 32-bit and 64-bit
  - Widely supported.
  - Include open source processor cores, tool chains, simulators and other key supporting components
- Open processor's architecture
  - In-order processor: Rocket Core
  - Out-of-order processor: BOOM (Berkeley Out-of-order Machine)

BOOM (Berkeley Out-of-order Machine)

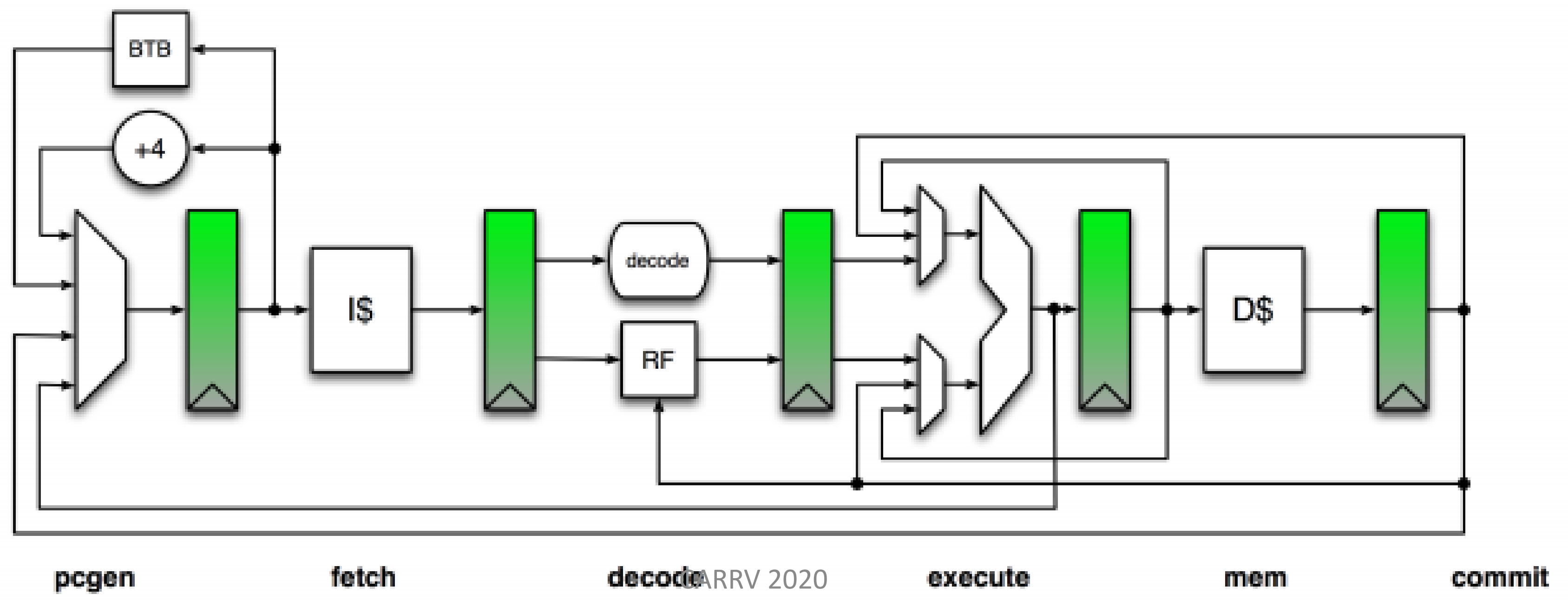
- Open-source synthesizable parameterized out-or-order RISC-V processor.
- Develop and maintain using Chisel RTL.
- Support Linux boot.
- 10-stages pipeline.

BOOM pipeline



Rocket core

- Micro-architecture of an in-order scalar processor.
- Provided with a library of processor components.
- 6-stages pipeline.



Spectre and Meltdown

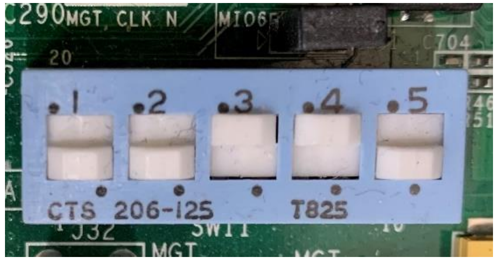
- Spectre
  - Break the isolation between different applications.
  - Take advantage of the speculative execution mechanism.
  - Allow attacker trick applications into leaking secret information.
- Meltdown
  - Break the isolation between user applications and the operating system.
  - Compromise processor when implementing the Out-of-Order execution.
  - Mostly exploit in Intel processors.

Implementation of BOOM on FPGA and benchmarks

FPGA Xilinx SoC ZC706

- Based on the Zynq®-7000 SoC.
- Combine both programmable FPGA circuitry and an ARM-based processor.
- Communicate using HTIF (Host/Target Interface)
- Execute program through ELF (Executable and Linkable Format) format.

ZC706 board



SW11 SD Boot mode

RISC-V Cores

BOOM

- 2-wide single-core
- 64 bit
- L1 Sets: 64
- L1 Ways: 8
- BTB Sets 512
- BTB Banks 2
- BTB Ways 4

Rocket

- Single-core
- 64 bit

Benchmark – Coremark

- Developed by the Embedded Microprocessor Benchmark Consortium (EEMBC).
- Coremark's score: the number of iterations per second with seeds of 0,0,0x66, and the buffer size of 2000 bytes.

	Iterations/s
RV64 Boom two-wide	3,737
RV64 Rocket	2,181

Replication of Side Channel Attack via Cache

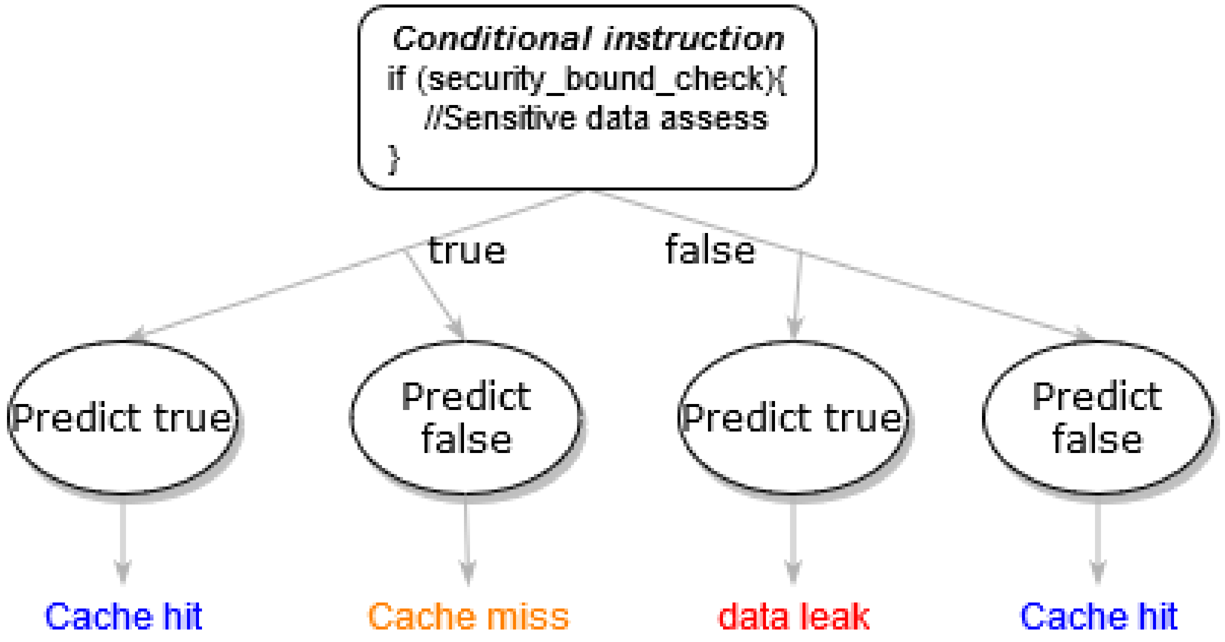
Spectre attacks

- Attack mechanisms:
  - Conditional branch
  - Indirect jump
  - Return instructions
  - Speculative store bypass
  - Data speculation
  - …

- BOOM’s features:
  - Branch Predictor Unit
  - Speculative Execution
  - Caching

Bound check bypass attack

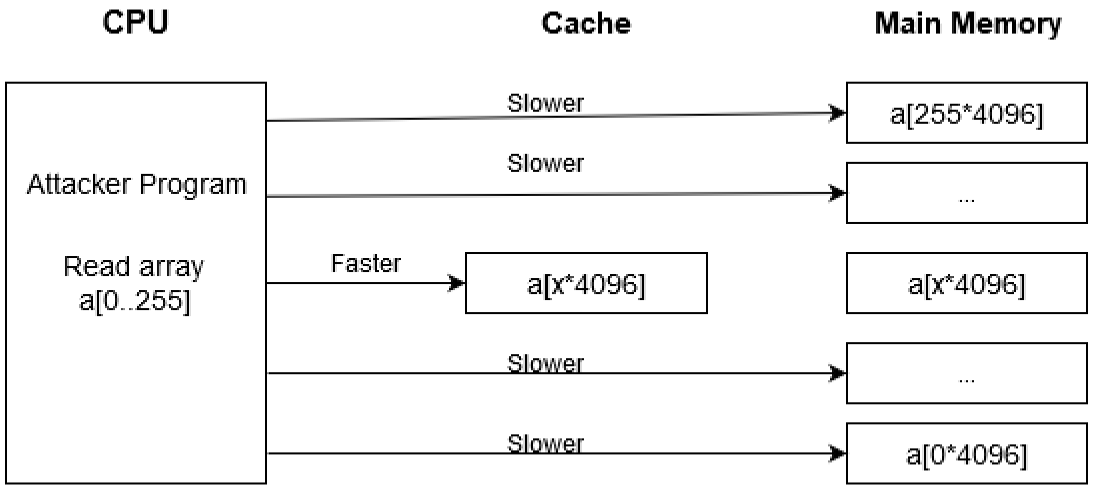
- Focus on branch misprediction.
- Exploit visible effects in the cache.



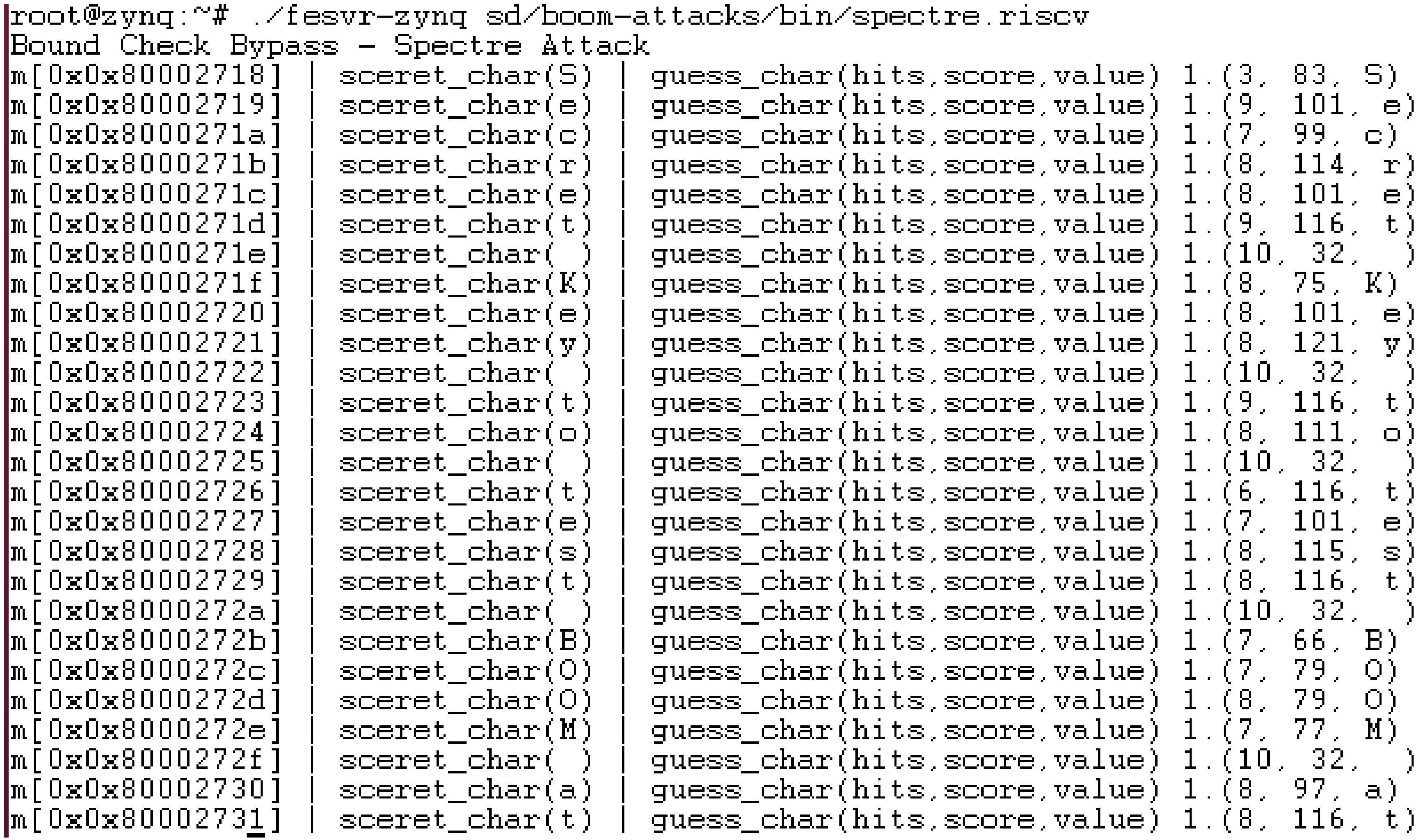
Attack scenario

FLUSH+RELOAD technique

- Step 1: Flush shared address in the memory cache
- Step 2: Wait for the victim function access the sensitive data
- Step 3: Reload to find out which element's loading time is faster than others



Results



```
root@zynq:~# ./fesvr-zynq sd/boom-attacks/bin/spectre.riscv
Bound Check Bypass - Spectre Attack
m[0x0x80002718] | sceret_char(S) | guess_char(hits,score,value) 1.(3, 83, S)
m[0x0x80002719] | sceret_char(e) | guess_char(hits,score,value) 1.(9, 101, e)
m[0x0x8000271a] | sceret_char(c) | guess_char(hits,score,value) 1.(7, 99, c)
m[0x0x8000271b] | sceret_char(r) | guess_char(hits,score,value) 1.(8, 114, r)
m[0x0x8000271c] | sceret_char(e) | guess_char(hits,score,value) 1.(8, 101, e)
m[0x0x8000271d] | sceret_char(t) | guess_char(hits,score,value) 1.(9, 116, t)
m[0x0x8000271e] | sceret_char( ) | guess_char(hits,score,value) 1.(10, 32,  )
m[0x0x8000271f] | sceret_char(K) | guess_char(hits,score,value) 1.(8, 75, K)
m[0x0x80002720] | sceret_char(e) | guess_char(hits,score,value) 1.(8, 101, e)
m[0x0x80002721] | sceret_char(y) | guess_char(hits,score,value) 1.(8, 121, y)
m[0x0x80002722] | sceret_char( ) | guess_char(hits,score,value) 1.(10, 32,  )
m[0x0x80002723] | sceret_char(t) | guess_char(hits,score,value) 1.(9, 116, t)
m[0x0x80002724] | sceret_char(o) | guess_char(hits,score,value) 1.(8, 111, o)
m[0x0x80002725] | sceret_char( ) | guess_char(hits,score,value) 1.(10, 32,  )
m[0x0x80002726] | sceret_char(t) | guess_char(hits,score,value) 1.(6, 116, t)
m[0x0x80002727] | sceret_char(e) | guess_char(hits,score,value) 1.(7, 101, e)
m[0x0x80002728] | sceret_char(s) | guess_char(hits,score,value) 1.(8, 115, s)
m[0x0x80002729] | sceret_char(t) | guess_char(hits,score,value) 1.(8, 116, t)
m[0x0x8000272a] | sceret_char( ) | guess_char(hits,score,value) 1.(10, 32,  )
m[0x0x8000272b] | sceret_char(B) | guess_char(hits,score,value) 1.(7, 66, B)
m[0x0x8000272c] | sceret_char(O) | guess_char(hits,score,value) 1.(7, 79, O)
m[0x0x8000272d] | sceret_char(O) | guess_char(hits,score,value) 1.(8, 79, O)
m[0x0x8000272e] | sceret_char(M) | guess_char(hits,score,value) 1.(7, 77, M)
m[0x0x8000272f] | sceret_char( ) | guess_char(hits,score,value) 1.(10, 32,  )
m[0x0x80002730] | sceret_char(a) | guess_char(hits,score,value) 1.(8, 97, a)
m[0x0x80002731] | sceret_char(t) | guess_char(hits,score,value) 1.(8, 116, t)
```

Conclusion

Conclusion

- Demonstrate the BOOM implementation
    - Run on physical FPGA ZC706
    - Proved that the in-order processor’s performance, the Rocket Core, was outplayed by the Berkeley Out-of-Order processor.
- Demonstrate side-channel attack technique that exposes the caching effect of Out-of-order RISC-V processor.

A heterogeneous multi-core processor

- Rocket Core does not issue data-memory accesses speculatively.

=> Hybrid core: Rocket Core + BOOM Core.

  - Developed to use the same open-source RocketChip SoC generator.
  - Take advantage of both processor's characteristics.

Thank you for listening!

Q&A