DiFuzzRTL: Differential Fuzz Testing to Find CPU Bugs

Jaewon Hur, Suhwan Song, Dongup Kwon, Eunjin Baek, Jangwoo Kim, Byoungyoung Lee

Computer Security Lab & High Performance Computer System Lab
Seoul National University

{hurjaewon, sshkeb96, dongup, ebaek, jangwoo, byoungyoung} @snu.ac.kr
Pentium FDIV bug
Wwaaaaa.. $475 million
Continued Verification

CPU vendors invest huge efforts into the **Functional verification**
Check if **CPU RTL design** correctly follows **ISA**
Complete Verification is Difficult

CPU vendors invest huge efforts into the **Functional verification**

Check if CPU RTL design correctly follows ISA

So we keep observing CPU bugs
Complete Verification is Difficult

CPU vendors invest huge efforts into the **Functional verification**
Check if CPU RTL design correctly follows ISA

So we keep observing CPU bugs
Complete Verification is Difficult

Pentium FDiv: The processor bug that shook the world
By Desire Athow, October 30, 2014
20 years already

The Ryzen 3000 Boot Problem With Newer Linux Distros Might Be Due To RdRand Issue
Written by Michael Larabel in AMD on 8 July 2019 at 09:42 AM EDT. 121 Comments

As outlined yesterday, AMD’s Ryzen 3000 processors are very fast but having issues booting newer Linux distributions. The exact issue causing that boot issue on 2019 Linux distribution releases doesn’t appear to be firmly resolved yet but some are believing it is an RdRand instruction issue on these newer processors manifested by systemd.

So we keep observing CPU bugs...
Complete Verification is Difficult

Pentium FDIV: The processor bug that shook the world
By Desire Athow  October 30, 2014
20 years already

The Ryzen 3000 Boot Problem With Newer Linux Distros Might Be Due To RdRand Issue
Written by Michael Larabel in AMD on 8 July 2019 at 09:42 AM EDT. 121 Comments

As outlined yesterday, AMD's Ryzen 3000 processors are very fast but having issues. The exact issue causing that boot issue on 2019 appear to be firmly resolved yet but some are on issue on these newer processors manifested by

So we keep observing CPU bugs
Complete Verification is Difficult

Pentium FDIV: The processor bug that shook the world
By Desire Athow  October 30, 2014
20 years already

The Ryzen 3000 Boot Problem Could Be Due To RdRand Issue
Written by Michael Larabel in AMD on 8 July 2019

Skylake bug causes Intel chips to freeze under ‘complex workloads’
By Joel Hrusko on January 11, 2016 at 4:22 pm

Inte has disclosed that its sixth-generation Core products (known as Skylake) suffer from a CPU bug that can cause a system to hang. The company has only publicly identified one application family that causes it, Prime95.

So we keep observing CPU bugs
DiFuzzRTL: Differential Fuzz Testing to Find CPU Bugs
DiFuzzRTL Found Real-world CPU Bugs

We found 16 real-world bugs in OpenRISC and RISC-V CPUs
DiFuzzRTL Fuzzes CPU RTL Designs

What does the Fuzzer do?
DiFuzzRTL Fuzzes CPU RTL Designs

What does the Fuzzer do?

Mutate inputs guided by a coverage!
DiFuzzRTL Fuzzes CPU RTL Designs

What does the Fuzzer do?

Mutate inputs guided by a coverage!
DiFuzzRTL Fuzzes CPU RTL Designs

How to fuzz CPU RTL designs?
DiFuzzRTL Fuzzes CPU RTL Designs

How to fuzz CPU RTL designs?

✓ Requirement 1. Framework for detecting the CPU bugs
DiFuzzRTL Fuzzes CPU RTL Designs

How to fuzz CPU RTL designs?

✓ Requirement 1. Framework for detecting the CPU bugs
✓ Requirement 2. New coverage definition for the RTL designs
[1] Framework for detecting the CPU bugs
Detecting CPU Bugs

CPU bug

Abnormal CPU behavior
Different from a predefined ISA

Framework for detecting the CPU bugs
Detecting CPU Bugs

CPU bug

Abnormal CPU behavior
Different from a predefined ISA

✓ misaligned lr instruction

Framework for detecting the CPU bugs
Detecting CPU Bugs

CPU bug

Abnormal CPU behavior
Different from a predefined ISA

✓ misaligned lr instruction

Testing CPU RTL design alone cannot detect bugs

Framework for detecting the CPU bugs
Differential Testing Framework

Detects CPU bugs by comparing with the ISA simulator

ISA simulator – Software implementation of the ISA

Framework for detecting the CPU bugs
Differential Testing Framework

Detects CPU bugs by comparing with the ISA simulator

ISA simulator – Software implementation of the ISA

Results should be same
Differential Testing Framework

Detects CPU bugs by comparing with the ISA simulator

ISA simulator – Software implementation of the ISA

Results should be same
Differential Testing Framework

Detects CPU bugs by comparing with the ISA simulator

ISA simulator – Software implementation of the ISA

Differential testing

Automatically detects CPU bugs by comparing the results

Results should be same
Testing CPU RTL Design and ISA simulator

Defines SimInput for a unified input space

SimInput – Fuzz input containing instruction, data, and interrupt

Differential testing

Results should be same

DiFuzzRTL approach

Framework for detecting the CPU bugs
Testing CPU RTL Design and ISA simulator

Defines SimInput for a unified input space

SimInput – Fuzz input containing instruction, data, and interrupt

Results should be same

Differential testing

Framework for detecting the CPU bugs
Testing CPU RTL Design and ISA simulator

Defines SimInput for a unified input space

SimInput – Fuzz input containing instruction, data, and interrupt

Results should be same

Differential testing

Framework for detecting the CPU bugs
Differential testing

Testing CPU RTL Design and ISA simulator

Defines SimInput for a unified input space

SimInput – Fuzz input containing instruction, data, and interrupt

Generates SimInput and tests both CPU and ISA simulator

Results should be same

Framework for detecting the CPU bugs
[2] New coverage definition for the RTL designs
Coverage for RTL Design

RTL (Register Transfer Level)?
Abstraction to describe hardware circuit implementation
Coverage for RTL Design

RTL (Register Transfer Level)?
Abstraction to describe hardware circuit implementation

New coverage definition for the RTL designs
Coverage for RTL Design

RTL (Register Transfer Level)?
Abstraction to describe hardware circuit implementation

Finite State Machine (FSM)

New coverage definition for the RTL designs
Coverage for RTL Design

Verification goal?
Explore as many states in the FSM

Finite State Machine (FSM)

New coverage definition for the RTL designs
Coverage for RTL Design

Verification goal?
Explore as many states in the FSM

Coverage should guide the input to find new states

FSM Modeling

Finite State Machine (FSM)

New coverage definition for the RTL designs
Limitation of Previous Coverage Measures

- **Branch coverage** [Vineeth et al. ETS’15], [Alif et al. DATE’18]
- **MUX control coverage** [Kevin et al. ICCAD’18]
- **FSM coverage** [Dinos et al. TC’98], [Jian et al. TCAD’15]

New coverage definition for the RTL designs
Limitation of Previous Coverage Measures

- **Branch coverage** [Vineeth et al. ETS’15], [Alif et al. DATE’18] Not accurate
- **MUX control coverage** [Kevin et al. ICCAD’18]
- **FSM coverage** [Dinos et al. TC’98], [Jian et al. TCAD’15]

Cannot capture FSM

New coverage definition for the RTL designs
Limitation of Previous Coverage Measures

- **Branch coverage**  
  [Vineeth et al. ETS’15], [Alif et al. DATE’18]  
  Not accurate

- **MUX control coverage**  
  [Kevin et al. ICCAD’18]  
  Not efficient

- **FSM coverage**  
  [Dinos et al. TC’98], [Jian et al. TCAD’15]

Cannot capture FSM  
Incurs large instrument overhead

New coverage definition for the RTL designs
Limitation of Previous Coverage Measures

- **Branch coverage** [Vineeth et al. ETS’15], [Alif et al. DATE’18] Not accurate
- **MUX control coverage** [Kevin et al. ICCAD’18] Not efficient
- **FSM coverage** [Dinos et al. TC’98], [Jian et al. TCAD’15] Not automatic

Cannot capture FSM Incurs large instrument overhead Needs manual efforts

New coverage definition for the RTL designs
Register Coverage: New Coverage for RTL
Register Coverage: New Coverage for RTL

**Accurate:** correctly captures FSM exploration
Register Coverage: New Coverage for RTL

**Accurate:** correctly captures FSM exploration

**Efficient:** incurs only 7% runtime overhead
Register Coverage: New Coverage for RTL

**Accurate:** correctly captures FSM exploration

**Efficient:** incurs only 7% runtime overhead

**Automatic:** requires no manual effort from developers

New coverage definition for the RTL designs
Capturing FSM Exploration

Monitors registers to correctly capture the FSM exploration

Finite State Machine (FSM)

New coverage definition for the RTL designs
Capturing FSM Exploration

Monitors registers to correctly capture the FSM exploration

Finite State Machine (FSM)

New coverage definition for the RTL designs
Capturing FSM Exploration

Monitors registers to correctly capture the FSM exploration

DiFuzzRTL approach

New coverage definition for the RTL designs
Capturing FSM Exploration

Monitors registers to correctly capture the FSM exploration

DiFuzzRTL approach

New coverage definition for the RTL designs
DiFuzzRTL approach

Capturing FSM Exploration

Monitors registers to correctly capture the FSM exploration

<table>
<thead>
<tr>
<th>Cycle</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_F</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>input_S</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Regstate: \( \{B_F, P_S\} \)

Finite State Machine (FSM)

New coverage definition for the RTL designs
Capturing FSM Exploration

Monitors registers to correctly capture the FSM exploration

DiFuzzRTL approach

New coverage definition for the RTL designs
Monitoring Control Register

Improves efficiency by monitoring only control registers
Control register – Registers wired into MUX select signal

<table>
<thead>
<tr>
<th>CPU</th>
<th>Number of registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>258</td>
</tr>
<tr>
<td>Rocket</td>
<td>1,300</td>
</tr>
<tr>
<td>Boom</td>
<td>4,900</td>
</tr>
</tbody>
</table>

New coverage definition for the RTL designs
Monitoring Control Register

Improves efficiency by monitoring only control registers
Control register – Registers wired into MUX select signal

<table>
<thead>
<tr>
<th>CPU</th>
<th>Number of registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>258</td>
</tr>
<tr>
<td>Rocket</td>
<td>1,300</td>
</tr>
<tr>
<td>Boom</td>
<td>4,900</td>
</tr>
</tbody>
</table>

New coverage definition for the RTL designs
Monitoring Control Register

Improves efficiency by monitoring only control registers
Control register – Registers wired into MUX select signal

<table>
<thead>
<tr>
<th>CPU</th>
<th>Number of registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>258</td>
</tr>
<tr>
<td>Rocket</td>
<td>1,300</td>
</tr>
<tr>
<td>Boom</td>
<td>4,900</td>
</tr>
</tbody>
</table>

DiFuzzRTL approach

New coverage definition for the RTL designs
Monitoring Control Register

Improves efficiency by monitoring only control registers
Control register – Registers wired into MUX select signal

<table>
<thead>
<tr>
<th>CPU</th>
<th>Number of registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>258</td>
</tr>
<tr>
<td>Rocket</td>
<td>1,300</td>
</tr>
<tr>
<td>Boom</td>
<td>4,900</td>
</tr>
</tbody>
</table>

New coverage definition for the RTL designs
Monitoring Control Register

Improves efficiency by monitoring only control registers
Control register – Registers wired into MUX select signal

<table>
<thead>
<tr>
<th>CPU</th>
<th>Number of registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>258</td>
</tr>
<tr>
<td>Rocket</td>
<td>1,300</td>
</tr>
<tr>
<td>Boom</td>
<td>4,900</td>
</tr>
</tbody>
</table>

DiFuzzRTL approach

New coverage definition for the RTL designs
Monitoring Control Register

Improves efficiency by monitoring only control registers
Control register – Registers wired into MUX select signal

<table>
<thead>
<tr>
<th>CPU</th>
<th>Number of registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>258</td>
</tr>
<tr>
<td>Rocket</td>
<td>1,300</td>
</tr>
<tr>
<td>Boom</td>
<td>4,900</td>
</tr>
</tbody>
</table>

New coverage definition for the RTL designs
Monitoring Control Register

Improves efficiency by monitoring only control registers

Control register – Registers wired into MUX select signal

<table>
<thead>
<tr>
<th>CPU</th>
<th>Number of registers</th>
<th>Number of control registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>258</td>
<td>90</td>
</tr>
<tr>
<td>Rocket</td>
<td>1,300</td>
<td>207</td>
</tr>
<tr>
<td>Boom</td>
<td>4,900</td>
<td>330</td>
</tr>
</tbody>
</table>

New coverage definition for the RTL designs
Monitoring Control Register

Improves efficiency by monitoring only control registers
Control register – Registers wired into MUX select signal

<table>
<thead>
<tr>
<th>CPU</th>
<th>Number of registers</th>
<th>Number of control registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>1,300</td>
<td>207</td>
</tr>
<tr>
<td>Rocket</td>
<td>4,900</td>
<td>330</td>
</tr>
</tbody>
</table>

Automatically identifies the control registers

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

1. Efficient new state identification

New coverage definition for the RTL designs
DiFuzzRTL approach

**RTL-based Coverage Instrumentation**

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

1. Efficient new state identification

New coverage definition for the RTL designs
DiFuzzRTL approach

RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

1. Efficient new state identification

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

1. Efficient new state identification

DiFuzzRTL approach

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

1. Efficient new state identification

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

1. Efficient new state identification

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

DiFuzzRTL approach

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations
Module-based coverage map instrumentation

2. Scalable coverage computation

Sum of covsum

New coverage definition for the RTL designs
RTL-based Coverage Instrumentation

Efficiently computes the number of new state explorations

Module-based coverage map instrumentation

Automatically instruments *regstate, covsum and covmap*

2. Scalable coverage computation

New coverage definition for the RTL designs
DiFuzzRTL

Accurate, Efficient, and Automatic fuzzer to find CPU bugs

- Coverage-guided input generation
- Automatic testing and bug detection
Implementation & Evaluation Setup

• Prototype with three CPU RTL designs: Mor1kx (OpenRISC), Rocket, and Boom (RISC-V)
Implementation & Evaluation Setup

- Prototype with three CPU RTL designs: Mor1kx (OpenRISC), Rocket, and Boom (RISC-V)

- RTL testing environments: Software simulation, and FPGA prototyping
What DifuzzRTL Found?

- Found **16 new CPU bugs**
  - 6 of those were assigned with CVE numbers.
- Showed the effectiveness of DiFuzzRTL
  - Case study with *Issue #492 (invalid rm bug)* and *CVE-2020-29561 (misaligned lr bug)*

<table>
<thead>
<tr>
<th>Project</th>
<th>ISA</th>
<th>Bug ID</th>
<th>Description</th>
<th>Confirmed</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mork1x</td>
<td>OpenRISC</td>
<td>CVE-2020-13455</td>
<td>Reservation is not cancelled when there is snooping hit between hva and swa</td>
<td>✓</td>
<td>pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CVE-2020-13454</td>
<td>Jump to link register does not assert illegal instruction exception</td>
<td>✓</td>
<td>pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CVE-2020-13453</td>
<td>Misaligned swa raise exception when reservation is not set</td>
<td>✓</td>
<td>pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #114</td>
<td>li.fl1, li.f1 instruction decoding bug</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #99</td>
<td>eax register not saving instruction virtual address when illegal instruction exception</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rocket chip</td>
<td>RISCV</td>
<td>Issue #2345</td>
<td>Instruction retired count not increased when ebreak</td>
<td>✓</td>
<td>pending</td>
</tr>
<tr>
<td>Boom</td>
<td>RISCV</td>
<td>CVE-2020-13251</td>
<td>Source field in ProbeAckData does not match the sink field of ProbeRequest</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #458</td>
<td>Floating point instruction which has invalid rm field does not raise exception</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #454</td>
<td>FS bits in mstatus register is set after fl.e, d instruction</td>
<td>✓</td>
<td>pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #492</td>
<td>When fmr is set DYN, floating point instruction with DYN rm field should raise exception</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #493</td>
<td>Rounding mode in fsqrt instruction does not work</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #503</td>
<td>invalid operation flag is not set after invalid fdiv instruction</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CVE-2020-29561</td>
<td>Misaligned lr instruction on a cached line set the reservation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Spike</td>
<td>RISCV</td>
<td>CVE-2020-13456</td>
<td>Misaligned lr.d should not set load reservation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #2390</td>
<td>Reading dpc register should raise exception in machine mode</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue #426</td>
<td>Faulting virtual address should not be written to mtrval when ebreak</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bug ID</th>
<th>Elapsed time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>riscv-torture</td>
</tr>
<tr>
<td>Issue #458</td>
<td>118</td>
</tr>
<tr>
<td>Issue #504</td>
<td>x</td>
</tr>
</tbody>
</table>

*Not able to reproduce bug*
Future Use Cases
Future Use Cases

• Detecting micro-architectural side channels, e.g., Spectre, Meltdown
Future Use Cases

• Detecting micro-architectural side channels, e.g., Spectre, Meltdown

• Fuzzing an entire SoC with DiFuzzRTL, e.g., memory consistency bug
Conclusion
Conclusion

- DiFuzzRTL, an accurate, efficient, and automatic fuzzer for CPU RTL designs
Conclusion

• DiFuzzRTL, an accurate, efficient, and automatic fuzzer for CPU RTL designs
• We found several real-world bugs with DiFuzzRTL

<table>
<thead>
<tr>
<th>CPU</th>
<th>Bug ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mor1kx</td>
<td>CVE-2020-13455, 2020-13453, 2020-13454 Issue 114, 99</td>
</tr>
<tr>
<td>Rocket</td>
<td>Issue 2345</td>
</tr>
<tr>
<td>Boom</td>
<td>CVE 2020-13251, 2020-29561 Issue 458, 454, 492, 493, 503</td>
</tr>
<tr>
<td>Spike</td>
<td>CVE-2020-13456 Issue 426, 2390</td>
</tr>
</tbody>
</table>
Thank you