A Study of the Speedups and Competitiveness of FPGA Soft Processor Cores using Dynamic Hardware/Software Partitioning

By: Roman Lysecky and Frank Vahid
Presented By: Anton Kiriwas
Disclaimer

- This specific paper is short on implementation details since it focuses on an application case study (6 pages)
- Details have been pulled from 3 other papers
- These papers introduce variations as the work has developed – I do my best not to mix up these variations – but no promises
Papers


Overview

- Introduction and Motivation
- MicroBlaze softcore processor
- MicroBlaze-based Warp Processor
  - Warp Processor background info
- Experiments
- Conclusions
Introduction and Motivation

- Reconfigurable computing offers flexibility
- Designs often require HW/SW co-design and partitioning to implement requirements
- ICs available with combination of RC and microprocessors
Introduction and Motivation (2)

- Companies now selling softcore processors as IP netlists
  - Altera
    - NIOS & NIOS II
      - 32bit configurable core
      - Up to 135 MHz clock speed
  - Xilinx
    - PicoBlaze and MicroBlaze from Xilinx
      - 32 bit configurable core
      - Up to 150 MHz clock speed
  - ARM
Introduction and Motivation (3)

- **DOWNSIDE**… soft processors typically have higher power consumption and decreased performance compared to their hardcore versions.

+ 

- **BUT**… it's been shown that SW/HW partitioning leads to speedups of 200 – 1000% and can reduce power usage by 99%
Related Work

- Dynamic Software Optimization
  - Dynamo – Dynamic binary optimizer
  - BOA – Optimizer for PowerPC
  - Crusoe – Similar from Transmeta

- Runtime Reconfigurable Systems
  - DISC – Dynamically swaps hardware regions into an FPGA
  - Chimaera – Treats reconfigurable logic as a cache of functional units
  - DPGA – Rapidly reconfigures to a predefined configurations
New Research

- Benefits of Warp Processing for Soft Processor Cores
  - Targets Xilinx MicroBlaze

- Goals:
  - Implement system with low cost FPGA
  - Potentially incorporate several processors
  - Increase overall system performance and lower power usage compared to softcore alone
MicroBlaze Soft Processor Background

- Harvard memory architecture
- Dual local memory buses
- On-chip Peripheral Bus
- Synthesized by Xilinx toolchain
- Configurable cache sizes
Warp Processor Background

1. Initially execute application in software only

2. Profile application to determine critical regions

3. Partition critical regions to hardware

4. Program configurable logic and update software binary

5. Partitioned application executes faster and with lower energy consumption
WPG - Processor

- Initially execute application in software only on the microprocessor
- W-FPGA is completely unused during this phase of execution

Note, that in the WORST case, the entire system continues to run like this, which makes slowdown impossible.
WPG - Profiler

- Dedicated profiler watches instruction addresses
- When backwards branching occurs, a small cache of 16 8-bit branch frequencies is incremented.
- Small associativity cache with shift-at-saturation to keep relative values

<table>
<thead>
<tr>
<th>#</th>
<th>Addr</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0xXXXX</td>
<td>11111111</td>
</tr>
<tr>
<td>2</td>
<td>0xYYYY</td>
<td>00011100</td>
</tr>
<tr>
<td>3</td>
<td>0xZZZZ</td>
<td>00000010</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

WPB - ROCPART

- On-chip CAD (also called Dynamic Partitioning Module - DPM in the paper) executes partitioning, synthesis, mapping and routing

ROCPART – Riverside on-chip Partitioning Tool

- Analyzes results from profiler to determine what kernels to implement
- Decompiles uP instructions into control/dataflow graph
- Constructs and synthesizes circuit implementing kernel
- Configure W-FPGA with new circuit
WPB – ROCPART (2)

- Some initial questions to keep in mind…
  - Reconfigurable computing tool-chains typically take anywhere from minutes to hours on complex designs in high-flexibility fabrics.
  - How can we embed this into a dynamic on-chip module and run it fast enough?
WPB – ROCPART (3)

- Riverside On-chip Computer-aided Design Tools
  - Algorithm implemented in soft processor
  - Lean and fast version of toolchain

WPB – ROCPART (4)

- Decompilation Phase
  - Converts each assembly instruction into equivalent register transfers
    - RT is an assignment statement that defines the value of a particular register or memory location
    - Acts as an ISA independent representation
  - Build a control flow graph and data flow graph

WPB – ROCPART (5)

- **Partitioning Phase**
  - Based on a heuristic to determine which kernel would maximize speedup/minimize energy

- **Conversion Phase**
  - Converts the control/data flow graphs into a hardware circuit
  - Hardware circuit into a netlist

WPB – ROCPART (6)

- **Synthesis Phase**
  - JIT FPGA compilation
    - Acyclic graph of the boolean logic network
    - Each node is an AND, OR, XOR, etc.
    - Two-level logic minimizer - Traverse graph nodes in breadth first manner
  - Technology Mapping
    - Greedy hierarchical graph-clustering algorithm
    - Traverses nodes in breadth first manner – combining nodes to form 3-input 2-output LUTs
    - Re-traverses LUT nodes to combine into CLBs utilizing adjacent connections

Placement Phase

- Greedy dependency based positional algorithm
  - Determines the critical path within the circuit and places these nodes into a single row
  - Analyzes dependencies of non-placed nodes on the already placed nodes
    - Place as close as possible to dependent CLB
    - Attempt to utilize adjacent routing resources whenever possible

WPB – ROCPART (8)

- Routing Phase
  - Global Routing - Based on Versatile Place and Route (VPR) algorithm
    - Routes without concern for overroutering
    - Reroutes adjusting cost of each path based on use
  - Detailed Routing – Selecting channels
    - Done via Brelaz’s vertex coloring algorithm

WPB – ROCPART (9)

- Binary Update Phase
  - Updates the original binary by removing instructions with a jump to hardware init code
  - Enables the HW by writing memory mapped registers
  - Then puts processor to sleep
  - Upon completion, an interrupt reawakens the processor
  - A jump then jumps back into original code

MicroBlaze-based Warp Processor

- Simple idea is to put together the idea of the MicroBlaze soft processor and the Warp Processor to dramatically increase performance and power efficiency of a software processor system.
MicroBlaze-based Warp Processor (2)

- Warp Configurable Logic Architecture (WCLA) and MicroBlaze share access to Data BRAM
- Execution of processor and WCLA are mutually exclusive to avoid issues coherency and consistency.
  - Parallel execution does not lead to significant improvement anyway

![Diagram of MicroBlaze-based Warp Processor](image)
Warp Configurable Logic Arch.

- **DADG** – Data Address Generator
  - Generates memory addresses for input/output of the pipeline
- **LCH** – Loop Control Hardware
  - State machine that controls computational machinery inside the pipeline
- 3 input/output registers
- Simplified reconfigurable logic
Simplified Reconfigurable Logic (2)

- Uses a simplified design for the RC fabric
- CLBs inputs/outputs directly connected to the switch blocks
- Algorithm only needs to consider the larger switch matrices
- Carry chains by direct connection between adjacent CLBs
- Place & Route can run 10X faster with 18X less memory
- Tradeoff - Results in lower clock speeds in general
  - Made up for by the dedicated DADG and MAC hardware

Based on previous work
Warp Configurable Logic Arch. (3)

Based on previous work

- Benefits increase on a MP system.
- Reconfigurable fabric and on-chip CAD processor can be shared between the processors, reducing the overhead.
Experiments and Methodology

- Simulation Methodology
  - Simulated the software application on the MicroBlaze using Xilinx Microprocessor Debug Engine for instruction trace
  - Fed instruction trade to the on-chip profiler to determine critical region of application
  - Execute ROCPART
  - Synthesize using Synopsys Design Compiler for UMC 0.18 um technology.
  - Calculate energy consumption of RC hardware using equations below
  - Calculate energy consumption of MicroBlaze using Xilinx XPower estimation tool
  - Comparison to ARM processor using SimpleScalar ARM model

Energy Equations

\[
E_{\text{total}} = E_{MB} + E_{HW} + E_{\text{static}}
\]

\[
E_{MB} = P_{MB\ (idle)} \times t_{idle} + P_{MB\ (active)} \times t_{active}
\]

\[
E_{HW} = P_{HW} \times t_{active}
\]

\[
E_{\text{static}} = P_{\text{static}} \times t_{total}
\]
Experiments and Methodology (2)

- Analyzed execution time and power consumption
- Powerstone and EEMBC benchmark suites
- Chose processor configuration based on `brev` and `matmul` from Powerstone benchmark
  - Include multiplier and barrel shifter
  - No floating point instructions
- Spartan3 FPGA
- MicroBlaze @ 85 MHz
- Remaining circuit up to 250 MHz
Results and Analysis

- Normalized speedup and energy consumption

---

[Graphs showing normalized speedup and energy consumption for different benchmarks and processors.]
Results and Analysis

- Chart made by me – estimating values from previous chart and computing a performance to energy ratio
Conclusions

- Softcore processors are attractive
  - Flexibility in system design and allows for reduction in chip count

- Softcore processors take a hit for energy and performance

- Warp Processing can balance the equation giving reduced energy and increased performance to software processor SoC