OpenPiton+Ariane in Action

Princeton University and ETH Zürich

http://openpiton.org
http://pulp-platform.org
Configuration
# Configurability Options

<table>
<thead>
<tr>
<th>Component</th>
<th>Configurability Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores (per chip)</td>
<td>Up to 65,536</td>
</tr>
<tr>
<td>Cores (per system)</td>
<td>Up to 500 million</td>
</tr>
<tr>
<td>Core Type</td>
<td>OpenSPARC T1, Ariane 64 bit RISC-V</td>
</tr>
<tr>
<td>Threads per Core</td>
<td>1/2/4</td>
</tr>
<tr>
<td>Floating-Point Unit</td>
<td>FP64, FP32, FP64, FP32, FP16, FP8, BFLOAT16</td>
</tr>
<tr>
<td>Stream-Processing Unit</td>
<td>Present/Absent</td>
</tr>
<tr>
<td>TLBs</td>
<td>8/16/32/64 entries</td>
</tr>
<tr>
<td>L1 I-Cache</td>
<td>8*/16/32kB</td>
</tr>
<tr>
<td>L1 D-Cache</td>
<td>4*/8/16KB</td>
</tr>
<tr>
<td>L1.5 Cache</td>
<td>Number of Sets, Ways (8kB, 4-way)</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>Number of Sets, Ways (64kB, 4-way)</td>
</tr>
<tr>
<td>Intra-chip Topologies</td>
<td>2D Mesh, Crossbar</td>
</tr>
<tr>
<td>Inter-chip Topologies</td>
<td>2D Mesh, 3D Mesh, Crossbar, Butterfly Network</td>
</tr>
<tr>
<td>Bootloading</td>
<td>SDHC Card, UART, RISC-V JTAG Debug</td>
</tr>
</tbody>
</table>

*L1 cache goes to 2-ways at smallest size*
Setting configuration options

• Change a flag in `sims`
• Add or edit flags in `piton/tools/src/sims/manycore.config`

• Examples:
  – `--config_l15_size=16384`
  – `--network_config=xbar_config`
Configuration under the hood

- **sims** passes cache configurations as environment variables to PyHP

- Some others are passed as verilog defines

  ```
  -config_rtl=
  PITON_NO_FPU
  ```

  ```bash
  $DV_ROOT/tools/src/sims/sims,2.0
  ```
PyHP and parameterisation

• PyHP is a python preprocessor used in OpenPiton for generating complex RTL
• Embeds python inside the Verilog (Inspired by PHP)
  – Use print to output verilog code
  – `<% print "wire a;" %>`
• Better than Verilog’s `generate`?
  – Provides visibility of intermediate code
• Not perfect but very practical
Configuration under the hood

- PYHP reads the environment variables, prints sizes to .pyv files

```python
# cache configurations
CONFIG_L15_SIZE = int(os.environ.get('CONFIG_L15_SIZE', '8192'))
CONFIG_L15_ASSOCIATIVITY = int(os.environ.get('CONFIG_L15_ASSOCIATIVITY', '4'))
CONFIG_L1D_SIZE = int(os.environ.get('CONFIG_L1D_SIZE', '8192'))
CONFIG_L1D_ASSOCIATIVITY = int(os.environ.get('CONFIG_L1D_ASSOCIATIVITY', '4'))
CONFIG_L1I_SIZE = int(os.environ.get('CONFIG_L1I_SIZE', '16384'))
CONFIG_L1I_ASSOCIATIVITY = int(os.environ.get('CONFIG_L1I_ASSOCIATIVITY', '4'))
CONFIG_L2_SIZE = int(os.environ.get('CONFIG_L2_SIZE', '65536'))
CONFIG_L2_ASSOCIATIVITY = int(os.environ.get('CONFIG_L2_ASSOCIATIVITY', '4'))
```

`piton/tools/bin/pyhplib.py`
Configuration under the hood

- .pyv files use these parameters
  - Some code will print the parameters as Verilog defines

```python
linesize = 16
# 512
nentries = CONFIG_L1D_SIZE / linesize
# 4
way = CONFIG_L1D_ASSOCIATIVITY
# 2
waywidth = math.log(CONFIG_L1D_ASSOCIATIVITY, 2)
# 512
print(\`\`define L1D_ENTRY_COUNT %d\`\` % nentries)
# 9 - 1 - 2 = 6
print(\`\`define L1D_SET_IDX_HI %d\`\` % (math.log(nentries, 2) - 1 - waywidth))
# 4
print(\`\`define L1D_WAY_COUNT %d\`\` % way)
# 2
print(\`\`define L1D_WAY_WIDTH %d\`\` % waywidth)
```

piton/design/include/lsu.h.pyv
Configuration under the hood

- Adjusts instantiations according to # of icache ways

```python
140    template = '':
141    bw_r_ict_array ictag_ary___WAYID(
142        .we  (we[___WAYID]),
143        .clk  (clk),
144        .way  (`IC_WAY_IDX_WIDTH'd___WAYID),
145        .rd_data(rdtag_y[`IC_TLB_TAG_WAY___WAYID_MASK]),
146        .wr_data(wrtag_y),
147        .addr  (index_y),
148        .dec_wrway_y (dec_wrway_y));
149
150    ...
151    for i in range(CONFIG_L1I_ASSOCIATIVITY):
152        arraytext = template.replace("___WAYID", str(i));
153        print(arraytext);
154    %>
```

piton/design/chip/tile/sparc/srams/rtl/bw_r_ict.v.pyv
Configuration Examples

• Define core count:
  - `sims -sys=manycore -x_tiles=16 -y_tiles=16 -ariane -vlt_build`
  • If specified in build command, should also be in run

• Define network topology
  - `sims -sys=manycore -network_config=xbar_config -x_tiles=4 -ariane -vlt_build`
Configuration Examples

• Cache Defaults:
  – L1 I-cache 16KB, 4-way
  – L1 D-cache 8KB, 4-way
  – L1.5: 8KB, 4-way
  – L2: 64KB, 4-way

• Decrease L2 cache by factor of 4:
  – sims -sys=manycore -vlt_build
    -config_l2_size=16384 -ariane
Configuration Examples

• Run a coherent "hello world" on 2x2 tiles:

```bash
-sims -sys=manycore -vlt_build
    -x_tiles=2 -y_tiles=2 -ariane

-sims -sys=manycore -vlt_run
    -x_tiles=2 -y_tiles=2 -ariane
    hello_world_many.c
    -rtl_timeout=1000000

-cat fake_uart.log
```