ReKonf: DYNAMICALLY RECONFIGURABLE MULTICORE ARCHITECTURE

RAJESH KUMAR PAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY DELHI

SEPTEMBER 2015

ReKonf: DYNAMICALLY RECONFIGURABLE MULTICORE ARCHITECTURE

by

RAJESH KUMAR PAL

Department of Computer Science and Engineering

Submitted

in fulfilment of the requirements of the degree of

Doctor of Philosophy

to the

Indian Institute of Technology Delhi
September 2015

Dedicated to my beloved wife *Sunita* and my *Dad*

Certificate

This is to certify that the thesis titled ***ReKonf*: Dynamically Reconfigurable Multicore Architecture** being submitted by **Mr. Rajesh Kumar Pal** for the award of **Doctor of Philosophy** in Computer Science and Engineering is a record of bona fide work carried out by him under our guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Delhi. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Sanjiva Prasad
Professor
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi, India 110 016

Place: New Delhi
Date: Sep 15

Kolin Paul
Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi, India 110 016

Place: New Delhi
Date: Sep 15

Acknowledgments

First and foremost, I would like to express my profound gratitude and sincere thanks to my supervisors Prof Kolin Paul and Prof Sanjiva Prasad. They taught me how to do research, how to think and approach a problem. Without their guidance, support, encouragement, deep-concern, and faith in me, it would not have been possible to complete this long academic journey. Prof Kolin Paul gave me freedom to pursue my areas of interest with necessary course corrections, much-needed understanding of my situations while I was balancing my job and research, setting up goals to keep me focused, and of course chasing me up whenever I slowed down. He has been more of a friend than a mentor who eases out pressure and creates an environment that encourages perseverance, exploration, and excellence. I am equally indebted to Prof Sanjiva Prasad for teaching me how to organize my thoughts, how to formalize my research results, and finally how to reduce them into technical writing. The manuscript quality always improved manifolds after it goes through his perfection-seeking and observing eyes. He motivated and boosted up my morale when I needed the most. It has been an honor to work with you sir.

The embedded systems group at CS&E department, in which we have renowned faculty members Prof M. Balakrishnan, Prof Anshul Kumar, Prof Preeti R. Panda, Prof Kolin Paul, and Prof Smruti Ranjan Sarangi, provided the much needed ecosystem to research in the area of architecture and embedded systems. I am thankful to the group for their valuable feedback on many occasions and to Prof Smruti Ranjan Sarangi for reviewing and criticizing a few of my papers. I also received support from Prof Naveen Garg in the topics of algorithms.

When I started my Ph.D. at IIT Delhi in 2008, I received valuable insight and guidance from my senior colleagues Neeraj Goel, Aryabartta Sahu, Anant Vishnoi, Lava Bhargava, and Sonali Chouhan. I threw many of my ideas to them, had got valuable feedbacks, and gained from their experiences. I am greatful

to my friends B. Sharat Chandra Varma, Arun Parakh, Sohan Sharma, Vaibhav Jain, Yamuna Prasad Shukla, Prathmesh Kallurkar, Sandeep Chandran, Rajshekar Kalyappan, Gayathri Ananthanarayanan, and Rajeswari Devadoss for their helps, cooperations, suggestions, discussions, and feedbacks on my work.

I would also like to thank the staff members of Computer Science department especially Ms. Vandana Ahluwalia, Mr. Som Dutt Sharma, Mr. Kali Ram Kaushik, Mr. Rajesh Kumar and Ms. Renu Bhatia for their help. They have opened their hearts and their doors, supplying me with resources I never could have found without them.

I am grateful to Prof Indranil Sengupta (my M.Tech. supervisor) and Prof Dipanwita Roychowdhury of IIT Kharagpur for encouraging and inspiring me to pursue Ph.D. Without your inspiration, I may not have started this journey itself.

I am thankful for the unconditional support and cooperation received from my IAF colleagues Girish Atrawalkar, Vikram Sinh Ghorpade, S.K. Upreti, S. Rajshekar, and Gunasekaran Dharmalingam. I would always remain grateful to Girish Atrawalkar for keeping confidence in me and providing the freedom to chase my goal.

When I look back the past 7 years, I find one constant that is the support and encouragement of my childhood friends Raghavendra Kumar Singh, Debashish Nath, Bhaskar Bora, Parag Mulay, Sachidanand Tripathi, Rajeshwar Pal, and Dr. Rajendra Prasath.

Finally I would like to thank my beloved wife Sunita for her patience, love, understanding, faith and standing beside me in this entire journey. My sons Aarush and Arnav gave me tremendous joy and energy to consistently move forward. I always received timely-advice, encouragement, and never-say-die attitude towards completion from my father Ramjee Pal. He would be the happiest person as I complete my Ph.D.

Rajesh Kumar Pal

Abstract

Applications differ in their computational requirements. A single application too can have diverse requirements during its different phases. In this thesis, we present ReKonf, a dynamically reconfigurable tile based multicore architecture that detects the program phase change at runtime and morphs itself into different configurations to suit the program phase behavior. We use space and time efficient performance counters for estimating performance gains from candidate architectural configurations. In our design space exploration we have identified the suitability of architectural components namely, number of cores, cache size, and cache sharing, for reconfigurability. To select the right configuration for a workload, we develop three cache reconfiguration and core clustering techniques namely, static, dynamic and adaptive reconfiguration techniques. The thesis shows that executing an application on the "right" configuration significantly improves performance over fixed architectures.

Further we explore the reconfiguration opportunities in an embedded multicore domain where saving energy is as important a goal as enhancing performance. Choosing video decoding as a typical application, we develop a dynamic core allocation algorithm for video decoding on embedded multicore platforms with the objective of reducing energy consumption while guaranteeing a quality of service (QoS). We show that substantial energy savings can be achieved on multicore architectures by employing dynamic core allocation. Finally we combine cache reconfiguration and dynamic core allocation in the ReKonf architecture and evaluate performance enhancement and energy preservation for video decoding. After obtaining the minimum number of cores for decoding a frame, we utilize the ReKonf infrastructure to identify the most beneficial cache configuration for that number of cores. The ReKonf architecture configures to the assigned number of cores and its associated cache configuration, before it decodes a frame. Our methodology may be used in embedded platforms for achieving energy savings by employing a combination of core allocation and cache reconfiguration.

Contents

List of Figures v

List of Tables ix

List of Abbreviations xi

1 Introduction 1

1.1 Background . . . 1

1.1.1 Multicore Architectures . . . 1

1.1.2 Motivations . . . 6

1.2 Dynamic Reconfigurability in Multicores . . . 7

1.2.1 Challenges in Dynamic Reconfiguration . . . 9

1.3 Objective . . . 10

1.4 Thesis Contributions . . . 10

1.5 Thesis Organization . . . 15

1.6 Summary . . . 17

2 Related Work 19

2.1 Reconfiguration . . . 19

2.1.1 Core Reconfiguration . . . 19

2.1.2 Cache Reconfiguration . . . 23

2.2 Adaptation . . . 27

2.2.1 Dynamic Scheduling . . . 28

2.2.2 Dynamic Power Management 30
2.2.3 Optimizations based on DVS/DVFS 31
2.2.4 Other Related Work 32
2.3 Summary 32

3 Design Space Exploration for Reconfigurability 33
3.1 Experimentation Methodology 35
3.1.1 Simulation Platform 35
3.2 Benchmarks 41
3.3 Experimental Results and Analysis 44
3.3.1 Cache Size 45
3.3.2 Number of Cores 46
3.3.3 CMP vs SMP 48
3.3.4 Cluster Size 49
3.4 Candidate Components for Reconfigurability 50
3.5 Summary 51

4 Design of *ReKonf*: A Reconfigurable Multicore Architecture 53
4.1 Parameters for Reconfiguration 53
4.2 Performance Counters 54
4.3 *ReKonf* Tiled Architecture 58
4.3.1 Configuration Controller 60
4.3.2 Switchable Bus 62
4.3.3 Fusible Caches 62
4.4 Summary 66

5 Cache Reconfiguration and Clustering 67
5.1 Reconfiguration in *ReKonf* 67
5.1.1 Static Reconfiguration Technique 70
5.1.2 Dynamic Reconfiguration Technique 73

5.1.3 Adaptive Reconfiguration Technique 76
5.2 Evaluation of *ReKonf* 79
5.2.1 Experimentation Methodology 79
5.2.2 Workloads 82
5.3 Experimental Results and Analysis 84
5.3.1 Static Reconfiguration Technique 84
5.3.2 Dynamic Reconfiguration Technique 87
5.3.3 Adaptive Reconfiguration Technique 90
5.3.4 Benefits and Overheads of Reconfiguration 91
5.4 Summary 93

6 Dynamic Core Allocation **95**
6.1 Motivations 95
6.2 Core Allocation for H.264 Decoder 97
6.2.1 Default Core Allocation 97
6.2.2 Dynamic Core Allocation 99
6.3 Performance Evaluation 102
6.3.1 Characteristics of the Video Traces 102
6.3.2 Experimental Setup 103
6.4 Results and Analysis 106
6.4.1 Energy Savings 106
6.4.2 Varying Frame Rates 108
6.4.3 Factors Influencing Frame Decoding Time 110
6.4.4 Comparison of Energy Savings 113
6.5 Summary 114

7 Core Allocation and Cache Reconfiguration **115**
7.1 Multicore Resource Adaptation at Frame Level 115
7.2 Video Decoding on *ReKonf* 117

7.3 Reconfiguration Overhead 119
7.4 Summary 119

8 Conclusions and Future Work **121**

References **127**

List of Figures

1.1 Block diagram of multicore systems. 3

1.2 An application has natural phases and different architecture configurations benefit across different phases. 7

1.3 Approaches for accommodating software diversity and our contributions. 11

3.1 Performance of FFT, LU and SpMxV on 3 architectures. 34

3.2 SESC tool set outline. 36

3.3 Manycore platforms simulated on SESC. 38

3.4 Impact of L2 cache size in CMP mode. 45

3.5 Speedup with increasing cores in CMP mode. 46

3.6 Speedup with increasing cores in SMP mode. 47

3.7 Comparison of select applications in CMP vs SMP. 49

3.8 Impact of cluster size. 50

4.1 Bit vector for determining cache utilization. The bit vector shown above has 1 active cacheline. 56

4.2 Estimation of cache sharing using bit vectors. 56

4.3 Bit Vector for estimating cache utilization and cache sharing. The length of bit vector is 4096, represented by a horizontal bar. The red and green colors show the occupied and vacant cache lines respectively. The total L2 cache accesses for each core is shown in parentheses. We note that merging the caches for these two applications is beneficial as better overall cache utilization is achieved. 58
4.4 A reconfigurable *ReKonf* tile. 59
4.5 *ReKonf* morphing into various configurations. 59
4.6 Configuration controller. 60
4.7 Fusible caches. 63
4.8 Associated bit vectors are not merged while fusing caches. 64
4.9 Fusion and decomposition of caches. 64

5.1 Static reconfiguration technique. 70
5.2 Dynamic reconfiguration technique. 74
5.3 Phase detection. Each point on the graph is an average over 10000 cycles. The graphs plot the average IPC over time. 76
5.4 Adaptive reconfiguration technique. 77
5.5 Phase-based adaptive reconfiguration on *ReKonf*. 78
5.6 Figure of Merit (FoM) of workloads obtained after sampling. The configuration with maximum FoM is selected for executing the application. 84
5.7 Performance of *ReKonf* versus other configurations. The configuration selected by *ReKonf* is shown with an asterisk on the corresponding workload. 86
5.8 Performance of static reconfiguration technique on *ReKonf* using mixed workloads. 87
5.9 Performance of *ReKonf*:DRT and *ReKonf*:ART compared with static configurations on multi-threaded applications. 88

5.10 Performance of *ReKonf*:DRT and *ReKonf*:ART compared with static configurations on mixed workloads. 89

6.1 H.264 decoder. 97
6.2 Data domain decomposition in H.264. 98
6.3 Opportunity to compensate for overshoot with slack amongst the frames. 100
6.4 Thread mapping on allocated cores. 100
6.5 Core allocation at frame level. 101
6.6 Frame decoding time with QoS requirement of 75 fps. 107
6.7 Energy consumption at a frame rate of 75 fps. 108
6.8 Frame decoding at different frame rates. 109
6.9 Energy consumption at different frame rates. 109
6.10 Relationship between frame size and decoding time. 111
6.11 Per-frame decoding time of *aerobatics* with different core count. . . 112
6.12 Energy delay product of *aerobatics* with different core count. 112
6.13 Snapshot of thread concurrency viewpoint obtained from Intel VTune Amplifier showing synchronization delay while video decoding of *aerobatics*. 113

7.1 Dynamic core allocation and cache reconfiguration in *ReKonf*. . . . 116
7.2 Frame decoding with core and cache reconfiguration. 118
7.3 Energy consumption with core and cache reconfiguration. 118

8.1 Approaches for accommodating software diversity and our contributions. 121

List of Tables

1.1 Recent multicore architectures. 5

2.1 Comparison of core reconfiguration approaches. 22

2.2 Comparison of performance improvements from different core reconfiguration approaches. 22

2.3 Comparison of cache reconfiguration techniques. 27

2.4 Adaptation via dynamic scheduling. 29

2.5 Adaptation via dynamic power management. 31

3.1 Architectural parameters of processors. 39

3.2 Architectural parameters of memory systems. 40

3.3 L2 cache hit and miss delay. 41

3.4 Benchmarks used for the experiments. 42

3.5 Unsupported pthread API in SESC. 43

5.1 Correlation between overall IPC and sample IPC. 69

5.2 Calculation of FoM using performance counters. The core performance is represented by the IPC. The IPC shown is the average IPC of all cores utilized by the application. The IF Value is number of cache lines shared between the cores. The cache sharing value represents the percentage cache lines shared amongst the cores. The Figure of Merit (FoM) is calculated using the static reconfiguration algorithm. 71

5.3 Architectural parameters of baseline configuration. 80
5.4 L2 cache hit and miss delay. 81
5.5 Different configurations used in *ReKonf* evaluation. 82
5.6 Multiprogrammed workloads. 83
5.7 Configuration selected by *ReKonf* based on FoM and the associated performance improvement. 86

6.1 Test videos used for measurements and simulations. 102
6.2 Architectural parameters 105
6.3 Core count selected by dynamic core allocation method with varied QoS for *avalanche* video. 107

7.1 Cache configuration for different core counts. 116

List of Abbreviations

ARM	advanced RISC machines
ASIC	application specific integrated chip
CISC	complex instruction set computer
CLMP	clustered multiprocessor
CMOS	complementary metal oxide semiconductor
CMP	chip multiprocessor
CU	control unit
DL1	data level 1 cache
DPM	dynamic power management
DSP	digital signal processing
FPGA	field programmable gate array
GPU	graphics processing unit
I-Frame	intra-frame
IL1	instruction level 1 cache
IPC	instructions per cycle
ISA	instruction set architecture
ITRS	international technology roadmap for semiconductors
L2	level 2 cache
L3	level 3 cache
McPAT	multicore power area timing
MIMD	multiple instruction multiple data
MIPS	microprocessor without interlocked pipeline stages
NoC	network on chip
OOO	out-of-order
P-Frame	predictive-frame
PARSEC	princeton application repository for shared-memory computers

PPU	power processor unit
QoS	quality of service
ROI	region of interest
SESC	super escalar
SIMD	single instruction multiple data
SMM	shared memory module
SMP	symmetric multiprocessor
SMT	simultaneous multithreading
SOI	silicon on insulator
SPLASH2	stanford parallel applications for shared memory
SPU	synergistic processing unit
TLP	thread level parallelism
TRIPS	tera-op, reliable, intelligently adaptive processing system
VLSI	very large scale integration