1-1-2011

Power Efficient Rapid Design Space Exploration of Integrated Scheduling and Module Selection in High Level Synthesis

Pallabi Sarkar
Ryerson University

POWER EFFICIENT RAPID DESIGN SPACE EXPLORATION OF INTEGRATED SCHEDULING AND MODULE SELECTION IN HIGH LEVEL SYNTHESIS

By

Pallabi Sarkar

Bachelor of Technology

Electronics and Communication Engineering

West Bengal University of Technology

Kolkata, India, 2008

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2011

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

* Signature

_____

Pallabi Sarkar

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

* Signature

_____

Pallabi Sarkar

ABSTRACT

Title of Thesis

POWER EFFICIENT RAPID DESIGN SPACE EXPLORATION OF INTEGRATED SCHEDULING AND MODULE SELECTION IN HIGH LEVEL SYNTHESIS

Thesis Submitted By:

Pallabi Sarkar, Master of Applied Science, 2011

Optimization Problems Research and Application Laboratory (OPR-AL)

Electrical and Computer Engineering Department, Ryerson University

Thesis Directed By:

Dr. Reza Sedaghat

Electrical and Computer Engineering Department, Ryerson University

High level Synthesis (HLS) or Electronic System Level (ESL) synthesis requires scheduling algorithms that have strong capability to reach optimal/near-optimal solutions with significant rapidity and greater accuracy. A novel power efficient scheduling approach using 'PI' method has been presented in this thesis that reduces the final power consumption of the solution at the expenditure of minimal latency clock cycles. The proposed scheduling approach is based on 'Priority indicator (PI)' metric and 'Intersect Matrix' topology methods that have a tendency to escape local optimal solutions and thereby reach global solutions. Application of the proposed approach results in even distribution of allocated hardware functional units thereby yielding power efficient scheduling solutions. The two main novel and significant aspects of the thesis are: a) Introduction of 'Intersect Matrix' topology with its associated algorithm which is used to check for precedence violation during scheduling b) Introduction of PI method using Priority

indicator metric that assists in choosing the highest priority node during each iteration of the scheduling optimization process. Comparative analysis of the proposed approach has been done with an existing design space exploration method for qualitative assessment using proposed 'Quality Cost Factor (Q- metric)'. This Q-metric is a combination of latency and power consumption values for the solution found, which dictates the quality of the final solutions found in terms of cost for both the proposed and existing approaches. An average improvement of approximately 12 % in quality of final solution and average reduction of 59 % in runtime has been achieved by the proposed approach compared to a current scheduling approach for the DSP benchmarks.

.

Acknowledgement

I would like to thank my supervisor, Dr. Reza Sedaghat for his thoughtful guidance and OPR-AL members for their endless support.

I am highly indebted to my parents for their great guidance and sacrifice all throughout my life. Further I highly owe them for being a constant source of love and motivation throughout my life, particularly in times of hardships and difficulty.

Moreover I am highly obliged to my grandparents for continuously supporting me and inspiring me to always do better than before.

I am also very thankful to my sister, my relatives and my friends, who helped me in tough times and provided me with encouraging words to accomplish my goals.

Table of Contents

List of Tables

List of Figures

Nomenclature

PF	Power fluctuation
T	Any arbitrary time instant
P	The power consumption at constant clock frequency
CS	Control step
Opn(i)	Movable operation under consideration
$\mathbf{S}_{\mathbf{before}}$	Difference in power consumption rate before movement of O_i
$\mathbf{S}_{\mathbf{After}}$	Difference in power consumption rate after movement of O_i
$\mathbf{N_{mul}}$	The number of multipliers present in the scheduling solution
$\mathbf{N_{add/sub}}$	The number of adder/subtractors present in the scheduling solution
$\mathbf{A_{mul}}$	The area occupied by a multiplier
$\mathbf{A_{add/sub}}$	The area occupied by an adder/subtractors
$\mathbf{p_c}$	Power consumed per area unit resource at a particular frequency
N_{mux}	The number of multiplexers required for the scheduling solution
N_{demux}	The number of demultiplexers required for the scheduling solution
N_{reg}	The number of registers required for the scheduling solution
$\mathbf{A_{mux}}$	The area occupied by each multiplexer
$\mathbf{A_{demux}}$	The area occupied by each demultiplexer
$\mathbf{A_{reg}}$	The area occupied by each register
W1	The weightage of the operating constraints for latency
W2	The weightage of the operating constraints for power consumption
L	The latency of the solution found

P	Power consumption of the solution found
$\mathbf{L_{max}}$	The value of maximum latency found by using minimum Functional Units
$\mathbf{P_{max}}$	The value of maximum power consumption found by using maximum Functional Units

Chapter 1
Introduction

1.1 Overview

The increased demand for performance improvement of Very Large Scale Integration (VLSI) systems, has forced VLSI designers to optimize the design at different levels of abstractions. Compared to the lower level of abstraction, it is well known that the optimization at the high level of abstraction has more impact on the design performance. Hence optimization at much higher level of abstraction known as 'architectural/algorithmic level' has gained momentum and the focus of many researches. A VLSI design at high-level of abstraction could be expressed in the behavioral domain in terms of algorithms. The algorithmic description specifies the inputs and outputs of the behavior of the algorithm in terms of operations to be preformed and data flow.

Moreover, the never ending increase in growth of the chip complexity has only been possible owing to efficient scheduling and exploration techniques proposed so far. The growth in capacity of the chip has enabled processing of huge amounts of data with greater flexibility and lesser expense. But the above condition can only prevail if the implementation cost satisfies the

user specified requirement of power consumption and latency. The application domain of the above mentioned requirements can be found out in the area of Digital Signal Processing (DSP), communications and network processing [1][2][3].

In the recent years there has been a major trend toward automating the design synthesis process at even higher levels of the design hierarchy. This automated design synthesis process called 'high-level synthesis' is therefore a process of conversion of the application from the algorithmic level to its respective RTL structure. High-level synthesis is gradually gaining acceptance in industry, and there has been considerable interest shown in Electronic system level designing by many well established EDA CAD vendors. Integrated Design Space Exploration of scheduling, allocation and binding in High Level Synthesis is often the most tedious process in the design process. Accurate exploration leads to high quality system design but may require extensive analysis resulting in increased design time. On the other hand, rapid exploration leads to reduced design time which eventually results in rapid marketing of the final end product, but may often be a victim of inferior quality solution due to limited precision during evaluation process. Hence a combination of the above two aspects of design space exploration: i) quality of the final exploration result ii) speed of the exploration process needs to be concurrently addressed in the design space exploration process based on the user specified objectives. This complicated process of design space exploration therefore involves tradeoffs between conflicting situations besides the contradictory objective parameters [1] [3]. Henceforth, exploration of an optimal/near-optimal solution that has the capability to encounter conflicting condition such as speed of the exploration process and quality of the solution found is extremely significant.

Moreover, the complicated process of exploration of final solution also requires tradeoff between the contradictory parameters of power and latency, besides the contradictory demands [3].

Additionally, recent advancements in areas of signal processing and multimedia have resulted in the growth of extensive array of applications requiring huge data processing at minimal power consumption expenditure. Such computation intensive applications demand acceptable performance with power competent hardware solutions. Hardware solutions should satisfy multiple contradictory performance parameters such as power consumption and time of execution. Since the selection process for the best design architecture is complex, an efficient approach to explore the design space for selecting the best design option is needed.

1.2 Related Works

In [1], the researchers have proposed an approach for design space exploration using priority factor method. The method uses a mathematically deduced framework called priority factor that is used for hierarchical arrangement of the vector design space consisting of all possible combination of design variants. Once the vector design space is hierarchically sorted in ascending/descending order then the border variant of each parameter from the design space is determined. Finally the Pareto optimal set is obtained that yields the final solution. The approach is highly efficient in terms of the exploration speed. But the drawback with this approach is the final quality of solution found, since in most cases due to partially arranged nature of the design space, the final solution found was local optimal in nature. Furthermore, authors in [2] introduce a tool called SystemCoDesigner that offers rapid design space exploration with rapid prototyping of behavioral systemC models. An automated integrated approach was developed by integrating behavioral synthesis into their design flow. SystemCoDesigner is a completely automated ESL

tool that provides a platform for hardware/software generation of System-on-Chip implementations. The approach performed tradeoffs between hardware cost and throughput. Since the thesis basically focused on bridging the gap from ESL to RT-level, hence scheduling algorithm was not the main focus of the work. But the proposed work focuses on ASAP latency constrained power efficient scheduling algorithm that mostly escapes the local optimal solutions. Authors in [4] have proposed a power optimization in SoC data flow systems. Authors have applied their optimization approach on 4G telecommunication modem system in order to show the power/energy savings obtained by their approach compared to existing approaches. Although the proposed optimization yielded significant results, but the focus of their work was not on scheduling approach but rather power optimization hardware during exploration. In [5], Genetic Algorithm (GA) has been suggested to yield better results for the design space exploration process. Authors have proposed the use of compact genetic algorithms for intrinsically evolvable hardware. Authors have improved upon the existing compact genetic algorithm that is based on probability vector based genetic algorithm that can be proficiently implemented in hardware. The results obtained on the benchmark resulted in increased efficiency and datapath design for implantation. The use of GA has also been suggested in [6] as a promising framework for DSE of data paths in high level synthesis. Their work employs robust search capabilities of the GA to resolve the datapath synthesis of scheduling and allocation of resources with the objective of finding a combination of scheduling and module/storage selection. Moreover the authors have used two different chromosome representations to encode the datapath schedules and functional unit part. Another approach introduced by researchers in [3] was based on Pareto optimal analysis using hybrid fuzzy searching algorithm. According to their work, the design space was arranged in the form of an architecture vector design space for architecture variant analysis and

optimization of performance parameters and then the proposed fuzzy search algorithm was applied for exploration. The fuzzy searching algorithm proposed is based on sets of fuzzy membership value functions that finds the border variant of architecture for the power consumption and performance parameters Although the method is extremely fast for exploration that reduces the final design time, but the approach also has a tendency to mostly yield local optimal solutions. Furthermore in [7] and [8], authors described another approach for DSE in high level systems based on binary encoding of the chromosomes. Authors in [7] have proposed a scheme for scheduling and allocation for functional and storage units. The method is based on power and latency constraint and performs pretty well for large designs. But the main drawback with this approach is the slow speed of exploration as well as the tendency to mostly find near-optimal solution. But the proposed approach aims to find a power efficient optimal solution at the expense of minimal latency expenditure which mostly has the tendency to reach global optimal. Authors in [9] have used an evolutionary algorithm for successful evaluation of the design for an application specific System on Chip. The work shown in [10] discusses about the optimization of area, delay and power in behavioral synthesis. But the work shown does not focus on an iterative hill climbing based design space exploration approach using selection value for power consumption and minimal latency constraint. The problem of exploration was also addressed in [11] by suggesting order of efficiency, which assists in deciding preferences amongst the different Pareto optimal points, while authors in [12] describe current state-of-the-art high-level synthesis techniques for dynamically reconfigurable system. In addition to above, authors in [13] have used GA to apply to the binding and allocation phase. The authors have introduced an unconventional crossover technique depending on a force directed datapath binding completion algorithm. One of the key features of their approach is the use of multiport

memories. The main drawback of the presented approach is that it accepts as input the scheduled data flow graph, thus is unable to handle the scheduling problem. This is because approach [13] is incapable to perform scheduling as mentioned. This is further evident because [13] can only perform exploration by accepting an already scheduled application. This is the major bottleneck in [13] since it needs some other approach to perform scheduling which could be used as an input for [13]. Besides, authors in [14], presented a time constrained scheduling based on genetic algorithm technique. The use of list decoder has been made to decode chromosome encoding by permutation of operations, into a valid schedule. Although the method is promising, but the method is slow compared to the other GA approaches. This is because [14] is based on Genetic Algorithm (GA) which has exponential time complexity unlike the proposed approach. Moreover, in order to find a good quality solution, the number of generations is always set to a value more than or equal to 100. This increases the total runtime to explore a good solution. In addition to above, authors in [15] have proposed a problem space genetic algorithm for design space exploration of data paths. The authors have used the concept of heuristic/problem pair to convert a data flow graph into a valid schedule. Another class of scheduling algorithms presented before were constructive approaches like As Soon As Possible (ASAP) [16], As Late As Possible (ALAP) [17], list scheduling [18], Force Directed scheduling [19]. The above approaches are very simple and fast in nature. The implementation complexity is also minimal for the above algorithms, but the above methods suffer from yielding poor solution in terms of hardware cost. Moreover Researchers in [20] have proposed an approach for synthesis of heterogeneous embedded systems by using Pareto Front Arithmetic (PFA) to explore the giant search spaces. Their method exploited the hierarchical problem structure for exploring the set of Pareto optimal solutions. Their method is quite promising, but the implementation complexity is large. Further

[20], does not consider power consumption optimization under minimal latency constraint during the scheduling process. Thus all the existing approaches on scheduling and design space exploration in ESL or high level synthesis has its own respective advantages and disadvantages.

1.3 Summary of Contribution

This thesis contributes to the following areas:

- Introduces a new topology for data dependency violation check of data flow graph based problems called 'Intersect Matrix'.
- Proposes a new algorithm in co-relation with the intersect matrix topology for determination of the parent-child relationship during data dependency violation check.
- Proposes a mathematical expression for Power Fluctuation based on the power consumption rate, which is used during determination of high priority nodes while searching for an optimal/near optimal scheduling solution.
- Presents a new priority function based selection criterion that takes into account the power fluctuation called 'Priority indicator (PI)'.
- Proposes a new iterative scheduling algorithm based on PI method.
- Presents a novel approach for finding the optimal/near optimal integrated solution to the problem of scheduling and module selection in High Level Synthesis.
- Provides a complete automated Design Space Exploration tool for rapid exploration of scheduling and module selection in high level synthesis design process.
- The proposed approach has successfully improved the quality of final solution on an average by 12% and reduced the exploration runtime on an average by 59% compared to a current approach for all the tested standard DSP Benchmarks.

1.4 Organization of the Thesis

The remaining part of the thesis is organized as follows: Chapter 2 gives a generic overview and background information on High Level Synthesis (HLS) and Design Space Exploration (DSE). Chapter 3 describes the proposed Dependency Matrix Topology for indicating the Data Dependency of Data Flow Graph (DFG). Chapter 4 provides the proposed Design Space Exploration Approach for Integrated Scheduling and Module Selection in High Level Synthesis. Chapter 5 demonstrates the proposed DSE Approach using the case study of Discrete Wavelet Transformation (DWT) Benchmark while Chapter 6 demonstrates the proposed Exploration by Considering Resource Binding (Interconnect Units) which uses the case study of Finite Impulse Response (FIR) Benchmark. Chapter 7 provides the experimental results, analysis and a vivid discussion of the proposed approach on DSP Benchmarks. Chapter 8 is dedicated to the conclusion and the future scope of the proposed work in this area. The list of publications related to this field of research study and the total list of citations are also provided thereafter. The thesis finally ends with the appendix.

Chapter 2

Background Information

2.1 A General Overview on High Level Synthesis

High-Level Synthesis (HLS) is an integral part of VLSI Electronic System Level designs. Lately, high-level synthesis has attracted significant attention in the CAD society. A lot of Electronic Design Automation (EDA) vendors based on CAD designs have shifted their design process to high-level synthesis. High-level synthesis traditionally is the conversion of the behavioral abstract description of the algorithm to its Register Transfer level (RTL) hardware structure. Important ingredients of high-level synthesis such as scheduling, allocation, binding and design space exploration have recently gathered renewed attention amongst the CAD researchers owing to its high capability to generate optimized hardware structures from high level specifications. High-level synthesis is the conversion from the abstract behavioral

description to its respective hardware description in the form of memory elements, storage units, multiplexers/demultiplexers and the necessary interconnections (called Register Transfer Level). But, this general process of High-level synthesis comprises of different complex procedural steps. These steps are very important in terms of the different research aspects of high-level synthesis. Research has been conducted and carried out in these different stratums of high-level synthesis.

2.2 Generic High-level synthesis Procedure

This section gives a vivid description of the different steps to be followed while reaching the final level called Register Transfer level (RTL). A framework has been constructed for the different procedural steps for high-level synthesis by discussing the General high-level synthesis procedure in this section. The generic high-level synthesis procedure can be described as follows. First, the process starts with the high level system specification such as area occupied by each resource, number of clock cycles needed to perform each operation by a specific resource, power consumed at a given frequency and also the user specified constraints for area, execution time and power consumption. Next the behavior or application required for the system is taken as an input which is then converted into a data flow graph. Subsequently the design space exploration (details about design space exploration are discussed in the next section) is carried out based on the user specified constraints like area, execution time, power etc. The following step called Scheduling represents the data flow graph of the application in the form of a sequencing flow graph into different time slots and binding the same type of operations i.e. grouping the same operations in same or different time slots. Scheduling can be represented in two different forms: time constrained scheduling and resource constrained scheduling. Time

constrained scheduling refers to finding the minimum cost schedule that satisfies the given set of constraints with the given maximum number of control steps. Resource constraint scheduling, on the other hand, refers to finding the fastest possible schedule that satisfies the given set of constraints with the given maximum number of resources. After the scheduling is correctly accomplished, the block diagram of the data path circuit is then developed. The controller structure is built next which provides the necessary synchronization signals. Finally, the combined structure consisting of the data path and the control path is the resulting system for the given application at the Register Transfer Level (RTL). The generic high-level synthesis overview is shown in Figure 1.

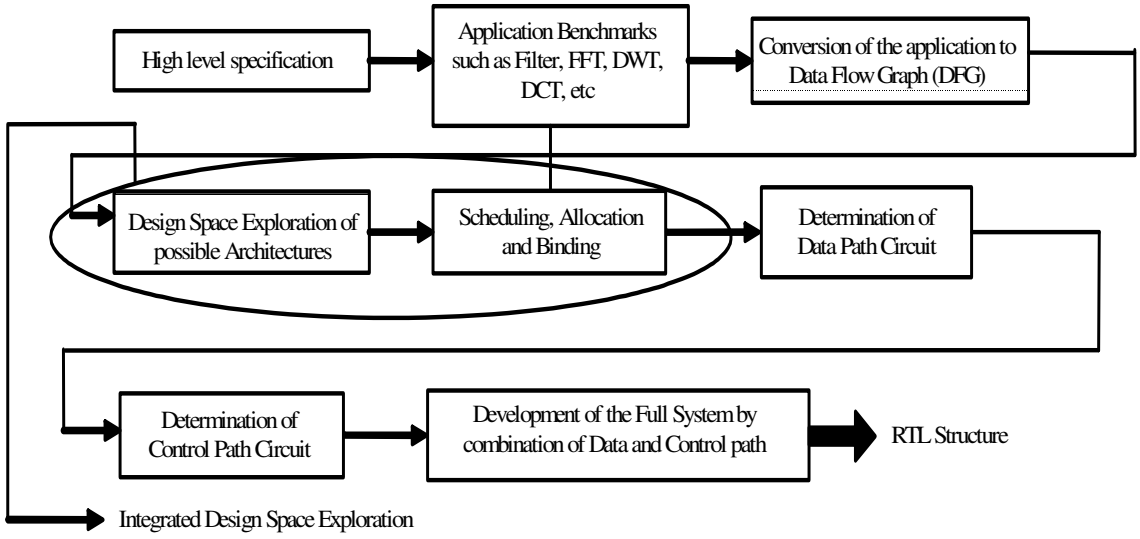


Figure 1. The Generic High-level synthesis

2.3 Overview on Design Space Exploration

The design of any modular VLSI systems is implementable in innumerable ways. Therefore the major challenge during the high-level synthesis designing process is to find the most suitable implementable hardware through design space exploration. Design space

exploration therefore generally involves the evaluation and selection of the optimal architecture based on the user specified requirements from the huge design space consisting of innumerable design alternatives. It is a procedure for analyzing the various design architectures in the design space to obtain the optimum architecture for the behavioral description according to the predefined user specifications.

Based on the constraints and specifications, the exploration of the optimal design point is very essential because this solution is to be carried forward in the next steps of high-level synthesis to reach the RTL structure. Also, if the constraints are satisfied while exploring the design space, an optimum result is expected further in the lower levels of abstraction. Based on the research performed till date, design space exploration in high-level synthesis can be broadly classified into two categories. First, design space exploration of architectures and second, the integrated design space exploration of scheduling, allocation and binding as discussed in the following two paragraphs respectively.

For the modular multi objective computing systems, fast and precise evaluation of the optimal system architecture is one of the most significant stages in the development process. The assessment and selection of the optimal design point is generally a complex procedure that requires lot of elaborate analysis. This process of architecture evaluation based on the user provided objective parameters are done through a sophisticated process called Design Space Exploration (DSE) of architectures. With the help of this exploration, several aspects are determined like the number of optimum resources, clock frequency etc.

For the modular VLSI computing architectures, the problem of solving the exploration process in a fast and accurate manner is very important. High-level synthesis is comprised of interdependent tasks such as scheduling, allocation, and module selection. For today's VLSI

designs, the cost of solving the combined scheduling, allocation, and module selection problem by exhaustive search is prohibitive. However, to meet design objectives, an extensive design space exploration is often critical to obtaining superior designs. Integrated design space exploration addresses multiple issues encountered during high-level synthesis such as scheduling, allocation and binding. These issues are highly critical for successful functioning of the system based on the user specified objectives. The characteristic of the integrated exploration lies in the fact that it does not only find the optimal architecture for the design but also explores the optimal scheduling and allocation needed to accomplish the task in given provided constraints.

2.4 Abstraction Level of Optimization in VLSI Design

In recent VLSI system designing, specifications are provides at a higher level of abstraction in order to attain maximum performance benefits at minimal cost. Further, specifying the requirements at a higher level of abstraction provides the designer with maximum flexibility for design optimization. Currently all the major EDA tool vendors are relying on high level synthesis which is designing the system from the highest level of abstraction. The EDA tools accept the application expressed in a high-level language as input and automatically produces the corresponding Register Transfer level implementation. All hardware systems can be classified into various levels of abstraction such as System level, Architecture level, Register Transfer Level (RTL), Layout level and Transistor level. In order to make the search for the optimal solution as effective as possible, the design decision taken at a very early stage of the development process provides more benefit in terms of the development time and also accuracy

in system development. Therefore, the focus for researchers has shifted towards optimization of multi parameters due to time to market pressure.

2.4 Importance and Significance of High Level Synthesis

In recent years there has been a trend towards automating synthesis at higher levels of the design hierarchy. Logic synthesis has gained acceptance in industry and there has been substantial interest shown in Register Transfer Level (RTL) synthesis. The significance of high level-synthesis are as follows [31]:

Reduction in errors and increase in reliability: If the synthesis process can be verified to be right-then there lies a greater assurance of the final design corresponding to the initial specification. This implies reduction in errors and an increase in reliability for new chips.

The ability to seek and explore the design space: A good synthesis system can produce several designs from the same specification in a reasonable amount of time. This allows the developer to explore different tradeoffs between cost, speed, power etc., or to take an existing design and produce a functionally equivalent one that is faster or less expensive. Even if the design is ultimately produced manually, automatically synthesized designs can suggest tradeoffs to the designer.

Decrease in the design cycle: If more of the design process is automated, it is possible to complete a design faster, and thus have a better chance of hitting the market window for the design. Moreover, since much of the cost of the chip is in design development, automating more of that process can lower the cost appreciably.

Documenting the design process: A track of design decisions made with their reasons and the effect of those decisions can be kept under the surveillance of an automated system.

Chapter 3

Proposed Dependency Matrix Topology

3.1 Dependency Matrix – A Matrix Topology for indicating the Data Dependency of the DFG

This section introduces a new matrix topology called '*Dependency Matrix*' which illustrates the data dependency present between the nodes of the data flow graph. Dependency Matrix represents all the information of the precedence relation present between the nodes of the DFG. This matrix is used in the proposed work for checking the data dependency between the predecessor and successor nodes during the scheduling process when during each iteration; a specific node will be moved for improving the scheduling solution. Thus a node is selected for movement as long as the node does not violate any precedence relationship indicated by the '*Dependency Matrix*'. The concept of '*Dependency Matrix*' is demonstrated with the aid of a popular DSP benchmark *'Discrete Wavelet Transformation (DWT)'* as shown in Figure 2. The

role of '*Dependency Matrix*' only comes into action when the scheduling of the DFG is to be performed. Thus the concept of '*Dependency Matrix*' can be explained through the data flow graph of the DWT benchmark shown in Figure 2.

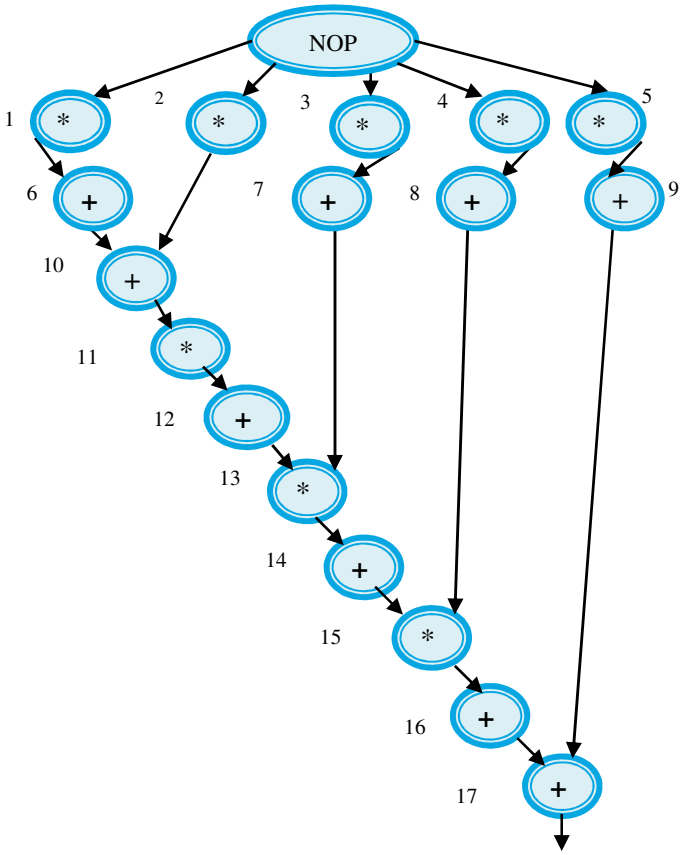
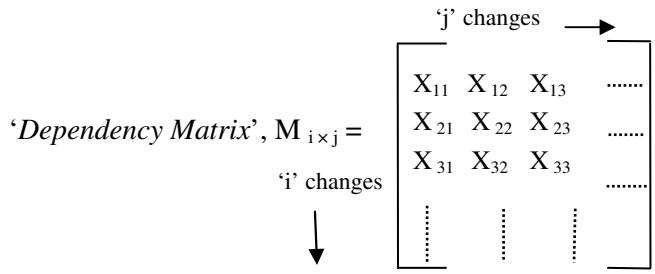


Figure2. DFG of the Discrete Wavelet Transformation

The '*Dependency Matrix*' is a matrix consisting of nodes of the DFG where any edge between the two nodes under test ('i' and 'j') is denoted by '1', while any non-existence of edge between the two nodes under test ('i' and 'j' is denoted by '0'. Therefore, the dependency relationship for an example matrix can be defined as follows. Let a '*Dependency Matrix*' M is defined as:



Where 'i' is the row of the matrix 'M' and 'j' is the column of the matrix 'M'. In the above matrix 'M' the dependency relation is defined as follows:

a) $X_{ij} = 1$; if there exists an edge between the two nodes under test ('i' and 'j').

b) $X_{ij} = 0$; if there exists no edge between the two nodes under test ('i' and 'j').

c) $X_{ij} = Z$; if i = j (This means that only one node is under test).

Operation	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
1	Z	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	Z	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	Z	0	0	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	Z	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	Z	0	0	0	1	0	0	0	0	0	0	0	0
6	1	0	0	0	0	Z	0	0	0	1	0	0	0	0	0	0	0
7	0	0	1	0	0	0	Z	0	0	0	0	0	1	0	0	0	0
8	0	0	0	1	0	0	0	Z	0	0	0	0	0	0	1	0	0
9	0	0	0	0	1	0	0	0	Z	0	0	0	0	0	0	0	1
10	0	1	0	0	0	1	0	0	0	Z	1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	Z	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1	Z	1	0	0	0	0
13	0	0	0	0	0	0	1	0	0	0	0	1	Z	1	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	Z	1	0	0
15	0	0	0	0	0	0	0	1	0	0	0	0	0	1	Z	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Z	1
17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	Z

Figure3. The Dependency Matrix for the Discrete Wavelet Transformation

Where, X_{ij} denotes any element of the matrix $M_{i \times j}$ ranging from a_{ij} to i_{ij}. Further, X_{ij} can contain any *intersect value* '0', '1', or 'Z'. From the previous general definitions on

'*Dependency Matrix*', the '*Dependency Matrix*' (M) for the discrete wavelet transformation benchmark is described above in Figure 3. Therefore as visible from the 'dependency matrix' in Figure 3, the matrix strictly follows the three rules enlisted before. The matrix consists of 17 nodes (operations) from the DFG of DWT, where each element of the matrix (X_{ij}) has been assigned an intersect value (0, 1 or Z). Once the dependency matrix is formed then the information of data dependency is clearly visible. But the determination of the parent and child from the matrix is still not evident. Hence, an additional algorithm has been proposed to determine (indicate) the parent and child for the two nodes under test. For example, in Figure 3, the intersect value of element $X_{16} = 1$. So this means that the two nodes under test (node 1 and node 6) do have a parent child relationship. But still we don't know who the parent is and who the child is (in other words, the parent and child is not yet indicated). Hence the determination of parent-child is not evident. Similarly, arguments apply for elements, X_{37}, X_{59} etc. The algorithm in Figure 4 is proposed to find out the parent-child status. According to the algorithm if the intersect value is '0' or 'Z' for the two nodes under test then the algorithm stops and does not need to determine the parent and child since there exists no parent-child relationship. But if the intersect value is '1' then the algorithm goes to step 2, to determine the parent and child. For example, in case of element X_{16} (for node 1 and node 6 under test), the intersect value is '1' signifying a parent-child relationship. Next according to step 2 of the algorithm, since the 1<6, hence node 1 is the parent and node 6 is the child. This conclusion is in compliance with the DFG in Figure 2. Hence, the dependency matrix and the algorithm in Figure 4 both provide a medium for precedence violation check during scheduling process.

Algorithm for Precedence Violation check and determination of Parent-Child status

1. *If* the *intersect value* is '0' between the two nodes under test ('i' and 'j') in the dependency matrix, then it indicates that **no parent-child relationship** exists between the two nodes.

   Next stop

   Else if the *intersect value* is '1' between the two nodes under test ('i' and 'j') in the dependency matrix then it signifies that **a parent-child relationship exists**.

   Next Goto Step 2

   Else if the *intersect value* is 'Z' in the dependency matrix for a same node under test ('i' = 'j') then it signifies that **no parent-child relationship exists**.

   Next stop

2. *If* i < j (if the numerical value of node 'i' is less than the numerical value of node 'j'), then i = Parent node and j = Child node

   Else, i = Child node and j = Parent node.

Figure4. Algorithm for Parent-Child Determination and Precedence violation check

Chapter 4

Proposed Design Space Exploration Approach for Integrated Scheduling and Module Selection in High Level Synthesis

4.1 Proposed Concept behind the Exploration Process

The input to the proposed DSE approach is a data flow graph of the application along with the set of all the module library information's. This consists of area, delay, and power consumption etc of the functional units. Once the DFG of the application is taken as an input then the As Soon As Possible (ASAP) scheduling is performed to schedule the operations in the least possible control step (CS). This is done in order to have an initial scheduling solution to the DFG problem. Although the initial solution obtained is not efficient in terms of hardware area usage but still ASAP algorithm is used as a preliminary method of finding the initial solution because

this algorithm schedules the operations in the earliest possible control step, thus providing an opportunity to the proposed approach to improve the quality of the scheduling solution by taking into account the power consumption rate and minimization of hardware area at minimal possible latency expenditure. The improvement in scheduling quality is obtained gradually in each iteration by selecting the high priority nodes using a selector metric called Priority Indicator (PI) which is a function of power fluctuation and cost of potentially movable resource. The short definition of power fluctuation is given in equation (1). The detailed explanation and application about equation (1) is given later in section 4.3.

Power fluctuation = [*Difference in total power consumption rate between control step T(j) and T(k) before movement of operation (o_i)]* − [*Difference in total power consumption rate between control step T(j) into T(k) after movement of operation (o_i)];*

Where *power consumption rate*: $\frac{\Delta P}{\Delta T} = \frac{(P_2 - P_1)}{(T_2 - T_1)}$ (1)

Where initial power consumption (P_1) is assumed to be zero watts because of no functional operation at initial time instant (T_1=0) while P_2 is the power consumed at any other time instant (T_2); Assuming constant clock frequency for above equation described above; 'P' is the power consumption at constant clock frequency; T(j) and T(k) are the consecutive control steps in the ASAP scheduled graph. Equation (1) is described in details later in equation (2) and (3) in section 4.3.

4.2 Proposed Framework of the Iterative Design Space Exploration Method

The Proposed framework for the design space exploration of scheduling and module selection is based on an iterative algorithm that takes into account the minimization of power

consumption rate of the resources and minimization of hardware area at the expense of least latency expenditure. The proposed approach is based on hill-climbing property where the inferior solutions obtained due to the bad moves is accepted hoping for a better solution. The resultant integrated solution obtained at the end through the proposed approach is a solution with minimum power consumption under ASAP constrained latency. The proposed approach accepts the DFG of the application as an input along with the set of module library information. Once the DFG is provided as an input, the approach first converts the DFG into an ASAP scheduling solution by always keeping a check on the precedence violation using the proposed dependency matrix and the algorithm in Figure4. The schedule solution along with the FU's needed to implement the ASAP acts as the initial solution for the proposed design space exploration approach. The proposed exploration approach uses this initial solution to find out an optimal/near-optimal integrated solution while the objectives are simultaneously met. The approach works on an iterative manner like algorithms such as simulated annealing, genetic algorithm etc, trying to improve the quality of the solution produced. According to the proposed approach, in each iteration only one operation (node) can be moved at a time into its next immediate control step as long as the dependency is obeyed. The selection of a particular operation (node) is chosen based on the value of 'PI'. The 'PI' acts as a determining metric to choose the highest priority node (operation) among the existing available movable operations that can result in reducing the cost of the final solution. The *PI* proposed in this work is a function of power fluctuation and cost of a particular resource since the main objective is to reduce the power consumption rate of the resources and minimize the hardware cost. The overview of the proposed design space exploration approach is shown in Figure5.

Proposed Scheduling Approach using PI Method

1. User specified module library with area, latency and power consumed per area unit is taken as an input.

2. Data flow graph (DFG) of the application is taken as the input.

3. ASAP scheduling will be performed for the data flow graph *(Note: By keeping a check on the precedence violation of the nodes using the proposed dependency matrix and the dependency algorithm)*. Once the ASAP schedule is done then the cost of the initial solution is calculated.

4. ASAP scheduling imposes a restriction on the number of control steps i.e. Latency constraint is imposed. Hence now the algorithm tries to improve the scheduling solution by minimizing the power fluctuation and hardware area.

5. In each iteration, all the movable operations are identified and then the value of '*Priority indicator (PI)*' is calculated for each movable operation (node). The '*Priority indicator*' is function of power fluctuation and cost of each resource.

6. The movable operation (node) with the highest '*Priority indicator (PI)*' value is selected for movement into its next immediate control step (*Note: Before the node selected by the 'PI' is moved, precedence violation check is performed using dependency matrix and the dependency algorithm).*

   - If there is a tie between the *Priority indicator* values then randomly any operation is chosen for movement. Once the new scheduling solution is found after the current iteration then the cost of the scheduling solution is again calculated.

7. This above procedure is repeated till the terminating condition is reached (The terminating condition chosen is the maximum number of nodes in the DFG. i.e. # of iterations = # of nodes in the DFG).

8. Finally, the integrated optimal/near-optimal solution with respect to hardware cost and power consumption is yielded. The solution indicates the optimal/near-optimal scheduling of the DFG and the optimal/near-optimal resource combination (FU's) needed for allocation.

(Note: The iteration which yielded the solution with the minimum cost among all iterations is chosen as the final optimal solution).

Figure5. The overview of the proposed design space exploration approach

4.3 Proposed Priority indicator (PI) metric used for Selection of high priority nodes during movement

Let us now elaborate the mathematical expression of power fluctuation defined before in section 4.1. The mathematical expression of power fluctuation as given from equation (1) before is explained in equation (2) and (3) respectively:

$$Power\ Fluctuation = S_{before} - S_{after} \quad (2)$$

Where, S_{before} = Difference in power consumption rate before movement (CS (j), CS (k))

And, S_{After} = Difference in power consumption rate after movement (CS (j), CS (k)).

Therefore, from equation (2) above:

Power Fluctuation (PF) = [Difference in power consumption rate before movement (CS (j), CS (k))] – [Difference in power consumption rate after movement (CS (j), CS (k))]

Equation (2) above can be further expanded into equation (3) as shown below:

PF = [*Power consumption rate at CS (j) - Power consumption rate at CS (k)*] –

[{*Power consumption rate at CS (j) – Power consumption rate for opn (i)*} – {*Power consumption rate at CS (k) + Power consumption rate for opn (i)*}] (3)

Where, CS (j) and CS (k) are the two immediate control steps in the temporary scheduling solution, opn (i) is the operation selected for movement through the Priority indicator (PI) metric. The metric above called 'power fluctuation' defined in equation (2) and (3) will be used in a function described later in equation (4) for selection of the highest priority node for movement during optimization (design space exploration). Power fluctuation defined above takes into account the change in power consumption rate when a certain operation is moved from one control step to another. The reduction in power consumption that can be achieved by the

proposed DSE is possible owing to balance in distribution of resources in the scheduling solution after the application of the proposed approach (which thereby helps in reducing the power consumption rate). Thus the balance in number of functional units can be obtained by decreasing the number of functional units in the control step where the power consumption rate is maximal. The concept above has been illustrated in the following chapters 5 and 6 where case studies of Discrete Wavelet Transformation (DWT) and Finite Impulse Response (FIR) benchmarks have been shown to demonstrate the proposed DSE approach.

Let us now introduce the proposed '*Priority indicator (PI)*' metric used for selection of high priority nodes during movement. The proposed 'Priority indicator' is shown in equation (4) below:

$$PI = Power\_fluctuation * Max[\frac{\Delta P}{\Delta T}(j), \frac{\Delta P}{\Delta T}(k)] * Cost[Opn(i)] \quad (4)$$

Where '*Cost [opn (i)]*' is obtained from the module library, power fluctuation is obtained from equation (3) and '*Max [*ΔP/ΔT *(j),* ΔP/ΔT *(k)]'* signifies the maximum of the power consumption rate between CS (j) and CS (k).

Chapter 5

Demonstration of Proposed DSE Approach

5.1 Case Study of Discrete Wavelet Transformation (DWT) Benchmark

This section illustrates the proposed integrated design space exploration framework described above with an example of discrete wavelet transformation benchmark. Application of the proposed DSE on DWT benchmark yielded impressive final results in terms of reduced power consumption (hence hardware area) under latency constraint. The final runtime taken to find the final global solution is also very less compared to other heuristic based approaches.

The Data Flow Graph of the DWT benchmark is shown in Figure 2 before and the As Soon As Possible (ASAP) scheduling of DWT is shown in Figure 6. This ASAP scheduling solution found acts as an initial solution for the proposed approach as mentioned in Figure 5. The latency of the ASAP scheduling solution calculated is 28 clock cycles (*Note: assuming that adder/subtractor and multiplier takes 2clock cycles (cc) and 4 cc respectively as specified in the*

module library). Hence ASAP scheduling imposes a latency constraint on the final solution that has to be found by the proposed approach. The cost of each schedule solution found is calculated using equation (5):

$$\text{Cost}_{ini} = \{(N_{mul} * A_{mul}) + (N_{add/sub} * A_{add/sub})\} * p_c \quad (5)$$

Where N_{mul} and $N_{add/sub}$ are the number of multipliers and adder/subtractors respectively present in the scheduling solution. A_{mul} and $A_{add/sub}$ is the area occupied by a multiplier and an adder/subtractor respectively and p_c is the power consumed per area unit at a particular frequency of operation.

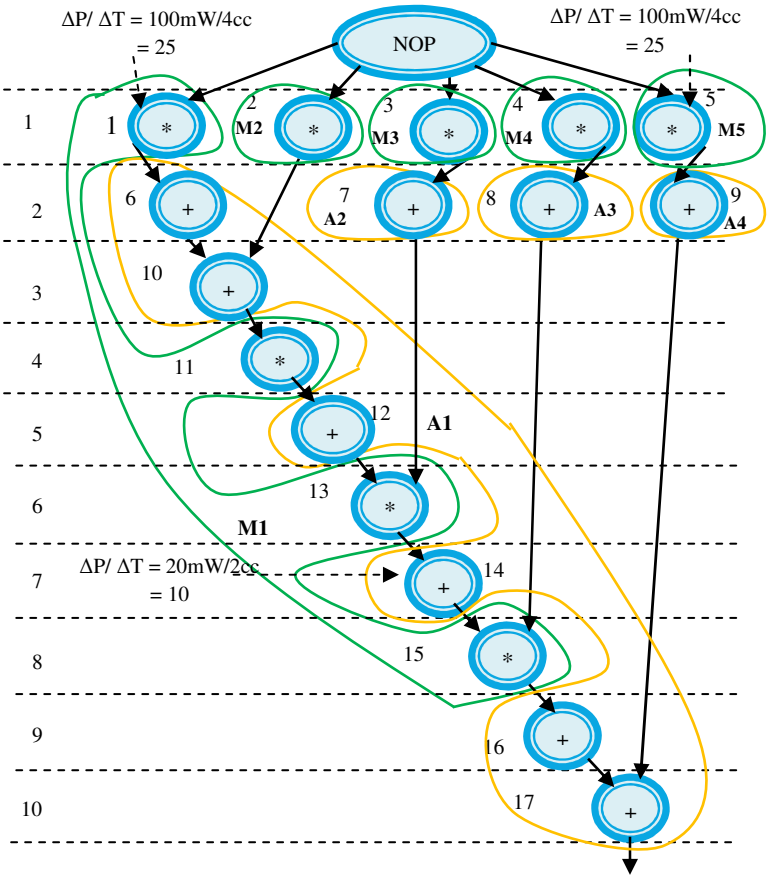


Figure6. ASAP scheduling of DWT benchmark

Therefore using equation (5) as mentioned before, the cost of initial ASAP schedule solution (of figure 6) is:

Cost $_{ini}$ = {(5 * 100) + (4 * 20)} * 1 = 0.58 Watts.

(*Note: assuming multiplier and adder/subtractor occupies 100 CLB's and 20 CLB's respectively; where 1 area unit (au) = 1 CLB has been assumed; Power consumed per au (p_c) at 24 MHz clock frequency is 1 milli-watt. The costs of multiplier and adder/subtractor have been assumed to be 5 units and 3 units respectively.*)

5.2 Description of the Iteration Process

After determining the cost of initial ASAP schedule solution as illustrated in section 5.1 before, the next step of the algorithm (as proposed in Figure 5) is to identify all the movable candidate operations. Once the movable operations (opn) are identified then the iteration process begins to improve the initial scheduling solution. The iteration process is described below:

Iteration (1):

i) Movement – opn 2 (1→2) (a)

ii) Movement – opn 7 (2→3) (b)

iii) Movement – opn 8 (2→3) (c)

iv) Movement – opn 9 (2→3) (d)

For example equation (a) above signifies that opn 2 is one of the identified movable operations that can be moved from Control Step (CS) 1 to CS 2. Now the 'PI' for each identified movable operation is calculated using equation (4). But before the 'PI' is calculated, the '*power fluctuation*' is determined as follows using equation (3):

Power fluctuation = [*Power consumption rate at CS (j) - Power consumption rate at CS (k)*]

– [{*Power consumption rate at CS (j) – Power consumption rate for opn (i)*} – {*Power consumption rate at CS (k) + Power consumption rate for opn (i)*}]

As shown in Figure 6, power consumption rate for adder is (20au * 1mW)/2cc = 10, while for multiplier is (100au * 1mW)/4cc = 25. (*Note: From [1] [2], total power consumption of resource (Ri) is:* $P_{Ri} = A_{Ri} * p_c$ *, where* A_{Ri} *is the area of Ri. Power consumed per au* (p_c) *at 24 MHz clock frequency is 1 milli-watt. Further, the production costs of multiplier and adder/subtractor have been assumed to be 5 units and 3 units respectively.)*

Before substituting in equation (4), the power fluctuation for the first case a) between CS (j) and CS (k) is calculated from Figure 6 using equation (3) as follows:

= [(25 + 25 + 25 + 25 +25) – (10 + 10 + 10 + 10)] – [(25 + 25 + 25 + 25) – (10 + 10 + 10 + 10 + 25)]

= [125 – 40] – [100 – 65] = 50.

Now the value for '*Power Fluctuation'* calculated above is substituted in equation (4) as shown below:

$$PI = Power\_fluctuation * Max[\frac{\Delta P}{\Delta T}(j), \frac{\Delta P}{\Delta T}(k)] * Cost[Opn(i)]$$

a) PI $^{opn\ 2}$ (1→2) = 50 * Max (125, 40) * 5 = 31,250 (selected).

Similarly, calculating the '*Power Fluctuation'* for each case and then finding the *Priority Indicator* yields:

b) PI $^{opn\ 7}$ (2→3) = 20 * Max (40, 10) * 3 = 2400.

c) PI $^{opn\ 8}$ (2→3) = 20 * Max (40, 10) * 3 = 2400.

d) PI $^{opn\ 2}$ (1→2) =20 * Max (40, 10) * 3 = 2400.

According to the next step of the algorithm, the highest PI is selected for movement; which is a) in this case. The respective scheduling solution found after iteration 1 is shown in Figure 7. The cost of the scheduling solution is:

Cost = {(4 * 100) + (4 * 20)} * 1 = 0.48 Watts.

Thus we see that the cost in terms of power consumption reduces from the initial solution.

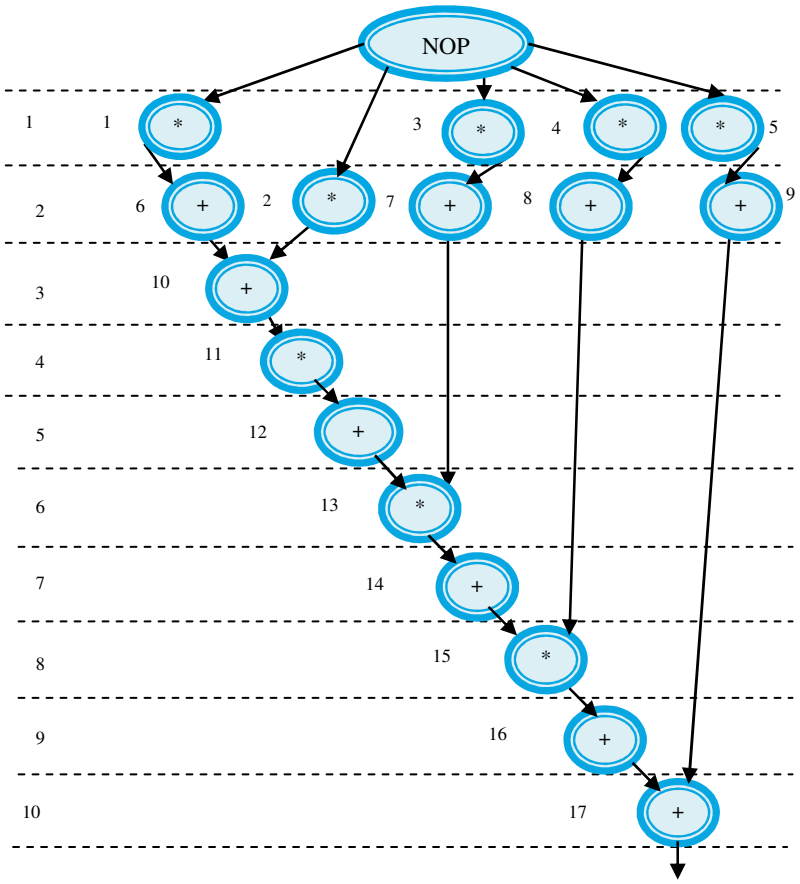


Figure7. Scheduling of DWT benchmark after 1st Iteration

Iteration (2):

i) Movement – opn 7 (2→3) (a)

ii) Movement – opn 8 (2→3) (b)

iii) Movement – opn 9 (2→3) (c)

Therefore the PI obtained for each operation is:

a) PI $^{opn\ 7}$ (2→3) = 3900 (selected).

b) PI $^{opn\ 8}$ (2→3) = 3900.

c) PI $^{opn\ 9}$ (2→3) = 3900.

Since there is a tie between the PI values hence the tie is randomly broken as per the algorithm in Figure 5. Thus operation 7 is chosen randomly for movement. The respective temporary scheduling solution found after iteration 2 is shown in Figure 8. The cost of this solution is:

Cost = {(4 * 100) + (3 * 20)} * 1 = 0.46 Watts.

Again after this iteration a reduction in power consumption is noted.

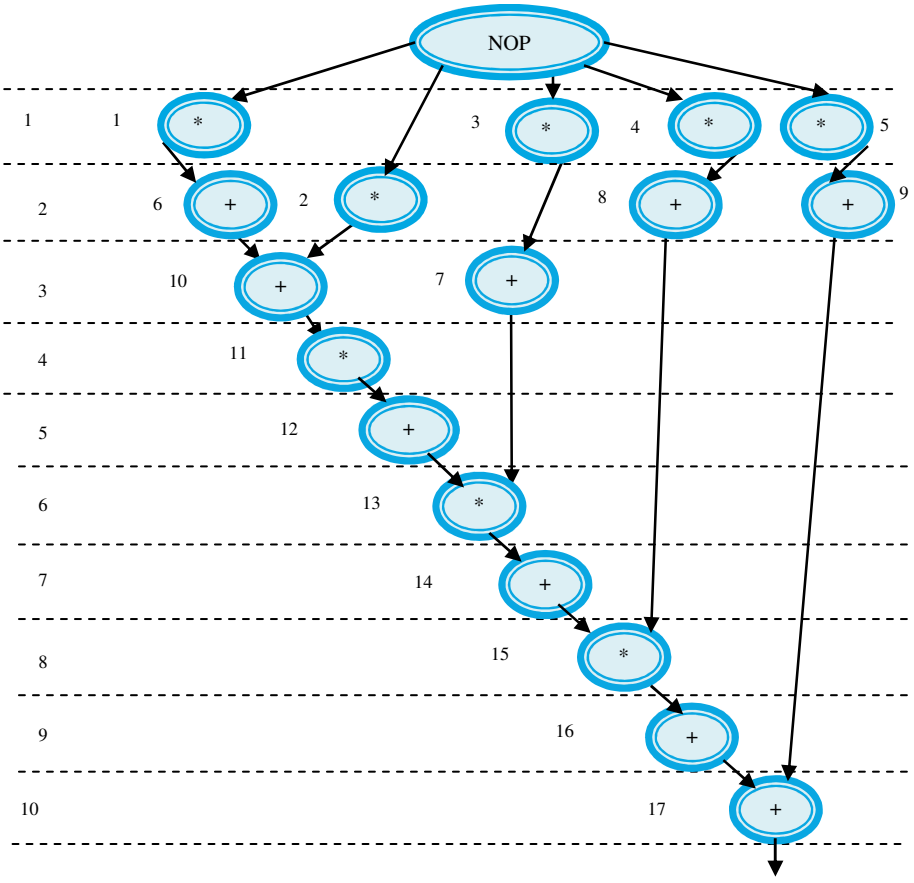


Figure8. Scheduling solution of DWT benchmark after 2nd Iteration

Iteration (3):

i) Movement – opn 3 (1→2) (a)

ii) Movement – opn 8 (2→3) (b)

iii) Movement – opn 7 (3→4) (c)

iv) Movement – opn 9 (2→3) (d)

Therefore the PI obtained for each operation is:

a) PI $^{opn\ 3}$ (1→2) = 25000 (selected).

b) PI $^{opn\ 8}$ (2→3) = 3300.

c) PI $^{opn\ 7}$ (3→4) = 1500.

d) PI $^{opn\ 9}$ (2→3) = 3300.

According to the algorithm, the highest PI is selected for movement which is a) in this case. The respective scheduling solution found after iteration 3 is shown in Figure 9.

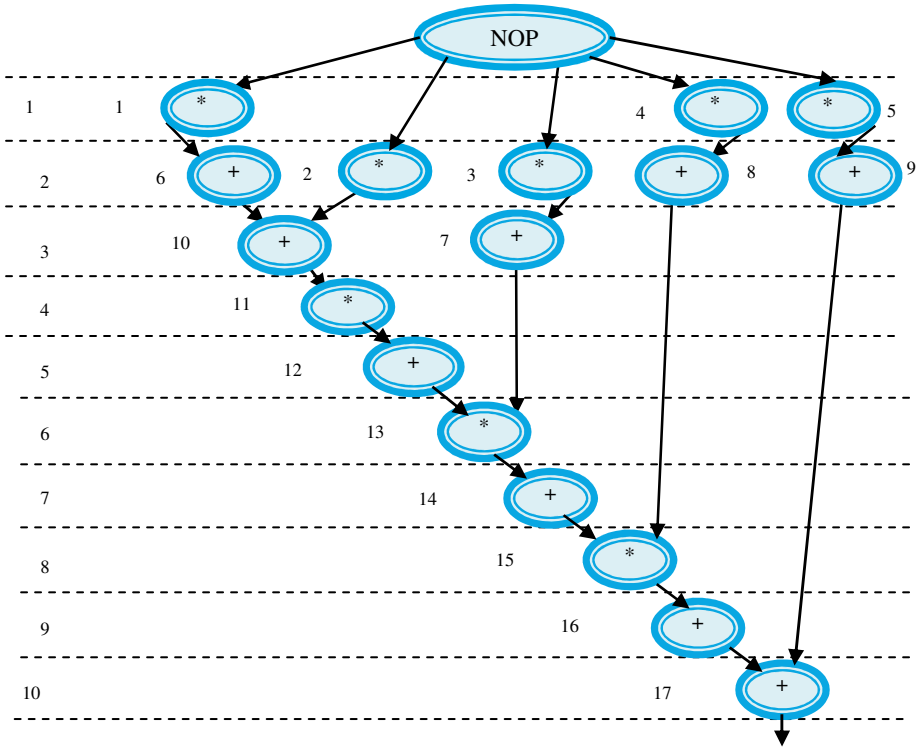


Figure9. Scheduling solution of DWT benchmark after 3rd Iteration

The cost of this respective scheduling solution is:

Cost = {(3 * 100) + (3 * 20)} * 1 = 0.36 Watts.

Again after this iteration a reduction in power consumption is noted compared to the previous scheduling solution.

Iteration (4):

i) Movement – opn 7 (3→4) (a)

ii) Movement – opn 8 (2→3) (b)

iii) Movement – opn 9 (2→3) (c)

Therefore the PI obtained for each operation is:

a) PI $^{opn\ 7}$ (3→4) = 20 * Max (20, 25) * 3 = 1500.

b) PI $^{opn\ 8}$ (2→3) = 20 * Max (80, 20) * 3 = 4800 (selected).

c) PI $^{opn\ 9}$ (2→3) = 20 * Max (80, 20) * 3 = 4800.

Where, the calculated value of '*power fluctuation*' is 20 in all the above cases.

According to the algorithm, the highest PI is selected for movement which is b) in this case. The respective scheduling solution found after iteration 4 is shown in Figure 10. The cost of this respective scheduling solution is:

Cost = {(3 * 100) + (3 * 20)} * 1 = 0.36 Watts.

No reduction in power consumption is noted in this scheduling solution. Since the algorithm will not be terminated until *# of iterations = # of nodes* in the DFG, hence any local optimal solution found will not restrict the algorithm from stopping.

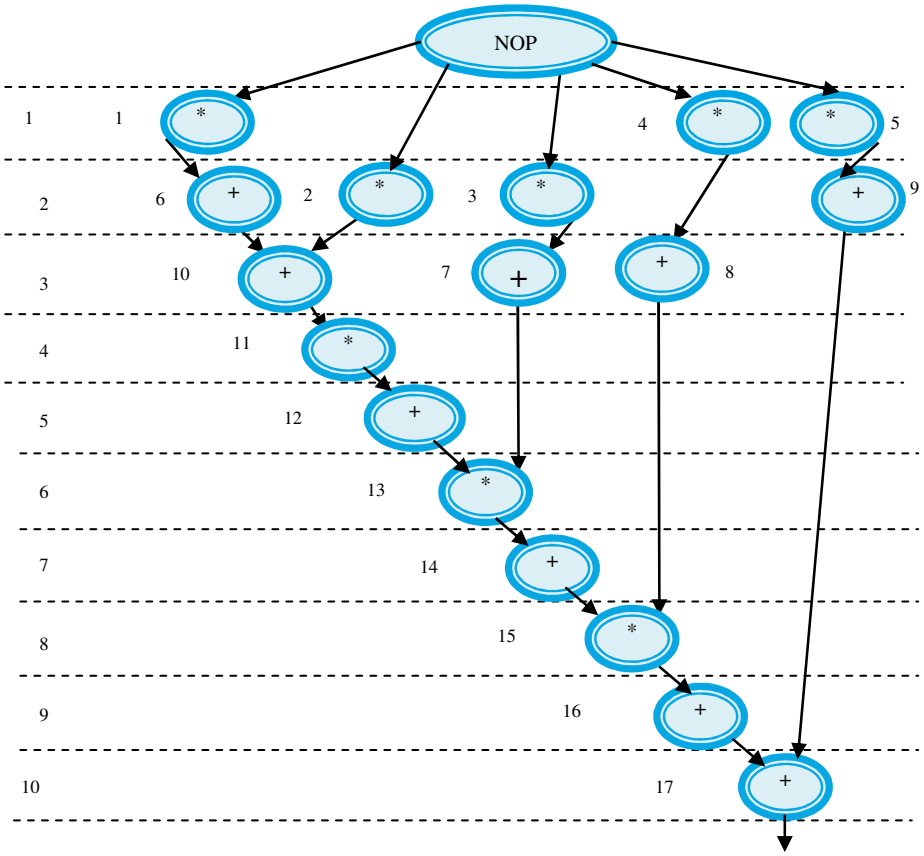


Figure10. Scheduling solution of DWT benchmark after 4th Iteration

Iteration (5):

i) Movement – opn 4 (1→2) (a)

ii) Movement – opn 8 (3→4) (b)

iii) Movement – opn 9 (2→3) (c)

iv) Movement – opn 7 (3→4) (d)

Therefore the PI obtained for each operation is:

a) PI $^{opn\ 4}$ (1→2) = 18750 (selected).

b) PI $^{opn\ 8}$ (3→4) = 1800.

c) PI $^{opn\ 9}$ (2→3) = 4200.

d) PI $^{opn\ 7}$ (3→4) = 1800.

According to the algorithm, the highest PI is selected for movement which is a) in this case. The cost of this respective scheduling solution is:

Cost = {(3 * 100) + (3 * 20)} * 1 = 0.36 Watts.

Again no reduction in power consumption is noted in this scheduling solution. Since the algorithm will not be terminated until # of iterations = # of nodes in the DFG, hence the local optimal solution found does not restrict the algorithm from stopping. This above iteration continues until the algorithm reaches iteration 17 (Since the maximum number of nodes present in this DFG is 17). Finally the algorithm yields the solution with the minimum final cost in these 17 iterations. Experiment revealed that iteration 11 yielded the scheduling solution with the minimum cost. The iteration 11 is described below:

Iteration (11):

i) Movement – opn 4 (2→3) (a)

ii) Movement – opn 8 (4→5) (b)

iii) Movement – opn 9 (3→4) (c)

iv) Movement – opn 7 (4→5) (d)

Therefore the PI obtained for each operation is:

a) PI $^{opn\ 4}$ (2→3) = 21250 (selected).

b) PI $^{opn\ 8}$ (4→5) = 2700.

c) PI $^{opn\ 9}$ (3→4) = 2700.

d) PI $^{opn\ 7}$ (4→5) = 2700.

According to the algorithm, the highest PI is selected for movement which is a) in this case. The respective scheduling solution found after iteration 11 is shown in Figure 11.

The cost of this respective scheduling solution is:

Cost = {(2 * 100) + (2 * 20)} * 1 = 0.24 watts.

Hence iteration 11 found the optimal solution to the scheduling problem. The final reduction in cost in terms of power consumption obtained compared to the initial solution (in Figure 6) is 0.58 Watts – 0.24 Watts = 0.34 Watts.

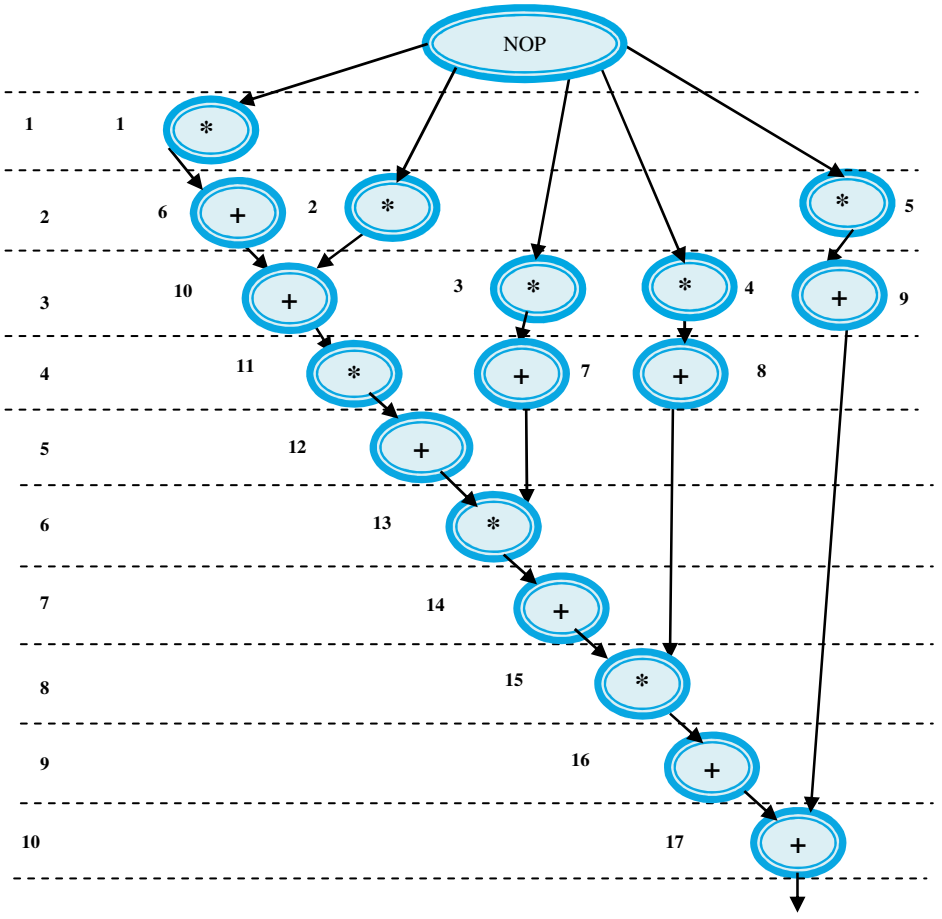


Figure11. Scheduling solution of DWT benchmark after 11th Iteration

Chapter 6

Demonstration of the Proposed Exploration by Considering Resource Binding (Interconnect Units)

6.1 Case Study of Finite Impulse Response (FIR) Benchmark

This section illustrates the proposed integrated design space exploration framework with an example of finite impulse response benchmark. Application of the proposed DSE on FIR benchmark yielded impressive final results in terms of reduced power consumption (hence hardware area) under maximum latency constraint. The final runtime taken to find the final global solution is also very less compared to other heuristic based approaches.

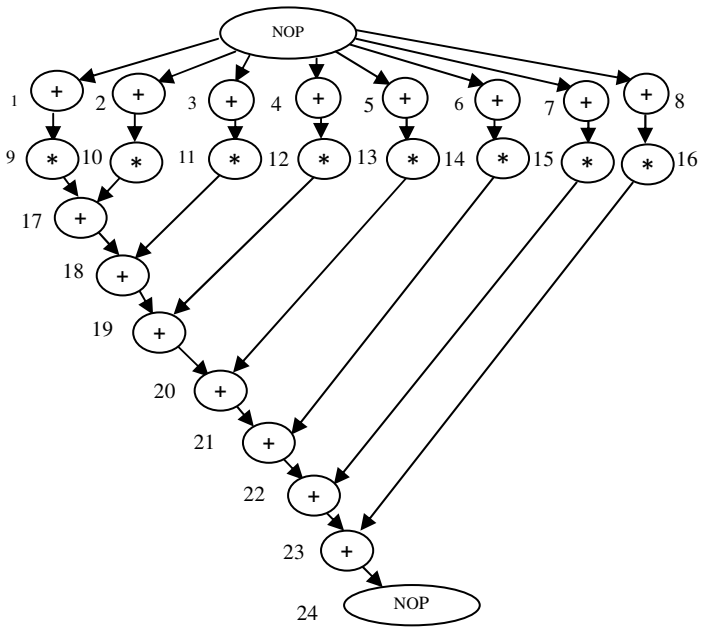


Figure12. DFG of the Finite Impulse Response (FIR) benchmark

The DFG of the FIR benchmark is shown in Figure 12 and the ASAP scheduling of FIR is shown in Figure 13. This ASAP scheduling solution found acts as an initial solution for the proposed approach as mentioned in Figure 5. The latency of the ASAP scheduling solution calculated is 20 clock cycles *(Note: assuming that adder/subtractor and multiplier takes 2 clock cycles (cc) and 4 cc respectively as specified in the module library).* Hence ASAP scheduling imposes a latency constraint on the final solution that has to be found by the proposed approach. The cost (power consumption) of the each schedule solution found is calculated using equation (6) from [1] [3] [22]:

$$\textit{Power consumption } (Cost_{ini}) = \{(N_{mul} * A_{mul}) + (N_{add/sub} * A_{add/sub}) + (N_{mux} * A_{mux}) + (N_{demux} * A_{demux}) + (N_{reg} * A_{reg})\} * p_c \quad (6)$$

Where, N_{mul} and $N_{add/sub}$ are the maximum number of multipliers and adder/subtractors needed for implementing the scheduling solution respectively. A_{mul} and $A_{add/sub}$ are the area occupied by

multipliers and adder/subtractors respectively. Further N_{mux}, N_{demux} and N_{reg} are the number of multiplexers, demultiplexers and registers needed respectively, while A_{mux} , A_{demux} and A_{reg} are the area of each multiplexer, demultiplexer and register in area units (au) respectively; 'p_c' is the power consumed per area unit at a particular frequency of operation.

Therefore using equation (6), the cost of initial ASAP schedule solution (of figure 13) is Cost $_{ini}$ = {(8 * 100) + (8 * 20) + (32 * 3) + (16 * 3) + (23 * 5)}* 4 = 4.87 Watts. *(Note: assuming multiplier and adder/subtractor occupies 100 au and 20 au respectively; also assuming each mux/demux is 3 au and each register is 5 au respectively; Power consumed per au (p_c) at 50 MHz clock frequency is 4 milli-watt. Further, the production costs of multiplier and adder/subtractor have been assumed to be 5 units and 3 units respectively.)*

Now according to the next step of the algorithm proposed in Figure 5, all the movable

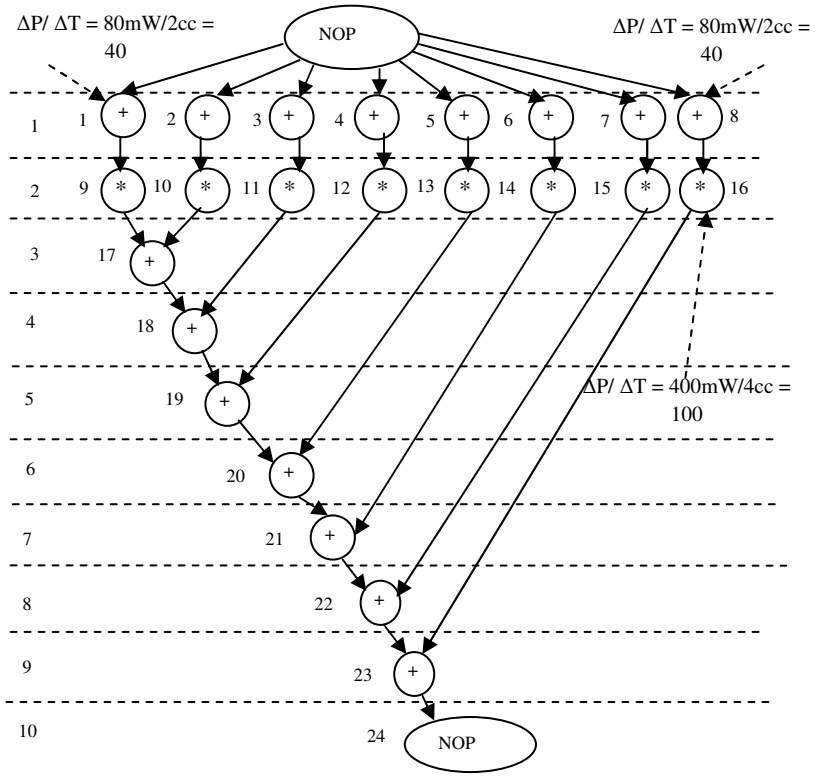


Figure13. ASAP scheduling for FIR benchmark

candidate operations are now identified for movement. Once the movable operations are identified then the iteration process begins to improve the initial scheduling solution. The iteration process is described below:

Iteration (1):

i) Movement – opn 11 (2→3) (a)

ii) Movement – opn 12 (2→3) (b)

iii) Movement – opn 13 (2→3) (c)

iv) Movement – opn 14 (2→3) (d)

v) Movement – opn 15 (2→3) (e)

vi) Movement – opn 16 (2→3) (f)

For example in equation (a) above signifies that opn 11 is one of the identified movable operations that can be moved from CS 2 into next CS 3. Now the Priority indicator (PI) for each identified movable operation is calculated using equation (4). But before the PI is calculated, the '*power fluctuation*' is determined as follows using equation (3):

Power fluctuation = [*Power consumption rate at CS (j) - Power consumption rate at CS (k)*]

– [{*Power consumption rate at CS (j) – Power consumption rate for opn (i)*} – {*Power consumption rate at CS (k) + Power consumption rate for opn (i)*}]

As shown in Figure 13, power consumption rate for adder is (20au * 4mW)/2cc = 40, while for multiplier is (100au * 4mW)/4cc = 100. (Note: From [1][2], total power consumption of resource (Ri) is: $P_{Ri} = A_{Ri} * p_c$,where A_{Ri} is the area of Ri).

Before substituting in equation (4), from Figure 13, the power fluctuation between CS (j) and CS (k) for the first case a) is calculated as follows:

= [(400/4 +400/4 + 400/4 + 400/4 +400/4 +400/4 +400/4 + 400/4) – (80/2)] – [(400/4 + 400/4 + 400/4 + 400/4 + 400/4 + 400/4 + 400/4) – (80/2 + 400/4)]

= [800 – 40] – [700 – 140] = 200.

Now substituting the value for 'Power Fluctuation' calculated above in equation (4) yields:

$PI^{opn\ 11}(2 \rightarrow 3) = 200 * Max(800, 40) * 5 = 800000$ (selected).

Similarly, calculating the 'Power Fluctuation' for each case and then finding the *Priority indicator (PI)* yields:

b) $PI^{opn\ 12}(2 \rightarrow 3) = 200 * Max(800, 40) * 5 = 800000.$

c) $PI^{opn\ 13}(2 \rightarrow 3) = 200 * Max(800, 40) * 5 = 800000.$

d) $PI^{opn\ 14}(2 \rightarrow 3) = 200 * Max(800, 40) * 5 = 800000.$

e) $PI^{opn\ 15}(2 \rightarrow 3) = 200 * Max(800, 40) * 5 = 800000.$

f) $PI^{opn\ 14}(2 \rightarrow 3) = 200 * Max(800, 40) * 5 = 800000.$

Since there is a tie between the 'PI' hence the tie is randomly broken as per the algorithm in Figure 5. The respective scheduling solution found after iteration 1 is shown in Figure 14. The cost of the scheduling solution is {(7 * 100) + (8 * 20) + (30 * 3) + (15 *3) + (23 * 5)} * 4 = 4.44 Watts. Thus we see that the cost in terms of power consumption reduces from the initial solution.

Iteration (2):

i) Movement – opn 3 (1→2) (a)

ii) Movement – opn 12 (2→3) (b)

iii) Movement – opn 13 (2→3) (c)

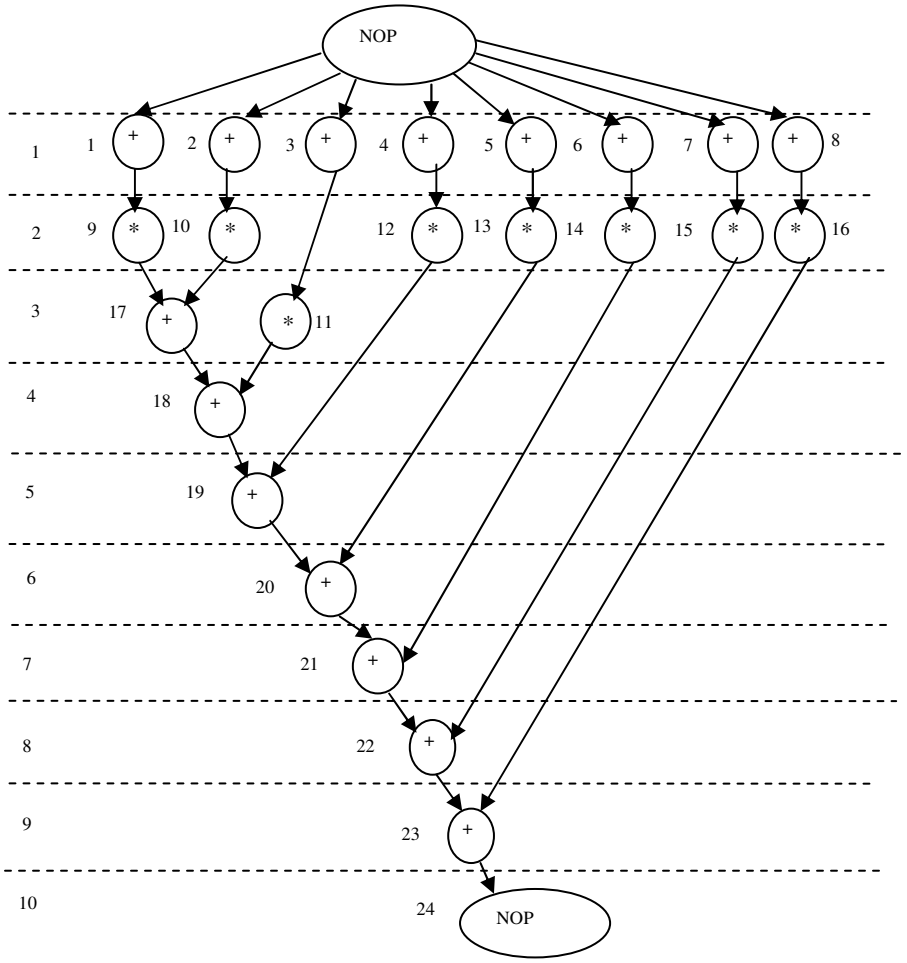


Figure14. Scheduling solution of FIR after 1st Iteration

iv) Movement – opn 14 (2→3) (d)

v) Movement – opn 15 (2→3) (e)

vi) Movement – opn 16 (2→3) (f)

Therefore the PI obtained for each operation is:

a) PI $^{opn\ 3}$ (1→2) = 168000.

b) PI $^{opn\ 12}$ (2→3) = 700000 (selected).

c) PI $^{opn\ 13}$ (2→3) =700000.

d) PI $^{opn\ 14}$(2→3) =700000.

e) PI $^{opn\ 15}$ (2→3) =700000.

f) PI $^{opn\ 16}$ (2→3) =700000.

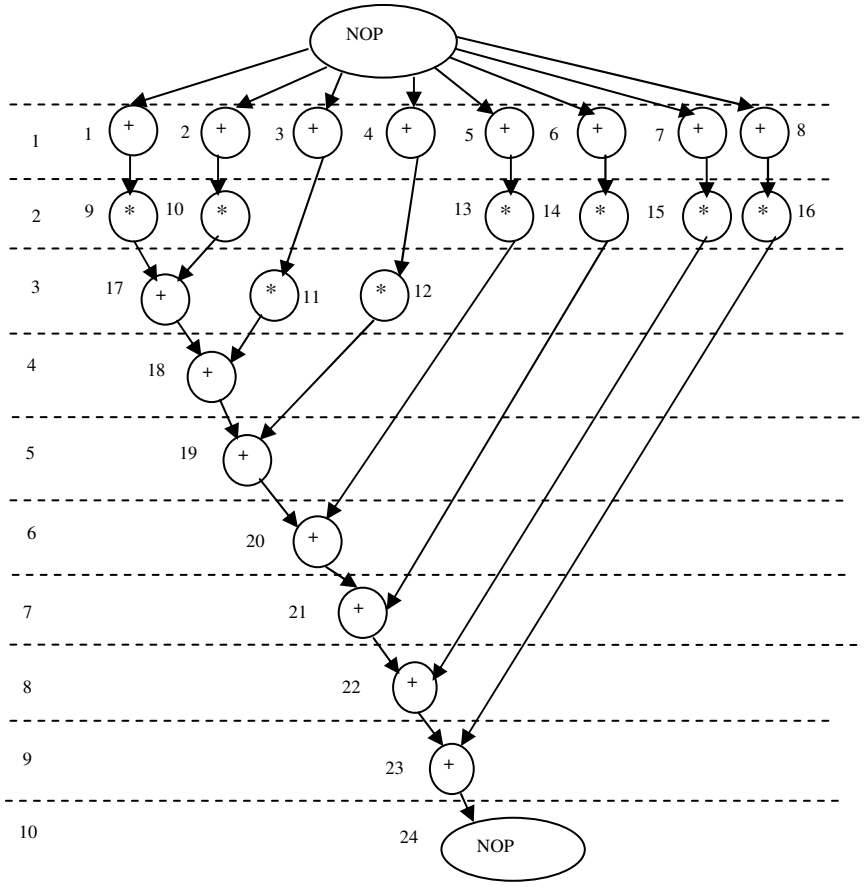


Figure15. Scheduling solution of FIR benchmark after 2nd Iteration

Since there is a tie again between the 'PI' of cases b), c), d), e) and f) hence the tie is randomly broken as per the algorithm in Figure 5. Thus operation 12 is chosen randomly for movement. The respective temporary scheduling solution found after iteration 2 is shown in Figure 15. The cost of this solution is {(6 * 100) + (8 * 20) + (28 * 3) + (14 * 3) + (24 * 5)} * 4 = 4.024 Watts. Again after this iteration a reduction in power consumption is noted.

Iteration (3):

i) Movement – opn 3 (1→2) (a)

ii) Movement – opn 4 (1→2) (b)

iii) Movement – opn 12 (3→4) (c)

iv) Movement – opn 13 (2→3) (d)

v) Movement – opn 14 (2→3) (e)

vi) Movement – opn 15 (2→3) (f)

vii) Movement – opn 16 (2→3) (g)

Therefore the PI obtained for each operation is:

a) PI $^{opn\ 3}$ (1→2) = 144000.

b) PI $^{opn\ 4}$ (1→2) = 144000.

c) PI $^{opn\ 12}$ (3→4) = 240000.

d) PI $^{opn\ 13}$ (2→3) = 600000 (selected).

e) PI $^{opn\ 14}$ (2→3) =600000.

e) PI $^{opn\ 15}$ (2→3) =600000.

g) PI $^{opn\ 16}$ (2→3) =600000.

According to the algorithm, the highest 'PI' is selected for movement is d). The cost of this respective scheduling solution is {(5 * 100) + (8 * 20) + (26 * 3) + (13 * 3) + (25 * 5)} * 4 = 3.60 Watts. Again after this iteration a reduction in power consumption is noted compared to the previous scheduling solution. Since the algorithm will not be terminated until # of iterations = # of nodes in the DFG, hence the local optimal solution found does not restrict the algorithm from stopping. This above iteration continues until the algorithm reaches iteration 24 (Since the maximum number of nodes present in this DFG is 24).

Iteration (24):

i) Movement – opn 4 (2→3) (a)

ii) Movement – opn 5 (1→2) (b)

iii) Movement – opn 6 (1→2) (c)

iv) Movement – opn 7 (1→2) (d)

v) Movement – opn 8 (1→2) (e)

Therefore the 'PI' obtained for each operation is:

a) PI $^{opn\ 4}$ (2→3) = 67200.

b) PI $^{opn\ 5}$ (1→2) = 67200(selected).

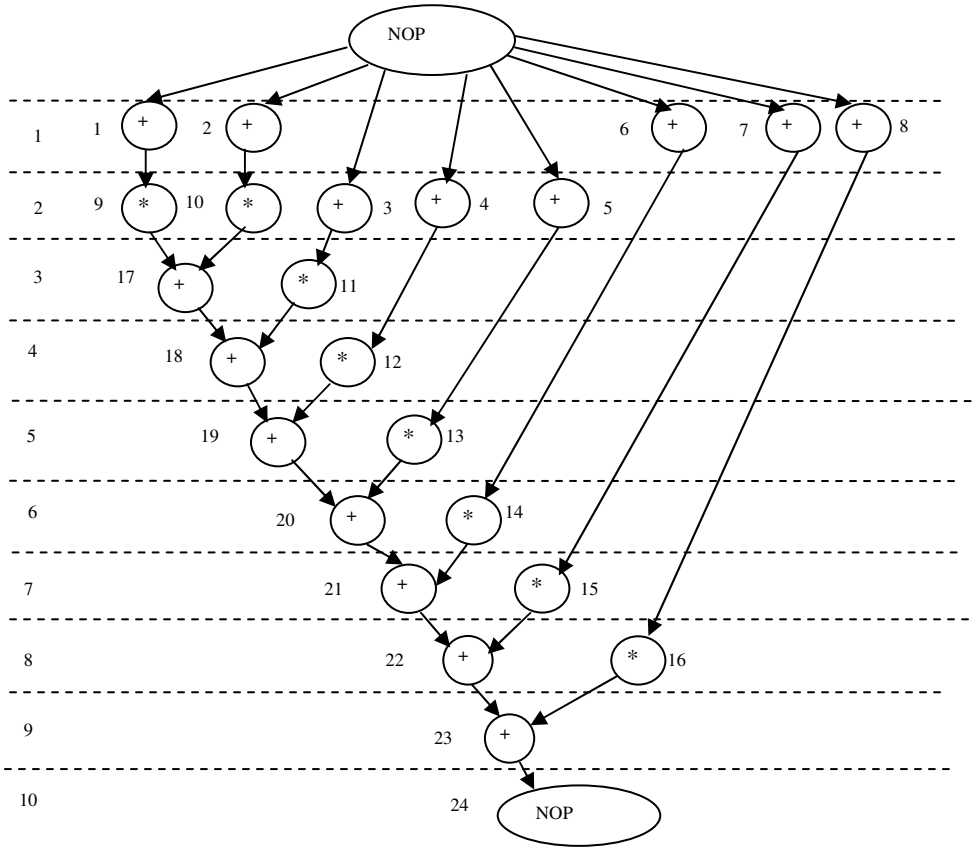


Figure16. Scheduling solution of FIR benchmark after 24th Iteration

c) PI $^{opn\ 6}$ (1→2) = 67200.

d) PI $^{opn\ 7}$ (1→2) = 67200.

e) PI $^{opn\ 8}$ (1→2) = 67200.

According to the algorithm, randomly operation 5 is selected for movement as shown below in Figure 16. The respective scheduling solution found after iteration 24 is shown in Figure 16. The cost of this respective scheduling solution is {(2 * 100) + (5 * 20) + (14 * 3) + (7 * 3) + (22 * 5)} * 4 = 1.89 watts. Hence iteration 24 found the local optimal solution to the scheduling problem. The final reduction in cost in terms of power consumption obtained compared to the initial solution (in Figure 5) is 4.87 Watts – 1.89 Watts = 2.98 Watts *(Since according to the algorithm, the iteration continues until the iteration # = maximum node #, hence the iteration continues till iteration 24 to eventually find an optimal solution).*

Chapter 7

Experimental Results for the Benchmarks

7.1 Implementation Details

A number of well known standard high level synthesis benchmarks were drawn from the literature for verification and comparison. The benchmarks adopted were IIR Chebyshev Digital Filter, Finite Impulse Response (FIR) [17], Discrete Wavelet Transformation (DWT) [17] [21], IIR Digital Butterworth filter, MPEG Motion Vectors (MMV) [17] and Band Pass Filter (BPF) [17]. The parameters for test chosen for the experiment were a) latency of the final scheduling solution (in clock cycles), b) the final allocated resource combination (FU's, mux, demux and registers) found, c) the cost of the final solution found in terms of power consumption, d) power reduction achieved and e) the runtime of the whole scheduling process.

The proposed integrated design space exploration approach has been implemented in C language and run on AMD Athlon 64 Processor with 3GB RAM with processor frequency 1.6

GHz. Previous approaches [6], [7], [8], [9], [13], [14] are all based on Genetic Algorithm (GA) which has exponential time complexity unlike the proposed approach. Approach [6] has already proved to be superior to other GA based approaches in terms of speed and quality. Hence [6] was selected for comparison with proposed approach. In order to perform a qualitative assessment of the proposed approach, the proposed approach has been compared with a GA based approach [6]. Moreover, the GA based approach [6] chosen from the literature for comparison is a well known design space exploration approach for scheduling and module selection. By using the exact parameters (such as terminating condition, user specified choice of constraints, genetic operators etc) as mentioned in [6], the proposed approach was compared. The user preference weight of each constraint for the DSE approach [6] was kept at 0.5 during the experiment, signifying that both the constraints (power consumption and latency) were given equal priority. In the comparison of the proposed approach with [6], the parameters of comparison chosen were:
a) The quality of the final solution found measured in terms of Quality Cost Factor (Q-metric). Q-metric is a metric which determines the quality of final solution found by both approaches. The metric is a combination of latency and power consumption of the scheduling solution which is given by equation (7) below:

$$Q-metric = W1 \cdot \frac{L}{L_{\max}} + W2 \cdot \frac{P}{P_{\max}} \tag{7}$$

Where, W1 and W2 are the weightage of the operating constraints for latency and power consumption (*Note: 0<=W1<=1 and 0<=W2<=1*). In our experiment, W1 = W2 = 0.5 has been kept, since equal priority was given to both latency of the final solution and the power consumption of the scheduling solution. 'L' and 'P' are the latency and power consumption of the solution found respectively. 'L_{max}' and 'P_{max}' are the values of maximum latency (found by

using minimum FU's) and maximum power consumption (using maximum FU's) respectively. Equation (7) has been divided with maximum values of latency and power respectively in order to obtain normalized values for each.

The value of total power consumption (P) is calculated using equation (8) from current literature [1, 3, 22, 23, 24, 25, 26, 27, 28] as shown below:

$$P = \sum A(R) \cdot p_c \tag{8}$$

Where p_c' is the power consumed per au at a particular frequency of operation; 'A(R)' is the area of the hardware resources including FU's (A_{Ri}), mux (A_{mux}), demux (A_{demux}) and registers (A_{reg}) and is calculated as shown in equation (9):

$$A(R) = A_{Ri} + A_{mux} + A_{demux} + A_{reg} \tag{9}$$

Where $A_{Ri} = N_{Ri}. K_{Ri}$; Where N_{Ri} represents the number of resource 'Ri' and 'K_{Ri}' represents the area occupied per unit resource Ri.

b) The actual runtime taken by both the scheduling approaches. The speed of the scheduling process was chosen as a parameter for comparison because in this current generation of Electronic Design Automation (EDA), reducing the design time helps in rapid marketing of the final end user product. The above metric was proposed for comparison since the quality of a solution cannot be solely determined from the latency expenditure or the power consumption, but rather a combination of both together.

7.2 Results of the proposed approach on DSP benchmarks

The discussion of the results obtained through the proposed approach is shown in Table I. The complexity of the design (benchmarks tested) has been taken into account with respect to the size of the application (no: of nodes). Wires and busses have not been considered in the proposed

Table I. Experimental Results of the Proposed DSE approach for the DSP Benchmarks

DSP Benchmarks [29] [30]	**Experimental Parameters** (Note: cc = clock cycles)						
	Resource combinations (Functional Units/Mux/Demux/ Registers)		**Latency (cc)**	**Initial Cost in terms of Power consumption (using eqn. 6)**	**Final Cost through proposed PI method (Power consumption)**	**% Reduction in Power consumption (cost reduction) through proposed PI method**	**Runtime of proposed PI method (secs)**
	Initial Solution	**Proposed PI method**	**Proposed PI method**				
Discrete Wavelet Transformation (DWT)	5(*), 4(+) 18 (mux), 9 (demux), 15 (registers)	2(*), 2(+) 8 (mux), 4(demux), 14 (registers)	28 cc	2.94 Watts	1.38 Watts	53.06 %	3.18 secs
Band Pass Filter (BPF)	4(*), 3(+/-) 14 (mux), 7 (demux), 20 (registers)	2(*), 3(+/-) 10 (mux), 5(demux), 19 (registers)	26 cc	2.49 Watts	1.60 Watts	35.74 %	1.38 secs
Finite Impulse Response (FIR)	8(*), 8(+) 32 (mux), 16 (demux), 23 (registers)	2(*), 5(+) 14 (mux), 7(demux), 22 (registers)	20 cc	4.87 Watts	1.89 Watts	61.19 %	5.63 secs
IIR Digital Butterworth Filter	5(*), 1(+/-) 12 (mux), 6 (demux), 14 (registers)	2(*), 1(+/-) 6 (mux), 3 (demux), 11 (registers)	12 cc	2.57 Watts	1.20 Watts	53.30 %	2.08 secs
IIR Digital Chebyshev Filter	5(*), 2(+) 14(mux), 7 (demux), 16 (registers)	3(*), 2(+) 10 (mux), 5 (demux), 16 (registers)	8 cc	2.60 Watts	1.86 Watts	28.46 %	1.56 secs
MPEG Motion Vectors (MMV)	14(*), 5(+) 38(mux), 19 (demux), 42 (registers)	5(*), 5(+) 20(mux), 10 (demux), 33 (registers)	10 cc	7.52 Watts	3.42 Watts	54.52 %	1.95 secs

approach, although interconnect units and storage elements besides FU have been considered. The optimization (minimization) obtained for the final resource combination (in terms of FU's, mux, demux and registers) as noted from the results for all DSP benchmarks such as DWT, BPF, FIR, Digital Butterworth filter, Chebyshev filter and MPEG are definitely noteworthy. For example, in case of DWT benchmark, the final resource combination found is 2(*), 2(+), 8

(mux), 4(demux) and 14 (register) compared to initial resource combination 5(*), 4(+), 18 (mux), 9(demux) and 15(register). Further, a drastic reduction in final cost (in terms of power consumption) is obtained through the proposed approach as reflected in Table I. Significant reduction in final cost of as high as 61.19 % and 54.52 % were noted for FIR and MPEG benchmarks respectively. For others benchmarks such as DWT, IIR Butterworth filter and BPF the proposed DSE also yielded significant minimization in final cost in terms of power consumption.

7.3 Results and Analysis of the comparison with recent Exploration approach

The implementation runtime of the proposed design space exploration and its comparison with a recent DSE approach [6] is illustrated in Table II. Table II also reflects the comparison of the final solution found by both the approaches. Since performing ASAP scheduling first as an initial schedule solution imposes a latency restriction on the final solution, hence it becomes very mandatory for the proposed DSE approach to arrive at the final solution keeping the latency constraint under consideration. Similarly, the GA based approach [6] was kept under exact same latency constraint limitation with maximum limitation on FU's, during the comparative analysis. Maximum limitation (minimum area) constraint has been imposed on [6] along with the same latency constraint (determined from ASAP schedule) as the proposed approach because the objective of the proposed approach is to find the final solution with minimum area overhead (or power consumption) under ASAP determined latency constraint. Therefore, for the sake of performing a qualitative estimation of the proposed DSE approach, two major parameters of comparison viz. a) Q-metric and b) runtime were chosen. Q-metric provides a comprehensive measurement of the quality of the final solution found (*Note: Q-metric is measured in terms of*

latency and power consumption of the hardware resources for the final solution) while runtime provides a solid summary of the speed of the exploration process.

Table II also highlights the percentage improvement in the quality of the final solution found by the proposed approach and the percentage reduction in runtime obtained by the proposed approach. For example, in case of small benchmark such as IIR digital Butterworth filter the improvement obtained in the quality of final solution by the proposed approach compared to [6] is 14.28 %. Furthermore for medium complexity benchmarks such as DWT and FIR, the improvement in the quality of final solution achieved is 9.03 % and 14.73 % respectively. Moreover, for high complexity benchmarks such as BPF and MPEG, the improvement in the quality of final solution achieved compared to [6] is 10 % and 14.28 % respectively. The results obtained indicated that the proposed approach has been capable to find better quality solutions for all benchmarks as compared to [6]. Thus application of proposed DSE

Table II. Results of comparison between proposed approach and [6] for DSP Benchmarks

DSP Benchmarks [29][30]	**Experimental Parameters for Comparison**							
	Quality Cost Factor (Q-metric)		**% Improvement in quality of final solution**	**%Average Improvement of quality final solution**	**Runtime (seconds)**		**% Reduction in Runtime using PI method**	**% Average reduction in runtime obtained**
	[6]	**Proposed PI method**			**[6]**	**Proposed PI method**		
Discrete Wavelet Transformation (DWT)	0.64	0.58	9.03 %	**11.62 %**	7.53 secs	3.18 secs	57.76 %	**59.10 %**
Band Pass Filter (BPF)	0.60	0.54	10.00 %		13.96 secs	1.38 secs	90.11 %	
Finite Impulse Response (FIR)	0.48	0.41	14.73 %		11.04 secs	5.63 secs	49 %	
IIR Digital Butterworth Filter	0.56	0.48	14.28 %		3.04 secs	2.08 secs	31.57 %	
IIR Digital Chebyshev Filter	0.58	0.53	7.41 %		2.69 secs	1.56 secs	42 %	
MPEG Motion Vectors (MMV)	0.35	0.30	14.28 %		12.32 secs	1.95 secs	84.17 %	

on DSP benchmarks has lead to an average percentage improvement of 11.62 % in quality of final solution compared to [6], which is quite significant. A comparison of the effective time taken to explore the optimal/near-optimal solution has been performed with the help of implementation runtime for both the DSE approaches. As revealed in Table II, for all benchmarks such as DWT, BPF, FIR, MPEG and IIR digital filter, the reduction in runtime has been imposing. The average reduction in runtime for all benchmarks is around 60 % as shown in Table II. Therefore it can be clearly seen that the proposed DSE has been capable to find better quality solutions for all benchmarks at the expense of approximately half runtime as [6]. In particular, for benchmarks such as BPF and MPEG, the reduction in runtime obtained is 90.11 % and 84.17 % respectively, which is definitely notable. Similarly for IIR digital filter and FIR benchmarks the reduction obtained is also significant ranging from 42 % to 49 % respectively. Hence, in both the proportions of comparative analysis a) quality of final solution and b) exploration runtime, the proposed approach has been able to perform better compared to scheduling approach [6].

In Figure 17, the variation of reduction in runtime and the improvement in quality of final solution obtained compared to [6] with the complexity of the benchmarks is shown. As clearly visible from Figure 17, a general increase in trendline in the percentage improvement of the quality of final solution with the increase in complexity of the benchmarks can be noted. In particular for medium and largely complex benchmarks such as FIR and MPEG, the value of improvement obtained in final quality of solution is seen to be quite significant. Furthermore, as seen in Figure 18, a general increase in trendline in the percentage reduction of runtime with increase in complexity of benchmarks can be noted. For complex benchmarks such as BPF and MPEG, the reduction in exploration runtime is seen to be significantly large, which proves that

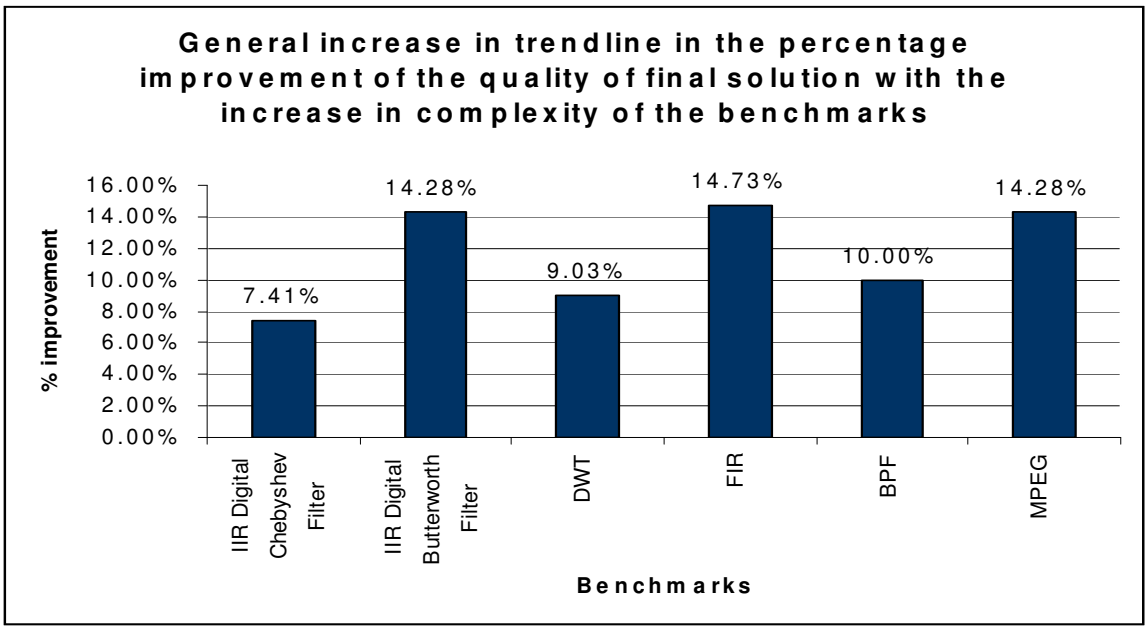


Figure17. A general increase in trendline in the % improvement of quality of final solution with increase in complexity of the benchmarks

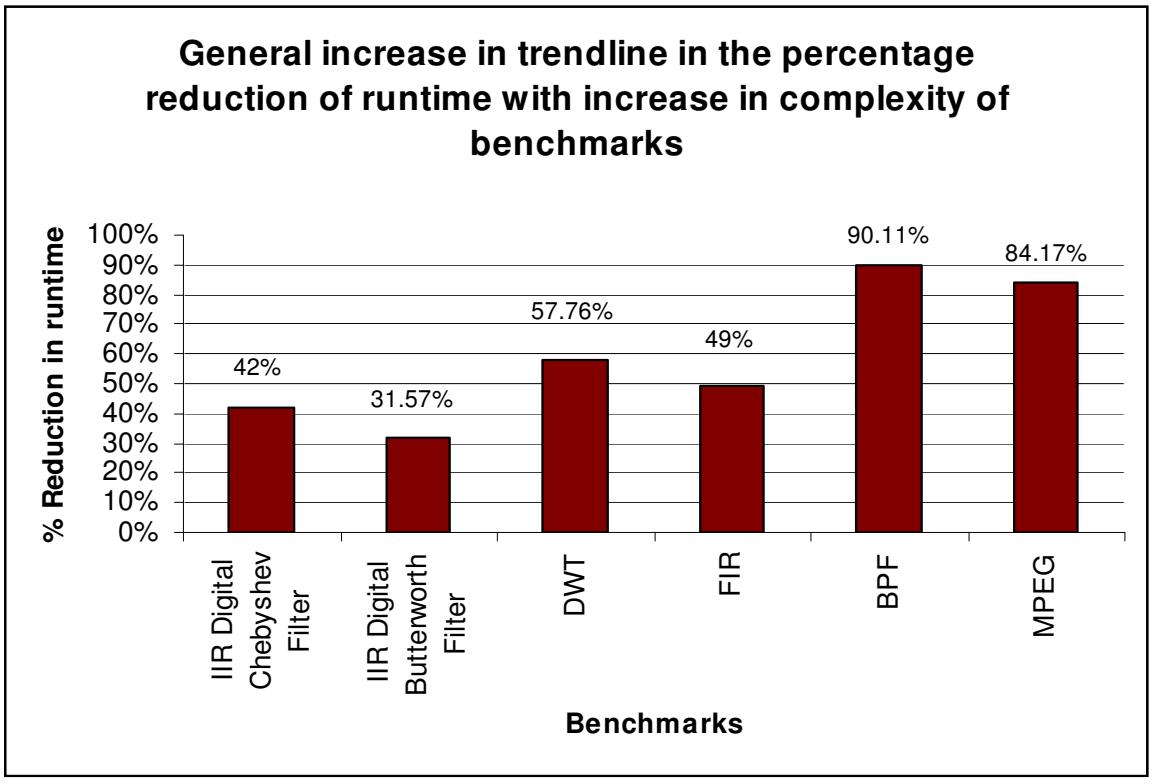


Figure18. A general increase in trendline in the percentage reduction of runtime with increase in complexity of benchmarks

the proposed DSE also has good scalability property. The % improvement in quality of final

solution and reduction in implementation runtime for the proposed approach compared to [6] is therefore seen to be equally improved for small, medium and large complexity benchmarks. Moreover a careful observation in Figure 17 and Figure 18 also reveals that for medium and high complexity benchmarks such as FIR, BPF and MPEG, the proposed approach is seen to perform better both in terms of providing better quality solution (an average improvement over 10 %) and reducing the time taken (an average reduction over 50 %) to yield the final optimal solution. As verified through the results obtained the proposed approach handles small, medium and large size applications in short runtime which dictates the ability of the proposed approach to handle growing size of the problem in a reasonable time. For example, small size and less complex benchmark such as IIR Digital filter the quality improvement and runtime reduction was 10 % and 31.57 %. For medium size problems, such as FIR, the quality improvement and runtime reduction was 14.73 % and 49 %. Finally, for large size and highly complex benchmark, MPEG the quality improvement and runtime reduction was 14.28 % and 84.17 %. This proves that the proposed approach has the ability to handle growing size of the problem in a reasonable time. Hence the approach is scalable. The implementation results for the DWT benchmark have been also shown in Figure 19 and Figure 20 respectively.

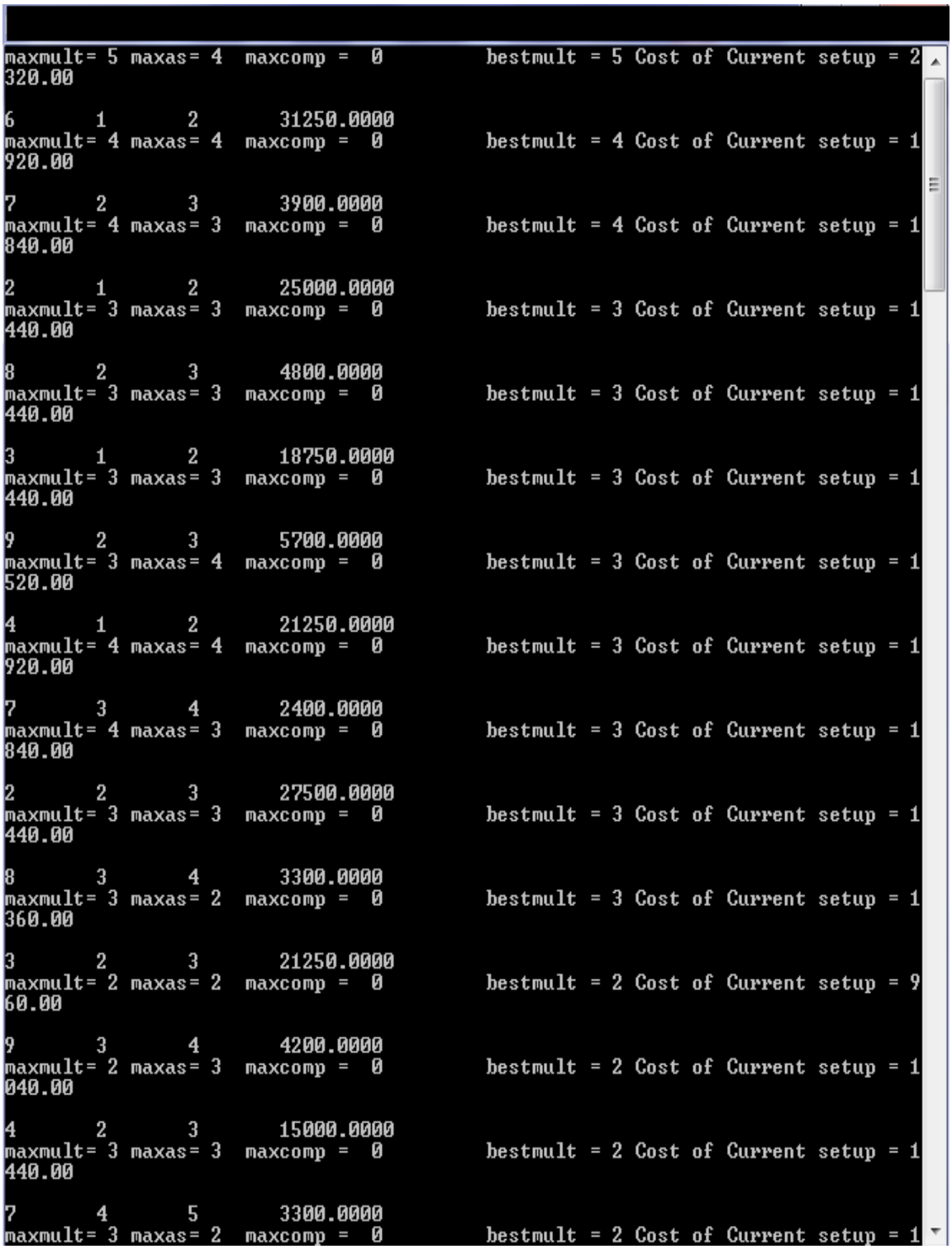


Figure19. Portion of the implementation result for DWT showing the iterations of the proposed method.

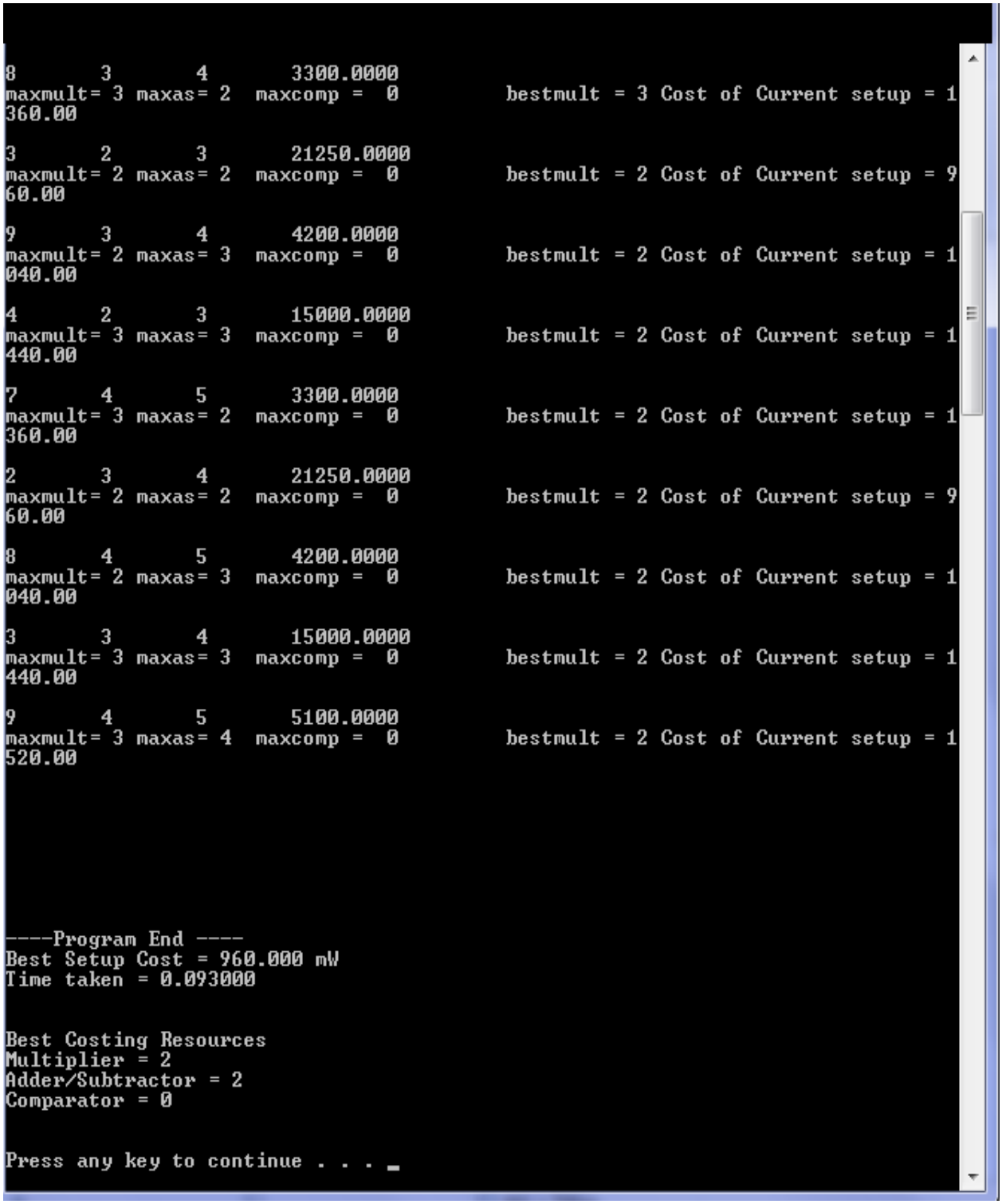


Figure20. Portion of the implementation result for DWT benchmark showing the final result obtained using the proposed approach.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This thesis introduced a novel power efficient rapid integrated design space exploration approach for scheduling and module selection in high level synthesis. The proposed DSE approach reduced the total power consumption of the resources at the expense of minimal latency expenditure. Thus the introduced DSE was capable of minimizing the power consumption and hardware area under strict minimal latency constraints. The DSE approach using iterative technique based on '*Priority indicator (PI method)*' function selects the operation with the highest PI from the available list of movable operations for further optimization. This scheduling process is repeated until an optimal solution is found. The second aspect of this thesis is the introduction of a novel topology called '*Intersect Matrix*' with its associated algorithm

used for checking the precedence violation between operations during the scheduling process. This topology is easy to implement because of its minimal complexity and straight forward characteristic.

The proposed approach was qualitatively compared with [6] in terms of quality of final solution and exploration runtime. A new metric called '*Quality Cost Factor (Q-metric)*' was proposed to compare the quality of solutions yielded by both approaches. The above metric was proposed since the quality of a solution cannot solely be determined from the latency expenditure or the occupied hardware area, but rather a combination of both. Results of comparison by Q-metric indicate that the proposed approach was able to provide ≈ 12 % improvement in the quality of final solution compared to [6]. Further, comparison of exploration runtime for both approaches indicates that the proposed approach was able to find an average reduction of 59.10 % in runtime compared to [6]. Therefore the proposed DSE found a better solution in half of the exploration runtimes compared to [6]. Hence the approach presented in this thesis is a novel versatile design space exploration approach that is rapid, power efficient in nature, and highly useful for data path synthesis in Electronic System Level (ESL) design. The approach has the ability to escape from the local optimal solution and therefore a tendency to reach global optimal solution.

8.2 Scope of Future Work

Further, global optimal solutions were found for almost all benchmarks, which dictate the capability of the proposed approach to escape local optima and find global optimal solutions. Although efficient in finding global optimal solutions in most cases, there may be few cases

where the proposed method is unable to find the global optima. This usually results in local optimal solutions where further scope for optimization is possible. Therefore, there is an aspect related to the suitable terminating condition of the proposed approach where further improvements can be made. Selecting an accurate terminating condition maintains the right balance between obtaining a high precision solution and minimizing the exploration runtime. Hence, future works are geared towards experimenting with various terminating conditions to evaluate the tradeoffs between solution accuracy and exploration runtime. This would help in selecting an optimal termination condition for the proposed approach to further improve the quality of solution.

Refereed Publications

Patents/Inventions

1. Reza Sedaghat, Pallabi Sarkar, Anirban Sengupta, “Power Efficient Rapid Scheduling Algorithm in High Level Synthesis for Computation-intensive Applications using PI Method”, Invention filed to MARS Innovation, Govt. of Canada, November 2010.

Refereed Journals

2. Pallabi Sarkar, Reza Sedaghat, Anirban Sengupta, “Power Efficient Rapid Design Space Exploration of Integrated Scheduling and Module Selection in High Level Synthesis” ‘Journal of Microelectronics Reliability’, Elsevier, 2010, Ref. No.: MR-D-10-00484.
3. Anirban Sengupta, Reza Sedaghat, Pallabi Sarkar, “Rapid Exploration of Integrated Scheduling and Module Selection in High Level synthesis for Application Specific Processor Design”, ‘Journal of Microprocessor and Microsystems’, Elsevier, 2010.

Refereed Conferences

4. Pallabi Sarkar, Anirban Sengupta, Reza Sedaghat, “Power Efficient Rapid Scheduling Approach in High Level Synthesis using PI Method”, IEEE/ACM Design Automation Conference (DAC), 2011, Submitted.
5. Pallabi Sarkar, Reza Sedaghat, Anirban Sengupta, “Priority Function Based Power Efficient Rapid Design Space Exploration of Scheduling and Module Selection in High Level Synthesis”, 24th IEEE Canadian Conference on Electrical and Computer Engineering, 2011, Submitted.

References

[1] Anirban Sengupta, Reza Sedaghat, Zhipeng Zeng, “Rapid Design Space Exploration for multi parametric optimization of VLSI designs”, In Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), Paris, pages: 3164-3167, 2010.

[2] Keinert, J., Streubuhr, M., Schlichter, T., Falk, J., Gladigau, J., Haubelt, C., and Teich, J. “SYSTEMCODESIGNER—An automatic ESL synthesis approach by design space exploration and behavioral synthesis for streaming application”, ACM Transactions on Design Automation of Electronic Systems (TODAES), January 2009, vol.14, issue: 1, Article 1.

[3] Anirban Sengupta, Reza Sedaghat, Zhipeng Zeng, “Rapid Design Space Exploration by Hybrid Fuzzy Search Approach for Optimal Architecture determination of Multi Objective Computing Systems”, Journal of Microelectronics Reliability, September 2010, Elsevier, doi:10.1016j.

[4] Philippe Grosse, Yves Durand, Paul Feautrier, "Methods for power optimization in SOC-based data flow systems", ACM Transactions on Design Automation of Electronic Systems (TODAES), 2009, vol. 14, issue 3, Article no.: 38.

[5] J. C. Gallagher, S. Vigraham, and G. Kramer “A family of compact genetic algorithms for intrinsic evolvable hardware,” IEEE Trans. Evolutionary Computation., April 2004, vol. 8, no. 2, pages: 111–126.

[6] Vyas Krishnan and Srinivas Katkoori, “A Genetic Algorithm for the Design Space Exploration of Datapaths During High-Level Synthesis”, IEEE Transactions on Evolutionary Computation, June 2006, vol.10, no.3, pages: 213-229.

[7] E. Torbey and J. Knight, “High-level synthesis of digital circuits using genetic algorithms,” in Proc. Int. Conf. Evol. Comput., May 1998, pages: 224–229.

[8] E. Torbey and J. Knight, “Performing scheduling and storage optimization simultaneously using genetic algorithms,” in Proc. IEEE Midwest Symp. Circuits Systems, 1998, pages: 284–287.

[9] Giuseppe Ascia, Vincenzo Catania, Alessandro G. Di Nuovo, Maurizio Palesi, Davide Patti, “Efficient design space exploration for application specific systems-on-a-chip” Journal of Systems Architecture, Elsevier, 2007, vol. 53, issue 10, pages: 733–750.

[10] A.C.Williams, A.D.Brown and M.Zwolinski, “Simultaneous optimisation of dynamic power, area and delay in behavioural synthesis”, IEE Proc.-Comput. Digit. Tech, November 2000, vol. 147, no. 6, pages: 383-390.

[11] I. Das. “A preference ordering among various Pareto optimal alternatives”. Structural and Multidisciplinary Optimization, Aug. 1999, vol.18, issue: 1, pages: 30–35.

[12] Xuejie Zhang and Kam W. Ng, “A review of high-level synthesis for dynamically reconfigurable FPGAs”, Microprocessors and Microsystems, Elsevier, August 2000, vol. 24, issue 4, pages 199-211,1.

[13] C. Mandal, P. P. Chakrabarti, and S. Ghose, “GABIND: A GA approach to allocation and binding for the high-level synthesis of data paths,” IEEE Transaction on VLSI, Oct. 2000, vol. 8, no. 5, pages: 747–750.

[14] M. J. M. Heijlingers, L. J. M. Cluitmans, and J. A. G. Jess, “High-level synthesis scheduling and allocation using genetic algorithms,” in Proc. Asia South Pacific Design Automation Conf., 1995, pages: 61–66.

[15] M. K. Dhodhi, F. H. Hielscher, R. H. Storer, and J. Bhasker, "Datapath synthesis using a problem-space genetic algorithm," IEEE Trans.Comput.-Aided Des., 1995, vol. 14, pages: 934–944.

[16] G. De Micheli, "Synthesis and Optimization of Digital Circuits". New York: McGraw-Hill, 1994.

[17] Saraju P. Mohanty, Nagarajan Ranganathan, Elias Kougianos and Priyadarsan Patra, "Low-Power High-Level Synthesis for Nanoscale CMOS Circuits" Chapter- High-Level Synthesis Fundamentals, Springer US, 2008.

[18] D. Gajski, N. Dutt, A.Wu, and S. Lin, High Level Synthesis: "Introduction to Chip and System Design". Norwell, MA: Kluwer, 1992.

[19] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASICs," IEEE Trans. Comput.-Aided Des., 1989, vol. 8, no.6, pages: 661–679.

[20] Christian Haubelt, Jurgen Teich,"Accelerating Design Space Exploration Using Pareto-Front Arithmetic's", In Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC'03), Japan, 2003, pages: 525-531.

[21] Jain, R., Panda, P.R. "An efficient pipelined VLSI architecture for lifting-based 2d-discrete wavelet transform", in Proceedings of the International Symposium on Circuits and Systems (ISCAS), 2007, pages: 1377– 1380.

[22] Zhipeng Zeng, Reza Sedaghat, Anirban Sengupta, "A Framework for Fast Design Space Exploration using Fuzzy search for VLSI Computing Architectures", In Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Paris, 2010, pages: 3176-3179.

[23] Anirban Sengupta, Reza Sedaghat, Zhipeng Zeng, "A High Level Synthesis design flow with a novel approach for Efficient Design Space Exploration in case of multi parametric

optimization objective", Journal of Microelectronics Reliability, Elsevier, 2010, vol. 50, issue 3, pages 424-437.

[24] Anirban Sengupta, Reza Sedaghat, Zhipeng Zeng, "Hardware Efficient Design of speed optimized Power stringent Application Specific Processor", In Proceedings of IEEE 21st International Conference on Microelectronics (ICM), 2009, pages: 167-170.

[25] Zhipeng Zeng, Reza Sedaghat, Anirban Sengupta, "A Novel Framework of Optimizing Modular Computing Architecture for multi objective VLSI designs", In Proceedings of IEEE 21st International Conference on Microelectronics (ICM), 2009, pages: 322-325.

[26] Summit Sehgal, Reza Sedaghat, Anirban Sengupta, Zhipeng Zeng, "Multi Parametric Optimized Architectural Synthesis of an Application Specific Processor", IEEE 14th International CSI Computer Conference, 2009, pages: 89-94.

[27] Anirban Sengupta, Reza Sedaghat, "Rapid Exploration of Power-Delay Tradeoffs using Hybrid Priority Factor and Fuzzy Search", Accepted for Publication in Proceedings of 22nd IEEE International Conference on Microelectronics (ICM), Cairo, Egypt, 2010, pp.355-358.

[28] Anirban Sengupta, Reza Sedaghat, "Integrated Scheduling, Allocation and Binding in High Level Synthesis using Multi Structure Genetic Algorithm based Design Space Exploration System", Accepted for Publication in Proceedings of 12th IEEE International Symposium on Quality Electronic Design (ISQED 2011), California, USA, March 2011.

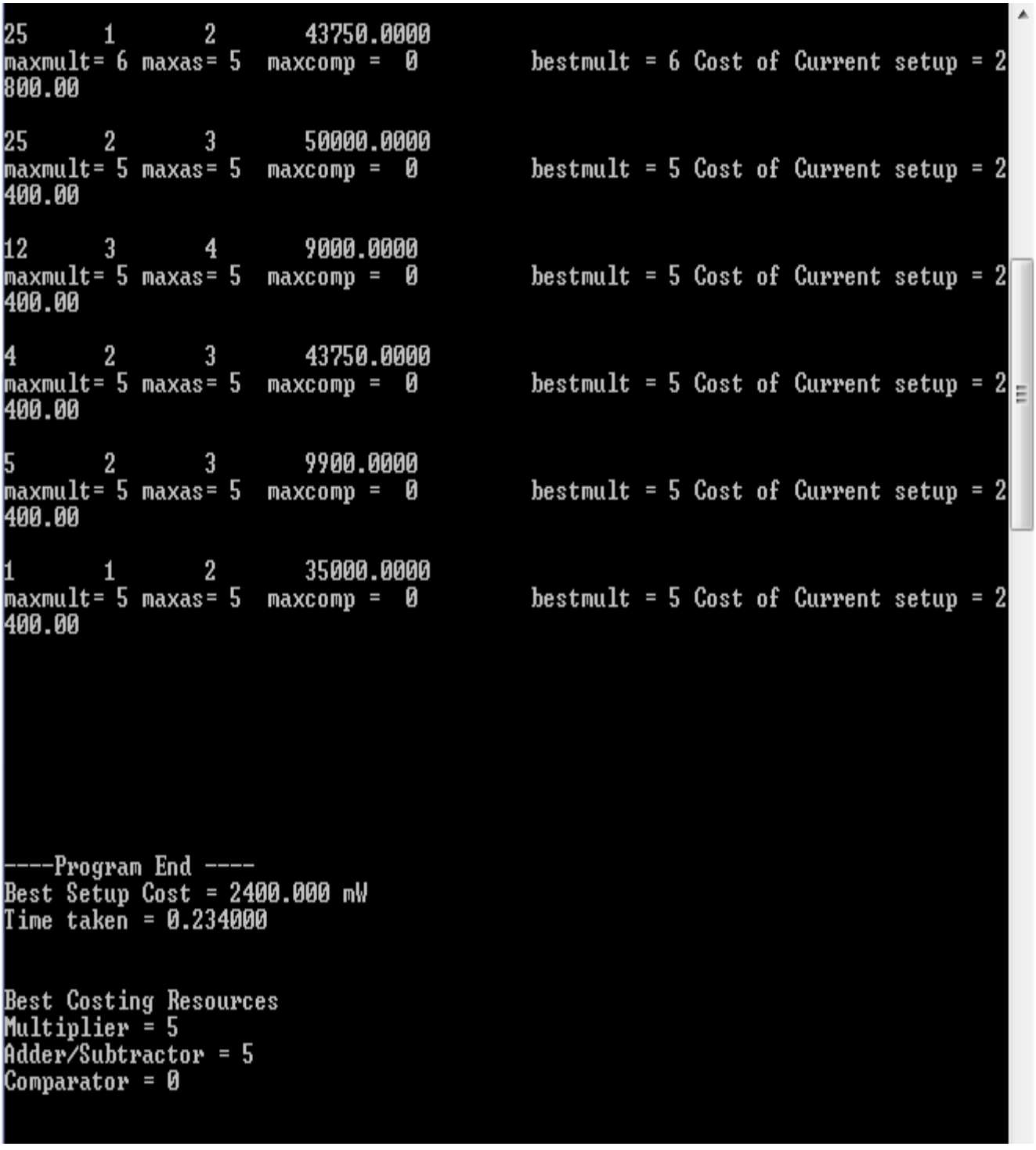
[29] Express: High-Level Synthesis Benchmarks. http://express.ece.ucsb.edu/benchmark/

[30] Express Benchmark Suite, http://express.ece.ucsb.edu/benchmark/ (From University of California, Santa Barbara).

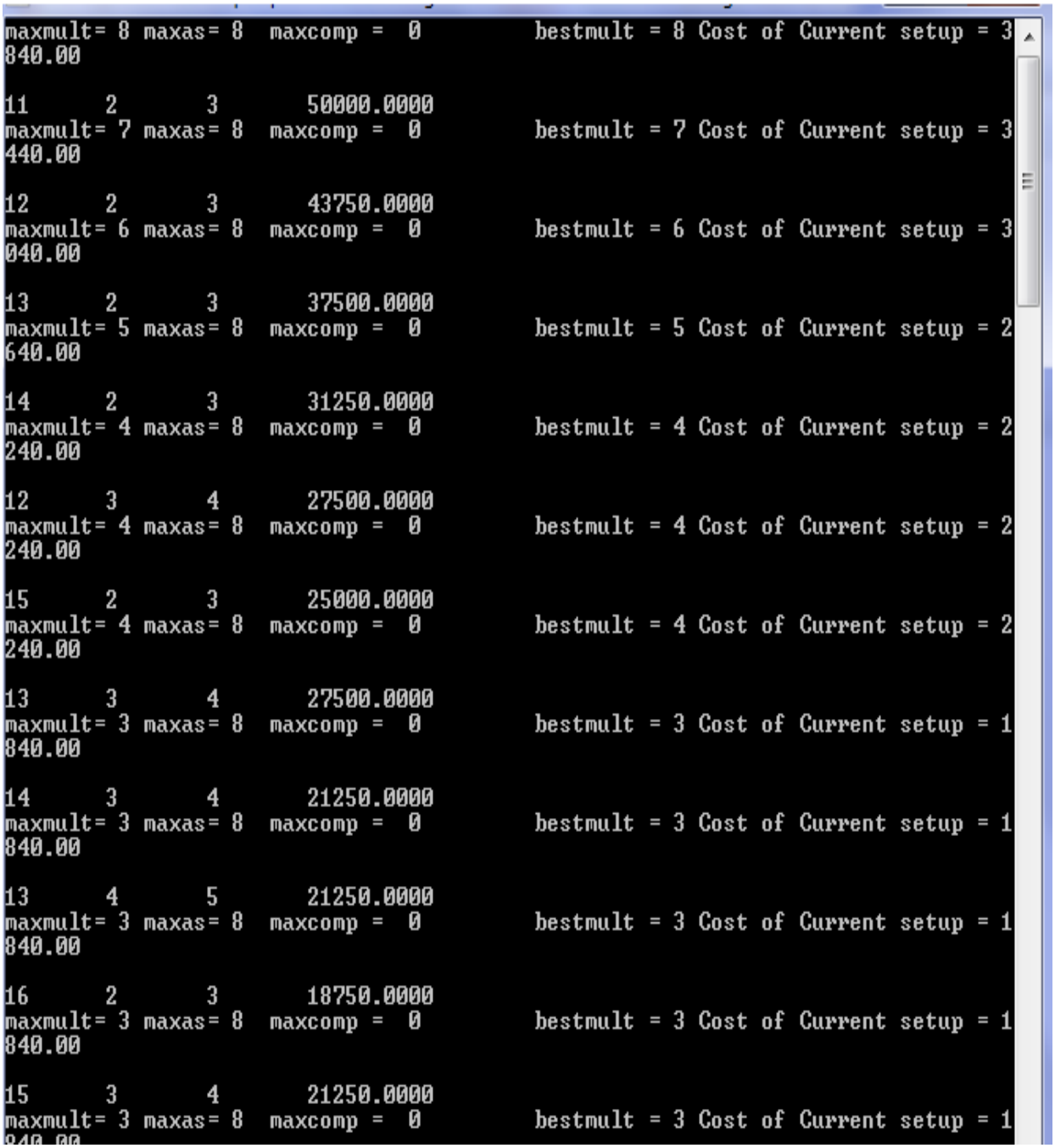
[31] McFarland, M.C. Parker, A.C. Camposano, R. "The high-level synthesis of digital systems", Proceedings of the IEEE, Feb 1990, Volume: 78, Issue: 2, page(s): 301-318.

APPENDIX

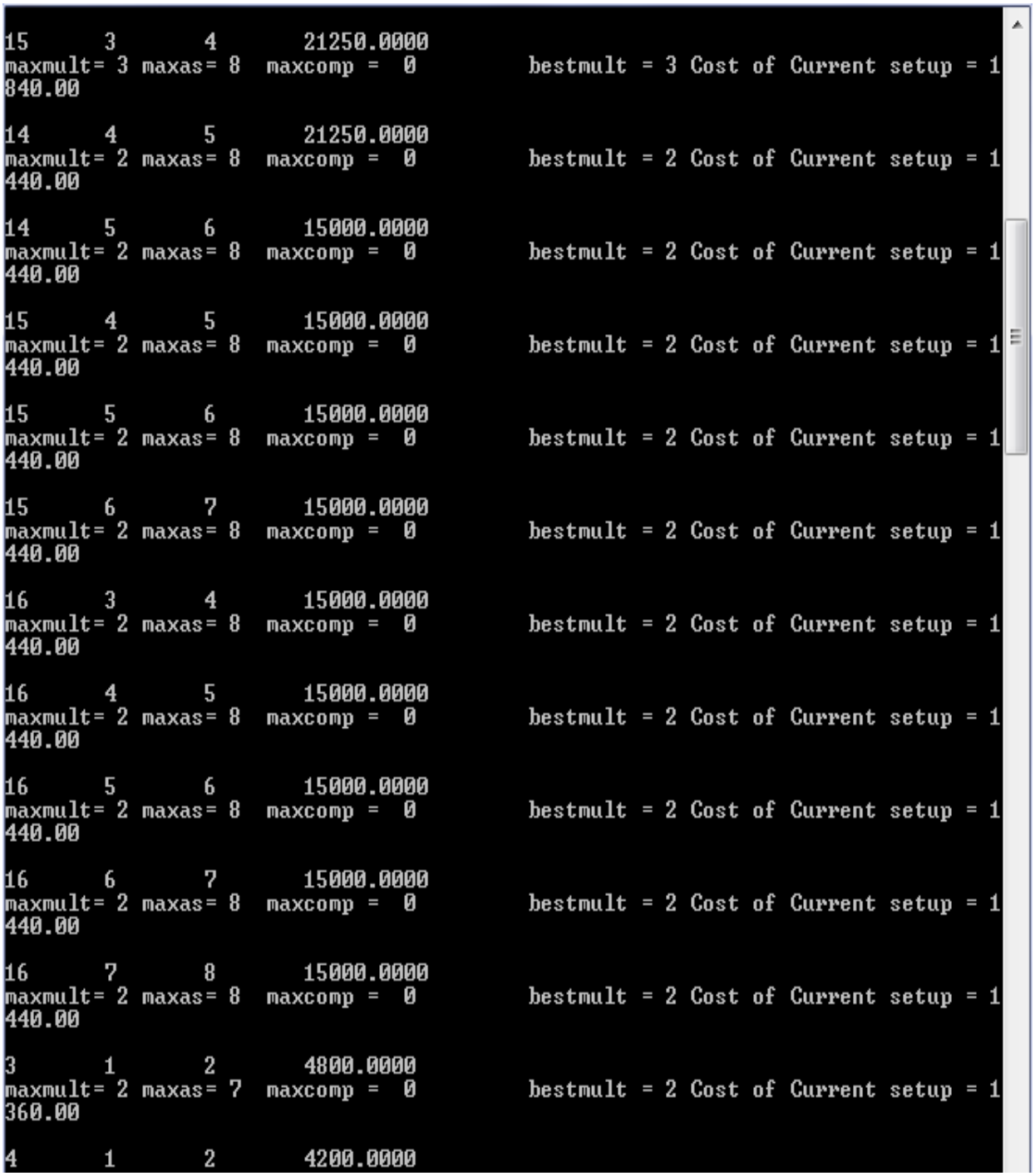
Implementation Result of MPEG (Motion Vectors) Benchmark in the Proposed Automated Exploration Tool – Part I



Implementation Result of MPEG (Motion Vectors) Benchmark in the Proposed Automated Exploration Tool – Part II



Implementation Result of FIR Benchmark in the Proposed Automated Exploration Tool (Part I)



Implementation Result of FIR Benchmark in the Proposed Automated Exploration Tool

(Part II)

Implementation Result of FIR Benchmark in the Proposed Automated Exploration Tool – Part III