rASIP: Reconfigurable Application Specific Processors

Challenges and Opportunities

Guillermo Payá-Vayá, Stephan Nolting, Florian Giesemann, and Holger Blume
Motivation – Why reconfigurable ASP?

- **Application Specific Processor (ASP)**
  - Baseline processor optimized for a specific application or application set
  - **Specialization** is performed during **design-time**
  - Resulting **processing efficiency** depends on the applications (benchmarks) used during **design-time**
    - Increase reusability by implementing “general purpose” application hardware accelerator (e.g., counting leading zeros)

- **Reconfigurable Application Specific Processor (rASP)**
  - Baseline processor optimized for a specific application or application set
  - **Specialization** is performed during **design-time** and **run-time**
  - Resulting **processing efficiency** can be improved during **run-time**
    - Reusability compatible with more “specific” application hardware accelerator
A Reconfigurable MIPS (rMIPS) Processor (I)

- 32-bit MIPS processor, 5 pipeline stages, C-compiler available (LLVM), I-/D-Cache
  - RFU (reconfigurable functional unit) for small tasks (custom instructions)
  - RCU (reconfigurable coprocessor unit) for more complex tasks (e.g. SINE)

Simplify scheme of a MIPS architecture
A Reconfigurable MIPS (rMIPS) Processor (II)

- Proof of concept on a Virtex-6 Xilinx FPGA
- Mathematical LibARITH library in software and hardware (RCU)
  - Sine and cosine (SINCOS)
  - Square root (SQRT)
  - Exponential function (EXP)
  - Natural logarithm (LN)
  - 3x3x6 Matrix multiplication (MM3x3x6)
  - 6x6x6 Matrix multiplication (MM6x6x6)

<table>
<thead>
<tr>
<th>Function</th>
<th>#SW (Cycles)</th>
<th>#HW (Cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINCOS</td>
<td>232</td>
<td>90</td>
</tr>
<tr>
<td>SQRT</td>
<td>750</td>
<td>92</td>
</tr>
<tr>
<td>EXP</td>
<td>300</td>
<td>128</td>
</tr>
<tr>
<td>LN</td>
<td>321</td>
<td>82</td>
</tr>
<tr>
<td>MM3x3x6</td>
<td>1225</td>
<td>456</td>
</tr>
<tr>
<td>MM6x6x6</td>
<td>4288</td>
<td>906</td>
</tr>
</tbody>
</table>
rMIPS: HW–SW LibARITH Library

**Challenge:** Time required during reconfiguration process

- **HW** - SW LibARITH

---

**Reconfig. Time**

- **240 SLICES**
- **~5630 cycles**

---

**number of calls (LibARITH)**

- **SW SQRT**
  - 750 cycles/call

- **HW SQRT**
  - 92 cycles/call
rMIPS: HW-SW LibARITH Library

**Challenge:** Time required during reconfiguration process

- **Reconfig. Time**
  - 240 SLICES
  - ~5630 cycles

- **HW_SQRT**
  - 92 cycles/call

- **SW_SQRT**
  - 750 cycles/call

- **FIX_HW_SQRT**
  - 92 cycles/call
**rMIPS: HW–SW LibARITH Library**

**Challenge:** Time required during reconfiguration process

- **SW_SQRT** (softward square root): 750 cycles/call
- **HW_SQRT** (hardward square root): 92 cycles/call
- **SW_LN** (softward log): 321 cycles/call
- **HW_LN** (hardward log): 8 cycles/call

Reconfig. Time: 240 SLICES ~5630 cycles

Payá-Vayá et al.
Tensilica Day, 9th February, 2016
rMIPS: HW–SW LibARITH Library

**Challenge:** Time required during reconfiguration process

**Opportunity:** Unlimited number of “specific” app. HW. accelerators
Institute of Microelectronic Systems

rMIPs – Reconfiguration (RCU-Slots) Scheduling

Challenge: Optimal reconfiguration scheduling could be not deterministic on design-time (e.g., profiling)

LibARITH provides pure SW-based and HW-accessing subroutines to compute a set of complex arithmetic operations (SINE, SQRT, ...)

Opportunity: Programmer does not need to think about explicitly using HW accelerators

Institute of Microelectronic Systems

rMIPS – Autonomous Reconfiguration (I)

- Dynamic Partial Reconfiguration (DPR) System
- Instruction Fetch (IF) Observer

### DPR System

- **OCP Bus**
- **IO Bus**
- **MIPS Processor**
- **IF Observer**
- **IF Controller**
- **DPR Controller**
- **ICAP Arbiter**
- **ICAP**
- **RCU**
- **Gateway**

### Access Counter (N) SW Function Address

<table>
<thead>
<tr>
<th>Value</th>
<th>Function</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>622</td>
<td>sw_sin</td>
<td>0x00002030</td>
</tr>
<tr>
<td>451</td>
<td>sw_sqrt</td>
<td>0x00004404</td>
</tr>
<tr>
<td>46</td>
<td>sw_exp</td>
<td>0x00006080</td>
</tr>
<tr>
<td>0</td>
<td>sw_in</td>
<td>0x00010202</td>
</tr>
</tbody>
</table>

### IF Observer

- Managed by hardware
- Programmable by user / application
rMIPS – Autonomous Reconfiguration (II)

- Dynamic Partial Reconfiguration (DPR) System
  - Instruction Fetch (IF) Observer
  - Instruction Fetch (IF) Controller
rMIPS – Autonomous Reconfiguration (III)

- Dynamic Partial Reconfiguration (DPR) System
  - Instruction Fetch (IF) Observer
  - Instruction Fetch (IF) Controller
  - DPR Controller
  - Reconfiguration Strategy

**Diagram**

- **OCP Bus**
- **IO Bus**
- **MIPS Processor**
- **IF Observer**
  - **Statistics**
- **IF Controller**
  - **ACK**
  - **EN**
  - **New destination**
- **DPR Controller**
- **Gateway**
- **ICAP Arbiter**
- **ICAP**
- **RCU**

**Diagram Details**

- **Access Counter (N)**
- **SW Function Address**
- **HW Function Address**
- **Bitstream Address**
- **HW avail?**
- **Corresp. RCU**

<table>
<thead>
<tr>
<th>Access Counter (N)</th>
<th>SW Function Address</th>
<th>HW Function Address</th>
<th>Bitstream Address</th>
<th>HW avail?</th>
<th>Corresp. RCU</th>
</tr>
</thead>
<tbody>
<tr>
<td>622</td>
<td>sw_sin: 0x00002030</td>
<td>hw_sin: 0x00022008</td>
<td>sin_rcu @ 0x00100000</td>
<td>true</td>
<td>Slot 2</td>
</tr>
<tr>
<td>451</td>
<td>sw_sqrt: 0x00004404</td>
<td>hw_sqrt: 0x00023400</td>
<td>sqrt_rcu @ 0x00110000</td>
<td>true</td>
<td>Slot 1</td>
</tr>
<tr>
<td>46</td>
<td>sw_exp: 0x00006080</td>
<td>hw_exp: 0x00024E00</td>
<td>exp_rcu @ 0x00120000</td>
<td>false</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>sw_ln: 0x00010202</td>
<td>hw_ln: 0x00024FF4</td>
<td>ln_rcu @ 0x00130000</td>
<td>false</td>
<td>-</td>
</tr>
</tbody>
</table>

**Legend**

- Managed by hardware
- Programmable by user / application
Hardware replacement policies derived from computer cache systems
- First-In-First-Out (FIFO)
- Random (RND)
- Least-recently used (LRU)
- Least-frequently used (LFU)

Implement an **on-demand concept**: An RCU is always loaded when its corresponding function is accessed.

<table>
<thead>
<tr>
<th>Function</th>
<th>#SW (Cycles)</th>
<th>#HW (Cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINCOS</td>
<td>232</td>
<td>90</td>
</tr>
<tr>
<td>SQRT</td>
<td>750</td>
<td>92</td>
</tr>
<tr>
<td>EXP</td>
<td>300</td>
<td>128</td>
</tr>
<tr>
<td>LN</td>
<td>321</td>
<td>82</td>
</tr>
<tr>
<td>MM3x3x6</td>
<td>1225</td>
<td>456</td>
</tr>
<tr>
<td>MM6x6x6</td>
<td>4288</td>
<td>906</td>
</tr>
</tbody>
</table>
rMIPS– Reconfigurable Strategy (II)

- **Weighted Replacement Policy (WRP) with Hysteresis**

  The called function corresponding to $RCU_y$ will substitute $RCU_x$ (already reconfigured and available) if $(N_x * W_{x,y}) + H < N_y$

<table>
<thead>
<tr>
<th>Function</th>
<th>#SW (Cycles)</th>
<th>#HW (Cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINCOS</td>
<td>232</td>
<td>90</td>
</tr>
<tr>
<td>SQRT</td>
<td>750</td>
<td>92</td>
</tr>
<tr>
<td>EXP</td>
<td>300</td>
<td>128</td>
</tr>
<tr>
<td>LN</td>
<td>321</td>
<td>82</td>
</tr>
<tr>
<td>MM3x3x6</td>
<td>1225</td>
<td>456</td>
</tr>
<tr>
<td>MM6x6x6</td>
<td>4288</td>
<td>906</td>
</tr>
</tbody>
</table>
An iterative synthetic benchmark consists of a large invocation sequence of LibARITH functions, featuring a recurring pattern separated by small random-distributed sections (e.g., simulate the behavior of data dependent applications)
- All six functions are used

A backtracking algorithm from the parallel robotic field. The position of a Stewart platform is estimated using the information provided by the sensor located in the motors.
- SINCOS, SQRT, and MM3x3x6
rMIPS – Evaluation with a iterative synthetic benchmark

- Executed Cycles
- Hardware Resources (Slices)
- Hardware Efficiency

19940 reconfig. (15.6 % Hits)
2314 reconfig. (39.2 % Hits)

Pure SW
FIFO
LRU
WRP
Fix_RCU (MM6x6x6)

MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

rMIPS+1(R)CU

Payá-Vayá et al.
Tensilica Day, 9th February, 2016
rMIPS – Evaluation with a iterative synthetic benchmark

- Executed Cycles
- Hardware Resources (Slices)
- Hardware Efficiency

1/(HW Resources * Executed Cycles)

MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

rMIPS+1(R)CU

# rMIPS – Evaluation with a iterative synthetic benchmark

## Executed Cycles, Hardware Resources (Slices), Hardware Efficiency

<table>
<thead>
<tr>
<th></th>
<th>rMIPS+1(R)CU</th>
<th>rMIPS+2(R)CU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure SW</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>FIFO</td>
<td>0.76</td>
<td>0.51</td>
</tr>
<tr>
<td>LRU</td>
<td>0.76</td>
<td>0.49</td>
</tr>
<tr>
<td>WRP</td>
<td>0.54</td>
<td>0.47</td>
</tr>
<tr>
<td>Fix_RCU (MM6x6x6)</td>
<td>0.64</td>
<td>0.50</td>
</tr>
</tbody>
</table>

**MIPS @ 100 MHz**
2848 Slices (8%)  
XC6VLX240T-1
**rMIPS – Evaluation with a iterative synthetic benchmark**

- **Executed Cycles**
- **Hardware Resources (Slices)**
- **Hardware Efficiency**

<table>
<thead>
<tr>
<th></th>
<th>rMIPS+1(R)CU</th>
<th>rMIPS+2(R)CU</th>
<th>rMIPS+6CU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MIPS @ 100 MHz</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2848 Slices (8%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XC6VLX240T-1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Payá-Vayá et al.**

Tensilica Day, 9th February, 2016
rMIPS – Evaluation with a Parallel Robotics application

MIPS @ 100 MHz
2848 Slices (8%)
XC6VLX240T-1

**rMIPS – Evaluation with a Parallel Robotics application**

<table>
<thead>
<tr>
<th></th>
<th>Executed Cycles</th>
<th>Hardware Resources (Slices)</th>
<th>Hardware Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure SW</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>FIFO</td>
<td>0.43</td>
<td>1.13</td>
<td>2.05</td>
</tr>
<tr>
<td>LRU</td>
<td>0.43</td>
<td>1.13</td>
<td>2.05</td>
</tr>
<tr>
<td>WRP</td>
<td>0.43</td>
<td>1.16</td>
<td>2.01</td>
</tr>
<tr>
<td>Fix_RCU (SQRD)</td>
<td>0.53</td>
<td>1.03</td>
<td>1.81</td>
</tr>
<tr>
<td>Fix_RCU (SINCOS, SQRT)</td>
<td>0.43</td>
<td>1.21</td>
<td>1.92</td>
</tr>
<tr>
<td>Fix_RCU (all)</td>
<td>0.33</td>
<td>1.09</td>
<td>2.20</td>
</tr>
</tbody>
</table>

**MIPS @ 100 MHz**
2848 Slices (8%)
XC6VLX240T-1

Challenges and Opportunities

- **Opportunities:**
  - Unlimited number of RCUs
  - Programmer does not need to think about explicitly using HW accelerators
  - Evaluation with both benchmarks shows that:
    - \( \text{rMIPS}+1\text{RCU} \) is faster than a MIPS+1CU \( \rightarrow 23\% \)
    - \( \text{rMIPS}+1\text{RCU} \) is more hardware efficient than a MIPS+1CU \( \rightarrow 11\% \)
    - \( \text{rMIPS}+2\text{RCU} \) is faster but less hardware efficient than a MIPS+2CU \( \rightarrow 8\% \)

- **Challenges:**
  - Predictive reconfiguration strategies are needed
  - Improve the LibARITH library
  - Reduce the time required to perform a reconfiguration
  - Improve the place-and-route of the RCUs on the reconfiguration partitions (since now less than 50% of hardware utilization)
Thank you for your attention!