Heterogeneous-ISA Polymorphic Architecture: Exploiting ILP-TLP tradeoffs above ISA affinity

Dual Degree Project Stage 1 Report

Submitted in partial fulfillment of the requirements
for the

Dual Degree Programme

by

Prakhar Diwan
(Roll No. 180100083)

Under the guidance of

Prof. Virendra Singh

Department of Electrical Engineering
Indian Institute of Technology Bombay
October 2022

Acknowledgement

I express my gratitude to my guide, Prof. Virendra Singh for allowing me to work on this topic. I am also grateful to Nirmal Kumar Boran and other members of Computer Architecture & Dependable Systems Lab (CADSL) for their support.

Prakhar Diwan
Electrical Engineering
IIT Bombay

Abstract

Modern workloads demand a microarchitecture that can adapt to software diversity by exploiting instruction level parallelism (ILP) from their sequential and thread level parallelism (TLP) from their parallel phases to achieve high performance and energy efficiency. After multi-core's inception, various solutions have been proposed to utilize heterogeneity and reconfigurability in microarchitecture. Prior research has shown ISA as another dimension of heterogeneity, providing significant performance and energy efficiency gains over single-ISA heterogeneous architectures.

This work proposes a reconfigurable heterogeneous-ISA multi-core architecture, which exploits the heterogeneity in ISA and microarchitecture for achieving high performance and energy efficiency. It will schedule the workload phases on optimal configuration (reconfigured at runtime) with respect to ISA affinity & ILP-TLP trade-offs, hence providing significant is aimed for both single & multithreaded workloads.

Contents

List of Figures **2**

1 Introduction **4**

2 Literature Review **6**

2.1 Heterogeneity in Microarchitecture 6

2.2 Heterogeneity in ISA 8

3 Proposed Work **9**

3.1 Introduction 9

3.2 Completed Work 9

3.3 Work to be done 10

List of Figures

1.1 Homogeneous Multicore 4
1.2 Single-ISA Heterogeneous Multicore 4
1.3 Heterogeneous-ISA Multicore 5

2.1 Composite Cores Architecture 6
2.2 MorphCore Microarchitecture 7
2.3 Core Fusion Conceptual Floorplan 7

Chapter 1

Introduction

During the last decades of the previous century, the central aim of microprocessor technology was to make the core faster. Performance was improved through semiconductor technology advancements and architectural innovations. However, by the end of the last century, the power density became equal to that of the hot plate, restricting the further increase in frequency for performance.

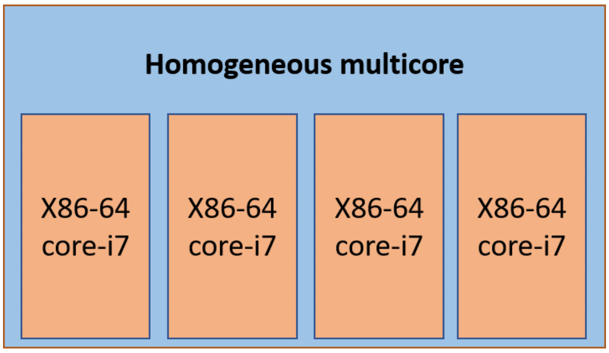

Figure 1.1: Homogeneous Multicore

The above limitation, termed as" power wall" forced the designers to shift to novel architectures. Researchers came up with multi-core systems to better cater to the need of applications. Overall performance was improved by exploiting Thread Level Parallelism (TLP) of multithreaded workloads through multiple cores. But these improvements saturated with single-threaded fraction of workloads becoming the bottleneck[1]. Also, the move to various cores led to an increase in energy consumption.

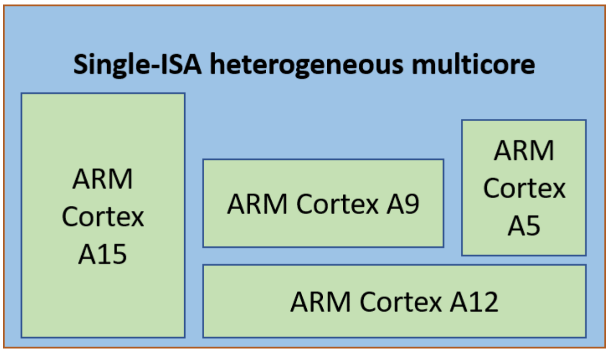

Figure 1.2: Single-ISA Heterogeneous Multicore

The initial proposed multicore architecture consisted of identical cores (Figure 1.1). Prior research has shown that exploiting heterogeneity is beneficial for a system's performance[2]

& power[3]. Architects utilized the heterogeneity in two dimensions, the first was using specialized cores for certain workloads (e.g., SIMD support), and the second was using cores with different microarchitectures (Figure 1.2) based on workload's needs, to extract high performance & energy efficiency.

Till early 2010s, the heterogeneity exploration was limited to single-ISA due to the high migration cost between heterogeneous-ISA systems. DeVuyst et al. [4] solves the problem of migration between heterogeneous-ISA cores, and Venkat et al. [5] highlighted the benefits brought by heterogeneous-ISA multicore (Figure 1.3).

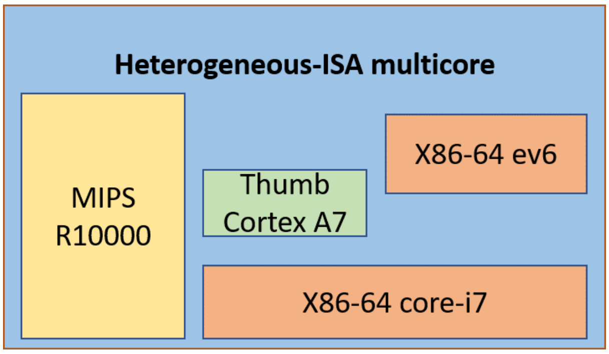

Figure 1.3: Heterogeneous-ISA Multicore

Another direction of research for improving single-threaded performance in multi-core systems was reconfigurable chip multiprocessors, where a group of independent cores can dynamically morph into a large core. These architectures could exploit Instruction Level Parallelism (ILP) in sequential phases of code by using the large core and effectively utilizing TLP from parallel phases by operating as small independent cores. These types of architectures are unexplored for heterogeneous-ISA cores.

Heterogeneous-ISA Dynamic Core (HIDC) has already been proposed, which exploits ISA (ARM, X86) and microarchitectural heterogeneity. The next chapter describes the research on dynamic & heterogeneous multi-core architectures. Heterogeneous-ISA Dynamic Multi-core (HIDM) architecture is proposed to obtain more benefits from heterogeneity. It is a reconfigurable multi-core architecture aimed at achieving high performance & energy efficiency. It is expected to perform well for single-threaded (SPEC CPU2006 suite [6]) as well as multithreaded workloads (PARSEC suite [7]) by dynamically catering to ILP & TLP needs.

Chapter 2

Literature Review

This chapter describes the previous work on heterogeneous multi-core architectures. The first section gives insight into various ways microarchitectural heterogeneity has been exploited by summarizing some proposed ideas. The second section discusses proposals associated with ISA heterogeneity: solution to migration between heterogeneous ISA cores, harnessing performance and energy benefits from ISA diversity, and claims towards ISA homogeneity.

2.1 Heterogeneity in Microarchitecture

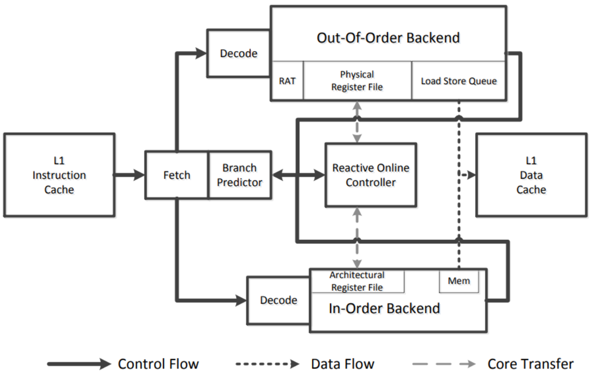

Figure 2.1: Composite Cores Architecture

Lukefahr et al. [8] proposed Composite Cores, an architecture which reduced migration overheads by accommodating heterogeneity within a core. The core consisted of a big (out-of-order) and a little (in-order) compute μengines (Figure 2.1). Both shared most of the architectural state leading to reduction in data transfer during migration between them, which led to lower switching overheads. This enabled fine-grained matching of application characteristics. An online reactive controller is used for making switching decisions, aiming to maximize energy savings under user configurable performance degradation constraint. It uses a linear regression model to predict CPI (cycles per instruction) for the μengine not being used in the current phase, and compares the CPI difference against the dynamic CPI threshold, if larger then execution is done on big μengine else on little μengine.

Suleman et al. [9] came up with the idea of MorphCore, an adaptive core microarchitecture to achieve high performance for single-threaded code and high throughput for multi-threaded code with no energy overheads. A large 2-way SMT (simultaneous multi-threading) out-of-order (OoO) core (capable of exploiting ILP) is used as the base substrate. Upon it minimal modifications are done to dynamically form a 8 SMT in-order (InO) core. Hence it provides two modes of execution (Figure 2.2): OoO and InO, and uses OoO mode (2-way SMT OoO core) for single-threaded code to provide high performance of a traditional OoO core with minimal degradation. For multithreaded code, it uses the InO mode (8-way SMT InO core) which provides same or better performance as an OoO core [10] and consumes less power. Due to no migration of instructions and data needed, switching overhead was negligible.

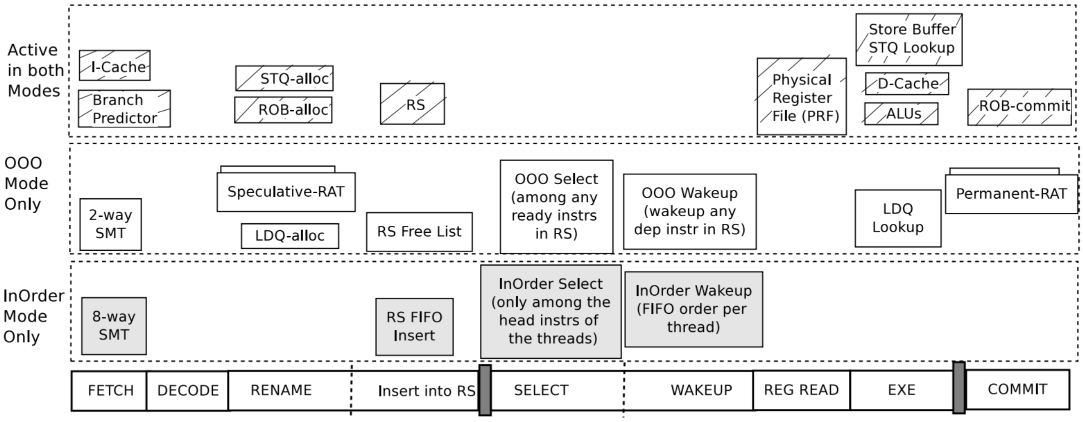

Figure 2.2: MorphCore Microarchitecture

Ipek et al. [11] presented core fusion, a reconfigurable chip multiprocessor (CMP) architecture which empowers groups of fundamentally independent CMP cores with the ability to "fuse" into a large CPU on demand from application. To effectively exploit ILP from sequential phases of application, it uses the fused mode configuration to create a large core, and for parallel phases it utilizes independent mode (base multicore). It is a fully hardware-based solution which uses hints from applications itself for making reconfiguration decisions. The CMP architecture consists of eight two-issue OoO cores with private L1-i and L1-d caches and common L2 cache, the figure 2.3 shows an example configuration of core fusion: one eight-issue (group of four CMP cores), one four-issue (group of two CMP cores) and two two-issue cores. Based on results shown for sequential, parallel and incrementally parallelized workloads, it was concluded that core fusion accommodates software diversity gracefully. Pricopi et al. [12] proposed Bahurupi, another reconfigurable homogeneous multicore architecture (comprising identical 2-way OoO cores), with aim to satisfy the conflicting demands of ILP and TLP put forth by

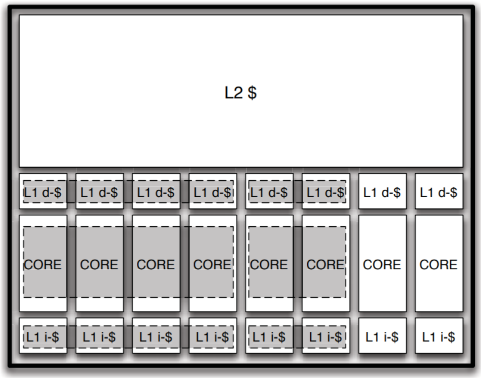

Figure 2.3: Core Fusion Conceptual Floorplan

workloads comprising fractions of parallel and sequential codes. A high ILP application can be accelerated by execution on a large core dynamically composed of four CMP-cores, a medium ILP application was scheduled on a 2-core coalition and low ILP applications were run on individual CMP cores. This architecture utilized a distributed execution model (compiler-hardware mixed solution) and achieved performance of equivalent fetch width complex OoO superscalar core without paying for complex hardware and associated energy inefficiency.

2.2 Heterogeneity in ISA

Devuyst et al. [4] claimed ISA diversity as an important dimension to gain performance using heterogeneity in cores. Multiple reasons for migration need were mentioned such as higher priority process being scheduled to high performance core with its currently executing process migrated to energy-efficient core or the process from high performance is switched to energy efficient core in battery saver mode. Authors listed various methods used during heterogeneous-ISA migration to reduce the overhead. They have modified compiler back end significantly to produce programs that keep most of their state in architectural neutral form. For maintaining stack consistency a stack transformer is used during migration. And for enabling instantaneous migration a binary translator is modified as required by the work and utilized.

Venkat et al. [5] carried out the design space exploration for finding optimal design to propose a heterogeneous-ISA CMP for general purpose computing. Various parameters affected by ISA diversity and their impact on performance was analyzed. The program was migrated between different ISA cores at equivalence points. In case not at equivalence point, the program is binary translated to the target ISA (to be migrated to) till the nearest equivalence point is reached in execution. Authors examined phase wise affinity for various program phases and obtained execution times taken for switching between different ISA cores. Upon scheduling the program phases on the best-suited core using oracle scheduler, improvements of 20.8% in performance and 23% in energy efficiency were observed over single-ISA design.

To harness maximum benefits from ISA heterogeneity, dynamic determination of affine ISA for the next phase of the program is necessary. Boran et al. [13] proposed a performance model for estimating execution time of heterogeneous ISA system. They extend the model further for making dynamic coarse-grained [14] and fine-grained [15] schedulers for heterogeneous-ISA systems.

Blem et al. [16] tried to find if the ISA plays an intrinsic role in performance and energy efficiency of the implemented system. For this authors performed detailed experiments by running real applications from various workloads on real hardware, and analysed the measurements to claim that ISA being RISC or CISC doesn't matter. However, this is not true as: using real hardware led to microarchitecural differences among considered systems, the platform utilized was not uniform across the four implementations, no ISA-specific compiler optimizations were performed for binaries, approximate scaling models have been used for frequency and technology nodes, plus these don't consider the type of transistor technology used like MOSFET, FinFET, CasFET etc. and many other reasons. Authors themselves listed multiple other infrastructural limitations in the paper. The claim made is weak as ARM and X86 systems aren't provided equal stages to be compared justly.

The next chapter describes the proposed work: introducing the problem, completed work and work to be done.

Chapter 3

Proposed Work

3.1 Introduction

Based on the observations from the literature review, Heterogeneous-ISA Dynamic Multicore (HIDM), a reconfigurable CMP architecture is proposed to achieve high performance energy efficiency by meeting ILP-TLP ISA affinity needs at runtime. ARM and X86 are the 2 ISAs considered based on their current popularity. As the base core, we will use Heterogeneous ISA Dynamic Core's (8-way OoO) smaller version (2-way OoO). For analyzing the performance of merged cores, we will be using corresponding multiplied width cores, like a group of 2 2-way OoO cores modeled using a 4-way core. This is conservative based on results from Bahurupi [12].

Another proposal is to make each of the 2-way OoO cores into an 8-way SMT InO core for highly threaded applications, which has been shown by Suleman et al. [9]. This will help in extending the performance and energy benefits of reconfigurability for a higher number of threads (32 for a four-core system).

3.2 Completed Work

I conducted an extensive literature review on various multi-core architectures, installed the gem5 simulator [17], and ran simple experiments for familiarization, completed the setup of the full system mode simulation environment for `ARM` and `X86` configurations.

I compiled the original 13 workloads from PARSEC [7] suite using `gcc 5.5.0` for `ARM` and `X86` ISAs. I faced issues with compilation of some benchmarks, as the suite was last updated in 2011 where they used `gcc 4.2.1` (unsupported). Also as majority of benchmarks use the `pthreads` library I have chosen to use it, but it doesn't support `freqmine` benchmark. As a result some of the benchmarks will be missing in analysis from the 13 original benchmarks.

For choosing the optimal configuration in reconfigurable multi-core architecture, it is required to obtain the active number of threads at runtime, based on which scheduling is done, as in work by Suleman et al. [9]. I performed an exhaustive search inside gem5 documentation and the internet to obtain this information by modifying gem5. During the investigation, I backtraced the `activateContext` and `suspendContext` functions, which led to various functions, ending with a stream of calls from shared libraries. I investigated the functions, but it led nowhere as the kernel handles the thread scheduling and has thread-related information; and in full system mode, the kernel is provided by the user as an unmodified binary along with the disk image. Hence, recovery of the active number of threads from gem5 FS mode wasn't possible. But through modifications in gem5, I could obtain the number of physical threads (same as the number of active cores) at runtime.

Then I made minimal modifications to the benchmarks to incorporate a thread counter which decremented on every thread termination, thus depicting the number of live threads at runtime. The counter is a global variable set to the number of threads created by the user and is decremented in the critical section formed using `mutex` locks and unlocks from `pthread` library. This addition made a negligible difference in results, as was confirmed from a test run of `blackscholes` benchmark with medium data-set on `ARM` simulated machine. I have tested the approach for counting threads on `blackscholes, bodytrack & canneal` benchmarks.

For the X86 full system (FS) simulations I used an available disk image with the compiled binaries from PARSEC 2.1 suite earlier for testing purpose. But now as I need to perform edits in the benchmarks, I'll have to compile them. I tried doing this using host machine (X86), but the application when run on simulated machine gave `glibc` version issue.

The work of finding the dynamic instruction counts at `mutex_unlock` accesses is in progress, I extracted the pcaddress for the function using objdump but it's not matching with any of the pcadresses used by the simulated machine.

Experiments on ILP-TLP trade-offs are being performed on the ARM-based simulated machine. I am facing the issue where I cross-compiled the benchmarks on host but they aren't being recognized as a binary leading to exec format error. I am working to fix this issue as well. Scripts for these experiments have been written.

3.3 Work to be done

- Experiments on ILP-TLP trade-offs: Running PARSEC benchmarks: with 4 threads on system with 4 2-way OoO cores; with 2 threads on system with 2 4-way OoO cores; with 1 thread (or serial) on system with 1 8-way OoO core. **Ongoing**

- Appropriate compilation of PARSEC 3.0 benchmarks for X86 ISA so that it works for gem5 simulated machine. (host machine compiled application gave `glibc` version issue on simulated machine)

- Obtaining dynamic instruction count after every `mutex_unlock` and dumping the simulation statistics. For this need to check the address of this function from disassembly of compiled binary, and check whenever this address is accessed and get the dynamic instruction count. **Ongoing**

- Analysis of statistics for each phase run on each configuration to determine optima. Approximation of reconfiguration delays

References

[1] G. M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS ’67 (Spring), (New York, NY, USA), p. 483–485, Association for Computing Machinery, 1967.

[2] R. Kumar, D. Tullsen, P. Ranganathan, N. Jouppi, and K. Farkas, “Single-isa heterogeneous multi-core architectures for multithreaded workload performance,” in *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, pp. 64–75, 2004.

[3] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen, “Single-isa heterogeneous multi-core architectures: the potential for processor power reduction,” in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pp. 81–92, 2003.

[4] M. DeVuyst, A. Venkat, and D. M. Tullsen, “Execution migration in a heterogeneous-isa chip multiprocessor,” in *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, (New York, NY, USA), p. 261–272, Association for Computing Machinery, 2012.

[5] A. Venkat and D. M. Tullsen, “Harnessing isa diversity: Design of a heterogeneous-isa chip multiprocessor,” in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pp. 121–132, 2014.

[6] J. L. Henning, “Spec cpu2006 benchmark descriptions,” *SIGARCH Comput. Archit. News*, vol. 34, p. 1–17, sep 2006.

[7] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The parsec benchmark suite: Characterization and architectural implications,” in *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 72–81, 2008.

[8] A. Lukefahr, S. Padmanabha, R. Das, F. M. Sleiman, R. Dreslinski, T. F. Wenisch, and S. Mahlke, “Composite cores: Pushing heterogeneity into a core,” in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 317–328, 2012.

[9] Khubaib, M. A. Suleman, M. Hashemi, C. Wilkerson, and Y. N. Patt, “Morphcore: An energy-efficient microarchitecture for high performance ilp and high throughput tlp,” in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 305–316, 2012.

[10] S. Hily and A. Seznec, “Out-of-order execution may not be cost-effective on processors featuring simultaneous multithreading,” in *Proceedings Fifth International Symposium on High-Performance Computer Architecture*, pp. 64–67, 1999.

[11] E. Ipek, M. Kirman, N. Kirman, and J. F. Martinez, "Core fusion: Accommodating software diversity in chip multiprocessors," ISCA '07, (New York, NY, USA), p. 186–197, Association for Computing Machinery, 2007.

[12] M. Pricopi and T. Mitra, "Bahurupi: A polymorphic heterogeneous multi-core architecture," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, Jan 2012.

[13] N. K. Boran, R. P. Meghwal, K. Sharma, B. Kumar, and V. Singh, "Performance modelling of heterogeneous isa multicore architectures," in *2016 IEEE East-West Design Test Symposium (EWDTS)*, pp. 1–4, 2016.

[14] N. K. Boran, D. K. Yadav, and R. Iyer, "Classification based scheduling in heterogeneous isa architectures," in *2020 24th International Symposium on VLSI Design and Test (VDAT)*, pp. 1–6, 2020.

[15] N. K. Boran, S. Rathore, M. Udeshi, and V. Singh, "Fine-grained scheduling in heterogeneous-isa architectures," *IEEE Computer Architecture Letters*, vol. 20, no. 1, pp. 9–12, 2021.

[16] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1–12, 2013.

[17] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, p. 1–7, aug 2011.