The Warp Processor

Dynamic SW/HW Partitioning

David Mirabito

A presentation based on the published works of

Dr. Frank Vahid - Principal Investigator
Dr. Sheldon Tan - Co-Principal Investigator
Dr. Walid Najjar - Co-Principal Investigator
et al.

(http://www.cs.ucr.edu/~vahid/warp/)
and supporting papers

So far For Us...

- Ben showed us how instruction collapsing works and its potential benefits.
  - Instead of implementing as $n$ LUT accesses, warp reconfigures the fabric.
- Kynan showed how Chimera uses pre-compiled code/bitstreams to increase performance.
  - Warp dynamically generates the bitstream and modifies code on the fly.

Warp combines the best of both worlds!

Warp Overview:

- Warp is the name for the family of processors, not an individual implementation
  - Website has an 100Mhz Arm7 as the main processor. And another for the DPM.
  - Quoted average speedup of 7.4 and energy reduction of 38-94%
- Can also apply to other Arms, MIPS, etc. x86 anyone?
On-Chip Profiler

- With current SOC, snooping address lines no longer an option.
- Requirements: non-intrusion, low power, and small area.
- Monitors sb's (short backwards branch) to show loop iteration.
- Cache indexed by sbb address. On hit 16bit counter is incremented. On saturation, all counters >> 1 to maintain correct ratios.
- 90-95% accuracy, power up by 2.4% and area up by 10.5% (or an extra 0.167 mm²)

On-Chip Profiler

- Much of the power consumption is from the cache lookup / write.
- Common cache power techniques -> 1.5%
- Can decrease this by coalescing: keep count when the same sbb is repeatedly seen and only updating the cache upon sight of a new sbb.
- Now 0.53% power overhead, 11% area.
- For a 5% drop in accuracy, we can allow every nth ssb to be processed. This sampling drops power consumption to 0.02% above normal for n = 50.

DPM

- Partitioning – Taken from profiler.
- Decompilation
- DMA Configuration
- RT (Register Transfer) Synthesis
- Now have a HW description of code more appropriate for further synthesis
- JIT FPGA Compilation
  - Logic Synthesis
  - Technology Mapping
  - Placement
  - Routing

Decompilation

- Previous binary decompilers were poor.
- Extra optimizations can be made if we have high level constructs available
  - Smart buffers
  - Loop unrolling
  - Redundant operations due to ISA overhead.
- So we decompile:
  - Loops analysed for linear memory strides to determine array access. Overlaps for iterations placed in smart buffers rather than main memory.
  - Loop bounds found for unrolling
  - Size of data types tracked, min bits used for ALU operations.
DMA Configuration

- Uses the memory access patterns of the decompiled code to configure the DMA controller.
- Initially all data will be fetched before the loop begins.
  - Then one block can be fetched/written per cycle. The same rate as the collapsed loop needs it.
- Finally, after the loop one more DMA is scheduled to write back the final data.

RT Synthesis

- The high level data is then fed into a standard netlist generator.
- Netlists generated this way were found to be 53% faster than other forms of binary synthesis.
- When compared to synthesis from source, decompilation resulted in identical performance with a 10% increase in surface area.
- Area increase to decompiler's inability to remove some temporary registers found in long expressions out of the datapath.

JIT FPGA Compilation

- Logic Synthesis
  - Logically analyses the netlist at a gate level to minimise the amount of gates required.
  - Uses the Riverside On Chip Minimiser, an algorithm optimised for fast execution in a low memory environment.
- Technology Mapping
  - Mapping at a gate level the netlist onto the configurable material.
  - Uses a standard algorithm.

JIT FPGA Compilation

- Routing & Placing:
  - Most expensive part of the synthesis process
  - Requires 14.8sec and 12M RAM -> Not good for an embedded system.
  - Commercial FPGAs are overly complex for implementing only a collection of instructions.
  - Designed a simple fabric: easy to route for.
  - Another part of Riverside On Chip Partitioning Tools (ROCPART) suite.
**JIT FPGA Compilation**

- Final design has 67x67 CLBs, channel width of 30.
  - Simpler, so easier to place and route.
  - Suitable for on-chip!

---

**Code Update**

- The DRM also updates the code memory.
  - Replaces one instruction with a branch to a specialises HW routine.
  - This starts the WCLA and places the processor in a sleep state.
  - Upon the WCLA’s completion, an interrupt wakes the processor, which then jumps back to the end of the loop in the original code.

---

**WCLA**

- DADG: Data Address generator.
  - All mem access to/from WCLA.
  - Generate addr for 3 arrays
  - Delivers data to Reg{0,1,2}
- LCH: Loop Control Hardware.
  - Enables zero overhead looping
  - Needs preset loop upper bound, but allows early breaks depending on some configurable result.
- Regs. Input registers to the configurable logic fabric. Wired to the MAC, but also accessible directly from the fabric.

---

**MAC: Multiply and Accumulator.**

- Most inner loops require some addition/multiplication. To save on logic area and routability, a dedicated MAC is included.

---

**SCLF: Simple Configurable Logic Fabric.**

- As described earlier.
- Designed for simple bitstream generation by on chip devices with limited time and memory resources.
Results – MIPS 60MHz

<table>
<thead>
<tr>
<th>Tool</th>
<th>Code Size</th>
<th>Binary size</th>
<th>Data size</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decompression</td>
<td>7,203</td>
<td>125</td>
<td>452</td>
<td>0.05</td>
</tr>
<tr>
<td>DMA Configuration</td>
<td>4,695</td>
<td>88</td>
<td>360</td>
<td>1.04</td>
</tr>
</tbody>
</table>

Details of the tools running on the co-processor.

Weight of the critical regions in tests.

Results using the WARP architecture. The whole process is automated, except binary modification with is currently done by hand.

Results – ARM 100MHz

The speedup of the replaced kernel. The notable high ones are due to the reimplementation being only bit shifting via wires, or mem access being replaced by a single block DMA transfer.

Overall speedup across the entire execution of the benchmark. Of note is the minimum speedup of 2.2 and an average of 7.4. This is accompanied with a 38-94% saving in energy consumption.

FPGAs All The Way

- The developers of Warp also investigated putting the entire system on a FPGA.
- Soft-core MicroBlazes on a Spartan3
  - One is the system processor
  - Other runs the DPM (currently)
- Ideally WCLA would directly utilise the underlying reconfigurable fabric.
  - Currently simulating the WCLA
  - And looking into implementing it on top.
  - Ideally use the spartan's own configurable fabric.

Implementation

- What they did:
  - Modified MicroBlaze to include the profiler.
  - Simulated execution on Xilinx apps to obtain program trace.
  - Trace used to simulate behaviour of profiler, found single critical region.
  - Used ROCPART to generate HW circuits.
  - Measured these in a VHDL model of WCLA, combined with traces to obtain final performance / energy measurements.
Results:

What they found:
- Warp architecture more than compensated for traditional speed/power issues normally found in FPGA solutions.
- Still maintains flexibility.
- Gives custom-built systems a run for their money.

Resources

  http://www.cs.ucr.edu/~vahid/pubs/cases03_profile.pdf
- Dynamic Hardware/Software Partitioning: A First Approach, Greg Stitt, Roman Lysecky and Frank Vahid. 
- A Configurable Logic Architecture for Dynamic Hardware/Software Partitioning, Roman Lysecky and Frank Vahid 
- A Study of the Speedups and Competitiveness of FPGA Soft Processor Cores using Dynamic Hardware/Software Partitioning, Roman Lysecky and Frank Vahid 
- Techniques for Synthesizing Binaries to an Advanced Register/Memory Structure, Greg Stitt, Zhi Guo, Frank Vahid and Walid Najjar. 