Tools Tutorials
Part A

Yunfei Gu
Washington University in St. Louis
Outline

RISC-V Z-scale Architecture

AHB-Lite protocol

Synopsys VCS
RISC-V Z-scale

- What is RISC-V Z-scale?

  Z-scale is a tiny 32-bit RISC-V core generator suited for microcontrollers and embedded systems

  Z-scale is designed to talk to AHB-Lite buses
  - plug-compatible with ARM Cortex-M series

  Z-scale generator also generates the interconnect between core and devices
  - Includes buses, slave muxes, and crossbars
Z-scale Pipelined

- 32-bit 3-stage single-issue in-order pipe
- Executes RV32IM ISA, has M/U privilege modes
- I-bus and D-bus are AHB-Lite and 32-bits wide
- Interrupts are supported
Z-scale

- ARM Cortex-M0 vs. Z-scale

<table>
<thead>
<tr>
<th>Category</th>
<th>ARM Cortex-M0</th>
<th>RISC-V Zscale</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA</td>
<td>32-bit ARM v6</td>
<td>32-bit RISC-V (RV32IM)</td>
</tr>
<tr>
<td>Architecture</td>
<td>Single-Issue In-Order 3-stage</td>
<td>Single-Issue In-Order 3-stage</td>
</tr>
<tr>
<td>Performance</td>
<td>0.87 DMIPS/MHz</td>
<td>1.35 DMIPS/MHz</td>
</tr>
<tr>
<td>Process</td>
<td>TSMC 40LP</td>
<td>TSMC 40GPLUS</td>
</tr>
<tr>
<td>Area w/o Caches</td>
<td>0.0070 mm²</td>
<td>0.0098 mm²</td>
</tr>
<tr>
<td>Area Efficiency</td>
<td>124 DMIPS/MHz/mm²</td>
<td>138 DMIPS/MHz/mm²</td>
</tr>
<tr>
<td>Frequency</td>
<td>≤50 MHz</td>
<td>~500 MHz</td>
</tr>
<tr>
<td>Voltage (RTV)</td>
<td>1.1 V</td>
<td>0.99 V</td>
</tr>
<tr>
<td>Dynamic Power</td>
<td>5.1 μW/MHz</td>
<td>1.8 μW/MHz</td>
</tr>
</tbody>
</table>
Z-scale

• Building a Z-scale System

- Working on “platform specification”
Z-scale

- Z-scale use cases

Microcontrollers
  - Implement your simple control loops
  - If code density matters

Embedded Systems
  - Build your system around Z-scale

Validation of Tiny 32-bit RISC-V Systems

Verilog versions of Z-scale is open-sourced under the BSD license

https://github.com/ucb-bar/zscale
https://github.com/ucb-bar/fpga-sparkan6
Outline

RISC-V Z-scale Architecture

AHB-Lite protocol

Synopsys VCS
AHB-Lite System

- components of AHB-Lite system
  - Master
  - Slaves
  - Address Decoder and
  - Multiplexer
AHB-Lite bus Master/Slave interface

- Global signals
  - HCLK
  - HRESETn
- Master out/slave in
  - HADDR (address)
  - HWDATA (write data)
  - Control
    - HWRITE
    - HSIZE
    - HBURST
    - HPROT
    - HTRANS
    - HMASTLOCK
- Slave out/master in
  - HRDATA (read data)
  - HREADY
  - HRESP
AHB-Lite signal definitions

- Global signals
  - HCLK: the bus clock source (rising-edge triggered)
  - HRESETn: the bus (and system) reset signal (active low)

- Master out/slave in
  - HADDR[31:0]: the 32-bit system address bus
  - HWDATA[31:0]: the system write data bus
  - Control
    - HWRITE: indicates transfer direction (Write=1, Read=0)
    - HSIZE[2:0]: indicates size of transfer (byte, halfword, or word)
    - HBURST[2:0]: burst transfer size/order (1, 4, 8, 16 beats or undefined)
    - HPROT[3:0]: provides protection information (e.g. I or D; user or handler)
    - HTRANS: indicates current transfer type (e.g. idle, busy, nonseq, seq)
    - HMASTLOCK: indicates a locked (atomic) transfer sequence

- Slave out/master in
  - HRDATA[31:0]: the slave read data bus
  - HREADY: indicates previous transfer is complete
  - HRESP: the transfer response (OKAY=0, ERROR=1)
Key to timing diagram conventions

• **Timing diagrams**
  - Clock
  - Stable values
  - Transitions
  - High-impedance

• **Signal conventions**
  - Lower case ‘n’ denote active low (e.g. RESETn)
  - Prefix ‘H’ denotes AHB
  - Prefix ‘P’ denotes APB
AHB-Lite signal

- Basic transfer - write
AHB-Lite signal

- Basic transfer - read
AHB-Lite signal

- AHB Pipelined transaction
AHB-Lite signal

- Adding wait states

**Diagram:**

- **HCLK:**
  - Address Phase A
  - Data Phase A
  - Address Phase B

- **HADDR:**
  - A
  - B

- **HWRITE:**
  - A
  - B

- **HWDATA:**
  - A

- **HREADY:**
  - A
  - A

- **HRDATA:**
  - A

- **HRESP:**
  - OKAY A
  - OKAY A

**Note:** Master will extend Address Phase B
Outline

RISC-V Z-scale Architecture
AHB-Lite protocol
Synopsys VCS
• What is Synopsys VCS?
VCS

- Compile your code

terminal command line:

```bash
% vcs -full64 -PP +lint=all,noVCDE +v2k -timescale=1ns/10ps <file>.v <file_tb>.v
```

_Successfully compiled_

CPU time: .238 seconds to compile +.257 seconds to elab + .341 seconds to link

_Do not need to recompile because nothing changed_

The design hasn't changed and need not be recompiled.
If you really want to, delete file simv.daidir/vcs.timestamp and run vcs again.
VCS

- **simulation your code**

A successfully compiling will print out on terminal “../simv up to date”. And it should generate an executable file named “simv” in the same folder where your codes are present. Then in the terminal run:

terminal command line:

```
% ./sim
```

![Simulation Report and Run Simulation messages](image)
View trace output with dve

After simulation report and "<file>.vcd" is generated, now type the following command in the terminal:

% dve

This is a viewer to plot and verify your results.
VCS

• View trace output with dve

Go to “File->Open Database” and select the “.vcd” file from the project folder. Then you will find the name of your test bench model in the Hierarchy box (Counter_tb here). Expand it so that you can find DUT in the options. If you click on DUT, select the signals listed (all or partial) and right click, you will find an option “Add to Waves”. Also it is easy to debug as shown below:
Acknowledgement


CS250 VLSI Systems Design (2009-2011) - University of California at Berkeley

The architecture for digital world: ARM