The Case for Soft-CPUs in Accelerator Control Systems

Wesley W. Terpstra
GSI Helmholtz Centre for Heavy Ion Research GmbH
Darmstadt, Germany  http://bel.gsi.de
What is a Soft-CPU?

➔ A full CPU
➔ Implemented in HDL
➔ Connected to the FPGA internal SoC bus
Why use a Soft-CPU? (vs. custom HDL)

- ✔ Re-use large body of C/fortran/etc code
- ✔ Complex execution order
- ✔ Dynamic resource management
- ✔ A single component to solve many tasks
- ✔ Easier to debug and trace

✗ Slower
✗ Requires a memory subsystem
Why use a Soft-CPU? (vs. external CPU)

- Physically separate chip
- ARM/etc integrated into FPGA
Why use a Soft-CPU? (vs. External CPU)

✗ Speed
✗ Standard OS and toolchain

✔ Customizable instructions
✔ One less part

✔ Runs synchronously with FPGA bus clock
  • Deterministic timing behaviour
  • Tight integration with custom HDL
Size? ... Memory!

- (Good) Soft-CPUs take ~2% of EP2AGX125
- A single Soft-CPU can run multiple programs

What's the cost of a Soft-CPU?

*The Memory Subsystem*

Trades FPGA gates for memory blocks
The more it does, the more it needs
The true cost of more memory

- More memory = further away from CPU
- Further away = more complicated timing (bus access cycles, prefetch, cache miss)
- MMU makes it even worse (TLB misses)

⇒ Might as well use an external CPU
Soft-CPU non-issues

• Most (good) Soft-CPUs have similar performance
  • Single cycle issue and ~175MHz (Arria2)
  • 3-staged fetch/decode/execute

• Comparable area (at most 3* different)
• Similar executable sizes (32-bit RISC)
Soft-CPUs, but Hard Requirements

– Open Source
  • Portable (Altera, Xilinx, Lattice, ... future proof)
  • Tweakable (custom instructions, bus choice, ...)

– Well documented, tested, and supported
  • Eliminates almost all open source softcores

– Survivors: LM32, LEON3, OpenRISC, (ZPU)
## Feature Comparison

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Purpose</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gcc-toolchain</td>
<td>re-use of C</td>
<td>-</td>
</tr>
<tr>
<td>JTAG access</td>
<td>debug bus, load F/W</td>
<td>ZPU</td>
</tr>
<tr>
<td>Debug support</td>
<td>trace program</td>
<td>ZPU</td>
</tr>
<tr>
<td>Flex. mem bus</td>
<td>max determinism</td>
<td>OR1k</td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
<td>ZPU</td>
</tr>
</tbody>
</table>

**Best choices: LM32 and LEON3**
## Size/speed trade-offs (Ballpark)

<table>
<thead>
<tr>
<th>CPU</th>
<th>Mhz</th>
<th>Size (LEs)</th>
<th>Cycles/mul</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZPU</td>
<td>300</td>
<td>175</td>
<td>400</td>
</tr>
<tr>
<td>LM32</td>
<td>250</td>
<td>175</td>
<td>900</td>
</tr>
<tr>
<td>LEON3</td>
<td>175</td>
<td></td>
<td>2500</td>
</tr>
<tr>
<td>OpenRISC</td>
<td>150</td>
<td></td>
<td>3700</td>
</tr>
</tbody>
</table>

*(All numbers are for an Altera Arria2)*
... and the winner is!

If you need an MMU / linux
LEON3 (or an external CPU)

If you need deterministic execution
If you need a configurable memory bus
If you need small and fast
LM32!
Questions?
LM32 Processor Overview

- 32-bit RISC architecture
  - 32 registers
  - 8-16 control registers
  - no side-effects / flags
- 6-stage pipeline
  - single cycle issue
  - 1-3 cycle result
- Harvard architecture
What's bad about NIOS/Microblaze?

- Not Open-Source
  - Vendor-specific (Altera/Xilinx)
  - Cannot be debugged / signal trapped
  - Cannot be tweaked or fixed

- On the other hand:
  - more comprehensive feature set
  - more complete instruction set
  - vendor specific HDL = slightly smaller
What's bad about ARM/MIPS?

• Patented instruction set
  • Even if you make your own Soft-CPU... bad!

• (If not Open Source):
  • Cannot be debugged / signal trapped
  • Cannot be tweaked or fixed

• On the other hand:
  • more comprehensive feature set
  • more complete instruction set
<table>
<thead>
<tr>
<th>Feature</th>
<th>Size</th>
<th>Impact</th>
<th>Suggestion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction cache</td>
<td>++</td>
<td>++++</td>
<td>KEEP</td>
</tr>
<tr>
<td>Data cache</td>
<td>+++</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Multiplier</td>
<td>.</td>
<td>+</td>
<td>KEEP</td>
</tr>
<tr>
<td>Barrel shifter</td>
<td>.</td>
<td>+</td>
<td>KEEP</td>
</tr>
<tr>
<td>Divider</td>
<td>+++</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>JTAG access</td>
<td>++</td>
<td></td>
<td>DEBUG</td>
</tr>
<tr>
<td>Watch/breakpoints</td>
<td>+</td>
<td></td>
<td>DEBUG</td>
</tr>
<tr>
<td>32 Interrupts</td>
<td>+</td>
<td>++</td>
<td>KEEP</td>
</tr>
</tbody>
</table>