MAERI Architecture and Implementation Details

Synergy Lab, Georgia Tech
Hyoukjun Kwon

http://synergy.ece.gatech.edu
Acknowledgement

• Mr. Charlie Hauck
• Dr. Rishiyur Nikhil

For providing BSV license for hands-on exercises

• Ananda Samajdar
• Eric Qin
• Yehowshua Immanuel

For discussions and ASIC/FPGA synthesis
Outline

Tool Flow of MAERI

• MAERI Implementation Details
• Using MAERI source code base
• Demo and exercises
Tool Flow of MAERI

**Resource Description**
- NumPE 64
- DistBW 4
- GatrBW 4

**Building Block Library (BSV RTL)**
- Adder
- Switch
- Mult.
- Switch
- Simple
- Switch
- Cntl
- ...

**MAERI Front-end**
- BSV Compiler

**RTL Generation**
- Verilog Files

**MAERI Framework**
- MAERI Input
- MAERI Outputs

**MAERI Outputs**
- Cycle-accurate Simulation
  - # Cycles
  - # Weight Distribution
  - # Input Distribution
  - # Local Communication
Bluespec System Verilog (BSV)

• A high-level hardware description language
  • Generates fully synthesizable Verilog

• Inspired by Haskell and System Verilog
  • Strong type-checking system and polymorphism
  • System Verilog-like syntax
  • Intuitive module interfaces

• Based on “guarded atomic action” blocks
  • Provides coarse-grained description of parallel actions
BlueSpec System Verilog (BSV)

• A high-level hardware description language
  • Generates fully synthesizable Verilog

• Inspired by Haskell and System Verilog
  • Strong type-checking system and polymorphism
  • System Verilog-like syntax
  • Intuitive module interfaces

• Based on “guarded atomic action” blocks
  • Provides coarse-grained description of parallel actions

For details, please refer to “BSV by Example”
Outline

• Tool Flow of MAERI
  MAERI Implementation Details
  • Using MAERI source code base
  • Demo and exercises
MAERI Implementation – Distribution Network

- # Multiplier Switches = 16
- Distribution Bandwidth = 4X
MAERI Implementation – Multiplier Network

- # Multiplier Switches = 16
MAERI Implementation – Reduction Network

• # Multiplier Switches = 16
• Reduction Bandwidth = 4X
MAERI Implementation – Reduction Network

• # Multiplier Switches = 16
• Reduction Bandwidth = 4X
MAERI Implementation – Reduction Network

- # Multiplier Switches = 16
- Reduction Bandwidth = 4X
Outline

• Tool Flow of MAERI
• MAERI Implementation Details
  Using MAERI source code base
• Demo and exercises
Source Code Directory Structure

maeri_code_hpca2019_tutorial

scripts

dir

src

- types
- lib
- distribution_network
- reduction_network
- multiplier_network
- ALUs

- maeri_accelerator

Custom BSV type definitions
Custom BSV libraries
Distribution tree
Augmented reduction tree
Multiplier switch and its array
Fixed point adder/multiplier
MAERI top module

MAERI core implementation
How to use MAERI front-end

• **Changing design parameters**
  • Modify AcceleratorConfig.bsv at the top directory
    • Distribution bandwidth
    • Reduction bandwidth
    • Number of multiplier switches

• **Cycle-accurate simulation and Verilog code generation**
  • `./Maeri -c` : Compile a simulation
  • `./Maeri -r` : Run compiled simulation
  • `./Maeri -w` : Launch GTKwave for waveform analysis
  • `./Maeri -v` : Generate Verilog code
  • `./Maeri -clean` : Clean up intermediate files
How to use MAERI front-end

• Simulation results example
  • Commands: “./Maeri –r” after “./Maeri –c”
How to use MAERI front-end

• Waveform Analysis
  • Commands: "./Maeri –r” and then "./Maeri –w”
How to use MAERI front-end

• Verilog code generation
  • Commands: “./Maeri –v”

* Verilog files are generated in “(Top_Directory)/Verilog”

Generated Verilog code is synthesizable!
MAERI Synthesis and PnR

• Synthesis/PnR Environment
  • Technology: 28nm
  • Clock frequency: 1GHz
  • Design: 64 multiplier switches and 31 adder switches
  • Distribution Bandwidth: 32/16/8/4 data per cycle
  • Gather Bandwidth: 32/16/8/4 data per cycle
  • RTL Code: Verilog generated using MAERI code base
  • CAD Tool Chain: Synopsys Design compiler, Cadence Innovus, Primepower
Post-layout Area and Power

Bandwidth: 8X

<table>
<thead>
<tr>
<th>Num PEs</th>
<th>Area (um²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>128</td>
<td>0</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
</tr>
</tbody>
</table>

Bandwidth: 8X

- Wire
- RN
- MN
- DN
Post-layout Area and Power

NumPEs: 64

Area (um²)

Bandwidth

Wire  RN  MN  DN

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Area (um²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4X</td>
<td>20000</td>
</tr>
<tr>
<td>8X</td>
<td>30000</td>
</tr>
<tr>
<td>16X</td>
<td>40000</td>
</tr>
<tr>
<td>32X</td>
<td>50000</td>
</tr>
</tbody>
</table>
FPGA Resource Usage

* Based on Virtex 7 board, synthesis frequency: 50MHz
FPGA Resource Usage

* Based on Virtex 7 board, synthesis frequency: 50MHz
Outline

• Tool Flow of MAERI
• MAERI Implementation Details
• Using MAERI source code base

Demo and exercises
Demo

• Launching cycle-accurate simulations
• Modifying user configuration
• Compiling simulations
• Launching wave form analysis
• Generating Verilog files
Outline

• Tool Flow of MAERI
• Source code Structure
• Using MAERI source code base
• Demo

Hands-on Exercises
Testbench Structure

Generated Simulation Model

Weights / Inputs

Interconnect Control

Outputs

Machine codes

Switch Configurations

RN_Config.vmh

Tile configurations

Layer_Info.vmh

VN0

VN1
Testbench Dataflow (MAESTRO description)

let vnSize = sizeof(R) x sizeof(S)
let numVNs = floor(NumMultSwitches / vnSize)

• Temporal_Map (1, 1) C
• Spatial_Map (1, 1) K
• Temporal_Map (sizeof(R), 1) Y
• Temporal_Map (sizeof(S), 1) X
• Cluster (vnSize, L)
• Temporal_Map (SizeOf(S), SizeOf(S)) S
• Spatial_Map (1,1) R
• Cluster(vnSize, P)
• Spatial_Map (1,1) S
• **Exercise#1**: Compile a simulation with default, early, and late layers with 32 PEs ("./Maeri –c, "). Run simulation and compare results.

• **Exercise#2**: Compile a simulation with 32 and 64 PEs using default setting ("./Maeri –c"). Run simulation and compare results.

• **Exercise#3**: Compile a simulation with 4X/8X/and 16X distribution bandwidth (fix reduction bandwidth as 8X). Run simulation and compare results.

• **Exercise#4**: Compile a simulation with 4X/and 8X reduction bandwidth (fix distribution bandwidth as 8X). Run simulation and compare results.