A Rapid Design Space Exploration Approach for Multi-Objective Optimization of DSP Filter Designs

by

Aakriti Tarun Sharma

Bachelor of Technology, Electrical Engineering,

Delhi Technological University, India, 2016

A project

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Engineering

in the program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2018

A Rapid Design Space Exploration Approach for Multi-Objective Optimization of DSP Filter Designs

Master of Engineering 2018

Aakriti Tarun Sharma

Electrical and Computer Engineering

Ryerson University

Abstract

The process of converting a behavioral specification of an application to its equivalent system architecture is referred to as High Level-Synthesis(HLS). A crucial stage in embedded systems design involves finding the trade off between resource utilization and performance. An exhaustive search would yield the required results, but would take a huge amount of time to arrive at the solution even for smaller designs. This would result in a high time complexity. We employ the use of Design Space Exploration (DSE) in order to reduce the complexity of the design space and to reach the desired results in less time.

In reality, there are multiple constraints defined by the user that need to be satisfied simultaneously. Thus, the nature of the task at hand is referred to as Multi-Objective Optimization. In this thesis, the design process of DSP benchmarks was analyzed based on user defined constraints such as power and execution time. The analyzed outcome was compared with the existing approaches in DSE and an optimal design solution was derived in a shorter time period.

Acknowledgements

I would like to acknowledge and thank my supervisor Dr. Reza Sedaghat for giving me the opportunity of joining OPR-AL in order to carry out this research project and for being a constant source of motivation during my time here.

I would also like to thank Patrick Siddaavaatam, whose expertise, and guidance helped me in working on my thesis. I would like to thank him for finding time out of his busy schedule to advise me throughout the research,thus helping me gain a better understanding in this domain.

I am grateful for my friends and fellow members of OPR-AL, Dimple Gamnani and Meher Bhagat for helping me immensely in refining the thesis and for the countless memories of the times we spent together in the lab.

Finally, I would like to thank my entire family and my friends circle for being the best support system in the world. This research project could not have been materialized without their encouragement and their faith in me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The major hurdle for Systems Design Engineers in the development of the latest embedded systems is to minimize the complexity of the design space. This can be achieved once a set of realistic solutions that meet all the performance objectives are met. Upon incorporating more silicon per unit area, the complexity of the design space increases exponentially. The goal is to optimize multiple parameters simultaneously whilst keeping in mind the user-defined constraints and specifications. Most often the multiple objectives that need to be satisfied are - power consumed, area occupied by the resources and the execution speed of the resources. In general, while designing the VLSI circuits, each parameter is a trade-off above the other as per the priority of the consumer. An ideal system is one, which can be implemented on a very small chip area that runs very fast and consumes the least power. The development of an embedded system undergoes the process of physical realization from its conceptualization through a sequence of stages where at each step, more information is imparted than the preceding stage. Two broad steps combine the design flow in VLSI systems: Front end design and Back-end design. Front End Design comprises of stages where the manufacturing takes place in software and verification of the functionality. These stages are as follows: Design Specification, Behavioral Description and RTL Verification. The stages in Back End design consist of making the Gate Level Net-list and the Physical Layout. Figure 1 shows a block diagram of the VLSI design flow.

Electronic System Level Synthesis is the process of rendering a digital hardware from the algorithmic description of the desired behavior and implementing it. The most important stage in the whole flow of design is performing Design Space Exploration (DSE) in order to make the randomly arranged design space into a partially arranged one. This happens because the process of DSE aims at satisfying most of the system constraints thus resulting in the best candidate architecture. Since this design process of the HLS flow is complicated, it can only be performed by the system design architects. An organized methodology needs to be employed to find the best candidate architecture from the ever expanding design space. Care needs to be taken in terms of time and design complexity while undergoing these

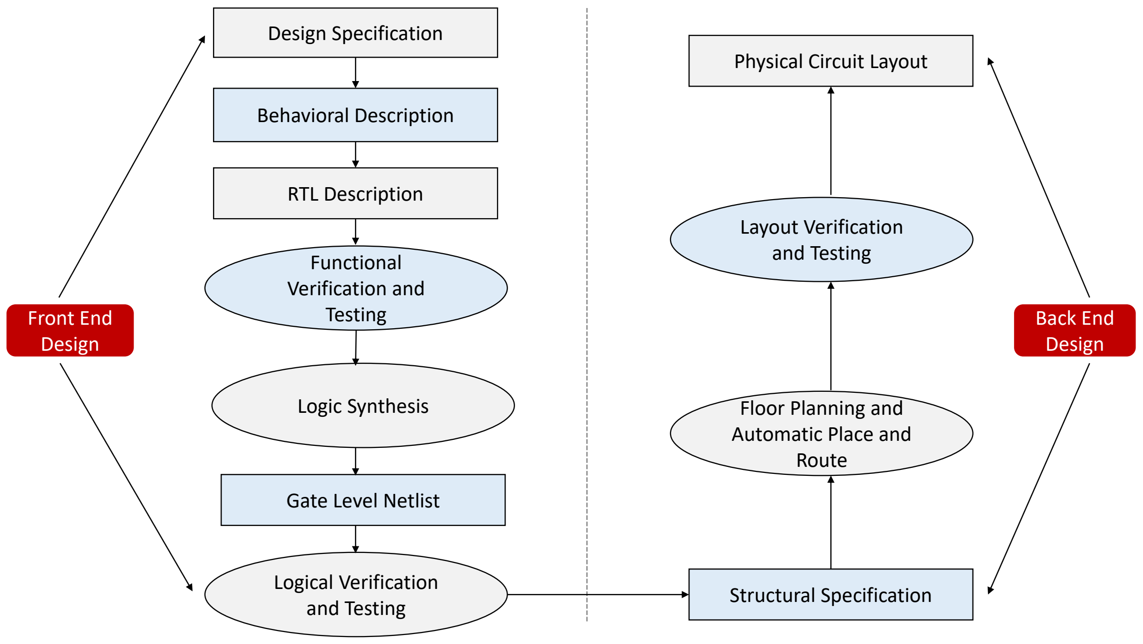


Figure 1: *ASIC Design Flow*

procedures. A Rapid Design Space Exploration Approach for Multi-Objective Optimization of DSP Filter Designs is presented in this thesis. An analysis is made of the existing procedures the Pareto Front method[5], and the evolutionary algorithm procedures[9],[6],[15] that are used to find the best design architecture and reducing the number of resources used per unit area.

1.2 Related Works

One of the earliest methods researchers introduced in order to analyze and manage the trade-offs between contradictory design space objectives was that of achieving the pareto optimal set. In [5] the authors use Pareto Front Arithmetic (PFA) to explore giant search spaces. The Pareto Optimal solutions were scrutinized in their method by utilizing the hierarchical problem structure. Based on their analysis, Piccolboni et al. proposed COSMOS in their research [11], where they implemented an approach for the exploration of complex accelerators that were made of multiple components by correlating the memory optimization tools and HLS. In [8], the authors used Architecture Configuration Graphs (ACG) for optimization of performance parameters and variant analysis to regulate the design space. In [14], the authors use Simulated Annealing to determine the optimal configuration of the resources. As research progressed into the domains of HLS and DSE, the use of evolutionary algorithms was proposed. Algorithms such as the Genetic Algorithms [9] and Particle Swarm Optimization [6] were used. Algorithms such as Simulated Annealing and Genetic Algorithms, although proved to be quite efficient often faced the problems of getting stuck in the local minima and being unable to find the global optimum solution.

To resolve the problem of the algorithm getting stuck in the overlapping domains, the authors in [12] used the concept of Stability in Competition first introduced by Hotelier [8]. The nature of the multi-objective optimization problem is a Multi-Dimensional Multi-knapsack problem, also addressed in [12]. However, for us, since the basic constraints that we need to satisfy are just two, we use the solution of the Bi-Dimensional Knapsack problem as described in [10]. An analysis is made of the various methods used for solving these multi objective optimization algorithms in the succeeding chapters.

1.3 Motivation of Research

There has been a significant progress in the advancement of Register Transfer Level (RTL) Synthesis development. With an ever increasing dependence on technology, researchers are attempting to automate the synthesis at higher level of the design hierarchy.The main motivation for carrying out this research was to evaluate the different methodologies used to carry out the DSE procedure and how these methodologies behave with different DSP filters, such as the Infinite Impulse Response Filter (IIR), the Finite Impulse Response Filter(FIR), Elliptic Wave Filter (EWF), Bandpass Filter (BPF) and the AutoRegressive Filter (ARF).

The various techniques investigated in this research are capable of generating numerous designs from the same specifications in a short amount of time. Thus for the developers, a settlement can be accomplished between the cost, area, power, speed etc, according to what the user desires for their application. In accelerating the design procedures, the companies manufacturing these chips can lower their development costs. Upon verification, if found that the initial design and the final outcome are corresponding to each other then that would mean less debugging time and fewer errors for the new chips.Hence, it is imperative to explore the various methodologies and discuss the ones most suited for Multi-Objective Optimization procedures.

1.4 Thesis Organization

The thesis has been arranged in the following manner: Chapter 2 discusses the concepts of High Level Synthesis (HLS), Design Space Exploration (DSE), Stability in Competition, Knapsack Problems and Evolutionary Algorithms. In Chapter 3, the design space is created with the initialization of all the resources. All the constraints and user defined specifications are also stated in this section. Chapter 4 describes the Priority Factor approach to solve the multi-objective optimization problem. A set of optimum design variants is achieved in this section and the data flow graph is made. Chapter 5 elaborates on the exploration process used by Genetic Algorithms to accelerate the search. The end of this chapter discusses the drawbacks of using Genetic Algorithms which are overcome by using the Particle Swarm Optimization algorithm reviewed in Chapter 6. The mathematical formulation is described and the best set of resource configuration is found using this method. The HLS flow is materialized in chapter 7 using Scheduling, Allocation and Binding. In Chapter 8, the results are discussed and analyzed for different DSP benchmarks. The thesis is conferred by discussing the Conclusion and Future work.

Chapter 2

Preliminaries

The context of the framework of work done in this thesis, is provided in this chapter. Sufficient background for High Level Synthesis, Design Space Exploration, Stability in Competition, Knapsack Problems and Evolutionary Algorithms are addressed and presented.

2.1 High Level Synthesis

The process of conversion of a high level (algorithmic) language to an RTL net list (Behavioral Language) is known as High Level Synthesis. It is also sometimes accredited names such as: Behavioral Synthesis, Algorithmic Synthesis, C Synthesis or Electronic System-Level (ESL) Synthesis. An example of the procedure is as follows - Conversion of the input (a language such as C) to a description language (VHDL/Verilog). The tools required for the successful termination of this process are logic networks and state diagrams. There are however, many constraints that need to be kept in mind such as the delay and area constraints. An array of input specification languages was examined initially but the applications in the market accept synthesizable subsets of languages such as MATALB/C/C++ etc. A gate can be manufactured using a logic synthesis tool after a Register Transfer Level (RTL) Hardware Description Language (HDL) is generated from the high level input language by thorough analysis given the user specified constraints and scheduling times given.

In the last decade, a quantum leap has been noticed in technology especially in Very Large Scale Integration (VLSI) electronics industry. This apparent shift endorsed new manufacturing technologies such as 3D integration [13],[3] and FinFET [7],[2] along with the heterogeneous Systems-On-Chip (SoC) design [4]. However, though, despite the advancements in this sector, no comparable improvement was found for the RTL description languages, logic synthesis, circuit simulation and automatic place and route. However, the system design complexity increases each day due to limited available silicon chip area. Thus in order to address this problem of complexity, researchers rely on Design Space Exploration for better HLS designs.

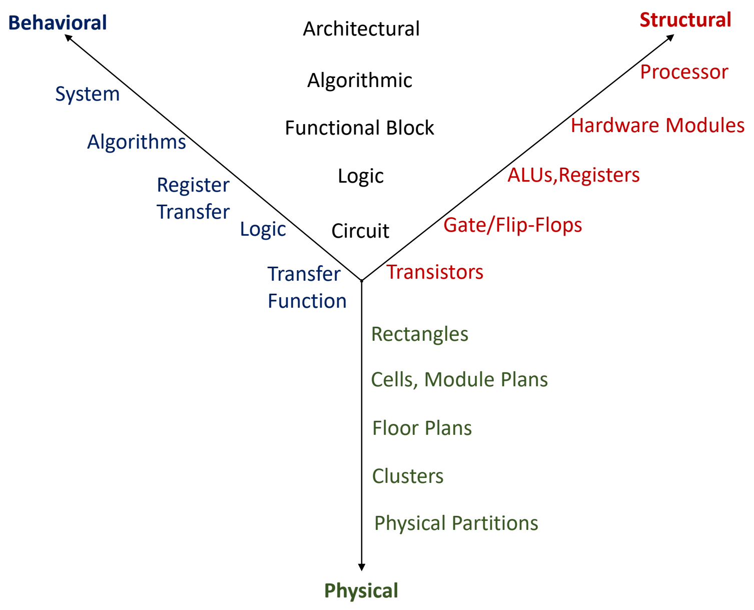


Figure 2: *The Gajski-Kuhn Chart*

2.1.1 Y Chart in VLSI Hardware Design

Different aspects of hardware design in VLSI can be shown in the Gajski-Kuhn chart, also known as the Y-diagram. Developed in 1983 by Daniel Gajski and Robert Kuhn and refined in 1985 by Robert Walker and Donald Thomas, this is now a widely used chart in the augmentation of integrated circuits. Hardware development is usually a top-down design problem that can be anticipated in three domains. These three domains can be shown as three different axes that produce a Y (the result). The degree of abstraction are described by the abstraction levels. Thus the layout goes top-down to further detailed abstraction levels. In general, the design process does not follow any one sequence of the diagram. As a designer, one has the freedom to switch between the three domains to see one view or another. The outer shells are generalizations and the inner shells are the refinements.

The primary properties of the electronic system are determined on the system level of the chart. To make the abstractions of the signals and their time responses in the behavioral description, block diagrams are used. These are memory chip, central processing unit (CPUs) etc. Blocks like the Arithmetic Logic Unit (ALUs) are used in the structural domain. The algorithmic level consists of the algorithms that define the loops, classes, signals, assignments and variables) while the logic level defines the Boolean equations. An even more descriptive level of abstraction is the register-transfer level (RTL). Here a detailed explanation of the logic units and the communicating registers is provided and the flow of data is determined. The design step of the floorplan is located in the geometric view.

2.2 Design Space Exploration

Depending on the design constraints and the user defined specifications, the analysis of eliminating the unwanted design points is known a Design Space Exploration (DSE). In electronics and embedded systems design, there is an overabundance of the choices of design based on what algorithms to choose, how many resources are needed to perform an operation, how many connections per resource etc. As a result, there is a need to have an analysis that is systematic and helps complete the exploration process without much time and factor complexity.

The basis of DSE is a trade off between two competing resources or parameters of interest. The most commonly used parameters of interest include power consumption by the resources, execution time of the resources, area occupied and the cost of the resources. For other smaller systems, we may or may not have other constraints needed to be satisfied such as weight, size, shape and energy dissipated. Since the process of exploration can get too intensive, researchers aim at finding an automated DSE where a software can complete this process, take a decision and come up with the optimal solution.

In order to cater to larger and more complex systems, the objective of the researchers is to raise the abstractions of the component and system definition. Many tasks in engineering such as system integration, chip optimization and rapid prototyping is achieved by DSE. It is responsible for organizing the design space in such a way that it becomes partially arranged.

2.3 Multi-Objective Optimization

A combinatorial class of NP complex problems that have more than one objective that needs to be satisfied simultaneously is known as Multi-Objective Optimization. As described in [10], the objectives that need to be satiated simultaneously are contradictory and thus lead to sub optimality and solutions in-feasibility.

Commonly used in combinatorial optimization, the knapsack problem is one which was conceptualized out of the idea of a person who has a fixed sized traveling knapsack and he/she can only fit in limited number of items without exceeding the capacity of this rucksack as shown in Figure 3. For a hiker, for example, there are various things that he/she may need to fit in their knapsack for their trip. The aim is to fit in as many things as possible into the bag without making it too heavy for the hiker to carry. Only the essential items make it into the bag. The same concept is used in hardware design where designers want to maximize the amount of resources, but without exceeding the maximum capacity.

The knapsack problem finds its applications in applied mathematics, cryptography, computer science applications and combinatorics. The 0/1 knapsack problem can be described as:

$$(0/1)\begin{cases} maximize \sum_{i=1}^{n} p_j x_j \\ st \sum_{j=1}^{n} w_{iz} \leq C_i, \\ x_j \varepsilon \{0,1\} \end{cases} \tag{2.1}$$

Figure 3: *An Example of the Multi-Knapsack Problem*

A multi knapsack problem (MKP) is similar to the 0/1 knapsack problem with the exception that it has multiple knapsacks. The formulation for which is given below:

$$(MKP)\begin{cases} maximize \sum_{i=1}^{n} p_j x_j \\ st \sum_{j=1}^{n} w_{iz} \leq C_i, i = 1, 2...m \\ x_j \varepsilon \left\{0, 1\right\}, j = 1, 2...n \end{cases} \tag{2.2}$$

However, since researchers mostly deal with two constraints Power consumed per unit area and Execution time, the problem of multi objective optimization can be modeled as a Two Dimensional Knapsack Problem, as done in [12]. It can be worked out as follows:

$$(BKP)\begin{cases} maximize \sum_{i=1}^{n} p_j x_j \\ st \sum_{j=1}^{n} w_{iz} \leq C_i, i = 1, 2 \\ x_j \varepsilon \left\{0, 1\right\}, j = 1, 2...n \end{cases} \tag{2.3}$$

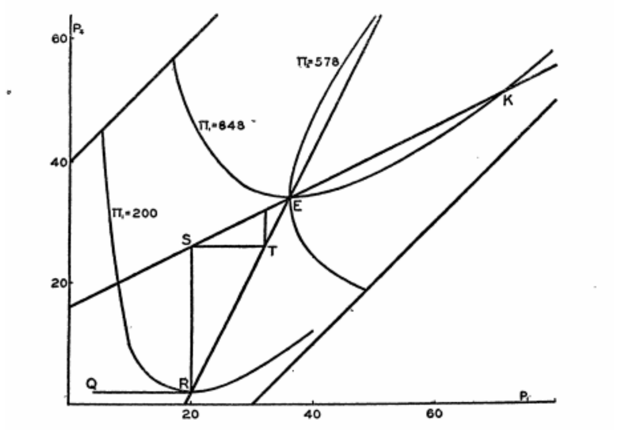


Figure 4: *Graph portraying the principle of Stability in Competition between two competing businessmen*

2.4 Stability in Competition

Proposed by Harold Hoteling in 1929 [6], Stability in Competition closely observes the relationship between two competing sellers and how they decrease or increase their supply of commodities as per the demand by the consumers in the market. The behavior between two competing variables until the point of stability is reached can be shown in Figure 4. Here each variable counter balances the others cumulative weight.

Between two competing buyers: A and B, if A is the smarter seller, they would decrease their costs first thus increasing their profits. In the graph, it can be seen as the horizontal move to point R. Seeing this B would lower their prices too, hence increasing their profits which can be seen as a vertical motion to point S. Then A would further lower their prices, following which B would do the same. This would go on and so forth, till point E or the point of stability is reached. This is also referred to as the point of equilibrium. Beyond this point, none of the two sellers would be able to decrease their costs without facing substantial losses to their business.

In the analysis of DSE, when an algorithm is under way looking for the most optimal resource configurations, there may be areas where two resources are mapped simultaneously at the same points or too close to each other, consequently leading to overlapping domains.

There is a major possibility of the algorithms to get stuck in the overlapping domains and keep searching within them, resulting in the problem of getting stuck in the local minima.

Using the concept of Stability in Competition in our analysis would help us to get rid of the overlapping domains and look for the global best solution while performing Design Space Exploration in High Level Synthesis.

Chapter 3

Design Space Exploration Framework

3.1 Creating a Randomly Arranged Design Space

Embedded systems and electronics are generally comprised of clock oscillators, hardware resources such as adders/subtractors, multipliers and dividers, software modules, interconnecting logics including the power distribution architecture, buses, multiplexers and demultiplexers. In this chapter, the framework for Design Space exploration is discussed and analyzed for the three different methods the Priority Factor Method, the Genetic Algorithm method and the Particle Swarm Method for exploration in order to find the resource set within the power constraints. It is essential to use these methodologies as reaching the solutions of the optimum design space randomly would not only be an exhaustive but also a cumbersome, time-consuming procedure. For larger applications, the design space grows exponentially due to the combinatorial nature of the DSE problems. Thus, this must most definitely be avoided. The work presented in this chapter shows that the solution front can be kept within the feasible regions but also must be made to avoid overlaps, using concepts such as Stability in Competition.

A vague search through the randomly ordered sets of resources renders the design space partially ordered [12], as can be seen from the Figure 5.

The intent of randomly ordering the design space before analysis has two reasons to lower the time complexity and to increase the utility of the solution. This is achieved when in an ordered design space, the solution is searched only within it. When the algorithm is confined from exploring the redundant configurations of the design space, less time is wasted and the complexity does not increase exponentially.

3.2 Commencement of the Design Flow

The first step in High Level Synthesis Design is to input the Design Specifications. In order to optimize the parameters of interest, the consumer needs to define the associated data structure of the end appli-

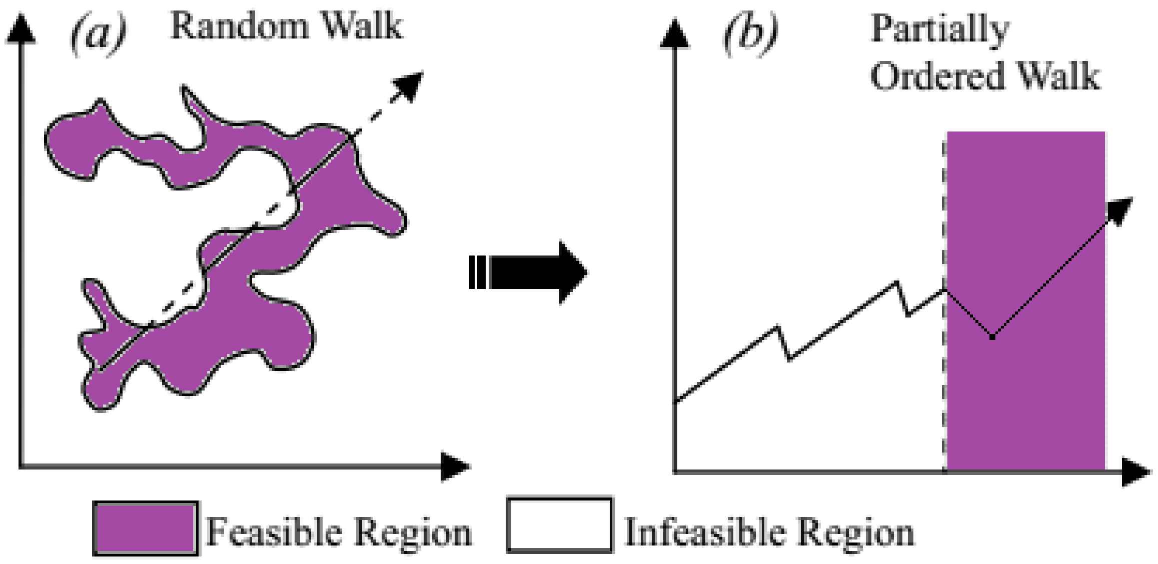


Figure 5: *: Hypothetical Search through (a) Unordered and (b) Partially Ordered Design Space*

cation to be used. A clear characterization of the systems constraints must be stated to the designer. Not only do these specifications and constraints govern the organization of the micro-architecture, but also serve as a guide for the design and development costs. In the analysis carried out in this thesis the following assumptions were made.

1. Maximum power consumption: 350 (mW)
2. Maximum resources available for the system design:
   (a) 2 Adder/subtractor Units.
   (b) 4 Multiplier Units
   (c) 2 Divider Units
   (d) 3 Clock Frequency Oscillators: 100, 150 and 200 MHz
3. Maximum time of execution: 62 ms

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

1. No of clock cycles needed for multiplier to finish each operation: 4 cc
2. No of clock cycles needed for divider to finish each operation: 4 cc

3. No of clock cycles needed by the Adder/subtractor: 2 cc
4. Area occupied by each adder/subtractor: 10 a.u. on the chip. (e.g. 10 CLB on an FPGA)
5. Area occupied by each Multiplier: 45 a.u. on the chip. (e.g. 45 CLB on an FPGA)
6. Area occupied by each divider: 55 a.u. on the chip. (e.g. 55 CLB on an FPGA)
7. Area occupied by the 100MHz clock oscillator: 5 area units
8. Area occupied by the 150 MHz clock oscillator: 6 area units
9. Area occupied by the 200 MHz clock oscillator: 10 area units

3.3 Problem Description

The behavioral description stage in High Level Synthesis Design is designated by the mathematical model of the end application the designers are trying to manufacture. From this model, we can evaluate the data dependency and the input/output relationship of the system. For analysis pertaining to this thesis, we use an Elliptic Wave Filter with transfer function:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.07819 - 0.221z^{-1} + 0.214z^{-2}}{1 - 2.54z^{-1} + 2.105z^{-2}} \tag{3.1}$$

Using convolution, we can convert the transfer function from its frequency domain to the time domain as follows:

$$y(n) = 0.07819x(n) - 0.221x(n-1) + 0.214x(n-2) + 2.54y(n-1) - 2.105y(n-2) \tag{3.2}$$

For the sake of simplicity of our analysis we assume the following:

1. $A = 0.07819$
2. $B = 0.221$
3. $C = 0.214$
4. $D = 2.54$
5. $E = 2.105$

So, the equation now becomes:

$$y(n) = Ax(n) - Bx(n-1) + Cx(n-2) + Dy(n-1) - Ey(n-2) \tag{3.3}$$

This case study is used in our analysis and an attempt is made to find the most optimum resource configuration for this particular filter.

Chapter 4

Analysis of Exploration of the Design Space using Priority Factor Method

4.1 Priority Factor Background

In [10], the authors solve the problem of Multi-Objective Optimization using the Priority factor method derived from [15]. A measurement of the resource contribution to the objective cost function defines the priority factor of that particular system. It can be devised as:

$$PF = \frac{\Delta N_{(Rn)} K_{(Rn)}}{N_{(Rn)}} \tag{4.1}$$

$\Delta N_{Rn} K_{Rn}$ refers to the contribution of the nth resource and K_{Rn} is the absolute per unit performance of the resource.

4.1.1 Mathematical Formulation of Priority Factor (Area)

Given a set of resources, such as the adders/subtractors, multipliers, dividers and clock oscillators, the total area occupied by all of these resources is simply the sum of all of the areas combined.

$$A = \Sigma A\left(R_i\right) + A\left(R_{CLk}\right) \tag{4.2}$$

Where A area occupied of the resources and R_i Resources used in systems designs

On taking the partial derivatives of the above equation we get:

$$\frac{\partial A}{\partial N_{R1}} = \frac{\partial \Sigma \left(N_{R1}K_{R1} + N_{R2}K_{R2} + ... + N_{RN}K_{RN} + (AR)(clk)\right)}{\partial N_{RN}} = K_{(R1)} \tag{4.3}$$

On finding the partial derivatives for all the available resources, we can find the absolute per unit performance for them all. Substituting this back in the equation, we can find the Priority Factors in terms of area occupied for all of the resources.

4.1.2 Mathematical Formulation of the Priority Factor (Power)

The total power consumption on a particular chip is given in terms of the area and the power consumed by them at a particular clock frequency.

$$P = \Sigma \left(N_{ri}K_{ri}\right)(\,p_c) \tag{4.4}$$

Where P is the total power consumed,
N_{ri} is the number of resources and
K_{ri} is the absolute per unit contribution of those resources.
P_c is the power consumed per area unit at a particular frequency of operation.
It can also be written as the following:

$$P = (N_{R1}K_{R1} + N_{R2}K_{R2} + ... + N_{RN}K_{RN})p_c \tag{4.5}$$

On calculating the partial derivative of this equation with respect to these resources, we can find the priority factors for power consumption as per the following equation:

$$PF = \frac{\Delta N_{(Rn)}K_{(Rn)}}{N_{(Rn)}}(p_c)^{max} \tag{4.6}$$

We use the maximum power consumed because we had considered maximum clock frequency. As a result, the total power consumed by all of the resources in the system is dependent on the change in the number of the resource.

4.1.3 Mathematical Formulation of the Priority Factor (Execution Time)

Similarly, in terms of time of execution of the resources, the priority factor of the resources can be given as:

$$PF(R_n) = \frac{(\Delta N_{Rn}T_{Rn})}{N_{Rn}}T_p^{\max} \tag{4.7}$$

where T_{Rn} is the time taken by each functional unit to finish operation and
$T_p^{\max}$ is the maximum clock period.

4.2 Analysis of Exploration of Power Consumed in the Design Space using Priority Factor Method

To calculate the priority factor of resources in terms of the power consumed, the following formula is used [12]. For Multiplier Unit:

$$PF(Multiplier) = \frac{\Delta N_{(Rn)} K_{(Rn)}}{N_{(Rn)}} (p_c)^{max} = \frac{(4-1)(45)}{4}(30) = 1012.5mW \tag{4.8}$$

For Adder/Subtractor Unit:

$$PF(Adder/Subtractor) = \frac{\Delta N_{(Rn)} K_{(Rn)}}{N_{(Rn)}} (pc)^{max} = \frac{(2-1)(10)}{2}(30) = 150mW \tag{4.9}$$

For Clock Resources:

$$\frac{\Sigma_{i=1}^{n} N_{Ri} K_{Ri}}{N_{Ri}} \Delta p_c = \frac{(2)(10) + 4(45)}{3}(30-10) = 1333.33mW \tag{4.10}$$

It can be seen from the calculations done above that with the change in the number of resources, there will be a change in the total power consumed by that resource too. There is a significant difference between the priority factors of the adder/subtractor units and that of the multiplier units. The clock however, has the greatest value (2066.66 mW). On arranging them in decreasing order we get,

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

Finding the priority factors of the resources in the circuit, helps in rendering a partially ordered circuit.

4.3 Analysis of Execution Time Exploration of Design Space using Priority Factor Method

To calculate the priority factor of resources in terms of the speed of execution of the operations by the resources, the following formulas are used [12].

For Multiplier Unit:

$$PF(R_n) = \frac{(\Delta N_{Rn} T_{Rn})}{N_{Rn}} T_p^{\max} = \frac{(4-1)(4)}{4}(0.01) = 0.03 \tag{4.11}$$

For Adder/Subtractor Unit:

$$PF(R_n) = \frac{(\Delta N_{Rn} T_{Rn})}{N_{Rn}} T_p^{\max} = \frac{(2-1)(4)}{2}(0.01) = 0.02 \tag{4.12}$$

For Clock Resource:

$$PFClock = \frac{\Sigma_{i=1}^{n} N_{Ri} K_{Ri}}{N_{Ri}} \Delta T_p = \frac{(2)(10) + 4(45)}{3}(0.01 - 0.005) = 0.3333 \tag{4.13}$$

Hence, when the number of resources used for an operation change, a difference in the change in time of execution is also observed. For a change in clock frequency from 100MHz to 200 MHz, there is a substantial change that can be observed as compared with the other resources, i.e., the adder/subtractor units and the multiplier units. On arranging the resources are per the decreasing order of Priority Factors we get that,

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

4.4 Using PF Analysis to order design space

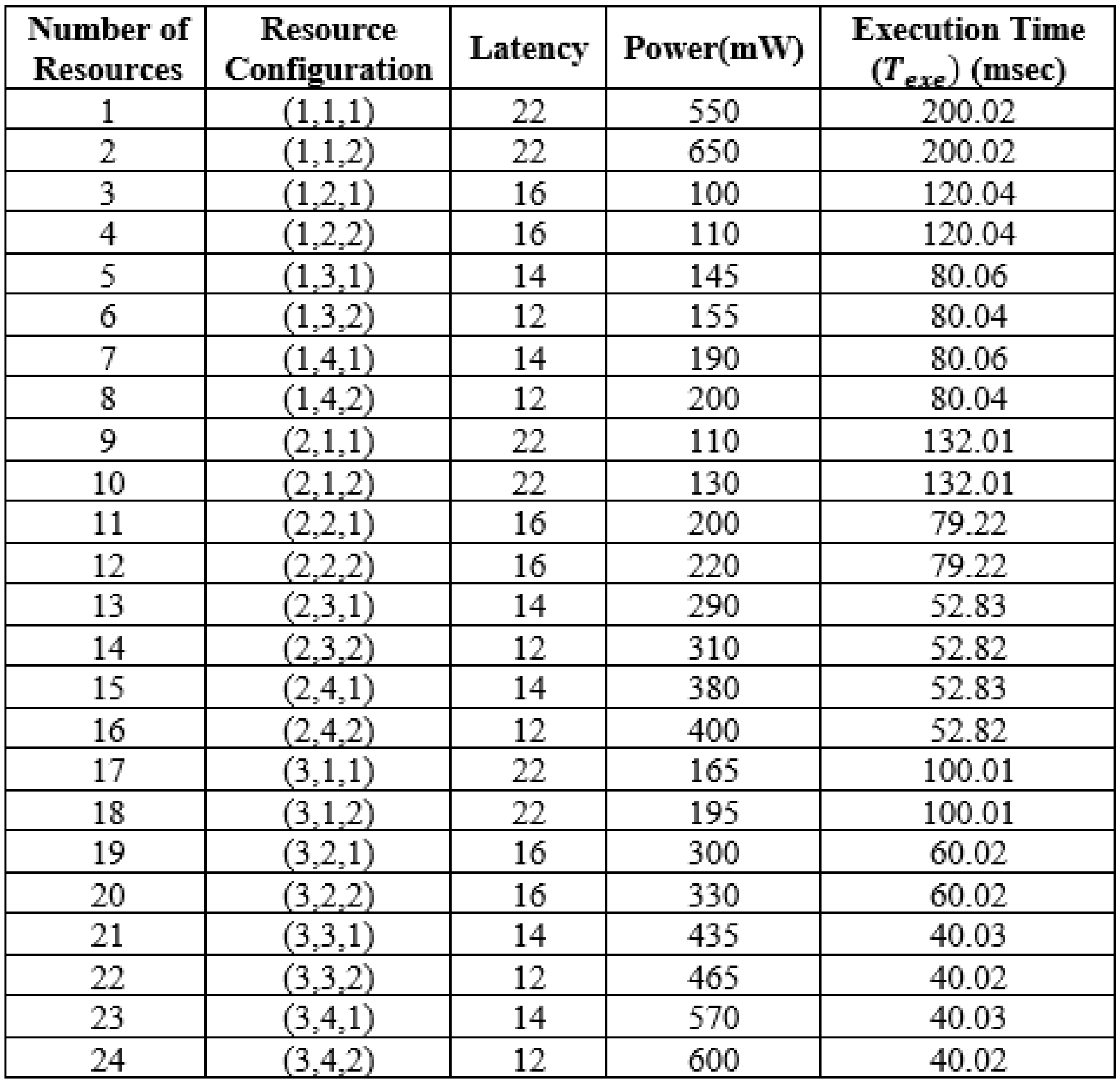
In Systems on Chip designs, the complexity of the design space has been ever increasing. A vital issue, thus, in the design of these systems, is selecting optimal parameters that are agreeable, not only in terms of the technical and behavioral specifications but also with all the user constraints.

A meticulous approach is needed to perform Design Space Exploration, for massive design spaces. When performed at higher levels of abstraction, it is more profitable than at the transistor levels. For any algorithm performing DSE operations, there are two main criteria that need to be met the search space needs to be searched in a short period of time (low time complexity) and achieving a trade off between two conflicting parameters. Table 1 shows all of the possible configurations in the design space that are used for analysis.

4.5 Choosing the most optimal resource configuration

An exhaustive search is used to find the variant best suited for the application at hand. Table 1 depicts a set of all design variants including the clock resources, adder/subtactors and multipliers. The power and speed of execution is then found based on the area occupied by the resources, power consumed by the clock and the Latency and cycle time for each operation. The best design variant is chosen from the whole table based on the restrictions placed on maximum power consumed and maximum time. The maximum power consumed, is given to us as 350mW and the maximum execution time is given as 62msec. Based on these two constraints, we have four sets of variants that are within these limitations.

These are depicted in Table 2.The multiplier given to us occupies more area (45 au) than the adder/subtractors (10 au). Given this situation, where we can choose between one or more sets of variants, the resource configuration having the least number of resources is chosen. Here we choose, variant 19 as the most optimum. It consists of two multipliers and one adder/subtractor at each level.



Number of Resources	Resource Configuration	Latency	Power(mW)	Execution Time (T_{exe}) (msec)
1	(1,1,1)	22	550	200.02
2	(1,1,2)	22	650	200.02
3	(1,2,1)	16	100	120.04
4	(1,2,2)	16	110	120.04
5	(1,3,1)	14	145	80.06
6	(1,3,2)	12	155	80.04
7	(1,4,1)	14	190	80.06
8	(1,4,2)	12	200	80.04
9	(2,1,1)	22	110	132.01
10	(2,1,2)	22	130	132.01
11	(2,2,1)	16	200	79.22
12	(2,2,2)	16	220	79.22
13	(2,3,1)	14	290	52.83
14	(2,3,2)	12	310	52.82
15	(2,4,1)	14	380	52.83
16	(2,4,2)	12	400	52.82
17	(3,1,1)	22	165	100.01
18	(3,1,2)	22	195	100.01
19	(3,2,1)	16	300	60.02
20	(3,2,2)	16	330	60.02
21	(3,3,1)	14	435	40.03
22	(3,3,2)	12	465	40.02
23	(3,4,1)	14	570	40.03
24	(3,4,2)	12	600	40.02

Table 1: Set of all possible Design Variants

Variant Number	**Variant 13**	**Variant 14**	**Variant 19**	**Variant 20**
Resource Configuration	(2,3,1)	(2,3,2)	(3,2,1)	(3,2,2)
Power Consumed	290mW	310mW	300mW	330mW
Execution Time	52.83msec	52.82 msec	60.02msec	60.02msec

Table 2: Sets of Best Resource Variants

Chapter 5

Analysis of Exploration of Design Space using Genetic Algorithms

5.1 Genetic Algorithms: Basics

Invented in 1960s, Genetic Algorithms are based on the theory of evolution that explains the origin of species in nature. Based on the concept of Survival of the Fittest, a theory proposed by Charles Darwin, the algorithm follows that weak and unfit species, in the process of natural selection, die sooner and are only survived by the stronger species. These species go on to become more dominant than the others. Sometimes, the genes of these species may undergo random changes to fit into changing times. If the result of these changes is a stronger gene then over time, it replaces the old one.

Chromosomes are structures that define the nature of the organisms. Each chromosome consists of genes. In Genetic Algorithm processes, these genes are binary, that is, either 0 or 1. Encoding is the process of the mapping techniques between these chromosomes and the solution space. A collection of these chromosomes are referred to as a Population. Over time, as the search matures, the algorithm includes fitter solutions than the ones first initialized eventually finding the set of best solutions.

5.2 Genetic Algorithms for Multi-Objective Optimization

Two main operators are used in Genetic Algorithms to solve problems pertaining to Multi-Objective Optimization:

1. Crossover

2. Mutation

A random set of solutions are initialized in the first step and chromosomes are found. Fitness of all of the chromosomes are found and two of the fittest solutions are chosen to perform the Crossover Operation.

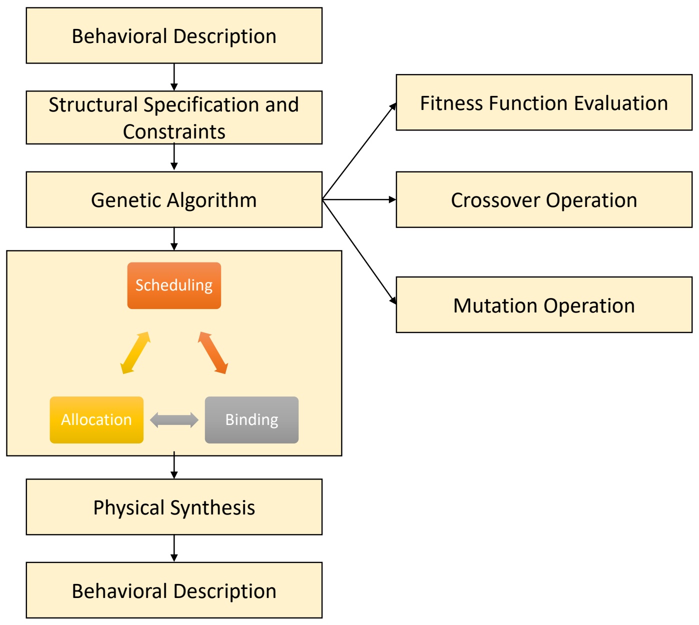


Figure 6: *Flow of the VLSI design flow using Genetic Algorithms*

This is done so that, the Offspring is fitter than the parent chromosomes. This operation is carried out iteratively till the best solution is found. The mutation Operator is used to find a better solution, this process takes place at the gene level and the probability of a gene to undergo mutation is very low. It is also dependent on the chromosome length. This process is important as, just using the crossover operation would lead to two or more similar solutions. Therefore, in order to introduce some diversity in the algorithm for it to move forward, mutation is done and the parents are found for the following iterations. Once the off springs are found, their fitness is calculated. If they are fitter than the parent chromosomes then they become the parent for the next generation else, they are discarded. Figure 6 explains the framework for HLS using Genetic Algorithms. The first step in the analysis of DSE using Genetic Algorithms is to obtain useful genetic operators that facilitate the Genetic Algorithm in reaching optimal or near-optimal solutions. A multi-chromosome representation is used in the analysis to encode chromosomes in the population. The algorithm must guarantee that both scheduling and allocation information is represented.

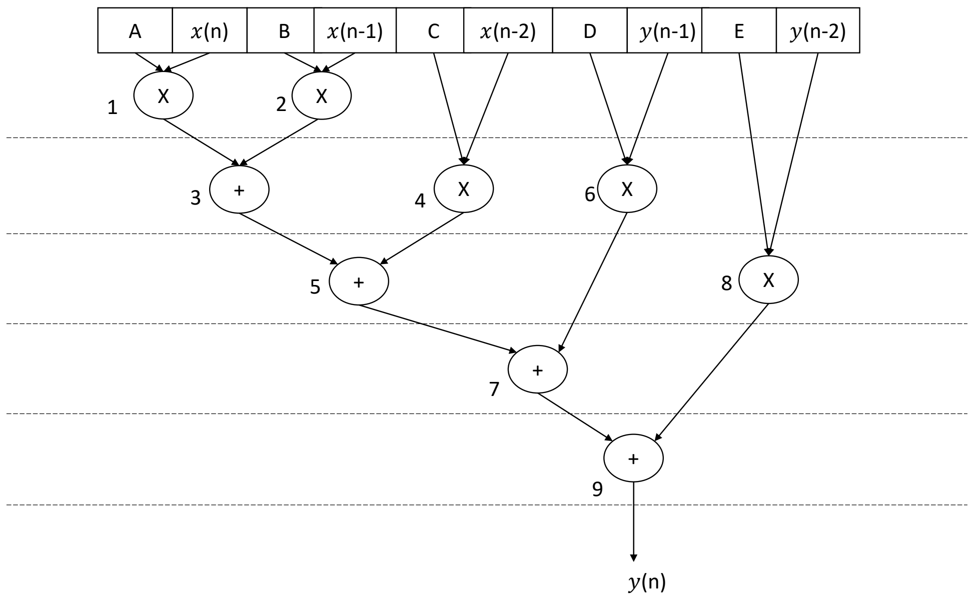


Figure 7: *Chromosome Encoding in Genetic Algorithms*

5.3 Chromosome Encoding

The chromosome is made up of two substrings namely the Node priority field and the Resource Allocation Field. The node priority field portrays an ambiguous representation of the final schedule. It is used to encode a list of tasks that need to be scheduled. The resource allocation field specifies the maximum available resources at each level of the scheduling process. It is essential that an apt problem encoding is chosen so that the algorithm attains convergence. This happens when both scheduling and allocation information are represented. The resource allocation field warrants that each possible combination of the resources are represented. Based on the constraints described in the resource allocation field of the chromosome, a schedule builder can be built to decode the node priority field to acquire a valid schedule. As soon as the operations become available to the functional unit in the next time step, as shown by the node priority field of the chromosome, a schedule binder is used as a list scheduling heuristic. If all the preceding nodes have finished their operation in the previous time step and the resource in which this operation has to take place, is available, then in the node priority field, this operation is said to be ready. Figure 7 shows the Data Flow graph used in Chromosome Encoding in Genetic Algorithms

5.4 Crossover Operations

This section discusses one of the two main operators used in Genetic Algorithms Crossover. Akin to biological reproduction and hereditary, crossover operations are used to diversify the structure of the chromosome from one level to another. Two chromosomes are taken as parent solutions and this operation is applied on them to find the children, for the next generation. The encoded chromosome has two substrings, namely the node priority field and the resource allocation field. Crossover operations for the both of them are summoned at different times. Each child chromosome produced after crossover must have a valid schedule. The precedence constraints established by the data-flow graph are retained in the permutation of tasks in the offspring, since the node priority field is responsible for encoding topologically sorted permutations of tasks. One-point Topological Crossover is used in the GA operations in the analysis carried out for this research.

5.4.1 One-Point Crossover

In one-point crossover, both the parent chromosomes are selected and all of the genes that are part of these chromosomes are switched between them resulting in two off-springs. This can be shown in the figure below. Both parents maintain the precedence relationships, hence it will also be preserved in the first off-springs generated. Order of the tasks inherited also maintain the tasks specified in the input data flow graph.

From the data flow graph shown in figure 7, we perform our analysis to get the off springs from the parent population. This can be seen in Figure 8. For resource allocation fields, the substring denotes the number of resources used at each level of the DFG The appropriate functional units are all independent of each other and thus, one-point crossover needs to be applied to the substring. Figure 8(b) shows the allocation of resources such as the clock oscillators, multipliers and the adder/subtactor units. On performing the one-point crossover operations, we get the off-springs containing 2 clock oscillators, 2 multipliers and 2 adder/subtractor units as one child and 4 clock oscillators, 3 multipliers and 2 adder/subtractor as the other child.

5.5 Mutation

When a gene undergoes a change in order to get a better solution in the case of chromosome encoding, this process is termed as Mutation. Similar to how the crossover operations are done, Mutation operations too are performed separately in the node allocation field and in the resource allocation field. After the crossover operations are performed, as discussed before, the off-springs retain the topological order of tasks. However, after a few iterations, there may be solutions that are similar to the parent chromosomes. As a result, there may not be enough variety after a few generations and the algorithm could face premature convergence. Mutation helps retain the diversity in the algorithm for it to continue looking for the global best solution.

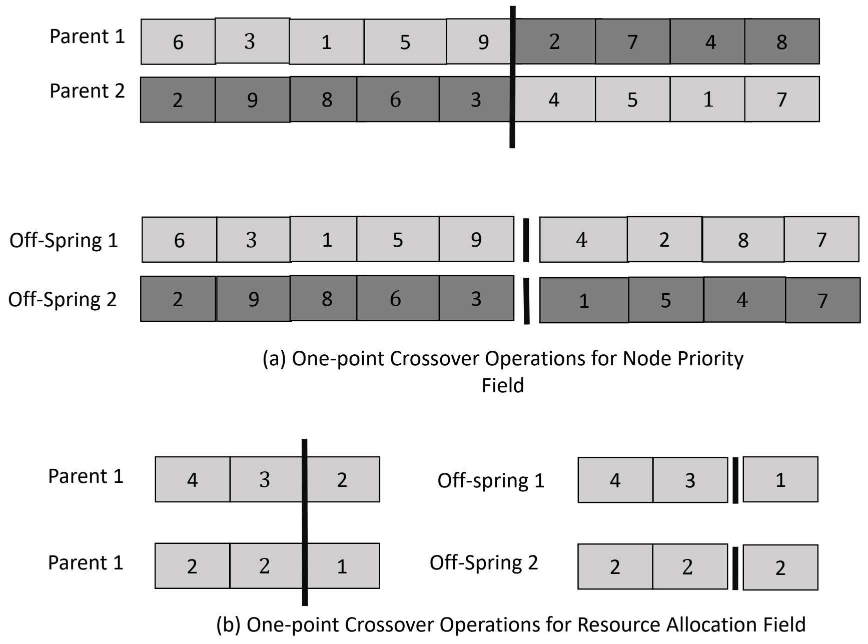


Figure 8: *Chromosome Encoding in Genetic Algorithms*

5.5.1 For Node Priority Substring

In the data flow graph shown in figure 7, for all the input tasks, there are two tasks happening before it but many tasks that are executed after it. The authors in [16] proposed the concept of a mutation scheme, which is used in the analysis of finding the most optimal variants. In this analysis, two random nodes (v_i, v_j) are chosen from a string of nodes, [k]. The value of the nodes is swapped among them thus transforming the nodal string.

5.5.2 For Resource Allocation Substring

The operation of mutation in Resource Allocation Substring is similar to the mutation operation that takes place in the node priority substring.two random nodes (v_i, v_j) are chosen randomly in the whole substring. The value of these nodes is either increased or decreased based on the constraints of the resources employed and also on the end application.

5.6 Disadvantages of Genetic Algorithms

Genetic algorithms, when conceptualized, started a monumental trend in the research of evolutionary algorithms. Although, pretty popular, in recent years, these algorithms have shortcomings in a number of applications, hence leading to the development of newer, much faster algorithms. Their disadvantages are as follows:

1. Difficulty of operation: Determining parameters such as the rate of mutation, parameters used for crossover operations and calculating the fitness function and normalizing it often need tweaks and adjustments. This is a trial and error method, thus making it more unreliable for important applications.

2. Time taken for termination: After the initialization process, there is a huge population of chromosomes that undergo the crossover and mutation operations. As a result, the solutions cannot be found faster. Sometimes, for bigger and more complex applications, one may have to wait for a few days to get the best solution.

3. Problems with convergence: For valuable applications, there is not only an issue with the time taken by the algorithms to finish operations but also with the rate of convergence. One of the biggest drawbacks of GA is that one may observe a loss of diversity after a few iterations, hence, leading the algorithm to a local minimum solution.

4. Vague solutions: The Fitness function of the functional units is literally the only way a user can interact with the system and exchange information. Because of which, the algorithm may or may not make any sense. If it continues in this manner, one could expect incomprehensible and inefficient solutions from a realistic engineering point of view.

Chapter 6

Analysis of Exploration of Design Space using Particle Swarm Optimization

6.1 Background

Particle Swarm Optimization (PSO) is a metaheuristic procedure that is based on the pattern of flight of a flock of birds as they fly in search of food/shelter. Each bird has its own position and maintains a particular velocity, both of which may be changed in order to mimic the behaviour of the rest of the flock as per requirements. The change in the position of the birds may be given as per the following equation:

$$x_i\left(t+1\right) = x_i + V_i\left(t+1\right) \tag{6.1}$$

The new position of the bird depends on its previous position and the updated velocity of the bird. The velocity can be updated using the following equation:

$$V_i\left(t+1\right) = \omega V_i\left(t\right) + c_1 r_1 \left(x_i^{lb} - x_i\left(t\right)\right) + c_2 r_2 \left(x_i^{gb} - x_i\left(t\right)\right) \tag{6.2}$$

Here ω is the inertia weight
c1 and c2 are the cognitive and the social learning factors respectively.
r1 and r2 are random numbers in the range [0,1].
x_i^{lb} is the position of the local best particle with respect to the minimization problem and
x_i^{gb} is the position of the global best particle.
The whole optimization algorithm aims at achieving x_i^{gb} , that is, all the particles try to converge at the best position of one particle, thus reaching a termination point.

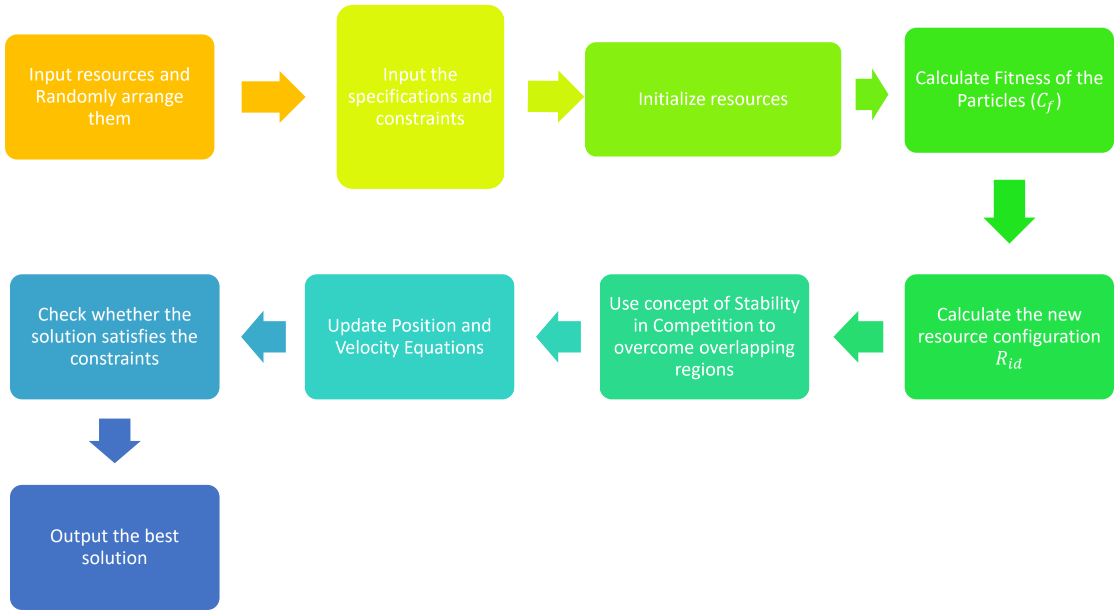

Input resources and Randomly arrange them
Input the specifications and constraints
Initialize resources
Calculate Fitness of the Particles (C_f)
Calculate the new resource configuration R_{id}
Use concept of Stability in Competition to overcome overlapping regions
Update Position and Velocity Equations
Check whether the solution satisfies the constraints
Output the best solution

Figure 9: *Flow of Particle Swarm Optimization for High Level Synthesis*

6.2 Particle Swarm Optimization for Multi-Objective Optimization

For the process of chip optimization in multi objective optimization problems, the PSO algorithm can be used to find the most optimum resource configuration, given the system constraints and specifications. The flow of the algorithm can be given in figure 9.

6.3 Framework of the model

The module library, user defined constraints and the behavioral description of the Data Flow Graph (DFG) are given as the inputs for the PSO algorithm. The following things are known to us at the start of the algorithm: The number of maximum resources available, energy consumed by each resource, time taken (in clock cycles) by the resources to finish one operation and the area occupied by the resources. In the Particle Swarm Optimization method, it is assumed that the Position of the particle is analogous to the resource configuration and the velocity of the particle to the process of exploration taken by the algorithm.

6.4 Initialization

The algorithm is first initialized by taking P1, the very first design variant, having minimum value. The second particle P2 which is the design variant having the maximum value and a third particle P3, which is the average of the other two. Rest of the design variants are all found using the formula stated below:

$$x_{id} = \frac{a+b}{2 \pm \alpha} \tag{6.3}$$

Here a least value of the resources
b- highest value of resources
α any arbitrary value between a and b

The initial velocity of all of the particles is zero, they are all at rest. Fitness of all of these particles (design variants) is then calculated in order to find the global optimal solution.

The particle with the minimum fitness is chosen as the global best resource configuration. After the algorithm begins, this process is repeated at every iteration and the values of position and velocity are updated, resulting in better resource configurations till the algorithm terminates.

6.5 Analysis of evaluation of design objectives using Particle Swarm Optimization

6.5.1 Power model analysis

The total power that is consumed by the functional resources is reliant on the static power and the dynamic power consumed by the resources. It can be given by the following equation:

$$P_T = P_D + P_s \tag{6.4}$$

Where P_T Total power consumed by the resources
P_D - Dynamic Power consumed by the resources and
P_s - Static Power consumed by the resources

The average dynamic power consumed by the resources is given as:

$$P_D = \frac{NE_{FU}}{T_{exe}} \tag{6.5}$$

Where N Total number of resources

E_{FU} -Total energy consumed by the resources
T_{exe} - Execution time

$$E_{FU} = E_{res} + E_{mux} + E_{demux}* \tag{6.6}$$

where E_{res} - Energy consumed by the resources
E_{mux} - Energy consumed by the multiplexers and
E_{demux} - Energy consumed by the demultiplexers

The total execution time is given by

$$T = L + (n-1)\,T_{cc} \tag{6.7}$$

Where L Latency of a scheduling solution N Number of Frames and T_{cc} - Cycle time of a scheduling algorithm

Thus P_D can be written as:

$$P_D = \frac{N(E_{res} + E_{mux} + E_{demux})}{L + (n-1)\,T_{cc}} \tag{6.8}$$

Static power P_S depends on the area of the resources and is independent of the dynamic activity of the module.

$$P_S = \sum_{i=1}^{n} N_{Ri} K_{Ri}.(p_c) \tag{6.9}$$

where N_{Ri} - Number of resources
K_{Ri} - Area occupied by the resources
p_c - Power consumed by the clock
Therefore, the total value of the power consumed by the resources can be given by the following equation:

$$P_T = \frac{N(E_{res} + E_{mux} + E_{demux})}{L + (n-1)\,T_{cc}} + \sum_{i=1}^{n} N_{Ri} K_{Ri}.(p_c) \tag{6.10}$$

6.5.2 Execution time model analysis

The time taken for the execution by a functional unit is given by the following equation:

$$T = L + (n-1)\,T_{cc} \tag{6.11}$$

Where L Latency of a scheduling solution
N Number of Frames and
T_{cc} - Cycle time of a scheduling algorithm

6.5.3 Fitness Function Calculation

The fitness of the particles in the algorithm can be found using the following formula:

$$C_{PSO} = \phi_1 \frac{T_{exe} - T_{cons}}{T_{exe}} + \phi_2 \frac{P_T - P_{const}}{P_T} \tag{6.12}$$

Where ϕ_1 and ϕ_2 are the weighted functions, whose value is determined by the user themselves. In our analysis, they are assumed equal and their value is taken to be 0.5.
These weights resonate with the trade-offs between time of execution and the power consumers. Taking equal weight signifies that both have the same priority during the analysis.
Lower the calculated fitness function of a particular resource, more power reduction can be achieved during employment of the functional units.

6.5.4 Energy model

Energy values obtained [16],[1] are used in finding the values of E_{FU} These are found using the equation used to find energy stored.

$$E_{FU} = \frac{1}{2} c V^2 \alpha \tag{6.13}$$

Where c Capacitance of the resource
V- Supply Voltage of the resource
α - Average switching activity at the inputs

The values of the stored energy are taken from [1] and is shown in the tables below. For different supply voltages 5V, the average energy acquired by the functional units is given in pJ for $\phi_1 = \phi_2 = 0.5$

6.6 Using Particle Swarm Optimization to solve the Multi-Objective Optimization Problem

The first set of design variants are chosen with the minimum number of resources, that is, one multiplier and one adder/subtractor: P1 = (1,1,1). The effectiveness of the particle swarm optimization partially

Functional Unit	Energy(pJ)
Adder/Subtractor	98.76
Multiplier	1990.53
Mux	24.05
Demux	68.032

Table 3: Average Energy acquired by the resources

depends on the initial population. The second set of design variants chosen are the ones with the maximum number of resources from the table, that is 4 multipliers and 2 adder/subtractors: P2 = (3,4,2). The third set of design variants is the average between the minimum and the maximum number of resources, that is
$\left(\frac{1+3}{2}, \frac{1+4}{2}, \frac{1+2}{2}\right) \approx (2,2,1)$

The values are rounded off to the nearest whole number. It depends on the user if they want to round it off to a value higher or to a value lower than the result obtained. In our case, we round it off to values lower than the decimal values. The initial velocity of all of the particles is set to zero in the first iteration. The social and the cognitive learning factors used in this algorithm can take any values between [1,4]. We then calculate the value of total power for resources (1,1,1)

$$P_T = \frac{N(E_{res} + E_{mux} + E_{demux})}{L + (n-1)\,T_{cc}} + \textstyle\sum_{i=1}^{n} N_{Ri} K_{Ri} . (p_c)$$

$$P_T = \frac{1000(98.76 + 1990.53 + 24.05 + 68.032)}{22 + (1000-1)\,20} + (1 \times 45 + 1 \times 10)\,(10)$$

$$= 109 \cdot 057 + 550 = 659 \cdot 05 mW$$

Execution time is given as :
$T_{Exe} = [L + (n-1)\,T_{cc}]T_{clk}$
$T_{Exe} = [22 + (1000-1)\,20](0.01) = 200\dot{0}2 msec$

Substituting these values into the formula, we can find the fitness function of the particles as follows:

$$C_{PSO} = \phi_1 \frac{T_{exe} - T_{cons}}{T_{exe}} + \phi_2 \frac{P_T - P_{const}}{P_T}$$

$$C_{PSO} = 0.5 \frac{200.02 - 62}{200.02} + 0.5 \frac{659.05 - 350}{6} 59.05$$

$$= 0\dot{3}45 + 0\dot{2}26 = 0\dot{5}71$$

For maximum number of resources
$P_T = \dfrac{N(E_{res} + E_{mux} + E_{demux})}{L + (n-1)\,T_{cc}} + \sum_{i=1}^{n} N_{Ri} K_{Ri}.(p_c)$

$$P_T = \frac{1000(2 \times 98.76 + 4 \times 1990.53 + 24.05 + 68.032)}{12 + (1000-1)\,8} + (4 \times 45 + 2 \times 10)\,(30)$$

$= 1030 \cdot 94 + 6000 = 7030 \cdot 94 mW$

Execution time is given as :
$T_{Exe} = [L + (n-1)\,T_{cc}]T_{clk}$
$T_{Exe} = [12 + (1000-1)\,8](0.005) = 40\dot{0}2 msec$

Substituting these values into the formula, we can find the fitness function of the particles as follows:

$$C_{PSO} = \phi_1 \frac{T_{exe} - T_{cons}}{T_{exe}} + \phi_2 \frac{P_T - P_{const}}{P_T}$$

$$C_{PSO}(Particle2) = 0.5\frac{40.02 - 62}{40.02} + 0.5\frac{7030.94 - 350}{7}030.94$$

$= -0\dot{2}7 + 0\dot{4}75 = 0\dot{2}75$

Similarly, for P3 (2,2,1)
$P_T = \dfrac{N(E_{res} + E_{mux} + E_{demux})}{L + (n-1)\,T_{cc}} + \sum_{i=1}^{n} N_{Ri} K_{Ri}.(p_c)$

$$P_T = \frac{1000(98.76 + 2 \times 1990.53 + 24.05 + 68.032)}{16 + (1000-1)\,12} + (2 \times 45 + 1 \times 10)\,(20)$$

$= 574 \cdot 34 + 2000 = 2347 \cdot 54 mW$

Execution time is given as :
$T_{Exe} = [L + (n-1)\,T_{cc}]T_{clk}$
$T_{Exe} = [16 + (1000-1)\,12](0.0066) = 79\dot{2}26 msec$

Substituting these values into the formula, we can find the fitness function of the particles as follows:

$$C_{PSO} = \phi_1 \frac{T_{exe} - T_{cons}}{T_{exe}} + \phi_2 \frac{P_T - P_{const}}{P_T}$$

$$C_{PSO}(Particle2) = 0.5\frac{79.226-62}{79.226} + 0.5\frac{2347.54-350}{2}347.54$$

$$= 0 \cdot 108 + 0 \cdot 425 = 0 \cdot 533$$

Hence, the particle functions have fitness functions as
Particle 1 (1,1,1): 0.571
Particle 2 (3,4,2): 0.205
Particle 3(2,2,1): 0.533
The one with the least fitness function becomes the global best particle, for that particular iteration. Here particle 2, is the global best solution for this iteration x_i^{gb} . In the next steps, when we find the fitness of those particles, if it is lower than the current best fitness, then it would become the local best solution x_i^{lb}

6.6.1 Determining the best particle

The next step in the algorithm is to update the position and the velocity of the particles and continue the search for the global best solution. The configuration of the new particle can be found using the following equation:

$$x_{ia}^{+} = f\left(V_{id}^{+}, x_{id}\right)$$

V_{id}^{+} is calculated using the following equation:

$$V_i\left(t+1\right) = \omega V_i\left(t\right) + c_1 r_1\left(x_i^{lb} - x_i\left(t\right)\right) + c_2 r_2\left(x_i^{gb} - x_i\left(t\right)\right)$$

Where ω- inertia weight,
c1 and c2 - cognitive and the social learning factors between [1,4] respectively
r1 and r2 - Random numbers in the range [0,1]
x_i^{lb} is the position of the local best particle with respect to the minimization problem
x_i^{gb} is the position of the global best particle

For the number of clock resources, we use the velocity equation to get,
$$V_i\left(t+1\right) = \omega V_i\left(t\right) + c_1 r_1\left(x_i^{lb} - x_i\left(t\right)\right) + c_2 r_2\left(x_i^{gb} - x_i\left(t\right)\right)$$

$$V_i\left(t+1\right) = 1 \times 0 + 2 \times 0\dot{5} \times (1-1) + 2 \times 0\dot{5} \times (2-1) = 1$$

And the position is given by

$$x_i\left(t+1\right) = x_i + V_i\left(t+1\right) \; x_i\left(t+1\right) = 1 + 1 = 2$$

Hence the updated resource configuration is (3,2,2) of the particle.

Chapter 7

Materializing the High Level Synthesis Design Flow

7.1 Scheduling and Binding

Acrylic graphs that are represented using edges and vertices, that help in the depiction of the resources are known as Scheduling and Binding graphs. Vertices denote the resources and the edges denote the path taken by the operations in these resources. The attributes of an operation are stated by Sequencing Graphs. They tell us what operations are happening in the resource at one particular time and help in visualizing the flow of the data elements. There are three kinds of scheduling problems that designers deal with. They are shown in the figure 10.
Resource Constrained Scheduling: The constraint is the number of resources. Fastest schedule is determined with respect to the constraints imposed on the functional units.
Time Constrained Scheduling: The number of time steps determine the cheapest possible scheduling and are taken as the constraints.
Feasible Constrained Scheduling: An output, if it exists, needs to be scheduled. The maximum number of resources and time are the constraints that need to be satisfied in parallel.
Thus in general, the scheduling problem is known to be three fold; based on dependency, time and resources. It is an NP complete problem. The manner in which an operation is specified to the functional units is known as Scheduling. This is done to improve the efficiency of the system and to also achieve goals that are crucial for that particular application. Scheduling algorithms are of different kinds, namely the As soon As possible Scheduling(ASAP), As late as possible (ALAP), Force Directed Scheduling, List Scheduling and so on. In the analysis required for this thesis, As Soon As Possible (ASAP) scheduling was used. This was done to make sure that as soon as the resource finishes one operation, it can be ready for the next. This helps in execution of the operations at a faster rate, especially in heavily pipelined circuits. Binding helps in comprehending the function that is used as a benchmark application in the manifestation of the optimized design flow. In computer architecture and synthesis, three dimensions

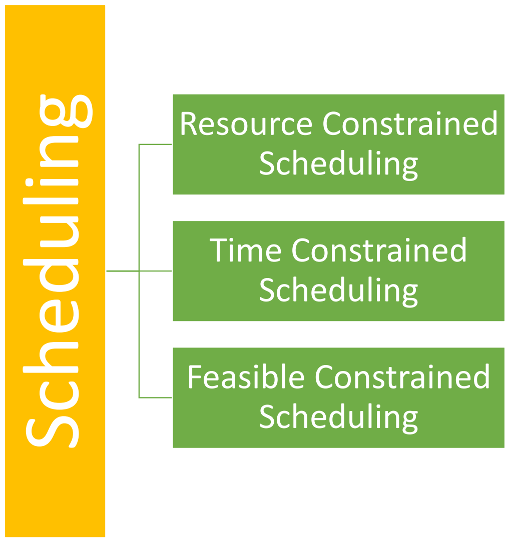

Scheduling
Resource Constrained Scheduling
Time Constrained Scheduling
Feasible Constrained Scheduling

Figure 10: *Three Divisions of Scheduling Problems*

can be used to specify a circuit. These three dimensions include the sequencing graph, number of functional units and the constraints. In Figure 11, a register R has been added in time step T3. The addition of another register was done because the adder/subtractor unit wasnt free in that particular time step. Thus is holds the value till the resource becomes free in the next time step and the operation can be executed. The latency of the function given in Figure 11 was calculated to be 16 clock cycles.

7.2 Resolving the Multiplexing Scheme

The next task for SoC designers is to incorporate the multiplexers and the demultiplexers into the data path of the circuit. This is done using Binding. In the process of high level synthesis design flow, determining the multiplexing scheme is one of the most relevant steps, it helps in symbolizing the functional units with their respective inputs, outputs and data flow along with the interconnections. Realizing the multiplexing scheme and later making a schematic diagram helps in figuring out the complexity of the circuit and it may also help in preventing failures at a later stage.

7.3 Block Diagram of the circuit

There are two distributions of the block circuit developed of the system after the multiplexing scheme has been made. These two sections are the data path and the control path. The data path is responsible

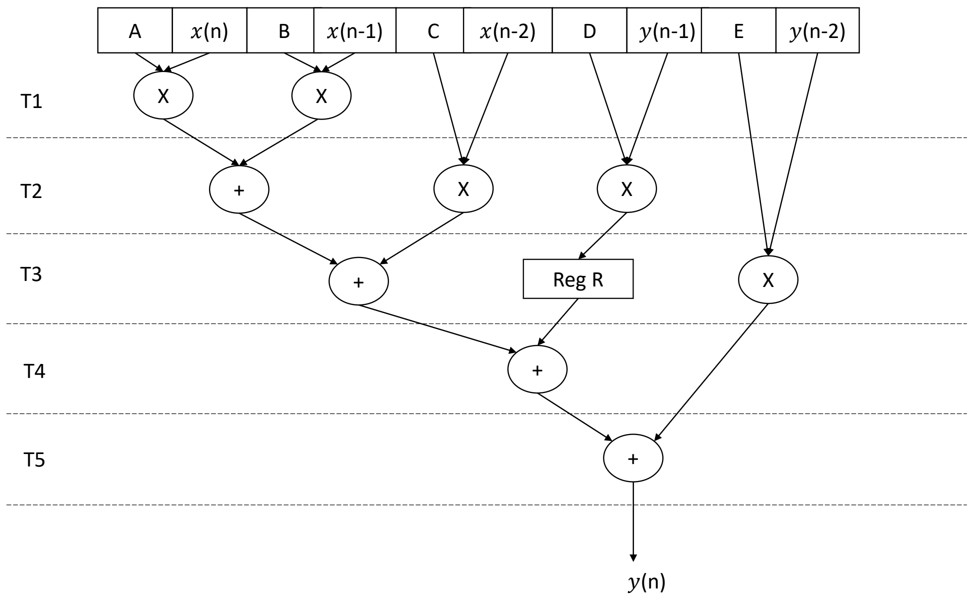


Figure 11: *Data Flow Graph with Register*

for administering the sequence of operations that need to be performed in the circuit. It is accountable for the flow of data in the circuit through the wires and the buses. Apart from the functional units performing the operations, the data path consists of memory elements such as latches. Latches help in storing the data until it is ready to be used again at a later stage. It also consists of registers, as well as multiplexers and demultiplexers. For the most optimum variant that we got in the analysis for the sake of our thesis, the block diagram consists of two multipliers and one adder/subtractors. The circuit also consists of a control unit that helps implement all the required timing and synchronization. The control unit is generally a finite state machine in which the state varies as per the activation and the deactivation of the other elements during different periods of time. Figure 12 shows the block diagram of the circuit.

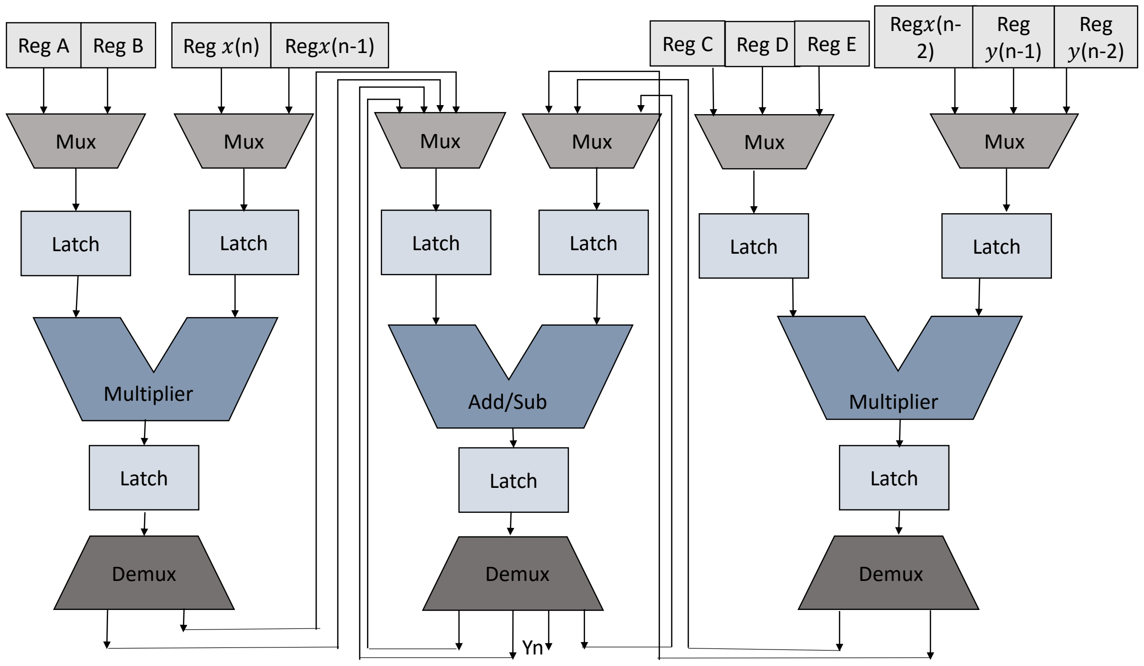


Figure 12: *Block Diagram of the System Data Path Description*

Chapter 8

Results and Analysis

8.1 Overcoming the Overlapping Domains

An analysis has been made of the three different methods for finding the most optimal configuration for the resources in the circuit. The border variants need to be found for all the three methods before selecting the most optimum architecture. In all of the three methods, there is a chance that there may be overlapping regions. These are dangerous in our search. This is because the algorithm gets stuck in the overlapping region and keeps searching within it. In case of evolutionary algorithms, this may lead to premature convergence of the algorithm. H. Hoteliers concept of Stability in Competition [8] has been used in order to make the search space as linear as possible. The figure shown below depicts the case of overlapping domains. In part (a) of the figure, a representative model of partially arranged design variable is shown which is overlapping. Arranged variables have overlapping domains if the following condition holds:

$N_{R1}k_{R1} > N_{R2}K_{R2}$

For the algorithm to avoid overlapping domains, it needs to fit the following criteria: $N_{R1}k_{R1} \approx N_{R2}K_{R2}$

8.2 Comparison of Particle Swarm Optimization with Genetic Algorithms and Priority Factor method

In order to find the most optimum resource, two of the evolutionary algorithmic procedures Particle Swarm Optimization and Genetic Algorithms were compared with the Priority Factor method. The results for different DSP filters obtained from benchmarks[3] are shown in the table below. The values of phi_1 and $phi_2$are defined by the user and for the sake of our analysis are taken as 0.5. As we can see, the results obtained by PSO algorithm are much better. These utilize lesser resources to solve the multi-objective optimization problem than the genetic algorithm or the priority factor method.

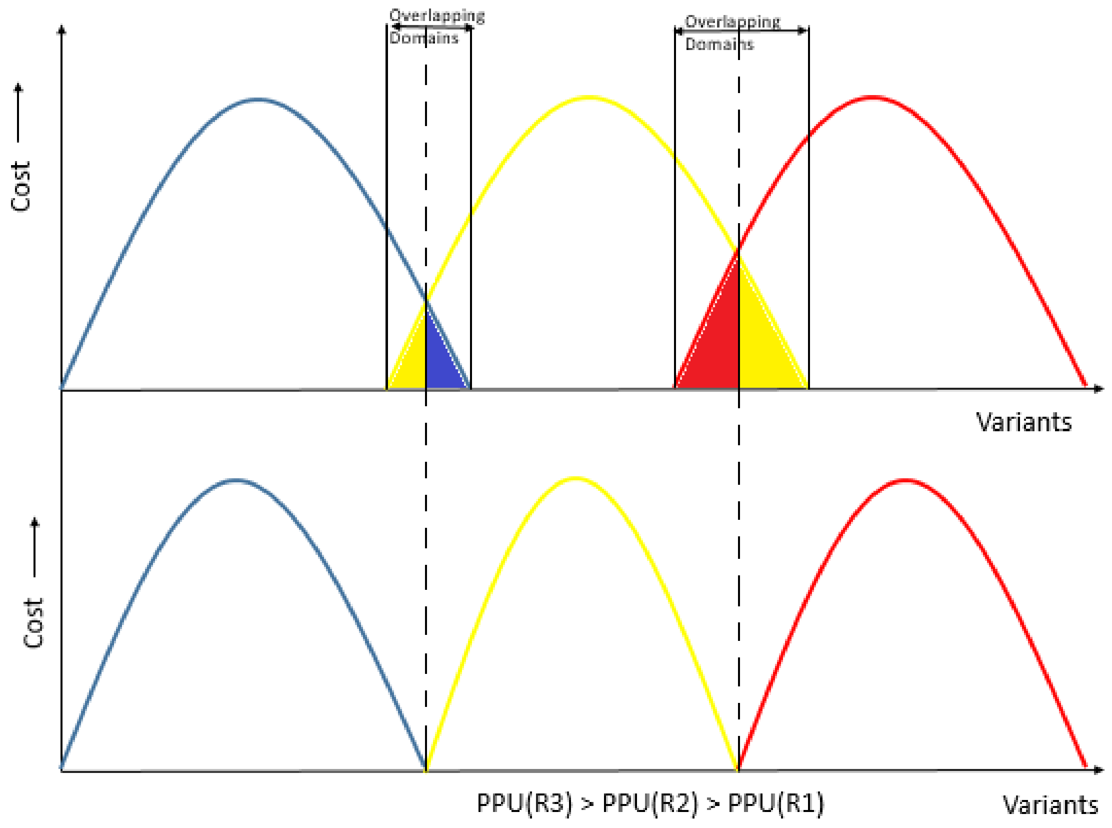


Figure 13: *Illustrations showing the overlapping domains of partially arranged design variables in PF hierarchy and the linearly ordered decision variables with restricted domains to avoid inferior domains*

DSP Filter	Resource Configuration		
	PSO	**GA**	**PF**
Elliptic Wave Filter	2(mult),1(add/sub)	1(mult),1(add/sub)	2(mult),1(add/sub)
Infinite Impulse Response Filter	2(mult),1(add/sub)	3(mult),1(add/sub)	4(mult),1(add/sub)
Finite Impulse Response Filter	3(mult),1(add/sub)	2(mult),2(add/sub)	4(mult),2(add/sub)
Bandpass Filter	2(mult),1(add/sub)	2(mult),2(add/sub)	4(mult),2(add/sub)
AutoRegressive Filter	3(mult),1(add/sub)	4(mult),1(add/sub)	4(mult),1(add/sub)

Table 4: Optimal Resource Configuration for the Benchmarks [3]

DSP Filter	**Latency**	**Power(mW)**	Execution Time	
			Constraint	**Solution**
Elliptic Wave Filter	16	300	55	52.83
Infinite Impulse Response Filter	18	520	62	60.42
Finite Impulse Response Filter	30	460	68	59.23
Bandpass Filter	10	600	25	11.7
AutoRegressive Filter	24	580	70	54.9

Table 5: Comparison of Power and Time of execution for the benchmarks[3]

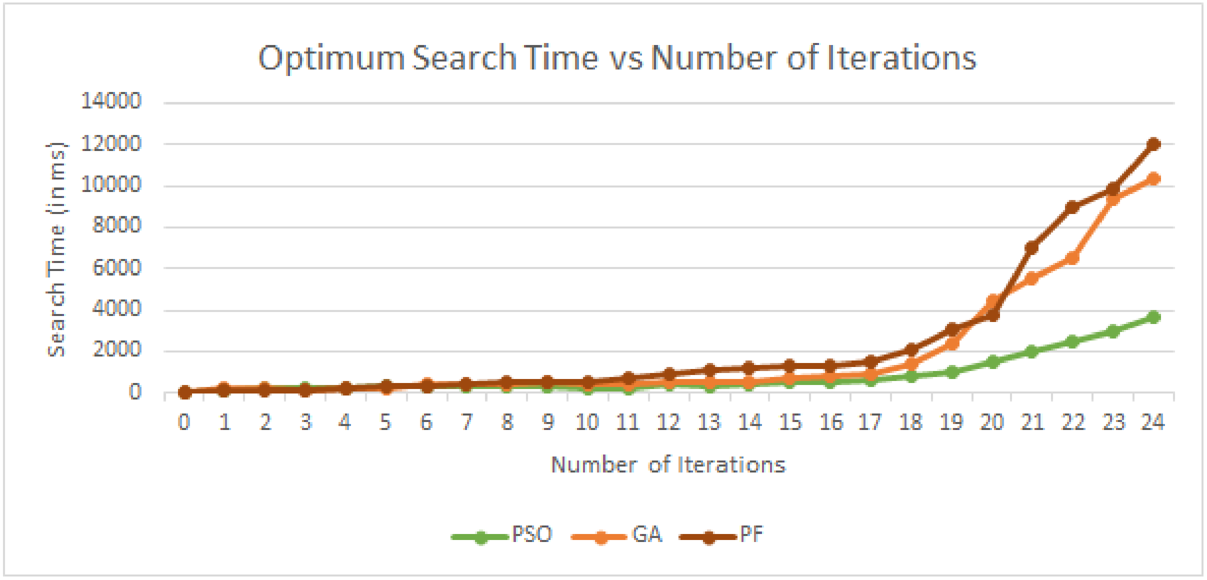


Figure 14: *Depiction of the search time with the Number of Iterations*

The speed of execution and power were also compared for the benchmarks, and the results are tabulated below:

8.2.1 Exploration Run Time and Time of Convergence of the algorithms

All of the three methods used in the Design Space Exploration process were also analyzed for the total time taken in searching for the best design solution and the results were plotted for the case study used in this thesis. From figure 14, it is observed that for smaller sets of data the Priority Factor method performs better than the evolutionary algorithmic methods of analysis. The behavior of the heuristics: The Genetic Algorithm and the Particle Swarm Optimization method is almost the same for smaller sets of data as well.

However, as the number of iterations keeps increasing, the search space keeps getting more and more non-linear. There may be cases of the algorithm getting stuck in the local minima as the chances of

Iteration 10	Methods of Exploration		
	PF	**GA**	**PSO**
Variant Number	(2,1,2)	(3,4,2)	(3,1,1)
Power (mW)	130	128.63	165
Time of Execution (msec)	132.01	100.02	80.06

Table 6: Comparison between the three methods at one particular iteration

one-to-many correspondences in the design space keep increasing too. The exhaustive search method of the Priority Factor analysis would take a lot of time. Genetic Algorithms, although better than the PF method, is still worse than the PSO method for bigger systems. The Particle Swarm Optimization method, as seen from the graph in figure 15, takes the least amount of search time for larger systems that require a higher number of iterations. The characteristics of the line depicting the PSO algorithm depends on the position and the velocity of the particles. The velocity of the particles are further dependent on c1 and c2: the cognitive and the social factors of the particles. On changing those parameters, we can see that the other two lines, the ones representing the Priority Factor method and the Genetic algorithm portray exponential time complexity whereas the PSO method is far better.

On comparing the three methods as seen in Table 6,we see that at iteration number 10, the priority factor method is at Variant number 10, the one having Power as 130mW and Time of execution as 132msec. For the Genetic Algorithm method, it would depend on the number of chromosomes encoded in the iterations before. Genetic Algorithms and Particle Swarm Optimization are meta-heuristics. While the operations performed by them cannot be predicted, our simulations show that for the 10th iteration, the resource allocation field node has (3,4,2) as the off-spring, the Power consumed by the off-springs(the resources) was 128.63mW and the time of execution was 100.02msec. For Particle Swarm Optimization, it was seen that the algorithm was almost near termination. Variant (3,1,1) found to be the global best for this iteration and the power and the time were 165 mW and 80.06 msec respectively.

The time that each algorithm takes in arriving at the best solution is known as the Time of Convergence. As per the analysis, the PSO takes the least amount of time to converge to a particular solution, Genetic Algorithms, another meta-heuristic that we use in our analysis takes more time and the Priority Factor method takes the most amount of time to reach the most optimum solution, especially if the number of design sets (configurations) are in hundreds and thousands. This is shown in Figure 15.

In figure 16, the variation of reduction in runtime has been obtained for different benchmarks. For complex benchmarks such as the Finite Impulse Response and the Bandpass Filter, the exploration runtime is higher than the other filters for Particle Swarm Optimization than the other methods. Thus for this method, the problem of the growing complexity of the design space is solved in a justifiable period of time.

8.2.2 Quality of Search

To perform the qualitative analysis of all of the filters in runtime, the Quality of Search (QoS) method is considered too. QoS is dependent on the total time taken by the circuit to be executed using the three

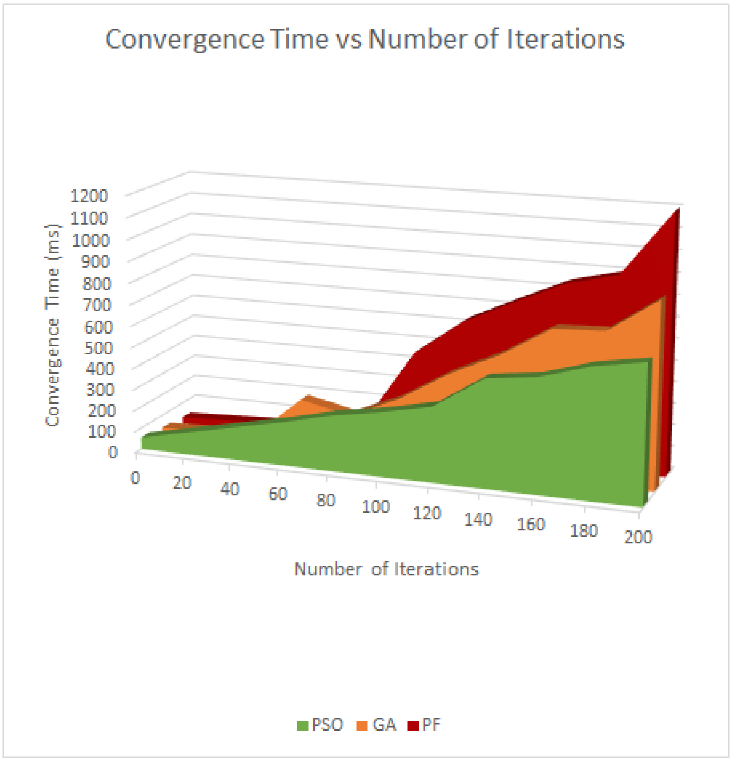


Figure 15: *Time of Convergence Vs Number of Iterations*

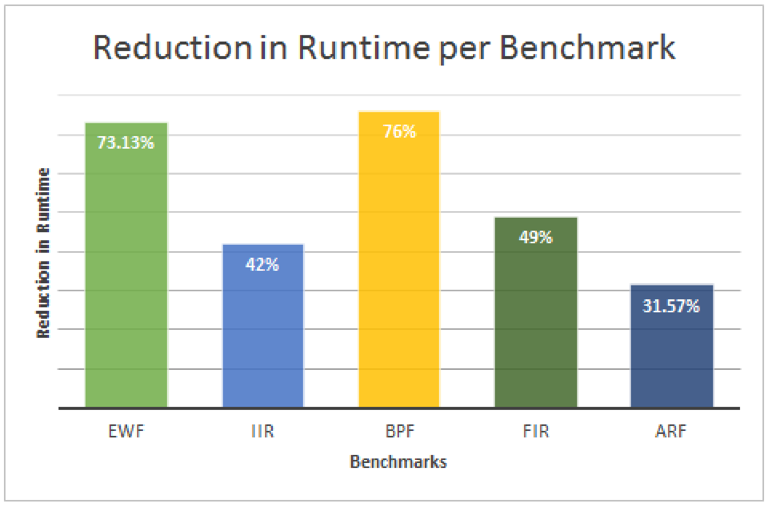
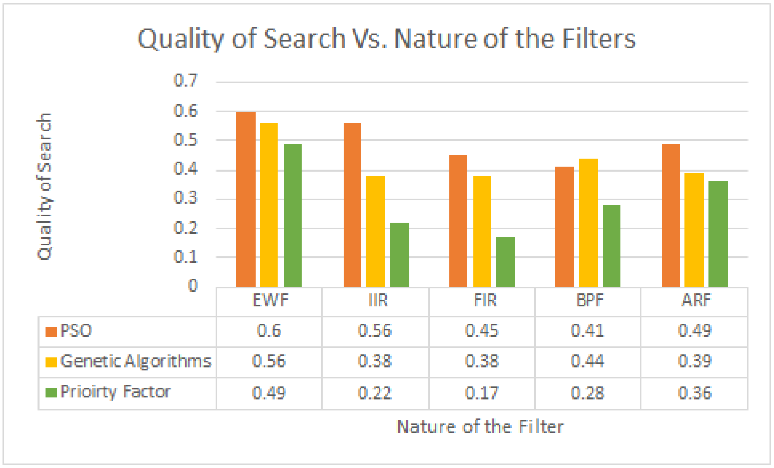


Figure 16: *Percentage Decrease in Runtime for DSP Filter benchmarks*



	EWF	IIR	FIR	BPF	ARF
🟧 PSO	0.6	0.56	0.45	0.41	0.49
🟨 Genetic Algorithms	0.56	0.38	0.38	0.44	0.39
🟩 Prioirty Factor	0.49	0.22	0.17	0.28	0.36

Figure 17: *Graph depicting the quality of search Vs. the nature of the filters*

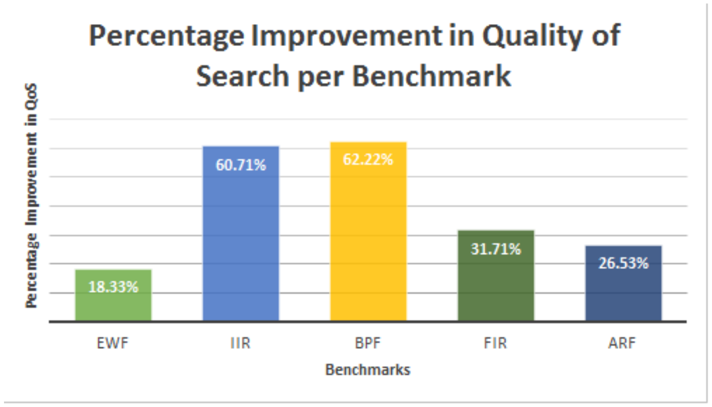


Figure 18: *Graph depicting the percentage improvement in quality of search for the filters*

algorithms and also on the power consumed by all of the functional units in the system. In [16], the authors compare the quality of their heuristic in terms of the cost. For Particle Swarm Optimization, the QoS is much better than the other two methods of searching the design space as can be seen from Figure 17: As seen in the figure, both the heuristics have almost the same performance for all of the filters. Since the PF method takes longer to reach the most optimal set of solutions, especially for an increased number of rounds, the Quality of Search of this algorithm is lower than the other heuristic methods.The percentage improvement in Quality of Search is shown in Figure 18

Chapter 9

Conclusion and Future Work

A comparison was made between the three methods for a rapid and a detailed design space exploration. The main objective of solving a multi-knapsack problem was to design the system in a way such that minimum resources are used. The power consumed by all of the functional units and the execution time needs to be within the constraints. In the first approach used, the Priority Factor Method, all of the resources were arranged in terms of their decreasing priority factors of Power and Time of execution. This lead to the ordering of the design sets. After this was accomplished, an exhaustive search determined the most efficient design set could be found that consumed power less than the maximum power used as a constraint and also had an execution time less than the constraint. Since this process uses exhaustive search, it isnt very profitable for larger design spaces. Here, we had only 24 sets of the possible design variants that needed to be searched. However, for larger applications, where we may have thousands of configurations, the priority method would take eons of time to accomplish this task. Thus other heuristic methods were considered for solving this multi-objective optimization problem. The most widely researched algorithm was the Genetic Algorithms [9]. The most optimum set of design constraints obtained from the Priority method were used to make a data flow graph. This DFG was used for chromosome encoding. The chromosomes consist of two types of substrings: Node Field Substring and the Resource Allocation Substring. The Node field substring portrays an ambiguous representation of the final schedule. It is used to encode a list of tasks that need to be scheduled and the resource allocation substring denoted the number of functional units required at each time step of the DFG. Two main operations of Genetic Algorithms, namely, the crossover and mutation operations were discussed. They were used to find the off-springs that had a fitness function better than the parent population. Nevertheless, Genetic Algorithms, come with their own drawbacks. Like the priority factor method, these too, take up a lot of time to reach a conclusion. A major drawback with using this method of solving the optimization problem is that the computer only recognizes the fitness function and does its analysis according to that. The output, however, may not be a feasible solution in terms of a realistic engineering problem. And so, this method isnt very reliable. Particle Swarm Optimization has been proved, time and again [16, 17] to be a better metaheuristic in solving the optimization problems. In case of Particle Swarm Optimization, three particles were initialized with zero velocities. One of them had the minimum number of resources, one had the maximum and the third particle had resources that were the average of the other two. The fitness of all of these particles was found, the particle with the least fitness became the global best particle for the next iteration. The position and velocities were updated accordingly in the next round. This algorithm goes on until it reaches the desired resource set or if the number of iterations to be performed are done. One major complication faced by all of these algorithms is that of overlapping regions. The concept of Stability in Competition is used to overcome this problem and help the search get out of the overlapping domains. Of all of the methods analyzed, the Particle Swarm Optimization was found to be the best in searching for the best possible sets of resources helping in bridging the gap between the ESL to the RTL in the shortest period of time.

9.1 Future Work

Today there are a bunch of applications that require the best performance in the shortest period of time, that also consumes minimal power. These two parameters, although contradictory, are both extremely important for all of these applications. The research in this domain is ongoing. There are various heuristics being developed that perform the exploration process. These algorithms may help in lowering the number of the sets of configuration that need to be analyzed. As a result, this would lead to a shorter search time and help in finding the best solution faster. This will not only be more time efficient but also cost efficient for the manufacturer. In our analysis, only power consumption and time of execution was considered. There is still scope for improvement in other aspects such as temperature and reliability, which havent been examined much earlier.

References

[1] Jui-Ming Chang and M. Pedram. Energy minimization using multiple supply voltages. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(4):436–443, Dec 1997.

[2] R. Courtland. The origins of intels new transistor, and its future. Technical report, IEEE Spectrum, 2011.

[3] R. Courtland. Memory in the third dimension. *IEEE Spectrum*, 51(1):60–61, January 2014.

[4] S. Damaraju, V. George, S. Jahagirdar, T. Khondker, R. Milstrey, S. Sarkar, S. Siers, I. Stolero, and A. Subbiah. A 22nm ia multi-cpu and gpu system-on-chip. In *2012 IEEE International Solid-State Circuits Conference*, pages 56–57, Feb 2012.

[5] C. Haubelt and J. Teich. Accelerating design space exploration using pareto-front arithmetics [soc design]. In *Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference, IEEE 2003.*, pages 525–531, Jan 2003.

[6] Harold Hotelling. Stability in Competition. *The Economic Journal*, 39(153):41–57, 1929.

[7] C. Hu. 3d finfet and other sub-22nm transistors. In *2012 19th IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits*, pages 1–5, July 2012.

[8] Lev Kirischian, Vadim Geurkov, Valeri Kirischian, and Irina Terterian. Multi-parametric optimisation of the modular computer architecture. *International Journal of Technology, Policy and Management*, 6(3):327–346, 2006.

[9] V. Krishnan and S. Katkoori. A genetic algorithm for the design space exploration of datapaths during high-level synthesis. *IEEE Transactions on Evolutionary Computation*, 10(3):213–229, June 2006.

[10] A.T. Sharma P. Siddavaatam, R. Sedaghat. An optimal framework for node selection using dynamic clustering for wireless sensor networks. *In Proceedings of International Conference for Internet Technology and Secured Transactions,Cambridge*, December 2017.

[11] Luca Piccolboni, Paolo Mantovani, Giuseppe Di Guglielmo, and Luca P. Carloni. Cosmos: Coordination of high-level synthesis and memory optimization for hardware accelerators. 16:1–22, 09 2017.

[12] P. Siddavaatam R. Sedaghat, A.T. Sharma. A novel multi-objective optimization approach for design flow in high level synthesis. *In Proceedings of International Conference on Electrical, Electronics, Computers, Communication, Mechanical and Computing (EECCMC), India*, January 2018.

[13] P. Ramm, A. Klumpp, J. Weber, N. Lietaer, M. Taklo, W. De Raedt, T. Fritzsch, and P. Couderc. 3d integration technology: Status and application development. In *2010 Proceedings of ESSCIRC*, pages 9–16, Sept 2010.

[14] B. C. Schafer, Takashi Takenaka, and Kazutoshi Wakabayashi. Adaptive simulated annealer for high level synthesis design space exploration. In *2009 International Symposium on VLSI Design, Automation and Test*, pages 106–109, April 2009.

[15] A. Sengupta and R. Sedaghat. Integrated scheduling, allocation and binding in high level synthesis using multi structure genetic algorithm based design space exploration. In *2011 12th International Symposium on Quality Electronic Design*, pages 1–9, March 2011.

[16] Anirban Sengupta and Reza Sedaghat. Swarm intelligence driven design space exploration of optimal k-cycle transient fault secured datapath during high level synthesis based on user powerdelay budget. *Microelectronics Reliability*, 55(6):990 – 1004, 2015.

[17] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317 – 325, 2003.