

1-1-2013

Rapid And Efficient Multi Objective Design Space Exploration Methods In High Level Synthesis Of Computation Intensive Applications

Anirban Sengupta
Ryerson University

Follow this and additional works at: <http://digitalcommons.ryerson.ca/dissertations>

 Part of the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Sengupta, Anirban, "Rapid And Efficient Multi Objective Design Space Exploration Methods In High Level Synthesis Of Computation Intensive Applications" (2013). *Theses and dissertations*. Paper 1631.

This Dissertation is brought to you for free and open access by Digital Commons @ Ryerson. It has been accepted for inclusion in Theses and dissertations by an authorized administrator of Digital Commons @ Ryerson. For more information, please contact bcameron@ryerson.ca.

RAPID AND EFFICIENT MULTI OBJECTIVE DESIGN SPACE EXPLORATION METHODS IN HIGH LEVEL SYNTHESIS OF COMPUTATION INTENSIVE APPLICATIONS

By

Anirban Sengupta

Master of Applied Science

Electrical and Computer Engineering

Ryerson University, Toronto, Canada, 2010

Bachelor of Technology

Electronics and Communication Engineering

West Bengal University of Technology, Kolkata, India, 2008

A dissertation

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2013

©Anirban Sengupta 2013

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

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

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.

I understand that my dissertation may be made electronically available to the public.

* Signature

Anirban Sengupta

ABSTRACT

Title of Dissertation:

**RAPID AND EFFICIENT MULTI OBJECTIVE DESIGN SPACE EXPLORATION
METHODS IN HIGH LEVEL SYNTHESIS OF COMPUTATION INTENSIVE
APPLICATIONS**

Dissertation Submitted By:

Anirban Sengupta, Doctor of Philosophy, 2013

Electrical and Computer Engineering Department, Ryerson University, Canada

Dissertation Directed By:

Dr. Reza Sedaghat

Electrical and Computer Engineering Department, Ryerson University, Canada

Design Space Exploration (DSE) is an indispensable segment of the High Level Synthesis (HLS) design process. Moreover, the enormous increase in complexity of the recent Very Large Scale Integration (VLSI) circuits has only been possible due to use of advanced DSE techniques during HLS process. This dissertation presents four automated optimization algorithms and methodologies that are capable to handle various multi-objective problems during design space exploration and high level synthesis of computation intensive applications. Algorithmic solutions to four different branches of DSE problems have been proposed in this dissertation viz. a) Solution to power-performance-area/cost trade-off of Digital Signal Processing (DSP) kernels using priority factor process which also includes deriving analytical mathematical model for modern performance parametric frameworks b) Solution to area-performance-power tradeoff/ power-performance-area tradeoff of DSP kernels using hybridization of fuzzy algorithm and

vector design space technique with Self-Correction Scheme c) Solution to dual parametric optimization using efficient multi structure genetic algorithm for integrated scheduling and allocation and d) Solution to control step bound static power optimization using power gradient methodology for integrated scheduling and allocation. Some techniques proposed are equipped with pipelined execution time parameter (based on need), in addition to hardware area, power and cost depending on the user's objective for exploration of a final solution in a short time. In addition to architecture exploration capability, rapid automated circuit generation of DSP kernels is also possible in a short time for verification and synthesis in Field Programmable Gate Array (FPGA) platforms. The proposed exploration approaches are applied to custom data intensive applications (application specific processors/custom processors) or standalone Application Specific Integrated Circuits (ASIC's). Results of the experiments for proposed approaches on all the standard DSP benchmarks have indicated improvements either in terms of exploration runtime, quality of final solution, reduced execution time, power and area or a multiple combination of all factors when compared to recent approaches.

Acknowledgement

I would like to thank my supervisor, Dr. Reza Sedaghat for his thoughtful guidance and advice as well as OPR-AL members for their endless support. Further, I am also thankful to my supervisor for providing me all the necessary support and required amenities to help me perform my incessant research for all the past years.

I am highly indebted to my parents for their great guidance and sacrifice all throughout my life. Further I highly owe them for being a constant source of love and motivation throughout my life, particularly in times of hardships and difficulty. They have always been a source of true guide inducing the feeling of eternal divine power in me.

Moreover I am highly obliged to my grandparents for continuously supporting me and inspiring me to always do better than before. I deeply express my gratitude for their eternal blessings.

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

Abstract -----iii

Acknowledgement -----v

Table of Contents -----vi

List of Tables -----ix

List of Figures -----xi

Nomenclature -----xiii

Chapter 1 Introduction -----1

 1.1 Overview -----1

 1.2 Fundamentals on Modular System Design -----3

 1.3 Related Work-----4

 1.4 Background Information-----7

 1.5 Summary of Contribution -----12

 1.6 Organization of Dissertation -----14

**Chapter 2 Rapid Design Space Exploration in High Level Synthesis Based on Power-
Performance Tradeoff using Priority Factor Metric-----16**

 2.1 Mathematical Derivation for Cost/Area Model-----17

 2.2 Mathematical Derivation for Execution Time Model-----20

 2.3 Mathematical Model for Power Consumption -----23

 2.4 Proposed Method of Design Space Exploration of Architecture Based on Power-
Performance tradeoff with area/cost as optimization criteria-----24

Chapter 3 Design Space Exploration in High Level Synthesis for Area/Power-Performance Tradeoff using Hybridization of Fuzzified Algorithm and Vector Design Space with Self-Correction Scheme-----	31
3.1 The Proposed Theory for Fuzzy Search during Design Space Exploration-----	32
3.2 The Steps Needed to Obtain the Final Variant of Architecture-----	38
Chapter 4 Priority Function Driven Design Space Exploration in High Level Synthesis Based on Power Gradient Technique-----	53
4.1 The Proposed Exploration Approach-----	54
4.2 Demonstration of the proposed approach-----	56
Chapter 5 A Multi Structure Genetic Algorithm for Integrated Design Space Exploration of Scheduling and Allocation in High Level Synthesis for DSP Kernels-----	61
5.1 The Proposed Framework Using MSGA-----	63
5.2. Description of the Proposed MSGA steps-----	68
Chapter 6 Fast Multi-Objective Exploration and RTL Circuit Generation using Architecture Synthesis Platform: Exploration Synthesizer-----	82
6.1 The Proposed Exploration Synthesizer Design Flow-----	83
6.2 Keystones of the Proposed Exploration Synthesizer Platform-----	86
6.3 Input Format and Intermediate Representation-----	87
6.4 Output Details of the Tool-----	88
Chapter 7 Implementation, Results and Analysis -----	90

7.1 Experimental results: The Proposed Approach ‘Rapid Design Space Exploration in High Level Synthesis Based on Power-Performance Tradeoff using Priority Factor Metric’ and comparison with recent approach -----	91
7.2 Experimental results: The Proposed Approach ‘Rapid Design Space Exploration in High Level Synthesis Based on Area-Performance and power-performance Tradeoff using Hybrid Fuzzified Algorithm’ and performance comparison ---	95
7.3 Experimental results: The Proposed Approach ‘Priority Function Driven Design Space Exploration in High Level Synthesis Based on Power Gradient Technique’ and comparison with recent approach-----	102
7.4 Experimental results: The Proposed Approach ‘A Multi Structure Genetic Algorithm for Integrated Design Space Exploration of Scheduling and Allocation in High Level Synthesis for DSP Kernels’ and comparison with recent approach -----	105
Chapter 8 Conclusion and Future work -----	108
Publications -----	111
Appendix -----	118
References-----	123

List of Tables

Table 1	System Specifications and Constraints for PF Method-----	26
Table 2	System Specifications for hybrid fuzzy approach -----	40
Table 3	Variants obtained for area after applying fuzzy search on the arranged design space-----	48
Table 4	Variants obtained for execution time after applying fuzzy search on the arranged design space -----	51
Table 5	Portion of Module Library for MSGA-----	77
Table 6	Data Extracted from the integrated solution of offspring 1-----	78
Table 7	Experimental results of comparison between proposed PF approach and recent GA approach-----	94
Table 8	Experimental results of comparison between the proposed hybrid Fuzzified DSE for Area-Performance trade-off with Power as optimization criteria with the current approach for large benchmarks -----	97
Table 9	Experimental results of comparison between the proposed hybrid Fuzzified DSE for Area-Performance trade-off with Power as optimization criteria with GA based current approach -----	98
Table 10	Experimental results of the proposed hybrid Fuzzified approach for Power-Performance trade-off with Area as optimization criteria compared with exhaustive analysis for Benchmarks -----	100

Table 11	Experimental results of the comparison between the proposed DSE for Power-Performance trade-off with Area as optimization criteria with the recent approach -----	101
Table 12	Experimental Results of the proposed Iterative Power Gradient approach for the DSP Benchmarks -----	102
Table 13	Comparison of measured power consumption through Xilinx Power Analyzer (XPA) 9.2i -----	103
Table 14	Experimental Results of the comparison between the proposed Iterative Power Gradient and recent approach -----	104
Table 15	Experimental Setup for MSGA-----	105
Table 16	Experimental Results of Comparison of MSGA with recent approach ---	106

List of Figures

Figure 1	Generic Overview of the Proposed PF Exploration Method-----	26
Figure 2	Flow chart model of the proposed algorithm -----	27
Figure 3	The arranged design space for execution time parameter-----	29
Figure 4	Graphical representation of the algorithm for area/power for searching a greater value in the design space -----	35
Figure 5	Graphical representation of the algorithm for execution time for searching a greater value in the design space -----	35
Figure 6	Graphical representation of the algorithm for area/power for searching a lesser value in the design space-----	35
Figure 7	Graphical representation of the algorithm for execution time for searching a lesser value in the design space -----	35
Figure 8	The flow for the steps required to obtain the optimal variant of architecture using the proposed Hybrid DSE -----	39
Figure 9	Design space with all possible resource combinations-----	45
Figure 10	Arranged Vector Design Space for area -----	48
Figure 11	Arranged Vector Design Space for execution time-----	50
Figure 12	Overview of the proposed heuristic approach -----	55
Figure 13	Details of the proposed heuristic Exploration approach -----	57
Figure 14	ASAP scheduling of DWT benchmark -----	58
Figure 15	Scheduling after 1 st Iteration-----	59
Figure 16	Scheduling solution after 11th Iteration -----	60
Figure 17	MSGGA design space exploration approach overview -----	64

Figure 18	Flow chart of the proposed MSGA -----	65
Figure 19	The perturbation algorithm for resource -----	67
Figure 20	DFG of the IIR Digital Filter-----	68
Figure 21	Scheduling of the IIR Digital Filter using ASAP-----	70
Figure 22	Chromosome Encoding for the first parent (P1) -----	70
Figure 23	Chromosome Encoding for the second parent (P2) -----	70
Figure 24	Chromosome Encoding for the third parent (P3) -----	71
Figure 25	Offspring 1 obtained for the nodal string obtained after crossover between P1 and P2-----	72
Figure 26	Offspring 2 obtained for the nodal string obtained after crossover between P1 and P2-----	72
Figure 27	Offspring 1 and Offspring 2-----	73
Figure 28	Mutation algorithm for the proposed approach -----	74
Figure 29	Proposed load-factor heuristic for the MSGA Framework -----	76
Figure 30	Integrated solution to offspring 1 (Decoding of the chromosome) for the IIR Digital Filter benchmark-----	77
Figure 31	Cycle time calculation during data pipelining for offspring 1-----	80
Figure 32	Design Flow of the proposed Exploration Synthesizer platform -----	84
Figure 33	DFG of the DWT Benchmark-----	88
Figure 34	Application library: The intermediate representation of DWT DFG which acts as the input format for the tool-----	88

Nomenclature

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

N_{RM}	Number of memory elements present (such as registers)
C_R	Total cost of all resources
C_{Ri}	Cost per area unit of the resource (such as adders, multipliers)
C_{Rclk}	Cost per area unit of the clock oscillator
C_{RM}	Cost per area unit of memory element
W	Total workload of all the resources
x	position of the variant in the set
τ	approximated membership value of the variant which is the x^{th} element in the sorted arrangement
α	order of the first element
β	order of the last element
τ_B	membership value of the border variant for the parameter in the space.
τ_V	membership value for the variant under test
$V_{Variant}$	Respective value of variant under test for a parameter
τ_{Min}	membership values of the minimum variants in the architecture space
τ_{Max}	membership values of the maximum variants in the architecture space
V_{Borde}	actual border variant value
τ_{ini}	calculated initial membership value
τ_{max}	Maximum membership value (equals to '1') corresponding to the fuzzy logic membership value rule
τ_{min}	Minimum membership value (equals to '0') corresponding to the fuzzy logic membership value rule
G	Power Gradient
$O(i)$	Operation of a schedule
CS	Control step of a schedule

PI	Priority Indicator
P_T	Power consumption due to dissipation of leakage current
ps	power dissipated per area unit when the transistors in the chip are not switching
L^{ASAP}	ASAP scheduling with maximum resource
load factor (α)	Load factor or workload of each resource
β	Encoded value of each operation (oi) of the second parent chromosome
μ	Random value equal or between ' α ' and ' β '
W1, W2	Tuning factor/Weightage factor
C_L	Local cost function
P_{cross}	Crossover probability
vn	Version of the each type of resources used
C_G	Global cost function
A_{FU}	Total area of the functional units.
T_{CONS}	Execution time constraint specified by user during genetic algorithm
T_{MAX}	Max execution time taken by a solution during the specific generation
A_{FU}	Total area of the functional units
A_{REG}	Total area of registers
A_{MUX}	Total area of registers
A_{DEMUX}	Total area of the demultiplexers used during implementation
P_{CONS}	Power consumption constraint specified by the user
P_{MAX}	Max power consumption of a solution during a specific generation
G(Max)	Maximum generation of an algorithm

Chapter 1

Introduction

1.1 Overview

The never ending increase in the growth of chip complexity has only been possible due to efficient scheduling and exploration techniques. The growth in chip capacity has enabled processing of huge amounts of data with greater flexibility and less expense. This requirement to yield high performance with a concurrent balance in power expenditure is often a primary specification in the area of Digital Signal Processing (DSP), communications and network processing. For e.g., Application Specific Processor cores are increasingly being used to simultaneously address the need for high performance, low area, minimum cost and timely operation in many embedded systems. Particularly elements used in mobile phones, such as the DSP cores, must be low cost and consume less power than their general purpose counterparts. Hence, exploration of an optimized solution that has the capability to encounter conflicting conditions such as minimizing the speed of the exploration process and maximizing the quality of the scheduling solution by limiting power expenditure at minimal control step (time) usage is extremely significant for the development of computation intensive DSP cores [1][2][3].

Moreover, the complicated process of exploration of the final scheduling solution also requires a tradeoff between the contradictory parameters of power and latency/performance in addition to the contradictory demands [1][3].

A DSE problem therefore considers two orthogonal issues: (a) how can a single design point be evaluated? (b) How can the design space be covered during the exploration process? The latter issue arises since an exhaustive exploration of the design space, including evaluating every possible design point, is usually prohibitive due to the sheer size of the design space. Therefore, trade-offs linked to the choice of appropriate evaluation and coverage methods are discussed. The designer must balance: the accuracy of the evaluation, the time it takes to evaluate one design point (including the implementation of the evaluation model), the precision/granularity of the design space coverage, as well as the possibilities for automating the exploration process.

Multi-objective algorithms could use combined objectives in order to reduce the number of dimensions to the problem. For example, it could make sense to only consider the speed-cost and the flexibility-cost ratios for a certain design and not speed, cost, and flexibility as separate optimization goals. One of the most prevalent combined objectives is energy-delay product. The energy-delay product is used to assess embedded systems. The delay-power ratio objective can be interpreted as a computational clock cycles versus power dissipated. The combined speed-cost ratio objective represents a computational complexity related to the costs of the design. It should be noted that there are also optimizer-specific metrics that guide the search, such as the steepness to surrounding solutions in the case of hill climbing or the number of dominated solutions in the case of some multi-objective, evolutionary algorithms. Hill climbing, for instance, evaluates the neighborhood of the current design to determine the next steepest step towards the optimization

goal. In order to avoid being trapped on top of a local maximum, hill climbing requires backtracking mechanisms which might be expensive in “bumpy terrains”. Moreover, the search becomes aimless on plains and is not able to recognize diagonal ridges since the probe directions would always lead to lower quality solutions [1][2][4]. The proposed method has a tendency to yield high quality solutions (which obeys multi parametric optimization requirement) due to its unique algorithmic and framework features. It also manages to balance the tradeoff proficiently between exploring a high quality solution and the runtime taken.

1.2 Fundamentals on Modular System Design

The design and development of modular systems with heterogeneous performance optimization objective requires extensive analysis and assessment of the design space, not only due to the assorted nature of the parameters, but also due to the diversity in architecture for implementation. Given the specifications and the system requirements the aim of designers is to reduce the large and complex design space into a set of feasible design solutions meeting performance objectives and functionality. For most modular systems and systems based on strict operational constraints the selection of the optimal architecture for system design is the most important step in the development process. Design space architecture can have innumerable design options for selection and implementation based on the parameters of optimization. Hence selection of the optimal architecture from the design space which satisfies all the performance objectives is crucial for the present generation of System-on-chip (SoC) designs [5, 6]. As it is always possible to implement different functions of a system on different hardware components, the architecture design space has become more complex to analyze. In the case of high level synthesis, performing design space exploration to choose the best candidate architecture by

concurrently satisfying many operating constraints and optimization parameters is considered the most important stage in the whole design flow (details on design space exploration is provided later). The method for exploration of the best candidate architecture should not only be less in terms of complexity factor and time but should also explore the variant in an efficient way meeting all the specifications provided. The process of high level synthesis design is very complicated and descriptive and is usually performed by system architects. Depending on the application, the process of defining the problem, performing design space exploration and the other steps required for its successful accomplishment are very time consuming. Furthermore, recent advancements in areas of communications and multimedia have led to the growth of a wide array of applications requiring huge data processing at minimal power expense. Such data hungry applications demand satisfactory performance with power efficient hardware solutions. Since the selection process for the best design architecture is complex, an efficient approach to explore the design space for selecting the best design option is needed [1,2,3,4].

1.3 Related Works

An engineering problem can generally be described as a phenomenon of analyzing and managing the tradeoffs between contradictory design objectives. The problem of obtaining a comprehensive Pareto optimal set [42-46] has been addressed by few researchers. In [7] the researchers proposed an approach for synthesis of heterogeneous embedded systems by using Pareto Front Arithmetic (PFA) to explore the giant search spaces. Their method utilized the hierarchical problem structure for exploring the set of Pareto optimal solutions. Similar problem was also addressed in [8] by suggesting order of efficiency, which assists in deciding preferences amongst the different Pareto optimal points. Work in [9] suggested the identification of a few

superior design points from the Pareto set is enough for an excellent design process. In [10] evolutionary algorithms such as the Genetic Algorithm (GA) had been suggested to yield better results for the design space exploration process. The use of GA had also been suggested as a framework for DSE of data paths in high level synthesis in [11]. Another approach was introduced by researchers in [12] which were based on Pareto optimal analysis. According to their work, the design space was arranged in the form of an Architecture Configuration Graph (ACG) for architecture variant analysis and optimization of performance parameters. Their results proved quite promising for architectural synthesis of digital systems. Furthermore in [13] and [14], authors described another approach for DSE in high level systems based on binary encoding of the chromosomes. Work shown in [15] for DSE suggests that authors used an evolutionary algorithm for successful evaluation of the design for an application specific SoC. The work shown in [16] discusses the optimization of area, delay and power in behavioral synthesis, but does not focus on the high level design flow with multi parametric optimization objective. Authors in [17] introduce a tool called SystemCoDesigner that offers rapid design space exploration with rapid prototyping of behavioral systemC models. In [17] an automated integration was done by integrating behavioral synthesis into the proposed design flow. Authors in [18] have proposed a power optimization in SoC data flow systems. Although the proposed optimization yielded significant results, the focus of their work was not on control time constrained scheduling but rather power optimization hardware during exploration. Authors in [19] describe current state-of-the-art high-level synthesis techniques for dynamically reconfigurable systems. In addition to above, authors in [20] have applied GA to the binding and allocation phase. The authors have introduced an unconventional crossover technique depending on a force directed data path binding completion algorithm. One of the key features of their

approach is the use of multiport memories. Its main drawback is that it accepts as input the scheduled data flow graph, thus is unable to handle the scheduling problem. Authors in [21] presented a time constrained scheduling based on the GA. A list decoder is used to decode chromosome encoding by permutation of operations, into a valid schedule. Although the method is promising, it is slow compared to the other GA approaches. In addition, authors in [22] have proposed a problem space genetic algorithm for design space exploration of data paths. They have used the concept of heuristic/problem pair to convert a data flow graph [23] into a valid schedule. Another class of scheduling algorithms presented previously includes constructive approaches such as As Soon As Possible (ASAP) [5], As Late As Possible (ALAP) [23], list scheduling [24] and Force Directed scheduling [25]. These approaches are very simple and fast in nature. These algorithms all suffer from inherent tendency to optimize one parameter at the expense of other. Moreover, non-consideration of multiple user objective and implementation runtime dilutes its ability to be used in the fore front of modern performance driven designing process. Additionally, the tradeoffs performed using above methods which tends to engulf high exploration/optimization runtime. Moreover, the execution time parameter is not taken into account during exploration where needed but only delay. For the modern generation of hardware systems, deficiency of pipelining provision (by considering only delay) is extremely fatal for efficiency enhancement. Hence, the thesis eliminates the deadlock associated with these techniques. Also in many cases, the optimization factor and performance goal of the user may completely change depending upon his design requirement. For example, besides the multiple user criteria viz. (a) Accelerated power-performance tradeoff with area/cost as minimization criteria (b) Accelerated Area-performance tradeoff with power as minimization criteria, there can be a third type of optimization goal viz. (c) static power optimization under minimum control

step usage. Thus, besides being able to tradeoff based on requirement a) and b), an efficient novel optimization methodology must also be available that can address (c) which is equally significant. But unfortunately, to the best of the authors' knowledge, all the approaches so far are deficient in addressing (c). This dissertation develops a novel technique for addressing deficiency in (c).

1.4 Background Information

1.4.1 Theoretical Background on High Level Synthesis

Interdependent tasks such as scheduling, allocation and module selection are important ingredients of the high level synthesis design process. High level synthesis is a methodology of transforming an algorithmic behavioral description into an actual Register Transfer Level (RTL) structure. Therefore high level synthesis methodology contains a sequence of tasks to convert the abstract behavioral description of the algorithm into its respective structural block at RT level. The design at the RT level comprises of functional units such as Arithmetic Logic Unit (ALU), storage elements, registers, busses and interconnections. The algorithmic description specifies the inputs and outputs of the behavior of the algorithm in terms of operations to be performed and data flow. A description of the algorithm is usually represented in the form of an acyclic directed graph known as a sequencing graph. These graphs specify the input/output relation of the algorithm and the data dependency present in the data flow. The graph is defined in terms of its vertices and edges, where the vertices signify the operations and the edges indicate the data dependency present in the function. High level synthesis is therefore a conversion from the abstract behavioral description to its respective hardware description in the form of Arithmetic Logic Units (ALU), memory elements, storage units, multiplexers/demultiplexers and the

necessary interconnections. The transformed algorithm at the RT level is comprised of a control unit and the data path unit. High level synthesis offers many advantages, such as productivity gains and efficient design space exploration [51-60]. Performing DSE at a higher level of abstraction provides more dividend than at lower levels of abstraction, i.e. transistor level or logic level. Traditional high level synthesis design methodology is much simpler than modern design techniques. In general, the initial step of synthesis is to compile the behavioral specification into an internal representation. The next step is to apply high level transformation techniques with the aim of optimizing the behavior as per the desired performance. In order to realize the structure, the final step is to perform scheduling to determine the time at which each operation is executed and the allocation, which is synthesizing the necessary hardware to perform the operations [5].

Scheduling can be of two different classes: time constrained scheduling and resource constrained scheduling. Time constrained scheduling refers to finding the minimum cost schedule that satisfies the given set of constraints with the given maximum number of control steps. Resource constraint scheduling, on the other hand, refers to finding the fastest possible schedule that satisfies the given set of constraints with the given maximum number of resources. Resource constraints are generally specified by the area occupied by the functional units like adders/subtractors, multipliers, dividers and ALUs. Although the data path of the system consists of registers and interconnections, they are not considered to be included as resource constrained because they are difficult to specify. High level synthesis can be broadly divided into the following steps: input description, internal representation, design space exploration, allocation, scheduling and binding. Therefore the final structure at the RT level consists of the data path and the control path. Traditional high level synthesis design flow falls short for the modern

generation of complex VLSI and SoC designs, because the conventional design flow just takes into account the optimization of two parameters, namely area and latency. But the new generation of system designs requires multi parametric optimization strategies in HLS while simultaneously utilizing rapid and efficient DSE approaches for finding the best suitable architecture [5].

1.4.2 Theoretical Background on Design Space Exploration

For the present generation of Very large Scale Integration (VLSI) designs with multi objective nature, the cost of solving the problem of scheduling, allocation and module selection discretely or simultaneously by exhaustive analysis is strictly prohibitive. Multi objective VLSI designs are used in low end Application Specific Integrated Circuits (ASICs) with low power dissipation and acceptable performance, as well as in high end ASICs with high performance requirements and satisfactory power expenditure. Hence, efficient design space exploration techniques are needed that not only satisfy the above requirements but also make efficient use of runtime, due to time to market pressure [6]. Design space exploration [51-60] is a procedure for analyzing the various design architectural alternatives in the design space to obtain the optimum architecture needed for the behavioral description based on the predefined user specifications. Design space exploration has always been a challenge for researchers due to the heterogeneity of the objectives and parameters involved. The current trend towards design space exploration has been the reduction of the design space into a set of Pareto optimal points [42-46] by Pareto optimal analysis. Sometimes even the Pareto optimal set can be very large for analysis and selection of the design for system implementation. In order to assist the decision maker in exploring the design space better, an accurate and fast approach efficient in terms of time

expended and quality of solution found is very significant for high level synthesis design of hardware systems.

1.4.3 Overview on the Abstraction Level of Optimization

Today's electronic systems are designed starting from specifications given at a very high level of abstraction. This is because many Electronic Design Automation (EDA) tools accept a design expressed in a high-level format as input and can automatically produce the corresponding RT/Logic/transistor-level implementation with very limited human intervention. All hardware systems can be classified into various levels of abstraction such as System level, Architecture level, Register Transfer Level (RTL), Layout level and Transistor level. This abstraction level also provides an insight into the hierarchy that a system can be classified into. Optimization performed at the higher levels of abstraction provides more flexibility, productivity and design specification awareness than performing only at the lower levels of abstraction. Moreover, although effective, performing optimization only at the transistor level is not sufficient for the current generation of high performance, power hungry application specific systems (used in embedded applications) due to the enormous complexity involved. The traditional method of optimization performed by circuit designers only at low level for area and latency is insufficient for current power and performance requirements. Therefore, the role of system architects has become extremely crucial. System architects consider user goals during the architecture selection process by performing optimization of the given application based on high level parametric models. The design process must consider user goals even at the very high abstraction level (during high level synthesis process) in order to generate a quality aware solution (at the RT-Level) with greater possibility of optimization at the transistor level.

1.4.4 Reasons for Studying High Level Synthesis

There has been a trend towards automating synthesis at higher levels of the design hierarchy in the recent years. Logic synthesis has gained acceptance in industry long back and currently there has been substantial interest shown in Register Transfer Level (RTL) design obtained from higher levels of abstraction (algorithmic). The reasons are the following [27]:

Reduced design time and high acceleration: If more of the design process is automated, a company can complete a design faster, and thus have a better chance of hitting the market window for that design.

Design is specification aware from the very earliest stage: Design space exploration to perform multi-objective optimization and tradeoff is needed from the very earliest stage of designing. This will enable the designers to start the development with an architecture that is already specification aware (high level optimized) from the highest level of abstraction thus rendering more chances that final design (logic/layout) corresponds to the given constraints.

The ability to search the design space (and design alternatives): A good synthesis system can produce several designs from the same specification in a reasonable amount of time. However, final selection can be challenging with many choices. Therefore, an efficient exploration method is needed to tackle the problem from the very high abstraction level to assure the designer a greater chance of optimization and flexibility to control architecture based on user requirements. This allows the developer to explore different tradeoffs between cost, speed, power etc. or to take an existing design and produce a functionally equivalent one that is more efficient.

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

1.5 Summary of Contribution

The proposed exploration approach can be used during the design process of application specific processors/custom processors or standalone Application Specific Integrated Circuits (ASIC's) custom data intensive applications. Therefore, systems that include adaptable applications which change dynamically during runtime should not be considered with these approaches.

This dissertation contributes to the following by removing bottlenecks in previous approaches:

- Solving the Problem of Design Space Exploration for Power-Performance-Cost/Area tradeoff in High Level Synthesis using novel Priority Factor approach:

(Note: Publications: S1, S2, S7, S8, S14, S15, S19, S20, S24, S22, S27 on Page: 111)

- a) Introduces /Derives mathematical model for modern parametric framework viz. performance (execution time) for Design Space Exploration.
 - b) Introduces/Derives mathematical model for modern performance parametric framework viz. Hardware Cost for Design Space Exploration.
 - c) Presents mathematical model for modern performance parametric framework viz. Power for Design Space Exploration
 - d) Proposes a new technique using Priority Factor Metric and Vector Design Space scheme for arranging the architecture design space.
 - d) Provides significant improvements in exploration speed compared to a recent technique for various signal processing benchmarks.
- Solving the Problem of Design Space Exploration for Hardware Area/power-Performance-power/area tradeoff in High Level Synthesis using Self-Correction Scheme based Hybrid Fuzzy approach:

(Note: Publications: S3, S4, S9, S10, S13, S16, S23 on Page: 111)

- a) Proposes Hybrid Fuzzy scheme based frameworks for all cases of exploration for Area/Power and Performance parameters.
 - b) Development of Fuzzy sets for representation of architecture design variances.
 - c) Algorithms with Self-Correction Scheme for exploring the final design point.
 - d) This hybrid technique provides an average improvements of greater than 22 % in exploration process compared to a recent technique for various size benchmarks.
- Solving the Problem of Integrated Exploration of Scheduling and Module Allocation in High Level Synthesis for static power optimization under minimum control step usage:
(Note: Publications: S5, S11, S18, S21 on Page: 111)
 - a) Proposes a mathematical expression for power gradient based on the power dissipation of the resources used during determination of high priority nodes.
 - b) Presents a new priority function called ‘Priority indicator (PI)’ based on selection criterion that takes into account the power gradient. This new iterative exploration approach method is used for exploring the optimal/sub optimal integrated solution to the problem of scheduling and module selection.
 - c) Provides a completely automated design space exploration tool for rapid exploration of scheduling and module selection in high level synthesis design process.
 - d) The proposed approach successfully improves the quality of final solution by an average of 5.07 % and reduces the exploration runtime by an average of 59% compared to a current approach for standard DSP Benchmarks.
 - Solving the Problem of Integrated Exploration of Scheduling and Module Allocation in High Level Synthesis for Power-Performance tradeoff using Heuristic Genetic Algorithm:
(Note: Publications: S6, S12, S17, S26 on Page: 111)

- a) Multi Structure Genetic Algorithm is based on a novel cost function based on the power consumption-execution time tradeoff.
 - b) The total execution time constraint considered in the cost function of the proposed approach is based on latency, cycle time and number of data (N) to be pipelined.
 - c) Multi structure genetic algorithm is based on a new structural topology where each functional unit type is represented by an independent chromosome.
 - d) Since the multi structure genetic algorithm incorporates a new seeding process with two special chromosomes, hence the final solution found is always certain to be global optimal or local optimal (in certain cases) in terms of the execution time (including latency and cycle time) and power.
 - e) The results produced by proposed approach are better compared to another genetic algorithm based approach, for almost all digital signal processing benchmarks.
- Introducing a design Automation Platform (DAP) in high level synthesis for multi-objective optimization and RTL circuit generation capable of:
 - (Note: Publications: S1, S2, S3, S4 on Page: 111)
 - a) Power-Performance Tradeoff using Area as Optimization Criteria.
 - b) Hardware Area- performance Tradeoff using Power as Optimization Criteria.

1.6 Dissertation Organization

The rest of the dissertation is organized as follows: Chapter 2 describes in details the proposed techniques behind solving the problem of Design Space Exploration for power-performance-area/cost tradeoff using priority factor and vector design space technique. Chapter 3

elaborates on proposing the solution for solving the problem of design space exploration for hardware area-performance-power tradeoff using fuzzy membership based algorithm and priority factor framework. Chapter 4 proposes the approach for solving the problem of integrated exploration of scheduling and module allocation for static power optimization under minimum control step usage based on power gradient theory, while in Chapter 5, the approach for solving the problem of integrated Exploration of scheduling and module allocation for power-performance tradeoff using multi structure genetic algorithm is proposed. Chapter 6, introduces a high level synthesis DAP for multi-objective optimization and RTL circuit generation capable of performing multi objective tradeoff. The results of the proposed DSE approaches for various well known high level synthesis benchmarks indicating exploration time and quality improvements obtained when compared to the current existing DSE approach are provided in Chapter 7. Chapter 8 is dedicated to conclusion and future scope of work in this area. The list of publications related to this field of research study and the total list of citations are also provided thereafter.

Chapter 2

Rapid Design Space Exploration in High Level Synthesis Based on Power-Performance Tradeoff using Priority Factor Metric

This chapter introduces the first algorithm of the dissertation based on priority factor metric which deals with proposing a solution to the design space exploration in high level synthesis for computation intensive applications. It is used for performing tradeoff based on power-performance constraint and area/cost as optimization criteria. The proposed approach is deterministic in nature and therefore finds the final architecture based on resolute evaluation steps (unlike heuristic methods). It is important to note that the proposed exploration approach is only applied to custom data intensive applications (application specific processors/custom processors) or standalone Application Specific Integrated Circuits (ASIC's). The priority factor metrics proposed in this chapter are based on mathematical models for modern parametric frameworks viz. power, performance area/cost. The approach also employs a special topology called vector design space based on priority order sequencing for sorted arrangement of the design space. The mathematical framework for each parameter is described and deduced below:

2.1 Mathematical Derivation for Cost/Area Model

A. The Proposed Framework for Hardware Cost

Let the area of the resources be given as 'A'. R_i denotes the resources available for system designing; where $1 < i < n$. 'n' represents the maximum resource available for designing. 'Rclk' refers to the clock oscillator used as a resource providing the necessary clock frequency to the system (Note: to simplify the mathematical modeling of area, the existence of multiple clocks operating in a single system has been ignored. For high level area modeling, only the global clock operating in a system has been considered. However, the user has the flexibility to declare in the module library various clock frequency oscillators available for selection based on the exploration result). The total area can be represented as the sum of all the resources used for designing the system, such as adder, multiplier, divider, clock frequency oscillator and the memory elements. At the high level all elaborate lower level details such as routing information (wire connection etc.) are not available. Thus they have been ignored in eqn. (1) for high level area estimation. As described in [5, 41], the total area of a system mainly consists of the areas of the functional blocks; the total area can be approximated as follows:

$$A = \sum A(R_i) \quad (1)$$

$$A = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + \dots + N_{Rn} \cdot K_{Rn}) + A(R_{clk}) + N_{RM} \cdot K_{RM} \quad (2)$$

Where ' N_{Ri} ' represents the number of resource ' R_i ', ' K_{Ri} ' represents the area occupied per unit resource ' R_i ', ' N_{RM} ' represents the number of memory elements present (such as registers) and ' K_{RM} ' represents the area occupied by each memory element. Let the total cost of all resources in the system is ' C_R '. Further, cost per area unit (in fiscal units) of the resource (such as adders, multipliers etc) is given as ' C_{Ri} ', the cost per area unit (in fiscal units) of the clock oscillator is

‘ C_{Rclk} ’ and finally the cost per area unit (in fiscal units) of memory element is ‘ C_{RM} ’. Therefore total cost of the resources in fiscal units is given as:

$$C_R = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + \dots + N_{Rn} \cdot K_{Rn}) \cdot C_{Ri} + A(R_{clk}) \cdot C_{Rclk} + N_{RM} \cdot K_{RM} \cdot C_{RM} \quad (3)$$

Applying partial derivative to equation (3) with respect to $N_{R1} \dots N_{Rn}$, with respect to N_{RM} and with respect to A_{Rclk} yields equation (4) to (7) respectively as shown below:

$$\frac{\partial C_R}{\partial N_{R1}} = \frac{\partial [(N_{R1} \cdot K_{R1} \cdot C_{R1} + \dots + N_{Rn} \cdot K_{Rn} \cdot C_{Rn}) + A(R_{clk}) \cdot C_{Rclk} + N_{RM} \cdot K_{RM} \cdot C_{RM}]}{\partial N_{R1}} = K_{R1} \cdot C_{R1} \quad (4)$$

$$\frac{\partial C_R}{\partial N_{Rn}} = \frac{\partial [(N_{R1} \cdot K_{R1} \cdot C_{R1} + \dots + N_{Rn} \cdot K_{Rn} \cdot C_{Rn}) + A(R_{clk}) \cdot C_{Rclk} + N_{RM} \cdot K_{RM} \cdot C_{RM}]}{\partial N_{Rn}} = K_{Rn} \cdot C_{Rn} \quad (5)$$

$$\frac{\partial C_R}{\partial A_{Rclk}} = C_{Rclk} \quad (6)$$

$$\frac{\partial C_R}{\partial N_{RM}} = K_{RM} \cdot C_{RM} \quad (7)$$

For the sake of simplicity, while applying partial derivative to a certain resource (equations (4) – (7)), the others resources are assumed fixed (or constant). In order to determine the contribution of a specific resource on the change in a parameter, other resources have to be kept fixed (constant). For example, in equation (7), while applying partial derivative with respect to N_{RM} , the change in number of resources $N_{R1} \dots N_{Rn}$ is assumed to be fixed. Without keeping the other resources $N_{R1} \dots N_{Rn}$ fixed during analysis, the impact of resource N_{RM} in the deviation of cost parameter cannot be determined. Now using the theory of approximation by differentials, the change in the total cost can be approximated by the following equation:

$$dC_R = \frac{\partial C_R}{\partial N_{R1}} \cdot \Delta N_{R1} + \frac{\partial C_R}{\partial N_{Rn}} \cdot \Delta N_{Rn} + \frac{\partial C_R}{\partial N_{RM}} \cdot \Delta N_{RM} + \frac{\partial C_R}{\partial A_{Rclk}} \cdot \Delta A_{Rclk} \quad (8)$$

Additionally, equation (8) above indicates the total change in C_R (total cost of resources) with

$$dC_R = \underbrace{\Delta N_{Rn} \cdot K_{Rn} \cdot C_{Ri}}_{\substack{\text{The change of cost} \\ \text{contributed by} \\ \text{resource Rn}}} + \underbrace{\Delta N_{RM} \cdot K_{Rn} \cdot C_{RM}}_{\substack{\text{The change of cost} \\ \text{contributed by} \\ \text{memory RM}}} + \underbrace{\Delta A(R_{clk}) \cdot C_{Rclk}}_{\substack{\text{The change of cost} \\ \text{contributed by} \\ \text{resource clock}}} \quad (9)$$

respect to change in number of resource $N_{R1} \dots N_{Rn}$, N_{RM} and R_{clk} . Substituting equations (4) to (7) into equation (8) yields equation (9) shown above:

Equation (9) represents the change in total cost of resources with the change in the number of all resources and the clock period (clock frequency). The Priority Factor (PF) for cost of resources is defined as:

$$PF(R1) = \frac{\Delta N_{R1} \cdot K_{R1} \cdot C_{Ri}}{N_{R1}} \quad (10)$$

.....

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn} \cdot C_{Ri}}{N_{Rn}} \quad (11)$$

$$PF(RM) = \frac{\Delta N_{RM} \cdot K_{RM} \cdot C_{RM}}{N_{RM}} \quad (12)$$

$$PF(Rclk) = \frac{\Delta A(Rclk) \cdot C_{Rclk}}{N_{Rclk}} \quad (13)$$

PF yields a real number, which suggests the extent to which the change in number of a particular resource contributes to the change in hardware cost. The PF is a determining factor which helps us to judge the influence of a particular resource on the variation of the optimization parameters like area, time of execution and power consumption. This PF is used later in our approach to organize the architecture design space consisting of variants in increasing or decreasing order of magnitude. The equation (10) and (11) indicates the change of cost with respect to change in resource $R1, \dots, Rn$. Similarly, equation (12) indicates the change of cost of the system with respect to change in number of resource 'RM'. Further equation (13) indicates

the change of cost of the system with respect to change in resource 'Rclk'.

2.2 Mathematical Derivation for Execution Time Model

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

Based on [5, 61,62,12], the time of execution can be represented by the following equation:

$$T_{exe} = [L + (D - 1) \cdot T_c] \quad (14)$$

where 'L' represents latency of execution, ' T_c ' represents the cycle time of execution, 'D' denotes the number of data elements to be processed. Equation (14) indicates the time needed to data pipeline an application based on data dependency and available functional units. The equation also captures any situation through initiation interval where an operation is not available for pipelining due to data hazard. It is important to mention the difference between data pipelining/data level parallelism and instruction level parallelism: In the former, there are no service operations such as instruction fetch, instruction decode, data fetch and write back. Only the execution stage can directly processes the input data and produce the output for the next operation. Therefore, no latch requirements are necessary during data pipelining. In contrast, instruction level pipelining includes service operations as well as execution stage and pipelining is effectively between the hardware units. Therefore, latches are necessary to store the information of the previous unit and to pass it to the next stage. (An example of data pipelining for a sample application is demonstrated in Chapter 6).

The term 'workload' of a resource signifies the time required (or clock cycles needed) to finish its assigned operation during scheduling. Hence the total workload (W) of all resources to finish

their respective operations during scheduling for ‘D’ sets of processing data can be represented by (15):

$$W = (N_{R_1} \cdot T_{R_1} + N_{R_2} \cdot T_{R_2} + \dots + N_{R_n} \cdot T_{R_n}) \cdot D \cdot T_p \quad (15)$$

Where N_{R_i} represents the number of resource ‘Ri’ and ‘ T_{R_i} ’ represents the number of clock cycles needed by resource ‘Ri’ ($1 \leq i \leq n$) to finish each operation. ‘D’ is the number of sets of data elements that must be processed. Note: In the mathematical modelling process in equation (15), the operations have been considered to operate in a sequential manner for theoretical assumption and simplicity purposes. In the experimental exploration process described later, actual data parallelism has been considered while evaluating a particular solution. Therefore if variable N_{R_i} in equation (15) is increased, then ‘W’ will increase. In equation (16) and (17), objective is to evaluate the deviation of workload (W) with respect to change in variable N_{R_i} . For example, the average deviation of ‘W’ with respect to change in N_{R_i} from 1 adder/subtractor to 3 adder/subtractors can be evaluated. Therefore, the motive is to determine the contribution of each resource type on a specific parameter (as explained in Section 3.2). The demonstration of this approach is described later in this chapter.

From the approximation of differentials the change in ‘workload’ is approximated in (16).

$$dW = D \cdot \left[\left(\frac{\partial W}{\partial N_{R_1}} \cdot \Delta N_{R_1} + \frac{\partial W}{\partial N_{R_2}} \cdot \Delta N_{R_2} + \dots + \frac{\partial W}{\partial N_{R_n}} \cdot \Delta N_{R_n} \right) + \Delta T_p \cdot \frac{\partial W}{\partial T_p} \right] \quad (16)$$

Applying partial derivative to the (15) with respect to N_{R_1}, \dots, N_{R_n} and T_p will produce the following set of equations:

$$\begin{aligned} \frac{\partial W}{\partial N_{R_1}} &= \frac{\partial [(N_{R_1} \cdot T_{R_1} + N_{R_2} \cdot T_{R_2} + \dots + N_{R_n} \cdot T_{R_n}) \cdot T_p \cdot D]}{\partial N_{R_1}} \\ &= T_{R_1} T_p \cdot D \end{aligned} \quad (17)$$

$$\frac{\partial W}{\partial N_{Rn}} = \frac{\partial[(N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + \dots + N_{Rn} \cdot T_{Rn}) \cdot T_p \cdot D]}{\partial N_{Rn}}$$

$$= T_{Rn} T_p \cdot D \quad (18)$$

$$\frac{\partial W}{\partial T_p} = \frac{\partial[(N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + \dots + N_{Rn} \cdot T_{Rn}) \cdot T_p \cdot D]}{\partial T_p} \quad (19)$$

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

Substituting equations (17), (18) and (19) in equation (16) yields equation (21):

$$dW = \Delta N_{R1} \cdot T_{R1} \cdot T_p \cdot D + \Delta N_{R2} \cdot T_{R2} \cdot T_p \cdot D + \dots + \Delta N_{Rn} \cdot T_{Rn} \cdot T_p \cdot D + D \cdot \Delta T_p \cdot (T_{R1} \cdot N_{R1} + T_{R2} \cdot N_{R2} + \dots + T_{Rn} \cdot N_{Rn}) \quad (21)$$

Equation (21) reflects the change in total workload with the change in number of all the resources and the clock period (clock frequency).

$\Delta N_{Rn} \cdot T_{Rn} \cdot T_p \cdot D =$ The change of 'W' contributed by the change in number of resource Rn.

$\Delta T_p \cdot (T_{R1} \cdot N_{R1} + T_{R2} \cdot N_{R2} + \dots + T_{Rn} \cdot N_{Rn}) \cdot D =$ The change of 'W' contributed by the change in clock period (frequency).

Considering constraint on the number of resources, the increase in total workload (W) will cause an increase in total execution time. Therefore, the more the workload increases, the more the execution time increases under resource constraints. Hence, the change in number of a resource (e.g. change in adder from one to three) that contributes to the change in total workload the most, also contributes to the change in total execution time the most. So based on above analysis, PF for execution time parameter is defined as:

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

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + \dots + N_{Rn} \cdot T_{Rn}}{N_{Rclk}} \cdot (\Delta T_p) \quad (23)$$

‘D’ is ignored in the expression for PF because it does not contribute to the change in Priority Order (PO) sequence described later in the paper. The factors defined above reflect the average change in execution time (T_{exe}) with the change in number of a resource (change in adder from one to three) at maximum clock period. These factors also reflect the average change in execution time (T_{exe}) with the change in clock frequency. In the expression for PF in (23), minimum clock frequency is considered because at this frequency the clock period is the maximum. Hence, the change in number of a specific resource at maximum clock period will influence the change in execution time the most, compared to the change in execution time at other clock periods.

2.3 Mathematical Model for Power Consumption

Based on [5, 64, 12], dynamic power (P_D) of a system as a function of operating frequency and number of devices switching due to frequency of operation can be represented as:

$$P_D = \sum_{i=1}^n (N_{Ri} \cdot K_{Ri}) \cdot p_c \quad (24)$$

Where ‘n’ is number of functional resources, ‘ N_{Ri} ’ represents the number of resource R_i as mentioned earlier. ‘ K_{Ri} ’ represents the area occupied per unit resource R_i and ‘ p_c ’ denotes the power consumed per area unit resource at a particular frequency of operation. Equation (24) models the dependency of power on the activity rate (which in turn is based on frequency of operation) of the modules in the system. Theoretically if there is no activity in the circuit the dynamic component of the power will be zero. The leakage power has been ignored in this model.

Applying the using partial derivative method on equation (24) (as shown in Section 2.1 for cost parameter), the Priority Factor (PF) for power consumption is extracted as follows (PF) [27]:

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \cdot (p_c)^{\max} \quad (25)$$

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + \dots + N_{Rn} \cdot T_{Rn}}{N_{Rclk}} \cdot (\Delta p_c) \quad (26)$$

The priority factor defined in equation (25) indicates the change of total power consumption with the change in number of resources at maximum clock frequency. In contrast, the priority factor defined in equation (26) indicates the deviation in total power consumption with respect to the change in number of clock oscillators from minimum to maximum. The priority factor helps to arrange the architectural variants of the design space in increasing or decreasing order of magnitude depending on the objective parameter. The PF is arranged in such a way that the resource with the minimum PF is chosen first, gradually increasing and then ending at the resource with the highest priority factor. The above rule applies for all three parameters described in this chapter.

2.4 Proposed Method of Design Space Exploration of Architecture Based on Power-Performance tradeoff with area/cost as optimization criteria

The overview of the proposed method is shown in Figure 1.

A. System specifications

The case study of a selected benchmark has been provided for demonstration of the proposed method based on some real system specifications (as shown in Table 1). The value assumed for area of each functional unit (in CLB slices) is obtained through synthesis in commercial logic synthesis tools. The values assumed for the clock cycle consumed for each functional unit is obtained from the literature [14, 22, 23]. If the user has different types of the same functional unit (such as 8-bit adder, 16 bit adder etc.), then the area of each functional unit type is also

specified in this stage simultaneously with the number of each functional unit type available for choice (e.g. Both the area of 8-bit adder (15 au) and the area of 16 bit adder (30 au) can be specified in addition to information 6) and 7) provided in Table 1).

Note 1: The parameters have fixed values assigned/specified by the user in this stage. These parametric constraints are the demand of the user and the final solution must meet these high level constraints as well as the requirement of the optimization parameter. For example, in Table 1, the assumed value of the user constraints for power and time is 8W and 140 us respectively. This indicates that the final solution must meet these high level constraints while also being minimum in occupied area. Therefore during the exploration process, the design space will be pruned based on these requirements and the best possible solution will result (as will be demonstrated later).

Note 2: The method assumes fixed hardware units such as 3 adder/subtractors, 4 multipliers, and 2 clock oscillators as shown in Table 1 because this is the specified maximum available units affordable by the user for this custom application specific system design. In other words, every user has a maximum limit on the permissible fixed hardware units that can be afforded based on specific configuration needs. However, for demonstration in this dissertation, these values are arbitrarily assumed for a sample application and are subject to flexibility depending on the requirement of the user and problem (as will be demonstrated latter).

Note 3: In Table 1 there are two available choices for clock frequency oscillators assumed to be specified by the user. The clock oscillator available for selection through exploration process is the global clock frequency of the system. For high level architecture decisions only global clock frequency has been considered in the proposed exploration method (as will be demonstrated later in this chapter).

The function of the selected second order digital IIR Butterworth filter benchmark is given in eqn (27).

$$y(n) = 0.167 x(n) + 0.5x(n-1) + 0.5x(n-2) + 0.167 x(n-3) - 0.33 y(n-2) \quad (27)$$

B. Calculation of the priority factor for each available resource for execution time parameter

For resource adder/subtractor (R1), multiplier (R2), clock oscillator (R_{clk}):

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

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

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

$$= (3 \cdot 2 + 4 \cdot 4) \cdot (0.02 - 0.005) / 2 = 0.165$$

According to the above analysis the change in number of adder/subtractor affects the change

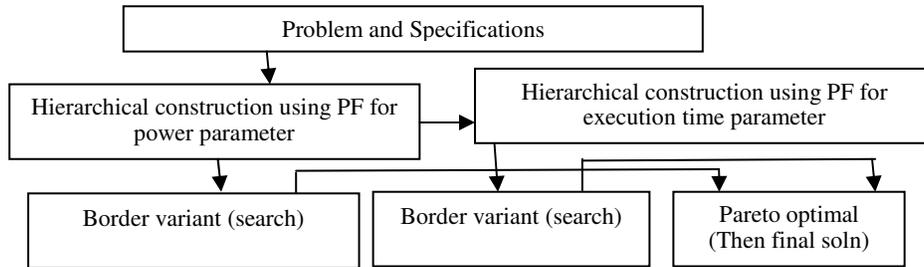


Figure.1. Generic Overview of the Proposed Exploration Method

Table1. System Specifications and Constraints for PF Method

1) Maximum power consumption: 8 watts (W)
2) Maximum time of execution: 140 μs (for D =1000 sets of data)
3) Hardware area of resources: Minimum
4) Maximum resources available for the system design: a) 3 Adder/subtractor units. b) 4 Multiplier units c) 2 clock frequency oscillators: 50 MHz and 200 MHz
5) No. of clock cycles needed for multiplier and adder/subtractor to finish each operation: 4 cc and 2cc
6) Area occupied by each adder/subtractor and multiplier: 20 area units (a.u) and 100a.u. (e.g. 20 CLBs on FPGA)
7) Area occupied by the 50MHz and 200MHz clock oscillator: 4 area units and 10 area units
8) Power consumed at 50 and 200MHz: 10mW/a.u. and 40mW/a.u.

in execution time the least, while the change in clock frequency from 50 MHz to 200 MHz affects the change in execution time the most. The minimum for adder/subtractor and multiplier

is one in above equations because the digital IIR Butterworth filter function at least requires one adder and one multiplier to successfully accomplish the functioning of the task.

C. Arrangement of resources in Priority Order based on calculation of PF for execution time

Based on the priority factors calculated, a new terminology called ‘Priority Order (PO)’ is defined. The priority order is a sequence ordering of the resource types ($R_1 \dots R_n$) based on reverse PF magnitude. In other words, the resource type with the lowest priority factor is assigned the highest priority order while the resource type with the highest priority factor is assigned the lowest priority order, i.e. the priority order of the resource increases with the decrease in priority factor of the resource.

For example, resource Rclk with the highest PF (see Section 2.4.B) has been assigned the lowest PO. On the other hand, resource R1 with the lowest PF (see Section 2.4.B) has been

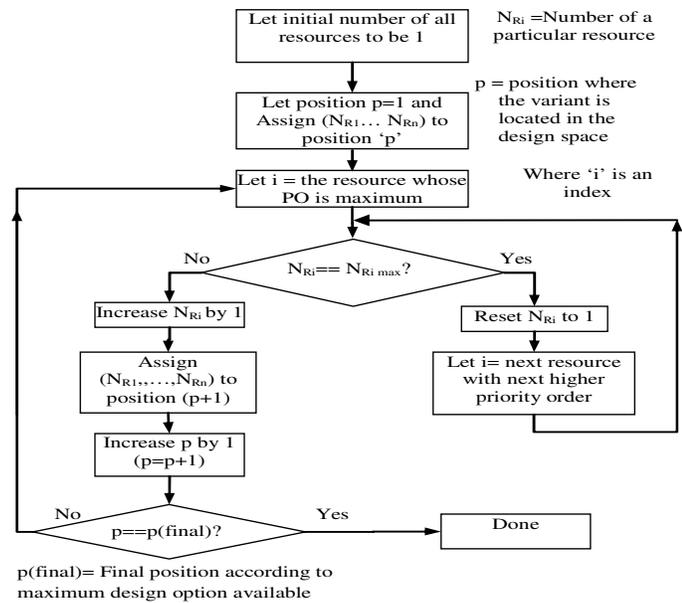


Figure2. Flow chart model of the proposed algorithm

assigned the highest PO. Therefore the following PO of resources is achieved for arranging the design variants in decreasing order for execution time.

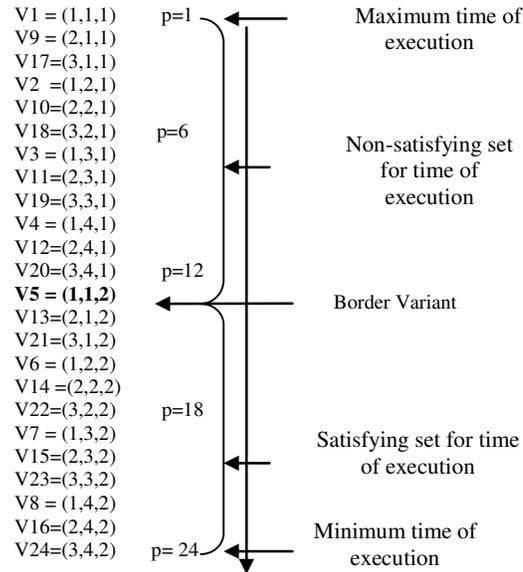
$$PO (R1) > PO (R2) > PO (Rclk)$$

D. Arrange the design space in decreasing order for execution time according to the priority order

This section presents an algorithm for arranging the random design space in an organized decreasing order for the execution time parameter. Before demonstrating the proposed algorithm it is imperative to identify the advantages of this algorithm compared to existing approaches [12]. The algorithm in Figure 2 differs from the existing techniques [12], which are based on the hierarchical criterion method, with respect to the topology used to represent the design variants in the design space. The proposed algorithm is based on vector design space topology while the approach used in [12] is based on tree topology. Further, the proposed algorithm does not need any variant analysis to arrange the design space in increasing/decreasing order. It therefore requires less time while performing exploration. On the contrary, approach [12] utilizes critical variant analysis to determine the hierarchy of each resource type and then constructs the arranged design in increasing/decreasing order.

The proposed algorithm is based on priority order sequencing as described in Section 2.4.C. The algorithm presented in Fig. 2 clearly describes the required steps in order to properly arrange the design variants. The PO obtained for execution time was $PO (R1) > PO (R2) > PO (Rclk)$. After using the model of the proposed algorithm the arranged design space for execution time is obtained and is shown in Fig.3. After the variants were organized in decreasing order the binary search algorithm [63] is applied to obtain the border variant for the execution time parameter. Border variant is the extreme design point in the architecture space that demarcates the points that satisfy and do not satisfy the parametric constraint specified [12]. Results of binary search

[63] on the design space shown in Fig.3 yielded ‘variant V5 (marked in bold) as the border variant for the execution time parameter. This signifies that variant V5 is the first variant in the design space that satisfies the constraint for execution time specified (as given in Table I).



Arrangement of time of execution in decreasing order from the top to the bottom element using the proposed algorithm

Figure3. The arranged design vector space in decreasing order for time of execution

E. Arrange the design space for Power in increasing order, Determination of Pareto Optimal points and Final solution

Similar to the execution time parameter, the PF for all the available resources was calculated to determine how much a change in each resource affects power consumption. Once the PF was determined then the PO was obtained following the procedure described in Section D. The PO sequence (PO (R1) > PO (R2) > PO (Rclk)) helped to obtain the arranged design space for power consumption using the algorithm in Fig. 2. Binary search [63] was then applied on the arranged design space for power consumption. Binary search yielded variant V21 as the border variant

that meets the constraint imposed for power consumption. Since the steps to obtain the border variant for power are exactly the same as those to obtain the border variant for execution time (steps from Section 2.4.B to Section 2.4.D) they have not been shown in the dissertation. The priority factor for area is determined using equations described in this chapter to arrange the variants of the Pareto optimal set in increasing order, similarly to the way it was determined for power and execution time (*Note: If cost was the third parameter then equations (10) – (13) should have been used to find PF and then the final solution*). After calculation of the PF the priority order is determined. The obtained priority order is: PO (Rclk) > PO (R1) > PO (R2). Using the algorithm in Figure2, the variants V5, V13, V21 of the Pareto set are arranged in increasing orders of magnitude. Since the design specification in Table 1 demanded minimum area overhead with simultaneous satisfaction of the constraints imposed by user, hence the aim is to find the variant with minimum area overhead. After the arrangement of the variants of Pareto optimal set the variant number ‘V5’ is found to be the only variant among twenty four variants that concurrently optimizes hardware area, power consumption and time of execution while meeting all the specifications provided. The optimal solution obtained through the proposed approach is reported later in Chapter 7.

Note: The results of this approach applied on various benchmarks and the results of exploration time improvement are reported in Chapter 7.1.

Chapter 3

Design Space Exploration in High Level Synthesis for Area/Power-Performance Tradeoff using Hybridization of Fuzzified Algorithm and Vector Design Space with Self-Correction Scheme

This chapter introduces the solution to another branch of architecture exploration problems based on area-performance constraint and power as optimization criteria. The second algorithm of the dissertation viz. hybridized fuzzified algorithm and vector design space technique with self-correction scheme will be presented. This algorithm proposes solutions to design space exploration in high level synthesis for computation intensive applications based on area-performance constraint with power as optimization criteria. It has also been applied on problems specified as power-performance constraint and area as optimization criteria. The algorithm is also deterministic in nature and therefore finds the final architecture based on resolute evaluation steps (unlike heuristic methods).

3.1 The Proposed Theory for Fuzzy Search Framework during Exploration

Before deducing the functions of fuzzy search for design space exploration, in this section the general concept behind the proposed theory is first discussed. The concept of assigning membership value to each respective element of the set considered in the proposed theory.

Fuzzy set theory involves manipulation of the fuzzy linguistic variables [28]. In fuzzy set theory, the characteristic function is generalized to a membership function that assigns every element 'x' a membership value. The membership function μ_F of a fuzzy set F is a function:

$$\mu_F : U \rightarrow [0,1]$$

A graphical representation of the proposed approach takes into consideration that architectural variants in the architectural design space are already organized in increasing or decreasing order. These architectural variants of the design space will be represented in the form of a fuzzy set where each variant will have a certain assigned membership value based on the characterized membership function as shown later. The membership value will be assigned to each variant taking into consideration that the values of the design space variants are organized in either increasing or decreasing from the left to the right extreme of the fuzzy set (which is equivalent to top to bottom of the design space). In this theory, only the extreme elements' actual values (which are the minimum and the maximum values or maximum and minimum values) are calculated at the beginning. The membership value of the variants between the two extremes will be considered to be directly proportional (sorted increasing order or sorted decreasing order) to the position of the variants in the sorted arrangement. Therefore, the membership value of a variant can be calculated by equations (28) or (29) for design space arranged in increasing or decreasing orders of magnitude:

$$\tau = \frac{x - \alpha}{\beta - \alpha} \quad (28)$$

$$\text{Or, } \tau = \frac{x - \beta}{\alpha - \beta} \quad (29)$$

The actual value of the variant is assumed proportional to the position of the variant in the sorted arrangement. In equation (28) and (29), 'x' is the position of the variant; 'τ' represents the approximated membership value of the variant which is the xth element in the sorted arrangement; 'α' and 'β' are the order of the first element and the last element in the same sorted arrangement. Thus, 'α' is equal to 1 and 'β' is equal to the total number of variants in the sorted arrangement. The above function represents a straight line which will aid in finding the border variant, e.g. the first variant which satisfies the execution time constraint and the last variant in the arranged design space which satisfies the specified constraint for area/power. In all figures shown below, the x-axes refer to the architectural variants of the design space and the y-axes refers to the actual values and membership values respectively. 'τ_B' is the membership value of the border variant for the parameter in the architecture space. Similarly, 'τ_V' is the membership value for the variant under test and V_{Variant} is its respective value. Similarly, 'τ_{Min}' and 'τ_{Max}' are the membership values for the minimum and maximum variants in the architecture space, while 'Max' and 'Min' are its respective values. The increase in trend line for area /power consumption and the decrease in trend line for execution time from left to right extreme of the design space are represented by membership value of each variant. Therefore the actual value of each variant is directly proportional to its associated membership value. An algorithm has been developed to search for the border variant with the given actual value. The graphical representations of the proposed algorithm are shown in Figures 4 to 7.

The trend line shown in Figure4 and Figure6 represents the increase in membership values of each variant in the design space for area/power parameters. The trend line shown in Figure5 and Figure7 represents the decrease in membership values of each variant in the design space for execution time parameter. The membership values in this theory are not calculated separately for each variant but rather by applying equations (28) and (29). After arranging the design space by determining the priority factors in increasing or decreasing order, the membership values of each variant are also arranged in increasing or decreasing order. The actual values of the variants in the design space are directly proportional to the membership values of those variants.

In Figure 4 the increase in membership value for area, i.e. actual area increase, is approximated by the straight line (OR) drawn from origin to the maximum. ‘M’ refers to the point in the line, corresponding to the actual border value (V_{Border}) being searched. ‘ V_1 ’ indicates the initial variant obtained corresponding to the calculated initial membership value (τ_{ini}). ‘P’ is a point in the straight line corresponding to the actual membership value (τ_v) and the actual variant value ($V_{Variant}$) of variant ‘ V_1 ’. If, for example, the variant value ($V_{Variant}$) calculated is less than the value searching for (V_{Border}), then the search should be performed between points ‘P’ and point ‘R’. A second straight line (PR) is approximated for the increase in membership values for area/power parameter. In this straight line point ‘N’ corresponds to the actual border value being searched. τ_{max} and τ_{min} are the maximum and minimum membership values (either ‘1’ or ‘0’). Using similarity of triangles ΔPNQ and ΔPRS the following function is easily derived:

$$\frac{\tau_{Max} - \tau_v}{\tau_B - \tau_v} = \frac{Max - V_{Variant}}{V_{Border} - V_{Variant}} \quad (30)$$

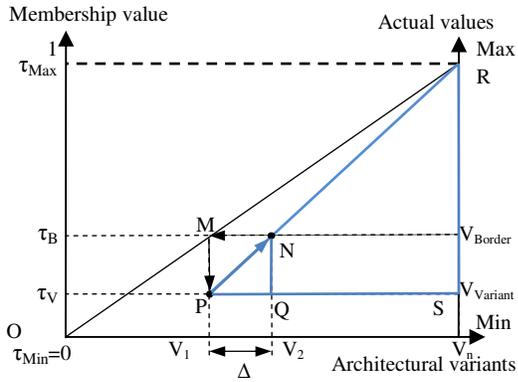


Figure4. Graphical representation of the algorithm for area/power for searching a greater value in the design space

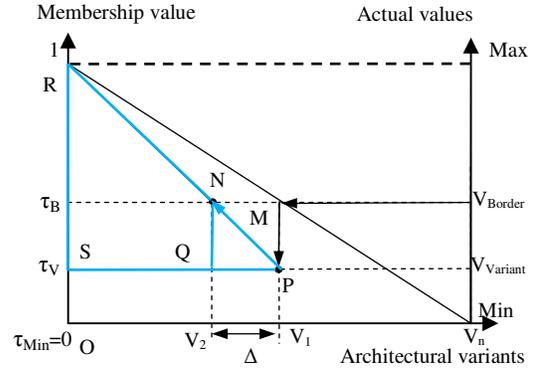


Figure5. Graphical representation of the algorithm for execution time for searching a greater value in the design space

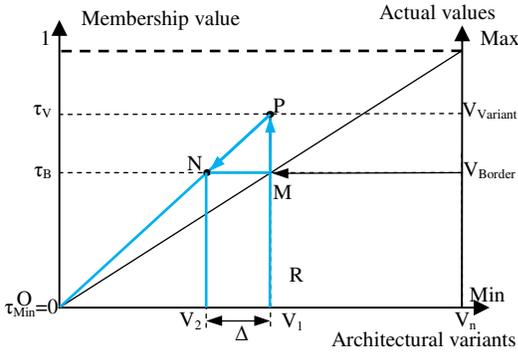


Figure6. Graphical representation of the algorithm for area/power for searching a lesser value in the design space

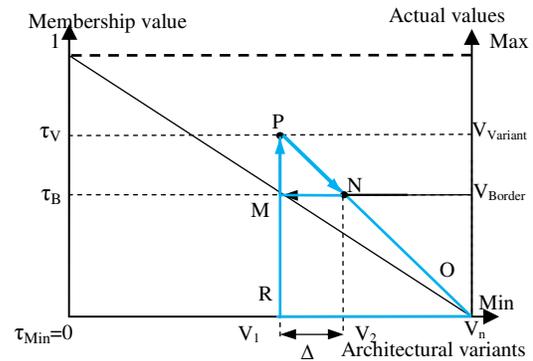


Figure7. Graphical representation of the algorithm for execution time for searching a lesser value in the design space

A similar analysis has been made for execution time with decreasing trend line (Figure 5). The trend line shows the decrease in magnitude of membership value based on the decrease in actual execution time in the arranged design space.

Similar to the previous section, Figure 6 represents the increase trend line for area/power. ‘M’ refers to the point on the line corresponding to the actual border value (V_{Border}). ‘ V_1 ’ indicates the initial variant obtained corresponding to the calculated initial membership value (τ_{ini}). ‘P’ is a point in the straight line corresponding to the actual membership value (τ_V) and the actual variant value (V_{Variant}) of variant ‘ V_1 ’. If, for example, this calculated variant value is more than the value searching for (V_{Border}), then the search should be performed between points ‘P’ and point ‘O’. A second straight line can be approximated for the increase in membership values for

area/power. In this straight line 'N' is a point corresponding to the actual border value searching for (V_{Border}). Using the similarity between the triangles ΔMPN and ΔRPO another function can easily be derived:

$$\frac{\tau_{Min} - \tau_V}{\tau_B - \tau_V} = \frac{Min - V_{Variant}}{V_{Border} - V_{Variant}} \quad (31)$$

Similar analysis has been made for execution time with decreasing trend line. The trend line in Figure 7 shows the decrease in membership value based on the decrease in actual execution time. Similar analysis for execution will yield equation (31). The proposed algorithm is described as follows:

Algorithm

Searching for the border variant (Border)

1. Define the Universe of discourse (The fuzzy set)
2. Identify and define the Linguistic variables
3. Assign the approximate membership values (τ) based on the function described in equation (28) or (29) for each variant in the universe of discourse based on trendline for that parameter (increasing or decreasing).
4. Calculate the initial membership value (τ_{ini}) based on the function:

$$\tau_{ini} = \frac{V_{Border} - Min}{Max - Min} ; \text{ where } \tau \text{ is the initial membership value corresponding to border variant}$$

(V_{Border}). 'Min' and 'Max' are the minimum and maximum values of the variants for a respective parameter.

5. Look for the variant (V) closest to ' τ_{ini} ' in the fuzzy set.
6. Calculate the value of the variant 'V', indicated by $V_{variant}$

7. If $V_{\text{variant}} < V_{\text{Border}}$ then go to step 8, else go to step 10.

8. Solve the membership value (τ_B) based on the following function:

$$\frac{\tau_{Max} - \tau_V}{\tau_B - \tau_V} = \frac{Max - V_{\text{variant}}}{V_{\text{Border}} - V_{\text{variant}}}$$

9. Jump to step 11.

10. Solve the membership value (τ_B) based on the following function:

$$\frac{\tau_{Min} - \tau_V}{\tau_B - \tau_V} = \frac{Min - V_{\text{variant}}}{V_{\text{Border}} - V_{\text{variant}}}$$

11. Look for the variant 'V' which has the closest membership value to ' τ_B ' calculated in step 8 or in step 10.

12. If variant 'V' has already been checked, then

{If $V_{\text{variant}} < V_{\text{Border}}$ then look for the unchecked variant with the next higher membership value in the set, and jump to step 13.

Elseif $V_{\text{variant}} > V_{\text{Border}}$ then look for the unchecked variant with the next lower membership value in the set, and jump to step 13}

Else variant 'V' has not been checked then go to step 13

13. Calculate the V_{variant} .

14. If still the 'Border' is not found then repeat step 7.

15. End

The above algorithm successfully determines the border variant for a respective parameter during searching. The border variant for area and power consumption indicates the last variant in the design space (design space which is arranged in increasing order of magnitude) to satisfy the V_{Border} specified by the user. In contrast, the border variant for execution time is the first variant

in the arranged design space (design space which is arranged in decreasing order of magnitude) that satisfies the V_{Border} specified by the user.

3.2 The Steps Needed to Obtain the Final Variant of Architecture

The proposed theory behind the framework for DSE will be used in the upcoming sections. Additionally the fuzzy search algorithm proposed in Section 3.1 will be used as a method for searching the final architecture after our design space is organized in increasing or decreasing order based on the PF calculation (as explained in Section 3.2.F). The steps required to obtain the final architecture for high level synthesis is explained in this section with three objectives being satisfied (for hardware area, time of execution and power consumption). The goal of the proposed DSE approach is to find the final optimal variant of architecture which satisfies all three parameters specified in the design problem. Figure 8 shows the exploration process steps which are required to obtain the optimal variant of architecture using the proposed DSE methodology for high level synthesis designing. The proposed exploration approach has been designed for custom data intensive applications (application specific processors/custom processors) or standalone Application Specific Integrated Circuits (ASIC's) rather than systems that include adaptable applications which change dynamically during runtime.

A. Problem formulation and Technical specifications

This stage marks the beginning of high level synthesis designing, beginning with the problem description and the technical specifications provided for the designer. The application should be properly defined with its associated data structure. This phase is very critical for the designer and the operational constraints should be clearly defined along with the parameters to be optimized.

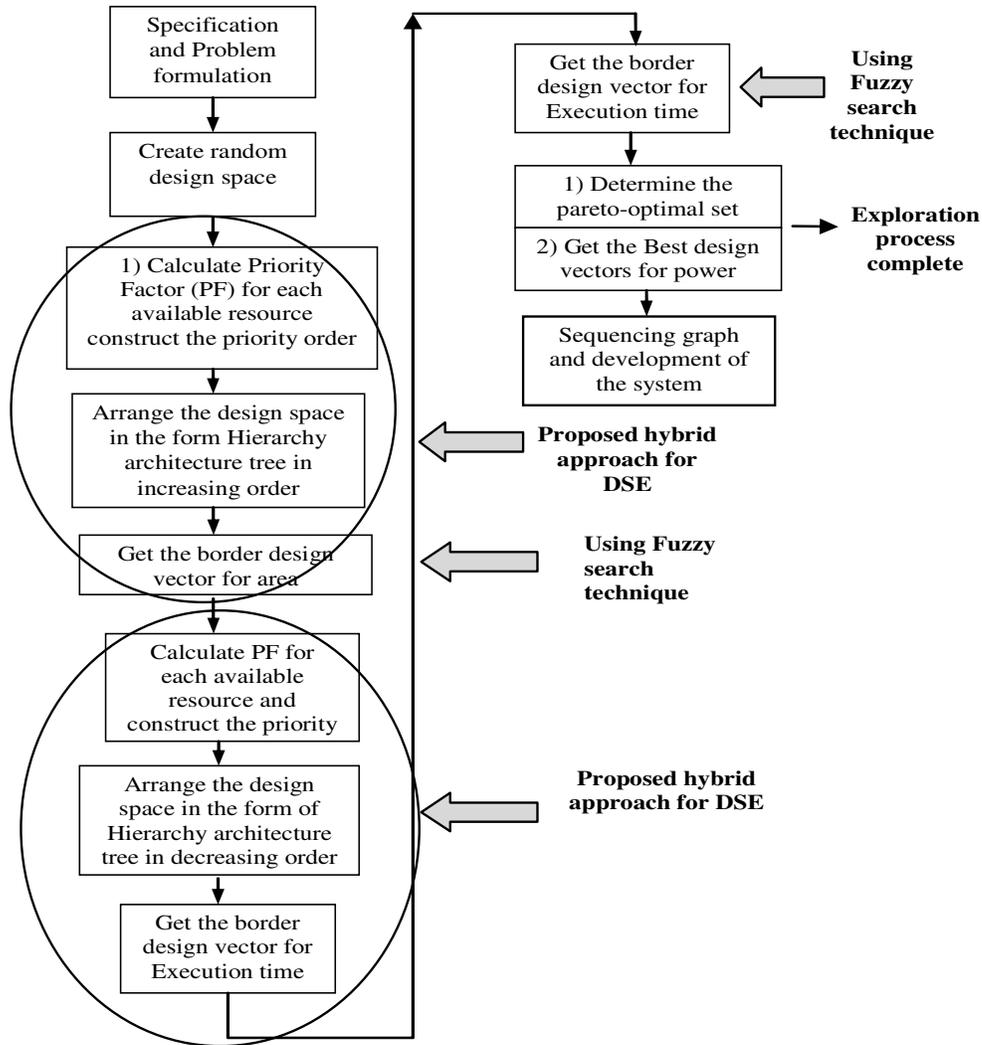


Figure8. The flow for the steps required to obtain the optimal variant of architecture using the proposed DSE

These specifications act as the input information for the high level synthesis tools. To demonstrate the DSE approach the following sample specifications were assumed as shown in Table 2. The specifications used for the area occupied by each adder/subtractor and multiplier were based on the results obtained after each module (resource) was synthesized and implemented using Xilinx ISE9.2i tool in Spartan 3E FPGA (XCS3E5000E-5fg320 FPGA). For example, synthesis and implementation of a type of adder/subtractor module occupies 12 CLB's in XCS3E5000E-5fg320 FPGA.

However, the provided constraints do not always yield a solution. There are two situations where this could occur: a) when the constraints provided are invalid or b) if the Pareto set is empty. This would signify that no solution exists which simultaneously satisfies the two constraints, which would then be considered too tight and need to be relaxed. Thus the two generalized algorithms proposed for constraints validation check of a parametric constraint are:

Algorithm 1 (Extremism check):

Inputs: Module Library, Data Flow Graph (or Mathematical function) of the application and user Constraints

Output: The decision whether the design process continues or terminates (i.e. constraints are

Table2. System Specifications for hybrid fuzzy approach

1) Maximum hardware area of resources: 160 area units (a.u) (Note: The specification for Power consumption could also be assumed)
2) Maximum time of execution: 200 μ s (For 1000 sets of data)
3) Power consumption: Minimum.
4) Maximum resources available for the system design: a) 3 Adder/subtractor units. b) 3 Multiplier units c) 3 clock frequency oscillators: 24 MHz, 100 MHz and 400 MHz (Note: The choice of the 3 clock frequencies is arbitrary and any other clock frequency oscillator could also be used)
5) No. of clock cycles needed for multiplier and adder/subtractor to finish each operation: 4 cc and 2cc
6) Area occupied by each adder/subtractor and multiplier: 12 a.u. and 65 a.u. on the chip (e.g. 12 CLB on FPGA for adder/subtractor)
7) Area occupied by the 24MHz, 100MHz and 400 MHz clock oscillators are: 6 a.u., 10 a.u. and 14 a.u. respectively.
8) Power consumed at 24MHz, 100MHz and 400 MHz: 10mW/a.u.,32 mW/a.u. and 100mW/a.u. respectively.

valid or invalid)

Repeat for all the user constraints specified

{

1. Calculate the **minimum value** of the optimization parameter under consideration. Calculate the ***minimum value*** of the hardware area (power consumption)/execution time based on the minimum resource/maximum resource (considering whichever parameter among hardware area, power consumption or execution time is the first user constraint) using any one of the functions described below based on the user requirement:

In case of hardware area:

$$A_{\min} = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + \dots + N_{Rn} \cdot K_{Rn}) + A(R_{clk})$$

Where, N_{Ri} represents the number of resource R_i and is equal to 1 for all cases. Therefore for calculating the minimum area, $N_{R1} = N_{R2} = N_{R3} = \dots = N_{Rn} = 1$. Also ' K_{Ri} ' represents the area occupied per unit resource ' R_i ' which is obtained from the user as input. $A(R_{clk})$ refers to the area of clock oscillator which occupies least area used for providing the necessary clock frequency to the system. ' K_{Ri} ' represents the area occupied per unit resource ' R_i ' ($1 \leq i \leq n$).

In case of power consumption:

$$P_{\min} = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + \dots + N_{Rn} \cdot K_{Rn}) \cdot p_c$$

Therefore for calculating the minimum area, $N_{R1} = N_{R2} = N_{R3} = \dots = N_{Rn} = 1$. Moreover, ' p_c ' is the slowest clock frequency available in the module library which consumes the least power per unit area.

In case of execution time:

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

‘L’ and ‘T_c’ should be calculated based on maximum resources considering $N_{R1} = N_{R2} = N_{R3} = \dots = N_{Rn} =$ Maximum resource of certain functional unit specified by user in the library. ‘L’ represents latency of execution, ‘T_c’ represents the cycle time of execution during data pipelining. Also, ‘N’ is the number of data sets to be pipelined as user input.

2. Calculate the **maximum value** of the optimization parameter under consideration. Calculate the ***maximum value*** of the hardware area based on the minimum resource (considering that hardware area is the first user constraint) using the function described below:

In case of hardware area:

$$A_{\max} = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + \dots + N_{Rn} \cdot K_{Rn}) + A(R_{clk})$$

Where, N_{Ri} represents the number of resource R_i . Therefore for calculating the maximum area, $N_{R1} = N_{R2} = N_{R3} = \dots = N_{Rn} =$ Maximum resource of certain functional unit specified by user in the library. Also ‘ K_{Ri} ’ represents the area occupied per unit resource ‘ R_i ’ which is obtained from the user as input.

In case of power consumption:

$$P_{\max} = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + \dots + N_{Rn} \cdot K_{Rn}) \cdot P_c$$

Therefore for calculating the minimum area, $N_{R1} = N_{R2} = N_{R3} = \dots = N_{Rn} =$ Maximum resource of certain functional unit specified by user in the library. Moreover, ‘ p_c ’ is the fastest clock frequency available in the module library which consumes the maximum power per unit area.

In case of execution time:

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

'L' and 'T_c' should be calculated based on minimum resources considering $N_{R1} = N_{R2} = N_{R3} = \dots = N_{Rn} = 1$.

3. Check if **Constraint specified satisfies** the **upper threshold** (maximum value) and **lower threshold** (minimum value) of the parameter calculated above in steps 1 and 2. In other words, let the constraint for hardware area is 'A_{const}', power consumption is 'P_{const}' and execution time is 'T_{const}'. Then, the following conditions are checked:

$A_{\min} \leq A_{\text{const}} \leq A_{\max}$ (For Hardware area)

$T_{\min} \leq T_{\text{const}} \leq T_{\max}$ (For Execution time)

$P_{\min} \leq P_{\text{const}} \leq P_{\max}$ (For Power consumption)

If the above conditions are satisfied **then**, the design process continues

Elseif the above conditions fail **then** the design process stops and constraints need to be corrected by the user.

}

END

Algorithm 2 (Relaxation Phase): This constraints validation check only comes into play when the Pareto optimal set formed as a result of the design space exploration process is absolutely vacant. A vacant Pareto optimal set signifies that the user constraints provided (which passed the test of 'Extremism Check') is too tight/strict in deadline. Therefore the strict user constraints of the given parameter need to be relaxed (self-corrected) to a certain extent:

1) Let the variants obtained in the pareto optimal set (P) after applying the proposed design space exploration approach be $P = \{V_a, V_b, V_c, \dots, V_n\}$, where $V_a, V_b, V_c, \dots, V_n$ are variants of the design space that are elements of the pareto optimal set.

2) If the Pareto optimal set, $P = \phi$ (Null), then there exists no variants in the set P. This indicates that the user constraints are too tight and needs to be relaxed. This is because there is no variant from the design space that simultaneously obeys both user constraints. *Continue the algorithm.*

Elseif $P \neq \phi$ (not null), then variants exists in the Pareto set, P. Continue the design process and stop the algorithm.

3) Relax the user constraints by 5 % to set new constraints for the parameter. Therefore, if the constraint for hardware area is ' A_{const} ', constraint for power consumption is ' P_{const} ' and constraint for execution time is ' T_{const} ', then depending on the user specified constraints, the new constraints after applying the relaxation phase are as follows:

a) $A_{\text{const}}(\text{new}) = A_{\text{const}}(\text{original}) + 5\% \text{ of } A_{\text{const}}(\text{original})$

b) $T_{\text{const}}(\text{new}) = T_{\text{const}}(\text{original}) + 5\% \text{ of } T_{\text{const}}(\text{original})$

c) $P_{\text{const}}(\text{new}) = P_{\text{const}}(\text{original}) + 5\% \text{ of } P_{\text{const}}(\text{original})$

B. Problem formulation stage

In the problem formulation stage the mathematical model of the application is used to define the behavior of the algorithm. The model suggests the input/ output relation of the system and the data dependency present in the function. In this paper the digital IIR Chebyshev filter is used

as an example benchmark to demonstrate the DSE method for high level synthesis. The transfer function of a second order digital IIR Chebyshev filter can be given as:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.041 + 0.082z^{-1} + 0.041z^{-2}}{1 - 1.4418z^{-1} + 0.6743z^{-2}} \quad (32)$$

$$y(n) = 0.041x(n) + 0.082x(n-1) + 0.041x(n-2) - 0.6743y(n-2) + 1.4418y(n-1) \quad (33)$$

Where $x(n)$, $x(n-1)$ and $x(n-2)$ are the input vector variables for the function. The previous outputs are given by $y(n-1)$ and $y(n-2)$, while the present output of the function is given by $y(n)$. For simplicity, constants 0.041, 0.082, 0.6743 and 1.4418 are denoted by 'A', 'B', 'D' and 'E' respectively. $x(n)$, $x(n-1)$, $x(n-2)$, $y(n-1)$ and $y(n-2)$ are denoted by X_n , X_{n1} , X_{n2} , Y_{n1} and Y_{n2} respectively.

C. Creation of a random architecture vector design space for area parameter

The architecture design space is represented in the form of vectors consisting of the resources available for the system. The random organization of the design space is shown only to represent the different combinations of the resources that make the total design space. The total design space is first created according to the specifications mentioned for total available resources in Section VI.A of the flow. $V_n = (N_{R1}, N_{R2}, N_{R3})$ represents the architecture design space. The variables N_{R1} , N_{R2} and N_{R3} indicate the number of adders/subtractors, multipliers and clock frequencies. Therefore, according to specification in Section Table2, $1 \leq N_{R1} \leq 3$, $1 \leq N_{R2} \leq 3$, and $1 \leq N_{R3} \leq 3$. The random design space in Figure 9 represents all the different combinations

V1 = (1,1,1)	V8 = (1,2,3)	V15 = (2,3,2)	V22 = (3,1,2)
V2 = (1,2,1)	V9 = (1,3,3)	V16 = (2,1,3)	V23 = (3,2,2)
V3 = (1,3,1)	V10 = (2,1,1)	V17 = (2,2,3)	V24 = (3,3,2)
V4 = (1,1,2)	V11 = (2,2,1)	V18 = (2,3,3)	V25 = (3,1,3)
V5 = (1,2,2)	V12 = (2,3,1)	V19 = (3,1,1)	V26 = (3,2,3)
V6 = (1,3,2)	V13 = (2,1,2)	V20 = (3,2,1)	V27 = (3,3,3)
V7 = (1,1,3)	V14 = (2,2,2)	V21 = (3,3,1)	

Figure9. Design space with all possible resource combinations

of available resources viz. adder/subtractor, multiplier and oscillators. The next section describes the methodology of calculation of priority factor (PF) for the area.

D. Calculation of the priority factor for each available resource and arrangement of the PF in increasing order for area parameter using DSE approach

Using equations described in Chapter 3, the PF for resource adder /subtractor (R1) is:

$$PF(R1) = 8$$

For resource multiplier (R2):

$$PF(R2) = 43.33$$

For resource clock oscillator (Rclk):

$$PF(Rclk) = 2.67$$

The above factors are a true measure of the change in area with the change in number of a specific resource. For example, according to the above analysis, the change in number of multiplier affects the change in area the most, while the change in clock frequency from 24 MHz to 400 MHz influences the change in area the least. Now based on the PF calculated for area, the hierarchy vector space is constructed using the algorithm in Figure 2 (in Chapter 2).

E. Arrangement of the hierarchy vector space comprising the design space in increasing order

Based on the PF calculated for each resource for area (as shown in previous section), the hierarchy vector space is constructed using the algorithm in Figure 2. The arranged design space for area in form of hierarchy vector space according to the explanation above is shown in Figure 10. This arrangement of the hierarchy architecture tree ensures that the design space becomes sorted. There is no additional necessity to sort the variants of the design space.

F. Fuzzy search technique for the determination of the border variant for the area and calculation of PF for execution time

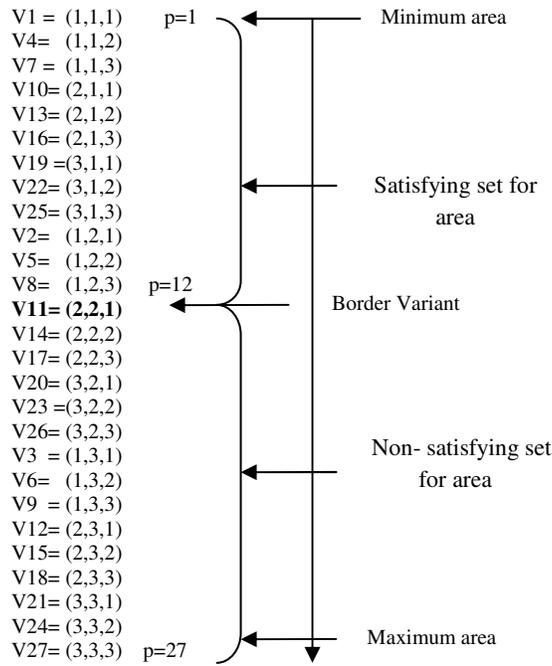
The presented fuzzy search technique is applied on the arranged design space in Figure 10. The design space shown is arranged in increasing orders of magnitude from the north extreme to the south extreme. This arrangement helps to prune the design space in order to obtain the border variant for area requirement. After arrangement of the vector space, the universe of discourse set is constructed based on the arrangement of the variants in the design space in increasing order in the vector space (In other words, the set reflects the arranged design space in the tree, for area as shown in Figure 10). The universe of discourse for area can be represented by the set shown below. After defining the set, the algorithm described in Section 3.1 is used.

Large area set (μ_L) =

$$\left\{ \frac{0}{V_1}, \frac{0.038}{V_4}, \frac{0.077}{V_7}, \frac{0.115}{V_{10}}, \frac{0.154}{V_{13}}, \frac{0.192}{V_{16}}, \frac{0.231}{V_{19}}, \frac{0.269}{V_{22}}, \frac{0.308}{V_{25}}, \frac{0.346}{V_2}, \frac{0.385}{V_5}, \frac{0.423}{V_8}, \frac{0.462}{V_{11}}, \frac{0.500}{V_{14}}, \frac{0.538}{V_{17}}, \frac{0.577}{V_{20}}, \frac{0.615}{V_{23}}, \frac{0.654}{V_{26}}, \frac{0.692}{V_3}, \frac{0.731}{V_6}, \frac{0.769}{V_9}, \frac{0.808}{V_{12}}, \frac{0.846}{V_{15}}, \frac{0.885}{V_{18}}, \frac{0.923}{V_{21}}, \frac{0.962}{V_{24}}, \frac{1}{V_{27}} \right\}$$

Where ‘Large area’ is a linguistic variable for the set defined above. According to step 4 of the fuzzy search algorithm the initial membership value (τ_{ini}) is calculated based on the Min and the Max value of area. According to the specification provided for area, $V_{Border} = 160$ a.u. while $Min = 83$ a.u. and $Max = 245$ a.u. (according to equation 1) are the calculated minimum and maximum values of the variants with minimum and maximum resources respectively.

As shown in Table 3, the fuzzy search technique finds the border variant in just one iteration. The border variant obtained is variant 11. This value indicates the last variant in the space which satisfies the constraint for area requirement (V_{Border}).



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

Figure10. The vector design space in increasing order for area

Table3. The variants obtained for area after applying fuzzy search on the arranged design space

Equations for obtaining the calculated membership values	Calculated membership values(τ)	Variants corresponding in the set according to the calculated ' τ '	Area	Decision based on the V_{Border}
$\tau_{ini} = \frac{160 - 83}{245 - 83}$	$\tau_{ini} = 0.475$	0.462/V11	$A^{11} = 2*12 + 2*65 + 6 = 160 \text{ a.u.}$	$A^{11} \leq V_{\text{Border}}$, stop

After the border variant for area is obtained, the priority factor for each available resource type for time of execution parameter is calculated. Hence the PF obtained for each resource for execution time parameter are as follows:

For resource adder/subtractor (R1):

$$PF(R1) = 0.055$$

For resource multiplier (R2):

$$PF(R2) = 0.1109$$

For resource clock oscillator (R_{clk}):

$$PF(Rclk) = 0.2346$$

The factors determined above indicate a measurement of the change in time of execution with the change in number of a specific resource. For instance, according to the above analysis the change in number of adder/subtractor affects the change in time of execution the least, while the change in clock frequency from 24 MHz to 400 MHz affects the change in time of execution the most. Now based on the PF calculated for execution time, the hierarchy vector space is constructed using the algorithm in Figure 2.

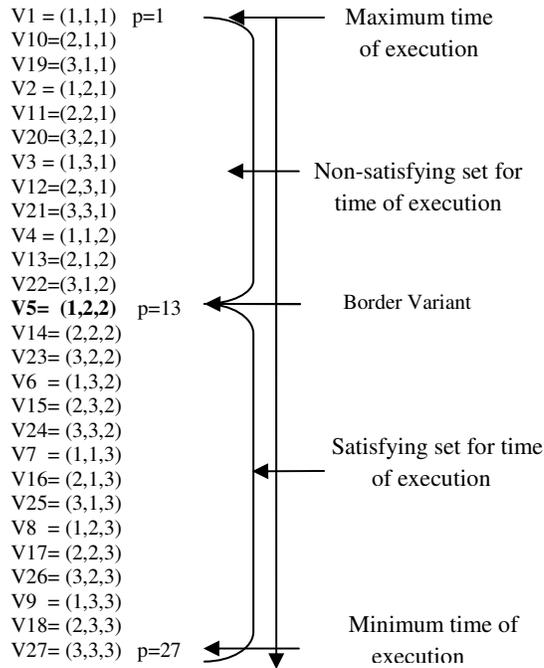
G. Arrangement of the hierarchy vector design space in decreasing order and use of fuzzy search technique to determine the border variant for time of execution parameter

(Note: The universe of discourse set for execution time is constructed based on the arrangement of the variants in the design space in decreasing order, similar to the way the set was constructed for area). As explained in Sections E and F, design space is similarly arranged in decreasing orders of magnitude for execution time as shown in Figure11. The universe of discourse set for time of execution parameter is defined as:

Small time of execution set (μ_s) =

$$\left\{ \frac{1}{V1}, \frac{0.962}{V10}, \frac{0.923}{V19}, \frac{0.885}{V2}, \frac{0.846}{V11}, \frac{0.808}{V20}, \frac{0.769}{V3}, \frac{0.731}{V12}, \frac{0.692}{V21}, \frac{0.654}{V4}, \frac{0.615}{V13}, \frac{0.577}{V22}, \frac{0.538}{V5}, \frac{0.500}{V14}, \frac{0.462}{V23}, \frac{0.423}{V6}, \frac{0.385}{V15}, \frac{0.346}{V24}, \frac{0.308}{V7}, \frac{0.269}{V16}, \frac{0.231}{V25}, \frac{0.192}{V8}, \frac{0.154}{V17}, \frac{0.115}{V26}, \frac{0.077}{V9}, \frac{0.038}{V18}, \frac{0}{V27} \right\}$$

According to the algorithm in step 4 for fuzzy search, the initial membership value (τ_{ini}) is calculated based on the Min and the Max value of time of execution. According to the specification, $V_{Border} = 200 \mu s$, while $Min = 20.01 \mu s$, $Max = 833.41 \mu s$ are the calculated minimum and maximum values (using equation 14) of the variants with maximum and minimum



Arrangement of time of execution in decreasing order from the top to the bottom element using the proposed algorithm

Figure11. The vector design vector space in decreasing order for time of execution

resources respectively. As shown in Table 4 the proposed fuzzy search technique finds the border variant (variant 5) in just five comparisons. This value indicates the first variant in the design space which satisfies the constraint for execution time specified (V_{Border}).

H. Determination of the Pareto-optimal set of design architecture

After successfully obtaining the border variants for both parameters the next step is to find the design variants which simultaneously satisfy the specifications for area and execution time defined in the design problem. This step is driven by the fact that the designer must arrive at a point of optimum suitability for the application according to the user specifications for the multi parameters provided. After satisfying both parameters, the next step is to satisfy the power consumption parameter for the variants obtained. After observing the design space shown in

Table4. The variants obtained for execution time after applying fuzzy search on the arranged design space

Equations for obtaining the calculated membership values	Calculated membership values(τ)	Variants corresponding in the set according to the calculated ' τ '	Execution time	Decision based on the V_{Border}
$\tau_{ini} = \frac{200 - 20.01}{833.41 - 20.01}$	$\tau_{ini} = 0.2213$	0.231/V25	$T_{exe}^{25} = (22 + (1000-1)*20) * 0.0025 = 50.005 \mu s$	$T_{exe}^{25} < V_{Border}$, search up in the design space
$\frac{1 - 0.231}{\tau_B - 0.231} = \frac{833.41 - 50.005}{200 - 50.005}$	$\tau_B = 0.378$	0.385/V15	$T_{exe}^{15} = (12 + (1000-1)*8) * 0.01 = 80.04 \mu s$	$T_{exe}^{15} < V_{Border}$, search up in the design space
$\frac{1 - 0.385}{\tau_B - 0.385} = \frac{833.41 - 80.04}{200 - 80.04}$	$\tau_B = 0.483$	0.500/V14	$T_{exe}^{14} = (14 + (1000-1)*10) * 0.01 = 100.04 \mu s$	$T_{exe}^{14} < V_{Border}$, search up in the design space
$\frac{1 - 0.500}{\tau_B - 0.500} = \frac{833.41 - 100.04}{200 - 100.04}$	$\tau_B = 0.568$	0.577/V22	$T_{exe}^{22} = (22 + (1000-1)*20) * 0.01 = 200.02 \mu s$	$T_{exe}^{22} > V_{Border}$, search down in the design space
$\frac{0 - 0.577}{\tau_B - 0.577} = \frac{20.01 - 200.02}{200 - 200.02}$	$\tau_B = 0.577$	0.538/V5 (Since V22 has been checked so according to the algorithm check V5)	$T_{exe}^5 = (14 + (1000-1)*10) * 0.01 = 100.04 \mu s$	Stop

Figure10 and Figure11, five architectural variants are found to simultaneously satisfy the constraints specified. Hence the variants V5, V7, V16, V25, V8 belong to Pareto optimal set.

Next, the priority factor for the power consumption parameter is determined according to Chapter 3. After calculation of the PF, the priority order is determined. The obtained priority order is: $PO(R1) > PO(R2) > PO(Rclk)$. Similar to the method described in Section 3.2.F, the variants of the Pareto set for power consumption are then arranged in increasing order of magnitude by using the proposed method. According to the specification (see Table 2) provided, the variant with the minimum power consumption should be selected. When the variants of the Pareto set are arranged in increasing orders of magnitude, the first variant in the order is guaranteed to be the variant with the minimum value (i.e. minimum power consumption) with the value increasing respectively with each next variant in the set. Hence, after arrangement of the variants in increasing order, it is found that 'V5' is the one with the minimum power consumption as it is the first variant after arrangement. Therefore, the final optimal solution found (V5) with 1 adder/subtractor, 2 Multiplier, 100 MHz clock occupies 152au. This signifies that in the FPGA the optimal solution for the hardware resources occupies approx. 152 CLB's.

Therefore, even if the Pareto optimal set obtained after determining the border variant using fuzzy search is large, the arrangement of the variants in the Pareto optimal set, in increasing order using PF is the only requirement to determine the variant with the minimum power consumption. Therefore the variant obtained is regarded as the final optimal variant for the system design as it concurrently satisfies the specification of all the three objectives for execution time, hardware area and power consumption.

Note: The detailed results of this approach applied on various benchmarks and the results of exploration time improvement are reported in Chapter 7.2.

Chapter 4

Priority Function Driven Design Space Exploration in High Level Synthesis Based on Power Gradient Technique

This chapter introduces the third novel algorithm of the dissertation viz. a novel heuristic based multi objective exploration process based on power gradient theory that simultaneously reduces the static power dissipation at the usage of minimal control step (time step) during scheduling. The proposed iterative power aware integrated optimization approach is based on Priority Indicator (PI) function which is responsible for minimizing allocated hardware functional units during the scheduling process. When the final solution was compared to a Genetic Algorithm for the benchmarks it resulted in a considerable power reduction, runtime reduction as well as improvement in the quality of final solution.

4.1 The Proposed Exploration Approach

The proposed approach accepts the data flow graph (DFG) of the application as an input along with the set of module library information. According to the proposed approach, in each iteration only one operation (node) can be moved at a time into its next immediate control step as long as the dependency is obeyed. The selection of a particular operation (node) is chosen based on the value of 'PI'. The PI acts as a determining metric to choose the highest priority node (operation) among the existing available movable operations that can result in reducing the power of the final solution. The flow of the proposed approach is shown in Figure 12. The detailed algorithm of the proposed approach is shown in Figure 13. The main motivation behind this research is the development of a novel technique that can simultaneously reduce the static power dissipation at the usage of minimal control step (time step) during scheduling by capturing the power gradient effect which arises during movement of operations.

4.1.1 Proposed Power Gradient and Priority Indicator (PI)

Power Gradient (G) is a metric that models the power consumption relationship between the two consecutive control steps (j) and (k) as well as the power consumption relationship between scheduling solutions before and after movement of an operation. Hence effectively 'G' signifies the difference in Power consumption (Pc) between control step (CS), CS (j) and CS (k) before movement of operation o(i) and difference in power consumption between CS (j) and CS (k) after movement of o(i). The mathematical expression of power gradient between any two arbitrary control steps CS (j) and CS (k) is shown below in equation (34).

$$G = S_{before} - S_{after} \quad (34)$$

S_{Before} = Difference in Pc before movement of o(i).

$$= (Pc) \text{ before movement at CS (j)} - (Pc) \text{ before movement at CS (k)}$$

$$\begin{aligned}
S_{After} &= \text{Difference in } P_c \text{ after movement of } o(i) \\
&= \{(P_c) \text{ after movement at CS } (j) - (P_c) \text{ opn } o(i)\} \\
&\quad - \{(P_c) \text{ after movement at CS } (k) + (P_c) \text{ opn } o(i)\}
\end{aligned}$$

Where, CS (j) and CS (k) are the two immediate control steps in the scheduling solution, opn (i) is the operation selected for movement through the ‘PI’ metric. The metric called power gradient defined in equation (34) will be used in the ‘PI’ metric described later for selection of the highest priority node for movement during optimization power gradient defined above takes into account

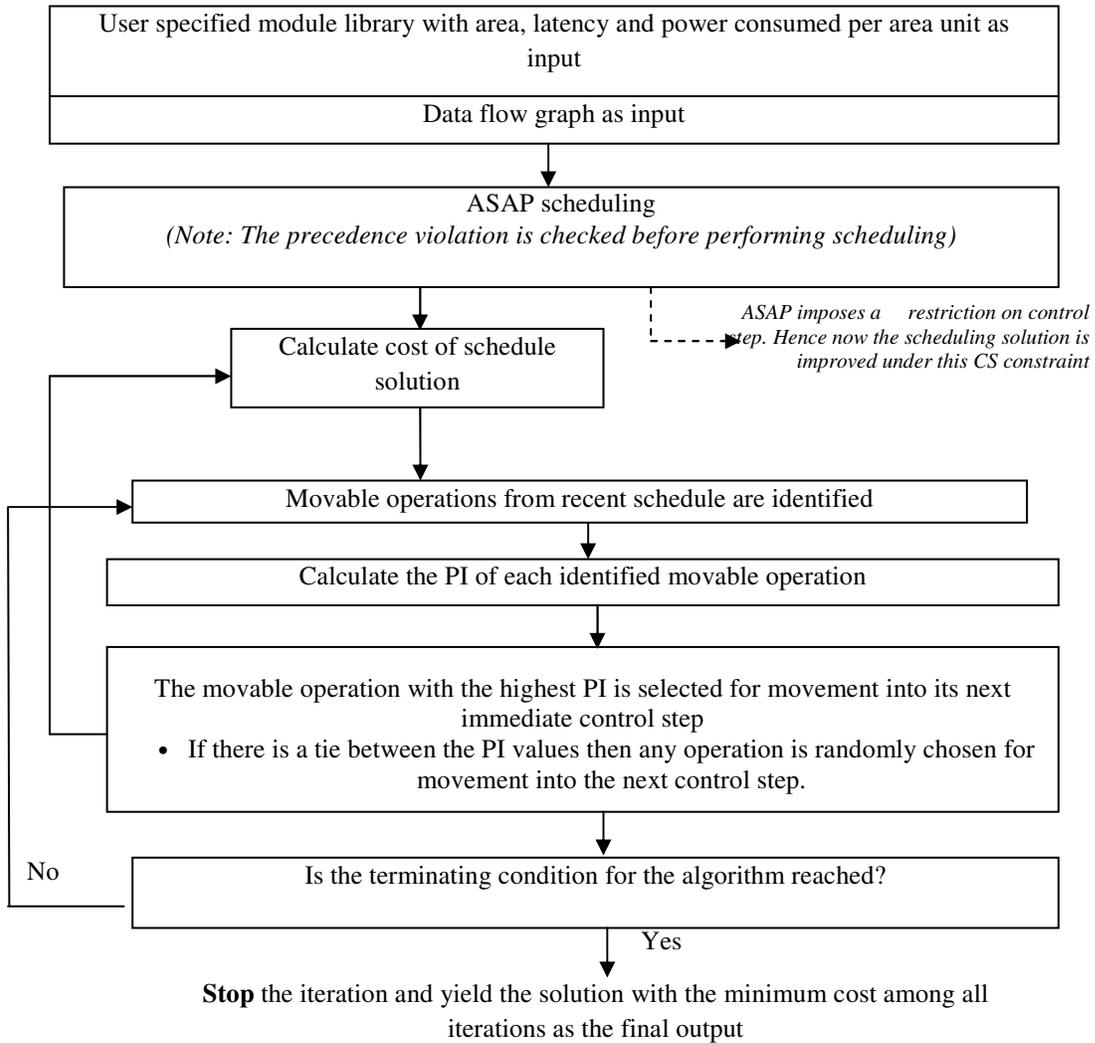


Figure12. The overview of the proposed heuristic approach

the change in power dissipation and its effect on a scheduling solution when a certain operation is moved from one control step to another. The proposed priority indicator metric used for selection of high priority nodes during movement is shown in equation (35):

$$PI = G * Max [Pc (j), Pc (k)] * cost [opn (i)] \quad (35)$$

Where ‘cost [opn (i)]’ is the development cost of resource o (i) obtained from the module library, ‘G’ is obtained from equation (34) and ‘Max [Pc (j), Pc (k)]’ signifies the maximum of the power dissipation between CS (j) and CS (k).

In eqn (35), ‘cost [opn (i)]’ is considered as the associated development cost of the respective operational resource (oi) which is one of the factors that contributes to the PI of the node during selection.

4.2 Demonstration of the proposed approach

In the proposed approach, the cost of each schedule solution is evaluated in terms of static power. One of the most significant ways to reduce static power is to minimize the number of functional units (i.e. reduce the number of resources in order to reduce the standby/leakage current of the system). Therefore in Equation (36), power dissipated per area unit multiplied by the total area of the resources (in area units) yields the total static power of the resources (i.e. contribution of static power by major functional resources). Therefore for high level estimation (contributed by major functional resources) the static power cost of each schedule solution can be determined using Equation (36).

$$C_{PT} = \sum_{i=1}^n (N_{Ri} \cdot K_{Ri}) \cdot S_P \quad (36)$$

' N_{R_i} ' represents the number of resource R_i as mentioned. ' K_{R_i} ' represents the area occupied per unit resource R_i and ' S_p ' denotes the power dissipated per area unit when the transistors in the chip are not switching (standby mode). The cost model used in (36) is only used in estimating the likely high level static power dissipation during the demonstration of the proposed method. According to the algorithm proposed in Figure 13, all movable candidate operations are identified for movement from initial As Soon As Possible (ASAP) scheduling of Discrete Wavelet Transformation (DWT) benchmark. The DWT benchmarks are standard high level synthesis benchmarks that were selected for demonstration of the proposed approach [23], [30]. The ASAP is shown in Figure 14.

Iteration (1):

- i) Movement – opn 2 (1→2) ii) Movement - opn 7 (2→3)
- iii) Movement – opn 8 (2→3) iv) Movement – opn 9 (2→3)

For example (i) above signifies that opn 2 is one of the identified movable operations that can be

```

1.0 Obtain ASAP schedule (put a limit on the number of CS) and determine its cost (PT)
1.1 Repeat {
Count = 1;
1.2 while there is a movable operation
Begin
1.3 Calculate the Power Gradient (G) of each movable operation;
//Power Gradient (G) is a metric that models the power dissipation relationship between consecutive control steps CS (J) and CS (k) as well as power dissipation relationship between scheduling solutions before and after movement of oi.//
1.4 Calculate the Priority Indicator (PI) of each movable operation;
// PI is the selection mechanism based on power when oi is moved from CS (J) to CS (K)//
1.5 Select a 'Movement' for moving an operation to the mentioned CS;
//Movement selects a duo of operation and its destined CS based on maximum PI of movable operation//
1.5.1 If there is a tie in the PI values, then
Any movement is randomly chosen;
1.6 move oi from CS (J) to CS (K) to obtain the new schedule;
1.7 Determine its associated cost (PT) of the schedule;
1.7 increment count, count = count + 1;
1.8 Freeze the Movement (also operation, oi, cannot traverse backward);
1.9 End;
} Until
N = 25:

```

Figure13. Details of the proposed heuristic Exploration approach

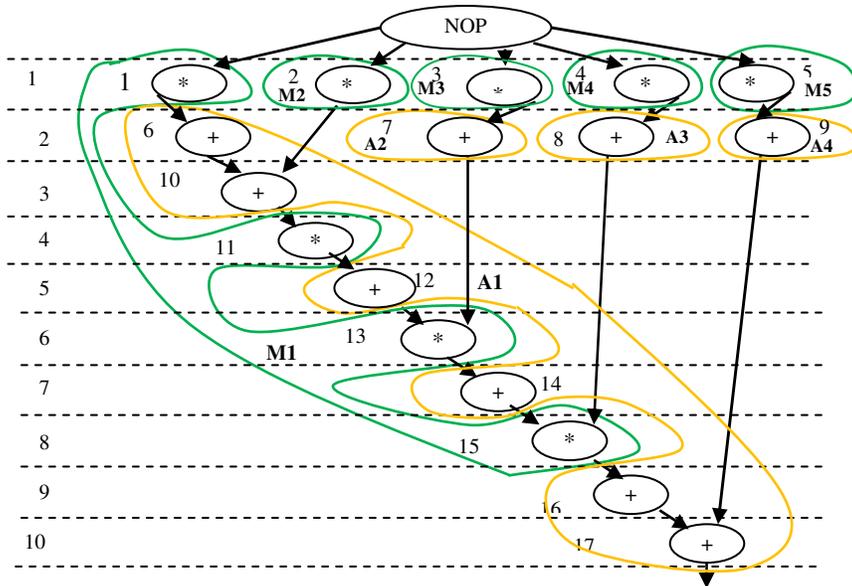


Figure14. ASAP scheduling of DWT benchmark

moved from CS 1 to CS 2. (*Note: Movable operations are those that have a free space to move in the immediate next CS without disturbing the data dependency present. The representation of the movable operations of ASAP schedule is shown above in i), ii), iii) and iv)). The priority indicator (PI) for each identified movable operation is then calculated using equation (35). But before the PI is calculated the ‘power gradient’ is determined using equation (34) as follows:*

$$G = [(400 + 400 + 400 + 400 + 400) - (80 + 80 + 80 + 80)] - [(400 + 400 + 400 + 400) - (80 + 80 + 80 + 80 + 400)] = 800.$$

(*Note: From equation (36) power dissipated by each multiplier and adder is 400mW and 80mW*)

Therefore substituting the value of respective ‘G’ for each case i), ii), iii) and iv) as calculated above for case i):

i) $PI_{opn\ 2\ (1 \rightarrow 2)} = 800 * \text{Max}(2000, 320) * 5 = 8000000$ (selected).

Similarly, calculating the ‘G’ for each case and then finding the Priority Indicator yields:

ii) $PI_{opn\ 7\ (2 \rightarrow 3)} = 160 * \text{Max}(320, 80) * 3 = 153600.$

iii) $PI_{opn\ 8\ (2 \rightarrow 3)} = 160 * \text{Max}(320, 80) * 3 = 153600.$

iv) PI opn 2 (1→2) = 160* Max (320, 80) * 3 = 153600.

According to the next step of the algorithm, the highest PI is selected for movement which is (i).

The power consumption of the scheduling solution is $\{(4 * 100) + (4 * 20) + (16 * 3) + (8 * 3) + (15 * 5)\} * 4 = 2.50$ Watts. (*Note*: assuming multiplier and adder/subtractor occupies 100 CLB's and 20 CLB's respectively while multiplexer, de-multiplexer and register occupies 3au, 3 au and 5 au respectively; where 1 area unit (au) = 1 CLB has been assumed; Power dissipated per au (ps) when in standby is 4 milli-watt.). Thus, the cost in terms of power consumption is reduced from the initial solution. The scheduling solution after first iteration is shown in Figure 15. Finally the algorithm yields the solution with the minimum final cost. Experiments revealed that iteration 11 yielded the scheduling solution with minimum cost:

Iteration (11):

- i) Movement – opn 4 (2→3) ii) Movement – opn 8 (4→5), iii) Movement – opn 9 (3→4)
- iv) Movement – opn 7 (4→5)

Therefore after the PI for each operation was calculated, opn 4 had the highest PI. Hence after iteration 11(fig.16) the optimal solution to the scheduling problem was found. The cost of this

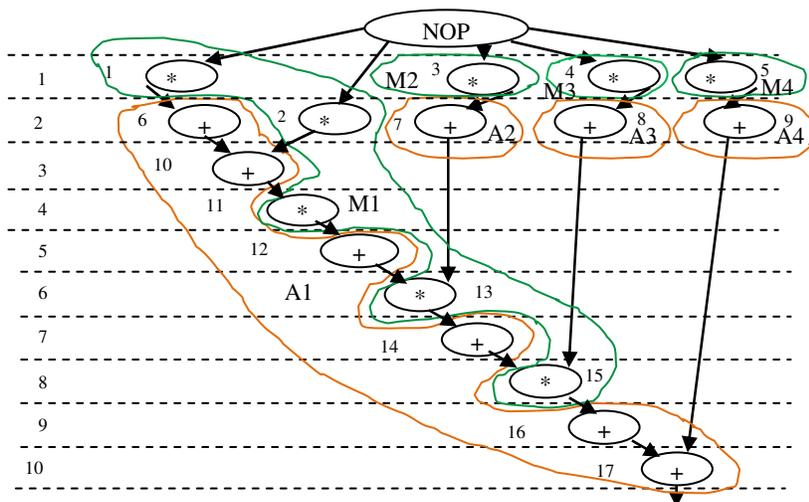


Figure15. Scheduling after 1st Iteration

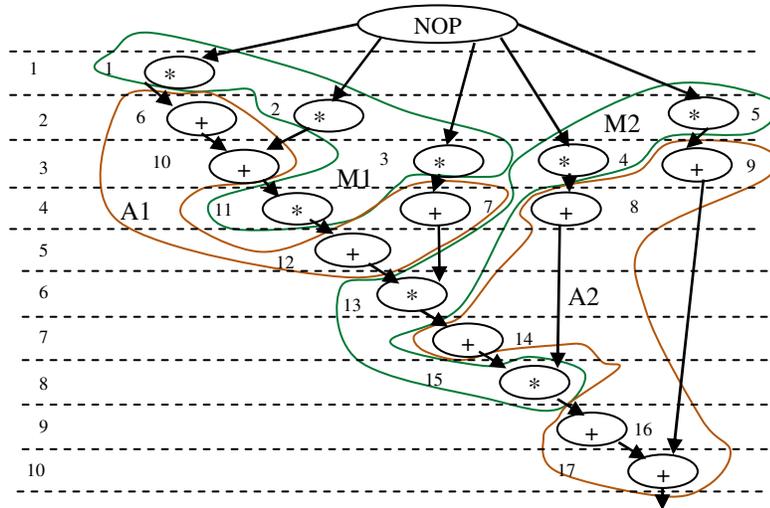


Figure 16. Scheduling solution after 11th Iteration

respective scheduling solution is $\{(2 * 100) + (2 * 20) + (8 * 3) + (4 * 3) + (14 * 5)\} * 4 = 1.38$ watts. The final reduction in cost in terms of power consumption obtained compared to the initial solution (in Figure 16) is 2.94 Watts – 1.38 Watts = 1.56 Watts. (*Note:* The cost of initial ASAP solution and schedule after iteration 11 respectively are both determined using Eq. (36)).

Note: The results of this approach applied on various benchmarks and improvements attained are reported in Chapter 7.3.

Chapter 5

A Multi Structure Genetic Algorithm for Integrated Design Space Exploration of Scheduling and Allocation in High Level Synthesis for DSP Kernels

This chapter presents the fourth algorithm/methodology of the dissertation viz. an integrated design space exploration of scheduling and allocation problem in high level synthesis using the heuristic based multi structure genetic algorithm. A cost function based on a combination of power consumption and pipelined execution time as well as a scheme to select functional unit type in case of multiple versions is proposed that can guide the genetic algorithm to global optimal or local optimal (in few cases) solution. The cost function model takes the functional units, registers, multiplexers and demultiplexers into consideration. The encoding process of the parent chromosome incorporates a special seeding process that enables the genetic algorithm to

search for an optimal solution. This type of seeding process was specifically incorporated because the optimal solution to a problem always lies between the maximum serial and parallel implementation. Therefore it is always capable of finding an optimal solution to the combined problem of scheduling and allocation based on the provided user specified constraints. Results of the comparison with another recent genetic algorithm based exploration technique indicated considerable reduction of execution clock cycle as well as power consumption for almost all the benchmarks.

For the sake of clarity, summarized below are the abbreviations that will be used in the rest of this chapter:

MSGGA: Multi Structure Genetic Algorithm

GA: Genetic Algorithm

HLS : High Level Synthesis

VLSI : Very Large Scale Integration

DFG : Data Flow Graph

CDGF : Control Data Flow Graph

CS : Control Steps

cc : clock cycle.

FU : Functional Unit

DSE : Design Space Exploration

SoC : System on Chip

ASAP : As Soon As Possible

ALAP : As Late As Possible

RASM : Resource Allocation Selection Mode

DSP : Digital Signal Processing

ARF : Auto Regressive Filter

DWT : Discrete Wavelet Transformation

EWF : Elliptic Wave Filter

WDF : Wave Digital Filter

BPF : Band Pass Filter

5.1. The Proposed Framework Using MSGA

5.1.1. The MSGA Design Space Exploration

The MSGA overview is shown in Fig.17. The input to the proposed framework is the behavioral description of the DFG that comprises data path structure, set of user specified design constraints for power and execution time (with the user specified weight factors), control parameters for the GA and module library. The module library contains four different information viz. the maximum available resources, clock cycle of each resource, hardware area of each resource, finally the details about the versions of each available functional resource. The proposed framework consists of two basic units. The first unit is the proposed heuristic that acts as an input to the skeleton for the GA. The second unit processes the information provided by the first unit to produce an optimal solution. The flow chart representation of the complete MSGA is shown in Fig.18. The proposed skeleton uses a new heuristic based on load factor criterion that assigns a specific priority for each operation in the chromosome structure. The first parent (P1) chromosome of the nodal string (this string is defined later in Section 5.1.2) is encoded based on the load factor (α) of each resource from the ASAP scheduling graph. On the contrary, each operation of the second parent (P2) nodal string is encoded based on the difference of the latency obtained by using ASAP scheduling with maximum resource (L) and the load factor (α) for each

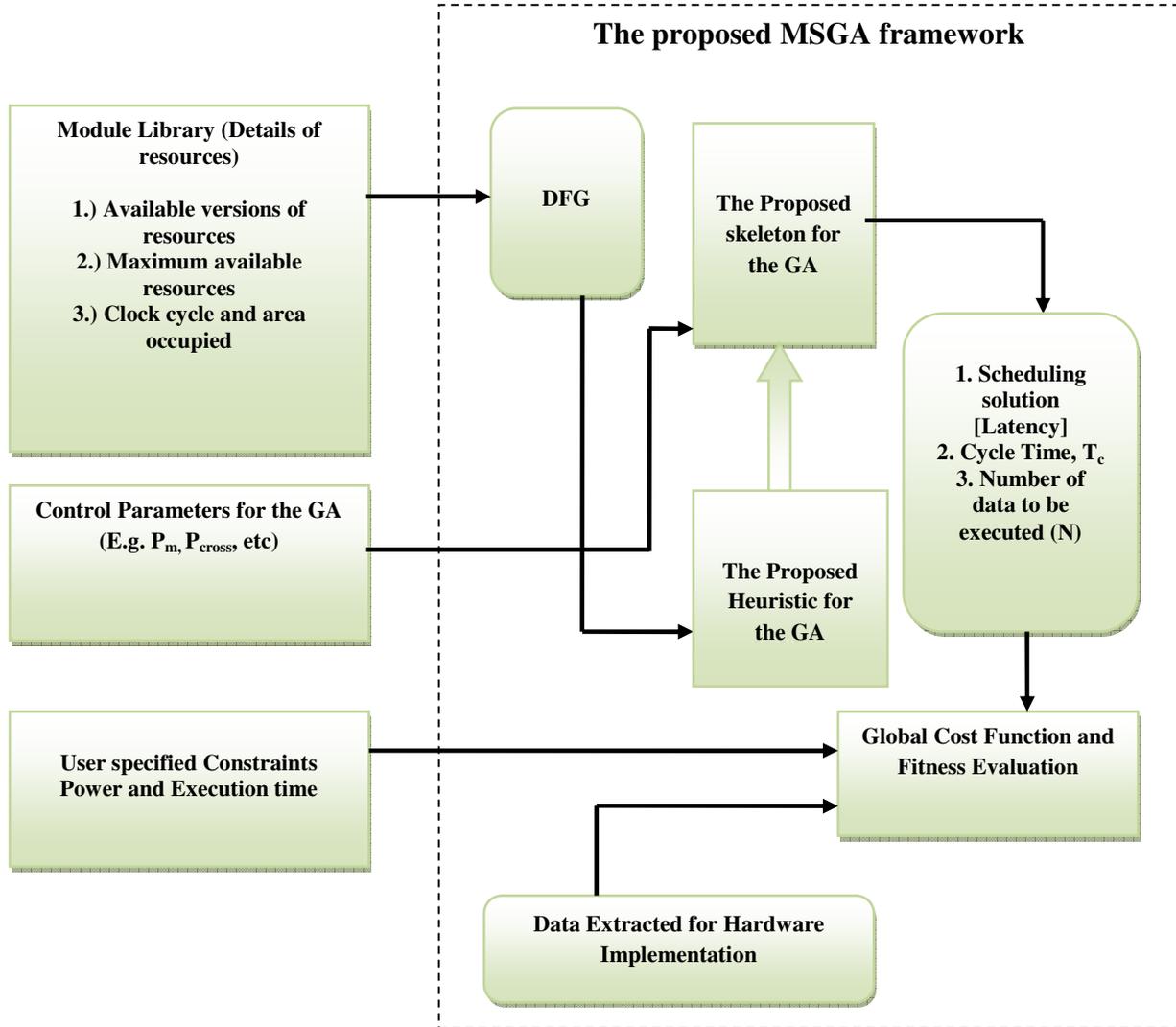


Fig.17 MSGA design space exploration approach overview

operation (o_i) obtained for P1 chromosome. The encoded value of each operation (o_i) of the second parent chromosome (P2) is calculated using Equation (37) below. Equation (37) yields integer values that will be used for encoding the genes of the nodal string of parent (P2). The main purpose behind the proposed formulation of Equation (37) is the deliberate insertion of diverse encoded gene values that could possibly capture the minimum load factor value of each operation. Equation (37) is described below:

$$\beta = L^{ASAP} - \alpha(o_i) \quad (37)$$

The rest of the parents of the population in the nodal string encoded with the load-factor values are obtained by random perturbation. The other parent chromosomes (P3....Pn) of the population obtained by the perturbation function should be individuals lying between the Parent

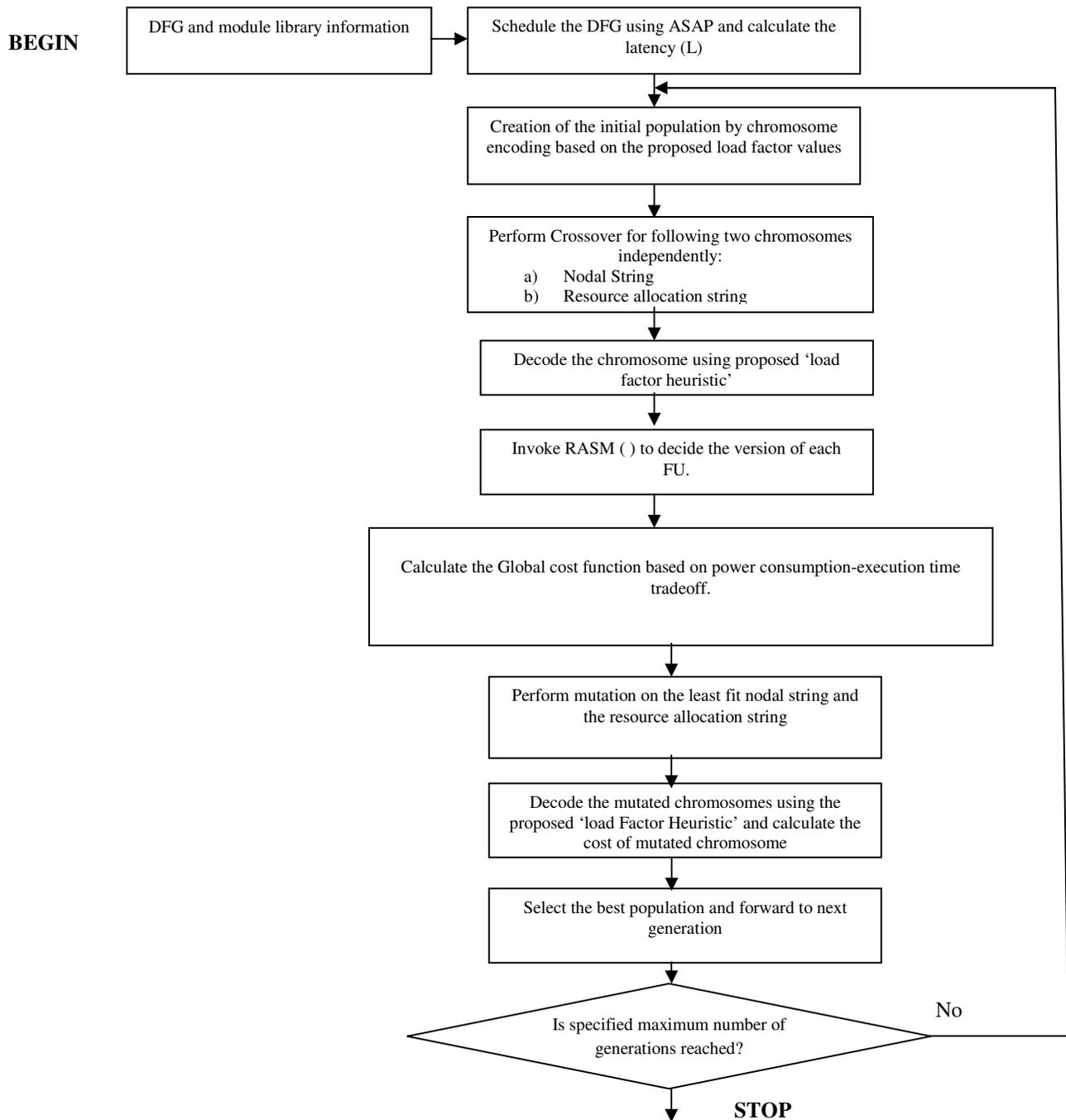


Figure18. Flow chart of the proposed MSGA

P1 derived from the schedule based on maximum resource and Parent P2 derived based on minimum resource. This is logical because the optimal solution to the integrated problem lies somewhere between the maximum and the minimum resource. The perturbation function developed which yields the load-factor values is given in equation (38):

$$PF = (\alpha + \beta) / 2 \pm \mu \quad (38)$$

where ‘ μ ’ is a random value equal or between ‘ α ’ and ‘ β ’. This function is used when encoding the values of the nodal string. On the other hand, the perturbation of the resource allocation string (this string is defined later in this chapter) for the other parents is obtained by applying the algorithm shown in Fig 19. Once the parents for the initial population are formed then direct crossover is applied. Crossover results in creation of off-springs in that generation. The next task is to decode the generated individuals of the first generation by applying a new ‘load factor heuristic’ that results in valid schedule. During the process of formation of the schedule solution, the data dependency is strictly followed before any operation is selected for scheduling. Once the valid schedules corresponding to all the specific individuals of the population are obtained, the RASM () must be selected. Since the module library contains multiple versions of the same resource, this mode selects the allocation type based on the user choice. This mode has four basic choices based on the user preference. The first choice is kept by default (Note: also by default equal weightage of area and latency is kept in the local cost function i.e. $W1 = 0.5$ and $W2 = 0.5$). This is because the first option chooses the units which have the lowest cost compared to all the available units of that kind. The cost is determined by a local cost function which is given in eqn (39) as:

$$C_L = W1 \cdot A + W2 \cdot L \quad (39)$$

Algorithm (Input: Resource allocation string; Output: new resource allocation string)

1. Randomly pick any two nodes (v_1, v_2) from the resource allocation string.
2. Randomly select any integer value (i) ranging between or equal to ' α ' and ' β ' for that specific node. Where, $\alpha \leq i \leq \beta$

Fig.19 The perturbation algorithm for resource

Where 'A' represents the area occupied by the specific type of resource and is obtained from the module library, while 'L' represents the latency of the specific type of resource. W_1 and W_2 are the user specified weightage for area and latency respectively ($0 \leq W_1 \leq 1$ and $0 \leq W_2 \leq 1$). The RASM () is composed of the following modes:

- Each operation is allocated to the FU of a particular kind with the lowest cost.
- All operations are allocated to fast FU, including operations on critical and non-critical path.
- Operations are randomly allocated to FU of that kind.
- All operations are allocated to FU with minimum area overhead of that kind.

Since RASM () selects the FU's of a specific kind for a specific operation based on the minimum cost, the next step is to retrieve the information from the module library. This information enables us to calculate the local cost function of each scheduling solution at the end of a specific generation. Calculation of the local cost function yields the FU version with the minimum cost. These low cost FU versions of each kind are then simply assigned to the specific operations of the valid scheduling solutions found before. Once the versions of the FU are assigned, the global cost function can be determined in order to judge the fitness of each individual. The least fit individual is mutated in order to hope for a better solution followed by decoding and fitness calculation. The best fit individuals from this first generation are then

forwarded to the next generation. This process continues until the maximum generation $G(\text{Max})$ specified is reached.

5.2. Description of the Proposed MSGA steps

A. Encoding of the Chromosome

The proposed approach uses independent strings to separately represent the priority of the nodes of the DFG and the resource allocation information. The approach is called multi structure because each FU (resource) is represented as an independent substring in the nodal string structure. It has two independent strings to separately represent the nodes of the DFG (called ‘nodal string’) and the resource allocation information (called ‘resource allocation string’). The ‘nodal string’ contains the load-factor values of each node which will determine the priority of the nodes during scheduling. The ‘load factor heuristic’ is used when decoding the nodal string in order to obtain a valid scheduling solution. The ‘resource allocation string’ contains list of integers which indicate the maximum number of resources allowed during scheduling. The encoding scheme for the ‘nodal string’ and the ‘resource allocation string’ is shown with an

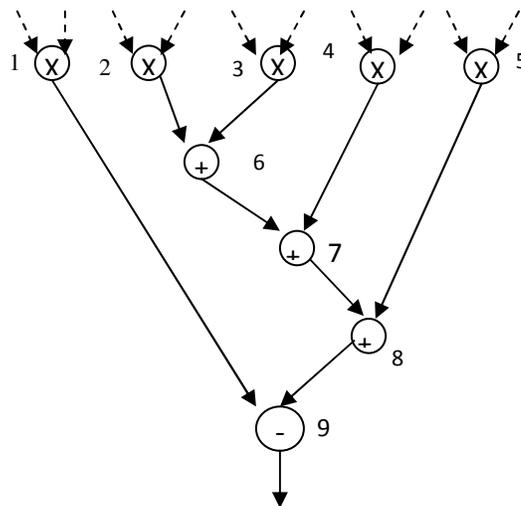


Figure20. DFG of the IIR Digital Filter

example of a benchmark ‘IIR Digital Filter’. Fig. 20 shows the DFG of the above benchmark. The schedule of the DFG of the differential equation solver using ASAP is shown in Fig.21. The latency (L) obtained is 12cc. (Note: Arbitrarily assuming multipliers and adders/subtractor takes 4cc and 2cc respectively). The corresponding chromosome encoding for the first parent (P1) of the nodal string is shown in Fig. 22. The total load-factor of each operation (node) is obtained by summing the load-factor of the successor operations following that node. E.g. for node 2, the load factor is $(4+2+2+2+2) \text{ cc} = 12\text{cc}$. The second parent (P2) chromosome is encoded based on the load-factor values obtained using equation (37). The second parent (P2) chromosome encoding is shown in Fig. 23. The rest of the parents of the initial population are obtained using equation (38) which is a perturbation function used to encode the load –factor values. The load factor values for the rest of the parents always lie between the values from the first and second parent. This scheme has been developed because the optimal solution to the problem should always lie between the serial and maximally parallel implementation [22]. On the other hand, the first parent (P1) shown in Fig. 22b and second parent (P2) of the resource allocation string shown in Fig. 23b are based on the user specified maximum and minimum resources respectively. The user specified maximum resource assumed here acts as the maximum resource constraint for the application. Thus, from Fig.22, the resource constraint for the number of multipliers, adders and subtractor are 4, 3 and 2 respectively. The assumed user specified constraint on number of clock oscillator is three (The details of the resources are provided later in the module library shown in Table 5). The rest of the parents (P3...P8) are obtained using Equation 38 and Figure19. The nodal string and the resource allocation string for rest of the parents are shown in Fig.24 respectively. For example, in case of Fig 24b, the encoding of the third parent for the resource allocation string is obtained first picking up randomly any two nodes

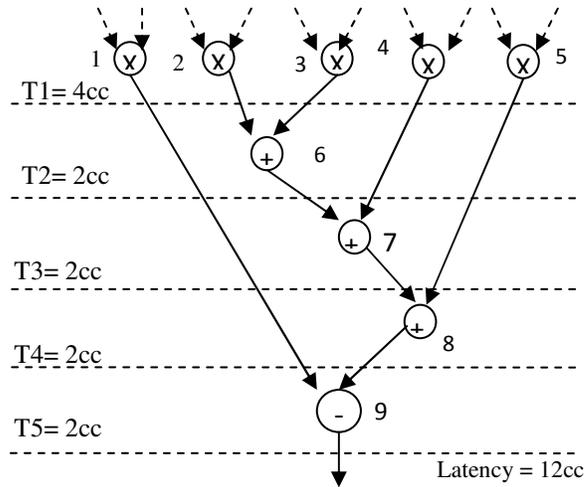


Fig.21. Scheduling of the IIR Digital Filter using ASAP

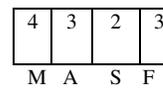
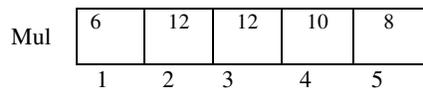
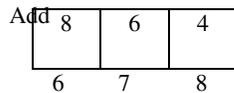
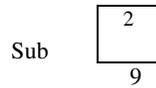


Fig.22a. Nodal String

Fig.22b. Resource allocation String (Max resources)

Fig.22. Chromosome Encoding for the first parent (P1)

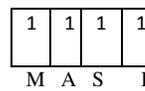
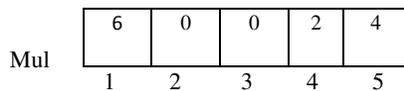
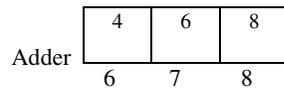
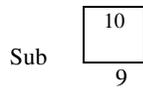


Fig.23a. Nodal String

Fig.23b. Resource allocation String (Min resources)

Fig.23. Chromosome Encoding for the second parent (P2)

M (multiplier) & A (adder) and then randomly selecting any integer value between '4' and '1' for M & between '3' and '1' for A. The randomly selected value for both M & A is '2'.

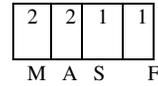
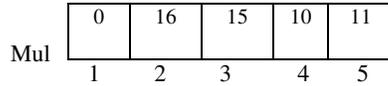
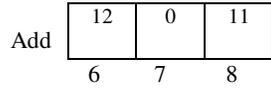
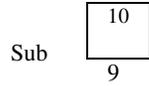


Fig.24a. Nodal String

Fig.24b. Resource allocation String

Fig.24. Chromosome Encoding for the third parent

Similarly, the rest of the parent chromosomes can be built by perturbation.

B. Crossover Scheme

The parents for crossing are selected by binary tournament selection method [29]. Proposed here is an independent direct crossover of the two independent strings viz. *nodal string* and *resource allocation string* to produce separate off-springs for each with a very high crossover probability ($P_{\text{cross}} = 1.0$). (Note: This value is similar to the values considered in other GA-based approaches such as in [11].) The direct crossover is applied to each sub structure of the nodal string structure. Since the nodal string encodes the load factor of each operation for a particular FU, the crossover does not damage the precedence relationship. This is because the nodal string in the proposed approach does not encode the topologically sorted permuted list of operations (nodes). Further, during the final decoding process, the *load factor heuristic* is followed which always produces feasible solutions. The detailed explanation on the decoding process is provided in Section 5.2.D.

B.1 Multi-Point Crossover of the Nodal String

Before the crossover scheme can be applied to the nodal strings, the two parents are randomly divided into two halves at point n (where ‘ n ’ represents a random cut point in the

nodal string). The proposed crossover is called multi-point because each substring of the nodal string representing independent FUs is divided at a different point. For example, applying the direct crossover operator to the nodal string between the first (Fig. 22a) and second parent (Fig.23a) at point 2 for multiplier and point 1 for adder and subtractor, yields offspring 1 and offspring 2 respectively. The offspring 1 inherits all the properties of the first half from the first parent. While the second half is inherited from the second parent. The properties that are inherited from the parents are the load factor values and corresponding node numbers (operations). The offspring 1 obtained after crossover operation between P1 and P2 is shown in Fig 25 while offspring 2 obtained after crossover between P2 and P1 is shown in Fig. 26. For the sake of brevity, the remaining off-spring have been omitted in the chapter.

B.2 Crossover of the Resource Allocation String

The resource allocation string is responsible for encoding the number of hardware functional units of each type available for scheduling operations in each time step. Since the number of allocated functional units of each type is totally independent of each other, the n-point crossover

can be easily applied. For instance, in case of

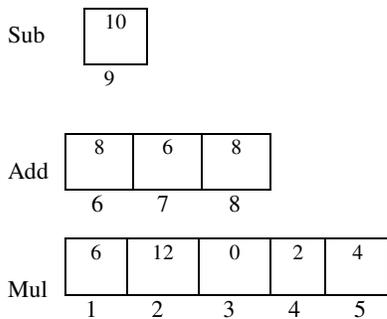


Fig.25. Offspring 1 obtained for the nodal string obtained after crossover between P1 and P2

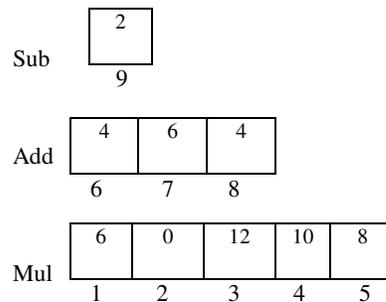


Fig.26. Offspring 2 obtained for the nodal string obtained after crossover between P2 and P1

the DFG for Digital IIR filter benchmark, the two parents (P1 and P2) for the resource allocation string are shown in Fig 22b and 23b respectively. P1 represents a solution with four multipliers, three adders, two subtractors, one comparator and third frequency oscillator while P2 represents a solution with one multiplier, one adder, one subtractor, one comparator and first frequency oscillator. Application of the direct crossover at a random cut point between P1 and P2 yields offspring 1 while between P2 and P1 yields offspring 2 in fig 27.

C. Mutation Operation

Mutation is performed on the least fit nodal string chromosome and the resource allocation string chromosome with probability, $P_m = 0.25$. $P_m = 0.25$ is the minimum number of time mutation is performed in each generation. Since any mutation probability between 0.20 to 0.25 is considered standard, refining the value slightly does not alter the final solution.

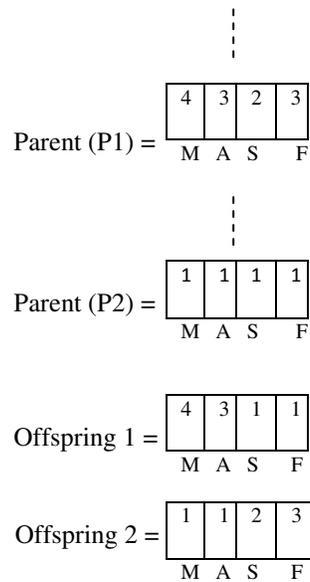


Fig.27 Offspring 1 and offspring 2

C.1 Mutation operator of the Nodal String

The mutation operator for the nodal string and resource allocation string is invoked independent of each other. The mutation algorithm for the nodal string is shown in Fig 28. According to the algorithm, any two nodes (v_i, v_j) in the string (k) are randomly selected for mutation. Next the load factor values of the two selected nodes are swapped. For example, let the load factor value for the two nodes (v_i) and node (v_2) selected be 'L1' and 'L2' respectively. Therefore, after mutation the new load factor values for node (v_i) is 'L2' and node (v_j) is 'L1'. This mutation technique drastically alters the load factor values of nodes which act as the priority indicator to select the operations for scheduling. Further, since the mutation scheme of nodal string alters just the load factor values of the nodes and does not displace the actual nodes themselves, this kind of mutation scheme always produces valid solutions.

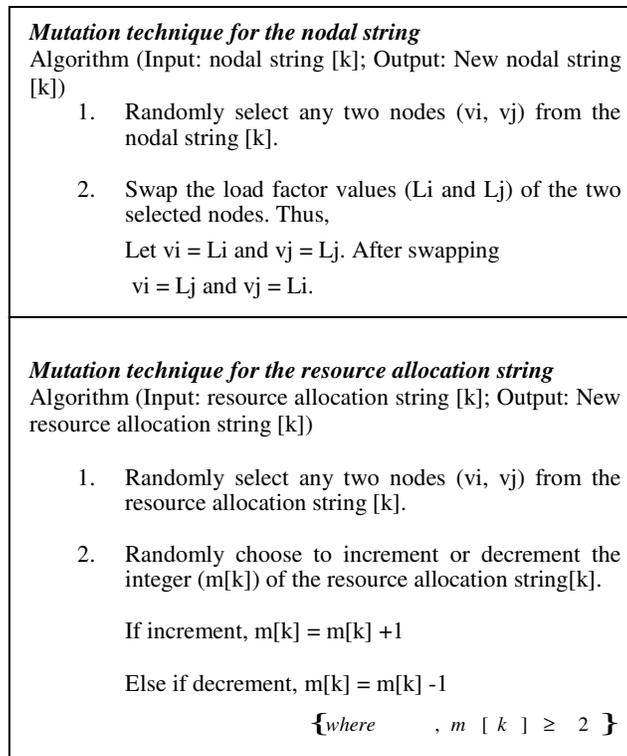


Fig.28. Mutation algorithm for the proposed approach

C.2 Mutation operator of the resource allocation String

The mutation algorithm for the resource allocation string is shown in Fig.28. The concept of the proposed mutation scheme has been adopted from [11]. All possible combinations of the functional units that can be allocated during a scheduling solution can be attained through this mutation technique. The algorithm first randomly selects any two nodes (v_i, v_j) and then randomly increments or decrements the value of the integer (m) which represents a particular type of FU. (Note: the integer is only randomly incremented or decremented if the minimum value of that FU is 2. This is because if the integer (m) is less than 2, then decrementing the integer will result in invalid number of FU of a specific type). (Note: Since the mutation scheme of the resource allocation string only alters the integer value of each resource, application of this mutation technique does not disobey any precedence relationship present in the DFG).

D. Decoding Process (Determination of a Valid Schedule)

The decoding of chromosome always results in a valid scheduling solution which strictly obeys the data dependency present between the operations. For the decoding process, a ‘load factor heuristic’ is proposed. The load factor heuristic is shown in Fig. 29. For example, in case of offspring 1, the nodal string and the resource allocation string are shown in Fig.25 and Fig.27 respectively. The resource allocation string of the offspring1 represents an allocation solution containing four multipliers, three adders, one subtractor, one comparator and first frequency oscillator. On the other hand, the priority of each operation for a particular type of FU is indicated by the load factor values in the nodal string (fig.25). Therefore, for the DFG shown in Fig.20, the scheduling solution of offspring 1 is shown in Fig. 30. The resulting solution is a valid schedule, allocation and binding obtained for offspring 1. The solution provides an

integrated solution to the concurrent problem of scheduling and allocation. The version (vn) of the each type of resources used is clearly indicated beside each node. As mentioned before, the RASM () is by default kept at first option. The FU was chosen based on the minimum cost obtained among the available FU choices, by using the local cost function (in equation (39)). The information and details about each FU is obtained from the module library during the local cost calculation process. A portion of the module library is shown in Table 5. The data extracted for the hardware implementation and global cost calculation from the integrated solution of offspring1 (Fig.30) is described in Table 6.

Load Factor Heuristic: Building a priority order of nodes in list L[k] based on load factor of each nodal string [k] of operation type and assigning the nodes of FU type in CS.

Algorithm (Inputs: DFG, the nodal string [k], FU type; **Output:** Scheduled DFG).

Step 1) CS = 1

Repeat

{

Step 2) Select independent nodes from list L[k], to assign in the current CS, based on two conditions: i) Number of FUs for each operation type is available ii) The nodes are not scheduled in the CS.

- a. **If** there is a tie during the selection **then** select the nodes with the higher load factor values.
- b. **If** there is still tie during the selection **then** randomly select the nodes as desired.

Step 3) Select dependent nodes with higher load factor value from list L[k] to assign in the current CS, based on two conditions: i) Number of FUs for each operation type is available ii) The parents of the dependent node are scheduled.

- a. **If** there is a tie between the load factors during selection **then** select the nodes with higher utilization factor
- b. **If** there is still a tie between the utilization factors during selection **then** randomly select the nodes as desired.

}

Step 4) C.S = C.S + 1, **Until** L[k] = Null

(Note: Utilization Factor refers to a metric which indicates the number of children of node v_i [k]. So if two nodes of operation $o(i)$ have the same load factor precedence is given to the node with more children)

Fig.29 The proposed load-factor heuristic for the MSGA Framework

Table5. Portion of Module Library for MSGA

(Note: The clock frequency shown in Table V has been chosen arbitrarily for demonstration of the proposed approach for MSGA. Clock frequency of other type could also have been included in the module library for MSGA)

Resources (FU)	FU type		Area (area units in a.u.)	Delay (cc)	Power consumption per area unit (au) @ 50 MHz (p _c) in milli-Watts (mW)	Power consumption per area unit (au) @ 100 MHz (p _c) in milli-Watts (mW)	Power consumption per area unit (au) @ 200 MHz (p _c) in milli-Watts (mW)
	#	Versions					
3 Adders (+)	1	V1	50	1	3	6	12
	1	V2	30	2			
	1	V3	15	3			
4 Multipliers (*)	1	V1	80	3			
	3	V2	50	4			
	#	Versions					
2 Subtractors (-)	1	V1	30	2			
	1	V2	15	3			
	#	Versions					

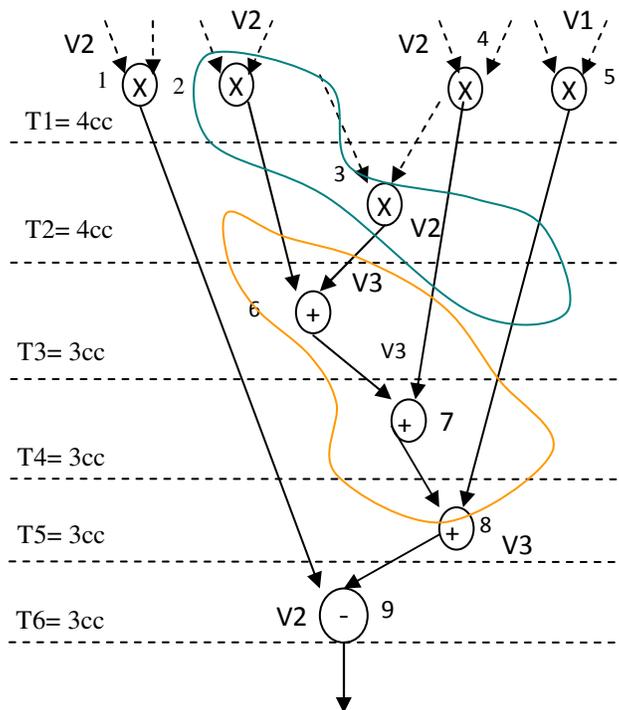


Fig.30. Integrated solution to offspring 1 (Decoding of the chromosome) for the IIR Digital Filter benchmark

Table6. Data Extracted from the integrated solution of offspring 1

#	FU Type
3	Multiplier (version V2)
1	Multiplier (version V1)
1	Subtractor (version V2)
1	Adder (version V3)
1	2 to 1 MUX for 1 st multiplier (V2)
1	3 to 1 MUX for adder (V3)
1	1 to 2 DEMUX for 1 st multiplier (V2)
1	1 to 3 DEMUX for adder (V3)
Number of Registers allocated using left edge algorithm [29].	

E. Global Cost Function and Fitness Evaluation Methodology

The objective of the proposed approach is to simultaneously reduce the total execution time required for execution of a specific set of data as well as the total power consumption expended. All of the previous approaches have only considered latency and hardware area as a design constraint such as [11] [20] [21] [22] and not total execution time which considers the latency, cycle time and also the number of sets of data to be executed. Thereby, the methods are not able to reduce the total execution time as they do not consider data pipelining. In the presented approach a sophisticated global cost function has been developed that considers the total execution time taking data pipelining as well as total power consumed into account.

The cost is calculated using the global cost function after each chromosome is decoded to obtain a new integrated solution. The decoding process strictly follows the ‘load factor heuristic’ and hence always results in a feasible solution. The global cost function (C_G) developed which considers total execution time and power expenditure/hardware area is shown in equation (40).

$$C_G = W1 \cdot \frac{T_{EXE} - T_{CONS}}{T_{MAX}} + W2 \cdot \frac{P_T - P_{CONS}}{P_{MAX}} \quad (40)$$

Where ‘P’ is the total dynamic power as a function of operating frequency and number of devices switching due to frequency of operation based on [5, 64, 12]:

$$P = \sum_{i=1}^n (N_{Ri} \cdot K_{Ri}) \cdot p_c \quad (41)$$

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

Where T_{EXE} = Total execution time in clock cycles taken for execution of the given ‘N’ sets of data is calculated using the function from Chapter 2 given in equation (42):

$$T_{EXE} = L + (N - 1) \cdot T_C \quad (42)$$

Based on [61,62, 12], the respective execution time in micro-seconds (assuming frequency in Mega Hertz) can be expressed as:

$$T_{EXE} = \{L + (N - 1) \cdot T_C\} \cdot T_P \quad (43)$$

L = Latency of the scheduling solution. N = Number of sets of data to be executed.

T_P = Time period of the clock frequency oscillator; T_C = Cycle time of the scheduling solution.

(Note: The cycle time is the difference in clock cycles between any consecutive outputs of pipelined data instances. The cycle time difference during data pipelining is the result of initiation interval (which is the difference in clock cycles between any consecutive inputs instances of pipelined data sets). For example, the cycle time calculation for the integrated solution (Fig. 30) is shown in Fig.31 below. The output for first set of data is arriving after 20cc while the output for second instance of data after 28cc. Thus, due to exploring initiation interval

of 8cc during pipelining, there is a cycle time difference of 8 cc, which is the result of considering the initiation interval. As seen clearly in Fig 31, the cycle time consideration in equation (42) has resulted from genuine data pipelining. Therefore the option of cycle time during pipelining has been also taken into account during the exploration process.)

C_G = Global Cost of the integrated solution

T_{CONS} = Execution time constraint specified by user.

T_{MAX} = Max execution time taken by a solution during the specific generation (G).

A_{FU} = Total area of the functional units.

A_{REG} = Total area of registers.

A_{MUX} = Total area of the multiplexer used during implementation.

A_{DEMUX} = Total area of the demultiplexers used during implementation.

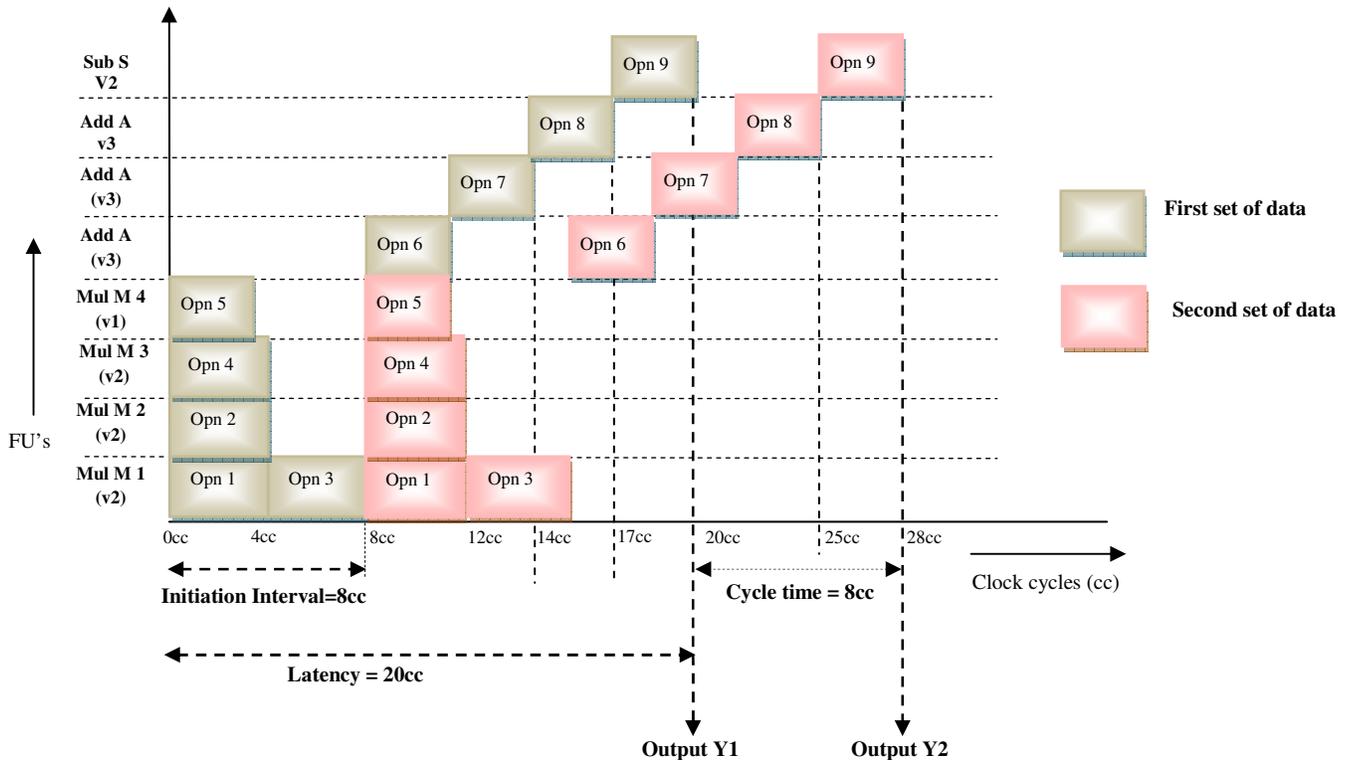


Fig.31.Cycle time calculation during data pipelining for offspring 1

P_{CONS} = Power consumption constraint specified by the user.

P_{MAX} = Max power consumption of a solution during a specific generation (G).

p_c = Power consumed at a particular frequency of operation.

W1 and W2 = User specified preference of the constraints.

The cost function requires input from various sources to evaluate the fitness of each solution found. The sources consists of the a) module library information, b) data extracted for the hardware implementation (e.g. as shown in Table 6), c) DFG and d) scheduling solution found after decoding the chromosome (latency), number of sets of data, cycle time together, for calculation of the execution time.

F. Termination Criterion for the MSGA

The maximum generation has been kept constant for each benchmark run. Although making the number of generations proportional to the problem size is more logical, settling on an average number of maximum generations for both small and large size benchmarks is a good compromise. Therefore, experiments dictated that retaining the maximum generation G(Max) to 100 is an optimal compromise between reasonable runtime and achieving high quality solution. Setting the G(Max) to some higher value will still produce the same solution but will also increase the computation time. On the other hand, setting G(Max) to a lower value will yield inferior solutions particularly for large benchmarks.

Note: The results of this approach applied on various benchmarks and its improvements are reported in Chapter 7.4

Chapter 6

Fast Multi-Objective Exploration and RTL Circuit Generation using Architecture Synthesis Platform: Exploration Synthesizer

A novel efficient architecture synthesis platform capable of performing extremely fast multi objective design space exploration of an optimal solution as well as register transfer level circuit generation of DSP applications is proposed in this chapter which is a combination of algorithms described in Chapters 2 and 3. Design space exploration based on trio parametric objectives can be performed through this platform due to the determination of border variant for each parameter followed by the formation of the Pareto optimal set obtained by the intersection of all satisfying variants. Currently, the experimental results reported here are based on three parametric objectives viz. hardware area, pipelined execution time (or performance) and power

consumption. The proposed tool is an hybrid combination of multi-objective design space exploration approach and fuzzy search heuristics (Chapters 3 and 4) based on priority factor pareto optimal method and fuzzy search heuristics, which support features such as functional pipelining. Due to lack of availability of a standard format for intermediate representation of the data flow graph, a custom intermediate format called ‘application library’ is developed. The key benefit of this platform is the rapid exploration/optimization time (in order of a few secs to mins) to find the optimal solution regardless of the complexity of the design space as well as generation of the complete processor schematic in a short time. This is followed by VHDL generation of centralized control unit and data path circuit. The results of the comparison with a current heuristic approach indicate 99% improvements in exploration time with ability to yield an optimal solution in almost all cases.

6.1 The Proposed Exploration Synthesizer Design Flow

Using the proposed Exploration Synthesizer tool, multi-objective architecture exploration is performed by a combination of the exploration techniques described in Chapters 2 and 3 (For details please see the exploration flow illustrated in Chapters 2 and 3). Hence, the proposed technology is a combination of novel hybridization based on the priority factor Pareto analysis and a hybrid fuzzy algorithm. Priority factor Pareto analysis (Chapter 2) is employed when user preference is power-performance (pipelined execution time) tradeoff with hardware area minimization, while the hybrid fuzzy algorithm (Chapter 3) is employed when the user requirement is hardware area-performance (pipelined execution time) tradeoff with power consumption minimization. It should be noted that the design space is extensive and consists of innumerable alternative combinations of adders-subtractors, multipliers and frequency

oscillators. The overview of the proposed approach of the tool including the multi-objective architecture exploration phase and its RTL circuit generation phase comprised of the centralized controller and data path unit is shown in Fig.32. After the exploration phase is complete the RTL circuit is generated based on the high level synthesis design flow described in [27]. The high level synthesis design flow described in [27] consists of various design stages such as architecture selection (or exploration) followed by scheduling, allocation, binding, determination

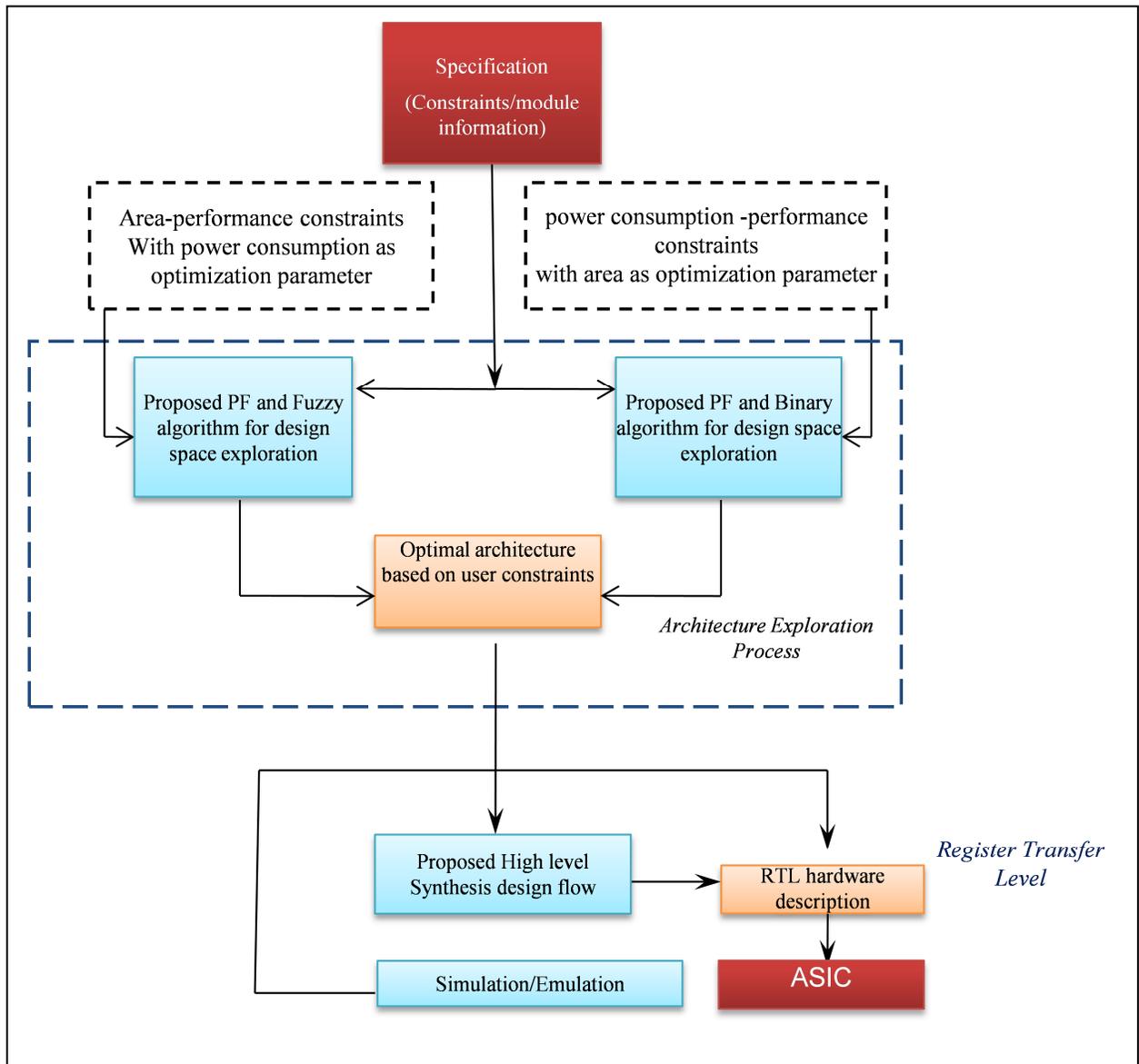


Figure32. Design Flow of the proposed Exploration Synthesizer platform

of interconnect units, development of block diagram and finally development of control structure. The exploration solution in the form of an architecture configuration is fed into an integrated scheduling algorithm that performs scheduling, allocation and binding steps, which is then subjected to an algorithm which determines the interconnect units, storage units etc and finally develops the schematic of the complete system.

Therefore, the RTL circuit generated by the tool (schematic in .giff format) is based on the high level design flow mentioned in the previous paragraph. In addition to generation of the complete schematic of the processor circuit, the tool also generates the hardware description (VHDL) of the controller in the form of a finite state machine and the data path in the form of components and port mapping details. The schematic of the circuit can be easily emulated in any commercially available logic synthesis tool such as Xilinx ISE or Altera Quartus for simulation and FPGA implementation. The VHDL obtained can be directly synthesized on the mentioned synthesis tools for generating the bit streams for FPGA implementation. The design space exploration process used in the tool describes that the random design space consisting of innumerable design alternatives is first hierarchically arranged in increasing or decreasing order (strictly or partially) using the priority factor metric and priority order logistics. Once the design space is sorted, the fuzzy logistic searching or binary searching [63] is applied to obtain the border variant of the specific optimization parameter (Refer to Chapters 2 and 3 for more details). The groups of all satisfying and non-satisfying variants are obtained from this step. Similar methodology is applied for the other parameters to obtain their satisfying and non-satisfying variants. All satisfying sets of variants are intersected to find the common set of variants. The set obtained consists of the variants which concurrently satisfy all operating constraints specified by the user. Finally, the obtained Pareto optimal set is again sorted for the

final optimization parameter using priority factor metric and priority order logistics to find the variant with the minimum value with respect to the optimization parameter. (Note: For all mathematical proofs, discussion and demonstration on the DSP kernels please refer to Chapters 2 and 3).

6.2 Keystones of the Proposed Design Automation Platform: Exploration

Synthesizer

1. The capability of ‘trio’ parametric optimization (including area, power and pipelined performance) provides the foundation for extremely rapid design space exploration (in seconds) regardless of the size of the design space during ASP design.
2. Efficient design space exploration enables additional selection of the frequency oscillator from the extensive design space besides the optimal architecture for the system design.
3. Development of a hybrid multi objective optimization flow for Register Transfer Level (RTL) design for generation of a RTL circuit consisting of a controller description and data path in a short time.
4. The platform also provides an added advantage to the designer to auto-correct the user specification if the constraints do not obey the upper threshold and lower threshold limit.

A synopsis of the proposed tool with its functionality for research and industry use is as follows [36- 40]:

- i) Architectural Exploration: Allows selection of the best architecture among the possible implementations.

ii) Automated Decision making capability for circuit generation: In the absence of the register transfer level design team, with high level synthesis systems companies can do high-level exploration to determine what will effectively run their applications.

iii) Shorter design time: Reduction in exploration runtime helps faster design time and generation of design with fewer errors and redesign.

iv) IP reuse and implementing algorithms in FPGA hardware: IP created by high level synthesis can be shared between companies.

The features supported by the proposed software platform include functional pipelining considering initiation interval.

The design automation platform/tool works as a 3 step process. The first step of the tool accepts as input the user specified module library, the application data flow graph in a custom intermediate format specified by the user, and the number of data elements to be pipelined. The second stage accepts as input the user specified constraint and optimization requirements for three parameters. Finally, the last stage of the tool produces multiple outputs: a) exploration result b) schematic of the complete system c) VHDL of the complete system. The snapshot of the developed design automation platform for a sample problem is shown in the appendix.

6.3 Input Format and Intermediate Representation

A customized intermediate representation of the DFG has been developed since there is no standard input format and intermediate representation of the DFG after the architecture synthesis design process. This intermediate representation of the application DFG serves as the input format for the proposed Exploration Synthesizer tool. The input format of the proposed tool is termed 'application library'. An explanation of the 'application library' is provided with the aid

of a Discrete Wavelet Transformation (DWT) DSP benchmark shown in Fig. 33 and 34 respectively. Every operator is denoted by four tuples, viz. operation, input 1, input 2 and output name. Hence, the operators are identified by $oi = (\text{operation}, ip1, ip2, op)$. The levels of the

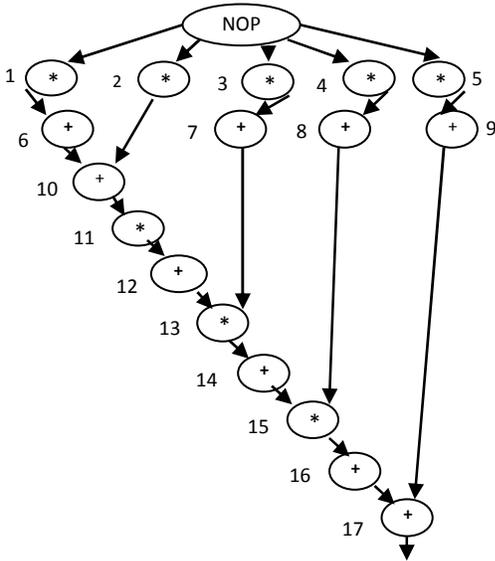


Figure33. DFG of the DWT Benchmark

```

*,I1,I2,1,*,I3,I4,2,*,I5,I6,3,*,I7,I8,4,*,I9,I10,5
+,1,1,6,+,3,3,7,+,4,4,8,+,5,5,9
+, 2,6,10
*, 10, 10, 11
+, 11, 11, 12
*, 7, 12, 13
+, 13, 13, 14
*, 8, 14, 15
+, 15, 15, 16
+, 9, 16, 17

```

Fig.34. Application library: The intermediate representation of DWT DFG which acts as the input format for the tool

operators in this custom format are obtained straight from As Soon As Possible (ASAP) schedule. Hence, opn 1, opn 2, opn 3, opn 4 and opn 5 are assigned level 1, opn 6, opn 7, opn 8 and opn 9 are assigned level 2 and so on. This rule is followed to obtain the application library format.

6.4 Output Details of the Tool

This final phase of the platform produces all the required results in a very short runtime which can be easily downloaded into the local machine such as:

1) Exploration solution result: comprising of the design architecture (Number of FU's and type of clock oscillator frequency) which meets all the user specs and constraints which is used to design the system. The execution time, estimated power consumed and hardware area is also

indicated which confirms that the solution produced meets in almost all cases the user constraints.

2) Custom processor schematic (.gif diagram): This can be used to imitate design of the application in commercial digital synthesis tools (Xilinx ISE, Altera Quartus etc).

3) VHDL (complete HDL description) of the custom processor of the application kernel (consisting of the controller description and data path circuit): This can be used to directly into commercial digital logic synthesis tools (Xilinx ISE, Altera Quartus etc) and Layout Synthesis CAD tools such as Synopsys Design Analyzer for further simulation or chip core designing in cadence encounter.

4) Module library and application input can also be downloaded.

5) Summary of the provided user constraints values is also indicated.

Note: A sample custom processor schematic and a portion of its VHDL description for Elliptic Wave Filter DSP benchmark with some arbitrary constraints generated by the tool are shown in the Appendix.

Chapter 7

Implementation, Results and Analysis

This chapter describes the complete experimental results of the four proposed algorithms/methodologies which resolves four branches of the design space exploration problem:

a) Design Space Exploration methodology for Power-Performance-Cost/Area tradeoff in High Level Synthesis using novel Priority Factor approach b) Design Space Exploration methodology for Hardware Area-Performance-power tradeoff in High Level Synthesis using Hybrid Fuzzified approach c) Methodology for Integrated exploration of Scheduling and Module Allocation in High Level Synthesis for static power optimization under minimum control step based on Power Gradient theory d) Methodology for Integrated Exploration of Scheduling and Module Allocation in High Level Synthesis for Power-Performance tradeoff using Heuristic Genetic

Algorithm which includes the implementation details, library details and improvements attained compared to the state of the art approaches.

7.1 Experimental Results: Proposed approach ‘Rapid Design Space Exploration in High Level Synthesis Based on Power-Performance-Area Tradeoff using Priority Factor Metric’ and Comparison with recent approach

This section describes the experimental results of the proposed approach based on power-performance-area/cost trade-off (elaborated in Chapter 2) and the improvements obtained compared to recent approach [11]. The proposed hybrid DSE approach has been implemented in Java language on AMD Athlon 64 X2 Dual-Core Processor TK-157 with 3072 MB DDR2 memory, 64KB L1D cache and 256 KB L2 cache memory. The processor frequency is 1.90 GHz. For a qualitative analysis, the proposed hybrid approach was tested on a number of DSP benchmarks ranging from small to large number of variants. Many large size benchmarks were selected for the experiment. The DSP benchmarks such as discrete wavelet transformation (DWT) [30], autoregressive filter (ARF) [31], and MPEG motion vectors (MMV) [32] were also adopted for experiments.

For determination of the optimal architecture, design space exploration requires elaborate analysis and evaluation of the architectural variants (design points). Before selecting the optimal architecture, the border variant of architecture for both the performance (execution time and area/power) parameters needs to be found separately. Binary search conducted on the arranged design space (increasing or decreasing) leads to the border variant, taking into account the operating constraints for execution time and area/power separately. The proposed DSE approach uses binary search after the arrangement of the design space using the priority factor method. The

search of the optimal architecture requires only $\log_2 \prod_{i=1}^n v_{Ri}$. Where 'n' = number of type of

resources and 'v_{Ri}' is the number of variants of resource 'Ri'. On the contrary, the exhaustive

search checks for $\prod_{i=1}^n v_{Ri}$ architectural variants during optimal architecture selection while

satisfying all operating constraints. In this design space exploration approach and in the design

flow three performance parameters have been used for optimization. The execution time and

power are the parametric constraints and area is the optimization parameter. Hence, the searching

has to be repeated for both the parameters to determine the border variant. Therefore the total

number of architecture evaluations using exhaustive search is given as: $M * \prod_{i=1}^n v_{Ri}$ And total

number of architecture evaluations using the proposed method is given as: $M * \log_2 \prod_{i=1}^n v_{Ri}$. Here,

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

two performance parametric constraints. The proposed approach was applied on various realistic

benchmarks to check the acceleration obtained through this DSE method. Results indicated

massive improvement in exploration time compared to the exhaustive approach. A sample

module library for this approach must consist of the following information:

- a) Type of functional units (FU)
- b) Number of FU's of each type (e.g. number of adder/sub resources)
- c) Area occupied by each FU (e.g. area in FPGA slices obtained through characterization during synthesis in CAD tool)
- d) Number of clock cycles for each FU needed to finish an operation,
- e) Number of clock frequency oscillator available for selection
- f) The power consumed per au/power consumed by each FU type.

The results of proposed design space exploration framework for the standard benchmarks and the improvements obtained with recent approach [11] are illustrated in Table 7. The percentage improvements in exploration time with respect to [11] as shown in Table 7 are obtained as follows: $[(\text{Exploration time}^{[11]} - \text{Exploration time}^{\text{Proposed}}) / \text{Exploration time}^{[11]}] * 100$. For example, in the case of Discrete Wavelet Transformation with user provided module library information, the proposed approach explores the architecture in 250 milli-secs while recent approach [11] in 3.40 secs. This provides an improvement in exploration time of $[(3.40 \text{ secs} - 0.25 \text{ secs}) / 3.40 \text{ secs}] * 100 = 92.64\%$ compared to the existing approach [11]. Similar improvements in exploration time are noted in case of other different size benchmarks such as Fast Fourier Transform and MPEG Motion Vectors. Additionally, the speedup is obtained as follows: $\text{Exploration time}^{[11]} / \text{Exploration time}^{\text{Proposed}}$. For example, for DWT benchmark, the speedup is: $3.40 \text{ secs} / 0.25 \text{ secs} = \sim 14 \text{ times}$. Further, as evident by the experimental results shown Table 7, the final solution was global optimal in nature for all the benchmarks because it meets both the user specified arbitrary constraints values for power and for execution time. For example, in case of DSW benchmark based on the specified time constraint of 400us and power constraint of 6W, the proposed PF approach finds the final solution which is optimal in nature. The final solution has execution time constraint value is 340.3us and power value of 2.74W. To the best of the author's knowledge, there are no other works except [12] that consider power and pipelined execution time as constraint and area as optimization criteria during the exploration process. A comparison of final results with [12] is not reported because both approaches find the same quality of solution through the exploration process. However, the proposed approach is able to achieve significant acceleration compared to [12]. Hence based on the experiments performed on the benchmarks it can be concluded that the proposed approach for DSE is able to

Table7. Experimental results of comparison between proposed PF approach and recent GA approach [11]

Benchmark	Execution time constraint (us)	Power consumption constraint (Watts)	Final Resource	Pipelined Execution time (us)	Power consumption (Static and Dynamic) and Hardware Area (CLB slices)	Exploration Runtime of proposed approach	GA based DSE [11] Approach G(Max) = 100	Improvement and speedup in Exploration compared to [11]
Discrete Wavelet Transformation (DWT)	400us	6W	ADD:2, MUL: 2, OSC: 50MHz	340.32us	Power: 2.74W Area: 254 CLB slices	250milli-sec	3.40 secs	Improvement = 92.64 %
								Speedup = 14 times
Band Pass Filter (BPF)	800us	6W	ADD:2, MUL: 2, OSC: 50MHz	640.12us	Power: 3.02W Area: 356 CLB slices	140milli-sec	10.19 secs	Improvement =98.26 %
								Speedup = 73 times
Finite Impulse Response Filter (FIR)	400us	10W	ADD:3, MUL: 2, OSC: 50MHz	360.24us	Power: 3.91W Area: 329 CLB slices	250milli-secs	1.30 secs	Improvement =80.76 %
								Speedup = 5 times
MPEG Motion Vector (MMV)	500us	6W	ADD:2, MUL: 3, OSC: 50MHz	480.08us	Power: 4.59W Area: 440 CLB slices	188milli-secs	4.14 secs	Improvement =95.45 %
								Speedup = 22 times
JPEG: Downsample	800us	4W	ADD:2, MUL: 1, OSC: 50MHz	720.12us	Power: 3.88W Area: 368 CLB slices	453milli-secs	24.14 secs	Improvement =98.12 %
								Speedup = 53 times
MESA: Feedback Points	900us	6W	ADD:2, MUL: 2, OSC: 50MHz	880.04us	Power: 5.98W Area: 578 CLB slices	340milli-secs	36.32 secs	Improvement =99.06 %
								Speedup = 106 times

provide increased acceleration without sacrificing the quality of the final solution when compared to existing DSE approach. The power values reported in Table 7 are not closer to reality power values for modern consumer applications but the main purpose of the reported values is to prove the efficiency of the proposed approach. Therefore, the proposed approach is able to handle the desired orthogonal issues encountered during DSE which is balancing the exploration speed and enhancing the preferred exploration result.

7.2 Experimental Results: Proposed approach ‘Rapid Design Space Exploration in High Level Synthesis Based on Area-Performance-Power Tradeoff and Power-Performance-Area Tradeoff using Hybrid Fuzzified Algorithm’ and Comparative study with recent approaches

This chapter primarily describes the experimental results of the proposed approach based on area-performance-power trade-off (elaborated in Chapter 3) and the improvements obtained compared to recent approaches [11] and [12]. This is because the proposed approach works most efficiently when trading off between hardware area and performance with the power kept as optimization criteria. However, for the sake of investigation and inquisitiveness, this approach has also been applied for exploring design points during power-performance tradeoff with hardware area kept as optimization criteria. A sample module library for this approach must consist of the same information as described in Section 7.1. The proposed hybrid DSE approach has been implemented and run on AMD Athlon 64 X2 Dual-Core Processor TK-157 with 3072 MB DDR2 memory, 64KB L1D cache and 256 KB L2 cache memory. The processor frequency is 1.90 GHz. For a qualitative analysis, the proposed hybrid approach was tested on a number of DSP benchmarks ranging from small to large number of variants. Many large size benchmarks were selected for the experiment. For example, Elliptic, an elliptic wave filter, and DiffEq, a differential equation solver, are from the NCSU CBL high-level synthesis benchmark suite [33]. Further DSP benchmarks such as discrete wavelet transformation (DWT), autoregressive filter (ARF), and MPEG motion vectors (MMV) [32] were also adopted for experiments. Additionally, benchmarks such as Finite Impulse Response (FIR) and Infinite Impulse Response Butterworth filter with large design solution spaces were also tested and compared to current DSE.

A. Experimental and Implementation Results for Area-Performance trade-off with power as optimization criteria using Hybrid Fuzzified Algorithm

In the proposed method, adding the fuzzy search technique to the priority factor method for DSE enhances the speed of the exploration process more than the current approach [12]. Approach [12] has been compared as it is also based on a multi parametric objective which considers hardware area, execution time and power consumption. The framework used during architecture design space construction as well as the searching method in [12] both have been considered during the comparison with the proposed approach. Experimental results in Table 8 indicate that the proposed hybrid approach is capable of achieving high acceleration compared to the exhaustive search as well as with approach [12]. Improvement of up to 45.45 % is achieved for the DWT benchmark compared to [12]. Similarly for large benchmarks like Elliptic Wave Filter (EWF), the proposed approach obtained improvement of 42 % compared to the DSE approach [12] as shown in Table 8. The results of the exploration time improvement and comparison with the current approach [12] for all tested benchmarks are shown in Table 8. For large benchmarks, the proposed hybrid approach yielded significantly improved results, with improvements ranging from 20% to 42% as evident from Table 8. Colossal acceleration of over 96% was obtained with the proposed DSE method compared to the exhaustive searching for Discrete Wavelet Transformation (DWT) benchmark as seen in Table 8. Additionally, verified through the experiments it was revealed for all the benchmarks that the proposed approach yielded optimal results as it met both the arbitrary user specified constraint values of area and execution time. To the best of the authors' knowledge there are no other works in the literature except [12] that considers area and pipelined execution time as constraint and power as optimization criteria

during exploration process. Since both the approaches find the same quality of solution through the exploration process, no comparison of results with [12] are reported. However, the proposed approach is able to achieve significant acceleration compared to [12]. For example, the runtime for the DWT benchmark using [12] is 273 milli-secs (ms), while the runtime using the proposed approach is 79 milli-secs (ms).

Table8. Experimental results of comparison between the proposed hybridized DSE with the current approach [12] for benchmarks

Benchmarks	Total possible architecture in the design space for exhaustive search	Architecture evaluation using [12] (Number of variants analyzed)		Architecture evaluation using proposed hybrid Priority Factor method with Fuzzy Search technique (Number of variants analyzed)	Improvement in architecture analyzed compared to current approach [12]	Improvement in architecture analyzed by proposed approach compared to exhaustive search	Run Time comparison	
		Area	Execution time				DSE Approach [12]	Proposed hybrid approach
Discrete Wavelet Transformation (DWT)	432	13	13	16	38.46%	96.29 %	0.273 sec (273 ms)	0.079 sec (79 ms)
		Total = 26						
Differential Equation Solver (HAL)	180	11	12	17	26.08%	90.55 %	0.156 sec (156 ms)	0.039 sec (39 ms)
		Total = 23						
Elliptic Wave Filter (EWF)	156	10	9	11	42.10%	87.82 %	0.447 sec (447 ms)	0.334 sec (334 ms)
		Total = 19						
Auto Regressive Filter	288	12	12	15	37.5%	94.79 %	0.433 sec (433 ms)	0.316 sec (316 ms)
		Total = 24						
MPEG Motion Vector (MMV)	756	14	13	31	-----	95.89 %	0.424 sec (424 ms)	0.430 sec (430 ms)
		Total = 27						
IIR Digital Butterworth Filter	1280	15	15	21	30 %	98.35 %	0.22 sec (220 ms)	0.101 sec (101 ms)
		Total = 30						
Finite Impulse Response Filter (FIR)	1200	15	15	24	20 %	98.00 %	0.395 sec (395 ms)	0.377 sec (377 ms)
		Total = 30						

Table9. Experimental results of comparison between proposed hybridized DSE with GA based approach [11]

Benchmarks	Total possible architecture in the design space for exhaustive search	Architecture evaluation using proposed hybrid approach (Number of variants analyzed)	Run Time comparison	
			Proposed hybrid approach	GA based DSE Approach [11]
				G(Max) = 100
Discrete Wavelet Transformation (DWT)	432	16	0.079 sec	4.27 sec
IIR Digital Chebyshev Filter	54	11	0.024 sec	2.35 sec
IIR Digital Filter 2	72	12	0.027 sec	1.90 sec
Elliptic Wave Filter	156	11	0.334 sec	19.71 sec
Auto Regressive Filter	288	15	0.316 sec	8.41 sec
MPEG Motion Vector	756	31	0.430 sec	9.37 sec
Infinite Impulse Response Digital Butterworth Filter	1280	21	0.101 sec	2.37 sec
Finite Impulse Response Filter	1200	24	0.377 sec	6.77 sec

The proposed method was also compared to another design space exploration approach based on the genetic algorithm [11] which considers dual parametric objectives. Although the method is very promising, it only considers two parameters, such as hardware area and latency, as design objectives during design space exploration. The proposed method, however, is based on multi parametric objective which additionally considers power consumption in addition to hardware area of resource and execution time (compared to only latency) as major design objectives. Secondly, [11] as considers only latency, thus it does not take into account the total execution time for ‘N’ sets of data during data pipelining. Total execution time is the total time taken for execution of ‘N’ sets of data and includes not only the first output delay (called latency), but also cycle time (difference in clock cycle between the outputs of two consecutive data by considering initiation interval). Hence the proposed method offers another advantage over method [11] as it considers the total execution time instead of just latency. The GA based approach [11] was run for maximum generation of G (max) =100 using the exact information for the GA parameters

provided in [11] in order to record the time taken to find the optimal solution. The runtime comparison of the proposed approach with [11] for different DSP benchmarks is reported in Table 9. As evident from the results obtained, the proposed approach achieves a significant reduction in time taken to perform design space exploration compared to [11], when run for provided G (max). For example, the runtime for the DWT benchmark using the proposed approach is just 79 ms while the runtime is 4.27 sec using [11] when $G(\text{max}) = 100$. Similar acceleration can be noted for other benchmark results as well. Therefore even by considering an extra parametric objective (i.e. power consumption) as well as taking into account the total execution time for data pipelining (instead of only latency), the proposed hybrid approach is able to provide respectable improvements for known benchmarks compared to [11].

B. Experimental and Implementation Results for Power-Performance trade-off with Area as optimization criteria using Hybrid Fuzzified Algorithm

It is worthwhile to mention again that the proposed approach works most efficiently when trading off between hardware area and performance with the power kept as optimization criteria. However, for the sake of investigation, curiosity and efficiency analysis, this section presents results when this approach has also been applied for exploration of power-performance tradeoff with hardware area kept as optimization criteria.

The sizes of the design space consisting of variants for the benchmarks are indicated in Table 10. For example, the total number of variants in the design space for DWT is 288; while on the other hand, the total number of variants of the design space for EWF and FIR are 450 and 1200 respectively. The results of the comparison of the proposed design space exploration process

with exhaustive analysis are shown in Table 10. Results indicate that the proposed approach is capable of achieving massive time improvements compared to the exhaustive search. Exploration time of up to 92.70 % is achieved for the well known Discrete Wavelet Transformation (DWT) high level synthesis benchmark. Moreover exploration time of 94.22 % and 97.75 % for EWF and FIR benchmarks are obtained respectively when compared to exhaustive search as shown in Table 10. Furthermore, the results are also compared with the design space exploration approach [12] as shown in Table 11. Investigations reveal that the proposed approach is able to provide high acceleration for design space exploration while simultaneously maintaining the accuracy needed in architecture selection. A exploration time of up to 37.50 % and 19.23 % is achieved for IIR Digital Filter 1 and Discrete Wavelet Transformation (DWT) benchmark respectively.

Table10. Experimental results of the proposed hybridized approach compared with exhaustive analysis for Benchmarks

Benchmarks	Total architectural variants in the Design space for one parameter	Total architectural variants in the design space for exhausted search for two parameters (Total size of the design space for the benchmark)	Proposed hybrid approach (Number of variants analyzed)			Improvements in variants analyzed using proposed approach compared to the exhaustive search (%)
			Variant searched for Power consumption	Variants searched for Execution time	Total variants Analyzed	
IIR Digital Butterworth Filter	24	48	4	4	15	68.75
IIR Digital Filter 1	32	64	4	4	15	76.56
IIR Digital Filter 2	36	72	4	5	16	77.77
IIR Digital Filter 3	48	96	5	6	18	81.25
Auto Regressive Filter	96	192	6	8	21	89.06
Discrete Wavelet Transformation	144	288	8	6	21	92.708
Differential Equation Solver	90	180	13	9	25	86.11
Digital IIR Chebyshev Filter	64	128	12	7	23	82.03
Elliptic Wave Filter	225	450	4	20	26	94.22
Finite Impulse Response Filter	600	1200	3	19	27	97.75

Further, the proposed approach was also verified for two benchmarks which consist of large number of variants in the design space (EWF with 450 variants and FIR with 1200 variants). Results indicated that the proposed approach when compared to [12] for EWF and FIR benchmarks yielded exploration time improvement of 13.33 % and 18.18 % respectively as shown in Table 11. Therefore as evident from Table 11, the proposed hybrid approach provides accelerated design space exploration with average exploration time improvement of more than 22 % for benchmark applications, compared to the previous approach in [12]. Additionally, as verified through the experiments it was revealed for all the benchmarks that the proposed approach yielded optimal results as it met both the arbitrarily specified constraint values of area and execution time. Therefore the proposed hybrid approach also provides increased acceleration in the design space exploration process during power-performance tradeoff with area as optimization criteria.

Table11. Experimental results of the comparison between the proposed DSE approach with approach [12]

Benchmarks	Total architectures in the design space for exhausted search for two parameters	Architecture evaluation using current existing approach [12] (Number of variants)	Architecture evaluation using proposed hybrid approach (Number of variants)	Percentage Improvements in architecture analyzed compared to current existing approach	Average improvement in architecture analyzed by the proposed approach wrt [12]
IIR Digital Butterworth Filter	48	22	15	31.81 %	22.57 %
IIR Digital Filter 1	64	24	15	37.50 %	
IIR Digital Filter 2	72	24	16	33.33 %	
IIR Digital Filter 3	96	25	18	28.00 %	
Auto Regressive Filter	192	24	21	12.50 %	
Discrete Wavelet Transformation	288	26	21	19.23 %	
Differential Equation	180	29	25	14 %	
Digital IIR Chebyshev	128	28	23	17.85 %	
Elliptic Wave Filter	450	30	26	13.33 %	
FIR Filter	1200	33	27	18.18 %	

7.3 Experimental Results: Proposed approach ‘Priority Function Driven Design Space Exploration in High Level Synthesis Based on Power Gradient Technique’ and Comparative study with a recent approach

This section primarily describes the experimental results of the proposed approach based on static power optimization under minimum control step usage (elaborated in Chapter 4) and the improvements obtained compared to recent approach [11].

The proposed integrated design space exploration approach has been implemented and run on

Table12. Experimental Results of the proposed approach for the DSP Benchmarks

DSP Benchmarks	Experimental Parameters (Note: cc = clock cycles)						
	Resource combination		Latency	Initial Cost in terms of Power dissipation	Final Power dissipation of proposed approach	% Reduction in Power dissipation	Runtime of proposed approach
	Initial Solution	Proposed approach	Proposed approach				
Discrete Wavelet Transformation (DWT)	5(*), 4(+),18 (mux), 9 (demux), 15 (Reg)	2(*), 2(+),8 (mux), 4(demux), 14 (Reg)	32cc	2.94 Watts	1.38 Watts	53.06 %	3.18 secs
Band Pass Filter (BPF)	4(*), 3(+/-), 14 (mux), 7 (demux), 20 (Reg)	2(*), 3(+/-),10 (mux), 5(demux), 19 (Reg)	28cc	2.49 Watts	1.60 Watts	35.74 %	1.38 secs
Finite Impulse Response (FIR)	8(*), 8(+), 32 (mux), 16 (demux),23 (Reg)	2(*), 5(+),12 (mux), 6(demux), 20 (Reg)	28cc	4.87 Watts	1.73 Watts	64.47 %	5.63 secs
IIR Digital Butterworth Filter	5(*), 1(+/-),12 (mux),6 (demux), 14 (Reg)	2(*), 1(+/-), 6 (mux),3 (demux), 11 (Reg)	16cc	2.57 Watts	1.20 Watts	53.30 %	1.62 secs
IIR Digital Chebyshev Filter	5(*), 2(+),14(mux), 7 (demux), 16 (Reg)	3(*), 2(+),10 (mux), 5 (demux), 16 (Reg)	10cc	2.60 Watts	1.86 Watts	28.46 %	3.19 secs
MPEG Motion Vectors (MMV)	14(*), 5(+),38(mux), 19 (demux),	5(*), 5(+),20(mux), 10 (demux),	14cc	7.52 Watts	3.42 Watts	54.52 %	1.95 secs

AMD Athlon 64 Processor with 3GB RAM and 1.6 GHz processor frequency. In order to perform a qualitative assessment, the proposed approach has been compared with a heuristic GA based approach [11]. Furthermore for comparison with [11], the parameters chosen were quality of the final solution found measured in terms of Effective Cost Metric (ECM) and optimization runtime. The metric is a combination of latency and power given by eqn. (44):

$$ECM = W_1 \cdot \frac{L}{L_{max}} + W_2 \cdot \frac{P}{P_{max}} \quad (44)$$

W1 and W2 are the weightage of the operating constraints for latency and hardware area (Note: $0 \leq W_1 \leq 1$ and $0 \leq W_2 \leq 1$). For this experiment, $W_1 = W_2 = 0.5$ has been kept, since equal priority was given to both latency of the final solution and the power dissipated by the solution. ‘L’ and ‘P’ are the latency and power dissipation of the solution found. ‘Lmax’ and ‘Pmax’ are the values of maximum latency (found by using minimum FU’s) and maximum power dissipation (using maximum FU’s) respectively. Equation (44) has been divided with maximum values of latency and power respectively in order to obtain normalized values for each. The above metric was proposed for comparison since the quality of a solution cannot be solely determined from the latency expenditure or the dissipated power, but rather a combination of both. The results obtained through the proposed approach are shown in Table 12. The power optimization (minimization) obtained for the final resource solution (FU’s) as noted from the

Table13. Comparison of measured power consumption through Xilinx Power Analyzer (XPA) 9.2i

DSP Benchmarks	Power Consumption of the initial solution (Static and Dynamic)	Power consumption of final solution Proposed method (Static and Dynamic)	% Reduction in Power consumption
Discrete Wavelet	2.43 W	2.10 W	≈ 11 %
Band Pass Filter	3.26 W	2.92 W	
Finite Impulse Response	3.54 W	2.98 W	
IIR Digital Butterworth Filter	1.90 W	1.74 W	
IIR Digital Chebyshev Filter	2.15 W	2.07 W	
MPEG Motion Vectors	6.14 W	5.30 W	

Table14. Experimental Results of the comparison between the proposed approach and recent approach [11]

DSP Benchmarks	Experimental Parameters for Comparison						
	ECM		Improvement in quality of final solution	Average Improvement of quality final solution	Runtime (seconds)		Reduction in Runtime
	[11]	Proposed approach			[11]	Proposed approach	
Discrete Wavelet	0.64	0.61	4.68 %	5.07 %	7.53	3.18	57.76 %
Band Pass Filter	0.60	0.56	6.67 %		13.96	1.38	90.11 %
Finite Impulse	0.46	0.46	----		11.04	5.63	49 %
IIR Digital	0.56	0.53	5.35 %		3.04	2.08	31.57 %
IIR Digital	0.58	0.55	5.17 %		2.69	1.56	42 %
MPEG Motion Vectors	0.35	0.32	8.57 %		12.32	1.95	84.17 %

results for all DSP benchmarks such as DWT, BPF, FIR, Digital Butterworth filter, Chebyshev filter and MPEG are impressive. The comparison results of the measured power consumption viz. dynamic power at 100MHz and static power for Spartan 3E FPGA in Xilinx Power Analyzer (XPower) 9.2i tool suite are shown in Table 13. Analysis of the measured power for the DSP benchmarks reveals that an adequate minimization of power consumption is obtained using the proposed approach. On average, power reduction of $\approx 11\%$ is obtained using the proposed approach compared to the initial solution (ASAP solution). The implementation runtime of the proposed optimization approach and its comparison with approach [11] is illustrated in Table 14. Table 14 also reflects the comparison of the final solution found by both approaches. As verified through the experiment, the proposed approach was able to find optimal solution for all the benchmarks (in most cases global optimal solution was found, however local optimal solution was found in some cases). Due to its ability to obtain optimal solution for all test cases verified, the average improvement in the quality of the final solution found is 5.07 % compared to [11]. Therefore, as evident in Table 14, the average reduction in runtime for all benchmarks is approx. 60 % and average improvement in quality of final solution is 5.07 % compared to [11].

7.4 Experimental Results: Proposed approach ‘A Multi Structure Genetic Algorithm for Integrated Design Space Exploration of Scheduling and Allocation in High Level Synthesis for DSP Kernels’ and Comparative study

This section primarily describes the experimental results of the proposed approach for integrated scheduling and module allocation based on power-performance tradeoff (elaborated in Chapter 5) and the improvements obtained compared to recent approach [11].

A short summary of the experimental setup is given in Table 15. The proposed MSGA has been implemented in high level language [34] [35] and run on AMD Athlon 64 X2 Dual-Core Processor TK-157 with 3072 MB DDR2 memory, 64KB L1D cache and 256 KB L2 cache memory. The processor frequency is 1.90 GHz. As mentioned before, W_1 and W_2 are the user specified preference for power consumption and execution time constraints. The proposed MSGA has been compared with another recent powerful GA-based approach [11] for $G(\text{Max}) = 100$ (keeping $W_1 = W_2 = 0.5$) to perform a qualitative assessment of the proposed approach. The proposed system is able to find the global optimal (sometimes local optimal) resource solution in terms of number of FU’s and clock frequency which is high quality in nature. Further, it is able to better optimize the power consumption and the execution time for the final resource solution

Table15. Experimental Setup for MSGA	
Machine	AMD Athlon 64 X2 Dual-Core Processor TK-157 with 3072 MB DDR2 memory, 64KB L1D cache and 256 KB L2 cache memory, processor frequency is 1.90 GHz
Benchmarks	IIR Digital Filter, ARF, DWT, Digital Butterworth filter, EWF, WDF, BPF
GA Parameters	1) $W_1 = W_2 = 0.5$, 2) $G(\text{Max}) = 100$ 3) $P_{\text{cross}} = 1.0$, 4) $P_m = 0.25$
Parameters of comparison	a) final resource combination b) latency (CS) c) execution time d) power consumption
# of Runs for averaging	25 runs performed for each benchmark
User preference	$W_1 = W_2 = 0.5$

found. The system also optimizes the latency of the final solution found. On the other hand, [11] is not able to optimize the execution time considerably due to its inability to create a genuinely pipelined functional data paths. Therefore the total execution time (*being a function of latency, cycle time and pipelined data (N) as shown in eqn. (42)*) does not get optimized for [11]. As mentioned before since [11] does not have the ability to explore the optimal clock frequency from a set of various clock frequencies it does not optimize the power. For only for the sake of comparison, the power consumed by the FU's, storage elements and interconnect units at 100 MHz obtained by [11] were measured to provide an estimate of average power consumption. For determination of execution time in [11], 'N' sets of processing data are simply multiplied with the delay of each data due to lack of genuine functional data pipelining capability. Thus, $T_{exe}^{[11]}$

Table16. Experimental Results of Comparison with [11] for the DSP Benchmarks

DSP Benchmarks	Parameters of Comparison							
	Note: cc= clock cycles and a.u. = area unit (1 au = 1CLB in Spartan 3E FPGA)							
	Explored final resource combination		Latency		Execution time in clock cycles (N=1000)		Power Consumed (Watts)	
	MSGA	[11]	MSGA	[11]	MSGA	[11]	MSGA	[11]
IIR Digital Filter	3(*), 2(+)	2(*), 1(+), 10(Reg)	14cc	18cc	12,002 cc	18,000 cc	0.91 W	1.45W
	8(mux),4(demux), 9(Reg), 50MHz							
ARF	4(*), 4(+)	4(*),3(+), 18(Reg)	24cc	25cc	10,014 cc	25,000 cc	1.62W	2.92 W
	16(mux), 8(demux), 18(Reg), 50MHz							
DWT	1(*), 2(+)	1(*),1(+), 10(Reg)	44cc	44cc	38,006 cc	44,000 cc	1.56 W	2.97W
	6(mux),3(demux), 10(Reg), 50MHz							
Digital Butterworth Filter	2(*), 1(+), 1(-)	2(*), 1(+), 1(-), 10(Reg)	18cc	18cc	12,006 cc	18,000 cc	1.59 W	1.59W
	8(mux),4(demux), 10(Reg), 100MHz							
WDF	2(*), 3(+),1(-)	1(*), 2(+),1(-), 15(Reg)	45cc	56cc	23,022 cc	56,000 cc	1.34 W	2.12W
	12(mux),6(demux), 11 (Reg), 50MHz							
BPF	1(*), 1(+), 2(-)	2(*), 2(+), 2 (-), 11 (Reg)	46cc	30cc	43,003 cc	30,000 cc	2.28 W	2.83W
	8(mux),4(demux), 15 (Reg), 100MHz							
EWF	1(*), 2(+)	1(*), 2(+), 11(Reg)	56cc	59cc	56,000 cc	59,000 cc	0.96 W	2.04W
	6(mux),3(demux), 9(Reg), 50MHz							

= $N*L$. In contrast, since proposed MSGA considers cycle time resulting from initiation interval and latency to create a genuinely pipelined functional data path, the execution time in clock cycles ($T_{\text{exe}}^{\text{MSGGA}}$) of the proposed MSGA is determined from eqn. (42).

The better result of power and execution time for the proposed MSGA compared to [11] for all the benchmarks is clearly evident in Table 16. An average of twenty five runs has been reported for both approaches in Table 16. For example in case of IIR digital filter benchmark, the resource combination found using proposed MSGA is 3(*), 2(+), 8(mux), 4(demux), 9(registers) and 50 MHz clock frequency oscillator based on the user specified constraints. The latency of the scheduling solution through MSGA is 14cc. The power consumed by the final solution found is 0.91W and the execution time for $N = 1000$ is 12,002 cc. On the other hand, [11] yields an optimal resource combination which is 2(*), 1(+) with latency of 20cc. However, the corresponding execution time is large, equal to 20,000 cc. The power consumed by [11] is also large due to inability to explore optimal clock frequency oscillator. Moreover, for WDF benchmark, the MSGA finds a final optimal scheduling solution in 45cc at the expense of just an extra multiplier and adder compared to 56cc taken by using [11]. Therefore for WDF benchmark, the MSGA saves 11cc at the expense of only an adder and multiplier. The proposed approach produces a solution which consumes 23,002 cc to process 1000 data and 1.34W of power compared to solution which consumes 56,000 cc execution time and 2.12W power using [11]. For benchmarks such as DWT and Digital Butterworth filter although the final solution found using proposed approach is similar to solution obtained using [11], but the proposed approach is able to achieve reduced cycle and execution times. Note: Power values reported in Table 12 and 16 are not in exact alignment with practical power values for modern consumer applications but the main purpose of the reported values are to prove the benefit relative to prior work.

Chapter 8

Conclusion and Future Works

The dissertation has presented multiple novel frameworks for addressing multi-dimensional issues in the design space exploration problem in high level synthesis of computation intensive applications (primarily DSP kernels). Each proposed framework is unique in its own kind in terms of the solution it proposes for resolving notorious optimization problems for different user requisite. In particular, the dissertation introduced four different frameworks for performing fast and efficient multi-objective tradeoff based on different user criteria viz. a) Novel Priority Factor based Pareto optimal framework methodology for accelerated design space exploration based on power-performance-area/cost tradeoff. Experiments revealed that this methodology provided exploration time improvement of greater than 90 % in exploration process compared to a recent technique for various signal processing DFG benchmarks b) Novel hybrid Fuzzy Algorithm Based Pareto optimal framework for exploration of Area- Performance- power tradeoff. Experimental results showed that this hybrid technique provides an average exploration time improvement of greater than 35 % during area-performance-power tradeoff and 22 % in

exploration process during power-performance-area tradeoff respectively when compared to a recent technique for various size signal processing DFG benchmarks. The above approaches a) and b) were successful in laying the foundation for exploring the design points from the architecture design space according to the performance objective and intended functionality. Moreover the above DSE were capable of resolving the conflicting objectives in DSE by concurrently maximizing the accuracy in evaluation of the design point and minimizing the time expended for design space assessment c) Novel Priority Function driven integrated design space exploration (scheduling and module selection) in high level synthesis based on Power Gradient technique for static power optimization under minimum control step usage. Experimental results indicated successful improvement in the quality of final solution by an average of 5.07 % and reduction in the exploration runtime by an average of 59% compared to a current approach for standard DSP DFG Benchmarks d) Novel Multi structure based Genetic Algorithm for integrated exploration of scheduling and allocation during power-performance tradeoff. The results produced by proposed approach are better compared to another genetic algorithm based approach, for almost all digital signal processing DFG benchmarks.

Therefore, the dissertation presents various solutions for multi-dimensional design space exploration problems encountered during multi-objective optimization in high level synthesis. It is also worthwhile to mention that the presented methods are applicable for computation intensive tasks/data hungry applications (i.e. applications that can be represented through data flow graphs). The proposed methodologies can be efficiently applied to perform exploration in various high level synthesis problems depending on the varied user criteria.

Scope of Future Work

There is much potential in the area of design space exploration and high level synthesis to improve the search time for finding the final design architecture, and thereby accelerate the speed of the exploration process. The developed design space exploration approach for high level synthesis can be improved further by decreasing the number of architectural variants to be analyzed during the exploration process. Reducing the analysis of the architectural variants directly reduces the search time which in turn impacts the design time and hence will help in faster designing. Another aspect of high level synthesis, which also has significant potential for improvement, is the development of many other parameters such as reliability, temperature etc., for high level estimation which stills lies in the nascent stage of development. As shown in the recent study [47-50] minimization of power does not guarantee complete minimization of temperature. The temperature of a chip depends not only on the activity rate of the modules but also on the past history of the activity rate of the modules. Therefore, temperature specific resource binding algorithms need to be developed and evaluated for these with novel parametric models. These algorithms can be integrated with existing high level synthesis techniques for generation of optimized RTL circuits. This will allow system architects to design systems based on performance-temperature trade-offs. Another aspect of high level synthesis that needs further research is the unification of physical level designing with high level synthesis design. Efforts can be made to incorporate floorplanning details into high level models to increase accuracy of the evaluation model. This will not only provide a significant boost to the circuit designer optimizing at low level, but would also benefit the system architects in precisely exploring the extensive design space based on user requirements.

Refereed Publications

Patents

- S1. **Anirban Sengupta** (with Reza Sedaghat), “System and Methodology for Development of System Architecture”, *US Patent allowed by United States Patent and Trademark Office (USPTO)*, Publication number: US 2012/0159119 A1, Publication Date: June 21, 2012. (*Invention: A Fast Multi Objective Design Space Exploration approach using Priority Factor Method*)
- S2. **Anirban Sengupta** (with Reza Sedaghat), “System and Methodology for Development of System Architecture”, *Canadian Patent filed to Canadian Intellectual Property Office (CIPO)*, Application no. 20925-25, December 21, 2010. (*Invention: A Fast Multi Objective Design Space Exploration approach using Priority Factor Method*)
- S3. **Anirban Sengupta** (with Reza Sedaghat), “System and Method for Development of System Architecture”, *US Patent filed to United States Patent and Trademark Office (USPTO)*, Application no. 13/118,139, May 27, 2011. (*Invention: A Fast Multi Objective Design Space Exploration approach using Fuzzy Searching Method in High Level Synthesis for ASIC's*”, *Accepted by MARS Innovation, Govt. of Canada*).
- S4. **Anirban Sengupta** (with Reza Sedaghat), “System and Method for Development of System Architecture”, *Canadian Patent filed to Canadian Intellectual Property Office (CIPO)*, Application no. 20925-28, May 27, 2011. (*Invention: A Fast Multi Objective Design Space*

*Exploration approach using Fuzzy Searching Method in High Level Synthesis for ASIC's”,
Accepted by MARS Innovation, Govt. of Canada.)*

Refereed Journals

- S5. **Anirban Sengupta**, Reza Sedaghat, Pallabi Sarkar “Rapid Exploration of Integrated Scheduling and Module Selection in High Level Synthesis for Application Specific Processor Design”, *Elsevier Journal of Microprocessors and Microsystems*, Volume 36, Issue 4, June 2012, Pages 303–314.
- S6. **Anirban Sengupta**, Reza Sedaghat, Pallabi Sarkar, “A Multi Structure Genetic Algorithm for Integrated Design Space Exploration of Scheduling and Allocation in High Level Synthesis for DSP Kernels”, *Elsevier Journal of Swarm and Evolutionary Computation*, Volume 7, December 2012, Pages 35–46.
- S7. **Anirban Sengupta**, Reza Sedaghat, Zhipeng Zeng, “Multi objective Efficient Design Space Exploration and Architectural Synthesis of an Application Specific Processor (ASP)”, *Elsevier Journal of Microprocessors and Microsystems*, Volume 35, Issue 4, June 2011, pp. 392-404.
- S8. **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”, *Elsevier Journal of Microelectronics Reliability*, Vol. 50, Issue 3, 2010, pp. 424-437.

(Note: This Research Journal paper featured in “SCIENCE DIRECT TOP 25 HOTTEST ARTICLE” from ‘Microelectronics Reliability’, Elsevier of the Engineering area in the first quarter of year 2010).

S9. **Anirban Sengupta**, Reza Sedaghat, Zhipeng Zeng, “Rapid Design Space Exploration by Hybrid Fuzzy Search Approach for Optimal Architecture determination of Multi Objective Computing Systems”, *Elsevier Journal of Microelectronics Reliability*, Vol. 51, Issue 2, 2011, pp. 502-512.

S10. **Anirban Sengupta**, Reza Sedaghat, “A High Level Synthesis Design Flow from ESL to RTL with multi-parametric optimization objective”, *IETE Journal of Research*, Volume 57, Issue 2, 2011, pp. 169-186.

Refereed Conferences

S11. **Anirban Sengupta**, Reza Sedaghat, “Priority Function Driven Design Space Exploration in High Level Synthesis Based on Power Gradient Technique”, *Accepted in Student Forum of 17th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC 2012)*, Australia, pp: 25, 2012.

- S12. **Anirban Sengupta**, Reza Sedaghat, “Integrated Scheduling, Allocation and Binding in High Level Synthesis using Multi Structure Genetic Algorithm based Design Space Exploration System”, In *Proceedings of 12th IEEE/ACM International Symposium on Quality Electronic Design (ISQED 2011)*, Silicon Valley, California, USA, March 2011, pp. 486-494.
- S13. **Anirban Sengupta**, Reza Sedaghat, “A Hybrid Fuzzy Search Approach for Fast Design Space Exploration of Multi-Objective VLSI Systems”, *Accepted in the Student Forum of 16th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, Yokhoma, Japan, 2011, Paper ID: SF15.
- S14. **Anirban Sengupta**, Reza Sedaghat, Zhipeng Zeng, “Rapid Design Space Exploration for multi parametric optimization of VLSI designs”, In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, Paris, France, pp: 3164-3167, June 2, 2010.
- S15. **Anirban Sengupta**, Reza Sedaghat, “Accelerated Exploration of Cost-Performance Tradeoffs for Multi Objective VLSI designs”, In *Proceedings of 22nd IEEE International Conference on Microelectronics (ICM)*, 2010, pp. 100-103.
- S16. **Anirban Sengupta**, Reza Sedaghat “Rapid Exploration of Power-Delay Tradeoffs using Hybrid Priority Factor and Fuzzy Search”, In *Proceedings of 22nd IEEE International Conference on Microelectronics (ICM)*, Egypt, 2010, pp. 355-358.

- S17. **Anirban Sengupta**, Reza Sedaghat, Pallabi Sarkar, “Integrated Scheduling, Allocation and Binding in High Level Synthesis for Performance-Area Tradeoff of Digital Media Applications”, *Proceedings of 24th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2011)*, Canada, 2011, pp. 533-537.
- S18. **Anirban Sengupta**, Reza Sedaghat, Pallabi Sarkar, “Priority Function based Power Efficient Rapid Design Space Exploration of Scheduling and Module Selection in High Level Synthesis”, In *Proceedings of 24th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2011)*, Niagara, Canada, May 2011, pp. 538-543.
- S19. **Anirban Sengupta**, Reza Sedaghat, Zhipeng Zeng, “Hardware Efficient Design of speed optimized Power stringent Application Specific Processor”, In *Proceedings of 21st IEEE International Conference on Microelectronics (ICM), Morocco*, pp: 167-170, December 22, 2009.
- S20. **Anirban Sengupta**, Reza Sedaghat, “Fast Design Space Exploration for Multi Parametric Optimized VLSI and SoC Designs”, *15th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC 2010)*, Taiwan, 2010, ID: 26.
- S21. Pallabi Sarkar, Reza Sedaghat, **Anirban Sengupta**, “Power Gradient Based Design Space Exploration in High Level Synthesis for DSP Kernels”, Accepted for Publication in *Proceedings of 23rd IEEE International Conference on Microelectronics (ICM)*, pp: 1 – 6, December 2011.

- S22. Summit Sehgal, Reza Sedaghat, **Anirban Sengupta**, Zhipeng Zeng, “Multi Parametric Optimized Architectural Synthesis of an Application Specific Processor”, In *Proceedings of 14th IEEE International CSI Computer Conference (CSICC)*, 2009, pp: 89-94.
- S23. Zhipeng Zeng, Reza Sedaghat, **Anirban Sengupta**, “A Framework for Fast Design Space Exploration using Fuzzy search for VLSI Computing Architectures”, In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), Paris, France*, June 2, 2010, pp: 3176-3179.
- S24. Zhipeng Zeng, Reza Sedaghat, **Anirban Sengupta**, “A Novel Framework of Optimizing Modular Computing Architecture for multi objective VLSI designs”, In *Proceedings of 21st IEEE International Conference on Microelectronics (ICM), Morocco*, 2009, pp: 322-325.
- S25. Summit Sehgal, Reza Sedaghat, **Anirban Sengupta**, “Automated Design Space Exploration for DSP Applications High Level Synthesis with Stability in Competition”, *Accepted for Publication, Proceedings of 2nd IEEE Latin American Symposium on Circuits and Systems (LASCAS)*, Columbia, February 2011.
- S26. **Anirban Sengupta**, Reza Sedaghat, Pallabi Sarkar “Integrated Design Space Exploration Based on Power-Performance Trade-off using Genetic Algorithm”, In *Proceedings of ACM International Conference on Advances in Computing and Artificial Intelligence*, 2011, pp. 76-80.

- S27. Pallabi Sarkar, Reza Sedaghat, **Anirban Sengupta**, “Application Specific Processor vs. Microblaze Soft Core RISC Processor: FPGA Based Performance and CPR Analysis”, In *Proceedings of ACM International Conference on Advances in Computing and Artificial Intelligence*, 2011, pp.81-84.
- S28. **Anirban Sengupta**, Reza Sedaghat, “A Study on Architecture Optimization of the RISC Processor used for System-on Chip (SoC) design”, In *Proceedings of Research Innovation Symposium, Ryerson University, Canada*, 2010, pp: 31.
- S29. Summit Sehgal, Reza Sedaghat, **Anirban Sengupta**, “Fault Monitoring Transformer Reliability ASIC Design based on Ringing Effect Signature Analyzer”, In *Proceedings of Research Innovation Symposium, Ryerson University, Canada*, 2010, pp: 32.

THESIS

- S30. **Anirban Sengupta** “A Fast Design Space Exploration Based on Priority Factor for a Multi Parametric Optimized High Level Synthesis Design Flow”, *Master of Applied Science (M.A.Sc) Thesis, Ryerson University, Toronto, Canada*, Jan 2010, (*Nominated for Governor General’s Gold Medal in Canada for the Master’s Thesis*).

Appendix

Power Up


Shut Down


Optimizations Summary


Admin Page


Step One

Module Library : module_library_EWF.txt

Input : EWF Benchmark.txt

Number of processing data to be pipelined (N)



Sponsored by:


MaRS Innovation

Step 1: Design Automation Tool (described in Chapter 7. See keystones of the tool in Section 7.2)

Power Up


Shut Down


Optimizations Summary


Admin Page


Step Two

Maximum Execution Time (us): Max - 1360.08 us
Min - 90.09 us

Select which field you would like to use.
Circuit area:
Power Consumption:

Maximum Circuit Area (au): Max - 290 au
Min - 124 au

Maximum Power Consumption (W): Max - 5.8 W
Min - 0.496 W



Sponsored by:


MaRS Innovation

Step 2: Design Automation Tool (described in Chapter 7. See keystones of the tool in Section 7.2)

Power Up



Shut Down



Optimizations Summary



Admin Page



Sponsored by:

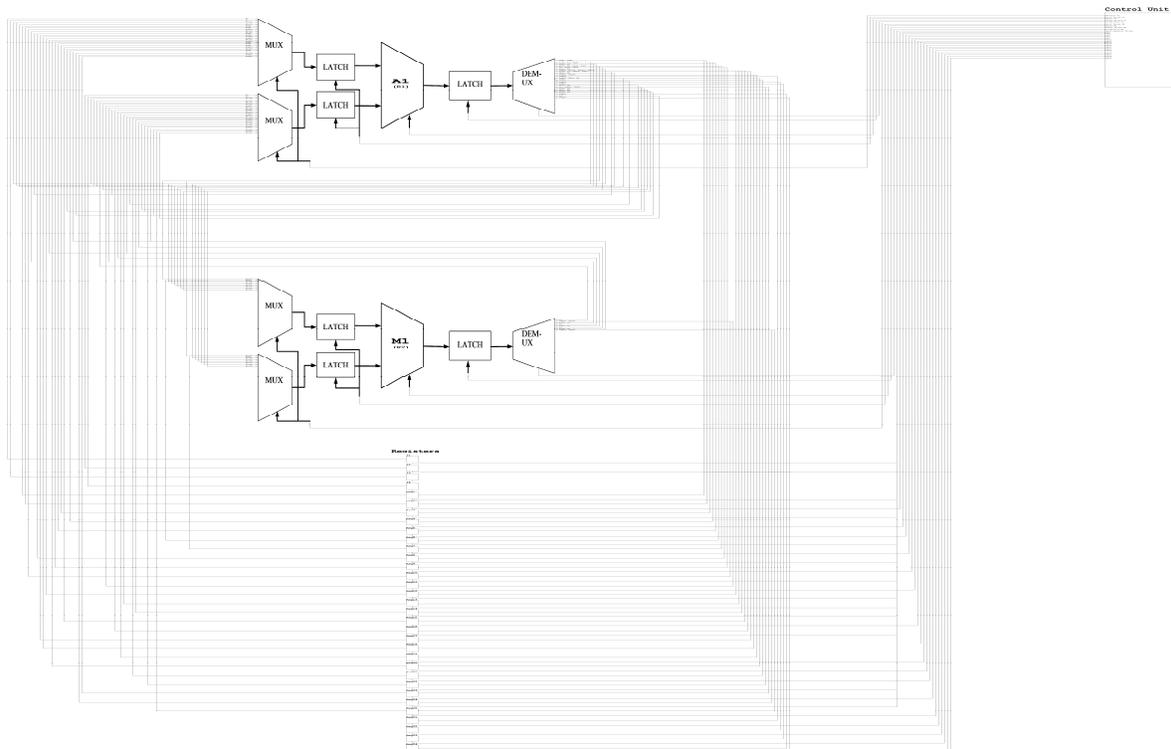
RYERSON UNIVERSITY



MaRS Innovation



Step 3: Summary of proposed Design Automation Tool (described in Chapter 7. See keystones of the tool in Section 7.2)



Sample schematic (data path and controller) output of the Design Automation Tool (in Chapter 7)

The portion of a sample VHDL produced by proposed tool (described in Chapter 7) for Elliptic Wave Filter benchmark containing port map detail is shown below

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--use work.ProjPackage.all;
---use work.filter_package.all;

entity FinalProduct is
  Port (
    clock : in  STD_LOGIC;
    resetn : in  STD_LOGIC;
    --    Busy : out STD_LOGIC;
    --    Ready : out STD_LOGIC
    I1 : in STD_LOGIC_VECTOR (15 downto 0);
    I2 : in STD_LOGIC_VECTOR (15 downto 0);
    I3 : in STD_LOGIC_VECTOR (15 downto 0);
    I4 : in STD_LOGIC_VECTOR (15 downto 0);
    Y1 : out STD_LOGIC_VECTOR (15 downto 0);
    Y2 : out STD_LOGIC_VECTOR (15 downto 0);
    Y3 : out STD_LOGIC_VECTOR (15 downto 0);
    Y4 : out STD_LOGIC_VECTOR (15 downto 0);
    Y5 : out STD_LOGIC_VECTOR (15 downto 0);
  );
end FinalProduct;
component control_unit
port(
  clock:in std_logic;
  reset:in std_logic;
  --count1 :out INTEGER;
  InputRegisterStrobe:out std_logic;
  addsub_A1:out std_logic;
  latch_strobe_A1:out std_logic;
  output_strobe_A1:out std_logic;
  enable_A1:out std_logic;
  latch_strobe_M1:out std_logic;
  output_strobe_M1:out std_logic;
  enable_M1:out std_logic;
  Strobe_Reg1:out std_logic;
  Strobe_Reg2:out std_logic;
  Strobe_Reg3:out std_logic;
  Strobe_Reg4:out std_logic;
  .....
  .....
  .....
  .....
  .....
  .....
  .....
begin

reset <= not resetn;
RegI1: Reg16 port map(I1,InputRegisterStrobe,DataRegI1);
RegI2: Reg16 port map(I2,InputRegisterStrobe,DataRegI2);
Reg1: Reg16 port map(DeMultiplexer_A1_0,Strobe_Reg1,DataReg1);

```

```

Reg2: Reg16 port map(DeMultiplexer_A1_0,Strobe_Reg2,DataReg2);
RegI3: Reg16 port map(I3,InputRegisterStrobe,DataRegI3);
RegI4: Reg16 port map(I4,InputRegisterStrobe,DataRegI4);
Reg3: Reg16 port map(DeMultiplexer_A1_1,Strobe_Reg3,DataReg3);
Reg4: Reg16 port map(DeMultiplexer_A1_1,Strobe_Reg4,DataReg4);
Reg5: Reg16 port map(DeMultiplexer_A1_2,Strobe_Reg5,DataReg5);
Reg6: Reg16 port map(DeMultiplexer_A1_4,Strobe_Reg6,DataReg6);
Reg7: Reg16 port map(DeMultiplexer_A1_4,Strobe_Reg7,DataReg7);
Reg8: Reg16 port map(DeMultiplexer_A1_5,Strobe_Reg8,DataReg8);
Reg9: Reg16 port map(DeMultiplexer_A1_5,Strobe_Reg9,DataReg9);

```

```

.....
.....

```

The portion of a sample VHDL of the controller for EWF is shown below:

```

architecture Behavioral of control_unit is
signal count : INTEGER RANGE 0 TO 132;
signal busy: std_logic;
begin
process(clock,reset)
begin
if (clock'event and clock='1') then
    if(reset='0')then
-----count 0-----
        if count=0 then
            --Reset all latches and units
            latch_strobe_A1<= '0';
            add_sub_A1<= '0';
            output_strobe_A1<= '0';
            enable_A1<= '0';
            Selector_A1<= "00000";
            Deselector_A1<= "00000";

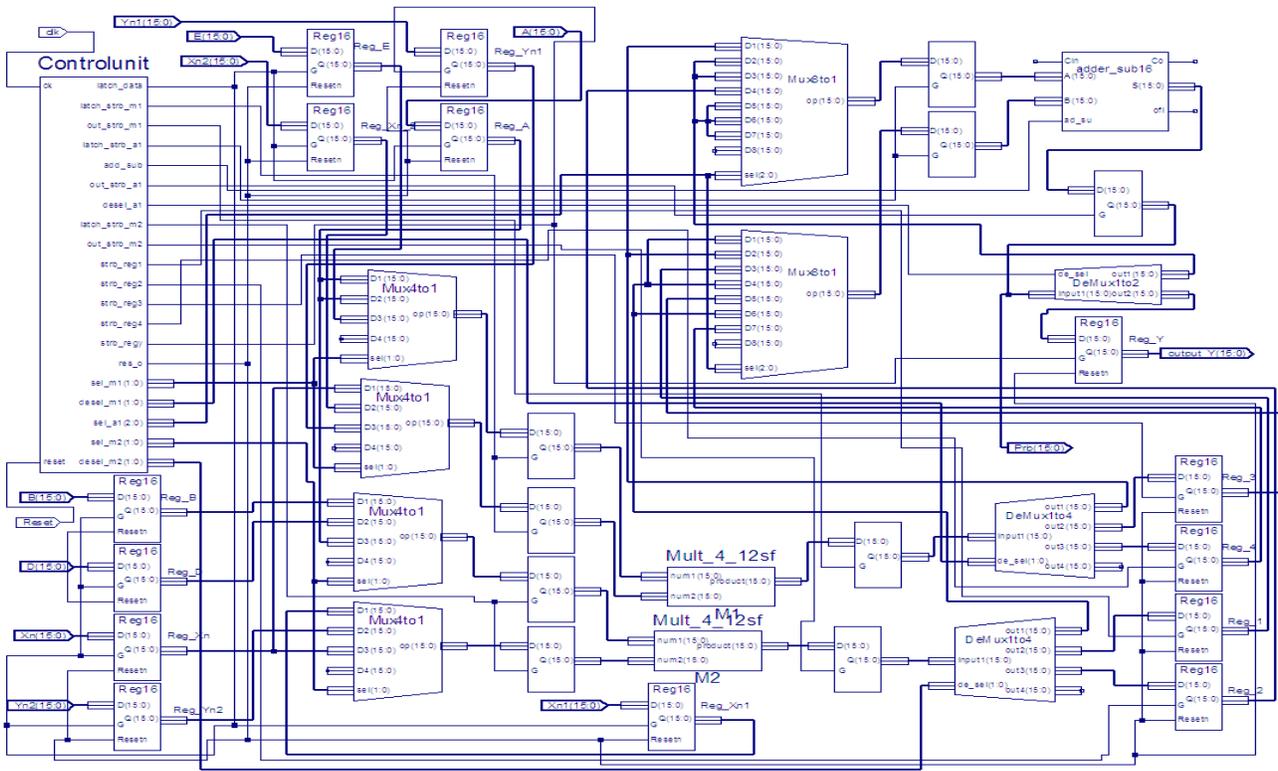
            .....
            .....
-----count 130-----

            if count=130 then
                enable_A1<='0';
                RegY5<='1';
                count<=count+1;
            end if;

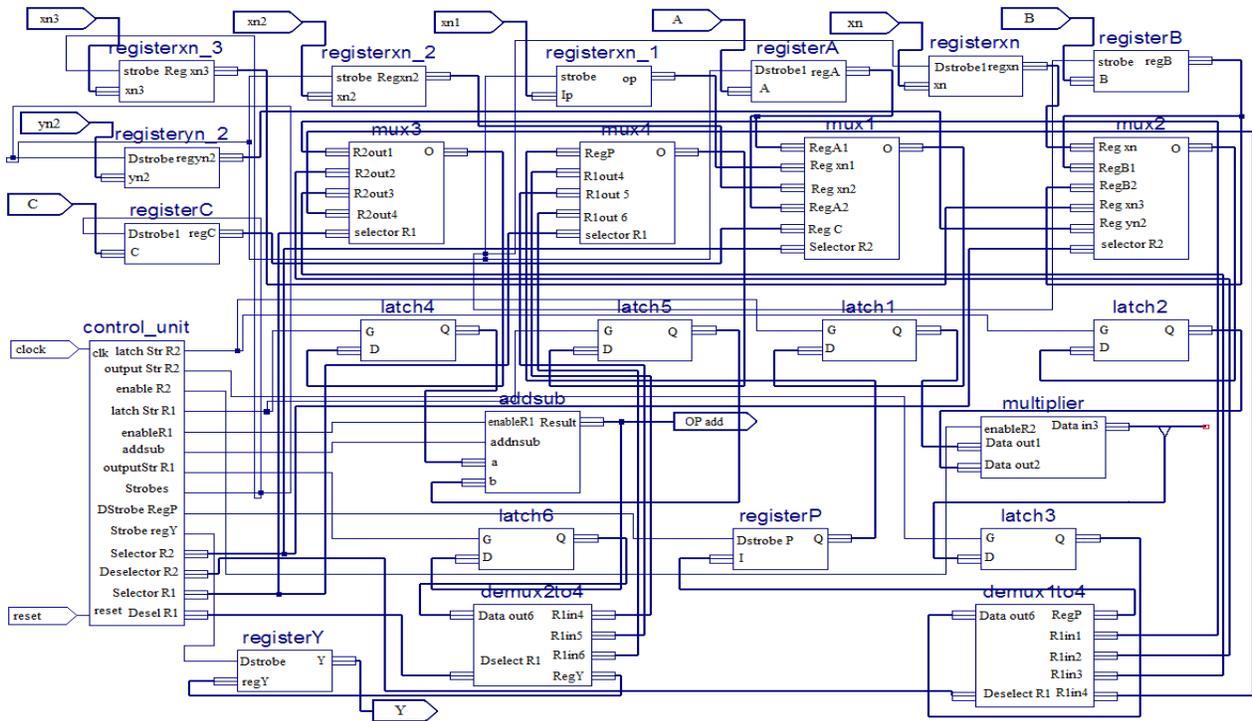
            .....

            .....

```



Schematic view of the designed IIR digital filter in Xilinx ISE tool



The schematic circuit of the designed IIR Butterworth filter in Xilinx ISE tool.

References

- [1] Azeddien M. Sllame, Vladimir Drabek, "An Efficient List-Based Scheduling Algorithm for High-Level Synthesis", Euromicro Symposium on Digital System Design (DSD'02), 2002, pp: 316.
- [2] Srinivas Katkoori, Ranga Vemuri, "Scheduling for Low Power under Resource and Latency Constraints", In Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 2000, pp. 53-56.
- [3] Niraj K. Jha, Lin Zhong, "Interconnect-aware high-level synthesis for low power", International Conference on Computer-Aided Design (ICCAD '02), 2002, pp: 110-117.
- [4] Philippe Grosse, Yves Durand, Paul Feautrier, "Methods for power optimization in SOC-based data flow systems", ACM Transactions on Design Automation of Electronic Systems (TODAES), 2009, vol. 14, issue 3, Article no.: 38.
- [5] De Micheli, G. Synthesis and Optimization of Digital Systems, McGraw-Hill Inc., 2000.
- [6] Robert Schreiber, Shail Aditya, Scott Mahlke, Vinod Kathail, B. Ramakrishna Rau, Darren Cronquist And Mukund Sivaraman, "PICO-NPA: High-Level Synthesis of Nonprogrammable Hardware Accelerators" Journal of VLSI Signal Processing, 2002, pp: 127–142.
- [7] Christian Haubelt, Jurgen Teich, "Accelerating Design Space Exploration Using Pareto-Front Arithmetic's", In Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC'03), Japan, 2003, pp: 525- 531.
- [8] I. Das. A preference ordering among various Pareto optimal alternatives. Structural and Multidisciplinary Optimization, 18(1), 1999, pp: 30–35.

- [9]Alessandro G. Di Nuovo, Maurizio Palesi, Davide Patti, Fuzzy Decision Making in Embedded System Design,” Proceedings of 4th International Conference on Hardware/Software Codesign and System synthesis, 2006, pp: 223-228
- [10]J. C. Gallagher, S. Vigham, and G. Kramer,“A family of compact genetic algorithms for intrinsic evolvable hardware,” IEEE Trans. Evol. Comput., vol. 8, no. 2, 2004, pp. 1–126.
- [11] Vyas Krishnan and Srinivas Katkooari, “A Genetic Algorithm for the Design Space Exploration of Datapaths During High-Level Synthesis, IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, 2006, pp: 213- 229.
- [12] Kirischian, L., Geurkov, V., Kirischian, V. and Terterian, I. ‘Multi-parametric optimisation of the modular computer architecture’, Int. J.Technology, Policy and Management, Vol. 6, No. 3, 2006, pp.327–346.
- [13]E. Torbey and J. Knight, “High-level synthesis of digital circuits using genetic algorithms,” in Proc. Int. Conf. Evol. Comput., 1998, pp.224–229.
- [14]E. Torbey and J. Knight, “Performing scheduling and storage optimization simultaneously using genetic algorithms,” in Proc. IEEE Midwest Symp. Circuits Systems, 1998, pp. 284–287.
- [15] 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, 2007, pp: 733–750.
- [16] Williams, A. C., Brown, A. D. and Zwolinski, M, "Simultaneous Optimisation of Dynamic Power, Area and Delay in Behavioural Synthesis", IEE Proceedings Computers and Digital Techniques, Volume: 147, Issue: 6, 2000, pp: 383-390.

- [17] Christian Haubelt , Thomas Schlichter , Joachim Keinert , Mike Meredith, "Automatic Design Space Exploration and Rapid Prototyping from Behavioral Models", Proceedings of DAC , California, 2008, pp: 580 - 585.
- [18] Philippe Grosse, Yves Durand, Paul Feautrier, "Methods for power optimization in SOC-based data flow systems", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14 , issue 3, 2009, Article no.: 38.
- [19] J. C. Gallagher, S. Vigham, and G. Kramer "A family of compact genetic algorithms for intrinsic evolvable hardware," *IEEE Trans. Evolutionary Computation.*, volume 8, no. 2, 2004, pp: 111–126.
- [20] C. Mandal, P. P. Chakrabarti, and S. Ghose, "GABIND: A GA approach to allocation and binding for the high-level synthesis of data paths," *IEEE Transaction on VLSI*, vol. 8, no. 5, 2000, pp: 747–750.
- [21] M. J. M. Heijlingers, L. J. M. Cluitmans, and J. A. G. Jess, "High-level synthesis scheduling and allocation using genetic algorithms," in *Proc.Asia South Pacific Design Automation Conf.*, 1995, pp: 61–66.
- [22] M. K. Dhodhi, F. H. Hielscher, R. H. Storer, and J. Bhasker, "Datapath synthesis using a problem-space genetic algorithm," *IEEE Trans.Comput.-Aided Des.*, volume 14, 1995, pp: 934–944.
- [23] Saraju P. Mohanty, Nagarajan Ranganathan, Elias Kougiianos and Priyadarsan Patra, "Low-Power High-Level Synthesis for Nanoscale CMOS Circuits" Chapter- High-Level Synthesis Fundamentals, *Springer US*, 2008.
- [24] D. Gajski, N. Dutt, A.Wu, and S. Lin, High Level Synthesis: "Introduction to Chip and System Design". Norwell, MA: *Kluwer*, 1992.

- [25] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASICs," *IEEE Trans. Comput.-Aided Des.*, volume 8, no.6, 1989, pages: 661–679.
- [26] McFarland, M.C. Parker, A.C. Camposano, R. "The high-level synthesis of digital systems", *Proceedings of the IEEE*, Volume: 78, Issue: 2, 1990, pp: 301-318
- [27] Anirban Sengupta, "A Fast Design Space Exploration Based on Priority Factor for a Multi Parametric Optimized High Level Synthesis Design Flow", *Master of Applied Science (M.A.Sc) Thesis, Ryerson University, Toronto, Canada*, 2010
- [28] Zadeh, L.A. "Fuzzy sets". *Information and Control*, volume 8, issue 3, 1965, pp. 338–353
- [29] T. Blickle and L. Thiele, "A mathematical analysis of tournament selection," in *Proceedings of 6th International Conference on Genetic Algorithms*, 1995, pp: 9–16.
- [30] Jain, R., Panda, P.R.: An efficient pipelined VLSI architecture for lifting-based 2d-discrete wavelet transform. In: *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, 2007, pp. 1377– 1380.
- [31] Antola, A., Ferrandi, F., Piuri, V., Sami, M.: Semiconcurrent error detection in data paths. *IEEE Transactions on Computers*, volume. 50, issue 5, 2001, pp. 449– 465.
- [32] Express: High-Level Synthesis Benchmarks. <http://express.ece.ucsb.edu/benchmark/>
- [33] <http://www.cbl.ncsu.edu/benchmarks/>, 2007.
- [34] [http://msdn.microsoft.com/en-us/library/aa288436\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/aa288436(VS.71).aspx), 2003
- [35] <http://msdn.microsoft.com/en-us/library/ms228593.aspx>, 2003
- [36] Shawn Mccloud, "High Level Synthesis Report 2011", Mentor Graphics, 2011
- [37] Shawn Mccloud, "High Level Synthesis Report 2011", Calypto Design Systems, 2011
- [38] <http://www.synopsys.com/Systems/BlockDesign/HLS/Pages/>, 2012
- [39] <http://www.synopsys.com/systems/blockdesign/hls/pages/>, 2012

- [40] <http://www.mentor.com/esl/>, 2012
- [41] Das, S. and Khatri, S. P. 2008. Resource sharing among mutually exclusive sum-of-product blocks for area reduction. ACM Trans. Des. Autom. Electron. Syst. volume 13, issue 3, 2008, Article 51.
- [42] M. Geilen, T. Basten, B. Theelen, and R. Otten. An algebra of Pareto points. In Proc. of 5th IEEE International Conference on Application of Concurrency to System Design (ACSD), 2005, pages 88–97.
- [43] Roman, C. , Evenly distributed pareto points in multi-objective optimal power flow, IEEE Transactions on Power Systems, Volume: 21 , Issue: 2, 2006, pp: 1011 - 1012.
- [44] T. Givargis, F. Vahid, and J. Henkel. System-level exploration for Pareto-optimal configurations in parameterized system-on-a-chip. IEEE Trans. VLSI Syst., volume 10, issue 4, 2002, pp. 416–422.
- [45] A. Baykasoglu, S. Owen, and N. Gindy. A taboo search based approach to find the Pareto optimal set in multiple objective optimisation. Journ. of Engin. Optimization, volume 31, 1999, pp. 731–748.
- [46] M. Voorneveld. Characterization of Pareto dominance. Operations Research Letters, volume 31, issue. 1, 2003, pp. 7–11.
- [47] Rajarshi Mukherjee , Seda Ogresci Memik , Gokhan Memik, Temperature-Aware Resource Allocation and Binding in High-Level Synthesis, Proc. of 42nd Design Automation Conf, 2005, pp. 196-201
- [48] Thermal Performance Challenges from Silicon to Systems – Ram Viswanath, Vijay Wakharkar, Abhay Watwe, Vassou Lebonheur, Manufacturing Group, Intel Corp, 2000.

- [49] Krum, A., Thermal Management, in The CRC Handbook of Thermal Engineering, F. Kreith, Editor, CRC Press: Boca Raton, 2000.
- [50] Mukherjee, R. , Memik, S. O. , An Integrated Approach to Thermal Management in High-Level Synthesis, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume: 14 , Issue: 11 , 2006, pp: 1165 – 1174
- [51] S. Mohanty, V. K. Prasanna, S. Neema, J. Davis, Rapid Design Space Exploration of Heterogeneous Embedded Systems using Symbolic Search and Multi-Granular Simulation, Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems, 2002, pp. 18 – 27.
- [52] M. Auguin, L. Capella, F. Cuesta, and E. Gresset, “CODEF: A System Level Design Space Exploration Tool,” Intl. Conf. on Acoustics, Speech, and Signal Processing, volume 2, 2001, pp. 1145- 1148.
- [53] A. Baghdadi, N-E. Zergainoh, W. Cesario, T. Roudier, and A. Jerraya, “Design Space Exploration for Hardware/Software Codesign of Multiprocessor Systems,” Intl. Workshop on Rapid System Prototyping, 2000, pp. 8.
- [54] H. J. Eikerling, W. Hardt, J. Gerlach, and W. Rosenstiel, “A Methodology for Rapid Analysis and Optimization of Embedded Systems,” Symposium on Engineering of Computer Based Systems, 1996, pp: 252- 259.
- [55] P. Lieverse, P. van der Wolf, E. Deprettere, and K. Vissers, “A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems,” Workshop on Signal Processing Systems, Volume 29 Issue 3, 2001, pp. 197 - 207.

- [56] S. Mohanty, S. Choi, J. Jang, and V. K. Prasanna, "A Model-based Methodology for Application Specific Energy Efficient Datapath Design using FPGAs," Proceedings of Application-specific Systems Architectures and Processors, 200, pp: 76 - 87.
- [57] Kanishka Lahiri , Anand Raghunathan , Sujit Dey, Efficient exploration of the SoC communication architecture design space, Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design, 2000, pp 424-430.
- [58] Javaid, H. , Rapid Design Space Exploration of Application Specific Heterogeneous Pipelined Multiprocessor Systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2010, pp: 1777 – 1789.
- [59] Stijn Eyerman , Lieven Eeckhout , Koen De Bosschere, Efficient design space exploration of high performance embedded out-of-order processors, Proceedings of the conference on Design, automation and test in Europe: Proceedings, 2006, pp. 351 – 356.
- [60] V. Srinivasan , S. Radhakrishnan , R. Vemuri, Hardware/software partitioning with integrated hardware design space exploration, Proceedings of the conference on Design, automation and test in Europe, 1998, pp.28-35.
- [61] Hong Shin Jun; Sun Young Hwang, Automatic Synthesis of Pipeline Structures with Variable Data Initiation Intervals, 31st Conference on Design Automation, 1994. pp: 537 - 541
- [62] Imed Eddine Bennour and El Mostapha Albouhamid, Lower bounds on the iteration time and the initiation interval of functional pipelining and loop folding, Design Automation for Embedded Systems, Springer, Volume 1, Number 4, 1996, pp. 333-355
- [63] Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Introduction to Algorithms (1st ed.). MIT Press and McGraw-Hill. ISBN 0-262-03141-8. 1990.
- [64] <http://www.altera.com/literature/an/an531.pdf>