

# Partitioning Methods and Algorithms for Configurable Computing Machines

Suresh Chandrasekhar

(ABSTRACT)

This thesis addresses the partitioning problem for configurable computing machines. Specifically, this thesis presents algorithms to partition chain-structured task graphs across configurable computing machines. The algorithms give optimal solutions for throughput and total execution time for these problems under constraints on area, pin count, and power consumption. The algorithms provide flexibility for applying these constraints while remaining polynomial in complexity. Proofs of correctness as well as an analysis of runtime complexity are given. Experiments are performed to illustrate the runtime of these algorithms.

## **ACKNOWLEDGEMENTS**

I would like to thank my advisor, Dr. Mark T. Jones, for giving me the opportunity to work on this project, and for his constant support throughout the project. I would also like to thank Dr. James R. Armstrong and Dr. Dong S. Ha for serving on my committee.

I would like to thank my parents for their constant encouragement over the years, and for giving me the extra push whenever I needed it. I would also like to thank my brother for his enthusiasm and support. Last but not least, I would like to thank all my friends, without whom life would be much less enjoyable.

## TABLE OF CONTENTS

<b>1 Introduction</b>	<b>1</b>
1.1 Motivation	1
1.2 The problem	3
1.3 The approach	4
1.4 Thesis organization	4
<b>2 Background</b>	<b>6</b>
2.1 Definitions, terms, and model used in the thesis	6
2.2 Past and present partitioning methods used for FPGAs	14
2.3 Comparison of distributed and configurable computing systems	16
2.4 Related work in distributed computing systems	18
2.5 Bokhari's method for solving the linear module and processor case	19
2.5.1 Illustration of Bokhari's algorithm	20
2.5.2 Basis for using Bokhari's method for CCMs	25
<b>3. Generalization of Bokhari's Algorithm for the case of CCMs</b>	<b>28</b>
3.1 Partitioning chain structured modules onto a chain of PEs	28
3.1.1 The problem	28
3.1.2 Assumptions	29
3.2 The solution	29
3.2.1 Layered assignment graph $G_a$	29
3.2.2 Minimum bottleneck assignment	30
3.2.3 Assignment of weights to the nodes of graph G	31
3.3 Proof of correctness for Algorithm 3.1	33
3.4 Improvement in the speed of execution	34

## **4. Solution to Partitioning Series Module Graphs onto Arbitrary PE Graphs 37**

4.1 Mapping chain structured modules onto a general PE graph	37
4.1.1 The problem	38
4.1.2 Assumptions	38
4.2 The solution	38
4.2.1 Construction of layered graph for each of the $G_{px}$	41
4.2.2 Assignments of weights to the nodes of the graph $G_a$	42
4.3 Proof of correctness of the algorithm	45
4.4 Complexity	47

## **5. Module Graph having a Fork onto a Chain of PEs 48**

5.1 Partitioning a module graph having a fork onto a chain of PEs	48
5.1.1 The problem	49
5.1.2 Assumptions	49
5.2 The solution	50
5.2.1 Construction of the assignment graph	51
5.2.2 Construction of nodes in the assignment graph $G_{al}$	52
5.2.3 Assignment of edges for the assignment graph $G_{al}$	53
5.2.4 Application of constraints	53
5.2.5 Assignment of weights	54
5.3 Proof of correctness of Algorithm 5.1	55

## **6. Results 57**

6.1 Implementation of Algorithms 3.1, 4.1, 4.2, and 5.1	57
6.1.1 Results for Algorithm 3.1 implementation	57
6.1.2 Results for Algorithm 4.1 and 4.2 implementation	62
6.1.3 Results for the implementation of Algorithm 5.1	67

<b>7. Conclusions and Suggestions</b>	<b>68</b>
7.1 Conclusions on the thesis	68
7.2 Suggestions for future work	69
<b>References</b>	<b>70</b>
<b>Vita</b>	<b>75</b>

## LIST OF FIGURES

1.1 General structure of a contemporary CCM.	2
2.1 A module graph.	7
2.2 Four PEs connected by links.	8
2.3 A module graph with five modules.	21
2.4 Nodes of the assignment graph.	21
2.5 Edge creation for the assignment graph.	22
2.6 Weights on the nodes of assignment graph.	23
2.7 Highlighted critical path representing minimum bottleneck assignment.	24
2.8 A nine-module chain mapped to a four-processor chain.	25
3.1 Node creation for graph $G_a$ .	31
3.2 Edge creation for graph $G_a$ .	32
3.3 Algorithm 3.1 for finding minimum bottleneck path or minimum total execution time from the assignment graph $G_a$ .	32
3.4 A layered graph for a linear array problem with nine modules and four processors.	34

3.5 An improved layered graph for a linear array problem with nine modules and four processors.	35
4.1 A chain module graph.	37
4.2 A PE graph shown having nine PEs and their links.	38
4.3 Example of a trail.	39
4.4 Partitioning problem A.	40
4.5 Partitioning problem B.	40
4.6 Algorithm 4.1.	44
4.7 Algorithm 4.2.	44
5.1 A PE graph with five PEs connected as a chain.	48
5.2 A module graph having a single fork.	49
5.3 Module graph showing the set of modules $V_{a1}$ , $V_{a2}$ , and $V_M$ .	51
5.4 Module graph $G_{m1}$ showing the set of modules $V_{C1}$ and $V_{a2}$ .	51
5.5 Algorithm 5.1 for $G_m$ .	54
6.1 Plot of bottleneck values versus the limitation of modules on a PE.	60
6.2 Comparison of two plots representing the runtimes of two different implementations of Algorithm 3.1.	61

6.3 Directed arrows showing a trail in a PE graph.	62
6.4 A PE graph with four PEs and four links.	65



## LIST OF TABLES

6.1 Table containing the results for Algorithm 3.1 implementation.	58
6.2 Results for Algorithm 3.1 under increasing constraints.	59
6.3 Results of Algorithm 4.1 for the case of PE inter-connection being a ring of 10 PEs.	63
6.4 Results for the two dummy PE case for the trail shown in Figure 6.3.	63
6.5 Results showing reduction in graph size and execution time of Algorithm 4.1 under increasing constraints.	64
6.6 Results for Algorithm 4.2.	65
6.7 Weights of the modules.	66
6.8 Runtimes for implementation of Algorithm 4.1 for CCM of Figure 1.1.	67
6.9 Results for the implementation of Algorithm 5.1.	67