Under 100-cycle Thread Migration Latency in a Single-ISA Heterogeneous Multi-core Processor

Elliott Forbes, Zhenqian Zhang, Randy Widialaksono, Brandon Dwiel, Rangeen Basu Roy Chowdhury,
Vinesh Srinivasan, Steve Lipa, Eric Rotenberg, W. Rhett Davis, Paul D. Franzon

Motivation

Single-ISA Heterogeneous Multi-core [1]: General purpose cores with different microarchitectures, tuned for different energy/performance points.

Performance and energy of a program can be optimized by migrating among the core types as program characteristics change. Prior research [2] has shown as much as a 50% improvement in energy when migrating every 1,000 cycles versus every 10,000 cycles. Such fine-grained thread migration requires very low migration overhead.

We propose hardware support for fast thread migration. To migrate a thread, committed register values and the program counter must be moved from the source core to the destination core.

Fast Thread Migration

Central to our fast thread migration is a Teleport Register File (TRF), which supports a one-cycle exchange of all TRF bitcells with another TRF [2,7].

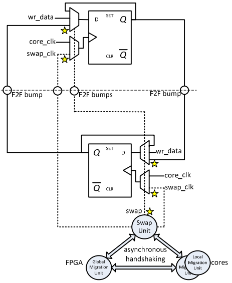
- Uses flip-flop based bitcells
- TRFs of the cores can be asynchronous to each other (GALS)
- Swap Unit used to control bitcell clocks and perform actual value exchange

Modern processors hold committed and speculative register values in a large physical register file.

- PRFs of cores are different sizes
- Register renaming means committed values may not be in contiguous registers
- PRF is implemented in SRAM

Keep PRF for normal program execution, add a TRF to support fast thread migration

- Must consolidate committed registers from PRF of a core to its TRF
- New instructions: move-to-TRF (MTTRF), move-from-TRF (MFTRF)



Migration Modes

Local Migration: migration is initiated by the program itself via a new MIGRATE instruction.

- Compiler can find potentially beneficial places to migrate, and move only live register values required for correct program execution.

Global Migration: when to migrate is determined outside the cores, taking activities of all cores into account.

- Migration could come at any time, so all registers must be transferred for correct program execution.

Migration Overheads

Local migration

- User program moves live register values to TRF
- Execute a MIGRATE instruction
- Swap Unit performs value exchange, then signals cores to resume
- Move register values from TRF
- Continue user program

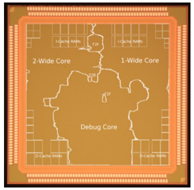
Global migration

- External interrupt initiates a pending migration, causing a pipeline flush in each core
- Interrupt handler called to move all register values to TRF and then suspend pipeline
- Swap Unit performs value exchange, then signals cores to resume
- Move register values from TRF
- Continue user program(s)

Other side-effects in either mode

- Repeat some cache misses (migration-induced misses)
- Predictor state (branch predictors, prefetchers, etc.) may be cold

Implementation



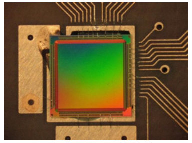
Phase 1: 2D test chip used to verify logic functionality

- Taped-out May 2013
- Two heterogeneous core types, a one-wide out-of-order core with small microarchitectural structures, and a two-wide out-of-order core with large microarchitectural structures
- Fast Thread Migration (FTM) hardware to support fast exchange of register values between the two cores
- Ability for a core to access either its own L1 caches or the other core's L1 caches, we call this Cache-Core Decoupling (CCD)

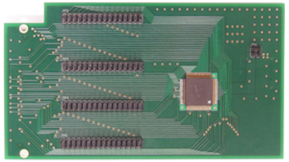
Phase 2: 3D stacked chip to prove concept, gather performance results

- Expected tape-out August 2015
- Same cores, FTM, and CCD as Phase 1
- One core on each tier
- Added support for cache coherence between L1 caches

Chips of both phases mounted on PCB with mezzanine connector, mated to Xilinx ML605 FPGA development board



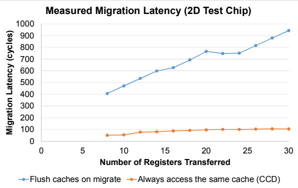
	Phase 1 (2D test chip)
Technology	IBM 8RF (130 nm)
Dimensions	5.25 mm x 5.25 mm
Area	27.6 mm^2
Transistors	14.6 Million
Cells	1.1 Million
Nets	721 Thousand
Memory macros	56
Clock domains	10

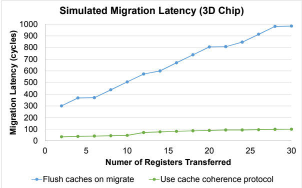


Compared to Other Architectures

	Thread migration latency	Distinct (separate) cores	Asynch-ronous (GALS)	Register-based ISA	Evaluation methodology
ARM big.LITTLE [4]	20,000 cycles	Yes	Yes	Yes	Real system
Composite cores [3][5]	≤ 32 cycles	No (shared frontend and data cache)	No	Yes	C++ simulator
Execution Migration Machine [6]	< 100 cycles	Yes	No	No (stack-based ISA for partial context transfer)	RTL simulation and synthesis; chip fabricated, measurements not yet reported
Our approach, FTM, before chip bringup [2]	< 100 cycles	Yes	Yes	Yes	C++ simulator for architecture results, RTL and layout for tapeout description
Our approach, FTM, in this Hot Chips submission	< 100 cycles	Yes	Yes	Yes	chip

Results





References

[1] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, D. M. Tullsen. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. MICRO-36, 2003.

[2] E. Rotenberg, B. Dwiel, E. Forbes, Z. Zhang, R. Widialaksono, R. Basu Roy Chowdhury, N. Tshibangu, S. Lipa, W. R. Davis, and P. D. Franzon. Rationale for a 3D Heterogeneous Multi-core Processor. ICCD-31, 2013.

[3] S. Padmanabha, A. Lukefahr, R. Das, and S. Mahlke. Trace Based Phase Prediction for Tightly-Coupled Heterogeneous Cores. MICRO-46, 2013.

[4] P. Greenhalgh. Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7. http://www.arm.com/products/processors/technologies/biglittleprocessing.php.

[5] A. Lukefahr, S. Padmanabha, R. Das, F. Sleiman, R. Dreslinski, T. Wenisch, S. Mahlke. Composite Cores: Pushing Heterogeneity into a Core. MICRO-45, 2012.

[6] M. Lis, K.-S. Shim, B. Cho, I. Lebedev, and S. Devadas. Hardware-level Thread Migration in a 110-core Shared-Memory Processor. Hot Chips 25. 2013.

[7] Z. Zhang, B. Noia, K. Chakrabarty, and P. Franzon. Face-to-Face Bus Design with Built-in Self-Test in 3D ICs. 3D Systems Integration Conference, 2013.

Center for Efficient, Scalable and Reliable Computing
Department of Electrical and Computer Engineering, North Carolina State University