FireSim Intensive
Chisel Community Conference 2018
Speaker: Alon Amid
Agenda

- FireSim Debugging Using Software Simulation
- FireSim Debugging Using Integrated Logic Analyzers
- Advanced FPGA-Based Debugging and Profiling Features
- The FireSim Vision for Debugging and Profiling
Debugging Using Software Simulation

My FireSim Simulation Is Not Working

What Am I doing?

- Modifying internal simulated target hardware, no new external endpoints
- Adding/Modifying new interfaces and endpoints, modifying simulation models

Target-Level SW Simulation

Simulator-Level SW Simulation

Midas-Level SW Simulation

FPGA-Level SW Simulation
Debugging Using Software Simulation

Target-Level Simulation
- Software Simulation
- Target Design Untransformed
- No Host-FPGA interfaces

MIDAS-Level Simulation
- Software Simulation
- Target Design Transformed by MIDAS
- Host-FPGA interfaces/shell emulated using abstract models

FPGA-Level Simulation
- Software Simulation
- Target Design Transformed by MIDAS
- Host-FPGA interfaces/shell simulated by the FPGA tools
Remember this slide from Sagar’s presentation?
Debugging Using Software Simulation

“FAME-1” Transformed RTL Design

RTL Design

FPGA Fabric

<- Resp Queue

 Req Queue ->

DRAM Model

100 cycle latency

Mem Channel

Physical DRAM

100ns latency

Target-Level SW Simulation
Debugging Using Software Simulation

“FAME-1” Transformed RTL Design

RTL Design

<- Resp Queue

DRAM Model

100 cycle latency

Req Queue ->

FPGA Fabric

MIDAS-Level SW Simulation

Target-Level SW Simulation

Mem Channel

100ns latency

FAME-1 Model

100ns latency
Debugging Using Software Simulation

“FAME-1” Transformed RTL Design

RTL Design

<- Resp Queue

Req Queue ->

DRAM Model

100 cycle latency

FPGA Fabric

Target-Level SW Simulation
MIDAS-Level SW Simulation
FPGA-Level SW Simulation

Berkeley Architecture Research
## Debugging Using Software Simulation

<table>
<thead>
<tr>
<th>Level</th>
<th>Waves</th>
<th>VCS</th>
<th>Verilator</th>
<th>XSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Off</td>
<td>~5 kHz</td>
<td>~5 kHz</td>
<td>N/A</td>
</tr>
<tr>
<td>Target</td>
<td>On</td>
<td>~1 kHz</td>
<td>~5 kHz</td>
<td>N/A</td>
</tr>
<tr>
<td>MIDAS</td>
<td>Off</td>
<td>~4 kHz</td>
<td>~2 kHz</td>
<td>N/A</td>
</tr>
<tr>
<td>MIDAS</td>
<td>On</td>
<td>~3 kHz</td>
<td>~1 kHz</td>
<td>N/A</td>
</tr>
<tr>
<td>FPGA</td>
<td>On</td>
<td>~2 Hz</td>
<td>N/A</td>
<td>~0.5 Hz</td>
</tr>
</tbody>
</table>
Debugging Using Software Simulation

• **Target-Level Simulation**
  
  In firesim/target-design/firechip/vsim
  
  $ make DESIGN=<YourDesign> CONFIG=<YourConfig> debug
  
  $ ./simv=<YourDesign>-<YourConfig>-debug +max-cycles=50000 +vcdplusfile=<WaveformFileName>.vpd
  
  ../tests/<InputTest>.riscv

• **MIDAS-level Simulation**
  
  In firesim/sim
  
  $ make <verilator|vcs>-debug
  
  $ make EMUL=<verilator|vcs> DESIGN=FireSimNoNIC run-asm-test-debug

• **FPGA-Level Simulation**
  
  In firesim/sim
  
  $ make xsim
  
  $ make xsim-dut <VCS=1> &
  
  $ make run-xsim SIM_BINARY=<PATH/TO/DRIVER/BINARY/FOR/TARGET/TO/RUN>

**FPGA-level simulation currently does not support DMA_PCIS (which is used for the NIC interface)**
“Everything looks OK in SW simulation, but there is still a bug somewhere”

“My bug only appears after hours of running Linux on my simulated HW”
Integrated Logic Analyzers (ILAs)

- Common debugging feature provided by FPGA vendors
- Continuous recording of a sampling window of up to 1024 cycles.
  - Stores recorded samples in BRAM.
- Realtime trigger-based sampled output of probed signals
  - Multiple probes ports can be combined to a single trigger
  - Trigger can be in any location within the sampling window
- On the AWS F1-Instances, ILA interfaced through a debug-bridge and server

From: aws-fpga cl_hello_world example
Debugging Using Integrated Logic Analyzers

AutoILA – Automation of ILA integration with FireSim
• Annotate requested signals and bundles in the Chisel source code
• Automatic configuration and generation of the ILA IP in the FPGA toolchain
• Automatic expansion and wiring of annotated signals to the top level of a design using a FIRRTL transform.
• Remote waveform and trigger setup from the manager instance
### Debugging using Integrated Logic Analyzers

<table>
<thead>
<tr>
<th>Pros:</th>
<th>Cons:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• No emulated parts – what you see is what’s running on the FPGA</td>
<td>• Requires a full build to modify visible signals/triggers (takes several hours)</td>
</tr>
<tr>
<td>• FPGA simulation speed - $O$(MHz) compared to $O$(KHz) in software simulation</td>
<td>• Limited sampling window size</td>
</tr>
<tr>
<td>• Real-time trigger-based</td>
<td>• Consumes FPGA resources</td>
</tr>
</tbody>
</table>
Advanced FPGA-Based Debugging Features

• FPGA-based simulation enables high simulation speed, which enables advanced debugging and profiling tools.

• Reach “deep” in simulation time, and obtain large levels of coverage and data

• Examples:
  • TracerV
  • Synthesizable Assertions
TracerV

- **Out-of-band** full instruction execution trace
- MIDAS widget connected to target trace ports
- By default, large amount of info wired out of Rocket/BOOM, per-hart, per-cycle:
  - Instruction Address
  - Instruction
  - Privilege Level
  - Exception/Interrupt Status, Cause
- TracerV can rapidly generate several TB of data.
TracerV

• Out-of-Band software-level debugging and profiling: Profiling does not perturb execution
• Useful for kernel and hypervisor level cycle-sensitive profiling
• Examples:
  • Co-Optimization of NIC and Network Driver
  • Keystone Secure Enclave Project
  • High-performance hardware-specific code (supercomputing?)
• Requires large-scale analytics for insightful profiling and optimization.
TracerV

Pros:
• Out-of-Band (no impact on workload execution)
• SW-centric method
• Large amounts of data

Cons:
• Slower simulation performance (40 MHz)
• No HW visibility
• Large amounts of data
Synthesizable Assertions

• Assertions – rapid error checking embedded in HW source code.
  • Commonly used in SW Simulation
  • Halts the simulation upon a triggered assertion. Represented as a “stop” statement in FIRRTL
  • By defaults, emitted as non-synthesizable SV functions ($fatal)

From: Trillion-Cycle Bug Finding Using FPGA-Accelerated Simulation Donggyu Kim, Christopher Celio, Sagar Karandikar, David Biancolin, Jonathan Bachrach, Krste Asanović. ADEPT Winter Retreat 2018

From: BROOM: An open-source Out-of-Order processor with resilient low-voltage operation in 28nm CMOS, Christopher Celio, Pi-Feng Chiu, Krste Asanovic, David Patterson and Borivoje Nikolic. HotChip 30, 2018
Synthesizable Assertions

• Synthesizable Assertions on FPGA
  • Transform FIRRTL stop statements into synthesizable logic
  • Insert combinational logic and signals for the stop condition arguments
  • Insert encodings for each assertion (for matching error statements in SW)
  • Wire the assertion logic output to the Top-Level
  • Generate timing tokens for cycle-exact assertions
  • Assertion checker records the cycle and halts simulation when assertion is triggered
Synthesizable Assertions

- Previously a research feature presented in DESSERT [1]
- Helped Identify BOOM bugs trillions of cycles into execution
- Integrated into FireSim in the latest release.

Synthesizable Assertions

Pros:
- FPGA simulation speed
- Real-time trigger-based
- Consumes small amount of FPGA resources (compared to ILA)
- Key signals have pre-written assertions in re-usable components/libraries

Cons:
- Low visibility: No waveform/state
- Assertions are best added while writing source RTL rather than during “investigative” debugging
The FireSim Vision: Speed and Visibility

• High-performance simulation
• Full application workloads
• Tunable visibility & resolution
• Unique data-based insights
Speed and Visibility – How do we get there?

• Integration of additional DESSERT features
  • Hardware state snapshot extraction from FPGA to software simulation using arbitrary software triggers
  • Easy-to-use information transfer between execution traces and software hooks

• Data-Processing pipeline for insights from large-scale traces
  • O(GB)-O(TB) out-of-band logs require big-data analysis methods for insights (Golden model comparison is one potential method)
  • Potential unique insights: can collect globally-cycle-accurate out-of-band instruction traces from a networked datacenter simulation.
Summary

• Debugging Using Software Simulation (docs)
  • Target-Level
  • MIDAS-Level
  • FPGA-Level
• Debugging Using Integrated Logic Analyzers (docs)
• Advanced Debugging and Profiling Features
  • TracerV (docs)
  • Assertion Synthesis (docs)
• FireSim Debugging and Profiling Future Vision

Check out https://docs.fires.im/ for more usage details