ACCELERATING RTL SIMULATION WITH HARDWARE-SOFTWARE CO-DESIGN

Fares Elsabbaghi, Shabnam Sheikhha, Victor A. Ying, Quan M. Nguyen, Joel S. Emer, Daniel Sanchez

MICRO 2023
Overview

- RTL simulation is crucial for digital design
- Current systems are ill suited for simulation
  - CPUs: Can’t exploit fine-grained parallelism
  - Emulation: Takes too long to compile, limited size
- ASH is a **co-designed architecture and compiler to accelerate RTL simulation**
  - Fine-grained parallelism
  - Selective Execution
- Compared to state-of-the-art software simulator running on server CPU
  - ASH is **32x faster** while using \( \frac{1}{3} \) of the area
Limitation of Current Systems

- Consider simulating a large design
  - 128-core graph processing accelerator [Chronos, ASPLOS’20]

- Software Simulation
  - RTL code → CPU program (e.g., parallel C++ Program)
  - Quick to compile / Slow simulation speed

- Hardware Emulation
  - RTL code → Logic gates on emulator
  - Slow to compile / Fast simulation speed

<table>
<thead>
<tr>
<th>System</th>
<th>Compile Time</th>
<th>Simulation Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verilator on 32-core</td>
<td>2 Minutes</td>
<td>11 KHz</td>
</tr>
<tr>
<td>Emulation with 2xFPGA</td>
<td>13 Hours</td>
<td>1.4 MHz</td>
</tr>
<tr>
<td>ASH</td>
<td>2 Minutes</td>
<td>414 KHz</td>
</tr>
</tbody>
</table>
Outline

- Limitations of Software Simulation
- ASH Overview
  - Dataflow ASH
  - Selective ASH
- Evaluation
Dataflow Graph Representation

Pipelined Circuit

Dataflow Graph
Extracting Parallelism from Dataflow Graph

- Task is the basic unit of computation
- Dataflow Node $\rightarrow$ Task
  - Tasks can be scheduled on different threads
- Tasks can only run when inputs are ready
- Coarser tasks reduce synchronization
  - Work within a task is serialized
Effects of Coarsening on Parallelism and Performance

- **Design**
  - 128-core graph processing accelerator

- **Simulator**
  - Verilator

- **Platform**
  - 32-Core Zen2 CPU

- CPUs require too much coarsening
- Coarsening limits other optimizations!!
Selective Execution Needs Small Tasks

- Assume a system that only runs tasks if inputs change from the previous cycle
- Fine-grained tasks $\rightarrow$ Only does necessary work
- Coarse-grained tasks $\rightarrow$ Does unnecessary work
Effects of Coarsening on Activity Factor

- **Design**
  - 128-core graph processing accelerator

- **Simulator**
  - Verilator

- **Platform**
  - 32-Core Zen2 CPU

- CPUs can’t exploit low activity factor
Outline

- Limitations of Software Simulation
- **ASH Overview**
  - Dataflow ASH
  - Selective ASH
- Evaluation
ASH Hardware Overview

- Uses a multicore system as base architecture
  - Non-coherent L2 Cache
- Dataflow ASH (DASH) exploits fine-grained parallelism
  - Extends base system with hardware to orchestrate Prioritized Dataflow Execution
- Selective ASH (SASH) exploits selective execution
  - Extends DASH with hardware to perform speculative execution
Compiler maps each task to a tile
- TMU can schedule tasks on cores
- Tasks can create argument descriptors to communicate data
Prioritized Dataflow Execution

- Unordered dataflow execution
  - Increases memory footprint because it produces values much earlier than needed
  - Hurts parallelism because it doesn’t focus on critical path
  - Requires expensive structures to track tasks

- DASH executes the graph in priority order
  - Avoids the pitfalls of previous architectures
  - Focuses on critical path; minimizes footprint; simple hardware

- See paper for details of prioritization!
ASH Compiler

- Builds on Verilator
- Unroll Dataflow graph
  - Increase parallelism near cycle boundaries
- Partitioning
  - Statically maps tasks to tiles
  - Reduce data communication, maintains load-balance
  - More details in paper!

Diagram:

```
  N
  Cycle N
  inputs

  N+1
  Cycle N+1
  inputs

  Cycle N
  output

  Cycle N+1
  output

  Cycle N+1
  +

  Cycle N
  +

  Cycle N
  +
```

- Extract Dataflow Graph
- Unroll Dataflow Graph
- Unroll DFG
- Partition
- Partition DFG
- Mapped DFG
- Coarsen
- Task DFG
- Prioritize
  - + Timestamps
  - + DTTs
- Allocate Args (Reg/Mem)
  - + Task C++
- Generate Task Code

Verilator IR

Basic DFG

Unrolled DFG

Mapped DFG

Task DFG

+ Timestamps

+ DTTs

+ Task C++
Selective ASH (SASH) Overview

- SASH adds selective execution to DASH
  - Leverages low activity factors by running only tasks whose inputs have changed on each cycle

- Dynamic scheduling strategies
  - A. If a task is not going to execute, broadcast to children
    - This is inefficient for very low activity factors
  - B. Utilize speculation to lower communication
Speculation with SASH

- TMU doesn’t wait for all arguments
  - Uses values from previous cycle
- If misspeculated, abort!
- Prioritization reduces aborts!
- SASH utilizes prior speculation techniques [Swarm, MICRO’15] to implement selective execution
- More details in the paper!!

A
B
C

Tile 0

A
B
C

Tile 1
Outline

- Limitations of Software Simulation
- ASH Overview
  - Dataflow ASH
  - Selective ASH
- Evaluation
ASH is simulated as an ASIC running on 7nm technology
- 256 simple in-order cores across 64 tiles, 1MB L2 cache per tile
- Task-unit (speculation, prioritized dataflow) only adds a 5% area overhead
  - 512 unmerged argument descriptors per tile

Baseline System
- Verilator running on 32-Core AMD Zen2 CPU (Threadripper 3975WX)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Type</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vortex</td>
<td>GPU system</td>
<td>32-cores, 4warps/core, 2lanes/warp</td>
</tr>
<tr>
<td>Chronos with specialized PEs</td>
<td>SSSP Accelerator</td>
<td>128-PE system</td>
</tr>
<tr>
<td>Chronos with RISC-V cores</td>
<td>Multicore system</td>
<td>128 VexRiscv cores</td>
</tr>
<tr>
<td>Number Theoretic Transforms (NTT)</td>
<td>Functional Unit</td>
<td>8-stage pipeline, 256-wide 1024 multiplies</td>
</tr>
</tbody>
</table>
Overall Performance

Performance speedup over Zen2 baseline

System
- Zen2
- DASH
- SASH

Vortex  Chronos/PE  Chronos/RV  NTT  gmean
More Results in the Paper

- ASH system scales linearly with an increase of cores
- Prioritized dataflow execution reduces footprint by gmean of 17x compared to unordered
- SASH spends little time on tasks that get aborted
We introduce a co-designed architecture and compiler to accelerate RTL simulation.

ASH hardware architecture utilizes novel prioritized dataflow execution and selective execution techniques in the design.

Compared to a 32-core server CPU running Verilator:
- SASH achieves 32x faster simulation speed with 3x less area.