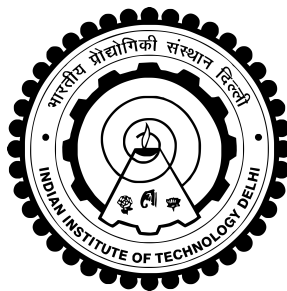


APPLICATION MAPPING ONTO RECONFIGURABLE COARSE-GRAINED ARRAYS

MANSUREH S. MOGHADDAM



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI
AUGUST 2015

APPLICATION MAPPING ONTO RECONFIGURABLE COARSE-GRAINED ARRAYS

by

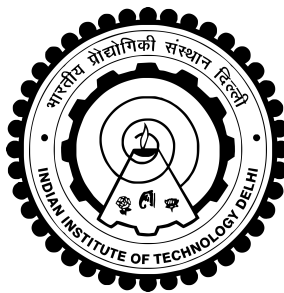
MANSUREH S. MOGHADDAM

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of
Doctor of Philosophy

to the



Indian Institute of Technology Delhi
AUGUST 2015

To My Parents

Certificate

This is to certify that the thesis titled “**APPLICATION MAPPING ONTO RECONFIGURABLE COARSE-GRAINED ARRAYS**” being submitted by Ms. **Mansureh S. Moghaddam** to the **Indian Institute of Technology Delhi**, for the award of the degree of **Doctor of Philosophy**, is a record of bona-fide research work carried out by her under our supervision. In our opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted to any other university or institute for the award of any degree or diploma.

M. Balakrishnan
Professor
Department of Computer Science
and Engineering
Indian Institute of Technology Delhi
New Delhi, India 110 016

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

Date:

New Delhi

Acknowledgements

I acknowledge my research supervisors Prof. M. Balakrishnan and Dr. Kolin Paul for their support throughout the course of the thesis. They introduced me to the art of technical writing and helped me improve my writing abilities with utmost patience. I never forget them sitting next to me and correcting portions of the paper in the process of teaching me. I specially thank them for the freedom and encouragement they have given me from the very beginning of my work that shaped my research. I should specially thank prof. M. Balakrishnan, whose insightful comments and questions helped me to think critically and improve my work, without his help it would have been impossible to debug my ILP-based mapping solution.

My sincere thanks to Dr. Marcus M. Edvall with whom I had many fruitful discussions on using TOMLAB/CPLEX to optimize large scale (M)ILP problems. I also thank Dr Todor Stefanov and Mohamed A. Bamakhrama (www.liad.nl) for their help and suggestions on using DAEDALUS framework.

Many thanks to my colleagues, Surender Kumar Verma, Madhulika Madhur and Mehdi Niktabar who were always ready to help and more than eager for any discussion.

I am grateful to my family for their unconditional love and support. My parents always encouraged me to do whatever interests me and without their support, I would not have been the same.

AUGUST 2015

Mansureh S. Moghaddam

Abstract

Coarse-Grained Reconfigurable Architectures (CGRAs) have become popular in recent times as the increased transistor densities have enabled greater integration of increasingly complex “compute cores”. These devices pack massive compute power and can be effectively used to build efficient solutions for applications which have a significant degree of parallelism. In many cases, these CGRAs are also partially reconfigurable. Clearly to make effective use of these highly “parallel compute platforms”, a good mapping flow is required to map the parallelism that is present in a target application. We propose an **optimal** mapping flow which also exploits the partial reconfiguration property of modern CGRAs. Hence this algorithm is not only suitable for applications which can be accommodated in the available silicon but also for larger applications (or a set of applications) which need more area than that provided by the CGRA platform. In this work partitions of the application are mapped partially and dynamically (during the life time of the application) onto the platform. While in most mapping flows, “*scheduling*”, “*binding*” and “*place & route*” steps are carried out in sequence, our proposed mapping flow integrates all three steps in one “*mapping*” step using an Integer Linear Problem formulation. The algorithm is implemented using TOMLAB/CPLEX toolbox and we assess its efficacy on a set of 40 synthetic task graphs and some real life multimedia applications.

In a multi-processing fabric like CGRA, tiles work in parallel and in a pipelined manner. Therefore processing the application task graph to redistribute the tasks to the nodes can be taken as a preprocessing step to the mapping problem. We deal with this problem in the thesis by proposing algorithms for rebalancing. We implement the proposed mapping flow on a Xilinx XUPV5 FPGA board. This platform is used to implement a 2D mesh of processing elements (tiles) alongwith a main controller. The main controller provides new content to the tiles. The context switch between snapshots is also controlled by the controller using two levels of handshaking. Further, to validate our approach, we integrate it to a mapping flow named Daedalus.

Table of Contents

Certificate	i
Acknowledgements	iii
Abstract	v
List of Contents	vii
List of Figures	xi
List of Tables	xiii
List of Abbreviations	xv
1 Introduction	1
1.1 Motivation	2
1.2 Mapping Approach	4
1.3 Outline of the Chapters	7
2 Literature Survey	9
2.1 CGRAs	9
2.1.1 CGRA Mapping	11
2.2 MPSoCs	12
2.2.1 Optimal Static Mapping	13
2.2.2 Heuristics-Based Static Mapping	21
2.2.3 Heuristics-Based Dynamic Mapping	28
2.3 Automated Mapping of Application to MPSoC	35
2.3.1 MAPS	35
2.3.2 MNEMEE	35
2.3.3 DAEDALUS	36
2.3.4 PTOLEMY II	37
2.4 Conclusion	38
3 Pre-Mapping Steps: Reconfiguration Cost and Rebalancing	41

3.1	A Coarse-Grained Reconfigurable Fabric	42
3.2	Modelling Applications	44
3.3	Radix-2 FFT Case Study	46
3.3.1	Radix-2 FFT Decomposition	47
3.3.2	Radix-2 FFT Performance Evaluation	49
3.4	JPEG Encoder Case Study	55
3.4.1	JPEG Encoder Manual Mapping	56
3.4.2	Balanced Task Assignment	59
3.5	Conclusion	65
4	Optimal Mapping of Tasks onto Tiles	67
4.1	Problem Definition	69
4.2	Mapping	73
4.3	Minimizing Total Reconfiguration Time	75
4.3.1	Constraints	76
4.3.2	Objective Function	78
4.4	Minimizing Total Execution Time	79
4.5	Backward Edge Coverage	83
4.6	MILP Analysis	85
4.7	MILP Solver Targeting Optimal Solution	86
4.8	3D Mapping	88
4.9	Extending to Other Interconnect Topologies	93
4.9.1	Arbitrary Interconnects	93
4.9.2	Heterogeneous Platform	97
4.10	Energy Consumption and Thermal Issues	99
4.11	Conclusion	100
5	Platform and Design Methodology	101
5.1	Platform Implementation	102
5.2	Reconfiguration Support	103
5.3	Software Environment	105
5.3.1	Application Description	105
5.3.2	Applying Mapping Approach	109
5.3.3	Context Switching	109
5.3.4	Loader/Cntlr Synchronization	112
5.4	Daedalus Framework	115
5.5	Integration into Daedalus framework	118
5.5.1	Manual Mapping through Daedalus	118
5.5.2	Modifications to Daedalus	120
5.6	Conclusion	122
6	Experimental Results	123
6.1	Generic Task Graphs	123
6.1.1	Experimental Setup	124
6.1.2	Experiments for 2D Mesh Platform	124

6.1.2.1	Total Execution Time	125
6.1.2.2	MILP Solver Memory Requirements	126
6.1.2.3	(M)ILP Solver Runtime	127
6.1.2.4	Experiments for 3D Mesh Platform	130
6.2	Test Cases	133
6.2.1	Platform Implementation	135
6.2.2	Task Graph Extraction	137
6.2.3	Computation Cost	138
6.2.4	Reconfiguration Cost	142
6.2.5	2×2 and 3×3 Platforms	147
6.3	Conclusion	150
7	Summary of Contributions and Future Work	153
7.1	Summary of contributions	153
7.2	Future Work	155

List of Figures

1.1	Task graph mapping onto a 3×3 platform	3
1.2	Mapping onto 3×3 platform for two different objectives	5
1.3	Overall flow of the thesis	6
2.1	Mapping from application to MPSoC in MAPS [1]	36
2.2	MNEMEE tool flow [2]	37
2.3	Mapping from application to MPSoC in Daedalus [3]	38
3.1	reMORPH array [4]	43
3.2	Reconfiguration system diagram	44
3.3	JPEG encoder sample partitioning	46
3.4	Radix-2 FFT processes	47
3.5	16-point R2FFT partitions	47
3.6	Different mappings for 16-Point Radix2 FFT	48
3.7	Butterfly for 16-point R2FFT	51
3.8	Vertical link connection	53
3.9	Execution time for 1024-point Radix2 FFT	53
3.10	Link cost impact on Radix2 FFT implementation	54
3.11	Task graph of JPEG encoder	55
3.12	More than One tile for a heavy task	58
3.13	reBalancing using Algorithm 1	59
3.14	Allocating more tiles to case “e” of Fig. 3.13	62
3.15	Average # of images processed in JPEG encoder application	64
3.16	Average PE utilization for JPEG encoder	65
4.1	A sample platform, task graph and the mapping	68
4.2	Task pipelining in CGRA	69
4.3	Mapping onto 3×3 platform for two different objectives	70
4.4	illustration of mapping of interconnections between tasks	71
4.5	Impact of V on critical path and overall performance	81
4.6	Task and edge path matrices (P_t , P_e)	81
4.7	Circular view of snapshots for backward edges of a task graph	83
4.8	Optimal mapping using two snapshots	87
4.9	Optimal mapping using three and four snapshots	87
4.10	MILP solver targeting optimal mapping	88
4.11	3D mesh platform	89

4.12	Arbitrary interconnection platform	94
4.13	Mapping onto 3×3 platform with heterogeneous tiles	99
5.1	Mapping task graph to the platform, and context switch between snapshots	101
5.2	Partial dynamic platform on XUPV5	102
5.3	PCM and tiles interaction using BUS or FIFO	104
5.4	Application C code	107
5.5	Application in BRAM	107
5.6	Generic format for task and edge	109
5.7	Context switch layout	110
5.8	Status and control registers	111
5.9	“DPR” “Loader” and “Cntlr” flowcharts	113
5.10	Synchronization between “PCM” and tiles	114
5.11	Daedalus framework-input & output	115
5.12	Generic format for tasks and edges	116
5.13	Platform description example in Daedalus	117
5.14	Platform description example in Daedalus	118
6.2	MIP solver memory requirements for 2D platform	127
6.4	Number of tiles for 2D platform	128
6.5	Generic task graphs G_{27} and G_{28} with different complexity	129
6.7	MILP solver memory usage for 3D platform	132
6.9	Runtime vs. number of tiles for 3D platform	133
6.10	Mapping tasks and edges of a task graph onto the 3×3 platform	134
6.11	Task graph for the three test cases	138
6.12	Huffman tree associated with the input stream file: h4.bin	140
6.13	Generate linker script in EDK	143
6.14	Test cases implementation on a 3×3 platform	147
6.15	Snapshots for 2×2 implementation of Sobel low-pass filter	147
6.16	Snapshots for 2×2 implementation of Huffman coding	147
6.17	Snapshots for 2×2 implementation of JPEG2000 decoder	148
6.18	Mapping JPEG encoder to a reMORPH of size 3×3	149
6.19	JPEG2000 decoder and H.264 decoder implementation on 3×3 grid	150
6.20	Application core graphs	150

List of Tables

2.1	Optimal static mapping of tasks onto multiprocessor platforms	17
2.2	Heuristic-based static mapping of tasks onto multiprocessor platforms . . .	25
2.3	Heuristic-based dynamic mapping of tasks onto multiprocessor platforms . .	32
3.1	1024-point Radix2 FFT computation	50
3.2	Optimized copy loop cost (ns)	52
3.3	JPEG encoder tasks-computation and reconfiguration cost	56
3.4	JPEG encoder: manual mapping	57
3.5	Binding processes to the tiles using Algorithm 1	63
3.6	Binding processes to the tiles using Algorithm 2	64
4.1	Computation and reconfiguration cost for the graph of Fig. 4.8(a)	87
4.2	Computation, reconfiguration and total execution time for graph in Fig. 4.8(a)	88
4.3	Computation and reconfiguration cost for the graph of Fig. 4.8(a)	98
6.1	Computation and reconfiguration cost for generic task graphs	124
6.2	Comparing both objective functions for the same set of graphs onto 2D mesh	125
6.3	Specifications of task graphs G_{27} and G_{28} (Fig. 6.5)	130
6.4	Comparing both objective functions for the same set of graphs onto 3D mesh	131
6.5	Experimental platform resource usage on XUP-V5 FPGA board	135
6.6	Computation and reconfiguration cost for generic task graphs	136
6.7	Sobel low-pass filter (pixel level) tasks/edges runtime (cycles)	138
6.8	Sobel low-pass filter (block level) tasks/edge runtime (cycles)	139
6.9	Input stream degree of complexity for Huffman coding	139
6.10	Tasks “main” part runtime in Huffman coding associated with A to G streams	140
6.11	Tasks (“get” loop) latency in Huffman coding associated with A to G streams	140
6.12	Tasks (“send” loop) latency in Huffman coding associated with A to G streams	140
6.13	Huffman coding tasks/edges average runtime (cycles)	141
6.14	JPEG2000 decoder tasks/edges runtime (cycles)	141
6.15	Sobel low-pass filter (pixel level) tasks/edges size (Fig. 6.11(a))	144
6.16	Sobel low-pass filter (block level) tasks/edges size (Fig. 6.11(a))	145
6.17	Huffman coding tasks/edges size (Fig. 6.11(b))	145
6.18	JPEG2000 decoder tasks/edges size (Fig. 6.19(a))	145
6.19	Sobel low-pass filter (pixel level) tasks/edges reconfiguration cost (cycles) .	146
6.20	Sobel low-pass filter (block level) tasks/edges reconfiguration cost (#cycles)	146
6.21	Huffman coding tasks/edges reconfiguration cost (#cycles)	146

6.22	JPEG2000 decoder tasks/edges reconfiguration cost (#cycles)	146
6.23	Test cases mapped to 2D mesh of MicroBlazes in XUPV5	148
6.24	JPEG encoder associated computation/reconfiguration cost	149
6.25	Mapping VOPD to 2D mesh platform	151
6.26	Mapping MWD to 2D mesh platform	151
6.27	Mapping mp3enc/mp3dec to 2D mesh platform	151
6.28	Mapping 263dec/mp3dec to 2D mesh platform	151
6.29	Mapping 263enc/mp3dec to 2D mesh platform	151
6.30	Mapping PIP to 2D mesh platform	152

List of Abbreviations

BN	B est N eighbour
BSB	B ase S ystem B uilder
BSP	B oard S upport P ackage
CA	C luster A gent
CCG	C ombined C ore G raph
CDCM	C ommunication D ependence and C omputation M odel
CDM	C ommunication D ependence M odel
CF	C ompact F lash
CGRA	C oarse G rained R econfigurable A rray
CGRM	C oarse G rained R econfigurable M odule
CSO	C ontext S witch O verhead
CTG	C ommunication T ask G raph
CTM	C ore T ile M apping
DAG	D irected A cyclic G raph
DCT	D iscrete C osine T ransform
DFG	D ata F low G raph
DMA	D irect M emory A ccess
DVS	D ynamic V oltage S caling
EDK	E mbedded D evelopment K it
ESL	E lectronic S ystem L evel
FF	F irst F ree
FFT	F ast F ourier T ransform
FIFO	F irst I n F irst O ut

FPGA	F ield P rogrammable G ate A rray
FSL	F ast S erial L ink
FU	F unctional U nit
GA	G enetic A lgorithm
GA	G lobal A gent
ICAP	I nternal C onfiguration A ccess P ort
IP	I ntellectual P roperty
ILP	I nteger L inear P roblem
KPN	K ahn P rocess N etwork
LACG	L ogical A pplication C ommunication G raph
LED-DN	L ow E nergy C onsumption D ependences N eighbourhood
LS	L ink S hutdown
MAC	M inimum A verage C hannel
mCSM	m any C ore S witch M apping
MILP	M ixed I nteger L inear P roblem
MIMD	M ultiple I nstruction M ultiple D ata
MIQP	M ixed I nteger Q uadratic P rogramming
MMC	M inimum M aximum C hannel
MPSoC	M ulti P rocessor S ystem O n C hip
MT-ADRES	M ulti T hreaded A DRES
NCR	N ear C onvex R egion
NN	N earest N eighbour
NoC	N etwork O n C hip
NRE	N on R ecurring E ngineering
PCM	P artial C onfiguration M anager
PDH	P ath-based D eterministic H euristic
PE	P rocessing E lement
PL	P ath L oad
PLB	P rocessor L ocal B us
PPN	P olyhedral P rocess N etwork
PRH	P ath-based R andomization H euristic

PSO	P article S warm O ptimization
QEA	Q uantum-inspired E valuation A lgorithm
QoS	Q uality O f S ervice
QP	Q uadratic P rogramming
RTL	R egister T ransfer L anguage
SANLP	S tatic A ffine N ested L oop P rogram
SDF	S ynchronous D ata F low
SDK	S oftware D evelopment K it
SNN	S mart N earest N eighbour
ST-TA	S imulated A nneling with T iming A adjustment
TGFF	T ask G raph F or F ree
TMTSC	T opology M apping and T raffic S urface C reating
TSM	T ile S witch M apping
UART	U niversal A synchronous R eceiver T ransmitter
WCTG	W eighted C ommunication T ask G raph
XMD	X ilinx M icroprocessor D ebg
XPS	X ilinx P latform S tudio