1-1-2011

Performance Driven Design Space Exploration for Multi-Objective Optimization Using Stability in Competition

Summit Sehgal
Ryerson University

Recommended Citation

Sehgal, Summit, "Performance Driven Design Space Exploration for Multi-Objective Optimization Using Stability in Competition" (2011). *Theses and dissertations.* Paper 798.

PERFORMANCE DRIVEN DESIGN SPACE EXPLORATION FOR MULTI-OBJECTIVE OPTIMIZATION USING STABILITY IN COMPETITION

By

Summit Sehgal

Bachelors of Engineering

Electronics and Electrical Engineering

Kurukshetra University

Kurukshetra, India, 2007

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2011

Author's Declaration

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

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

* Signature

____

Summit Sehgal

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

____

Summit Sehgal

ABSTRACT

Title of Thesis:

PERFORMANCE DRIVEN DESIGN SPACE EXPLORATION FOR MULTI-OBJECTIVE OPTIMIZATION USING STABILITY IN COMPETITION

Thesis Submitted By:

Summit Sehgal, Master of Applied Science, 2011

Optimization Problems Research and Application Laboratory (OPR-AL)

Electrical and Computer Engineering Department, Ryerson University

Multi Parametric Design Space Exploration (DSE) for optimal micro-architecture synthesis is an extremely complex yet crucial stage in embedded systems development. Often it is very time complex to find the best suitable configuration to map the inherently contradictory performance parameters into systems silicon real estate. Owing to its exponentially exploding design space and multi way combinatorial mapping, DSE has proven to be notoriously hard and intractable for VLSI CAD tools. The presented work introduces a highly scalable and generalized analytical approach to identify the best configuration of systems architecture while maintaining prime accuracy resolution. This DSE approach coupled with *Stability in Competition* principles has been applied to a number of well known benchmark High Level Synthesis (HLS) applications, with an impressive 71.80% aggregate speedup and results being more pronounced for larger design space HLS applications.

Acknowledgement

I would like to acknowledge and thank my supervisor, Dr. Reza Sedaghat, for his consistent guidance and support throughout this research work.

I shall also remain highly obliged to OPR-AL lab members for their support and co-operation. I am also very thankful to IT and Support staff in the EE department for their help with the design and development of the tool set chain.

I shall always remain highly obliged to my family members for the consistent support and endless cooperation and for the time I could not spend with them for the sake of this research work.

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	Area of the resources
$\mathbf{R_i}$	The resources available for system designing
$\mathbf{R_{clk}}$	The clock oscillator used as a resource providing the necessary clock frequency to the system
$\mathbf{N_{Ri}}$	The number of resource R_i
$\mathbf{K_{Ri}}$	The area occupied per unit resource ‘Ri’
$\mathbf{\Delta}$	Delta operator
n	Functional resources
L	Latency of execution
$\mathbf{T_c}$	Cycle time of execution
N	Number of data elements to be processed
$\mathbf{T_{Ri}}$	Number of clock cycles needed by resource ‘Ri’
$\mathbf{T_p}$	Time period of the clock
$\mathbf{p_c}$	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
$\mathbf{P_{optimal}}$	The constraint for Power Consumption
$\mathbf{T_{optimal}}$	The constraint for Execution Time
$\mathbf{v_{Ri}}$	Number of variants of resource ‘Ri’
M	Each Performance Parameter
s.t.	Such That

DS	Data Structure
HLS	High Level Synthesis
DSE	Design Space Exploration
PF	Pareto Front
PFA	Pareto Front Arithmetic
GA	Genetic Algorithm
CFDG	Control Flow Data Graph
ACG	Architecture Configuration Graph
SVM	Support Vector Machine
AI	Artificial Intelligence
RTL	Register Transfer Logic
MDMKP	multi-demand multi-constrained knapsack problem
∂	Partial Derivative Operator
IP	Intellectual Property
COTS	Custom of the shelf

Chapter 1

Introduction

1.1 Overview

For the design of modern embedded systems, high data processing performance and low power consumption numbers play the most important role [11,25,33,4,30,12]. The latest pursuit of converging data applications and ever increasing availability of silicon real estate per square inch has made the optimized selection of system resources increasingly important. The objective performance gains with aggressive parallelization and clocking at higher frequencies is becoming archaic; the need of modern reconfigurable systems is to adapt dynamically to reflect the changing operational conditions onto the system's micro-architecture. A typical embedded system development from conceptualization to physical realization goes through a series of steps and abstraction stages with each stage revealing more detailed and relevant information to the succeeding stage[14,22]. This kind of hierarchal and modular top-down design implementation is becoming the universal *de facto* industry standard [17,10,21,13] for complex real time embedded application development for its predictable design flow and information traceability. Fig. 1 shows the overview of typical top-down embedded systems design flow stages along with the abstraction stages from the behavioral/algorithm level to structural/physical level.

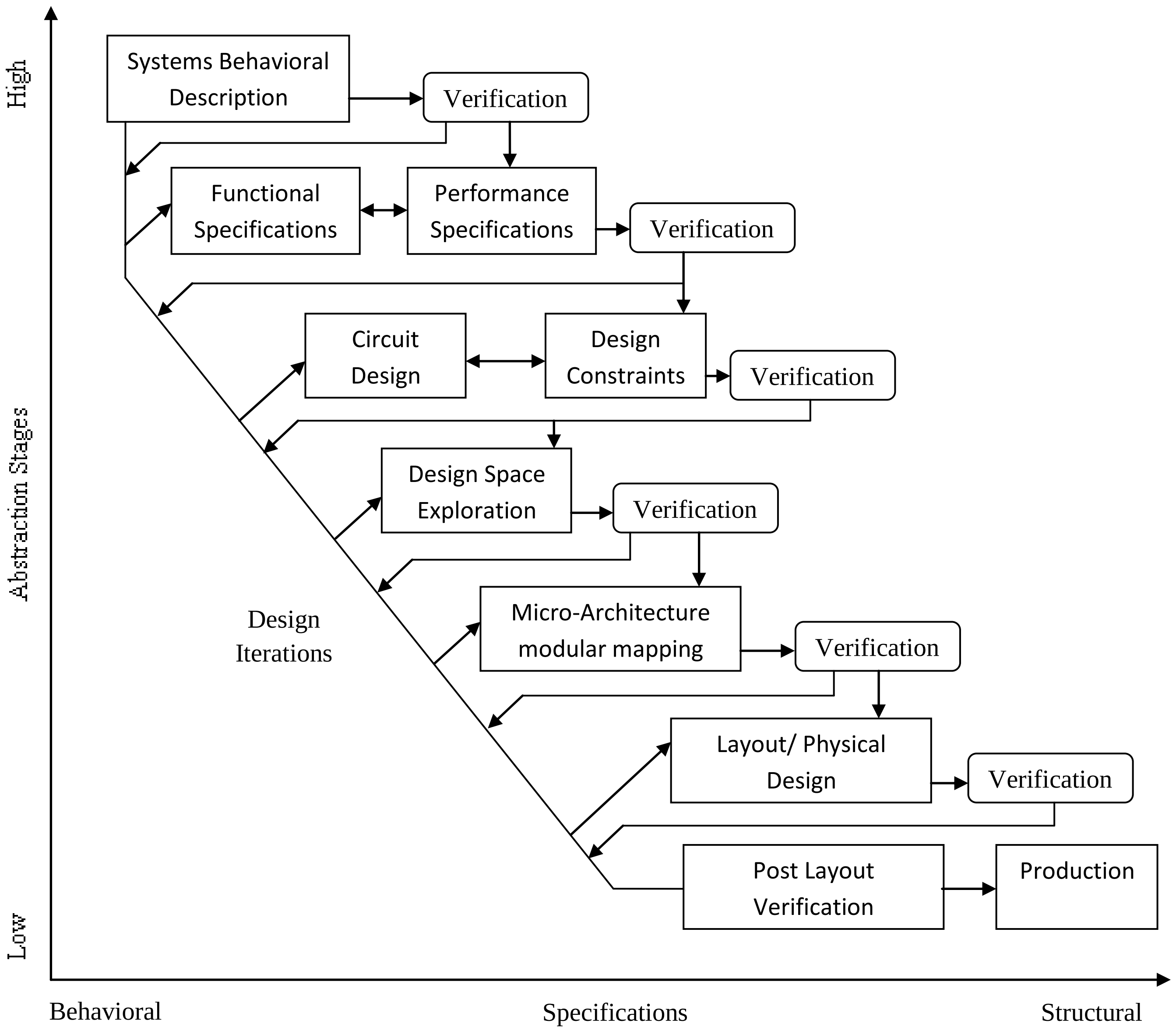

Figure 1. Top-down VLSI embedded systems design flow from behavioral description to in-silicon implementation

High Level Synthesis coupled with DSE effectively bridges the gap between behavioral and structural description of a system's micro-architecture [13,12]. This exposes a major design challenge: how to efficiently search and arrive at the best suitable systems configuration in practical time lines while maintaining prime solution accuracy given its exponentially exploding design space and multi way output-input combinatorial mapping. Design Space Exploration (DSE) can effectively address this important design stage to arrive at suitable systems micro architecture configuration. DSE yields maximal advantage when applied at the initial stages of the system design but is rarely exploited to its full potential for the lack of implementation information available in the early design stages. The next sub-section gives an introduction on Design Space exploration.

1.2 Design Space Exploration Overview

Design Space Exploration is a set of framework and efficient search strategies to overcome the design space complexities. As the design space variables can be combine in a combinatorial manner, the design space candidate solution set variants grows exponentially. A random or an exhaustive search through an unorganized design space can take infinite time to arrive at an acceptable solution even for a mediocre size application. Therefore Design Space Exploration looks at a set of tools and mathematical/ algorithmic search strategies to limit the design space inherent complexity. The problem of efficient design space exploration compounds if the subjected application poses totally different characteristics along various objectives of a multi-objective optimization problems. The selection of the optimal system configuration given a number of contradictory performance requirements poses an imposing design problem. Advanced Design Space Exploration techniques can provide solutions to this complex problem. Typically, DSE encompasses design space modeling and performance evaluation, system level tradeoffs to most effectively capture the targeted application in silicon real estate [11,13,5,7,36,12]. Although effective, DSE is often cursed by high dimensions of the exponentially exploding design space for practical size applications with an enormous amount of architectural configurations possible for each modular implementation. As each of the possible system architectural configurations characterizes itself with varying performance space relationship, it soon becomes beyond human comprehension to evaluate

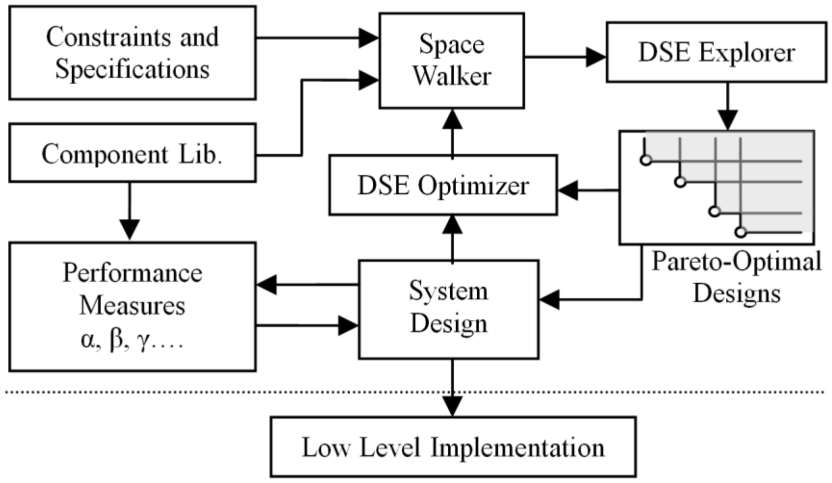

Figure 2. DSE exploration framework including DSE explorer, DSE Optimizer and Space walker

and perform wise resource utilization and system level trade-offs to achieve the desired operational characteristics.

This research develops an efficient, structured and traceable DSE technique that makes it possible to perform system level evaluations and considerate architectural candidate selection from a large pool of probable candidate solutions. DSE is thus one of the most imperative and beneficial phases of embedded project development, leveraging maximal tradeoffs and performance gain benefits, especially when applied at the early stages of system development. Fig. 2 shows the block level view of the proposed DSE exploration framework. The figure shows different phases of DSE exploration including performance estimation; candidate set generation; spacewalk (DSE Explorer) and optimization (DSE Optimizer) strategies as will be discussed in detail in the Design Exploration chapter.

Space walk tools are designed to automatically generate possible design configurations variants for the given high level application task [36,34,19]. The DSE explorer takes the input from the Space Walk tools and decides on the direction and the best possible way for further exploration. The DSE Optimizer subjects the possible design configurations to the systems constraints and the bounding conditions. The DSE Optimizer also decides on the suitability of a particular design configuration to the others. The DSE Optimizer also consists of the terminating condition of the exploration loop which, when met or exceeded, terminates the exploration process. The proposed DSE optimizer in the presented research also incorporates a novel design technique that decides the order of design variable exploration as well as the extent of local resource exploitation.

1.3 Related Work

Design Space Exploration (DSE) techniques for multi-objective optimization problems have been an active area of research, which is receiving even more importance for its power aware high level synthesis applications. In recent times extraction and evaluation of the *pareto optimal* [19,33,7,17,36,12] design points have emerged as the most promising and time efficient way to overcome the design space complexities. In [7], the authors have developed Pareto Front Arithmetic (PFA) techniques to explore very large design space in a time efficient manner. Their method exploited the hierarchal problem structure for exploring the set of pareto-optimal design solutions. The similar approach has been developed by authors in [2], suggesting a few superior design points suffice the optimal variant selection from a large search space. A slightly different approach by the researchers in [17] have suggested the importance of order in design space exploration, which helps in design space feasibility studies and deciding the preference among various pareto-optimal design points. Although the authors in [2,17] have established the importance and applicability of pareto-optimal design set in selection and search for suitable design solution, their developed methodologies lack the generalized analytical formulation to support the underlying HLS principles. This research addresses the underlying analytical framework with

Priority Factors (PF) that can rigorously be applied to large design space problems. Additionally, the presented work develops an analytically exact method to utilize the most suitable range of the ordered design space variables making the search more accurate in terms of solution resolution, faster, as well as more easily traceable on to the design space configurations. Another noteworthy approach based on pareto-optimal design configuration is developed by authors in [36], addressing the multi-dimensional output-input space mapping technique to extract the pareto-optimal design points for analog circuit topologies. Although the research showed promising results in selection of optimal design variants in performance space it fails to trace back the performance space parameters to the design space variables, thus making the mapping exhaustive. In [18], the authors have employed evolutionary algorithms such as Genetic Algorithm (GA) to yield the non dominated design points for the solution front selection and progression through an unarranged design space. Their research shows very impressive results but needs manual intervention converting the HLS problem into chromosomes representation as well as setting the mutation and cross over rate parameters for optimum performance, which can get challenging for large design set problems. The use of GA has also been suggested by authors in [7] for DSE framework and high level data path synthesis with inspiring results.

Another probabilistic speculation technique has been put forward by authors in [2] for control flow intensive HLS applications for resource allocation and selection to meet the dynamic in loop performance requirements. In [8,32] authors have tried to exploit the statistical information to trace the DSE process. In yet another approach [17], authors have deployed Architectural Configuration Graphs (ACG) and constrained Control Flow Data Graph (CFDG) to address the architectural variant analysis and selection of optimal variants with primal dual optimization technique. Their results shows a very significant speed up in design space exploration but however fails to address the intricacies of partially arranged design space and lacks the generalized analytical formulations that can be applied rigorously to large design space problems with ease.

In recent times Support Vector Machines (SVM) are also finding increasing attention from researchers in systems automation domains. Authors in [36,18,23] have attempted SVM, for feature set selection at transistor level analog design and optimization studies. The technique shows very promising prospects for non-linear design space exploration but would need further research in the coming years for system level HLS applications. Another approach, based on Artificial Intelligence (AI) has been suggested by authors in [3,37] using *fuzzy logics* to address the DSE complexity with inspiring results. Thus, DSE for HLS is a very rich and diverse topic owing to its ever increasing importance in system level design. The presented research work is based upon design space organization and feasibility studies [6,10,33], which will provide a comprehensive and thorough design space exploration framework for automated high level synthesis of DSP applications. The presented work will arrange the design space in the priority factor order based upon exact mathematical formulation of the performance parameters. The partially arranged design space will be traversed in an efficient manner along the pareto-optimal design points while parsing away redundant/ dominated regions of the design space.

1.4 Summary of Contribution

The main contributions of the presented work are listed below:

- A robust mathematical Design Space Exploration framework which can rigorously be applied to large scale high level synthesis applications
- Mathematical formulation based on the Economics principle of *Stability in Competition* to restrict the solution front along the non-dominated/ non-overlapping region(s) of the design space

- Proposed a novel integrated framework to order the design space variables and limit the extent of resource exploration to arrive at the optimal architectural configuration variant
- Proposed a highly sophisticated way to convert the partially arranged design space into fully arranged design space to overcome the problem of solution getting stuck in local optima
- Proposed the framework to solve the multi-objective combinatorial optimization problem with exact mathematical formulations
- Presented the design flow framework to convert the high level system requirements and synthesis function into its equivalent low level (Register Transfer Level) circuit mapped on to the available module library set

1.5 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 provides the preliminary concepts and topics on optimization and performance space modeling leading to the proposed design space organization and exploration framework detailed thoroughly in Chapter 3. Chapter 3 details the proposed DSE framework with detailed explanation of design space organization, Local and global arrangement of resources, Priority factor introduction, partially and fully arranged design space and finally the Optimization of multi-objective HLS problem. Chapter 4 presents a comprehensive case study example of an IIR filter synthesis application using the proposed DSE framework. Chapter 5 details the results and analysis of case study, complexity analysis, limitations and conditions of applicability of the presented framework as well as comparison of the proposed framework to an existing DSE approach. Finally, Chapter 6 concludes the research and the present’s ideas for future extension of the presented research work.

Chapter 2

Preliminaries

This chapter presents the background information for the presented research work. Theoretical background on High Level Synthesis (HLS), formalization of Multi-Objective Optimization Problem, performance space modeling and the relevance of DSE in the presented research context are addressed. This chapter addresses the HLS theoretical background, multi-objective optimization problem formulation and related topics on performance space modeling in details.

2.1 Theoretical Background on High Level Synthesis

High Level Synthesis (HLS) is automated design process that converts the algorithmic behavioral specification of an application/ task to its equivalent circuit structure /system architecture [1,9,20,28,30,35]. The starting point of HLS synthesis is the algorithm defined as a function or some programming format. The function is then analyzed and subjected to performance requirements and the

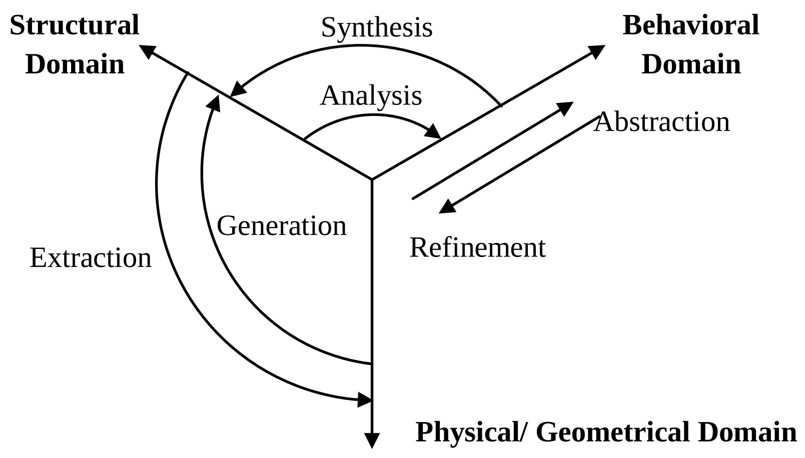

Figure 3 Y- Chart: Various stages of System Design Viz. analysis, Optimization and Synthesis

limiting/ budgeting constraints to arrive at a suitable architecture configuration that behaves in the way as described by the application function and meets the performance/ budgeting constraints. HLS includes System design level tasks including behavioral specifications, task and data structure determination/ break-down, Design Space Exploration and achievable performance estimation, determination of suitable architectural variant that meets all the objective constraints, library module allocation to specific task(s), scheduling, binding and multiplexing of resources to achieve a system that replicates the application task behavior. System design from conceptualization to physical realization goes through a series of steps and abstraction stages with each stage revealing more detailed and relevant information to the succeeding stage. High Level Synthesis (HLS) bridges the gap between the behavioral description and the RTL level structural model of a system. Fig. 3 shows the various stages of the System design depicted in the form of Y-Chart. The three axes represent the Behavioral (Algorithmic), Structural (Architecture) and Physical (Geometrical) domains of system design. The Behavioral level specification is the system description in the form of function or code, the structure of the system follows the system behavior and incorporates the features to accommodate and take into account the design performance and limiting constraints. The physical domain is the actual in-silicon implementation of the circuit at the gate level. All these three process are closely inter-related and the circuit/system design is subjected to gradual and necessary verification stages in design hierarchy for the optimal design cycle of development. Typically in system design flow, the application tasks behavioral description is converted into pseudo-algorithms, which are realized as systems micro-architectural components, programmable links, i/o and memory components. An optimized micro-architecture designed for a specific application should reflect the following important features:

a) Work load specifications: This is the typical behavioral description of the application algorithms and associated data structures of the embedded system under development. The tasks are mapped unto the library modules/member elements which make up the system data path circuitry. The data path works under the timing and close co-ordination signals originating from the module Control Unit.

b) Constrained Specifications: These are the qualitative measures that define the modular performance requirements and budgeting constraints including speed/throughput, worst latency, area and power restrictions etc.

c) Component Library: These include the low level modules and functional operators that implement the desired behavior on the data. Component library can include functional operator modules, interfaces, memory, multiplexing and glue logics forms the primitive components and topology libraries available for system level design

2.2 Multi-Objective Optimization:

Multi Objective Optimization belongs to the *combinatorial* class of problems with more than one objective to be satisfied simultaneously. The parameterized objectives are invariably contradictory in nature leading to the conditions of *infeasibility* and *sub-optimality* of a solution. Often, binding *cost function(s)* is used as a quantitative measure for ascertaining the wellness of a particular solution towards the desired objectives. As the multi-objective optimization problems are *combinatorial* in nature, the problems can have a range of non-dominated local solutions, formally known as *pareto-optimal* solution sets. A *global optimal* solution is the candidate solution with most desirable cost function characteristics among all the objective function parameters. Formally, a multi-objective optimization problem can be represented as a multi constrained minimization/ maximization problem [15,26] as shown below:

$$\min/\max : \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t. } \sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad \forall i \in \{1,.....,m\} \qquad (1)$$

$$\sum_{j=1}^{n} a_{ij} x_j \ge b_i \qquad \forall i \in \{m+1,.....,m+q\} \qquad (2)$$

$$\text{s.t. } x_j \in \{0,1.....,m\} \quad \forall j \in \{1,.....,n\}$$

Where, $b_i > 0$ $\forall i \in \{1,.....,m+q\}$ and $a_{ij} \ge 0$ $\forall i \in \{1,.....,m+q\}$, $\forall j \in \{1,.....,n\}$. Here each of the *m* constraints of family (1) are called *knapsack constraints*, while each of the *q* constraints of family (2) can be interpreted as *demand constraints*. This specialized case of Knapsack problems and is also formally known as the *multi-demand multi-constrained knapsack problem* (MDMKP) [25], which is in fact the nature of DSE Optimization problems. Later in this work, a thorough DSE exploration framework will be developed to solve the MDMKP problem efficiently and accurately from an HLS point of view.

2.3 Performance Space Modeling:

The idea of performance space modeling is to provide a quick estimate of achievable performance parameters for a candidate design solution [24,29,31]. This is a kind of experimental simulation technique which extends the scope of conventional simulation toolset by incorporating additional features warranted for solving optimization problems in time and space efficient manner. The most distinguishable features of performance space modeling techniques from conventional simulation and modeling tool set include the creation of feasible design space configurations and determining the best way to traverse the generated design space while looking for the optimal candidate solution set.

Thus the main objectives of performance space modeling technique(s) include maintaining good solution tractability and prime solution accuracy while maintaining low overall time complexity. A comprehensive list of various prevalent performance modeling techniques can be found in [10]. The presented work is limited to the *top down constrained* space modeling technique, which has proven to be far superior to any other contemporary design and modeling technique for its hierarchal, modular and predictable exploration characteristics. Fig. 4 shows a multi dimensional output-to-input mapping model where a set of performance space specifications are mapped back to the resource set *R1*, *R2*, *Rn* in more than one possible way. This multi-way mapping of the subject output attribute onto the input variable space in more than one possible way is a very typical *combinatorial* problem characteristic. Only one of the possible combinations of these multi-way mapping provides a superior design configuration while the rest are dominated by the superior solution set. The proposed DSE framework restricts the solution front from exploring the redundant and dominated regions of the search space, and thus keeps the progressing

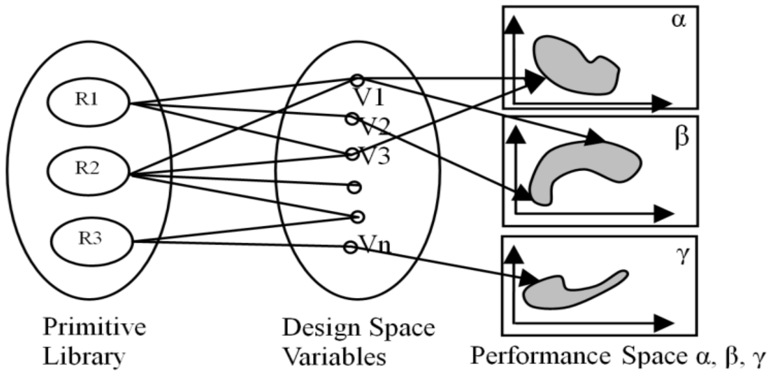

Figure 4. Illustration of multi way many to many output input mapping relationships in performance space to design space

solution along pareto-optimal wave front. For example as shown in Fig. 4, variant 1 and variant 3 have equivalent values in one of the performance space parameter ‘α’ but corresponds to totally different magnitude when mapped to some other performance space parameter ‘β’. Thus selection of optimal architectural variant involves prudent tradeoffs between different performance space parameters which involve power consumption, speed of execution, area, cost etc. The combinatorial nature of DSE problems along with budgeting constraint specifications and the inherent discreet integer programming characteristics of system design makes the process of finding a solution very time consuming and often untraceable. This work will develop a time efficient and accurate design space exploration strategy while addressing the above discussed design issues and complexities pertaining to *one* of *many* mapping possibilities.

2.4 Motivation for the Research

The motivation behind this research was to automate the much needed gap between the behavioral (algorithm) of a given design specification into structural (system architecture) for a given set of performance requirements/ constraints. As the embedded electronics technology matures the precise configuration of the system to most approximately reflect the exact behavioral model on system architecture without over-killing/ over-utilizing any of the resources is necessary. Moreover as embedded systems are custom designed any over/under utilization of the available resources adversely affects the power consumption numbers as well as the system cost. Therefore the precise mapping of the performance parameters on system architecture is essential. A brief summary of the research implications are listed below:

1) **Design Automation:** This approach can determine the optimal architecture configuration for a given set of library functions (modules), synthesis function (Algorithm) and budgeting/performance constraints with minimal human intervention.

2) **Predictable Design Flow:** This approach provides a rigorous set of mathematical models that can be applied to real world applications for repeated and expected design configurations in contrast to other heuristic approaches

3) **High Resolution Accuracy:** As the proposed approach restricts the solution exploration front along the non-dominated/ non-overlapping regions of the design space the solution accuracy resolution is within the error range of least important/ least precise design space variable.

4) **Ease of Tradeoff Analysis:** The most significant contribution of this approach is the ease with which the design space configuration can be modified to reflect the system level tradeoffs between different performance parameters.

5) **Lower Error Rate:** As a secondary effect of predictable design flow and high degree of automation the extent of error in system level design and architecture tradeoff analysis can be minimized.

6) **Reusability:** As the system design automation is highly modular and follows the conventional top-down hierarchal design approach, it is most suitable for existing design reusability. A systems tasks mapped on to lower abstraction stages can be subjected to plug-n-play IP cores or COTS elements.

Chapter 3

Proposed Design Space Exploration Framework

This chapter details the Design Space Exploration research framework to arrive at a suitable micro-architecture design configuration using Ordered Design Space and Stability in Competition principles. The ordered design space will help maintain solution feasibility whereas Stability in Competition will help restrict the solution front progression from exploring the dominated and redundant regions of the design space.

A typical embedded system design consists of data processing elements such as hardware components, software modules, interconnecting logics including buses, multiplexing schemes, input-output, module to module interface logic along with clocking and power distribution infrastructure. Efficiently mapping the system high level requirements on low level system micro-architecture requires mapping a given high level task on a particular library module for the application under development. A random walk through the possible design configuration would be exhaustive and should be avoided for any practical application. The arrangement of design resources/ library modules in order of their respective correspondence to design performance parameters helps traverse and search the optimal design configuration (variant) which is addressed as follows:

3.1 Design Space Organization:

The combinatorial nature of DSE problems makes the design space grow exponentially with the number of design variants. The presented work applies order and extent of resource exploration to keep the solution front within feasible and relatively improving marginal returns region(s). Fig. 5.a shows a hypothetical search through random design space with solution front traversing feasible and infeasible regions of multi dimensional performance space. The arrangement of resources makes the design space *partially ordered* [21,6] with nonlinearities occurring at the change of base variable along the partially ordered design space as shown in Fig. 5.b. This research will address the intricacies and the possible resolutions to overcome the shortcomings of partially ordered design space in the later sections. Establishment of order along design space variables serves a dual purpose in the proposed DSE strategy as described below:

a) ***Maintain solution feasibility****:* The solution feasibility is achieved by keeping the solution search within the limiting border variants in ordered design space.

b) ***Reduce Time complexity****:* The ordered design space exploration helps restrict the solution front from exploring the redundant regions/ configurations of the design space.

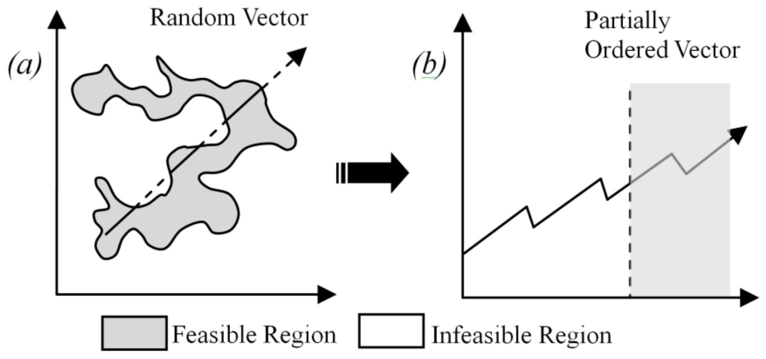

Figure 5. Hypothetical search through (a) Unordered (b) partially ordered design space

This research addresses intricacies of partial arranged design space configurations and the efficient ways to traverse the non-dominated regions of the solution set coupled with multi-dimensional binary search techniques. Two basic design space arrangements, namely the *local* and *global* variable arrangement, are discussed for design space organization.

3.2 Local arrangement of design space variables:

The local arrangement of design space variables corresponds to performance space parameters in increasing or decreasing order of effect. The component set library composed of building block elements has a finite number of variants for each functional module variable $R_{i,j}$, where i=1,2,….m are the modules and j=1,2,….n are the number of variants of each functional module. The variants $\forall j \in \{1,\ldots\ldots,n\}$ belonging to each functional module $\forall i \in \{1,\ldots\ldots,m\}$ are characterized by specific attributes towards the scaled performance parameters. For example, a *multiplier* module can have number of available variants such as in booth, MAC, carry look-a-head etc. with each having characteristic attributes in terms of power consumption, area and speed rating. Some of these desirable attributes come at the cost of additional power rating and area requirements. Since the HLS is inherently combinatorial in nature the combined effect of the composing design variables in a given variant configuration has to be considered for their respective performance characteristics.

3.3 Global arrangement of decision variables:

The global arrangement of decision variables help determine the relative order of importance of the decision space variables to the asserting performance space directive. The global decision variables are ordered as per their *Priority Factors* (PF), which are discussed below:

3.3.1) Priority Factor:

The priority factor is a measure of the marginal resource contribution to the objective cost function normalized over its variant range. Mathematically the priority factor for a resource R_1, R_2 …..R_n can be given as:

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \qquad (3)$$

Where, $\Delta N_{Rn} \cdot K_{Rn}$ is the marginal contribution by the n^{th} resource to the cost function and K_{Rn} is the absolute per unit performance (PPU) contribution of a design resource to the asserting cost function. Mathematically the PPU can be defined as:

$$PPU_{Ri} = \frac{\partial P}{\partial Ri} = \frac{\partial f(R_1, R_2 ....R_n)}{\partial R_i} = K_{Ri} \qquad (4)$$

Where *Ri* is the i^{th} resource with *n* variants. In the presented DSE technique, the PF will help judge the relative influence of a particular resource to the optimization problem objective functions such as area, execution time, resource power consumption etc. The PF will be used to organize the decision variable space in order to provide the sense of direction to DSE solution front. The next section formulates and analyzes the necessary mathematical relationships for various performance measures including power, resource area and frame execution speed parameters.

3.3.2) Analysis for hardware area of the resources:

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 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. Therefore, for a system with 'n' functional resources the total area can be given as:

$$A = \sum A(Ri) = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) + A(R_{clk}) \quad (5)$$

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 derivative to equation (5) with respect to N_{R1}, N_{R2}….N_{Rn} yields equation (6) and equation (7) respectively as shown below:

$$\frac{\partial A}{\partial N_{R1}} = \frac{\partial(\sum (N_{R1} \cdot K_{R1} + ... + N_{Rn} \cdot K_{Rn}) + A(R_{clk})}{\partial N_{R1}} = K_{R1} \quad (6)$$

$$\frac{\partial A}{\partial N_{Rn}} = K_{Rn} \quad (7)$$

According to the ‘theory of approximation by differentials’ the change in the total area can be approximated by equation in (8):

$$dA = \frac{\partial A}{\partial N_{R1}} \cdot \Delta N_{R1} + \frac{\partial A}{\partial N_{R2}} \cdot \Delta N_{R2} + ... + \frac{\partial A}{\partial N_{Rn}} \Delta N_{Rn} + \Delta A(R_{clk}) \qquad (8)$$

Substituting equations (6) and (7) into equation (8) yields equation (9):

$$dA = (\Delta N_{R1} \cdot K_{R1}) + \dots + \Delta N_{Rn} \cdot K_{Rn} + \Delta A(R_{clk}) \qquad (9)$$

Where, $\Delta N_{Rn} \cdot K_{Rn}$: Resource ‘Rn’ area contribution and $\Delta A(R_{clk})$: The clock variants area contribution to the cost objective.

The equation above indicates the average change of area with respect to resource R1, R2 ….R_n. Here the clock oscillator is a resource which contributes to the area occupied by the resources. The PF is a determining factor to judge 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 in the DSE approach to organize the architecture design space consisting of design variants in increasing or decreasing order of magnitude. The Priority factor for the resources R_1, R_2 …..R_n can be given as:

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \qquad (10)$$

$$PF(Rclk) = \frac{\Delta A(Rclk)}{N_{Rclk}} \qquad (11)$$

The factor defined above determines average variation in area affected by the change of number of a certain resource.

3.3.3) Analysis for power consumption:

For a system with 'n' functional resources the total power consumption '*P'* of the resources can be represented by equations (12) and (13):

$$P = \sum_{i=1}^{n} (N_{Ri} \cdot K_{Ri}) \cdot p_c \qquad (12)$$

$$= (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + .. + N_{Rn} \cdot K_{Rn}).p_c \qquad (13)$$

Where 'N_{Ri}' represents the number 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. By applying the partial derivatives to equation (13) the Priority Factor (PF) for power consumption can be given as follows:

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \cdot (p_c)^{\max} \qquad (14)$$

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + .. + N_{Rn} \cdot T_{Rn}}{N_{R_{clk}}} \cdot (\Delta p_c) \qquad (15)$$

Where the symbols have their pre-defined meanings, the priority factors defined in equations (14) and (15) indicate the average change in the total power consumption with the change in number of resources at maximum clock frequency. For example, equation (19) indicates the change of total power consumption with the change in the number of that particular resource at maximum clock frequency. The priority factor will help to arrange the architectural variants of the design space in increasing or decreasing order of magnitude depending on the parameter of optimization. This further facilitates in the selection of the optimal design point that satisfies all the operating constraints and optimization requirement specified. In the above equations, maximum clock frequency was considered because at this frequency the total power consumption is the maximum. Thus the change in 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.

3.3.4) Analysis for Execution Time:

According to the amount which different resources contribute to the change in *execution time*, Priority Factor (PF) for execution time can be defined as follows:

$$PF(Rn) = \frac{T_{Rn}^{Max} - T_{Rn}^{Min}}{N_{Rn}} \qquad (16)$$

$$PF(Rclk) = \frac{T_{Rclk}^{Max} - T_{Rclk}^{Min}}{N_{Rclk}} \qquad (17)$$

In equation (16) 'T_{Rn}^{Max}' and 'T_{Rn}^{Min}' are the maximum and minimum value of execution time when the resource 'Rn' is minimum and maximum respectively at maximum clock frequency. In equation (17), 'T_{Rclk}^{Max}' and 'T_{Rclk}^{Min}' are the maximum and minimum values of 'execution time' when the clock period is maximum and minimum respectively and all the available resources have the maximum value. The PF defined in equation (16) and (17) indicates the average change in execution time with the change in number of a particular resource and change in clock frequencies respectively.

3.4 Design Space Traversal:

The arrangement of resources/ design space variables in PF hierarchy renders the search space *partially ordered*. As such the solution front can get stuck in local minima while searching for the *optimal* design candidate solution. The purpose of this section is to critically analyze the *overlapping domains* responsible for making the search space non linear. This section will also develop the necessary mathematical formulation to overcome the shortcomings of partially ordered design space and keep the solution front along the non dominated pareto-optimal design variants.

3.4.1 Root cause of the problem: Overlapping domains

A decision variable *Ri*, can be used to contribute a discreet range of values over its constrained range of local variants towards an asserting cost function, as shown below:

$$LocalDomain(Ri) = K_{Ri} \times \Delta N_{Ri}^{\max} \qquad (18)$$

Where, K_{Ri} is the PPU contribution of resource *Ri* and $N_{Ri}^{\max}$ is the maximum range of constrained domain of decision variable *Ri*. Fig 6.a shows a representative model of partially arranged decision variables in PF hierarchy, with each circle representing a decision variable and diameter corresponding to PPU. The local variants are uniformly distributed at the outer periphery of each decision variable. An arranged variable will have an overlapping local domain with the next variable in PF hierarchy if the following condition holds:

$$K_{Ri} \times N_{Ri}^{\max} > K_{R(i+1)} \times N_{R(i+1)}^{\min} \qquad (19)$$

Where $K_{R(i+1)}$ and $N_{R(i+1)}^{\min}$ are the PPU and minimum number of variants of next variable $PF_{(i+1)}$, in hierarchy. The overlapping domains are responsible for making the decision space *partially ordered* resulting in more than one correspondence for a specific cost function objective. Ideally each of the design space variables would have an independent *one*-to-*one* correspondence for the range of values it represents. This can only be possible if the local domains of decision variables just touch at the outer periphery resulting in a smooth transition as shown in Fig.6.b. A careful observation of the non

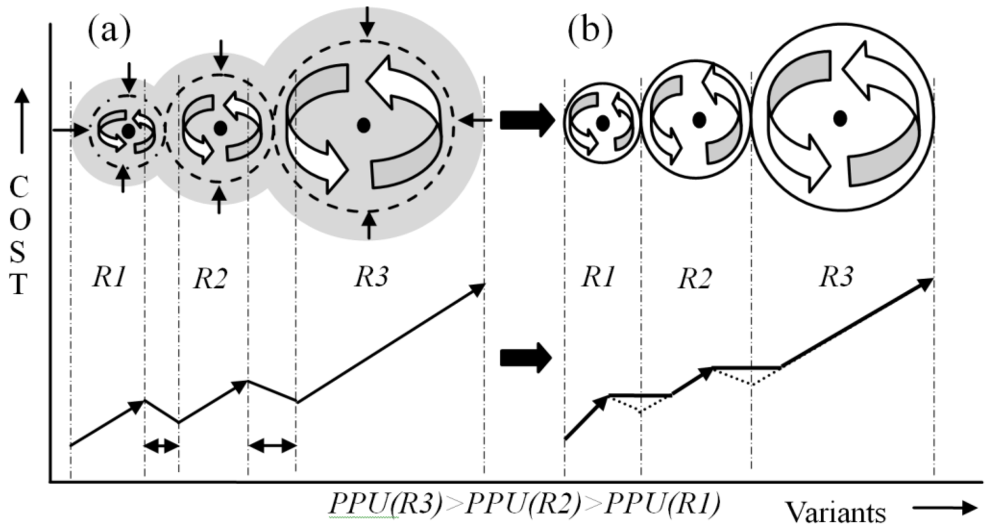

Figure 6 (a) the overlapping domains of partially arranged design variables in PF hierarchy (b) linearly ordered decision variables with restricted domains to avoid inferior dominated regions circle.

overlapping domain model of Fig 6.b reveals its resemblance to modern positional number systems to a certain extent. Although the DSE problem does not strictly comply with a standard positional system *base* or *radix*, this research utilizes the positional independent domain property and thus will try to make the search space linear and in proper order. Next section will develop the necessary mathematical relationships to achieve proper ordered design space. The perfectly ordered design space will be subjected to *binary search* algorithm for exploring the optimal design variant quickly and accurately.

3.4.2 Stability in Competition: An Overview

Overlapping domains of the partially ordered design space variables are responsible for generating dominated *one-to-many* correspondences and in turn *inferior regions* of the search space. As such the DSE algorithms can easily get stuck in the local minima's along the design space traversal and will lack accuracy as well as speed due to redundant calculations. Therefore there is a need to make the search space as linear as possible by restricting the extent of resource exploration in the superior non-dominated regions of the PF arranged design space. This objective closely correlates to the well established economics principle of *stability in competition.* This research will exploit the similarity of Stability in Competition principle to address the DSE problem in question.

H. Hotelling [16] established in 1929 the principles of *Stability in Competition* between competing variables for the desired *cost* objective. Fig. 7 shows the act of self counter-balancing between two competing variables to the point of stability, where each variable counterbalances the other's cumulative weight. The point of instantaneous balance marked by letter E in Fig. 7 signifies marginal stability between the competing variable set. Any disturbance at this point would make the other variable gradually follow the next variable to the point of next marginal stabilization. The presented HLS DSE Optimization problem displays similar behavior with competing design space variables trying to increase their per unit resource utilization over their local domain while keeping the solution front non-dominated. The resources in the solution set can only be used in discreet multiples and thus makes the HLS discreet integer optimization problem which also gets due consideration in this proposed DSE framework. The next section will develop the necessary mathematical relationships to restrict the extent of ordered resource exploration to keep the solution front non-dominated.

3.4.3 Limiting the Solution Bounds to Non Dominated regions of the Design Space:

Let *P* be a sample performance space parameter directive being asserted on the design space variables *R1, R2*, ….., *Rn.* The relationship between the performance space and design space variables can be described as:

$$P \xrightarrow{f_P} (R1, R2, ...Ri, ...Rn) \qquad (20)$$

where f_p is an illustrative performance space parameter and can represent power, area or frame execution time. For the initial mathematical formulation only two variables are considered in the solution set and a summation function as the cost objective. The same analysis holds true for any complex output-input function with any number of variables arranged in PF hierarchy.

For a summation objective and dual variable, the performance objective can be given as:

$$P = \Delta N_{R1}.K_{R1} + \Delta N_{R2}.K_{R2} \qquad (21)$$

where K_{Ri} is the per unit performance contribution of the resource *Ri* and can be calculated as*:*

$$\left.\frac{\partial P}{\partial R}\right|_{R=R1} = K_{R1} \quad \text{Or,} \quad \left.\frac{\partial P}{\partial R}\right|_{R=Ri} = K_{Ri} \qquad (22)$$

The next sub-section defines the *absolute* and *compounded* average of the objective function P over the design variables *R1* and *R2*. This analysis establishes the bounds on the solution space.

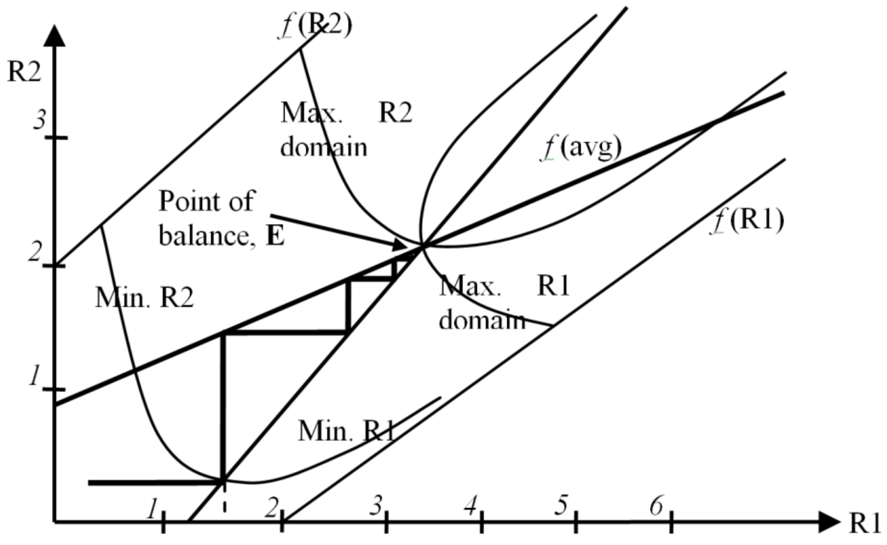

Figure 7. Illustration of balancing conditions between dual competing variables. The co-ordinates represent the decision variable local variants and the point E represents point of balance. The system traverse through R1 and R2 variants till the point of balance reached at E. the curves through balance point E shows the maximal non dominated rage of R2 and R1 respectively.

The design space exploration is optimally mapped on the design space variables between these defined bounds. Assuming *R2* has greater per unit resource contribution than resource *R1* towards the sample performance space directive f_P, the *absolute* and *compounded* cost objective function of design variables *R1* and *R2* can be defined as:

a) Absolute Function: The cost performance directive as an independent function of design space variables provides the *upper* and *lower bounds* to the solution space. In case of multi variable design space, the least *PF* variable would provide the lower bounds and the highest PF variable the upper bounds. For the illustrative performance directive f_p, and dual variable design space:

$$P_{\min} = N_{R1}^{\min}.K_{R1} \approx K_{R1} \qquad (23)$$

$$P_{\max} = N_{R2}^{\max}.K_{R2} \qquad (24)$$

P_{min} is the smallest step function possible in the direction of cost function objective and P_{max} is the maximum growth possible in the cost function along resource R2.

b) Compounded Function: The compounded average takes the combined effect of resources into consideration averaged over the total number of resources in the solution set and can be described mathematically as:

$$P_{avg} = \frac{\Delta N_{R1}.K_{R1} + \Delta N_{R2}.K_{R2}}{no.\_of\_resorces} \qquad (25)$$

P_{min} and P_{max} provide the *lower* and *upper* bounds of the solution space. The objective is to keep the compounded average P_{avg} in between the limiting bounds. By doing so, the solution front is restricted from exploring the overlapping regions of the solution space and keeps the solution front along pareto-optimal design points having one to one output-input relationship. Fig. 9 to 14 show the graphical representation of *absolute* and *compounded average* of *R1* and *R2* being driven by sample performance space directive. The graphical representation shows the compounded average of cost function *P* averaged over the design variable variants of *R1* and *R2.* For example (*1*, *R1*) shows the compounded average of *R2* and *R1*, with *R2* held at unity and *R1* allowed to vary over a discreet range of restricted integer values. Also shown in Fig. 9 to 14 are the *lower* and *upper* bounds of *P* as an independent function of *R1* and *R2*. The objective is to gradually expand the cost function while maintaining good resolution and low time complexity. The framework attempts to restrict the solution front in most optimal range of *R1* and *R2* in between the *upper* and *lower* bounds. Fig. 10 shows the discreet points marked by the letters *a* to *g*, signifying the *point of inflection,* where the cost functions *compounded average* touches the *lower* bounds of resource *R1.* For any given curve, the framework restricts the range of exploration to these critical points, as the further exploration along *R1* would make the solution *R1* dominated, thus rendering *diminishing returns* on per unit additional resource utilization. Fig.11 shows the strictly increasing cost function graphs which are most suitable tradeoff between *R2* and *R1*. Fig. 12 and 13 show possible ways to traverse the design space with increasing cost function derivatives along *R2* and *R1*. As shown in Fig. 13, an ideal *path* would be along the geometrical centers of the increasing sets of cost functions. This is not feasible due to the imprecise resolution of *R2* to represent the values at the mid points of the instantaneous cost functions. Fig. 14 shows a feasible and most suitable *stepping function* with the following important properties: (i) a continuous and seamless transition along the local domains of *R1* and *R2* (ii) an ideal balance between the precision of low per unit contributing resource *R1* and the higher efficiency but lower precise *R2*. Therefore the balancing point between two competing variables is given by:

$$K_{R1}.\Delta N_{R1} \approx K_{R2}.N_{R2}^{\min} \qquad (26)$$

Or in general,

$$K_{Ri}.\Delta N_{Ri} \approx K_{(Ri+1)}.N_{R(i+1)}^{\min} \qquad (27)$$

where $K_{R(i+1)}$ is the next immediate variable to active solution set in the ordered PF hierarchy. Equation (27) is the general equation for restricting the extent of exploration of a local variable before proceeding to the next variable in the partially ordered design space. By maintaining this condition the solution front can be restricted from exploring the dominated regions of the design space having diminishing returns. Fig. 8 summarizes the DSE algorithm.

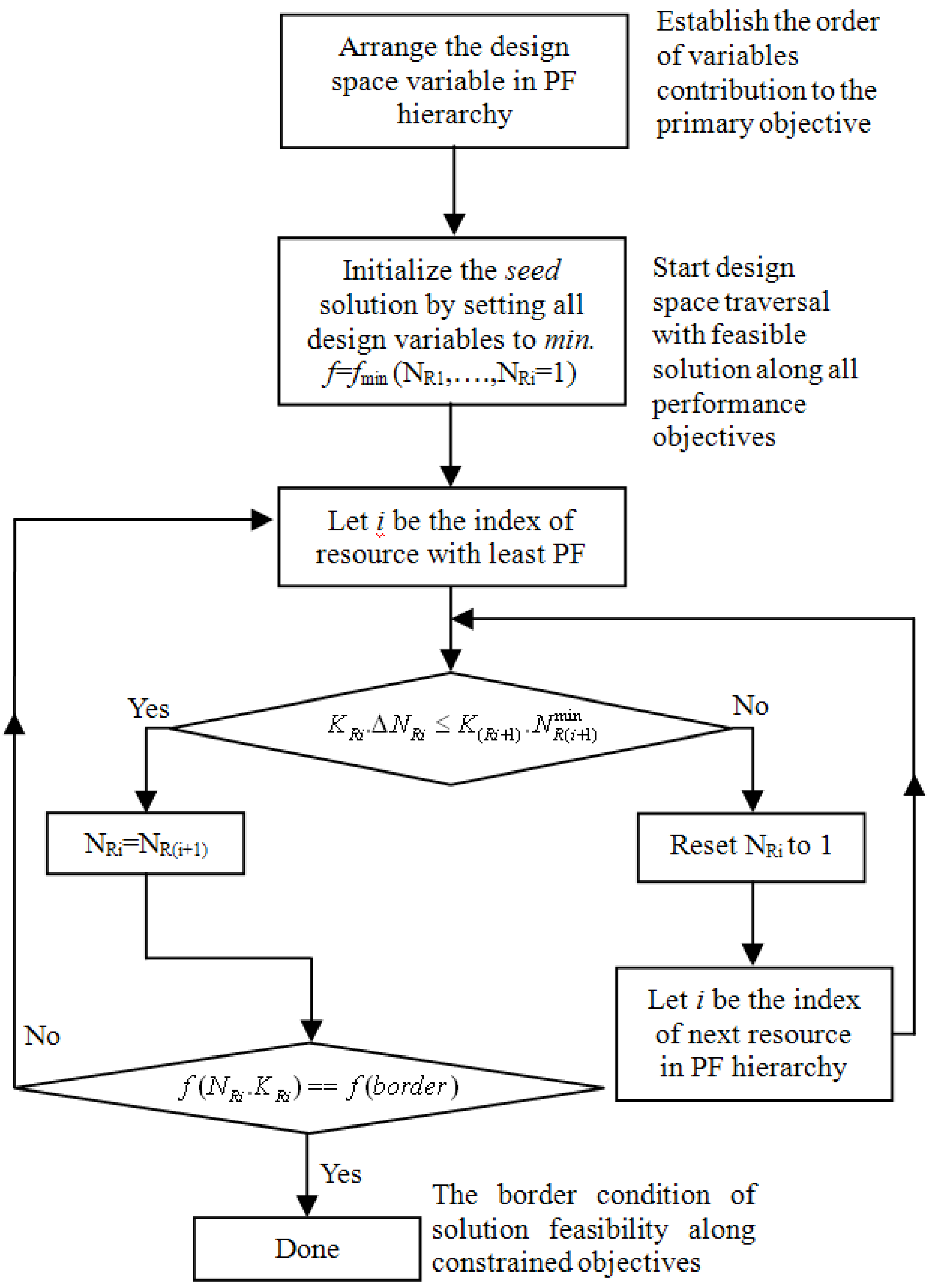
Arrange the design space variable in PF hierarchy
Establish the order of variables contribution to the primary objective
Initialize the *seed* solution by setting all design variables to *min.* $f=f_{min}(N_{R1},\ldots,N_{Ri}=1)$
Start design space traversal with feasible solution along all performance objectives
Let *i* be the index of resource with least PF
Yes
No
$K_{Ri}.\Delta N_{Ri} \le K_{(Ri+1)}.N^{min}_{R(i+1)}$
$N_{Ri}=N_{R(i+1)}$
Reset N_{Ri} to 1
Let *i* be the index of next resource in PF hierarchy
No
$f(N_{Ri}.K_{Ri}) == f(border)$
Yes
Done
The border condition of solution feasibility along constrained objectives

Figure 8. Flow Chart of the proposed DSE framework

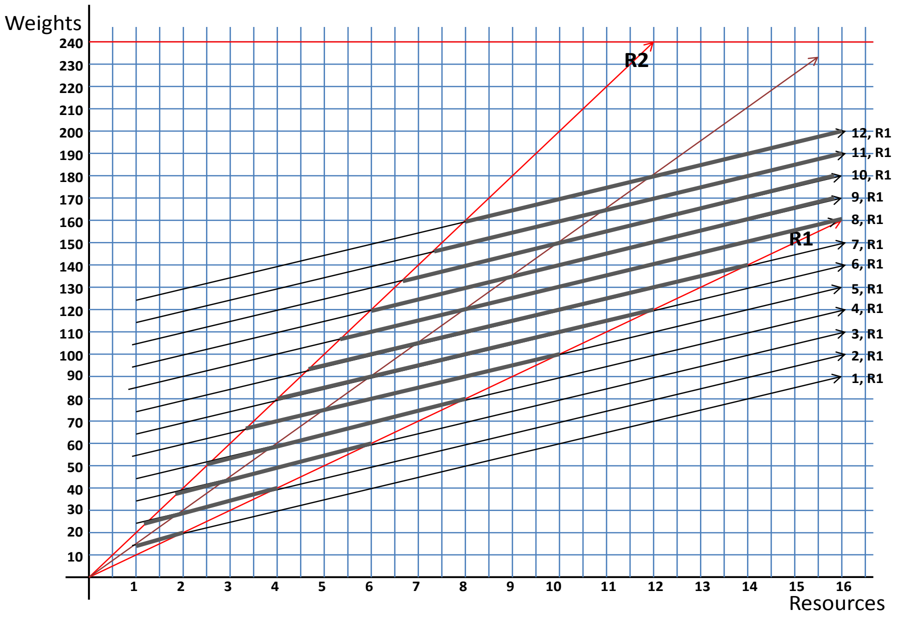

Figure 9. The absolute cost functions of resources *R1* and *R2* provide the lower and upper bounds on the solution space. The compounded average shown as (R2, R1) provides the average return per variant in the solution space. For example (1, R1) shows the compounded average of *R2* and *R1*, with *R2* held at unity. R1 is allowed to vary over a discreet range of integer values in its local domain.

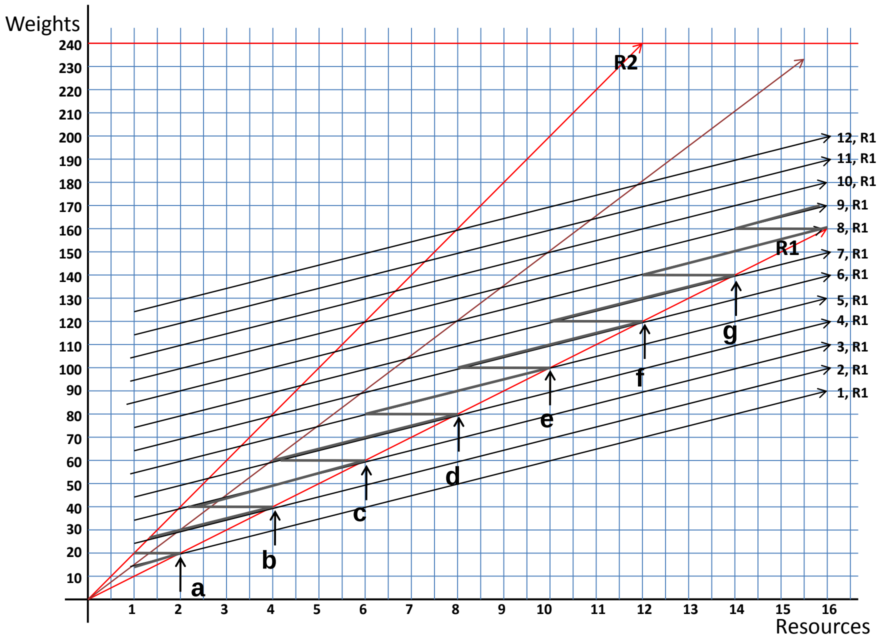

Figure 10. Strictly increasing cost function(s) between the upper and the lower bounds of the solution space.

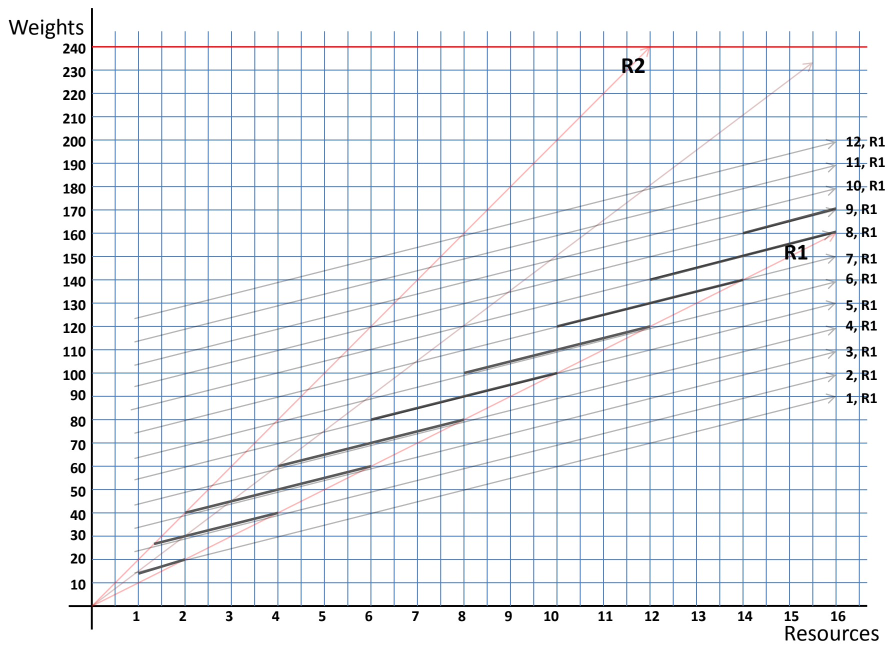

Figure 11. Strictly increasing cost function(s) between the upper and the lower bounds of the solution space.

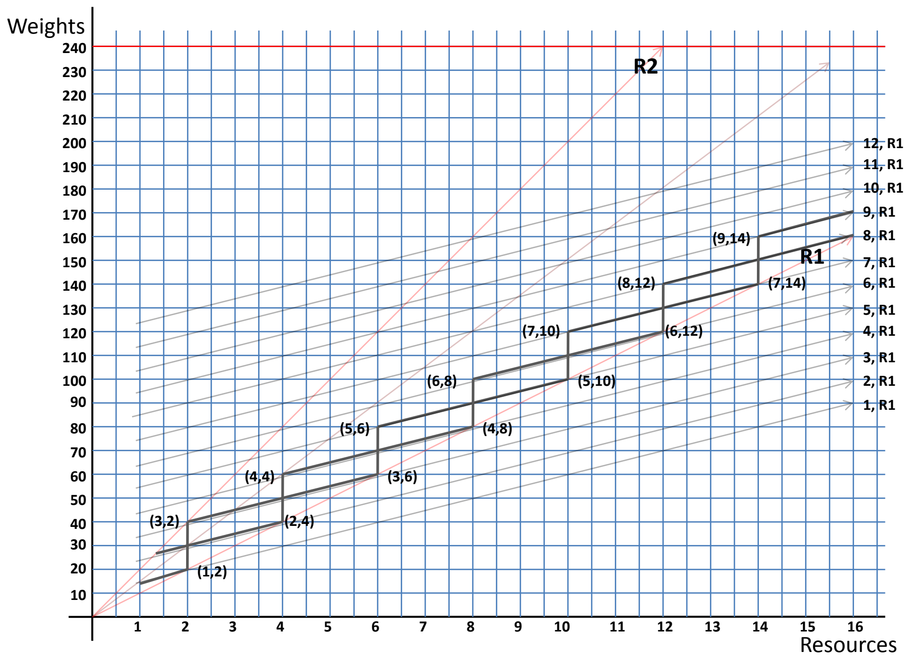

Figure 12. Design space traversal along the strictly increasing cost function(s).

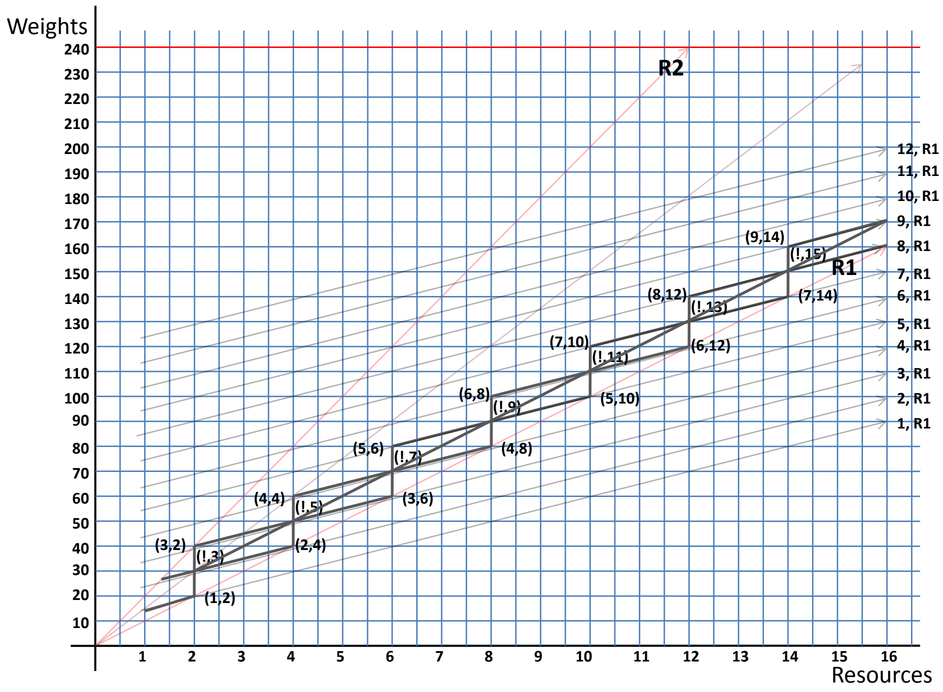

Figure 13. Ideal design space traversal path through the geometrical centers of the strictly increasing cost function(s). The path is infeasible for R2 due to the discreet integer programming conditions.

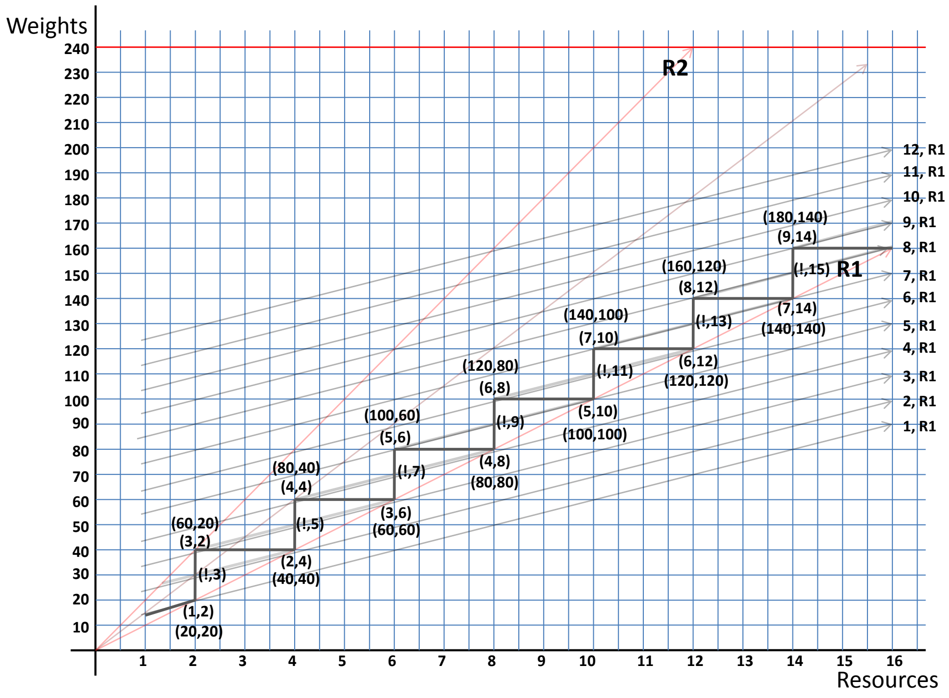

Figure 14. The feasible and most optimal design space traversal with a step function through the local domains of *R1* and *R2*. The solution front is restricted to the point of inflections and is feasible everywhere. At the corners of the step functions the design variables are balanced.

Chapter 4

Case Study

4.1 Design Flow Initiation

This chapter provides a comprehensive case study of IIR filter synthesis application to elaborate upon the proposed DSE HLS framework. Here the IIR filter is defined as the behavioral/ high level function corresponding to its characteristic function. The high level IIR filter function will be subjected to the proposed DSE framework which helps identify the structure/ configuration of the module for the given set of desired performance characteristics and budgeting constraints. This phase of the design stage is critical because the operational constraints must be clearly defined along with the parameters to be optimized. These specifications will act as the input information for the high level synthesis tools.

4.2 Problem Description

4.2.1 Input–Application Specification:

The IIR filter application can be defined by equation (28) as follows:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.041(1+z^{-1})^2}{1-1.4418z^{-1}+0.6743z^{-2}} \qquad (28)$$

Or

$$y(n) = 0.041x(n) + 0.082x(n-1) + 0.041x(n-2) + \\ 1.4418y(n-1) - 0.6743y(n-2) \qquad (29)$$

where, *x(n)*, *x(n-1)* and *x(n-2)* are the input vector variables for the function. The prior outputs are given by *y (n-1)* and *y(n-2)*, while the present output of the function is given by *y(n)*. For ease of representation, the following constants 0.041, 0.082, 0.6743 and 1.4418 are represented as *A*, *B*, *C* and *D* respectively. *x(n)*, *x(n-1)*, *x(n-2)*, *y (n-1)* and *y (n-2)* are denoted by *Xn*, *Xn1*, *Xn2*, *Yn1* and *Yn2* respectively.

4.3 Performance Requirements and System Constraints:

The performance requirements and budgeting constraints dictate the system micro-architecture and organization, as well as the design and development costs. From the practicality point of view, the following critical performance space measures are included in the presented case study design flow: power consumption, execution speed while minimizing area, and system development cost. Performance requirements and budgeting constraints assumed for the case study are as follows:

1) Maximum power budget: Max 11 watt

2) Maximum resource availability:

(a) Three clock frequencies (50, 100, 200 MHz)

(b) Multiplier variants: Maximum of four

(c) Adder variants: Maximum of two

3) Maximum execution time: 62 msec.

4) Optimization goal: Minimize power utilization while satisfying the above constraints

4.4 Components–Library Specification:

The component library consists of the primitive processing elements (PE) providing the basic building blocks of the system. Each PE is characterized by its area, latency and power consumption, with power being a complex function of area and operational frequency. The case study undertakes the following design specifications for the primitive library components [12]:

1) No. of clock cycles needed for multiplier to finish each operation: 4 cc.
2) No of clock cycles needed by the adder/ subtractor: 2 cc.
3) Area occupied by each adder/ subtractor: 20 a.u. on the chip (e.g. 20 CLB on an FPGA).
4) Area occupied by each multiplier: 80 a.u. on the chip. (e.g. 80 CLB on an FPGA).
5) Area occupied by the 50, 100, 200 MHz clock oscillator: 4, 6, 10 area units respectively.
6) Power consumed at 50 MHz: 10 mW / area unit.
7) Power consumed at 100 MHz: 20 mW / area unit.
8) Power consumed at 200 MHz: 60 mW / area unit.

4.5 Solution Overview:

Behavioral synthesis of a system module consists of at least three basic stages: mapping, scheduling and binding.

a) Mapping: Mapping involves the selection of basic building block elements and their respective variant(s) from the component library to map the high level task function on system micro architecture. The function or task mapping must adhere to overall system objectives in compliance with performance requirements as well as budgeting constraints.

b) Scheduling: The scheduling policy determines the order and method by which the system resources and processing elements are accessed by the application tasks. The need for scheduling policy arises from the requirement of multiple tasks trying to access a limited number of resources in the constrained system. Scheduling policy works in very close co-ordination with task mapping to maximize the system performance for the given objectives.

c) Binding: Resource binding determines the sharing of constrained resource(s) for different operations in the tightly coupled data path pipeline. The resources can be bound in spatial and temporal domains depending upon the application data structure and data dependencies. Multiplexing and de-multiplexing schemes along with temporary memory elements facilitate resource sharing.

As discussed above, task *mapping* acts as liaison between objective function and performance space requirements; it is one of the most important phases to effectively capture the design requirements, application data structure, data dependencies and resource utilization in the modular system design. This case study will concentrate on the task mapping and resource allocation, while remaining confined to standard ASAP scheduling algorithms.

4.6 Design Space Organization:

The next section determines the relative importance of the design variables (Namely adder/ multiplier/ clock) for the objective performance parameters (viz. area/ power / execution time). The order of these variables will help determine the border variants for solution feasibility studies.

4.7 Analysis for Power Consumption:

With Power as the performance parameter, the relative importance of each of the decision variable resources in the design space is ascertained. PF is calculated for each of the design resources to calculate its relative importance/ contribution towards the power objective:

1) PF of adder / subtractor:

$$PF(R1) = \frac{\Delta N_{R1}.K_{R1}}{N_{R1}}.p_c^{\max} = \frac{(2-1).20}{2}.60 = 600 \quad (30)$$

2) PF of multiplier:

$$PF(R2) = \frac{\Delta N_{R2}.K_{R2}}{N_{R2}}.p_c^{\max} = \frac{(4-1).80}{4}.60 = 3600 \quad (31)$$

3) PF of clock oscillator:

$$PF(R3) = \frac{N_{R1}.T_{R1} + N_{R2}.T_{R2}}{N_{Rclk}}.(\Delta p_c) \quad (32)$$

$$= \frac{(2.20 + 4.100)}{3}.(60-10) = 7333.33$$

The above factors indicate the true resource contribution capacities to the objective cost function with the change in number of a specific resource. For example, clock resource is the most detrimental factor for increased power consumption over its range from 50 to 200 MHz. The adder/ subtractor make the least effective contribution to the power requirements. As clock is the most effective yet least precise contributor to power consumption objective, it is at the top of the hierarchy followed by lesser per unit contributors, multiplier and adder units, in the objective cost function. Thus,

PF (Clock) > PF (Multiplier) > PF (Adder/ Subtractor)

The order of contribution decides the direction of exploration for DSE explorer to effectively map the performance objective on the Design space variables.

4.8 Analysis for execution speed parameter:

Similar to Power consumption PF analysis, the relative importance of design variables towards frame execution speed is investigated:

1) PF of adder/ subtractor:

$$PF(R1) = \frac{\Delta N_{R1}.T_{R1}}{N_{R1}}.T_p^{\max} = \frac{(2-1).2}{2}.(0.02) = 0.02 \quad (33)$$

2) PF of multiplier:

$$PF(R2) = \frac{\Delta N_{R2}.T_{R2}}{N_{R2}}.T_p^{\max} = \frac{(4-1).4}{4}.(0.02) = 0.06 \quad (34)$$

3) PF of clock resource:

$$PF(R3) = \frac{N_{R1}.T_{R1} + N_{R2}.T_{R2}}{N_{Rclk}}.(\Delta T_p)$$

$$= \frac{(2.2 + 4.4)}{3}.(0.02 - 0.005) = 0.1 \qquad (35)$$

Thus the relative order of exploration for frame execution speed parameter will be as follows:

PF (Clock) > PF (Multiplier) > PF (Adder/ Subtractor)

The above measures indicate the relative resource contribution from different system resources to affect the frame execution speed. Thus, clock can be the most detrimental factor to speed up the frame execution time followed by multiplier and adder/ Subtractor unit.

4.9 Design Space Traversal with Power as Performance Directive:

Once the relative importance of design space variables is ascertained for different performance space parameters, the design space must be explored to identify the optimal design variant configuration that meets all the design objectives and constraints. *Power* is chosen as the primary performance objective for its ever increasing importance in HLS domain, which is gradually mapped on the design space variables. The DSE explorer optimally reflects the expanding power budget on design space variables by guiding the solution front through the pareto-optimal points. This results in a range of *local optimal* solutions limited to the *globally optimal* solution at the *border variant* [33,37] as described in the next section.

4.10 Selection of Optimal Variant:

With power as the primary performance directive the DSE explorer gradually expands the power budget on design space variables. The resource sets are explored in the order of PF hierarchy starting with adder (R1), the least PF variable, gradually moving up the hierarchy to multiplier (R2) and clocking (R3) variants:

$$\underbrace{\text{PF (clock)}}_{\text{R3}} > \underbrace{\text{PF (multiplier)}}_{\text{R2}} > \underbrace{\text{PF (adder/ subtractor)}}_{\text{R1}}$$

Table 1 lists the design space configuration variants for the presented IIR filter synthesis case study with each variant characterized in terms of power consumption, latency, area and frame execution time parameters. In order to overcome DSE time complexity, the ordered design space is subjected to multi-dimensional feasibility study coupled with binary search engine [21,6].

1. The design exploration augments the most frugal power consumption variant with minimal resources (one adder, one multiplier) operating at minimum frequency at 50 MHz. Thus,

$$P_{\min} = (N_{R1}^{\min}.K_{R1} + N_{R2}^{\min}.K_{R2}).P_c^{\min}$$

$$= (1\times 20 + 1\times 80).10 = 1 \text{ watt} \quad (36)$$

where, $P_c^{\min}$ is the minimum per unit power consumption rate, namely 10 mW / unit area at 50 MHz.

2. The DSE explorer expands the solution front in the local domain of active variable resource R1 to balance the next variable in PF hierarchy:

$$P_{(Variant\_2)} = (N_{R1.2}.K_{R1} + N_{R2}^{\min}.K_{R2}).R_{3.1}$$

$$= (2\times 20 + 1\times 80).10 = 1.2 \text{ watt} \quad (37)$$

where, *Ri.j* represents the jth variant of ith resource. For example, *R1.2* represents the 2nd variant of type 1 resource.

Variants	Design Vectors	Power (watt)	Latency (c.c)	Execution Speed (msec)
	(R3, R2, R1)	$(\Sigma N_{ri}.K_{ri}).P_c$	(ASAP)	$[lat.+(frame-1).T_{cc}].T_{clk}$
V1	(1,1,1)	1.0	22	400.04
V2	(1,1,2)	1.2	22	400.04
V3	(1,2,1)	1.8	16	240.08
V4	(1,2,2)	2.0	16	240.08
V5	***(1,3,1)***	***2.6***	***14***	***200.08***
V6	***(1,3,2)***	***2.8***	***12***	***160.08***
V7	***(1,4,1)***	***3.4***	***14***	***200.08***
V8	***(1,4,2)***	***3.6***	***12***	***160.08***
V9	(2,1,1)	2.0	22	200.02
V10	(2,1,2)	2.4	22	200.02
V11	(2,2,1)	3.6	16	120.04
V12	(2,2,2)	4.0	16	120.04
V13	(2,3,1)	5.2	14	100.04
V14	(2,3,2)	5.6	12	80.04
V15	***(2,4,1)***	***6.8***	***14***	***100.04***
V16	***(2,4,2)***	***7.2***	***12***	***80.04***
V17	(3,1,1)	6.0	22	100.01
V18	(3,1,2)	7.2	22	100.01
V19	(3,2,1)	10.8	16	60.02
V20	***(3,2,2)***	***12.0***	***16***	***60.02***
V21	***(3,3,1)***	***15.6***	***14***	***50.02***
V22	***(3,3,2)***	***16.8***	***12***	***40.02***
V23	***(3,4,1)***	***20.4***	***14***	***50.02***
V24	***(3,4,2)***	***21.6***	***12***	***40.02***

Table 1. Design space variants with different configurations of system resources and corresponding power, execution time parameters. The table also shows pruned variants between variants 4 and 9 with inferior power to speed ratio to other non-dominated regions.

3. The active variable *R1* exhausts its local domain and is unable to balance the next variable in PF hierarchy as:

$$K_{R1}.\Delta N_{R1} < K_{R2}.N_{R2}^{\min}$$

Or,

$$(20\times 2).R_{3.1} < (80\times 1).R_{3.1} \qquad (38)$$

In order to expand the solution wave front, the DSE explorer perturbs the next variable in PF hierarchy. Thus, *R2* is included in the basic solution set.

4. The DSE explorer expands the cost function (Power) while searching for the *border variants* [6,21]. At *variant 4*, the DSE explorer reaches the *point of inflection,* where resource R1's two variants (R1.2) and resource R2's two (R2.2) variants combines to balance the equivalent cost contribution of resource R3's two variants (R3.2):

$$(\mathrm{R}_{1.2} \times 10 + R_{2.2} \times 10) \mid R_{3.1} \equiv (\mathrm{R}_{1.1} \times 20 + \mathrm{R}_{2.1} \times 20) \mid R_{3.2}$$

Or,

$$(2\times \underbrace{20+2\times 80)\times 10}_{\downarrow\ \text{\textit{Variant 4} with 2 adders, 2 multipliers, operating at 50 MHz}} = \underbrace{(1\times 20+1\times 80)}_{\downarrow\ \text{\textit{Variant 9} with 1 adder, 1 multiplier, operating at 100 MHz}} \times 20 = 2\ \text{watt} \qquad (39)$$

At this point the DSE explorer restricts further exploration in the local domains of (R1, R2). Resource area overhead is traded for higher system clock frequency at 100 MHz with reduced parallelization, while limited to the same power consumption rating of 2 watts. Note, exploration between variants 4 and 9 would have provided diminishing returns on power speed ratio making the solution space R1, R2 dominant. By pruning the dominated design space, the DSE explorer has kept the solution space to be R3 dominated with the higher PF. The frame execution time parameter for these variables is ascertained to demonstrate their superior characteristics to the non-dominated design variants. The frame execution speed is calculated as follows:

$$T_{exec} = [latency + (n-1)\tau_{cycle}] \times \tau_{c.c.} \quad (40)$$

where T_{exec} is the frame execution time, n is the number of frames, τ_{cycle} is the pipeline throughput measured in clock cycles and $\tau_{c.c.}$ is system clock rate. Thus for 1000 frame execution, variants 4 and 9 have the following frame execution time:

$$T^4_{exec} = [16 + 999 \times 12] \times (0.02) = 240.08 \text{ msec} \quad (41)$$

$$T^9_{exec} = [22 + 999 \times 20] \times (0.01) = 200.02 \text{ msec} \quad (42)$$

where, T^n_{exec} is the n^{th} variant execution time. Although both design variants (4 and 9) compete for the same power ratings, variant 9 provides better execution speed per unit power consumption. The variants in the pruned dominated region (#V4 to #V9) have an inferior power to speed ratio relative to their non-dominated counterparts. A closer look reveals that the DSE explorer has effectively traded of excessive hardware resources to clock at higher speed while limited to the same power budget (2W). In fact, all non dominated solutions can be verified as the locally optimal solution for their instantaneous power budgets. The pruned regions (V5, V6, V7, V8) and (V15, V16) are the overlapped regions between the design variable domains having inferior one to many correspondence in other dimensions of the performance space.

5. The DSE explorer expands the solution front, traversing seamlessly through the local domains of the arranged variables trading area with increasing clock rate to the point of border variant, with global

constrained power budget of 11 Watt. *Variant 19* represents the border variant consuming 10.8 watts of power as shown below:

$$P_{(Variant\_19)} = (N_{R1.1}.K_{R1} + N_{R2.2}.K_{R2}).R_{3.3}$$
$$= (1 \times 20 + 2 \times 80) \times 60 = 10.8 \text{ watt} \quad (43)$$

Thus, two multipliers and an adder/ subtractor unit clocking at 100 MHz provide the best execution speed performance with minimal area. Table 1 summarizes the progression of the solution front through the ordered design variable space. Traversing the design space and pruning variants as per the *condition of stability* (eq. 27), the DSE explorer identifies not only the *global optimal* solution at the border variant but also all the *locally optimal* solutions, up to the limiting conditions of system optimization. To accelerate the search process, the DSE framework incorporates multi dimensional binary search instead of linear search along with the design pruning conditions. Interested readers can refer to [21,6] for a detailed explanation on multi-dimensional binary search.

4.11 Module Synthesis: Quick Overview

This section presents an overview of IIR filter synthesis at RTL level. The author's previous publications can be referred to for a more detailed explanation on the topic [33]. This section looks at the most important stages of modular design synthesis including development of *data path*, *control path*, *Sequencing* and *binding* of resources, determination of *control unit* and *multiplexing scheme* for the IIR filter case study.

4.12 Scheduling: Sequencing and Resource Binding

Sequencing graphs (SG) [12] are an effective way to represent the spatial and temporal domain of the sequence of operations applied on application data set. Scheduling algorithms sequence the system resources in time and space to effectively map the behavioral function on the module data set. This work uses ASAP as the standard scheduling algorithm for the presented case study. Fig. 15 shows the ASAP scheduling graph of IIR optimal architectural variant. The figure also shows the binding of system resources, namely the adder and multiplier units in different time slots along with peripheral registers P holding interleaved data.

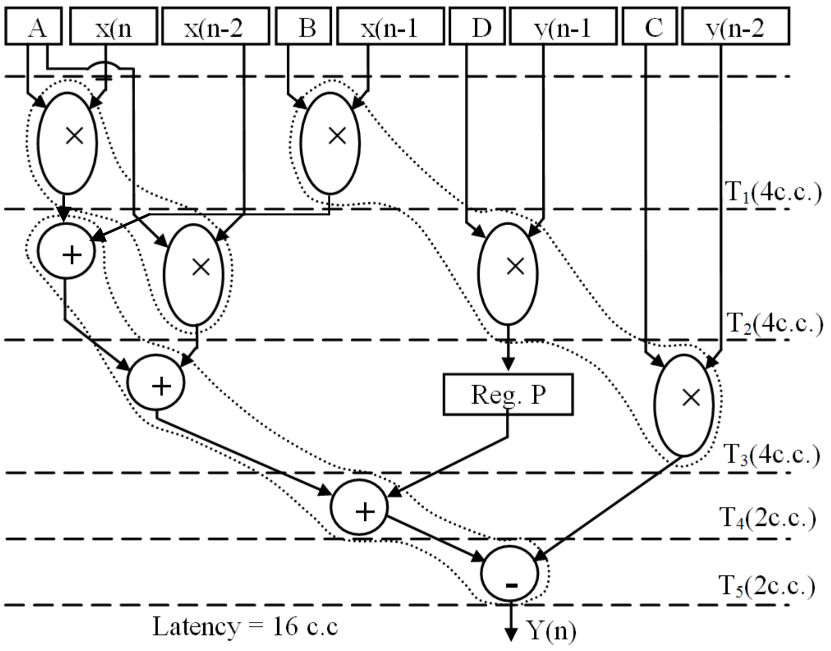

Figure 15. Sequencing and binding graphs of the optimal variant (#19)

Time	**Operation**	**Input 1**	**Input 2**	**Output**
0	No-Op	Reg. A	Reg. x(n)	---
1	×	Reg. A	Reg. x(n-2)	R3 in
2	×	---	---	R3 in
3	No-Op	---	---	---
4	No-Op	---	---	---

Table 2. Multiplexing scheme for resource ‘Multiplier 1’

Time	**Operation**	**Input 1**	**Input 2**	**Output**
0	No-Op	Reg. B	Reg. x(n-1)	---
1	×	Reg. D	Reg. y(n-1)	R3 in
2	×	Reg. C	Reg. y(n-2)	R3 in
3	×	---	---	R3 in
4	No-Op	---	---	---

Table 3. Multiplexing scheme for resource ‘Multiplier 2’

Time	**Operation**	**Input 1**	**Input 2**	**Output**
0	No-Op	---	---	---
1	No-Op	R1 out	R2 out	---
2	+	R1 out	R3 out	R3 in
3	+	R3 out	R2 out	R3 in
4	+	R3 out	R2 out	R3 in
5	–	---	---	Reg. Y

Table 4. Multiplexing scheme for resource ‘adder/ subtractor’

4.13 Determination of Multiplexing Scheme:

Binding of resources in the presented design flow warrants a multiplexing and de-multiplexing scheme to maintain the continuous and desired data flow through the data processing engine [12]. A multiplexing scheme works closely with a *control unit,* which is detailed separately in the preceding sub-sections. The main task of the multiplexing scheme is to configure the processing elements in the data path before subjecting them to the data to be processed. Tables 2, 3 and 4 summarize the necessary multiplexing scheme for the multiplier, adder/ subtractor respectively clocking at the desired frequency.

4.14 Module Block Representation:

Once the module micro-architectural configuration is determined along with the associated multiplexing/ de-multiplexing scheme the design can be converted into its equivalent Register Transfer Level (RTL) representation. Fig.16 shows the IIR filter module with optimal design variant resources embedded in the data path along with the associated control path signals. In addition to processing elements and multiplexing scheme the block representation also includes necessary memory elements, latches and the temporary register elements. The processing elements embedded in the *data path* impart its characteristic function to the implied data flowing through well defined interfaces in compliance with application data structure. The module works in complete synchronization under the control unit.

4.15 Control Path Description:

The control unit provides the necessary timing and synchronization signals to different processing elements. For the synchronous flawless operation of the system, the control unit must produce the desired

	Clock →																											
Resources ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
R1(Multipler1)																												
Selector (x)	0	0					1																					
ip_L_Strobe (x)	0	1	0					1	0																			
Enable (x)	0		1	x	x	x	0		1	x	x	x	0															
op_L_Strobe(x)	0						1	0					1	0														
Deselector (x)	0					0						1																
R2(Multipler2)																												
Selector (xx)	00						01						10															
ip_L_Strobe (x)	0	1	0					1	0					1	0													
Enable (x)	0		1	x	x	x	0		1	x	x	x	0		1	x	x	x	0									
op_L_Strobe(x)	0						1	0					1	0					1					0				
Deselector (xx)	00					00						01						10										
R3(Addr/Subt)																												
Selector (xx)	00						00						01						10				11					
ip_L_Strobe (x)	0						0	1	0					1	0					1	0			1	0			
Enable (x)	0							0	1	†	0				1	†	0				1	†			1	—		
Op-Code (x)	0							0						0						0				1				
op_L_Strobe(x)	0										1			0			1			0			1	0			1	0
Deselector (xx)	00									00						01						10				11		
Strb_Reg. P	0													1						0								
Strb_Reg. Y	0																											1
Busy	0	1																										0

Table 5. Control Unit timing specifications. The coordinates represent the system clock cycles along horizontal axis and processing element resources with associated control signals along vertical axis.

signals at exactly the right time. Failure to observe the timing description, as shown in Table 5, can result in fatal consequences, including latch failing issues and wrong or delayed output. The rows of the control unit table indicate the resources and associated micro signals. The columns show the time in system clock cycles. The multiplier resource is given a 4 cycle processing time and the adder takes 2 clock cycles for data processing as per the design library specifications.

4.16 Schematic capture of the module in ISE simulator:

Schematic capture provides a good way to visualize the structure and connectivity at RTL level. Fig. 17 shows a snapshot of the IIR filter module consisting of systems data path elements and associated control path signals (courteously Xilinx ISE design suite V10.1). The elements used in the data path of the IIR module are individually tested and verified for parametric extraction and behavioral functional verification. The design was implemented, tested and verified on Xilinx Spartan III E FPGA.

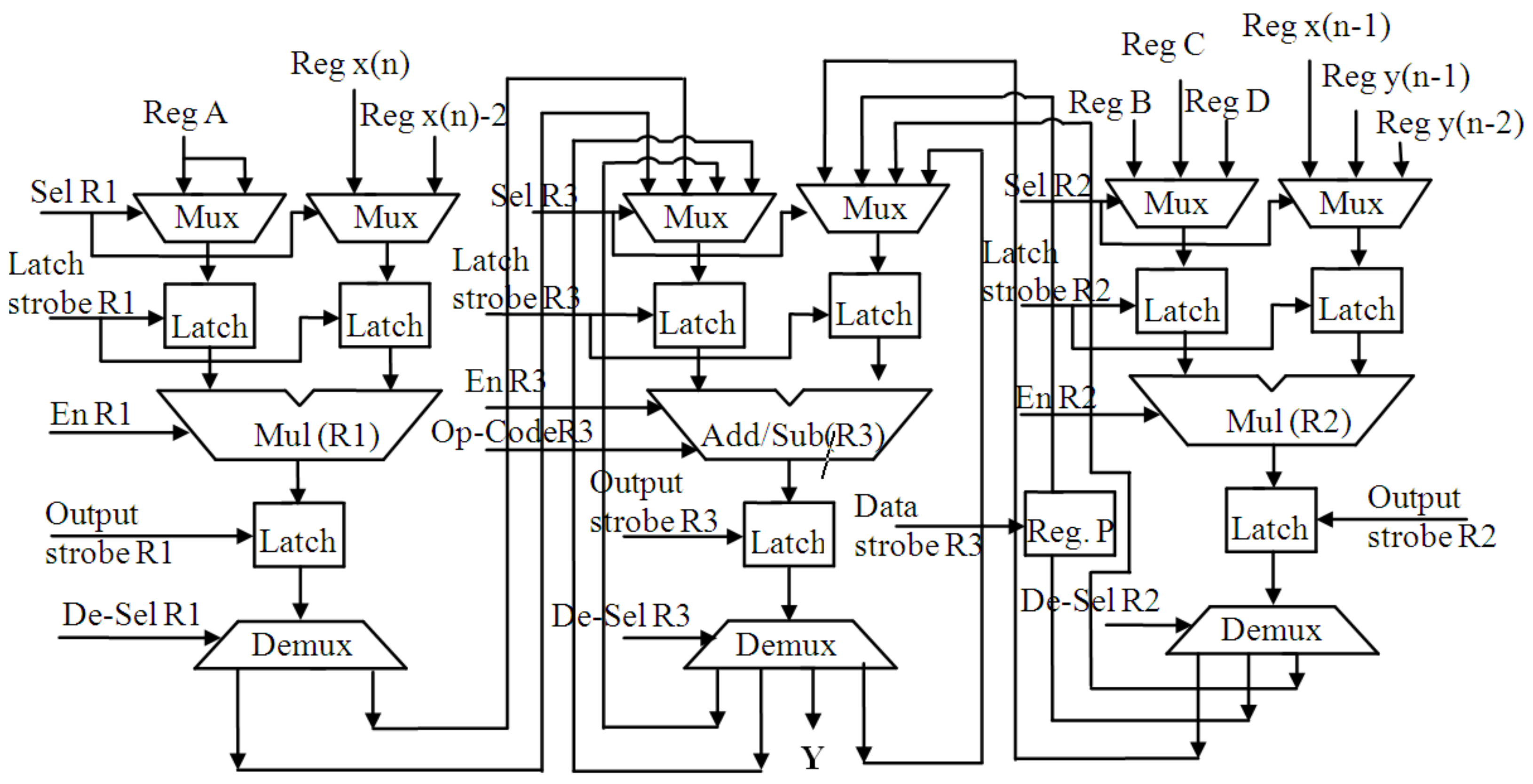

Figure 16. Block Diagram of the system data path description

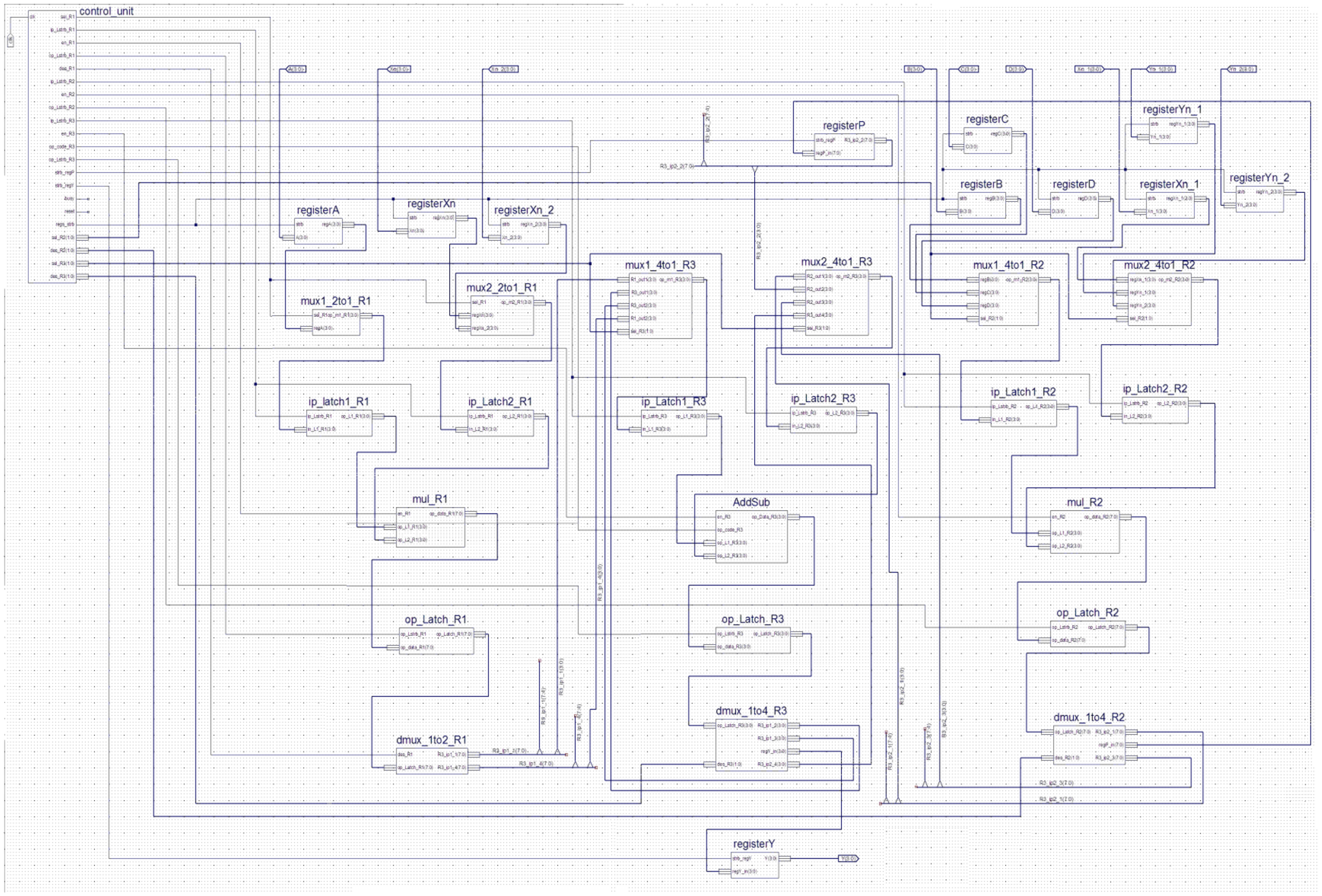

Figure 17. Schematic capture view of the system (Xilinx ISE 10.1)

Chapter 5

Results and Analysis

5.1 Simulation Results

The IIR filter design was tested pre implementation and post implementation with ISE simulator. The design was targeted at Spartan III E FPGA platform. Fig. 18 shows the ISIM simulator results. The design was stress tested for a variety of input test vectors and the exact results indicate the success of the design implementation at silicon level. The synthesized design was implemented and converted into bit stream and targeted at the development hardware for in circuit implementation. The circuit shows expected results as per the IIR filter synthesis function and thereby verifies the successful design implementation.

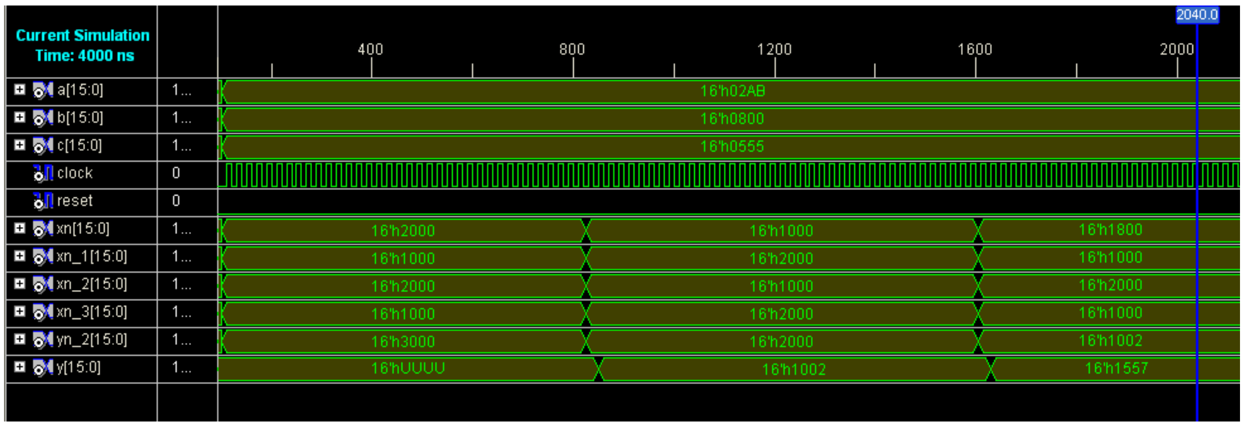
Current Simulation Time: 4000 ns
2040.0
400
800
1200
1600
2000
a[15:0] 1... 16'h02AB
b[15:0] 1... 16'h0800
c[15:0] 1... 16'h0555
clock 0
reset 0
xn[15:0] 1... 16'h2000 16'h1000 16'h1800
xn_1[15:0] 1... 16'h1000 16'h2000 16'h1000
xn_2[15:0] 1... 16'h2000 16'h1000 16'h2000
xn_3[15:0] 1... 16'h1000 16'h2000 16'h1000
yn_2[15:0] 1... 16'h3000 16'h2000 16'h1002
y[15:0] 1... 16'hUUUU 16'h1002 16'h1557

Figure 18. IIR filter synthesis application simulation results

5.2 Benchmark Results:

The proposed DSE design flow was applied to a number of well known High Level Synthesis benchmark applications to ascertain the exploration speedup. These benchmarks were adopted from [30] for HLS studies. The proposed algorithm demonstrated an impressive 71.8 % average speed up over a range of the tested benchmark applications. Table 6 summarizes the observed speed up attained for the optimal architecture variant selection. The observed DSE speedup was more pronounced for larger benchmark applications (i.e. FIR was observed at 85.83% acceleration) indicating the practical scalability of the proposed approach to large scale applications. Table 6 summarizes the number of variants in the design space, the total number of variants for along all the parameter directions, the actual number of variants analyzed by the proposed DSE approach and the comparative speedup achieved in DSE exploration. Fig. 19 elaborates upon the DSE speedup progression through various benchmark applications viz. IIR digital filter, ARF Filter, DWT transformation, Elliptic Wave filter and MPEG motion vector.

DSP Benchmarks	Architectural variants in the Design space	Total architectural variants in the design space for exhausted search with two parameters	Total Variants Analyzed	Speedup of the proposed approach	Average Speedup
IIR Digital Chebyshev Filter	48	96	49	48.95 %	71.80 %
Auto Regressive Filter (ARF)	144	288	93	67.70 %	
Discrete Wavelet Transformation	216	432	130	69.97 %	
Elliptic Wave Filter (EWF)	225	450	108	76.00 %	
MPEG Motion Vectors(MMV)	378	756	133	82.40 %	
Finite Impulse Response (FIR)	600	1200	170	85.83 %	

Table 6. Experimental results of the proposed hybridized approach for Benchmark applications
*(**Note**: The DSP Benchmarks used in the experiments are well known High-Level Synthesis Benchmarks adopted from [30])*

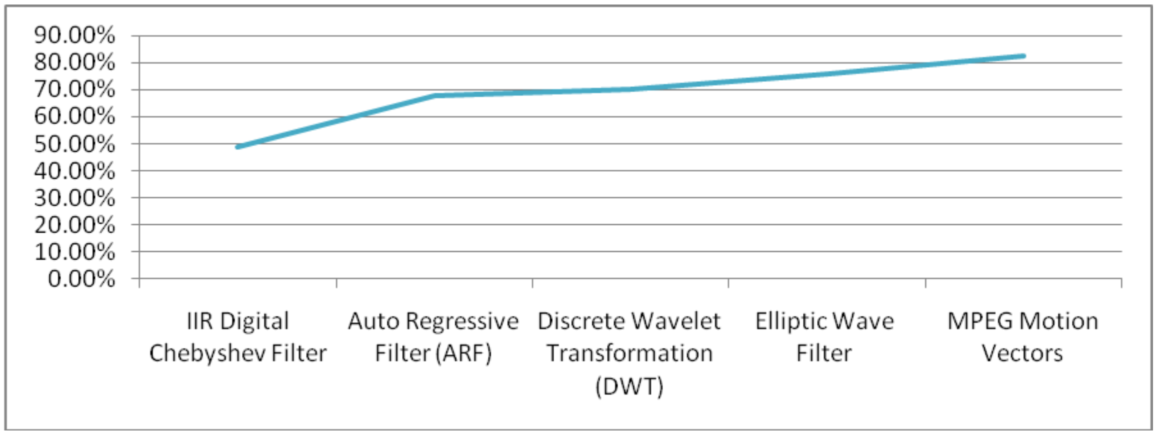

Figure19. The strong scalability property of the proposed approach is highlighted where the speedup (blue curve) rises with the increase in size of the design space

Fig. 20.b shows the fully arranged design space of the FIR benchmark application with power performance parameter. This increasing power performance parameter was attained by keeping the solution exploration front along the non-dominated variants of the solution space. In comparison, Fig. 20.a shows a partially arranged design space for the same application. As evident, the partially arranged design space has one-to-many correspondence(s) in the gaps where the design space variable domain overlaps. In contrast, the fully arranged design space with restricted domains, as shown in fig. 20.b, has an absolute one-to-one correspondence for any given performance specification mapped onto the design space variables.

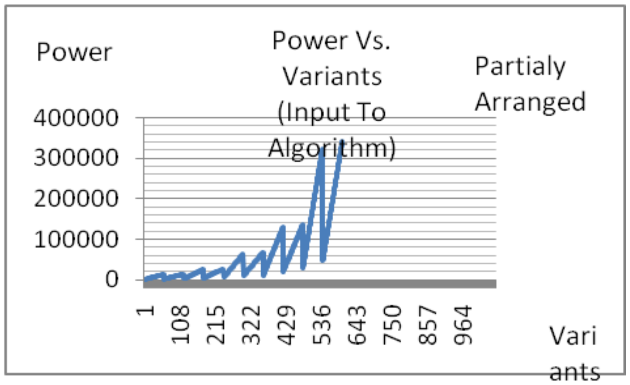

Figure20 (a) Partially Arranged FIR filter power parameter

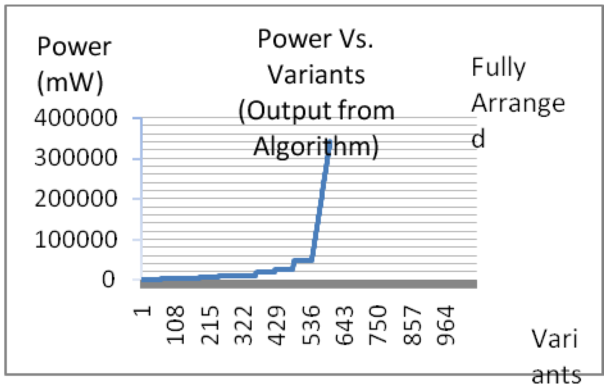

Figure20 (b) Fully Arranged FIR filter power parameter

5.3 Complexity Analysis

The presented work converts the partially arranged design space to absolutely arranged design space. When the design space variables are arranged/ explored in PF hierarchal order, the cost function increases linearly but in sections depending upon the overlapping of domains. This results in the partially arranged design space. The presented framework applies the binary search just across each of the partially arranged sections and look for the point of balance where an active/base variable combines to balance the next variable in hierarchy.

The complexity of the binary search is observed to be of the order of *$log_2 n$*, where '*n'* is the number of member variants in the search space. Therefore the search computational complexity of the instantaneous point of balance would require $\log_2 \prod_{i=1}^{n} v_{Ri}$ (where 'n' = number of type of resources and 'v_{Ri}' is the number of variants of resource 'Ri') for each arranged section of the partially arranged design space. If there are '*n'* variables in the design space that would correspond to *n* arranged sections in partially arranged design space. Each of the arranged section of the design space is subjected to binary search therefore, the complexity across a given performance cost objective would be of the order of $n * \log_2 \prod_{i=1}^{n} v_{Ri}$, where n is the number of design variables in DSE space.

The proposed DSE framework searches along the multiple objectives of the performance space. For example in the case study power consumption, area and execution speed constitutes three dimensions of the performance space. Therefore the search has to be performed thrice along each of the scaled dimension. Similarly for '*M'* performance parameters, the computational complexity of the proposed framework will be of the order of:

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

Although the complexity is quadratic but is still a lot faster (of the order of 72%) than in comparison to an exhaustive search, which would require $M * \prod_{i=1}^{n} v_{Ri}$ while achieving the accuracy just as good as in an exhaustive approach. The next section would compare and contrast the presented work to an existing approach which achieves faster exploration by the factor of n (number of design variables), but readily comprises on achievable accuracy resolution.

5.4 Comparative Studies

This sub-section compares the presented framework computational complexity with an existing DSE approach [21]. The authors in [21] have subjected the partially arranged design space to multi-dimensional binary search min-max analysis to arrive at a suitable design variants. Table 7 shows the relative speedup achieved by the proposed framework in comparison to the [21] DSE framework. Although, as apparent the presented framework is 'n' times computationally slower than the compared existing work but it has higher accuracy resolution to the factor of least significant PF variable to come closest to the desired cost objective (budget). The presented framework only traverses the most suitable/desirable range of the partially arranged design space and changes the base variable to the next variable in PF hierarchy to keep the cost function absolutely increasing. In contrast, although the compared framework [21] achieves a significant speedup in comparison by a factor of 'n', but can easily get stuck in the local optima of the partially arranged design space and compromise upon the achievable solution accuracy. The absolute figures of the solution accuracy comparison are subjective and will wary from one example to other.

Benchmarks	Total possible architecture in the design space for exhaustive search for two parameters	Architecture evaluation using Hierarchical arrangement of the ACG binary search [21] (Number of variants analyzed)	Architecture evaluation using Proposed method (Number of architecture)	Exploration Speed up difference compared to an existing DSE approach [21]	Speedup using proposed approach compared to the exhaustive search
IIR Digital Chebyshev Filter	96	12	49	-38.6%	48.95%
Auto Regressive Filter (ARF)	288	24	93	-23.89%	67.70%
Discrete Wavelet Transformation (DWT)	432	26	130	-24.8%	69.97%
Elliptic Wave Filter (EWF)	450	29	108	-17.55%	76.00%
MPEG Motion Vector (MMV)	756	28	133	-13.89%	82.40%

Table 7. Experimental results of comparison between the proposed DSE approach with the current approach (kirischians) [21] for benchmark applications

5.5 Scope of Framework Application and Known Limitations

5.5.1 Data Path vs. Control Unit

The presented work optimizes the data path as per the given constraints, performance specifications and behavioral description of the application. The optimization framework is concentrated upon utilizing the given data processing elements in the most efficient manner. But as a consequence or side-effect of optimizing a given resource in the data path, the associated multiplexing and control circuitry/signals gets complex. Although not as dramatic as to completely offset the benefits extracted from the data path optimization, the control unit complexity does effect the overall improvement in terms of cost/area savings, power utilization etc.

As a rule of thumb a definite cost weight can be subjected to achievable improvements to make corrections for the actual/ achievable performance figures.

5.5.2 Relationship in between design space variables

In order for the design space exploration to readily applicable to multiple dimensions of the multi-objective optimization problem, the design variables are expected to have a definite relationship which can be directly in-order or exactly reverse order. The parsed space by the ‘Stability in Competition’ principle would than correspond and hold to each of the performance space of the multi-objective optimization problem.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis has presented a novel framework to ascertain the optimal design architectural configuration from a large design space. This framework is based upon the order and extent of design variable exploration. The order in design space exploration variable helps determine the border variant of the feasible solution space, whereas the extent of variable exploration helps keep the solution front from exploring the redundant regions of the design space containing diminishing returns. The well known economics Stability in Competition [16] principle is applied in this novel approach to restrict the extent of ordered resource exploitation. As a result, this work is successfully able to attain absolute and fully arranged design space along the performance parameter axis.

The improved and fully arranged design space is subjected to multi-dimensional binary search for the optimal design variant selection. This framework presents an improvement over existing approaches, which are prone to be caught in local optimum. A robust set of mathematical formulations along with the graphical representation of the framework is presented, which can be applied to real world synthesis applications with minimal human intervention. An average 71.8% speedup in design space exploration was achieved while maintaining prime accuracy resolution and better speedups for larger design space benchmark applications. The improvement in speedup for larger design space problems shows the scalability of the framework to large scale applications. A thorough case study of IIR filter synthesis application has been presented in Chapter 5, Chapter 6 elaborates upon the results. This high level synthesis design space exploration approach is applicable to a very large number of commercial applications at the system architecture level. Based upon the required performance specifications and various budgeting constraints this framework can effectively translate high level design requirements into low level system architecture configurations for a given set of library components.

This approach can also be integrated in Electronic Design automation (EDA) tool chain sets to determine the optimal modular configuration for the contradictory performance requirements/ constraints. The quest of modern embedded systems for high portability and enormous processing power lends itself to the contradictory parameters of speed and power consumption numbers. This approach can help determine the optimal architecture configuration without the overkill of any of the available design resources. It can thus serve as the stepping stone between the behavioral and structural stage of the high level synthesis design flow.

6.2 Future Work:

This approach can be applied in real time to reconfigure the system architecture data path as per the instantaneous design specifications and design rules. The reconfigurable hardware can have redundant parallelization along the data path and the custom requirements can be mapped in real time to reflect the instantaneous design goals, budgeting constraints and loads. Rigorous mathematical formulations and algorithm(s) have been presented to ascertain the optimal design configuration for a given set of high level requirements.

Secondly, the presented work concentrates on the synthesis application data path optimization. This concept can also be expanded for control path optimization. Control path optimization poses specialized and even more complex problems as the control unit and multiplexing scheme becomes affected by data path optimization. A thorough analysis can be performed and the framework can be worked out taking into account the data path and control path tradeoff analysis for a given HLS application as well as work load and budgeting constraints.

Publications

Refereed Journals

1. Summit Sehgal, Reza Sedaghat, Anirban Sengupta, "ESL Design Optimization- Multi Objective Design Space Exploration in High Level Synthesis for DSP Applications", Microelectronics Reliability journal, Science Direct, Elsevier, Submitted, 2010

2. Summit Sehgal, Reza Sedaghat, Anirban Sengupta, "Design for Performance: Multi-Objective Design Space Exploration in High Level Synthesis for DSP Applications", Microelectronics journal, Science Direct, Elsevier, Submitted, 2010

Refereed Conferences

3. 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), 2009, pp 89-94

4. Summit Sehgal, Reza Sedaghat, Anirban Sengupta, "Automated Design Space Exploration for DSP Applications High Level Synthesis with Stability in Competition", Accepted for publication in the IEEE Latin America Symposium on Circuits and Systems, Bogota, Colombia, 2011, to be presented on Feb. 23, 2011

Other peer reviewed poster publications:

5. Summit Sehgal, Reza Sedaghat, Anirban Sengupta, "Fault Monitoring Transformer Reliability ASIC Design based on Ringing Effect Signature Analyzer", published in Research Innovation Symposium Proceedings, Ryerson University, pp 32, April 29, 2010.

References

[1] Adam Postula, David Abramson, Ziping Fang, Paul Logothetis, “A Comparison of High Level Synthesis and Register Transfer Level Design Techniques for Custom Computing Machines”, Proceedings of the Thirty-First Hawaii International Conference on System Sciences, vol.7 , 6-9 Jan 1998, pp. 207-214

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

[3] Alessandro G. Di Nuovo, Maurizio Palesi, Davide Patti, “An Hybrid Soft Computing Approach for Automated Computer Design”, Proceedings of the Third Starting AI Researchers' Symposium, STAIRS 2006, pp. 84-95

[4] Almitra Pradhan, Ranga Vemuri, “Fast Analog Circuit Synthesis Using Sensitivity Based Near Neighbor Searches”, in DATE 2008, pp.523-526.

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

[6] 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”, International Journal of Microelectronics Reliability, Science Direct, Elsevier, 2010,Volume 50, Issue 3, pp. 424-437.

[7] Christian Haubelt, Jurgen Teich,“Accelerating Design Space Exploration Using Pareto-Front Arithmetics”, in Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC’03), Japan, 2003.

[8] Davide Bruni, Alessandro Bogliolo, Luca Benini, "Statistical design space exploration for application-specific unit synthesis", Proceedings of the 38th conference on Design automation 2001, ISBN: 1-58113-297-2, pp. 641-646

[9] Diviya Jain, Anshul Kumar, Laura Pozzi, Paolo Ienne "Automatically Customising VLIW Architectures with Coarse Grained Application-Specific Functional Units", Software and Compilers for Embedded Systems Lecture Notes in Computer Science, 2004, Volume 3199/2004, pp. 17-32, DOI: 10.1007/978-3-540-30113-4_3

[10] Georges Gielen, Trent McConaghy, Tom Eeckelaert, "Performance Space Modeling for Hierarchical Synthesis of Analog Integrated Circuits", in DAC 2005, pp. 881-886.

[11] Gianluca Palermo, Cristina Silvano, Vittorio Zaccaria, "Multi-objective design space exploration of embedded systems", Journal of Embedded Computing, IOS press, Volume 1, Number 3/2005, pp. 305-316

[12] Giovanni De Micheli , Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994. Chapter 4 Architectural Synthesis, pp. 141-183.

[13] Giuseppe Ascia, Vincenzo Catania, Alessandro G. Di Nuovo, Maurizio Palesi, Davide Patti, "Efficient design space exploration for application specific systems-on-a-chip", Journal of Systems Architecture Volume 53, Issue 10, October 2007, pp. 733-750

[14] Gul N. Khan and U. Ahmed, "CAD tool for Hardware Software Cosynthesis of Heterogeneous Multiple Processor Embedded Architectures", Int. Journal Design Automation for Embedded Systems, Vol. 12, No. 4, pp. 313-343, December 2008

[15] Hans Kellerer, Ulrich Pferschy, David Pisinger, Knapsack Problems, Springer 2004, 1 edition, ISBN-10: 3540402861

[16] Harold Hotelling, “Stability in Competition”, The Economic Journal, (Mar., 1929), vol. 39, no. 153, pp. 41-57.

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

[18] John C. Gallagher, Vigraham Saranyan, and Gregory Kramer,“A family of compact genetic algorithms for intrinsic evolvable hardware,” IEEE Trans. Evol. Comput., Apr. 2004, vol. 8, no. 2, pp. 1–126.

[19] Jürgen Teich, “Pareto-Front Exploration with Uncertain Objectives”, EMO 2001 Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, ISBN:3-540-41745-1, pp. 314-328

[20] Kamal S. Khouri, Ganesh Lakshminarayana, Niraj K. Jha, “High-Level Synthesis of Low-Power Control-Flow Intensive Circuits”, IEEE Transaction on computer-aided design of integrated circuits and systems, vol. 18, no. 12, December 1999, pp. 1715-1729

[21] Lev Kirischian, Vadim Geurkov, Valeri Kirischian, Irina Terterian (2006), “Multi-parametric optimisation of the modular computer architecture", Int. J.Technology, Policy and Management, Vol. 6, No. 3, pp.327–346.

[22] Luca Benini, Giovanni de Micheli, “System-level power optimization: techniques and tools”, ACM Transactions on Design Automation of Electronic Systems, Volume 5, Issue 2, April 2000, pp. 115-192

[23] Manuel Barros, Jorge Guilherme, Nuno Horta, “GA-SVM Feasibility Model and Optimization Kernel Applied to Analog IC Design Automation”, in GLSVLSI 2007, Italy, pp. 469-472.

[24] Matthias Gries, “Methods for evaluating and covering the design space during early design development”, Integration, the VLSI Journal, Volume 38 Issue 2, December 2004, pp 131 – 183

[25] Michael Eisenring, Lothar Thiele, and Eckart Zitzler, “Conflicting criteria in embedded system design”, IEEE Design & Test, Volume 17 Issue 2, April 2000, ISSN: 0740-7475, pp. 51-59

[26] Pawan K. S. Nain, Kalyanmuy Deb, “Computationally Effective Search and Optimization Procedure Using Coarse to Fine Approximations”, proceedings of the 2003 Congress on Evolutionary Computation, Vol.3, 8-12 Dec. 2003, pp. 2081-2088

[27] Pierre g. Paulin, John P. Knight, “Algorithms for High-Level Synthesis”, IEEE Design & Test of Computers, Volume 6, Issue 6, Dec. 1989, pp. 18-31

[28] Rajiv Dutta, Jayanta Roy, Ranga Vemuri “Distributed Design-Space Exploration for High-Level Synthesis Systems”, Proceedings of the 29th ACM/IEEE Design Automation Conference, DAC 1992, pp. 644 – 650

[29] Rob A. Rutenbar, Georges G. E. Gielen, Jaijeet Roychowdhury, “Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs”, Proceedings of the IEEE, March 2007, pp. 640-669

[30] 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, pp. 5-46

[31] Soumya Pandit, Chittaranjan Mandal, Amit Patra, “Systematic Methodology for High-Level Performance Modeling of Analog Systems”, Proceedings of 22nd International Conference on VLSI Design, New Delhi, pp. 361-366

[32] Sumit Gupta, Nick Savoiu, Sunwoo Kim, Nikil Dutt, Rajesh Gupta, Alex Nicolau, “Speculation Techniques for High Level Synthesis of Control Intensive Designs” in DAC 2001, pp. 269-272.

[33] 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), 2009, pp. 89-94

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

[35] Youn-long lin, “Recent Developments in High-Level Synthesis”, ACM Transactions on Design Automation of Electronic Systems, Vol. 2, No. 1, January 1997, pp. 2–21

[36] Yu Liu, Masato Yoshioka, Katsumi Homma, Toshiyuki Shibuya, “Efficiently Finding the ‘Best’ solution with Multi-Objectives from Multiple Topologies in Topology Library of Analog Circuit”, in ASP-DAC 2009, pp. 498-503.

[37] Zhipeng Zeng, Reza Sedaghat, Anirban Sengupta, “A Framework for Fast Design Space Exploration using Fuzzy search for VLSI Computing Architectures”, IEEE International Symposium on Circuits and Systems (ISCAS), 2010, pp. 3176-3179.