A FAST DESIGN SPACE EXPLORATION BASED ON PRIORITY FACTOR FOR A MULTI PARAMETRIC OPTIMIZED HIGH LEVEL SYNTHESIS DESIGN FLOW

By

Anirban Sengupta

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, 2010

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

_____

Anirban Sengupta

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

_____

Anirban Sengupta

ABSTRACT

Title of Thesis:

A FAST DESIGN SPACE EXPLORATION BASED ON PRIORITY FACTOR FOR A MULTI PARAMETRIC OPTIMIZED HIGH LEVEL SYNTHESIS DESIGN FLOW

Thesis Submitted By:

Anirban Sengupta, Master of Applied Science, 2010

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

This thesis introduces a novel approach to rapid Design Space Exploration (DSE) and presents a formalized High Level Synthesis (HLS) design flow with multi parametric optimization objective using the same design space exploration approach. The proposed approach resolves issues related to DSE such as the precision of evaluation, time exhausted during evaluation and also automation of the exploration process. During DSE a conflicting situation always exists for the designer to concurrently maximize the accuracy of the exploration process and minimize the time spent during DSE analysis. This technique is capable of drastically reducing the number of architectural variants to be analyzed for accurate selection of the optimal design point in a short time. The DSE results for many benchmarks are presented along with a comparison to an existing DSE approach that uses the hierarchal structure method for architecture evaluation. Results indicated significant improvement in speedup compared to the current existing approach.

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 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

A	Total Area of the resources
R_i	The resources available for system designing
R_{clk}	The clock oscillator used as a resource providing the necessary clock frequency to the system
N_{Ri}	The number of resource R_i
K_{Ri}	The area occupied per unit resource 'Ri'
n	Functional resources
L	Latency of execution
T_c	Cycle time of execution
N	Number of data elements to be processed
T_{Ri}	Number of clock cycles needed by resource 'Ri'
T_p	Time period of the clock
p_c	Power consumed per area unit resource at a particular frequency
H (z)	The transfer function of the filter in the frequency domain
p	Position where the variant is located in the design space
i	An index
$P_{optimal}$	The constraint for Power Consumption
$T_{optimal}$	The constraint for Execution Time
v_{Ri}	Number of variants of resource 'Ri'
P	Total power consumption
T_{exe}	Total execution time
M	Each Performance Parameter

Chapter 1
Introduction

1.1 Overview

The design and development of systems with heterogeneous performance optimization objective requires extensive analysis and assessment of the design space, not only due to the assorted nature of the parameters, but also due to the diversity in architecture for implementation. Given the specifications and the system requirements, the aim of designers is to reduce the large and complex design space into a set of feasible design solutions meeting performance objectives and functionality.

As the design complexity factor increases in leaps and bounds by the addition of more silicon per unit area, the design method for modular systems with multi-parametric optimization objectives must be formalized. Simultaneously optimizing multiple performance parameters with conflicting objectives, such as hardware area (or power consumed) and time of execution is becoming more complex and significant for successful design of these systems. Designs decisions at the Electronic System Level (ESL) have more impact on the design quality than the

decisions made at low level i.e. logic level. For superior design quality the assessment and selection should be comprehensive and accurate [1] [6].

For most modular systems and systems based on strict operational constraints, the selection of the optimal architecture is the most important step in the development process. Design space architecture can have innumerable design options for selection and implementation based on the parameters of optimization. Hence, selection of the optimal architecture from the design space, which satisfies all the performance objectives, is crucial for the present generation of System-on-Chip (SoC) designs [1]. Furthermore, the striking increase in demand for portable embedded computing devices has led to new System-on-Chip (SoC) architectures intended for embedded systems. To be economically sensible these SoC architectures should nurture a broad suite of different applications, leading to the recent focus on parameterized SoC. The embedded computing devices, on the other hand, have varying design objectives such as execution time and hardware area. Hence these parameterized SoC architectures must be configured in such a way so that the system concurrently satisfies the varied performance objectives for a class of applications. As a result, multi objective design space exploration approaches have emerged for resolving this heterogeneous problem [31]. As it is always possible to implement different functions of a system on different hardware components, the architecture design space has become more complex to analyze [2].

In the case of high level synthesis, performing design space exploration to choose the best candidate architecture by concurrently satisfying many operating constraints is generally considered to be the most important stage in the whole design flow. Since the design space is

huge, there needs to be an efficient way to explore the best candidate architecture for the system design based on the target application. The method for exploration of the best candidate micro-architecture should not only be less in terms of complexity factor and time, but should also explore the variant in an efficient way meeting all the specifications provided. The process of high level synthesis design is very complicated and descriptive and is usually performed by system architects. Depending on the application, the process of defining the problem, performing design space exploration and the other steps required for its successful accomplishment are very time consuming. The modern high level synthesis design flow should be multi-parametric optimized in terms of area occupied, execution time and power consumption. Furthermore, recent advancements in areas of communications and multimedia have led to the growth of a wide array of applications requiring huge data processing at minimal power expense. Such data hungry applications demand satisfactory performance with power efficient 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 quite complex, an efficient approach to explore the design space for selecting the best design option is needed.

A novel approach for finding the best design architecture with multi-parametric optimization objective, useful for accelerating design space exploration in high level synthesis, is presented in this thesis. Until now, no published works have explicitly concentrated on formalizing the design steps of a multi-parametric optimized high level design flow useful for the current generation of high data processing applications and complex SoC design. This work introduces a new formalized high level synthesis design flow to radically reduce the number of

architectural variants to be analyzed for finding the pareto-optimized design point. The introduction of the proposed approach for multi-parameter optimization and design space exploration in high level synthesis design flow allows total automation of the proposed high level design for the current high level synthesis tools.

1.2 Related Works

An engineering problem can generally be described as a phenomenon of analyzing and managing the tradeoffs between contradictory design objectives. The problem of obtaining a comprehensive Pareto optimal set has been addressed by many researchers. In [1] the researchers proposed an approach for synthesis of heterogeneous embedded systems by using Pareto Front Arithmetic (PFA) to explore the giant search spaces. Their method utilized the hierarchical problem structure for exploring the set of Pareto optimal solutions. The problem was also addressed in [3] by suggesting order of efficiency, which assists in deciding preferences amongst the different Pareto optimal points. Work in [4] suggested the identification of a few superior design points from the Pareto set is enough for an excellent design process. In [5] evolutionary algorithms such as the Genetic Algorithm (GA) had been suggested to yield better results for the design space exploration process. The use of GA had also been suggested as a framework for DSE of data paths in high level synthesis in [6]. Another approach was introduced by researchers in [2] which were based on Pareto optimal analysis. According to their work, the design space was arranged in the form of an Architecture Configuration Graph (ACG) for architecture variant analysis and optimization of performance parameters. Their results proved quite promising for architectural synthesis of digital systems. Furthermore in [7] and [8], authors described another approach for DSE in high level systems based on binary encoding of the chromosomes. Work

shown in [9] for DSE suggests that authors used an evolutionary algorithm for successful evaluation of the design for an application specific SoC. The work shown in [10] discusses the optimization of area, delay and power in behavioral synthesis, but does not focus on the high level design flow with multi parametric optimization objective. Authors in [11] introduce a tool called SystemCoDesigner that offers rapid design space exploration with rapid prototyping of behavioral systemC models. In [11] an automated integration was done by integrating behavioral synthesis into the proposed design flow. Additionally, authors in [12] describe current state of the art high-level synthesis techniques for dynamically reconfigurable system. The proposed method avoids constructing hierarchical structures such as in [2] for architecture evaluation and thereby minimizes time overhead. Further evolutionary algorithms such as GA used in [5] [7] [8], although efficient, are very slow in finding the global optimum solution. Moreover it does not always guarantee the selection of global optimum and might eventually end up in finding the local minima. The method proposed in this thesis is capable of evaluation the design space for finding the optimal design solution accurately and rapidly. It can thereby assist the designer in finding the best architecture for the design with increased acceleration.

1.3 Summary of Contribution

This thesis contributes to the following areas:

- Introduces a mathematical structure for Design Space Exploration of the performance parameter hardware area.
- Presents a mathematical framework for Design Space Exploration of the performance parameter power consumption.

- Presents a mathematical framework for Design Space Exploration of the performance parameter Execution time.
- Proposes a new algorithm for arranging the architecture design space in increasing/decreasing order.
- Presents a novel approach for finding the best design architecture with multi-parametric optimization objectives, which is useful for accelerating design space exploration in high level synthesis.
- Explicitly concentrates on formalizing the design steps of a multi-parametric optimized high level design flow useful for the current generation of high data processing applications and complex SoC design.
- Lays the platform for complete automation of the proposed Design Space Exploration for the current Exploration tools as well as automation of the complete HLS designing.
- Develops the complete system of the application at the Register Transfer Level (RTL).

1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 provides some background information related to Design Space Exploration (DSE) and High Level Synthesis (HLS). Chapter 3 describes the proposed frameworks behind Design Space Exploration in High Level Synthesis. Chapter 4 elaborates the High Level Synthesis design flow for Multi Parametric Optimization objective. The exploration process of Architectural Design Space for Power Consumption parameter is described in Chapter 5 while in Chapter 6, the exploration process of Architectural Design Space for execution time parameter is discussed. Chapter 7 provides the detailed insight into the determination of Pareto optimal set of architecture and the selection

methodology for the optimal design variant of architecture. The development of the whole system based on the proposed design flow and the implementation of the same is described in Chapter 8. The results of the proposed DSE approach for various well known High Level Synthesis benchmarks and the speedup obtained when compared to the current existing DSE approach are both provided in Chapter 9. Chapter 10 is dedicated to the conclusion and the future scope of work in this area. The list of publications related to this field of research study and the total list of citations are also provided.

Chapter 2

Background Information

2.1 Theoretical Background on High Level Synthesis

Interdependent tasks such as scheduling, allocation and module selection are important ingredients of the high level synthesis design process. High level synthesis is a methodology of transforming an algorithmic behavioral description into an actual Register Transfer Level (RTL) structure. Therefore high level synthesis methodology contains a sequence of tasks to convert the abstract behavioral description of the algorithm into its respective structural block at RT level. The design at the RT level comprises of functional units such as ALU, storage elements, registers, busses and interconnections. The algorithmic description specifies the inputs and outputs of the behavior of the algorithm in terms of operations to be preformed and data flow [13]. A description of the algorithm is usually represented in the form of an acyclic directed graph known as a sequencing graph [14]. These graphs specify the input/output relation of the algorithm and the data dependency present in the data flow. The graph is defined in terms of its vertices and edges, where the vertices signify the operations and the edges indicate the data dependency present in the function. High level synthesis is therefore a 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. The transformed algorithm at the RT level is comprised of a control unit and the data path unit [15]. High level synthesis offers many advantages, such as productivity gains and efficient design space exploration. Performing DSE at a higher level of abstraction provides more dividend than at lower levels of abstraction, i.e. transistor level or logic level. Traditional high level synthesis design methodology is much simpler than modern design techniques. In general, the initial step of synthesis is to compile the behavioral specification into an internal representation. The next step is to apply high level transformation techniques with the aim of optimizing the behavior as per the desired performance. In order to realize the structure, the final step is to perform scheduling to determine the time at which each operation is executed and the allocation, which is synthesizing the necessary hardware to perform the operations.

Scheduling can be of two different classes: 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. Resource constraints are generally specified by the area occupied by the functional units like adders/subtractors, multipliers, dividers and ALUs. Although the data path of the system consists of registers and interconnections, they are not considered to be included as resource constrained because they are difficult to specify. High level synthesis can be broadly divided into the following steps: input description, internal representation, design space exploration, allocation,

scheduling and binding. Therefore the final structure at the RT level consists of data path and the control path. Traditional high level synthesis design flow falls short for the modern generation of complex VLSI and SoC designs, because the conventional design flow just takes into account the optimization of two parameters, namely area and latency. But the new generation of system designs requires multi parametric optimization strategies in HLS while simultaneously utilizing rapid and efficient DSE approaches for finding the best suitable architecture.

2.1 Theoretical Background on Design Space Exploration

For the present generation of VLSI designs with multi objective nature, the cost of simultaneously solving the problem of scheduling, allocation and module selection by exhaustive search is strictly prohibitive. Multi objective VLSI designs are used in low end Application Specific Integrated Circuits (ASICs) with low power dissipation and acceptable performance, as well as in high end ASICs with high performance requirements and satisfactory power expenditure. Hence, efficient design space exploration techniques are needed to make efficient use of time, due to time to market pressure [6]. Design space exploration 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. Design space exploration has always been a challenge for researchers due to the heterogeneity of the objectives and parameters involved. Typical design space exploration is a multi objective search problem, where the optimization parameters are generally hardware area, execution time and power consumption. Although several recent works have focused on the optimization of reliability as a performance metric, it is still in its nascent stage of development. The current trend towards design space exploration has been the reduction of the design space into a set of Pareto optimal

points by Pareto optimal analysis. Sometimes even the Pareto optimal set can be very large for analysis and selection of the design for system implementation [4]. In order to assist the decision maker in exploring the design space better, an accurate approach efficient in terms of time is very significant for high level synthesis design of hardware systems.

2.3 Overview on the Abstraction Level of Optimization

Today's electronic systems are designed starting from specifications given at a very high level of abstraction. This is because many EDA tools accept a design expressed in a high-level language as input and can automatically produce the corresponding transistor-level implementation with very limited human intervention. 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. This abstraction level provides an insight into the hierarchy that a system can be classified into. Optimization performed at the higher levels of abstraction provides more flexibility and productivity than performing at the lower levels of abstraction. In order to make the search for the optimal solution as effective as possible, the design decision taken at a very early stage (architecture level) of the development process obviously 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 Reasons for Studying 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 reasons are the following [30]:

Reduced design time and high acceleration: If more of the design process is automated, a company can complete a design faster, and thus have a better chance of hitting the market window for that design. Additionally, since much of the cost of the chip is in design development, automating more of that process can lower the cost significantly.

Decreased error probability: If the synthesis process can be verified to be correct, then there is a greater guarantee that the final design will correspond to the initial specification. This means fewer errors and less debugging time for new chips.

The ability to search 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.

Documenting the design process: An automated system can keep track of what design decisions were made and what the consequences of those decisions were.

Easy availability of IC technology: As more design expertise is moved into the synthesis system, it becomes easier for non-expert designers to manufacture a chip that meets a given set of specifications and operating constraints.

Chapter 3

Proposed Framework behind Design Space Exploration in High Level Synthesis

3.1 Mathematical Derivation for Area Model

Let the area of the resources be given as 'A'. R_i denotes the resources available for system designing; where 1<i<n.

R_{clk} refers to the clock oscillator used as a resource providing the necessary clock frequency to the system. The total area can be represented as the sum of all the resources used for designing the system. Hence total area is given in equation (1):

$$A = \sum A(Ri) \tag{1}$$

Area can be expressed as the sum of the resources i.e. adder/subtractor, multiplier, divider etc and also the clock frequency oscillator. Therefore for a system with 'n' functional resources equation (1) can also be represented as shown in equation (2):

$$A = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) + A(R_{clk}) \tag{2}$$

Where N_{Ri} represents the number of resource R_i and 'K_{Ri}' represents the area occupied per unit resource '*Ri*' (1<=i<=n); Applying partial derivatives to equation (2) with respect to N_{R1}, N_{R2}….N_{Rn} yields equation (3), equation (4) and equation (5) respectively as shown below:

$$\frac{\partial A}{\partial N_{R1}} = \frac{\partial(\sum(N_{R1}\cdot K_{R1}+...+N_{Rn}\cdot K_{Rn})+A(R_{clk})}{\partial N_{R1}} = K_{R1} \tag{3}$$

$$\frac{\partial A}{\partial N_{R2}} = \frac{\partial(\sum(N_{R2}\cdot K_{R2}+...N_{Rn}\cdot K_{Rn})+A(R_{clk})}{\partial N_{R2}} = K_{R2} \tag{4}$$

$$\frac{\partial A}{\partial N_{Rn}} = K_{Rn} \tag{5}$$

According to the theory of approximation by differentials [17] the change in the total area can be approximated by equation (6):

$$dA = \frac{\partial A}{\partial N_{R1}}\cdot\Delta N_{R1} + \frac{\partial A}{\partial N_{R2}}\cdot\Delta N_{R2} + ... + \frac{\partial A}{\partial N_{Rn}}\Delta N_{Rn} + \Delta A(R_{clk}) \tag{6}$$

where symbol 'Δ' is called the delta operator.

Substituting equation (3), (4) and (5) into equation (6) yields equation (7) shown below:

$$dA = \Delta N_{R1}\cdot K_{R1} + \Delta N_{R2}\cdot K_{R2} + ... + \Delta N_{Rn}\cdot K_{Rn} + \Delta A(R_{clk}) \tag{7}$$

$$dA = (\Delta N_{R1}\cdot K_{R1}) + (\Delta N_{R2}\cdot K_{R2}) + \ldots + \Delta N_{Rn}\cdot K_{Rn} + \Delta A(R_{clk}) \tag{8}$$

The change of area contributed by resource R1	The change of area contributed by resource R2	The change of area contributed by resource Rn	The change of area contributed by resource clock

The equation above indicates the rate of change of area with respect to resource R1, R2 ….R_n. Here the clock oscillator has been considered a resource, which contributes to the area occupied by the hardware resources.

The term Priority Factor (PF) will be used often when exploring the design space in the proposed approach. The PF is a determining factor for determining the influence of a particular resource on the variation of the optimization parameters such as area, time of execution and power consumption. This PF will be used later to organize the architecture design space consisting of variants in increasing or decreasing order of magnitude The PF for the resource R_1, R_2 …..R_n is given as:

$$PF(R1) = \frac{\Delta N_{R1} \cdot K_{R1}}{N_{R1}} \tag{9}$$

$$PF(R2) = \frac{\Delta N_{R2} \cdot K_{R2}}{N_{R2}} \tag{10}$$

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \tag{11}$$

$$PF(Rclk) = \frac{\Delta A(Rclk)}{N_{Rclk}} \tag{12}$$

The factor defined above determines how the variation in area is affected by the change of number of that certain resource. Hence, the PF is the rate of change of area with respect to the change in number of resources.

3.2 Mathematical Derivation for Execution Time Model

For a system with ‘n’ functional resources the time of execution can be represented by the following formula:

$$T_{exe} = [L + (N-1) \cdot T_c] \tag{13}$$

where ‘L’ represents latency of execution, ‘T_c’ represents the cycle time of execution , ‘N’ denotes the number of data elements to be processed.

Since the number of data elements to be processed is large for real life applications, '*L*' can be ignored and cycle time (T_c) becomes a primary factor. The maximum cycle time under maximum resource constraint can be given as equation (14):

$$T_c = (N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p \quad (14)$$

N_{Ri} represents the number of resource of Ri and 'T_{Ri}' represents the number of clock cycles needed by resource '*Ri*' (1<=i<=n) to finish each operation and 'T_p' is the time period of the clock. From the theory of approximation of differentials, the change in the total cycle time can be approximated as in equation (15).

$$dT_c = (\frac{\partial T_c}{\partial N_{R1}} \cdot \Delta N_{R1} + \frac{\partial T_c}{\partial N_{R2}} \cdot \Delta N_{R2} + ... + \frac{\partial T_c}{\partial N_{Rn}} \Delta N_{Rn}) + \Delta T_p \cdot \frac{\partial Tc}{\partial T_p} \quad (15)$$

Applying partial derivatives to equation (14) with respect to N_{R1}, N_{R2}.....N_{Rn} and T_p will produce the following set of equations:

$$\frac{\partial T_c}{\partial N_{R1}} = \frac{\partial (N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p}{\partial N_{R1}} = T_{R1} T_p \quad (16)$$

$$\frac{\partial T_c}{\partial N_{R2}} = \frac{\partial (N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p}{\partial N_{R2}} = T_{R2} T_p \quad (17)$$

$$\frac{\partial T_c}{\partial N_{Rn}} = \frac{\partial (N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p}{\partial N_{Rn}} = T_{Rn} T_p \quad (18)$$

$$\frac{\partial T_c}{\partial T_p} = \frac{\partial (N_{R1} \cdot T_{R1} + N_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p}{\partial T_p} \quad (19)$$

$$= N_{R1 \cdot} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn} \quad (20)$$

Now substituting equations (16), (17), (18) and (20) in equation (15). The substitution yields the following equation (21):

$$dT_c = \Delta N_{R1} \cdot T_{R1} \cdot T_p + \Delta N_{R2} \cdot T_{R2} \cdot T_p + ... + \Delta N_{Rn} \cdot T_{Rn} T_p$$

$$+\Delta T_p \cdot (T_{R1} \cdot N_{R1} + T_{R2} \cdot N_{R2} + ... + T_{Rn} \cdot N_{Rn}) \quad (21)$$

Equation (21) represents the change in total cycle time with the change in the number of resources and the clock period (clock frequency).

$\Delta N_{R1} \cdot T_{R1} \cdot T_p$ = The change of *'T_c'* caused by the change in the number of resource R1; Similarly,

$\Delta N_{Rn} \cdot T_{Rn} \cdot T_p$ =The change of *'T_c'* caused by the change in the number of resource Rn.

Finally, $\Delta T_p \cdot (T_{R1} \cdot N_{R1} + T_{R2} \cdot N_{R2} + ... + T_{Rn} \cdot N_{Rn})$ = The change of 'T_c' caused by the change in clock period (clock frequency) and the change in the number of all resources available.

The priority factor (PF) can be defined for the 'time of execution' parameter. The (PF) for the resource R_1, R_2 …..R_n is given as:

$$PF(R1) = \frac{\Delta N_{R1} \cdot T_{R1}}{N_{R1}} \cdot (T_p)^{\max} \quad (22)$$

$$PF(R2) = \frac{\Delta N_{R2} \cdot T_{R2}}{N_{R2}} \cdot (T_p)^{\max} \quad (23)$$

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot T_{Rn}}{N_{Rn}} \cdot (T_p)^{\max} \quad (24)$$

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + .. + N_{Rn} \cdot T_{Rn}}{N_{Rclk}} \cdot (\Delta T_p) \quad (25)$$

The factors defined above indicate the rate of change of cycle time (T_c) with the change in number of resources at minimum clock frequency. For example, equation (22) indicates the rate of change of cycle time with a change in the number of that particular resource (e.g. change in number of adders/subtractors from one to three adders/subtractors) at minimum clock frequency.

Minimum clock frequency is considered because the clock period is the maximum at this frequency. Hence, the change in the number of a specific resource at maximum clock period influences the change in the cycle time the most, compared to the change in cycle time at other

clock periods. The PF yields a real number, which suggests the extent to which the change in number of that particular resource contributes to the change in cycle time.

3.3 Mathematical Derivation for Power Consumption Model

Therefore for a system with 'n' functional resources, the total power consumption (P) of the resources in a system can be represented by the following equation (26):

$$P = \sum_{i=1}^{n} (N_{Ri} \cdot K_{Ri}) \cdot p_c \tag{26}$$

$$P = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + .. + N_{Rn} \cdot K_{Rn}).p_c \tag{27}$$

'N_{Ri}' represents the number of resource of resource Ri as mentioned before. 'K_{Ri}' represents the area occupied per unit resource Ri and 'p_c' denotes the power consumed per area unit resource at a particular frequency of operation.

Using the theory of approximation of differentials the change in power consumption can be formulated as shown in equation (28):

$$dP = (\frac{\partial P}{\partial N_{R1}} \cdot \Delta N_{R1} + \frac{\partial P}{\partial N_{R2}} \cdot \Delta N_{R2} + ... + \frac{\partial P}{\partial N_{Rn}} \Delta N_{Rn}) + \Delta p_c \cdot \frac{\partial P}{\partial P_c} \tag{28}$$

Applying partial derivative to equation (27) produces the following equations:

$$\frac{\partial P}{\partial N_{R1}} = \frac{\partial [(N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) \cdot p_c]}{\partial N_{R1}}$$

$$= K_{R1} \cdot p_c \tag{29}$$

$$\frac{\partial P}{\partial N_{R2}} = \frac{\partial [(N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) \cdot p_c]}{\partial N_{R2}}$$

$$= K_{R2} \cdot p_c \tag{30}$$

$$\frac{\partial P}{\partial N_{Rn}} = \frac{\partial [(N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) \cdot p_c]}{\partial N_{Rn}}$$

$$= K_{Rn} \cdot p_c \tag{31}$$

$$\frac{\partial P}{\partial p_c} = \frac{\partial[(N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) \cdot p_c]}{\partial p_c} \tag{32}$$

$$= N_{R1\cdot} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn} \tag{33}$$

Substituting equations (29), (30), (31) and (33) in equation (28) yields equation (34) below:

$$dP = (\Delta N_{R1} \cdot K_{R1} \cdot p_c + \Delta N_{R2} \cdot K_{R2} \cdot p_c + ... + \Delta N_{Rn} \cdot K_{Rn} p_c)$$

$$+ \Delta p_c \cdot (K_{R1} \cdot N_{R1} + K_{R2} \cdot N_{R2} + ... + K_{Rn} \cdot N_{Rn}) \tag{34}$$

Equation (34) represents the change in total power consumption with the change in the number of all resources and the clock period (or clock frequency).

$\Delta N_{R1} \cdot K_{R1} \cdot p_c$ = The change of '*P*' contributed by the change in the number of resource R1; Similarly, $\Delta N_{Rn} \cdot K_{Rn} \cdot p_c$ = The change of '*P*' contributed by the change in the number of resource Rn;

Finally, $\Delta p_c \cdot (K_{R1} \cdot N_{R1} + K_{R2} \cdot N_{R2} + ... + K_{Rn} \cdot N_{Rn})$ = The change of '*P*' contributed by the change in clock period (or clock frequency) and the change in the number of all the resources available.

$$PF(R1\ ) = \frac{\Delta N_{R1} \cdot K_{R1}}{N_{R1}} \cdot (p_c)^{\max} \tag{35}$$

$$PF(R2\ ) = \frac{\Delta N_{R2} \cdot K_{R2}}{N_{R2}} \cdot (p_c)^{\max} \tag{36}$$

$$PF(Rn\ ) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \cdot (p_c)^{\max} \tag{37}$$

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + .. + N_{Rn} \cdot T_{Rn}}{N_{R_{clk}}} \cdot (\Delta p_c) \tag{38}$$

The PF defined from equations (35) to (37) indicates the rate of change in the total power consumption with the change in number of resources at maximum clock frequency. For example,

equation (35) indicates the rate of change of total power consumption of system with the change in the number of that particular resource (e.g. change in number of adders from one to three) at maximum clock frequency. The PF helps in arranging the architectural variants of the design space in increasing or decreasing order of magnitude depending on the parameter of optimization. This facilitates the selection of the optimal design point that satisfies all operating constraints and optimization requirements specified.

In the above equations, the maximum clock frequency was considered because the total power consumption is at the maximum at this frequency. Hence, the change in the number of a specific resource at maximum clock frequency will influence the change in the total power consumption (P) the most, compared to the change at other clock frequencies. The PF yields a real number, which suggests the extent to which the change in number of that particular resource contributes to the change in total power consumption for the system.

The PF is arranged in such a way that the resource with the minimum PF is chosen first, gradually increasing and then ending at the resource with the highest priority factor. The above rule applies for all three parameters described in this chapter.

Chapter 4

High Level Synthesis Design Flow with Multi Parametric Optimization Objective

4.1 Proposed High Level Synthesis Design Flow using the Priority Factor Method for Design Space Exploration

The priority factor is used to organize the design space in increasing order (or decreasing) based on a proposed algorithm. The next section focuses entirely on design flow starting with the real specification and formulation, and eventually obtaining the register transfer level structure,

by performing design space exploration. Three parameters are optimized during the demonstration of design flow for high level synthesis: power consumption, time of execution and hardware area of the resources. Fig. 1 shows the entire design flow for high level synthesis using

the proposed methodology for DSE. The details of the design flow are discussed in the next chapters.

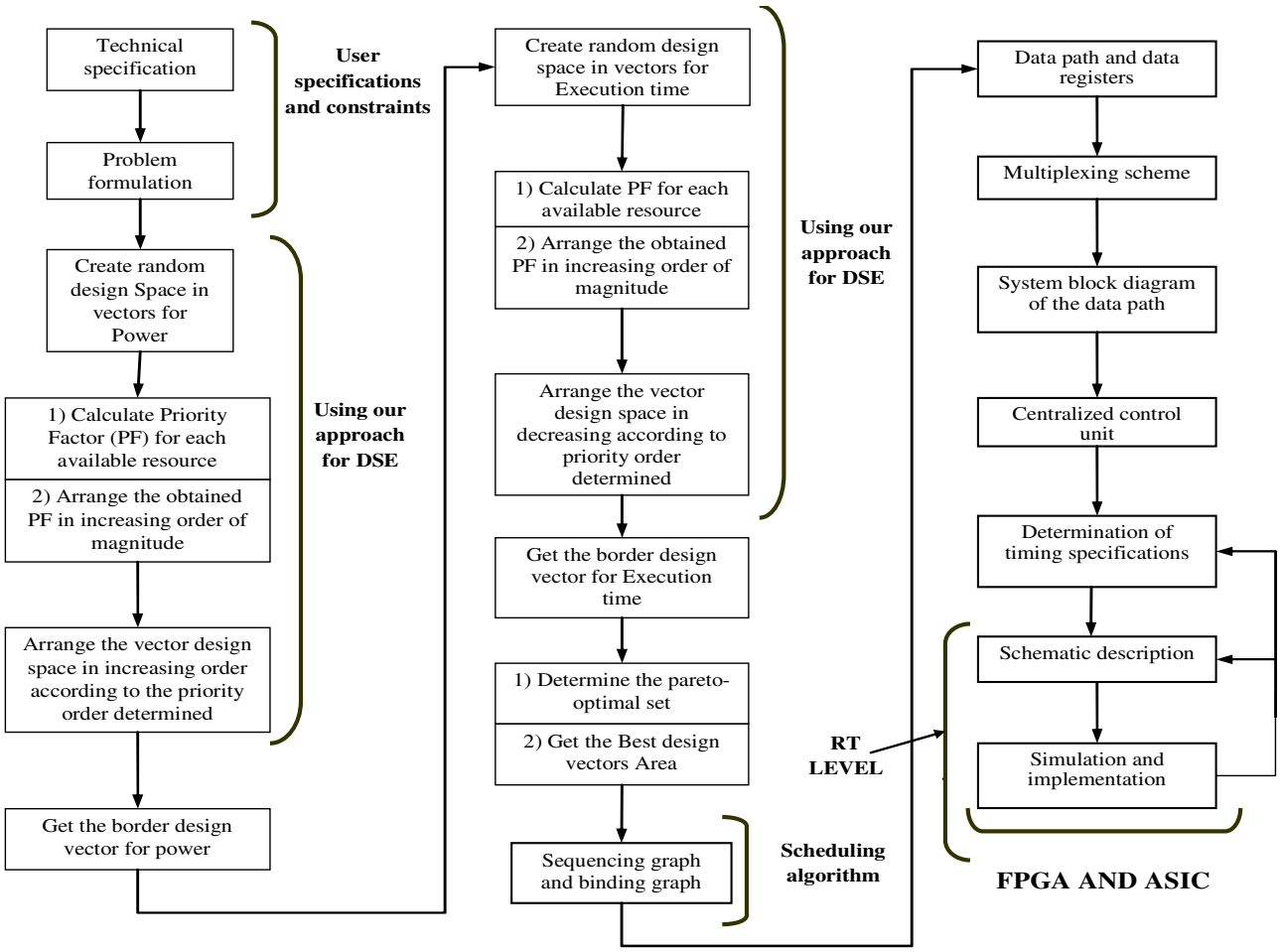


Figure1. The proposed high level design flow for multi-parametric optimization requirement

4.2 Design Flow Initiation

This stage marks the beginning of the high level synthesis design flow starting with the technical specifications. The application should be properly defined with its associated data structure. This phase of the design stage is critical for the designer because the operational

constraints must be clearly defined along with the parameters to be optimized. These specifications act as the input information for the high level synthesis tools. For the demonstration of design flow the following real specifications are assumed.

1) Maximum power consumption**:** 8 watts (W)
2) Maximum resources available for the system design:
    a) 3 Adder/subtractor units.
    b) 4 Multiplier units
    c) 2 clock frequency oscillators: 50 MHz and 200 MHz
3) Maximum time of execution: 140 μs (For 1000 sets of data)
4) Hardware area of resources**:** minimum while satisfying the above constraints.

The following specifications are also assumed as an example for each resource available for system design.

a) No of clock cycles needed for multiplier to finish each operation: 4 cc

b) No of clock cycles needed by the Adder/subtractor: 2 cc

c) Area occupied by each adder/subtractor: 20 a.u. on the chip. (e.g. 20 CLB on an FPGA)

d) Area occupied by each multiplier: 100 a.u. on the chip. (e.g. 100 CLB on an FPGA)

e) Area occupied by the 50MHz clock oscillator: 4 area units (a.u.)

f) Area occupied by the 200 MHz clock oscillator: 10 area units (a.u.)

g) Power consumed at 50 MHz: 10mW/area unit.

h) Power consumed at 200 MHz: 40mW/area unit.

4.3 Problem Description

During the problem formulation stage for high level synthesis, the mathematical model of the application is used to define the behavior of the algorithm. The model suggests the input/ output

relation of the system and the data dependency present in the function. In this work, a transfer function of an IIR Butterworth filter is used to demonstrate the high level synthesis design flow. The choice of IIR Butterworth filter is arbitrary and any other filter can also be used for demonstration. In this work, the selected filter has just been used as an example benchmark application. The conversion of the analog filter to its digital counterpart is not shown in the thesis because there are well known methods to obtain it, such as Bilinear Transformation, impulse invariant [18] etc. The transfer function of a second order IIR digital Butterworth filter function can be given as [18]:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^3 + 3z^2 + 3z + 1}{6z^3 + 2z} \tag{39}$$

$$\frac{Y(z)}{X(z)} = \frac{1/6 + (1/2)Z^{-1} + (1/2)Z^{-2} + (1/6)Z^{-3}}{1 + (1/3)Z^{-2}} \tag{40}$$

$$y(n) = 0.167x(n) + 0.5x(n-1) + 0.5x(n-2) + 0.167x(n-3) - 0.33y(n-2) \tag{41}$$

Where *H (z)* denotes the transfer function of the filter in the frequency domain and *x(n)*, *x(n-1)*, *x(n-2), x(n-3)* represent the input variables for the filter in time domain. *y(n)* and *y(n-2)* represent the present output of the filter and the previous output of the filter in the time domain. '*z*' represents the unit delay operator. For simplicity in explanation, constants 0.167, 0.5 and 0.33 are denoted with 'A', 'B' and 'C' respectively.

Chapter 5

Exploration of the Architectural Design Space for Power Consumption

5.1 Creation of Random Architecture Design Space for Power consumption

The architecture design space for hardware area is represented in the form of vectors consisting of the resources available for the system. The random organization of the design space is used as a method to represent the different combinations of the resources that comprise the total design space. This initial arrangement can be made in any order and is simply used for visualizing the total architectural variants available in the system. The design space can change based on the available resources of a system. The total design space is first created according to the specifications mentioned for total available resources for the system design (see Chapter 4). The variable Vn = (N_{R1}, N_{R2}, N_{R3}) is used to represent the architecture design space. The variables N_{R1}, N_{R2} and N_{R3} indicate the number of adders/subtractors, multipliers and clock frequencies. According to the specification in Chapter 4, $1<=N_{R1}<=3$, $1<=N_{R2}<=4$ and

1<=N_{R3}<=2. The design space in Fig.2 shows the different combinations of available resources viz. adder/subtractor, multiplier and clock during system design.

V1 = (1,1,1)	V2 = (1,2,1)	V3= (1,3,1)	V4= (1,4,1)
V5= (1,1,2)	V6= (1,2,2)	V7= (1,3,2)	V8= (1,4,2)
V9= (2,1,1)	V10= (2,2,1)	V11 = (2,3,1)	V12 = (2,4,1)
V13= (2,1,2)	V14= (2,2,2)	V15= (2,3,2)	V16= (2,4,2)
V17= (3,1,1)	V18= (3,2,1)	V19= (3,3,1)	V20= (3,4,1)
V21= (3,1,2)	V22 = (3,2,2)	V23= (3,3,2)	**V24= (3,4,2)**

Figure2. Design space with all possible combinations of resources

5.2 Calculation of Priority Factor (PF) for each available resource to determine the Priority Order (PO)

For resource adder/subtractor (R1):

$$PF(R1\ ) = \frac{\Delta N_{R1} \cdot K_{R1}}{N_{R1}} \cdot (p_c)^{\max} = \frac{(3-1)\cdot 20}{3} \cdot 40 = 533.33$$

For resource multiplier (R2):

$$PF(R2) = \frac{\Delta N_{R2} \cdot K_{R2}}{N_{R2}} \cdot (p_c)^{\max} = \frac{(4-1)\cdot 100}{4} \cdot 40 = 3000$$

For resource clock oscillator (R_{clk}):

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2}}{N_{Rclk}} (\Delta p_c)$$

$$= \frac{(3 \cdot 20 + 4 \cdot 100) \cdot (40 - 10)}{2} = 6900$$

The above factors are a true measure of the change in power consumption with the change in number of a specific resource. For example, according to the above analysis the change in clock frequency from 50 MHz to 200 MHz affects the change in power the most, while the change in

number of adder/subtractor affects the change in power consumption the least. Similarly, the change in number of multipliers influences the change in power consumption more than the adder/subtractor, but less than the clock. According to the priority factors calculated, the priority order (PO) can be arranged so that the resource with the lowest priority factor is assigned the highest priority order while the resource with the highest priority factor is assigned the lowest priority order. The priority order of the resources increases with the decrease in priority factor of the resources. Therefore the following PO of the resources for arranging the design variants in increasing order can be attained: $PO\ (R1) > PO\ (R2) > PO\ (Rclk)$

Based on the above priority the variants from the design space are chosen so that the design space for power consumption can be organized in increasing orders of magnitude. The next section shows how to arrange the elements in increasing order using the proposed algorithm. The arrangement of the design variants in increasing order helps to prune the design space for obtaining the border variant for power consumption.

5.3 Arrangement of Architectural Design Space in increasing order using Algorithm

Since the design space is large for the present generation of complex multi objective VLSI designs, analyzing the design space exhaustively to find the architecture that best meets the user specified objectives is strictly prohibited. Due to increased complexity in VLSI and SoC designs, the major problem has been the examination of the design variants in the large design space for selecting the optimal design option, which is acceptable in terms of all the user constraints and predefined specifications [6]. Hence, obtaining a superior quality design for the

user specified specification requires a structured methodology for exploring the large design spaces. Design space exploration when performed at the higher level of abstraction is more beneficial than performing it at the lower level of abstraction such as the logic or the transistor level. The job of design space exploration is a battle between optimizing the following two contradictory conditions: selecting the optimum design option and efficiently searching the space in a short time. Hence, there is always a tradeoff not only between the contradictory parameters of optimization during high level synthesis design, but also between the above mentioned conditions during design space exploration in high level synthesis. To proficiently analyze the complex design spaces, a fast but efficient means of arriving at the best result is needed. Analyzing the design to obtain the best architecture according to the requirements specified, requires an efficient design space exploration technique. This section presents an algorithm for arranging the random design space in an organized increasing order for the power consumption parameter. The algorithm is based on priority order sequencing. The elements are placed in such a way, so that the element on the top has the least power consumption and the element on the bottom has the highest power consumption. A flow chart model of this algorithm is shown in Fig.3 which describes the steps involved in organizing the elements. The arranged design space for power consumption is shown in Fig.4.

5.4 Determination of Border Variant for Power Consumption using Binary Search

After the elements are arranged in increasing order, the design space is pruned to obtain the border variant for power consumption. Binary search is applied to the design space shown in Fig.4.

Binary search is preferred to the other search techniques because it is extremely fast. It uses the principle of 'divide and conquer' to rule out half of the elements in each comparison. Hence, the border variant can be determined very soon. Furthermore, since binary search works well for large size sorted elements, binary search can find the border variant in the large sorted design space with a complexity of log N (see chapter 9 for more explanations). The binary search algorithm yields the design variants shown in Table 1. The obtained variants are further analyzed for power consumption according to equation (26). '$P_{optimal}$' is the value of power consumed that is specified as a constraint at the beginning of the design flow. 'P^i' is the value of power consumption for the variant#i. When the value of P^i is less than the value of specified $P_{optimal}$,

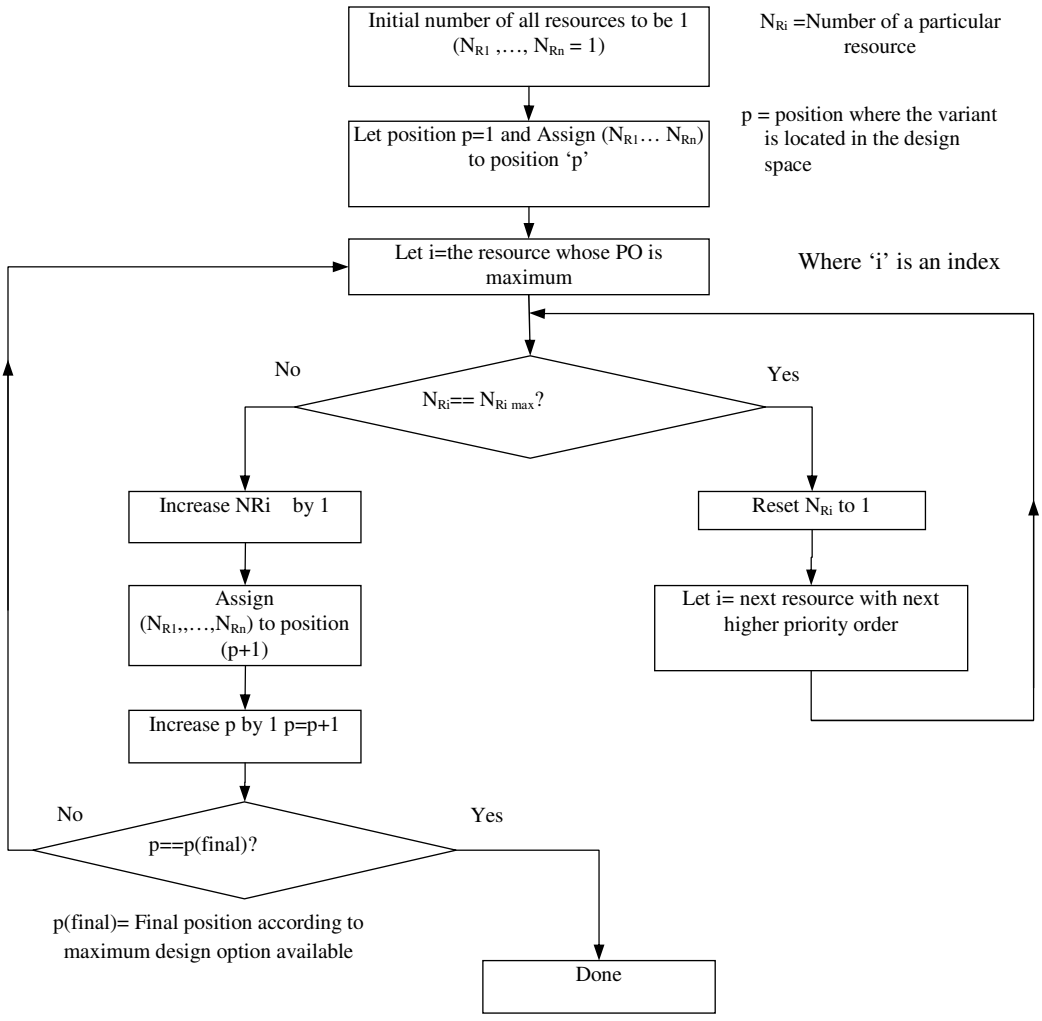
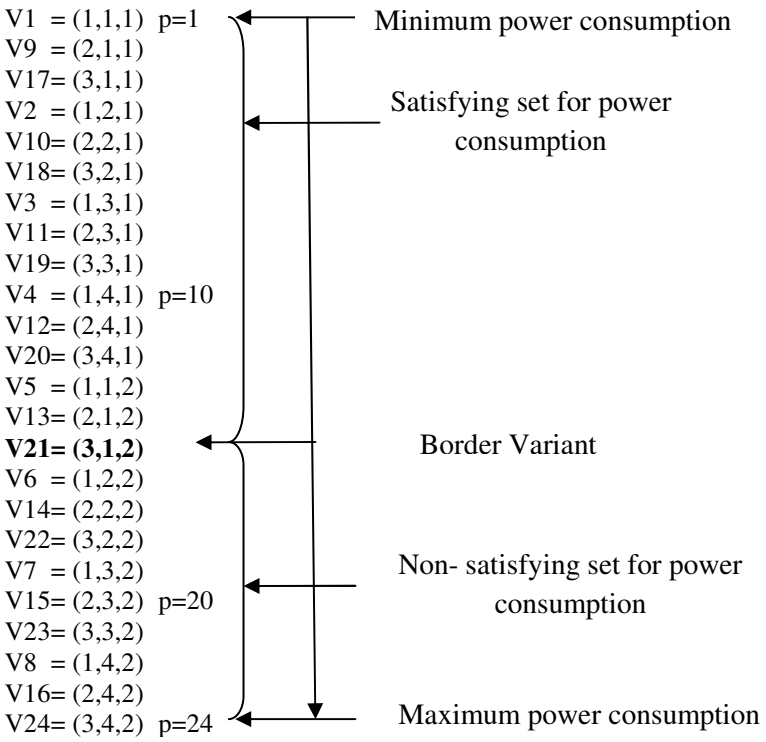


Figure3. Flow chart model of the proposed algorithm

then the southern portion (down) of the design space with respect to the calculated value of P^i, is searched. On the contrary, if the value of the P^i is more than the value of specified $P_{optimal}$, then the northern portion (up) of the design space with respect to the calculated value of P^i is searched. After four comparisons it is evident that variant 21 ('V21') is the last variant which satisfies the $P_{optimal}$.



Arrangement of Power in increasing order from the top to the bottom element using the proposed algorithm

Fig.4. The arranged design space for power consumption

Table 1. Variants obtained after pruning the design space for power consumption

Variants	Power Consumption in watts (W)	Decision based on the $P_{optimal}$
V20	[(3*20)+(4*100)]*10mw = 4.6 W	$P^{20} < P_{optimal}$, search down in the space
V22	[(3*20)+(2*100)]*40mw = 10.4 W	$P^{18} > P_{optimal}$, search up in the space
V21	[(3*20)+(1*100)]*40mw = 6.4 W	$P^{7} < P_{optimal}$, search down in the space
V6	[(1*20)+(2*100)]*40mw = 8.8 W	Stop

This variant is referred to as the border variant for power. The significance of this border variant lies in the fact that all variants to the top of the border variant satisfy the condition of $P_{optimal}$, while any variant to the bottom fails to meet the constraint.

Chapter 6

Exploration of the Architectural Design Space for Execution Time

6.1 Creation of Random Architecture Design Space for Execution time

The architecture design space for execution time is also represented in the form of vectors consisting of the resources available for the system. The random organization of the design space is used as a way to represent the different combinations of the resources that comprise the total design space. This initial arrangement can be made in any order and is simply used for visualizing the total architectural variants available in the system. The design space can change based on the available resources of a system. The total design space is first created according to the specifications mentioned for total available resources for the system design (see chapter 4). The variable Vn = (N_{R1}, N_{R2}, N_{R3}) is used to represent the architecture design space. The

variables N_{R1}, N_{R2} and N_{R3} indicate the number of adders/subtractors, multipliers and clock frequencies. According to the specification in Chapter 4, $1<=N_{R1}<=3$, $1<=N_{R2}<=4$ and $1<=N_{R3}<=2$. The design space in Fig.2 shows the different combinations of available resources viz. adder/subtractor, multiplier and clock during system design.

6.2 Calculation of Priority Factor (PF) for each available resource for Execution Time parameter to determine the Priority Order (PO)

For resource adder/subtractor (R1):

$$PF(R1) = \frac{\Delta N_{R1} \cdot T_{R1}}{N_{R1}} \cdot (T_p)^{\max} = \frac{(3-1) \cdot 2}{3} \cdot (0.02) = 0.026$$

For resource multiplier (R2):

$$PF(R2\ ) = \frac{\Delta N_{R2} \cdot T_{R2}}{N_{R2}} \cdot (T_p)^{\max} = \frac{(4-1) \cdot 4}{4} \cdot (0.02) = 0.06$$

For resource clock oscillator (R_{clk}):

$$PF(R_{clk}) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2}}{N_{Rclk}} \cdot (\Delta T_p)$$

$$= \frac{(3 \cdot 2 + 4 \cdot 4) \cdot (0.02 - 0.005)}{2} = 0.165$$

The factors determined above measure the change in time of execution with a corresponding change in the number of a specific resource. For instance, according to the above analysis the change in number of adder/subtractor affects the change in time of execution the least, while the change in clock frequency from 50 MHz to 200 MHz affects the change in time of execution the most. Similarly, the change in multiplier influences the change in execution time lesser than the change in clock frequencies. As explained before in Chapter 5, the following priority order (PO)

for arranging the design variants in increasing order can be organized according to the above priority factors calculated:

$$PO(R1) > PO(R2) > PO(Rclk)$$

6.3 Arrange of Architectural Design Space in decreasing order using the Proposed Algorithm

This approach is based on the multi-parametric optimization requirement for efficient DSE. The arrangement of the design space in decreasing order with the proposed algorithm in Chapter 5 enables the designer to rapidly determine the border variant for execution time. But before the

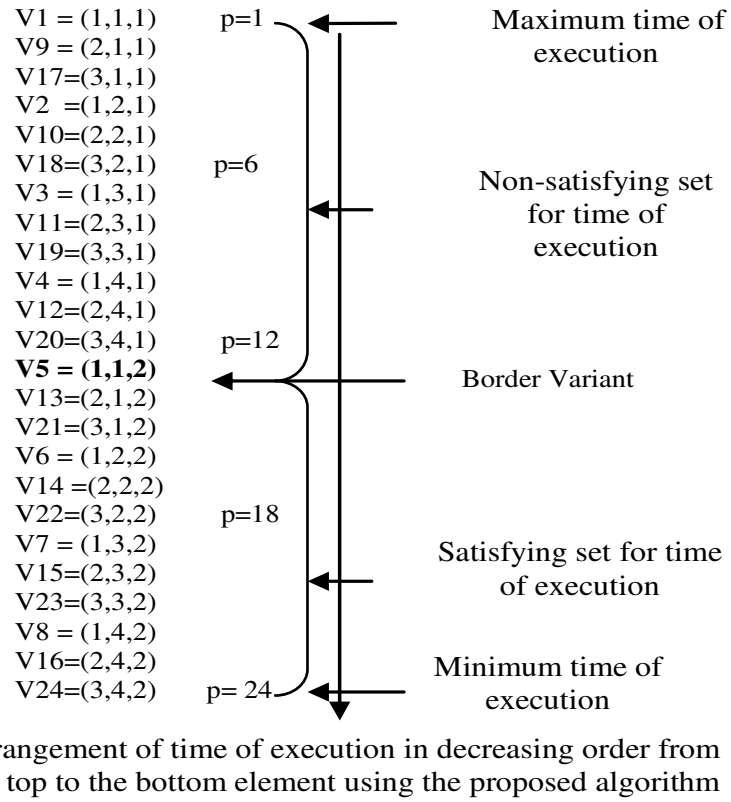


Figure5. The arranged design vector space in decreasing order for time of execution

border variant for execution time can be determined, setting pairs of the performance parameters contradictory to each other must be done by a designer. Setting the performance parameters contradictory to each other facilitates in the reduction of the large design space into a small size, which can be easily evaluated for exploration. After the design space is arranged in decreasing orders of magnitude, then selecting the border variant becomes very easy. The organized design space for time of execution in decreasing order based on the proposed algorithm using the priority factor method is shown in Fig.5.

6.4 Determination of Border Variant for Execution Time using Binary Search

The arrangement of the design space in decreasing order allows the design space to be pruned for finding the border variant of time of execution. As discussed in Chapter 5, binary search algorithm is beneficial compared to the other search techniques when it comes to the question of searching a large size sorted list like a large design space. This is because it is

Table 2. Variants obtained after pruning of the design space for execution time

Variants	Execution time (in µs)	Decision based on the $T_{optimal}$
V20	T_{exe}^{20}= 12 +(1000-1)*8 *0.02 = 160.08 µs	$T_{exe}^{20} > T_{optimal}$ search down in the space
V22	T_{exe}^{22} = 16 +(1000-1)*12 *0.005 = 60.02 µs	$T_{exe}^{22} < T_{optimal,}$ search up in the space
V21	T_{exe}^{21} = 22 +(1000-1)*20 *0.005 = 100.01 µs	$T_{exe}^{21} < T_{optimal}$ search up in the space
V5	T_{exe}^{5} = 22 +(1000-1)*20 *0.005 = 100.01 µs	stop

.

extremely fast and works well for large size sorted lists of elements. Since the design space for real applications is always large, binary search finds the border variant at a complexity of log N (see chapter 9 for more explanation) in the sorted design space. The binary search algorithm is applied to the design space shown in Fig5 and the variants are analyzed according to equation (13) to determine the best variants (Table 2). Analysis reveals that variant number 'V5' is the border variant for the 'time of execution' parameter. Hence all the design variants to the south of the design space must satisfy the constraint imposed by the user.

Chapter 7

Pareto optimal set of architecture

7.1 Determination of Pareto optimal set for the design variants

Once the border variants for the parameters of optimization have been successfully derived, the next phase of the high level synthesis flow is to find the Pareto optimal set of architecture. This set contains all those architectural variants that equally satisfy the constraints imposed by the user. Hence the process of analyzing the initial large design space is reduced to analyzing only the architectural variants in the Pareto-optimal set. After investigation it is found that just three architectural variants from the each satisfying set of optimization parameters, power consumption and time of execution, simultaneously satisfy both power consumed and execution time. The variants are V5, V13 and V21 (see fig4 and Fig5). The priority factor for area is determined using equations (9)-(12) to arrange the variants of the Pareto optimal set in increasing order, similarly to the way it was determined for power and execution time. After

calculation of the PF the priority order is determined. The obtained priority order is: *PO (Rclk)* > *PO (R1)* > *PO (R2).* Using the algorithm described in chapter 5, the variants V5, V13, V21 of the Pareto set are arranged in increasing orders of magnitude. Since the design specification in chapter 4 demanded minimum area overhead with simultaneous satisfaction of the constraints imposed by user, hence the aim is to find the variant with minimum area overhead. After the arrangement of the variants of Pareto optimal set the variant number 'V5' is found to be the only variant among twenty four variants that concurrently optimizes hardware area, power consumption and time of execution while meeting all the specifications provided.

7.2 Verification of the optimal design variant

Performing the analysis for the verification of the optimality of the design variant obtained through the proposed method, is a very important step in the development process. Verifying the optimality of the best design variant obtained ensures that the variant found strictly obeys all the operating constraints provided for power consumption, execution time and hardware area. Further verification plays an important role in the development process because it can easily detect any mistakes made during any design space exploration process. For example, the results of the design space exploration suggested that variant 'x', is the optimal variant which satisfies the entire optimization requirement specified. But that optimal variant 'x' obtained might have resulted due to the mistake done by the designer during manually evaluating the design variants during the design space exploration. Hence verification of the design variant later surely detects the erroneous step performed by the designer earlier. The proposed DSE approach is highly compatible for automation and therefore does not require the manual effort for evaluating the design variants for selecting the best option. Nevertheless, the addition of the

verification stage provides the extra confidence in the early decisions made regarding the selection of best design architecture (variant).

The verification of the best variant obtained in Chapter 7 (Section 7.1), is verified as explained above for all the three optimization parameters. Investigations indicate that the variant obtained is in compliance with all the operating constraints and optimization requisites provided in the design specification.

Chapter 8

Implementation of the Proposed High Level Synthesis Design Flow

8.1 Scheduling and Binding of operations

Representation of the resources in temporal and spatial domain is performed with the aid of a sequencing and binding graph. These structures represent a class of acyclic graphs generally represented using vertices and edges. The vertices indicate the resources used for the system design and the edges denote the dependency of data flow among those used resources. The flow of data elements through different operators in the data path can be visualized with the help of sequencing graphs [11]. Sequencing graphs are used to specify the nature of operation i.e. at which time what resources are subjected to which operation. This graphical representation of the application (algorithm) distinctly underlines the operations in discrete time steps while maintaining the precedence constraints specified in the algorithmic description.

A scheduling problem can be classified into three different categories. Time constrained scheduling must find the cheapest possible schedule that satisfies the constraint with the given

maximum number of time steps. Resource constrained scheduling is used to determine the fastest schedule satisfying the constraints imposed with the given maximum number of resources. Finally, feasible constrained scheduling, whose objective is to schedule an output if it exists, by simultaneously satisfying the constraints for the resource and time. Most of the practical formulations of the scheduling problem are NP-complete in nature [29]. The scheduling problem in general is three fold. It is a combination of timing, dependency and resource constraints.

Scheduling is a process that states the time slot for every operation while fixing the timing length (latency) in such a manner so that the synthesized hardware structure meets the timing restriction specified [6]. A classical example of time constraint scheduling where the scheduler must achieve the goal with a minimum number of functional units possible to realize the behavior is shown here. The scheduling of operations is performed based on the As Soon As Possible (ASAP) algorithm [16] [17]. Though many algorithms are used for scheduling operations such as the As Late as Possible (ALAP), List scheduling, Force Directed scheduling, ASAP, etc., ASAP scheduling algorithm was selected because in the proposed work, the operations should be done as soon as the resources become free. As the processed data is ready, the prepared data from the previous stage is used for the next operation. The sequencing and binding graph for the sample benchmark used in this work is shown in Fig.6.

The concept of binding graph is used in further design stages to realize the function used as a benchmark application for demonstration of the optimized high level synthesis design flow.

8.2 Scheduling and Binding of operations with Data Registers

In terms of architectural synthesis and optimization a circuit is generally specified by the following three dimensions. First, a sequencing graph, second, a set of functional resources described in the form of area and latency, and finally, the operating constraints. The function of registers is to perform data storage and the wires interconnect the different discrete components [11]. In the sequencing graph of the design, Register P has been added in time slot T2 because the results of the adder/subtractor at time slot T1 are not used until time slot T3. The sequencing graph with data registers is shown in Fig.7 (see next page). The latency for the function is calculated as 11 clock cycles. Fig.8 shows the cycle time calculation for the best architectural variant obtained.

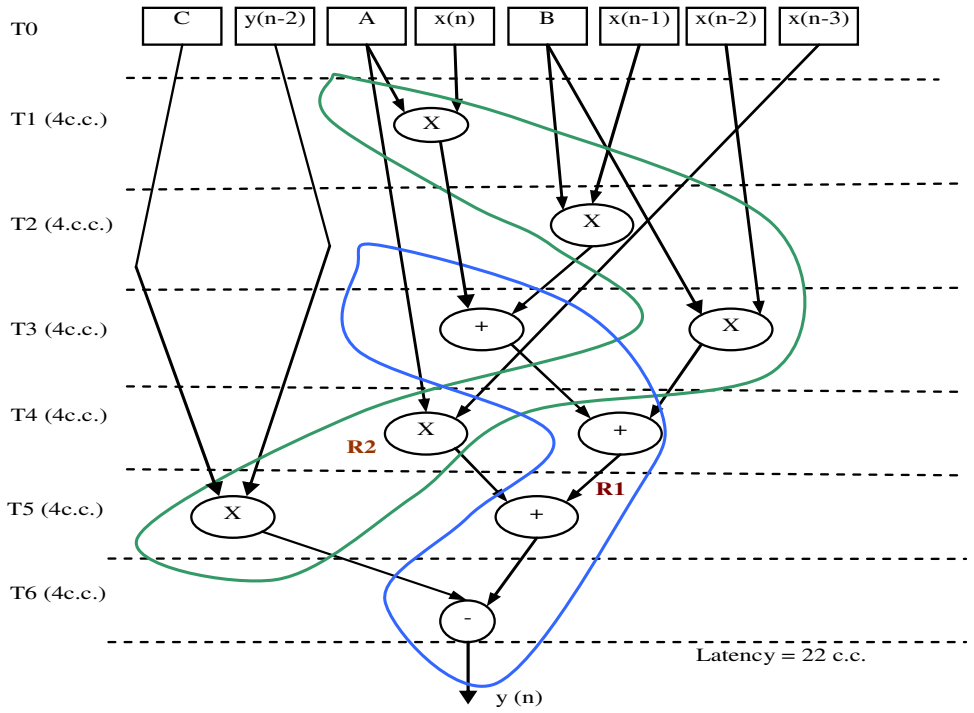


Figure6. The sequencing and binding graph for the best variant obtained

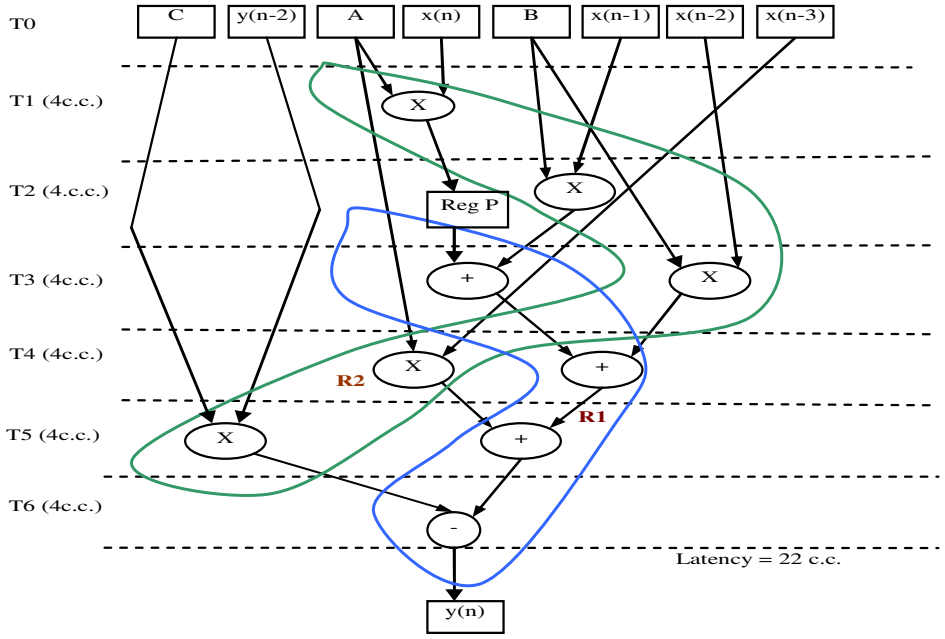


Figure7. Sequencing graph with data registers

8.3 Determination of Multiplexing Scheme.

The binding of the resources performed in Fig.7, enables a methodology to be formalized that incorporates the multiplexers and demultiplexers into the data path circuit of the system. The multiplexing scheme is one of the most important stages in high level synthesis design flow. Multiplexing scheme is a procedure for representing each system resource with respective inputs, outputs, operations and the necessary interconnections. This scheme highlights the actual usage

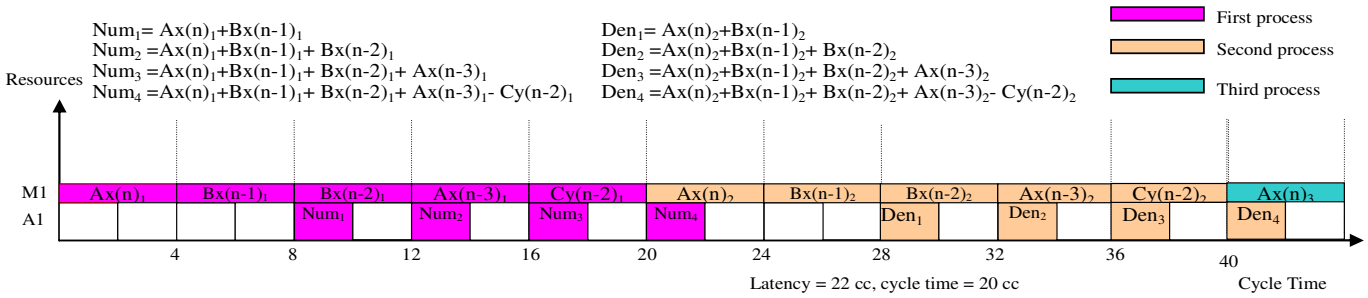


Figure8. Cycle time diagram for the best variant

of resources by the operands at different times while strictly adhering to the data dependency present. The illustration acts as an important guide for the designer to develop system block diagrams before developing the control unit structure for the data path. This scheme prevents any errors in the final hardware structure that could have eventually resulted in catastrophic consequences later during functioning. The control unit is responsible for the coordination of the data path of the system. Multiplexers and demultiplexers can be easily constructed and assigned to their respective inputs and outputs based on the multiplexing scheme, keeping in mind the dependency of the data. In this work, two functional resources viz. one adder/subtractor and one multiplier, perform different functions for the circuit. A multiplexing scheme for each of the above mentioned resources was developed as shown in Tables 3 and 4 respectively.

Table 3.Multiplexing scheme for Adder/subtractor resource (R1)

Time	Operation	Input 1	Input 2	Output
0	------	-------	-------	-------
1	**------**	-------	-------	-------
2	------	R2out	RegP	-------
3	+	R2out	R1out	R1in
4	+	R2out	R1out	R1in
5	+	R2out	R1out	R1in
6	-	------	------	RegY
7	------	------	------	------

Table 4. Multiplexing scheme for Multiplier resource (R2)

Time	Operation	Input 1	Input 2	Output
0	------	RegA	Regx(n)	-------
1	*	Regx(n-1)	RegB	RegP
2	*	Regx(n-2)	RegB	R1in
3	*	RegA	Regx(n-3)	R1in
4	*	RegC	Regy(n-2)	R1in
5	*	------	------	R1in
6	------	------	------	------
7	------	------	------	------

8.4 Determination of Block Diagram of the Data Path unit of the system

After the multiplexing scheme has been successfully performed, the next phase of the design flow is the development of the system block diagram. The system block diagram consists of two divisions, data path and the control path. The data path is responsible for the flow of data through the buses and wires after the operations have been performed by the components present in the data path circuit. Thus, the data path provides the sequence of operations to be performed on the arriving data, based on the intended functionality. The data path consists of registers for storage of data, memory elements such as latches for sinking of data in the next stage, as well as multiplexers and demultiplexers for preparation of data at run time by change of configuration. Last but not least, the data path unit also consists of functional resources which perform the operations on the incoming data. The block diagram for the benchmark application consists of two resources (an adder/subtractor and a multiplier) for executing their respective assigned operations. Another component of the system block diagram is the control unit or the controller. A centralized control unit controls the entire data path and provides the necessary timing and synchronization required by data traversing through the data path structure. The control unit acts as a finite state machine that changes its state according to the requirement of activating and deactivating the various elements of the data path at different instances of time. Based on the multiplexing scheme, the block diagram of the data path circuit was constructed to demonstrate design flow for the benchmark application as shown in Fig. 9.

8.5 Determination of Timing specification and Development of Control Unit

The next design stage is the development of the control unit structure, which is accountable for any mis-coordination in timing among the various elements of the data path. The function of

the controller is to activate and deactivate the different elements of the data path based on the timing specification determined for the objective function. This major unit prepares the data path units for the incoming data by changing the configuration to perform the next assigned function. For synchronous functioning of all data elements in the system the controller must respond to the requirement exactly at the right moment. Failure to activate or deactivate any functional block in the data path will result in fatal consequences in the system output. The determination of the timing specification from the control unit helps to create an error free structure of the controller. VHDL [18] was used here as the hardware description language for designing the control unit.

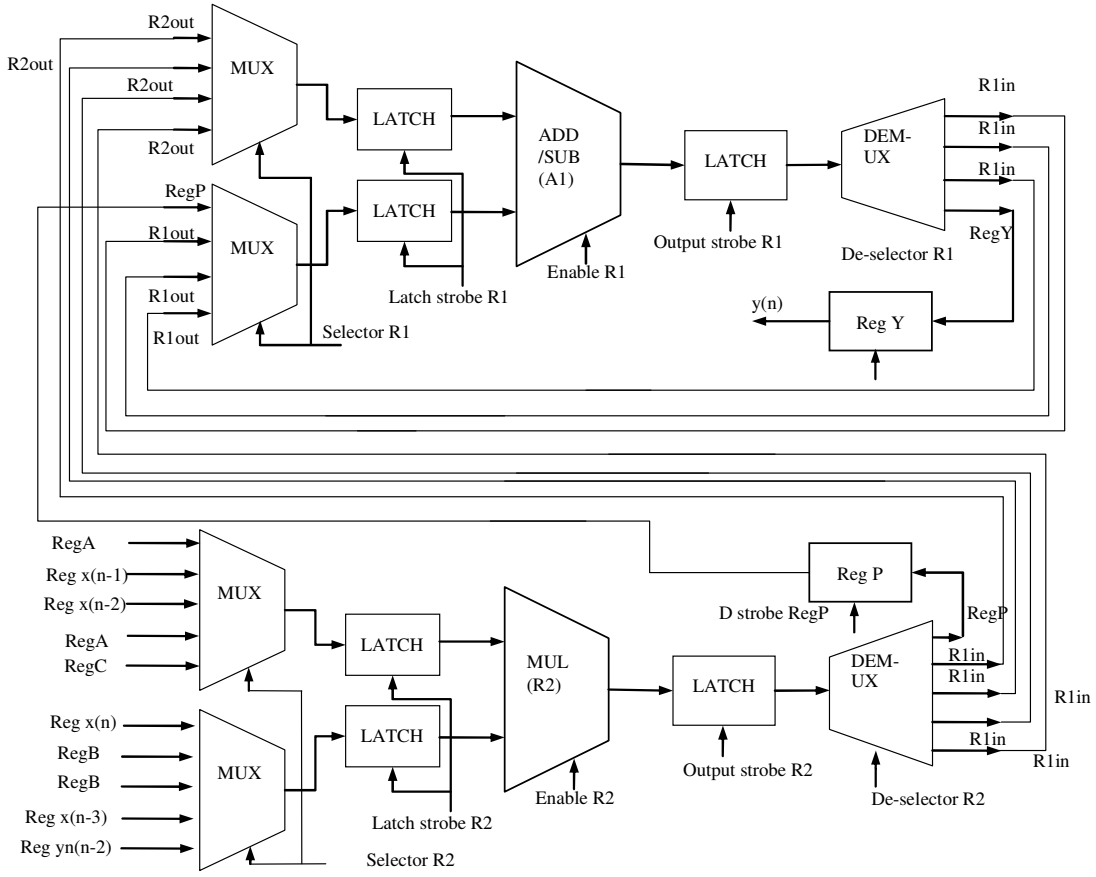


Figure9. Block diagram of the data circuit

The timing specifications data shown in Table 5A and Table 5B are developed with clock cycles placed in the Y-axis and the control signals placed in the X-axis. At every count the transition of the different control signals can be clearly observed. This facilitates in the description of the control structure in a hardware description language.

8.6 Development of the whole system at the RT-Level in Synthesis tool

After all the above steps have been completed successfully, the schematic structure of the device is ready for development in any of the synthesis tools available. All the components in the data path were described and implemented in VHDL before verification. Then the schematic

Table 5A.

Timing specification for the data path circuit

Adder (R1)						Multiplier (R2)					Strobes			
Latch strobe	enable	add_sub	Output strobe	Selector	Deselector	Latch strobe	enable	Output strobe	selector	Deselector	Strobes_register	Dstrobe_regP	Strobe_regY	Clock Cycles
0	0	0	0	00	0	0	0	0	000	000	0	0	0	0
									000		1			1
						1								2
							1		001	000				3
						0								4
														5
														6
								1						7
							0					1		8
						1								9
				00			1	0	010	001				10
						0								11
														12
														13
							0	1						14
1						1								15
	1	0		01	00		1	0	011	010				16

Table 5B.

Timing specification for the data path circuit

														17
0						0								18
														19
	0		1				0	1						20
1						1								21
	1	0	0	10	01		1	0	100	011				22
0						0								23
														24
														25
	0		1				0	1						26
1						1								27
	1	0	0	11	10		1	0		100				28
0						0								29
														30
														31
	0		1				0	1						32
1														33
	1	1	0		11									34
														35
0			1											36
	0												1	37

structure of the whole device was designed and implemented in Xilinx Integrated Software Environment (ISE) version 9.2i [19]. The schematic structure of the whole device as designed in Xilinx ISE 9.2i is shown in Fig. 10.

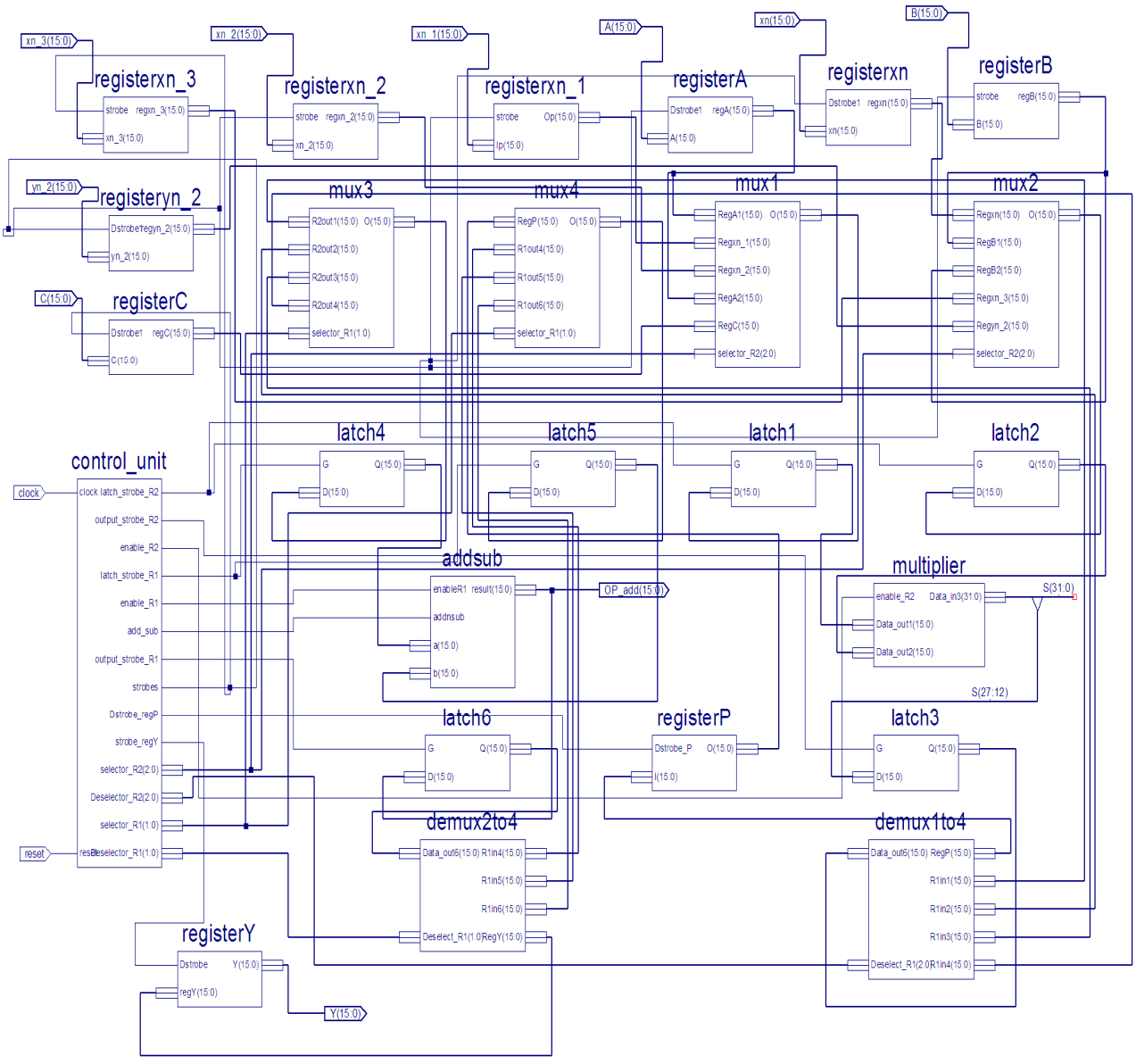


Figure10. Schematic view of the system (Xilinx ISE 9.2i)

Chapter 9

Results, Analysis and Implementation

9.1 Simulation Results

Using the proposed DSE method, the next step is to verify the device, designed through the proposed high level synthesis design flow, for its accurate functionality. The design was checked for a wide array of input vectors and results indicated that the design was successfully implemented in the Spartan 3E FPGA [20] [21]. Investigations suggested that the device was working perfectly. The successful result of the simulation shown in Fig.11 (see next page) suggested that the designed system was producing the expected output. After its successful implementation the design was imported in Synopsys tool [22] for flattening of the circuit. After flattening, the steps needed for floorplanning, power planning, placement and routing were executed in Cadence Encounter SoC [23]. Fig.12 shows the routing of the chip.

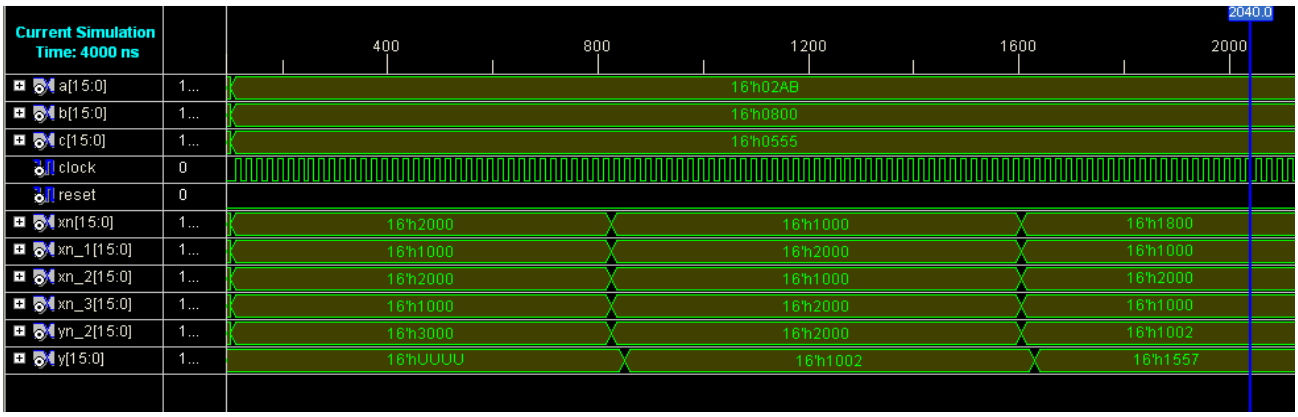


Figure11. Simulation result for the benchmark application

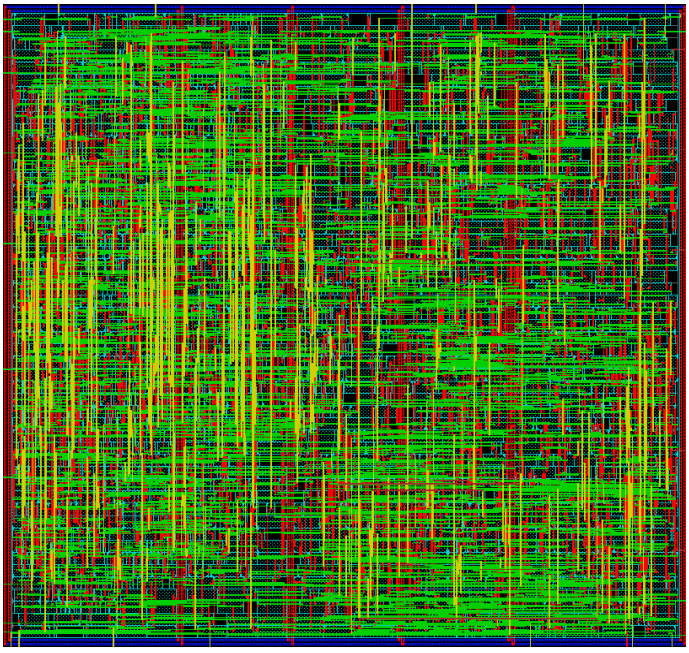


Figure12. Final routing of the chip (Cadence encounter SoC)

9.2 Comparative study of the Proposed Multi parametric optimized Design Space Exploration methodology with the current existing approach

For determination of the optimal architecture, design space exploration requires elaborate analysis and evaluation of the architectural variants (design points). Before selecting the optimal architecture, the border variant of architecture for both the performance (execution time and area/

power) parameters needs to be found separately. Binary search conducted on the arranged design space (increasing or decreasing) leads to the border variant by taking into account the operating constraints for that parameter (such as constraint for execution time or constraint for area/power.) The proposed DSE approach uses binary search after the arrangement of the design space using the priority factor method. The search of the optimal architecture requires only $\log_2 \prod_{i=1}^{n} v_{Ri}$ .Where 'n' = number of type of resources and 'v_{Ri}' is the number of variants of resource 'Ri'. On the contrary, the exhaustive search checks for $\prod_{i=1}^{n} v_{Ri}$ architectural variants during optimal architecture selection while satisfying all operating constraints. In this design space exploration approach and in the design flow three performance parameters have been used for optimization. The execution time and power are the parametric constraints and area is the optimization parameter. Hence, the searching has to be repeated for both the parameters to determine the border variant.

Therefore the total number of architecture evaluations using exhaustive search is given as:

$$M * \prod_{i=1}^{n} v_{Ri} \ .$$

And total number of architecture evaluations using the proposed method is given as:

$$M * \log_2 \prod_{i=1}^{n} v_{Ri} \ .$$

Here, 'M' denotes each performance parameter. In this case the value of 'M' is two because there are two performance parametric constraints.

9.3 Experimental Results of the Proposed Exploration process for High Level Synthesis Benchmarks

The proposed approach was applied on various benchmarks to check the acceleration obtained through this DSE method. Results indicated massive acceleration in the speedup compared to the exhaustive approach. The results of proposed design space exploration framework for the benchmarks are illustrated in Table 6. Fig.13 illustrates the speedup results when using the proposed approach for DSE compared to the exhaustive variant analysis. Table 7 and Table 8 shows the comparative study of the proposed approach with the approach in [2] that utilizes hierarchical structure for evaluation of design space, for many realistic HLS benchmarks. Fig.14 shows the comparison of the number of architectural variants analyzed between the current existing approach and the proposed approach, while Fig.15 represents the speedup attained by the proposed method compared to the hierarchy tree structure approach using ACG [2]. Investigations of the results reveal that the proposed approach is capable of drastically improving the acceleration time for finding the optimal architecture compared to the current approach shown in [2].

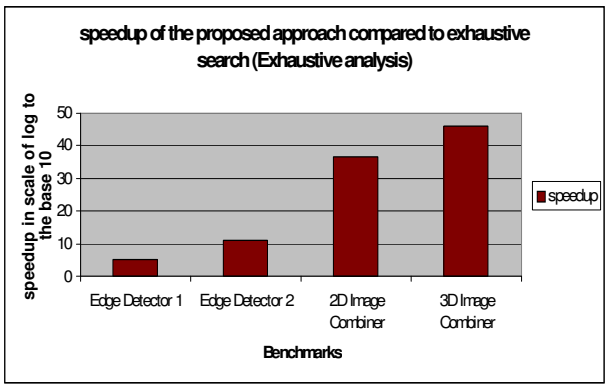


Figure13. Representation of the speedup the scale of log to the base 10 (log $_{10}$ speedup) compared to the exhaustive variant analysis

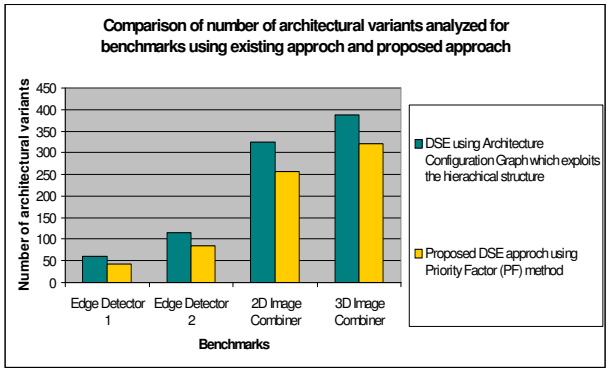


Figure14. Comparison of the number of architectural variants analyzed between the current existing approach and the proposed approach

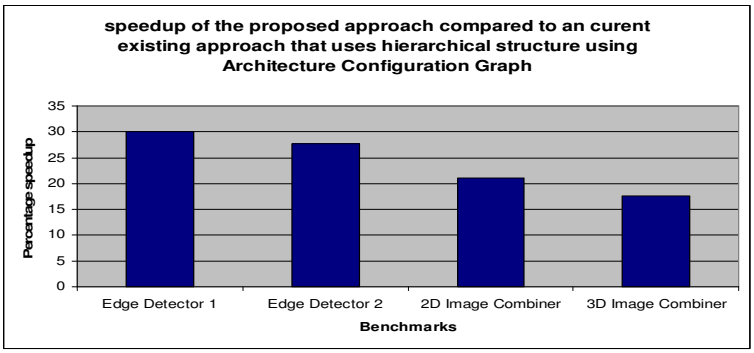


Figure15. Representation of the speedup attained by the proposed DSE compared to a current approach that uses hierarchical structure

Table 6. The results of the proposed DSE approach for the Benchmarks

Benchmark	Type of resources	Number of variants of each resource	Total possible architecture for exhausted search	Evaluated architectures using the proposed DSE strategy	Speedup obtained	Log_{10}(speedup)
Edge Detector 1	7	8	$4.2x10^{6}$	42	$9.9\ x10^{4}$	4.995635
Edge Detector 2	14	8	$8.8\ x10^{12}$	84	$1.05\ x10^{11}$	11.02119
2D Image Combiner	32	16	$6.81\ x10^{38}$	256	$2.66\ x10^{36}$	36.42488
3D Image Combiner	32	32	$2.92\ x10^{48}$	320	$9.13\ x10^{45}$	45.96047

For example, in the case of edge detector 1, when the types of resources are less than the other benchmark applications, then the proposed approach provides a percentage speedup up to 30% compared to the existing approach described in [2]. Further for large well known high level synthesis benchmarks such as Discrete Wavelet Transformation (DWT) [25], Differential Equation Solver (HAL) [24], Elliptic Wave Filter (EWF) [24], Auto Regressive Filter [26][27] and MPEG Motion Vector [28], the speedup results compared to the current approach [2] is over 30 %. Hence based on the experiments performed on the benchmarks it can be concluded that the proposed approach for DSE is able to provide increased acceleration when compared to existing DSE approaches.

Table 7. Comparative study of the proposed DSE approach with one of the current approaches [2]

Benchmark	Architecture evaluation using Hierarchical arrangement of the ACG (Number of total architecture)	Proposed Priority Factor method (Number of architecture)	Speed up of the proposed approach compared to the existing approach (%)
Edge Detector 1	60	42	30
Edge Detector 2	116	84	27.58621
2D Image Combiner	324	256	20.98765
3D Image Combiner	388	320	17.52577

Table 8. Comparative study of the proposed DSE approach with one of the current approaches [2] for 5th order WDF benchmark

Benchmark	Type of resources	Number of variants of each resource		Total possible architecture for exhausted search	Architecture evaluation using Hierarchical arrangement-ACG [2] (Number of total architecture)	Proposed Priority Factor method (Number of architecture)	Speed up of the proposed approach compared to the existing approach
		Adder	Multiplier				
5th order Wave Digital Filter (WDF)	2	26	8	416	24	16	33.33 %

Table 9. Experimental results of comparison between the proposed DSE approach with the current approach [2] for large benchmarks

Benchmarks	Total possible architecture in the design space for exhaustive search	Architecture evaluation using Hierarchical arrangement of the ACG with binary search [2] (Number of variants analyzed)	Architecture evaluation using Proposed Priority Factor method (Number of architecture)	Percentage speed up compared to current DSE approach [2]	Speedup using proposed approach compared to the exhaustive search
Discrete Wavelet Transformation (DWT) [25]	432	26	18	30.76 %	95.88 %
Differential Equation Solver (HAL) [24]	180	26	14	46.15 %	92.22%
Elliptic Wave Filter (EWF) [24]	156	20	14	30 %	91.02 %
Auto Regressive Filter (EWF) [26][27]	288	24	16	33 %	94.44 %
MPEG Motion Vector (MMV) [28]	756	28	20	28.57 %	97.35 %

Chapter 10

Conclusion and Future Work

The thesis has presented a new framework for rapid and accurate design space exploration. The approach was successful in laying the foundation for exploring the design points from the architecture design space according to the performance objective and intended functionality. The presented method determines the priority factor of each resource for system designing. After the architectural design points were organized in increasing and decreasing order based on the priority factor calculated, the procedure for applying the search algorithm became very simple. As a result the proposed approach was able to drastically reduce the number of architectural variants to be analyzed for selection of the system architecture. The proposed mechanism for DSE was able to resolve conflicting objectives in DSE, by concurrently maximizing the accuracy needed in the evaluation of the design point as well as minimizing the time expended in design space assessment. This approach is applicable to all system architectures based on modules with known performance requirements and system specifications. The priority factor functions for the DSE framework were first deduced mathematically and then used in the proposed high level

synthesis design flow to highlight its success in finding the best architecture.. Formalizing the design methodology for multi parametric HLS is useful for many industrial projects and modern automated high level synthesis tools. Based on the experiments performed on the benchmarks, it can be concluded that the proposed approach for DSE is able to provide increased acceleration compared to the current existing approaches for DSE in HLS.

Due to the enormous growth of the design complexity, the gap between the electronic system level and the register transfer level must be filled. A complete design methodology is described in this thesis, which allows automation of the high level synthesis design flow. This design flow can provide the foundation for a fully automated high level synthesis tool that performs not just design space exploration but also synthesis. This is the first work known to the author that elaborates the design steps and exclusively concentrates on the design flow methodology useful for multi parametric optimization requirement using a fast DSE method. Compared to the traditional high level design flow, the modern high level flow must be more efficient, and multi parameter optimized. The design flow should have formalized steps to be followed for designing the system without making errors. The design approach for high level synthesis considering multi parametric optimization shown here in this thesis contributes significantly for designing Application Specific Integrated Circuits (ASIC) and Application Specific Processors (ASP) used in system-on-chip design. Any computing core developed through this method could be efficiently used in real time systems where the time of execution is the major decisive factor for successful functioning of the system.

New standards in the area of communications, signal processing and multimedia have led to a wide array of applications demanding high performance at minimal power consumption. The modern generations of massive power hungry portable devices like Mp3 and Mp4 players are

required to provide high performance at low power. High performance output means performing more operations per clock cycle, i.e. more power usage. However, this demand is in total contradiction to the current trend of portable devices which should operate on low power. Although parameters like execution time and power are contradictory, both are equally vital for system design. In such cases, this high level design flow for multi parametric optimization requisite will work wonders for the designer. While describing the design flow all these factors were taken into account when designing the system. Investigations revealed that the designed system met all the specified stringent operational constraints for execution time, hardware area, power dissipation. The results were in accordance with the technical specifications provided. Thus the flow successfully bridged the gap from the ESL to the RTL.

Scope of Future Work

There is much potential in the area of design space exploration and high level synthesis to improve the search time for finding the optimal design architecture, and thereby accelerate the speedup of the exploration process. The developed design space exploration approach for high level synthesis can be improved further by decreasing the number of architectural variants to be analyzed during the exploration process. Reducing the analysis of the architectural variants directly reduces the search time and hence will help in fast marketing of the product. Another aspect of high level synthesis, which also has significant potential for improvement, is the optimization of many other parameters such as reliability, temperature etc., which stills lies in the nascent stage of development.

Refereed Publications

Refereed Journals

1. **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", Microelectronics Reliability, Elsevier, In press, Corrected Proof, 2009, doi:10.1016/j.microrel.2009.11.015, Available online December 22, 2009.

2. **Anirban Sengupta**, Reza Sedaghat, Zhipeng Zeng, "A High Level Synthesis Design Flow from ESL to RTL with multi-parametric optimization objective", IETE Journal of Research, 2009, Submission no: IETE JR_523_09 (under review)

3. Zhipeng Zeng, Reza Sedaghat, **Anirban Sengupta**, "A multi parametric optimized High level synthesis Design Flow for multi objective VLSI and SoC designs", Integration VLSI journal, Elsevier, 2009, Submission no: VLSI-D-09-00120 (under review)

Refereed Conferences

4. **Anirban Sengupta**, Reza Sedaghat, Zhipeng Zeng, "Hardware Efficient Design of speed optimized Power stringent Application Specific Processor", IEEE 21st International Conference on Microelectronics (ICM), Published in the 21st Conference Proceedings,

2009, Pages: 167-170, IEEE Catalogue no: 978-1-4244-5815-8/09, Presented on December 22, 2009, Morocco.

5. Zhipeng Zeng, Reza Sedaghat, **Anirban Sengupta**, "A Novel Framework of Optimizing Modular Computing Architecture for multi objective VLSI designs", IEEE 21st International Conference on Microelectronics (ICM), Published in the 21st Conference Proceedings, 2009, Pages: 322-325, IEEE Catalogue no: 978-1-4244-5815-8/09, Presented on December 21, 2009, Morocco.

6. **Anirban Sengupta**, Reza Sedaghat, Zhipeng Zeng, "Rapid Design Space Exploration for multi parametric optimization of VLSI designs", 43rd IEEE International Symposium on Circuits and Systems (ISCAS), 2010, Accepted for publication in the 43rd Proceedings of the Conference, Paris, France, Article # 2016, To be presented on June 2, 2010 in Paris.

7. **Anirban Sengupta**, Reza Sedaghat, "Fast Design Space Exploration for Multi Parametric Optimized VLSI and SoC Designs", 15th Asia and South Pacific Design Automation Conference (ASP-DAC), 2010, Accepted for Poster Presentation in the Conference, Poster ID: 26, Taiwan. To be presented on January 19, 2010 at Taipei, Taiwan. (http://www.asp-dac.itri.org.tw/aspdac2010/student_forum/index.html).

8. Zhipeng Zeng, Reza Sedaghat, **Anirban Sengupta**, "A Framework for Fast Design Space Exploration using Fuzzy search for VLSI Computing Architectures", 43rd IEEE International Symposium on Circuits and Systems (ISCAS), 2010, Accepted for

publication in the 43rd Proceedings of the Conference, Paris, France, Article # 2019. To be presented on June 2, 2010 in Paris.

9. Summit Sehgal, Reza Sedaghat, **Anirban Sengupta**, Zhipeng Zeng, "Multi Parametric Optimized Architectural Synthesis of an Application Specific Processor", IEEE 14th International CSI Computer Conference (CSICC), Published in the 14th Conference Proceedings, 2009, Pages: 89-94, IEEE Catalogue no: 978-1-4244-4262-1/09. (Also available on IEEE Xplore)

10. Summit Sehgal, Reza Sedaghat, **Anirban Sengupta**, "FAULT MONITORING TRANSFORMER RELIABILITY ASIC DESIGN BASED ON RINGING EFFECT SIGNATURE ANALYZER" , IEEE 23rd Canadian Conference on Electrical and Computer Engineering (CCECE), 2010. (under review)

References

[1]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.

[2]Kirischian, L., Geurkov, V., Kirischian, V. and Terterian, I. (2006)'Multi-parametric optimisation of the modular computer architecture', Int. J.Technology, Policy and Management, Vol. 6, No. 3, pp.327–346.

[3]I. Das. A preference ordering among various Pareto optimal alternatives. Structural and Multidisciplinary Optimization, 18(1):30–35, Aug. 1999.

[4]Alessandro G. Di Nuovo, Maurizio Palesi, Davide Patti, Fuzzy Decision Making in Embedded System Design," Proceedings of 4th International Conference on Hardware/Software Codesign and System synthesis, 2006, October 2006, Pages: 223-228

[5]J. C. Gallagher, S. Vigraham, and G. Kramer,"A family of compact genetic algorithms for intrinsic evolvable hardware," IEEE Trans. Evol. Comput., vol. 8, no. 2, pp. 1–126, Apr. 2004.

[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, vol. 10, no. 3, June 2006.

[7]E. Torbey and J. Knight, “High-level synthesis of digital circuits using genetic algorithms,” in Proc. Int. Conf. Evol. Comput., May 1998, pp.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, pp. 284–287.

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

[10] Williams, A. C., Brown, A. D. and Zwolinski, M, "Simultaneous Optimisation of Dynamic Power, Area and Delay in Behavioural Synthesis", IEE Proceedings Computers and Digital Techniques, 2000, Volume: 147, Issue: 6, On page(s): 383-390.

[11]De Micheli,G.(1994) Synthesis and Optimization of Digital Systems, McGraw-Hill Inc., 580 p.

[12] McFarland, Parker, A.C, Camposano, R. “Tutorial on high-level synthesis” Proceedings of the 25th ACM/IEEE Design Automation Conference, 1988, Atlantic City, New Jersey, United States, Pages: 330 – 336.

[13]Giuseppe Ascia, Vincenzo Catania, Alessandro G. Di Nuovo Maurizio Palesi, Davide Patti, “Effcient design space exploration for application specific systems-on-a-chip” Journal of Systems Architecture 53, Science Direct, Elsevier, 2007, Pages: 733-750

[14]Ron Larson, Robert P.Hostetler, Bruce H.Edwards, “Calculas with Analytic Geometry”, Houghton Mifflin Company, Eighth Edition, 2006, Pages: 918-919

[15]S Salivahanan, A Vallavaraj and C Gnanapriya, "Digital Signal Processing", Tata McGraw-Hill Publishing Company Limited, 2006, pp. 439- 444.

[16]Pierre G. Paulin and John P. Knight, "Scheduling and Binding Algorithms for High-Level Synthesis, 26th conference on Design Automation, 1988, Pages: 1-6

[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]Brown,S. and Vranesic, Z. (2005) Fundamentals of Digital Logic with VHDL Design, 2nd ed., New York, NY: McGraw-Hill, 940 p.

[19]ISE 9.2i Quick Start Tutorial, Xilinx ISE 9.2i , Software Manuals and Help, http://www.xilinx.com/support/sw_manuals/xilinx92/download/

[20] http://www.xilinx.com/publications/xcellonline/xcell_54/ xc_ssinterface54.htm

[21] http://www.xilinx.com/publications/xcellonline/xcell_54/xc_pdf/ xc_ssinterface54.pdf

[22]http://www.synopsys.com/Tools/SLD/AlgorithmicSynthesis/Pages/default.aspx]

[23] http://www.cadence.com/support/university/Pages/default.aspx

[24] http://www.cbl.ncsu.edu/benchmarks/.

[25] 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), pp. 1377– 1380 (2007)

[26]Antola, A., Ferrandi, F., Piuri, V., Sami, M.: Semiconcurrent error detection in data paths. IEEE Transactions on Computers 50(5), 449– 465 (2001)

[27] Antola, A., Piuri, V., Sami, M.: A low-redundancy approach to semi-concurrent error detection in datapaths. In: Proceedings of the Design Automation and Test in Europe, pp. 266–272 (1998)

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

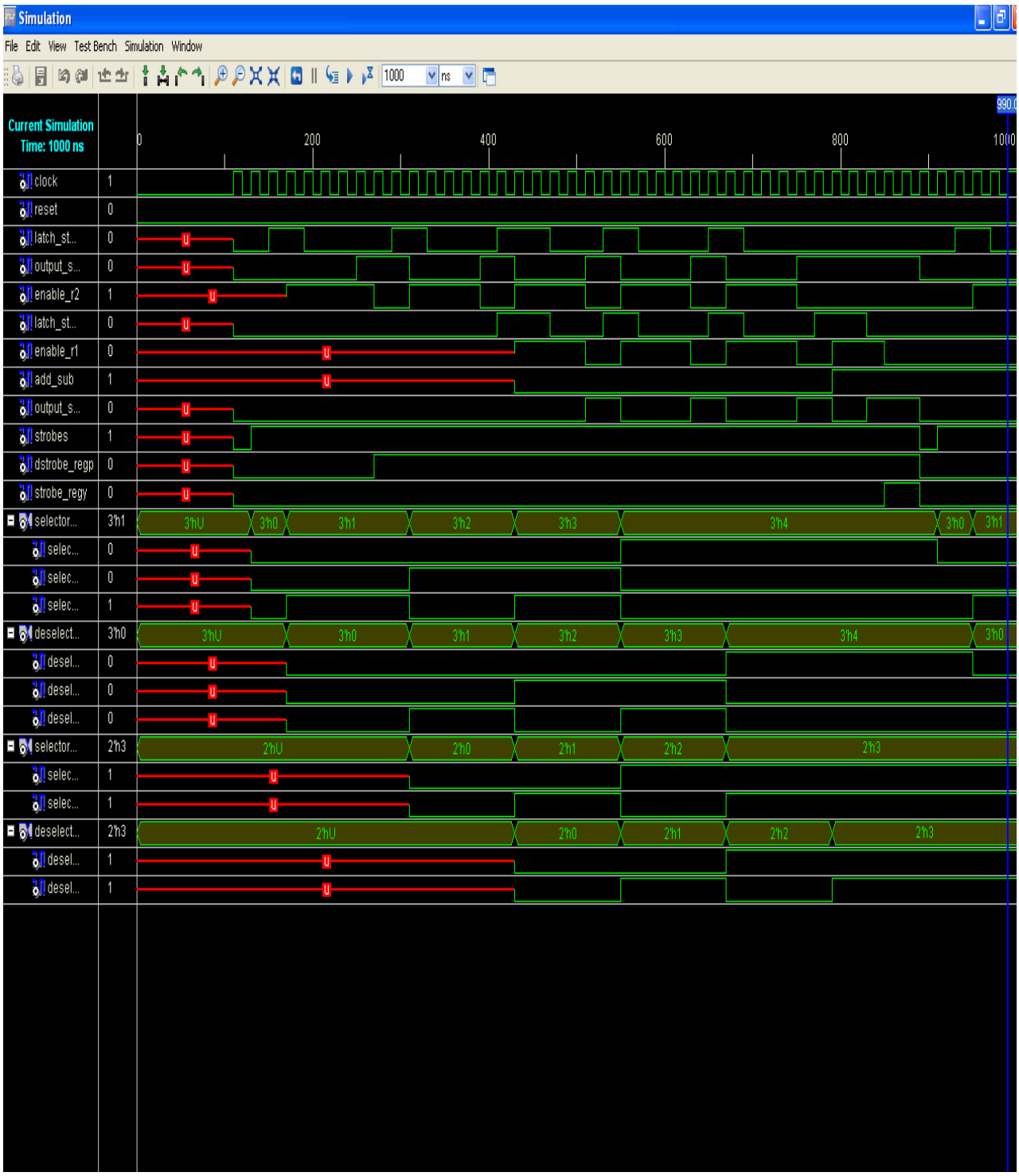
[29] Cheng-Tsung Hwang, Jiahn-Humg Lee, and Yu-Chin Hsu, “A Formal Approach to the Scheduling Problem in High Level Synthesis”, IEEE Transactions on computer-aided design, vol. 10**,** no. 4, april 1991.

[30] 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

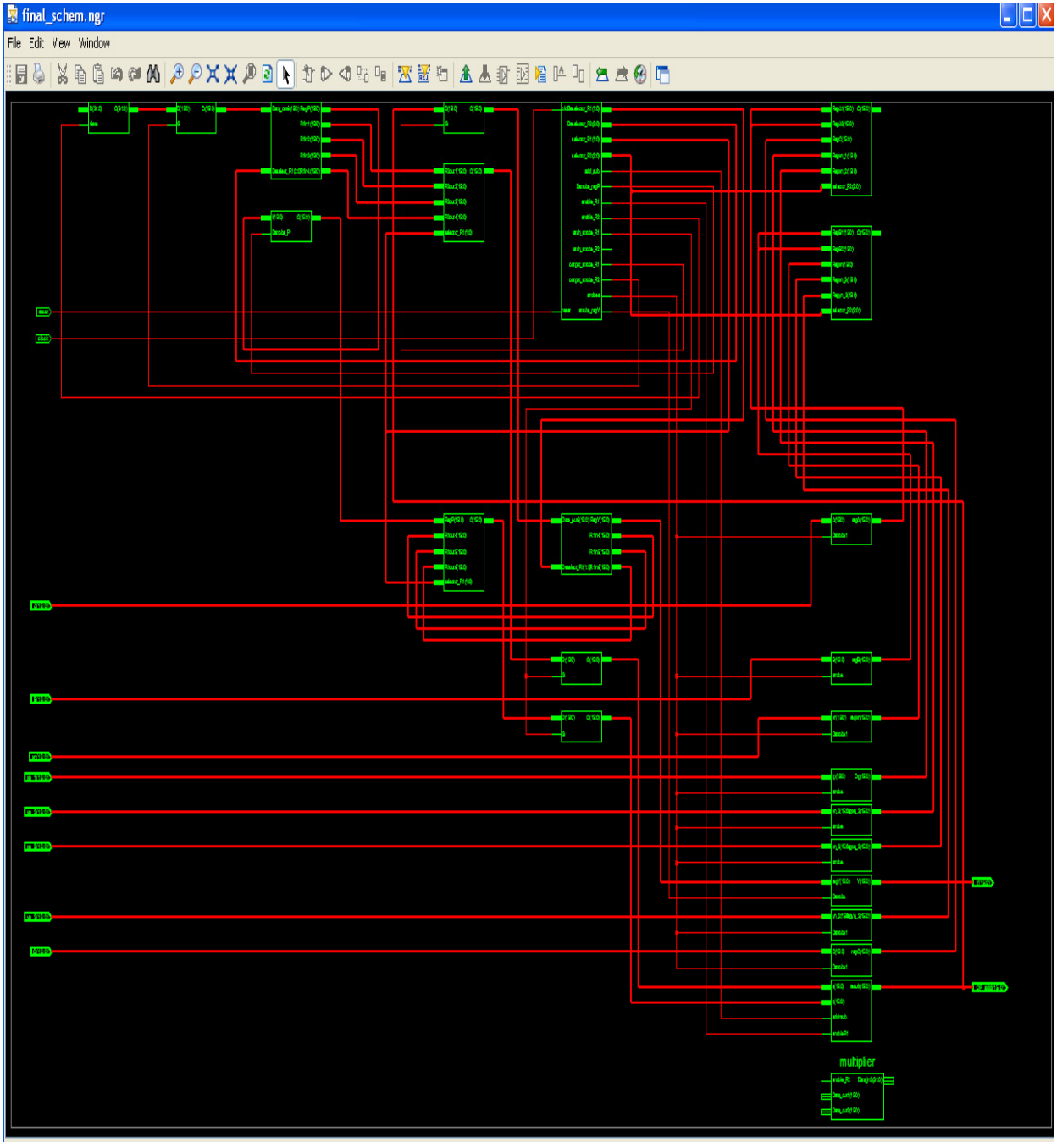
[31] Maurizio Palesi, Tony Givargis, “Multi-Objective Design Space Exploration Using Genetic Algorithms”, Proceedings of the tenth international symposium on Hardware/software codesign”, Estes Park, Colorado, 2002, Pages: 67 – 72.

APPENDIX

1. The Control Unit Simulation Results



2. The RTL Schematic Generated in Xilinx Integrated Software Environment (v9.2i)



3. The Routing information for Spartan 3E FPGA generated in Xilinx Integrated Software Environment (v9.2i)