RISC-V Hardware-Accelerated Dynamic Binary Translation

6th RISC-V Workshop

Rokicki Simon - Irisa / Université de Rennes 1
Steven Derrien - Irisa / Université de Rennes 1
Erven Rohou - Inria Rennes
Embedded Systems

Tight constraints in
- Power consumption
- Production cost
- Performance
Systems on a Chip

- Complex heterogeneous designs
- Heterogeneity brings new power/performance trade-off

Overhead from in-order to Out-of-Order

Performance vs. Power Graph

- Out-of-order Superscalar
- In-order core
Systems on a Chip

- Complex heterogeneous designs
- Heterogeneity brings new power/performance trade-off
- Are there better trade-off?
Architectural choice

Out-of-Order processor
- Dynamic Scheduling
- **Performance portability**
- Poor energy efficiency

VLIW processor
- Static scheduling
- No portability
- **High energy efficiency**

Hardware Accelerated Dynamic Binary Translation
The best of both world?

Dynamically translate native binaries into VLIW binaries:

- Performance close to Out-of-Order processor
- Energy consumption close to VLIW processor
Existing approaches

• Transmeta Code Morphing Software & Crusoe architectures
  • x86 on VLIW architecture
  • User experience polluted by cold-code execution penalty

• Nvidia Denver architecture
  • ARM on VLIW architecture

• Translation overhead is critical
• Too few information on closed platforms
Our contribution

• Hardware accelerated DBT framework
  ➢ Make the DBT cheaper (time & energy)
  ➢ First approach that try to accelerate binary translation

• Open source framework
  ➢ Allows research
Outline

• Hybrid-DBT Platform
  • How does it work?
  • What does it cost?
  • Focus on optimization levels

• Experimental Study
  • Impact on translation overhead
  • Impact on area utilization
  • Performance results

• Conclusion & Future work
Outline

• Hybrid-DBT Platform
  • How does it work?
  • What does it cost?
  • Focus on optimization levels

• Experimental Study
  • Impact on translation overhead
  • Impact on area utilization
  • Performance results

• Conclusion & Future work
How does it work?

- RISC-V binaries cannot be executed on VLIW
How does it work?

- Direct, naive translation from native to VLIW binaries
- Does not take advantage of Instruction Level Parallelism
How does it work?

- Build an Intermediate Representation (CFG + dependencies)
- Reschedule Instructions on VLIW execution units
How does it work?

- Code profiling to detect hotspot
- Optimization level 1 only on hotspots
What does it cost?

- Cycle/instr: number of cycles to translate one RISC-V instruction
- Need to accelerate time consuming parts of the translation
Hybrid-DBT framework

- Hardware acceleration of critical steps of DBT
- Can be seen as a hardware accelerated compiler back-end

- RISC-V binaries
- VLIW binaries
- Insert Profiling
- IR Builder
- IR
- IR Scheduler

Optimization level 0
- First-Pass Translation
- No ILP

Optimization level 1
- ILP

Hardware Accelerated Dynamic Binary Translation
Optimization level 0

- Implemented as a Finite State Machine
  - Translate each native instruction separately
  - Produces 1 VLIW instruction per cycle
  - 1 RISC-V instruction => up to 2 VLIW instructions
- Simple because ISA are similar
Optimization level 1

- Build an higher level intermediate representation
- Perform Instruction Scheduling

- Critical to start exploiting VLIW capabilities
Choosing an Intermediate Representation

IR advantages:
- Direct access to dependencies and successors
- Regular structure (no pointers / variable size)

Hardware Accelerated Dynamic Binary Translation
Details on hardware accelerators

- Developing such accelerators using VHDL is out of reach
- Accelerators are developed using High-Level Synthesis
  - Loops unrolling/pipelining
  - Memory partitioning
  - Memory accesses factorization
  - Explicit forwarding

See DATE’17 paper for more details!
Outline

• Hybrid-DBT Platform
  • How does it work?
  • What does it cost?
  • Focus on optimization levels

• Experimental Study
  • Impact on translation overhead
  • Impact on area utilization
  • Performance results

• Conclusion & Future work
Impact on translation overhead

- VLIW baseline is executed with ST200simVLIW
- **Fully functionnal** Hybrid-DBT platform on FPGA
  - JIT processor: Nios II
  - Altera DE2-115

![Diagram of VLIW baseline translation process](image-url)
Impact on translation overhead

- Cost of **optimization level 0** using the hardware accelerator
Impact on translation overhead

- Cost of **optimization level 1** using the hardware accelerator
Impact on area/resource cost

- Resource usage for all our platform components
  - ASIC 65nm

![NAND equivalent gates chart](chart.png)

- Overhead from Hybrid-DBT

Hardware Accelerated Dynamic Binary Translation

25
Performance results

- Comparison against OoO architectures
  - Compare area, power consumption and performance with BOOM

Power Consumption (mW)

- BOOM
- VLIW4

5x

Speed-up OoO vs VLIW

- adpcm dec
- g721 dec
- g721 enc
- matmul

Stay at opt level 0
Conclusion

• Presentation of Hybrid-DBT framework
  • Hardware accelerated DBT
  • Open-source DBT framework RISC-V to VLIW
  • Tested FPGA prototype

• Sources are available on GitHub: https://github.com/srokicki/HybridDBT
Simty: a synthesizable SIMT CPU

- GPU-style SIMT execution assembles vector instructions across threads of SPMD applications
  - Alternative to vector processing based on scalar ISA

- Simty: proof of concept for SIMT on general-purpose binaries
  - Runs the RISC-V instruction set (RV32I), no modification
  - Written in synthesizable VHDL
  - Warp size and warp count adjustable at synthesis time
  - 10-stage in-order pipeline

More details on [https://team.inria.fr/pacap/simty/](https://team.inria.fr/pacap/simty/)!!

- Scales up to **2048 threads per core** with 64 warps × 32 threads
Questions

https://github.com/srokicki/HybridDBT
https://team.inria.fr/pacap/simty/
FPGA synthesis of Simty

On Altera Cyclone IV

- Up to 2048 threads per core: 64 warps \times 32 threads
  - Sweet spot: 8x8 to 32x16

- Overhead of per-PC control is easily amortized