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.

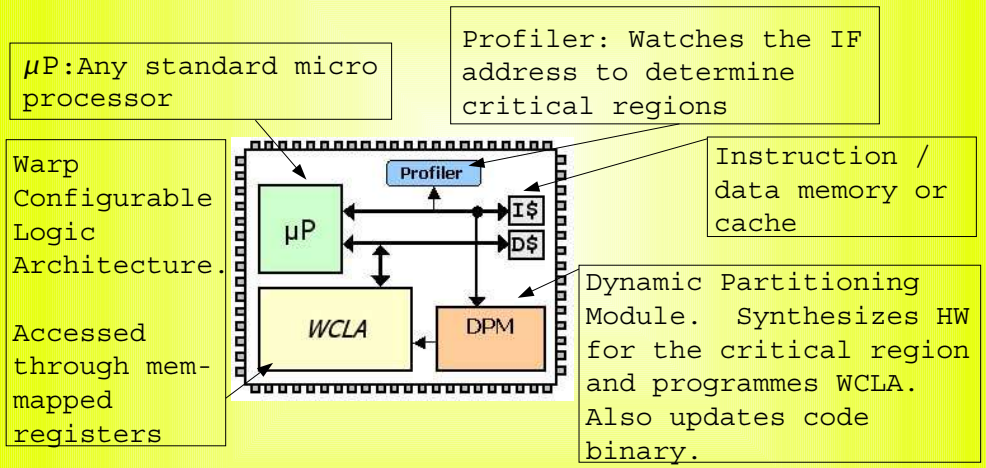
(http://www.cs.ucr.edu/~vahid/warp/)
and supporting papers

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:

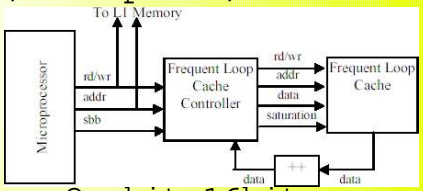


Warp Overview:

- 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 mm^2)

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 n^{th} 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:

Difficulties:

most expensive part of the synthesis process

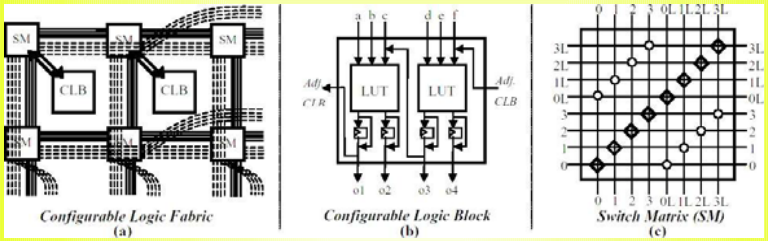
- Requires 14.8sec and 12M RAM -> Not good for an embedded system.

Solutions:

- Developed Riverside On Chip Router (ROCR)
  - 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

Results:



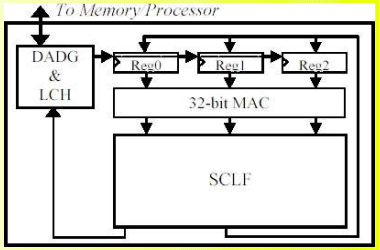
- Final design has 67x67 CLBs, channel width of 30.
  - Simpler, so easier to place and route.
- Suitable for on-chip!

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.

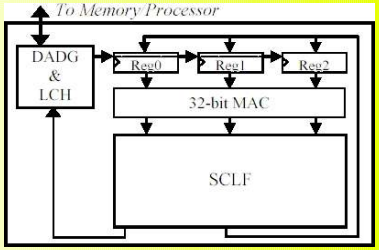
WCLA

- 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.



WCLA

- 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

Tool	Code Size (Lines)	Binary size (Kbytes)	Data size (Kbytes)	Time (s)
Decompilation DMA Config. RT Synthesis	7,203	125	452	0.05
Logic Synthesis Tech. Mapping Place & Route	4,695	88	360	1.04

Details of the tools running on the co-processor.

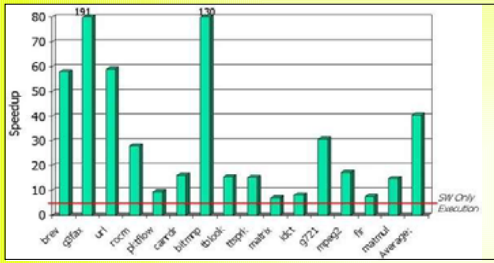
Example	Total Ins	Loop Ins	Loop Time%	Loop Size%	Ideal Speedup
brev	992	104	70.0%	10.5%	3.3
g3fax1	1094	6	31.4%	0.5%	1.5
g3fax2	1094	6	31.2%	0.5%	1.5
url	13526	17	79.9%	0.1%	5.0
logmin	8968	38	63.8%	0.4%	2.8
Avg:			55.3%	2.4%	2.8

Weight of the critical regions in tests.

Example	Sw Time	Sw Loop Time	Hw Loop Time	Sw/Hw Time	S
brev	0.05	0.03	0.001	0.02	3.1
g3fax1	23.50	7.35	0.82	16.98	1.4
g3fax2	23.50	7.39	1.49	17.61	1.3
url	379.90	303.74	13.29	89.45	4.2
logmin	16.32	10.42	0.21	6.12	2.7
Avg:		65.78	3.16	26.03	**2.6**

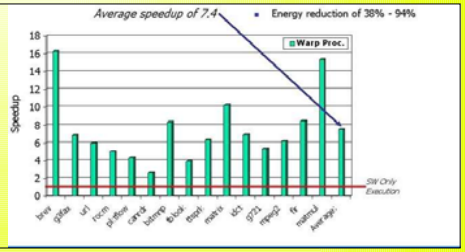
Results using the WARP architecture. The whole process is automated, except binary modification with is currently done by hand.

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% savings 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

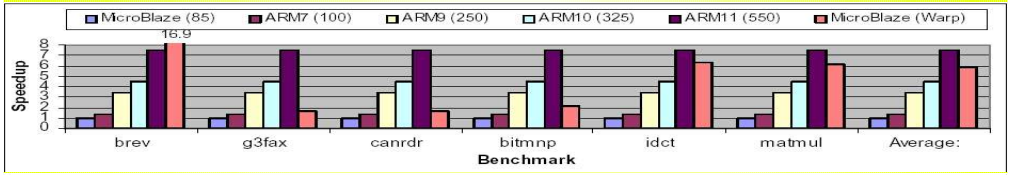
What they did:

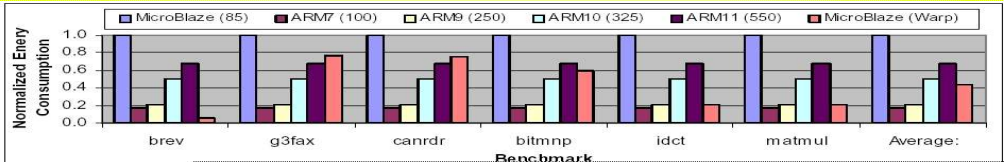
- Modified MicroBlaze to include the profiler.
- Simulated execution on Xilinx apps to obtain program trace.
- Trace used to simulate behaviour of profiler, found single critical region.
- Used ROCPART to generate HW circuits.
- Measured these in a VHDL model of WCLA, combined with traces to obtain final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

- *Frequent Loop Detection Using Efficient Non-Intrusive On-Chip Hardware*, Ann Gordon-Ross and Frank Vahid. http://www.cs.ucr.edu/~vahid/pubs/cases03_profile.pdf
- *Dynamic FPGA Routing for Just-in-Time FPGA Compilation*, Roman Lysecky, Frank Vahida and Sheldon X.-D. Tan. http://www.cs.ucr.edu/~vahid/pubs/dac04_jitfpgaroute.pdf
- *Dynamic Hardware/Software Partitioning: A First Approach*, Greg Stitt, Roman Lysecky and Frank Vahid. http://www.cs.ucr.edu/~rlysecky/papers/dac03-dhsp.pdf
- *A Configurable Logic Architecture for Dynamic Hardware/Software Partitioning*,Roman Lysecky and Frank Vahid http://www.cs.ucr.edu/~rlysecky/papers/date04_clf.pdf
- *A Study of the Speedups and Competitiveness of FPGA Soft Processor Cores using Dynamic Hardware/Software Partitioning*, Roman Lysecky and Frank Vahid http://www.cs.ucr.edu/~vahid/pubs/date05_warp_microblaze.pdf
- *Techniques for Synthesizing Binaries to an Advanced Register/Memory Structure*, Greg Stitt, Zhi Guo, Frank Vahid and Walid Najjar. http://www.cs.ucr.edu/~vahid/pubs/fpga05_binsyn.pdf