DATAFLOW ACTOR NETWORK PARTITIONING FOR MULTIPLE FPGAS

CHIN YONG HUAN

A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering (Computer and Microelectronic Systems)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

JUNE 2016

To my beloved mother and father

ACKNOWLEDGEMENT

I would like to thank all who in one way or another contributed in the completion of this thesis. First and foremost, I would like to acknowledge and deliver a highest appreciation to my project supervisor, Dr. Ab Al Hadi Ab Rahman, for his valuable direction and advice rendered. Sincere thanks too for his contribution, guidance, care, patience and effort in guiding and inspiring me throughout the project. His suggestions had often inspired me in conducting this project.

Secondly, I would like to thank Intel Technology Sdn Bhd for providing supports and also adequate facilities which allow me to complete this project in time. Also, thanks to all of my colleagues who are supportive, understanding and often provide encouragement to me.

Finally, a humble and honorable thanks goes to my family for their understandings and supports to me in completing this report. Besides that, I would like to thank my friends for helping me when I encountered dificulties by sharing their useful opinions and suggestions.

ABSTRACT

Dataflow actor network is used to display the relation between different actors in a directed graph. It is suitable for modelling signal and video processing in software applications. In this paper, the use of dataflow actor network is extended to the hardware implementation of streaming applications via dataflow actor network partitioning for multiple FPGAs based on the number of cuts, connection workload, resource utilization ratio and latency. Multiple FPGAs partitioning is often required for implementing design with large logic count, for cost reduction, multi clock and multi power domains design implementation. The motivation of using the dataflow actor network is due to the nature of the network which closely resembles the structural view and the inter-connections of a design at the architecture level. This representation in the form of a dataflow actor network is suitable for implementing graph partitioning algorithms. The KL algorithm, GA, PSO, SA and WOA are used for single objective partitioning while the MOPSO, MOSA and MOWOA have been used for multi objective partitioning. The objective of this study is to develop partitioning algorithm suitable for use in dataflow actor network and to determine the appropriate partitioning criteria. Results showed that SA has better performance as compared to other partitioning algorithm for single objective partitioning. On the other hand, for multi objective partitioning the MOPSO has better performance for small design while MOSA has better performance for larger design.

ABSTRAK

Rangkaian aliran data digunakan untuk menunjukkan hubungan antara aktor yang berbeza dalam graf berarah. Ia sesuai untuk pemodelan isyarat dan pemprosesan video dalam aplikasi perisian. Dalam thesis ini, rangkaian aliran data aktor akan digunakan dalam pembahagian rangkaian aliran data pelakon untuk perkakasan aplikasi pemprosesan video melalui beberapa FPGAs berdasarkan bilangan pemotongan antara partition yang berbeza, beban komunikasi sambungan, nisbah penggunaan sumber dan kependaman. Pembahagian reka bentuk melalui beberapa FPGAs sering diperlukan untuk melaksanakan reka bentuk yang mempunyai kiraan logik besar untuk tujuan mengurangkan kos dan untuk pelaksanaan reka bentuk yang memerlukan pelbagai kelajuan jam dan domain kuasa. Antara motivasi menggunakan rangkaian aliran data pelakon adalah kerana sifat rangkaian yang hampir menyerupai pandangan struktur dan saling sambungan reka bentuk pada peringkat seni bina. Perwakilan ini dalam bentuk rangkaian aliran data pelakon sesuai untuk melaksanakan algoritma pembahagian graf. Algoritma KL , GA, PSO , SA dan WOA digunakan untuk pembahagian objektif tunggal manakala MOPSO, MOSA dan MOWOA telah digunakan untuk pembahagian pelbagai objektif. Objektif kajian ini adalah untuk membuat algoritma pembahagian sesuai untuk digunakan dalam rangkaian aliran data pelakon dan untuk menentukan kriteria pembahagian yang sesuai. Eksperimen menunjukkan SA mempunyai prestasi yang lebih baik berbanding dengan algoritma pembahagian lain untuk pembahagian objektif tunggal. Sebaliknya, untuk pembahagian pelbagai objektif, MOPSO mempunyai prestasi yang lebih baik untuk reka bentuk kecil manakala MOSA mempunyai prestasi yang lebih baik untuk reka bentuk yang lebih besar.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACO	-	Ant Colony Optimization
ALAP	-	As-Late-As-Possible
ASAP	-	As-Soon-As-Possible
BPSO	-	Binary Particle Swarm Optimization
CAL	-	CAL Actor Language
CSDF	-	Cyclo-Static Dataflow
DFG	-	Data Flow Graph
DPN	-	Data Process Network
DVD	-	Digital Versatile Disk
EDA	-	Electronic Design Aided
FIFO	-	First In First Out
FIR	-	Finite Impulse Response
FPGA	-	Field Programmable Gate Array
FSM	-	Finite State Machine
GA	-	Genetic Algorithm
HDL	-	Hardware Description Language
HEM	-	Hardware Program Execution Buffer Size Minimum
HEO	-	Hardware Program Execution Buffer Size Optimization
HEVC	-	High Efficiency Video Coding
IDE	-	Integrated Development Environment
ITRS	-	International Technology Roadmap of Semiconductor
KL	-	Kernighan Lin
KPN	-	Kahn Process Network
MOPSO	-	Multi Objective Particle Swarm Optimization
MOSA	-	Multi Objective Simulated Annealing
MOWOA	-	Multi Objective Whale Optimization Algorithm
MPEG	-	Moving Picture Experts Group
ORCC	-	Open RVC-CAL Compiler
PSDF	-	Parameterized Synchronous Dataflow
PSO	-	Particle Swarm Optimization

RVC-CAL	-	Reconfigurable Video Coding
SA	-	Simulated Annealing
SCR	-	Super Computer Research
TEM	-	Trace Execution Buffer Size Minimization
TEO	-	Trace Execution Buffer Size Optimization
VLSI	-	Very Large Scale Integration
SDF	-	Synchronous Dataflow
WOA	-	Whale Optimization Algorithm
WNS	-	Worst Negative Slack
XDF	-	Extended Markup Language Directed File

CHAPTER 1

INTRODUCTION

1.1 Project Background

The role of semiconductors have been growing at a tremendous rate in many fields and had even extended their applications in fields previously known as unrelated to semiconductors such as in fashion industries, automobiles and wearables (spectacle, watch, jewelry etc.). Thus, electronics and integrated circuits with low operating power and small die size are in great demands. Following these aspects, the application specific integrated circuit (ASIC) design and fabrication techniques are constantly being improved by scaling down device length and further push the performance of the device. While this approach is still feasible and complies with Moore's Law, it is often accompanied by the increase in complexity and non-recurring engineering cost. Furthermore, with the increase in competitiveness in semiconductor fields, semiconductors and integrated circuits business units had to put in a lot of effort in achieving the shortest time to market in order to rise above other competitors. Hence, there have been a shift in recent years from ASIC to field programmable gate array.

FPGAs are used in many applications. Some common uses include ASIC design prototyping, data networking, medical, automotive, communication, audio, video and image processing. Although FPGA can be used as a single device, there are also occasions where multiple FPGA devices are more desirable for implementing a system in order to improve performance. One example of such system is the SPLASH system developed by Supercomputing Research Center (SRC). The SPLASH system consists of an array of linearly connected reconfigurable logic made up from 32 Xilinx 3090 FPGAs used to perform compute intensive applications and for testing hardware-based systolic algorithms [1].

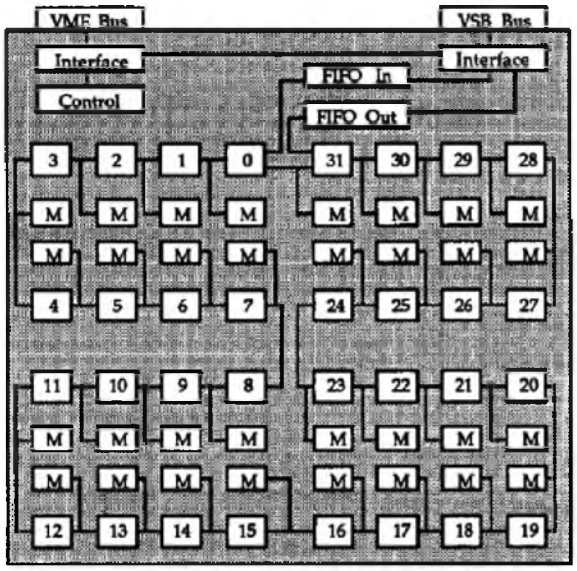

Figure 1.1: Architecture of SPLASH [1]

Besides specialized architectures, there are other instances which require usage of multiple FPGAs. One such instance is design partitioning. Partitioning in general refers to dividing something into smaller independent divisions. However, in terms of FPGAs partitioning, it refers to division of a RTL design into a few individual FPGA devices. In one of the white paper from the International Technology Roadmap of Semiconductor (ITRS) entitled "More than Moore", modern very large scale integrated circuit (VLSI) design does not only demands scaling down of device sizes, but also demands the increase in functionality of the device [3]. This leads to logic design getting larger and more complex. Prototyping is no longer feasible as it is not possible for the design to fit into a single FPGA device. Hence, multiple FPGAs are used to increase the resources of the FPGA device such as CLB, flip flop, static RAM (SRAM) and I/O pin. By partitioning the design into separate different FPGA devices, each partition can be connected to one another by means of external connection through the device pin.

Although there have being new release of FPGA device with large number of logic cells such as the Xilinx UltraSCALE, it is sometimes still preferable to use multiple smaller FPGA devices to reduce the cost. In addition, some designs may require operation in multiple power domains, clock frequency or even different technology node which could only be made possible by using multiple FPGAs devices. Furthermore, multiple FPGAs partitioning can be used to model the partitioning of a design into several different integrated circuit (IC) packages to test for functionality

as the size of IC packages are often fixed and several sub functions in a design might need to be separate into different IC packages. Multiple FPGAs partitioning can be done either manually or by means of an electronic design automation (EDA) tool. Conventional approach favors the manual method over the dependence on EDA tools as designers have more flexibility and control over the distribution of the logic cells to obtain balance partitions. Nevertheless, the improvement of EDA tools and also the increase in the size of a design had cause the shift in favor over EDA tools than manual efforts as manual efforts are cumbersome and hard to achieve timing convergence without the aid of an EDA tool. Commercial tool such as Xilinx PlanAhead had also used automated partitioning to assign logic cells to different CLBs. In addition, designers are also given the flexibility to prioritize on the criteria used for deciding the partitions – either based on resource constrain, timing constrain or balanced partitioning.

It is undeniable that these few criteria are the main criteria for obtaining good quality partitions. However, in specific application designs such as streaming application design requires additional criteria. Other performance metrics that are needed to be taken into account includes critical path delay, latency, throughput and maximum allowable operating frequency. In this thesis, the cut size and the communication rate are considered when performing partitioning. Cut size refers the number of interconnections crossing the boundaries between multiple FPGAs while the communication rate refers to the number of bits transfer over an interconnection. These performance metrics are important as partitioning at interconnections with high communication rate and high number of cut size would introduce unnecessary delay and interference to the data transferred. Furthermore, it will also affect the overall throughput of the system due to the introduction of parasitic delay in the external wires.

In order to partition the design based on cut size and communication rate, a suitable representation of the overall system to be partitioned is necessary. While there are many partitioning techniques which perform partitioning based on RTL netlist of a design, CAL actor language (CAL) is used in this thesis. Using CAL, the design is represented in the form of a dataflow actor network. Each actor represents one of the functional units in the design while the edges of the dataflow networks represent the interconnection between the functional units. Advantage of using the CAL actors is that a design is already being represented in graph form and hence there is no need for an intermediate stage to transform RTL netlist into graph representation. Furthermore, dataflow actor networks are better representation for streaming applications as compared to behavioral or netlist representation.

The organization of this thesis is as follows. Chapter one provides some background information on dataflow actor network partitioning based on cut size and communication rate. The problem statements, objectives, motivation and the scope of study of this thesis are also being discussed for comprehension of readers. Chapter two provides the review of previous related works regarding CAL actors and partitioning techniques. In chapter three, the methodology used for CAL actor partitioning is being shown and described in details. Chapter four shows the preliminary results and the associated discussions. Chapter five concludes and provides some insight on CAL actor partitioning.

1.2 Problem Statement

Partitioning problems have being regarded as classical problems involving graph which consists of nodes and edges. The nodes of the graph is usually partitioned into two different groups, known as bi-partitioning. The bi-partitioning problem is first addressed by Brian Wilson Kernighan and Shen Lin. Both of them had come out with an algorithm known as the Kernighan-Lin (KL) algorithm to solve the traveling saleman problem which later has being adapted for graph partitioning [4]. Many of the variation of partitioning algorithms are based on the KL algorithm and its application had being used in VLSI circuit partitioning.

Recent papers such as in [5–7], the graph partitioning problems had also being used to perform hardware software partitioning of dataflow graph network. The sub blocks partitioned into hardware acts as an accelerator to improve its speed performance. However, there are yet to be found the literature on partitioning of dataflow graph network as a fully dedicated hardware design.

Besides that, in most partitioning problem, only the resource constrain and timing constrain is being considered. In addition, the partitioning of the dataflow actor network often done for multicore processor using actor workload as the partitioning criteria which is not suitable for use as a partitioning criteria in FPGA implementation. Lastly, general partitioning is often performed at lower abstraction level hence suitable for smaller design only or require more complicated partitioning algorithm.

1.3 Objectives

There are several objectives to be met in this project. They are:-

1. To develop partitioning algorithm to perform hardware partitioning of the actors in a network of a design into multiple FPGA devices.
2. To investigate suitable partitioning criteria for use of dataflow actor network partitioning.

1.4 Motivation

The dataflow actor network resembles the structural view and the inter-connections of a design at the architecture level suitable for implementing graph partitioning algorithms. In addition, the current partitioning of the dataflow actor network are either fully software implemented in the multicore processor or hardware software co-design implemented in both FPGA and processor. For the partitioning of dataflow actor in multiprocessor, the common partitioning criteria used is the actor workload obtained through software profiling while for hardware software co-design, the partitioning criteria commonly used is the actor execution time and resource area. The partitioning criteria for both cases are not suitable to be used for a fully hardware dedicated FPGA implementation.

1.5 Scope of Study

The study of dataflow actor network covers a very large field of research. However, not every aspect of dataflow actor network is covered in this thesis. The partitioning of actor network demonstrated in this thesis is limited to bi-partitioning problems. Nevertheless, using the same basic principal, the partitioning could be expended to handle k-way partitioning. In terms of the partitioning criteria, this thesis focuses more on the number of cuts, connection workload, resource utilization ratio and actor latency.

Since part of the study done in this thesis involved creating partitioning algorithm for the dataflow actor network, it is inevitable that some of the processes

need to be automated. However, automation would only be done in several aspects such as acquiring the RVC-CAL design, performing characterization and performing partitioning. Other intermediate processes such as converting the RVC-CAL design into its equivalent HDL representation would be done manually.

Lastly, the partitioning methodology and the use of RVC-CAL as design representation introduced in this thesis are for streaming application designs. Other forms of applications had yet to be tested. The test case used for evaluating the partitioning algorithm is the four-tap FIR digital filter, HEVC decoder and MPEG4 decoder.

REFERENCES

1. M. B. Gokhale, *Splash: A reconfigurable linear logic array*. Supercomputing Research Center, 1990.
2. C. Lucarz, P. Faure, G. Roquier, M. Mattavelli, and V. Noël, "Cal methodology,"
3. H. Kopetz, "The time-triggered model of computation," in *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, pp. 168–177, IEEE, 1998.
4. S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.
5. E. Bezati, H. Yviquel, M. Raulet, and M. Mattavelli, "A unified hardware/software co-synthesis solution for signal processing systems," in *Design and Architectures for Signal and Image Processing (DASIP), 2011 Conference on*, pp. 1–6, IEEE, 2011.
6. G. Roquier, E. Bezati, and M. Mattavelli, "Hardware and software synthesis of heterogeneous systems from dataflow programs," *Journal of Electrical and Computer Engineering*, vol. 2012, p. 2, 2012.
7. B. Traskov, "Hardware/software partitioning of dataflow programs: Rapid prototyping of computer systems in the cal actor language," 2011.
8. S. Louise, P. Dubrulle, and T. Goubier, "A model of computation for real-time applications on embedded manycores," in *Embedded Multicore/Manycore SoCs (MCSoc), 2014 IEEE 8th International Symposium on*, pp. 333–340, IEEE, 2014.
9. D. Fontanelli, L. Palopoli, and L. Abeni, "The continuous stream model of computation for real-time control," in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, pp. 150–159, IEEE, 2013.
10. K. Gilles, "The semantics of a simple language for parallel programming," *In Information Processing*, vol. 74, pp. 471–475, 1974.
11. E. A. Lee and T. M. Parks, "Dataflow process networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–801, 1995.

12. F. Rahimian, A. H. Payberah, S. Girdzijauskas, M. Jelasity, and S. Haridi, "Ja-be-ja: A distributed algorithm for balanced graph partitioning," 2013.

13. B. Bhattacharya and S. S. Bhattacharyya, "Parameterized dataflow modeling for dsp systems," *Signal Processing, IEEE Transactions on*, vol. 49, no. 10, pp. 2408–2421, 2001.

14. G. Bilsen, M. Engels, R. Lauwereins, and J. A. Peperstraete, "Cyclo-static data flow," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 5, pp. 3255–3258, IEEE, 1995.

15. J. Gorin, M. Raulet, and F. Prêteux, "Mpeg reconfigurable video coding: From specification to a reconfigurable implementation," *Signal Processing: Image Communication*, vol. 28, no. 10, pp. 1224–1238, 2013.

16. G. Cedersjö and J. W. Janneck, "Software code generation for dynamic dataflow programs," in *Proceedings of the 17th International Workshop on Software and Compilers for Embedded Systems*, pp. 31–39, ACM, 2014.

17. E. Bezati, "High-level synthesis of dataflow programs for heterogeneous platforms," 2015.

18. J. W. Janneck, I. D. Miller, D. B. Parlour, G. Roquier, M. Wipliez, and M. Raulet, "Synthesizing hardware from dataflow programs: An mpeg-4 simple profile decoder case study," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, pp. 287–292, IEEE, 2008.

19. A. Ab Al Hadi Bin, *Optimizing Dataflow Programs for Hardware Synthesis*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2014.

20. F. Palumbo, D. Pani, E. Manca, L. Raffo, M. Mattavelli, and G. Roquier, "Rvc: A multi-decoder cal composer tool," in *Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on*, pp. 144–151, IEEE, 2010.

21. S. Dutt, "New faster kernighan-lin-type graph-partitioning algorithms," in *Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on*, pp. 370–377, IEEE, 1993.

22. F. Vahid and T. D. Le, "Extending the kernighan/lin heuristic for hardware and software functional partitioning," *Design Automation for Embedded Systems*, vol. 2, no. 2, pp. 237–261, 1997.

23. T. N. Bui and B. R. Moon, "Genetic algorithm and graph partitioning," *Computers, IEEE Transactions on*, vol. 45, no. 7, pp. 841–855, 1996.

24. P. Saini and E. Kaur, "Study of the circuit partitioning using genetic algorithm,"

25. S. S. Gill, R. Chandel, and A. Chandel, "Genetic algorithm based approach to circuitpartitioning," *International Journal of Computer and Electrical Engineering*, vol. 2, no. 2, p. 196, 2010.

26. K. S. Kumar, U. Bhaskar, S. Chattopadhyay, and P. Mandal, "Circuit partitioning using particle swarm optimization for pseudo-exhaustive testing," in *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on*, pp. 346–350, IEEE, 2009.

27. J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5, pp. 4104–4108, IEEE, 1997.

28. S. Mirjalili and A. Lewis, "S-shaped versus v-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.

29. J. D. McCaffrey, "Graph partitioning using a simulated bee colony algorithm," in *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, pp. 400–405, IEEE, 2011.

30. M. S. Soliman and G. Tan, "Graph partitioning using improved ant clustering," in *Advances in Swarm Intelligence*, pp. 231–240, Springer, 2010.

31. S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

32. A. Prakash and R. Lal, "Pso: An approach to multiobjective vlsi partitioning," in *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on*, pp. 1–7, IEEE, 2015.

33. S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: Amosa," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 3, pp. 269–283, 2008.

34. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

35. S. Bandyopadhyay, S. K. Pal, and B. Aruna, "Multiobjective gas, quantitative indices, and pattern classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 5, pp. 2088–2099, 2004.

36. J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization.," tech. rep., DTIC Document, 1995.