PERFORMANCE ENHANCEMENT OF SYSTOLIC ARRAY DESIGN USING COMPUTATIONAL INTELLIGENCE

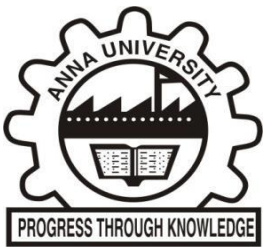
A THESIS

Submitted by

SELVAKUMAR R

in partial fulfillment of the requirements for the degreeof

DOCTOR OF PHILOSOPHY



FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

ANNA UNIVERSITY

CHENNAI 600 025

JULY 2015

ANNA UNIVERSITY

CHENNAI 600 025

CERTIFICATE

The research work embodied in the present Thesis entitled "**PERFORMANCE ENHANCEMENT OF SYSTOLIC ARRAY DESIGN USING COMPUTATIONAL INTELLIGENCE**" has been carried out in the Electronics and Communication Engineering, Karpagam College of Engineering,Coimbatore. The work reported herein is original and does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion or to any other scholar.

I understand the University's policy on plagiarism and declare that the thesis and publications are my own work, except where specifically acknowledged and has not been copied from other sources or been previously submitted for award or assessment.

SELVAKUMAR R
RESEARCH SCHOLAR

Dr.DHARMISHTAN K.VARUGHESE
SUPERVISOR
Professor
Electronics & Communication Engineering
Karpagam College of Engineering
Coimbatore

ABSTRACT

The main aim of using the existing system effectively and developing new architectures is to run more complex application faster than ever before. The ongoing demand for increased computing power makes researchers to develop high parallel scalable multiprocessing systems. In general, parallelism is an appealing and intuitive concept. Fundamentally, there are two ways of improving the performance of computer with respect to computational speed. First method is to use the faster devices i.e. VLSI chips. The faster and faster VLSI components have ability to improve the computational speed and at the same time breakthroughs in increasing circuit densities and switching speed of VLSI devices are very complex and also costly in future. The second is to use the parallel processing architectures along with the multiple processors in order to perform the computation task. At the same, while working together with multiple processors, appropriate architecture is essential in order to achieve maximum performance and also in a most cost effective manner. In general, systolic arrays are most suitable for computationally intensive applications along with the inherent massive parallelism and this is because they have capability on modular, concurrent, regular, synchronous, rhythmic processes that require repetitive and intensive computation. In various areas of science and engineering systolic architecturesare considered as one of the suitable way for implementing parallel algorithms. The systolic model suits very well for special purpose and high performance computer devices.

Until recently, the computation intensive tasks were generally handled by high performance supercomputers including the multiprocessor systems, pipelined computers, and array processor. At the same time, the general purpose of these systems leads to the severe system overheads and

complicated system organization. These types of machines are generally not efficient from the view of electronics for real time signal processing and this is because, for such processing, very high throughput rate is most important. The major solution for real time signal processing is to use the array processor and also essential to minimize processing concurrency by using both or either pipeline processing or parallel processing. As long as, the communication in VLSI remains restrictive, the locally-interconnected array is given more importance.

Nowadays, there are increasing pressures to the researchers to develop computing architecture which are suitable to surrounding environment in order to achieve the major goal in an optimum manner. This pressure makes researchers to choose the approach called trial and error method in order to identify the compactable solution which is not only time consuming process but also the case of suboptimal solution which is accepted at end. So, there is a strong need to identify the automated approaches which have ability to deliver several possible solutions efficiently and quickly with best possibilities. This goal can be achieved by using the computational intelligence for the exploration of solution. As a result, in order to develop the efficient automated solution environment in order to design efficient systolic architecture which uses computations like, genetic algorithm, chaotic optimization, evolutionary programming, and swarm intelligence have been considered as a solution.

The characteristics and quality of obtained systolic architecture are fully based on the fundamental design vectors. Iterative algorithm is considered for identifying the optimal values for three fundamental design vectors such as projection, scheduling and processor vector. This ability makes efficient architecture to be feasible along with the maximum hardware utilizing efficiency. This approach also has ability to minimize the total

number of delay which is involved in the systolic architecture design. Apart from these, mathematical formulation has been considered as fitness function in order to explore the heuristic method, where the estimation of vectors is identified as problem of constraint optimization and its main objectives is to minimize the total delay which are associated with each edge of the processing element and also to maximize the hardware utilizing efficiency. These both objectives are formulated into a single objective problem called the problem of minimization.

NIC (Nature inspired computation) is considered as the major area in computational intelligence, where processing mechanism and natural entity behavior and so on are chosen for mathematical modeling consideration in order to maintain the natural properties in modeling which in turn helps to achieve optimum solution. The main advantages of using nature inspired computation are fault tolerant, self-adaptive, and parallel processing where human made solutions can have ability to suffer directly

Evolutionary programming is very efficient heuristic under the evolutionary computation and also it works for real value domain. In general, evolutionary programming has developed in order to identify several optimal design vectors for systolic architecture. Another successful swarm intelligence based concept, particle swarm optimization has applied in discrete form in order to identify the comparative performance analysis and also identify the optimal solution

An automated method is proposed in this study in order to estimate and explore the systolic array design vectors by using the help of hybridization of genetic algorithm and also chaos based chaotic optimization (HGACH). Here, the hybridization is applied on proposed method in order achieve the solution in a faster manner. It is possible to develop the number of different systolic

architecture which has the characteristics of fully pipelined and partial pipeline method with the proposed method. In order to test the proposed solution more detail, numerical benchmark problems have also tested and the results are appreciable. HGACH method is used for designing different architectures of systolic array corresponding to various computational intensive problems like Matrix multiplication, Convolution, Vector quantizer and Digital correlator.

ACKNOWLEDGEMENT

I wish to record my deep sense of gratitude and profound thanks to my research supervisor **Dr. Dharmistab K.Varughese**, Professor, Electronics & Communication Engineering, Karpagam College of Engineering, Coimbatore, for his keen interest, inspiring guidance, constant encouragement with my work during all stages, to bring this thesis into fruition.

I also thank the faculty and non-teaching staff members of the Electronics & Communication Engineering, Karpagam College of Engineering, Coimbatore for their valuable support throughout the course of my research work.

SELVAKUMAR R

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence CI
APU	-	Arithmetic Processing Unit
BPSO	-	Binary element Particle Swarm Optimization
CE	-	Chaotic Evolution
CI	-	Computational Intelligence.
CMOS	-	Complementary Metal-oxide semiconductor
CPSO	-	Chaotic Particle Swarm Optimization Algorithm
DFG	-	Data Flow Graph
DFT	-	Discrete Fourier Transform
DG	-	Dependence Graph
DP-PSO	-	Decreasing Population PSO
DSP	-	Digital Signal Processing
EA	-	Evolutionary Algorithm
EC	-	Evolutionary Computation
EP	-	Evolutionary Programming
EP-PSO	-	Extending Population PSO
EPROM	-	Erasable Programmable Read Only Memory
EPSF	-	Embedded Parallel Systolic Filters
ES	-	Evolutionary Strategy
ESEA	-	Enhanced Self-adaptive Evolutionary Algorithm
FPGA	-	Field Programmable Gate Array
GA	-	Genetic Algorithm

GNBM	-	Gaussian Normal Basis Montgomery
GPSO	-	Geometric Molecule Swarm Advancement
GTD	-	Generalized Triangular Decomposition
HDL	-	Hardware Description Language
HGACH	-	Hybridizing of Genetic Algorithm and Local Chaotic Optimization
HLS	-	High Level Synthesis
HNN	-	Hopfield Neural Network
HUE	-	Hardware Utilization Efficiency
I/O	-	Input and Output
IR	-	Intermediate Representation
LCS	-	Longest Common Subsequence
MIMD	-	Multiple Instruction Multiple Data
NML	-	Nano magnet Logic
OFDM	-	Orthogonal Frequency Division Multiplexing
PEs	-	Processing Elements
PSO	-	Particle Swarm Optimization
QSA	-	Quantum-dot Cellular Automata
RAM	-	Random Access Memory
RDG	-	Reduced Dependence Graph
RIA	-	Regular Iterative Algorithm
RTL	-	Register Transfer Language
SA	-	Systolic Array
SAP	-	Systolic Array Processor

SFG	-	Signal Flow Graph
SIMD	-	Single Instruction Multiple Data
SISD	-	Single-Data Stream Device
SPE	-	Serial or Parallel
2D	-	Two-dimensional
V-BLAST	-	Vertical Bell Labs Layered Space-Tim
VLSI	-	Very Large Scale Integration

CHAPTER 1

INTRODUCTION

1.1 RESEARCH BACKGROUND

Researchers have re-examined the path of computation alliance scatters refining effects, and posterity devices will sustain this course with greater bandwidth transmissions and greater computing achievement. One solution to attain superior computational achievement is to apply certain lateral coprocessors to execute operations such as agitation and colour base of high-definition presence. Eventual computationally accelerated operations fitted for desktop computation devices count actual-time passage, language, and image processing. These functions need colossal parallelism. Several computing goals are by their very common persistent: for other goals the grade of parallelism differs. Hence, a colossal lateral computational structure must have enough operation adaptability and computational capability Johnson (1993).

Computational intelligence (CI) is the course of the structure of creative abettor. An abettor is one that moves in a circumstanceit performs something. An intelligent abettor is a structure that moves sensibly: What it accomplishes is relevant for its environment and its task, it is adaptable to varying circumstances and varying goals, it grasps from practice and it forms relevant decisions given visceral restrictions and definite computation. The basic scientific task of computational intelligence is to interpret the basis that creates intelligent attitude desirable, in real or unreal methods. The primary preposition is that logic is computation. The basic structuring task is to define systems for the structure of effective, intelligent antiquities. It is this variance

which determines computational intelligence from other subjective science restraints. CI analysers are attentive in testing common prepositions about the quality of intelligence by creating devices which are brilliant and which don't easily mime humans or systems. This also gives an advent to the query "Can computers certainly realize?" by examining the related question "Can airliners certainly fly?" Crowley (2005).

For this logic, computational intelligence mechanics are now generally being applied, specifically neural computing Miller (1990) & Lewis (1998), evolutionary computing Davidor (1991) & fuzzy systems Lee (1990), as they give energetic tools for the cognizance of good and high capable restriction systems without the demand for definite systems. These mechanics all apply a common restriction structure, with combined criterions that are flexible to enhance the applicable achievement dimensions. These dimensions can wrap the clear demands of speed and veracity, as well as other significant demands such as balance, accuracy and defence.

IEEE Computational Intelligence Society describes its course of attention as neural communications, fuzzy logics and evolutionary communications, counting multitude intelligence. The access considered by the journals and by the articles authors is to feast computational intelligence as a shelter under which high and high systems will be appended. A fine description of the operation is hence impractical, as various people enter or evict various systems under the similar CI caption. Chess chores based on analytical research are once in the supernatural computational intelligence sector, but they do not accord to CI described in such a manner. In the ancient days of CI few experts proved to clearly ignore such troubles. Consider for instance this description: "A sequence is computationally intelligent when it: accord only with arithmetical (small level) data, has a model perception

element, and does not apply intelligence in the artificial intelligence (AI) function" Bezdek (1994).

Systolic arrays are elegantly capable for computationally accelerated functions. If operating as a zealous fixed-program graphics processor or a high multiplexed and adaptable coprocessor communal across a system, a systolic array essentially adventures colossal parallelism. Collapsing into a space among vector computers and colossal lateral computers systolic arrays frequently merge accelerated local network and computing with broadcast parallelism in a firm package. They subsidize on general, compatible, balanced, identical, concerted processes that need accelerated, repeated computation. As systolic arrays basically were applied for settled or certain-objective structures, the systolic idea has been continued to important category objective SIMD and MIMD structures Johnson (1993).

Superior execution, unique-objective personal computers are generally employed to face certain appliance demands or to remove computing that are exclusively tedious to regular-scope computers. Since hardware expense and size progress to crumb and refining demands turn into fine knowledge in fields such as alerts and image processing, more unique-objective computers are being organized. Nonetheless, as majority of these systems are made on ad hoc and support for certain goals, technological effort in this field is limited. As the awareness attained from personal confessions is neither acquired nor perfectly constructed, the similar errors are endless. I/O and computing inequality is an evident instance- always; the case that I/O networks cannot pile up with an equipment speed is invented only after formation a superior speed, unique-objective equipment Foster (1980).

The essentially of algorithms for several digital signal processing (DSP) operation will basically be purposeful by their computing workability.

It bet on recklessly-specifically, for actual-time alert preparing on the lateral processing abilities, in both speed and quantity, provided by form of the trade computing device. The possibility of reduced-cost, increased-volume, high-speed VLSI apparatus, and of the arriving computer-assisted structure efficiencies, forecast a primary improvement in the eventual structure of colossal lateral processor. Existing lateral computer obtain serious system beyond: hence, VLSI-aligned array processors are better imploring for superior-speed signal refining. The structure of similar system needed, nonetheless, a wide education of the affiliation between lateral computer algorithms and the designs of array hardware and software. The borderline amongst software and hardware has turned into progressively unsure in the surrounding of VLSI system structure. This augments the aligning of algorithms to construction. Accordingly, a very wide spectrum of modernisms will be needed for achieving highly lateral array transforming. These modernisms will count modern concepts on transmission, array construction, programming systems, processor/designed aboriginals, and algorithmic attainments of DSP algorithms Kung (1984).

As far as latterly, computing-intensive goals were managed by great completion supercomputers, counting pipelined computers, array processor and multiprocessor integrals. The growth of these computer integrals has tangled a complete examination of lateral computing, dynamic programming, and resource augmentation. Nonetheless, the regular-objective aspect of these devices has driven to difficult system management and cruel system beyond. These devices are not very adequate from electronics perspective for actual time alert processing where a very great throughput value is really important. An elucidation to the actual-time need of alert processing is to employ certain-objective array processor and decrease the processing adequacy by either pipeline processing or lateral processing or

these both. As far as transmission in VLSI stays confining, locally-hooked array will be of huge significance. A raise of capability can be conventional if effort charge while penetrating the demand of vicinity, i.e. short transmission ways these resources of the inventor of VLSI algorithm, and ultimately drive to new structures of construction and communication. The initial such certain-objective VLSI construction is systolic architecture. The systolic architectural idea was refined at Carnegie-Mellon University and forms of systolic processors are being structured and made by many industrial and governmental enterprises Selvakumar et al (2015).

1.1.1 High Level Synthesis

There are several interesting algorithms for analogy circuit synthesis, which can be determined organizing and obligatory issues. But these algorithms are time destruction for huge structures and they cannot recognize many pressures concurrently. Three eugenic algorithms are refined to determine organizing, module obligation and record allotment issues, and then a co-progressive design combines the output of these three algorithms, focusing augmentation of the structural criterion. Stimulates in VLSI technology and processing to wide sub-micron components, have convoluted the synthesis criterion. Typical synthesis algorithms were formed to enhance the field and detention of synthesized region. Nowadays some modern structure parameters expansion such as power destruction and proof optimization are of significant. High-level synthesis is distressed with the form and application of circuits from a detectable confession expose to a set of tasks and pressures. Two primary goals achieve in high level synthesis, one is scheduling and the other one is mapping. Scheduling appoints functions to clock periods and resources allotment or mapping is distressed with appointing functions and differences to hardware. At the time of allotment, registers are allotted to stock volatiles, functions hired to operational units,

and networks which are circuitous, busses, or a sequence of both, are applied for association Banaiyan et al (2006).

The action of a circuit can be described using an extreme-level hardware definition language, and it should be adapted into an appropriate common format, for instance, a flow graph. In the detectable stipulation, a key block is described as a direct line flow of statements that does not involve sections. Therefore, a data flow graph (DFG) can be adopted from the key block where every node is linked with a function and every arc is connected with a volatile. Multiplicity of operations enforced on installed and programmable methods develop with the stimulations in dimensions and abilities of these methods. Mapping operations onto them standardly is enhancing a very annoying goal. This pulls scrutiny to applying high-level synthesis not over structure flows. Concurrently, it is important to offer a bending conception of optimization purposes as well as to achieve adequate outlining for different form intentions early on in the structure flow. In this effort, we notice these problems in the situation of data flow graph (DFG) organizing which is an important aspect inner the high-level synthesis flow. An algorithm that organizes a sequence of functions with data responsibilities among successive functions at one operation is promoted Farland et al (1990).

1.1.2 VLSI Array Processor

Immense speed signal processing comfort seriously on lateral processor method. In several operations, regular-objective lateral computers cannot always gratify actual-time refining speed owing to detach system beyond. Hence, for actual-time digital signal processing (DSP) methods, certain objective array processors have incline the only engaging substitute. In structuring or applying such array processor, several alert processing algorithms split the serious aspects of precision, recursiveness, and local

language. As far as latterly, computing-intensive goals were managed by great completion supercomputers, counting pipelined computers, array processor and multiprocessor integrals. The growth of these computer integrals has tangled a complete examination of lateral computing, dynamic programming, and resource augmentation. Nonetheless, the regular-objective aspect of these devices has driven to difficult system management and cruel system beyond. These machines are not adequate for actual-time alert processing where a very throughput value is really important. The initial such certain-objective VLSI construction is systolic and wave front array, which bluster extremely colossal readiness. The readiness in the systolic/wave font array is adopted from pipeline processing or lateral processing. The systolic array is very responsive to VLSI application. It is particularly useful to a certain kind of computing-bound algorithm, catching benefit of their normal, bounded data progress. As several certain-objective chips will be formed in comparably small volumes, the structure value must be set low. Systolic algorithms have many benefits which support to minimize this price Fisher (1985).

1.1.3 Systolic Architecture

Superior execution, unique-objective personal computers are generally employed to face certain appliance demands or to remove computing that are exclusively tedious to regular-scope computers. Since hardware expense and size progress to crumb and refining demands turn into fine knowledge in fields such as alerts and image processing, more unique-objective computers are being organized. The systolic constructive idea was refined at Carnegie-Mellon University and forms of systolic processors are being formed and made by many industrial and governmental enterprises Gentleman & Kung (1982).

1.1.4 Systolic Array Processor Developments

The association of systolic array processing methods and VLSI invention pledges to offer regularity in the application of matrix functions for alert-processing with throughput enhancing with the count of cells applied. In active to perform this, nonetheless, several structures accord must be formed.The strip implementation of a systolic processing aspect will need affability in its structure if it is to be formed in huge sufficient volumes to reduce the unit value so that huge arrays can be formed Kung et al (2012).

1.1.5 Systolic Architecture Design Methodologies

Systolic architectures produce a connection of processing aspects that delicately computer and evade data over the system. These processing aspects generally push data in and out so that a normal progress of data is sustained. As an output, systolic methods trait modularity and uniformity, which are considered to be the significant aspects for VLSI structures? The systolic array may be applied as a coprocessor in association with an anchor computer. The processing units in a systolic array are regular and completely pipelined, i.e., all networking points between the processing units have lag aspects, and the entire system normally have only local interrelationships. In the systolic architecture form technology where several systolic architectures can be formed for any produced normal repeated algorithm applying straight mapping or projection methods. The straight systolic mapping technology is imported to acquire systolic array which assume how the organizing point is preferred and the mapping method to habituate normal repeated algorithm that edge to dependency graph with lags. Existing lateral computers can be classified into three architectural divisions, the first one is vector processors, the second is multiprocessor methods, and the third is array processors. The first and the second systems accords to the regular-objective computer

section. The growth of these methods needs a difficult structure of restriction units and optimized programs for the allotment of device leverages. The third system, nonetheless, accords to the section of certain-objective computers, and the structure of similar methods needs a wide awareness of the affiliation among lateral computation algorithms and optimal computation hardware and software designs Parhi (2007).

In specific, locally mutually dependent computing systems, such as systolic and wave front arrays, are fine fitted to their colossal parallelism a large section of alert processing algorithms, owing to their colossal parallelism and normal data progress. Such formations assure actual-time results to a major collection of modern computational targets. Pipelining, array processing, and multiprocessing produce definitive systems in computer architecture which are regularly applied for superior-speed processing, to minimize the latent intricacy in the structure of huge-scale multiprocessor arrays. Different solutions have been suggested, which all hook above massive a specific grade of certain-objective control over the operation. For instance, mutually dependent architecture at the cost of a little controlled section of operations. Single instruction multiple data stream (SIMD) computers, multiprocessors, and VLSI arrays are delegates of different trials at determining the lateral processing issues. Common cases are the ILLIAC IV for the SIMD array processing, dataflow devices for multiprocessor methods, and systolic/ wave front processors for VLSI arrays(Jha CK 2009).

1.1.6 Systolic Array Processor (SAP)

The systolic array approach comprises the internal steep throughput and integrity produced by a grid of similar processing units. The possibilities for formulating an array unit (or many units) on a lone chip turn up very superior. Nonetheless, several particulars of the algorithms, data

stream, restriction, input/output, numerical efficiency, speed, etc. have to be steadfast before a specific chip structure can be attempted. The task of this effort is to make a systolic array test bed which is versatile sufficient to permit examination with algorithms and architectures so that brilliant judgments can be formed when it attains time to define the chip structure for a certain set of operations. The hardware application is shown visionary in Figure 1.1. The array comprises of a closet involving an array of systolic processor circuit boards, a mother board, and a counter for the host-interface components Kung et al (2012).

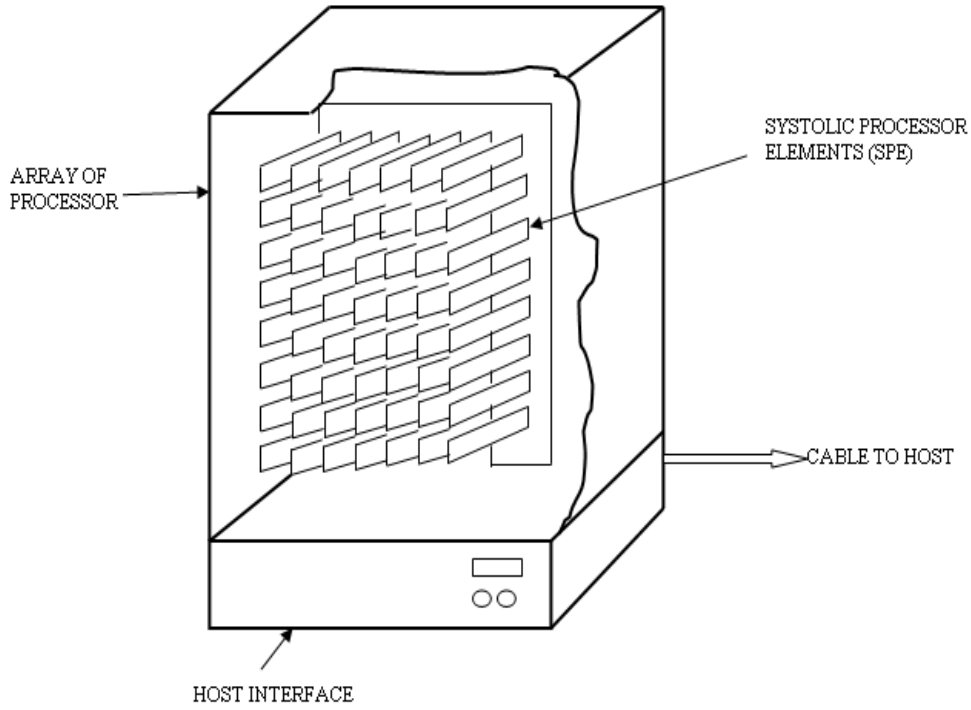


Figure 1.1 Systolic array configurations

1.1.7 Design History and Rationale

In the first structure of this test bed, several queries respecting contracts and opinions had to be replied. For example, should one use bit-

serial or bit-parallel computing? What progressive range is essential for a beneficial processor? What speed of computing is acceptable? How complicated or wise should every processor dwell? How many units will be acceptable to hook up? And what will be the confine access? Only the end structure will be granted this, without defining the several feasible substitutes. As the determining matter in this test bed is for adaptability to permit research, it was destined to apply a microprocessor with EPROM and RAM, to permit the ultimate programmability in the systolic processing units Kung et al (2012).

As for the effective temper, deliberations of alert processing uses drive us to fix on a 32-bit sailing-point capacity. Here a contract among software and hardware application drive to the benefit of an arithmetic processing unit (APU), e.g., Intel- 8231 or AMD-9511.

Bit serial computing was examined as a probability as of its substantiality and less pin count. Nonetheless the demand for broad effective range and the elongate structure period a bit-serial structure would need to form the bit-lateral access of the APU further interesting.

The kind of interaction way to the SPE (serial or parallel) consequence the speed of function and the hardware needed for an array. Serial transmission was raised to be further beneficial for many logics. As every SPE has six I/O ports, an eight-bit lateral path would need 48 pins and 48 drivers/receivers bolster in every SPE. Additionally, as in VLSI structure, inter linkage could turn into a dominant issue. Moreover, to access adaptability in the inter transmission of the array; multiplexers have been located in some of the data ways. The benefit of lateral transmission would have efficiently maximized the volume of hardware needed Kung et al (2012).

1.1.8 The Systolic Processor Elements (SPE)

The blocklayout for the systolic array processing units is exposed in the Figure 1.2. The microcontroller applied is the Intel 8031. This gadget was picked for its speed, domestic RAM, several I/O pins and content of bit and byte direction. The data acts about in the SPE through a multiplexed 8-bit data/address bus.

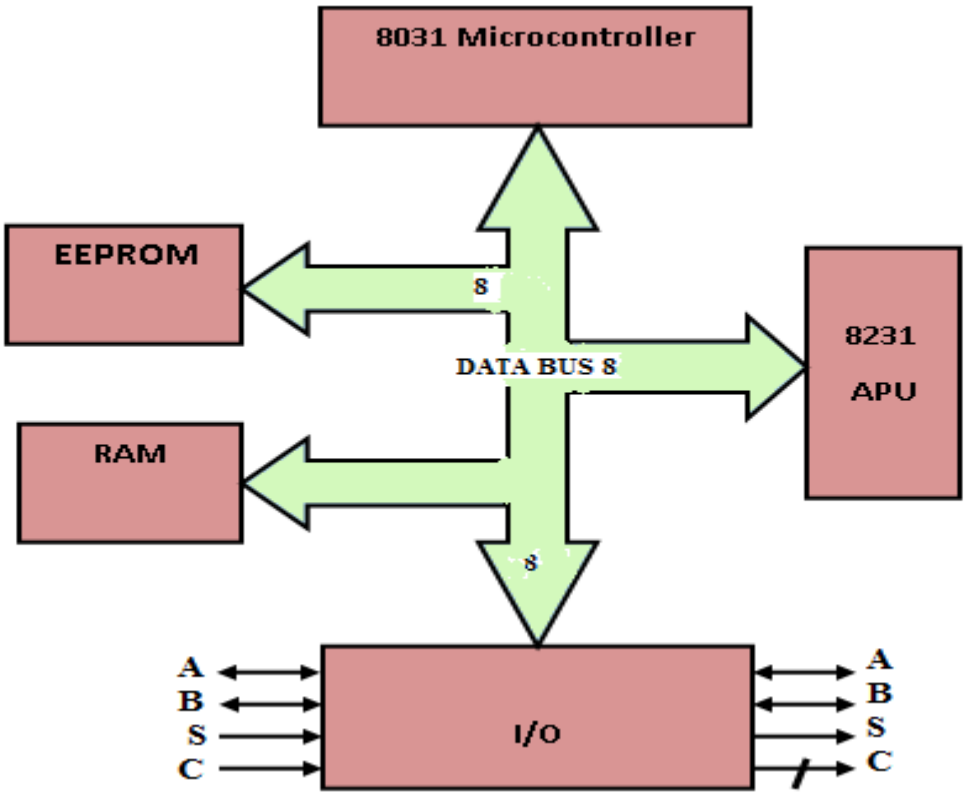


Figure 1.2 Systolic processor element block diagram

Computing is practiced in the arithmetic processing unit (APU). This element or unit is proficient of many structures (16-bit fixed point, 32-bit fixed points, and 32-bit sailing points). Several functions are possible such as addition, subtraction, multiplication, division, square root, many trigonometric applications, etc. The EPROM and RAM are common gadgets.

The EPROM is 4k*8 bits. The RAM is 1k*8 bits. The EPROM is benefit to save cycles which achieve data administration. This produces the system a graded advent so that an individual byte broadcast to the processor begins a course of functions. The RAM can be utilized for data stockpile. This impacts in a ‘3rd dimension’ of matrix gathered which is essential for sectioned matrix function. The RAM can also be utilized for gathering of schedules during algorithm growth. One the algorithm have been fulfilled, they will be insert into EPROM. The I/O from the processor is bit sequent. The major logic for serial I/O is to reduced pin outs and driver/receiver demands. Four global 8-bit lateral/serial shit annals compose the I/O ports. Three annals are applied for the A, B and C data. The fourth annals or register, the S register, accepts the information; transmit along a line, which reveals the SPE which cycles to operate. The I/O registers are consigned and learn by the microprocessor beneath program restriction Pasca (2011).

1.2 PROBLEM STATEMENT

Advanced processing applications have generated an increasing demand for highly scalable, localized and efficient computing architecture. The main aim of evolving new computer architectures is to run more complex application faster than ever before. The continuous demand for increased computing power makes several researchers to develop the high parallel scalable multiprocessing systems. In general, systolic architectures are suitable for implementing parallel algorithms in various fields of science and engineering. There are two fundamental ways to improve the performance of computer with respect to computational speed. First method is to use faster devices is VLSI chips. Systolic architecture is one of the best possible options to develop parallel processing architecture. The second method is to use the parallel processing architectures which enable multiple processors in order to perform computation task. Thus in this study, two techniques are proposed in

order to enhance the performance of systolic array design. There are several challenges associated with the developing systolic architecture. The systolic architecture characteristics and qualities are completely depends upon mapping vectors and manual exploration to achieve the optimal vectors is very difficult. Processing units should have the maximum processor utilization efficiency which depends on the pipelining period, the block pipeline period and data input scheme. Here, this study intends to develop efficient automated solution environment for designing efficient systolic architecture which using nature inspired computations like, genetic algorithm, evolutionary programming, swarm intelligence and chaotic optimization. In this study, multiple automatic design vector generation for efficient systolic architecture is developed by using NIC and systolic architecture is developed by using the hybridization of GA and local chaotic optimization.

1.3 OBJECTIVES OF THE STUDY

The following are the primary and secondary objectives of this research:

- The main aim of the study is to propose a technique for the performance enhancement of systolic array design by using the computational intelligence.
- To identify the importance & possibilities of systolic array in the various regular iterative algorithms.
- To explore the possibility to integrate the computation intelligence in designing of systolic array architecture.
- To develop an efficient automated solution environment for designing efficient systolic architecture which using nature inspired Computations like, Genetic algorithm, Evolutionary Programming, Swarm intelligence and Chaotic optimization.

1.4 SIGNIFICANCE OF THE STUDY

This study intends to propose the technique for the performance enhancement of systolic array design. Until recently, computation-intensive tasks were handled by high performance supercomputers, including pipelined computers, array processor and multiprocessor systems. The development of these computer systems has involved a through exploration of parallel computing, efficient programming, and resource optimization. However, the general-purpose nature of these machines has led to complicated system organization and severe system overheads. These machines are not very efficient from electronics point of view for real time signal processing where a very high throughput rate is absolutely essential. A solution to the real-time requirement of signal processing is to use special-purpose array processor and minimize the processing concurrency by either pipeline processing or parallel processing or both. As long as communication in VLSI remains restrictive, locally-interconnected array will be of great importance. An increase of efficiency can be expected if work load while observing the requirement of locality, i.e. short communication paths these properties of the designer of VLSI algorithm, and eventually lead to new designs of architecture and language.

1.5 CHAPTERIZATION PLAN

The following is an overview of the contents of the chapter that presented in this research

Chapter 2 Chapter 2 provides a detailed literature survey on systolic array, its importance and properties of systolic architectures.This is followed by the review of the types of parallel computers and features of systolic arrays. Also the recent work in systolic

mapping methodologies is reviewed.

Chapter 3 Chapter 3 gives discussion to understand the evolutionary computation from level of requirement to stage of implementation a design. Different parts of evolutionary computation are explored to understand the befit and usage in developing the solutions. Different applications of evolutionary computation is reviewed.

Chapter 4 Chapter 4 gives flow for mapping iterative algorithms and designing a systolic array by using three vectors projection vector, processor vector and scheduling vectors.

Chapter 5 Chapter 5 gives the exploration for designing systolic array using evolutionary computation and Chaotic optimization. Systolic array is designed for iterative algorithms using the vectors generated ny HGACH method.

Chapter 6 Chapter 6 is the conclusion chapter that describes about the summary of findings obtained through the analysis and also provides conclusion to the research followed by recommendations for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter provides an overview to the study on systolic array and also it discusses about the importance and properties of systolic architectures. Apart from these, this review discusses in detail about the types of parallel computers and features of systolic arrays. In addition to these, this chapter also discusses about the advantages and disadvantages of systolic arrays.

2.2 OVERVIEW OF SYSTOLIC ARRAY

"A systolic system is a connection of processors which delicately compute and ravine data over the system. Physiologists use the term 'systole' to mention to the delicately periodic deliberation of heart and arteries which flows blood over the body. In a systolic computing method, the operation of a processor is comparable to that of heart. Each processor frequently flows data inside and outside, every time achieving some brief computing, so that a frequent pulse of data is maintained up in the connection" Davis (1984).

"A systolic array is a computation network acquiring with the aspect of: synchronous, compatibility and uniformity, structural locality, pipeline capability, reproducibility and superior parallelism".

"A systolic array is an interaction of processors in which the processors can be located at network edge of a bound so that:

- Topologically: if there is focused link from the processor at place *I* to the processor at place (*I*+*d*) for some *d*, then there is such a link for each *I* inner the grid.
- Computationally: if a processor accepts a credit on an aid link at time t, then it accepts a credit at time (t+Δ) on the interrelated yield link, where Δ is time length that is autonomous of the communication size, the adaptation of the link and the place of the processor(Davis 1984).

2.3 IMPORTANCE AND PROPERTIES OF SYSTOLIC ARRAY

The primary aspects of accepting systolic arrays for certain-objective processing structures are: Simple and formal structure, adequacy and interaction, and compensating computing with I/O Sawartzlander (2012).

2.3.1 Simple and Regular design

In combined-circuit method, the expense of elements is descending greatly; nonetheless, the value of structure develops with the intricacy of the system. By applying a formal and simple structure and manipulating the VLSI method, huge accumulations in structure cost can be accomplished. Additionally, simple and formal designs are inclined to be standard and hence adaptable to different attainment tasks.

2.3.2 Concurrency and Communication

An essential aspect in the probable speed of a computation system is the benefit of adequacy. For certain objective systems, the adequacy bet on the hidden algorithms applied by the system. When a huge number of processors act unruffled, interaction becomes important. In VLSI structure, chasing costs outshine the power, time, and space needed to gadget a

computing. Hence, formal and narrow interaction in systolic arrays is beneficial Sawartzlander (2012).

2.3.3 Balancing Computation with I/O

A systolic array is commonly used as an adhered array processor, and it accepts data and outputs solutions over an anchor computer. Hence, I/O applications have to be considered into narration in the comprehensive performance. The extreme performance task of an array processor method is a computing rate that stables the accessible I/O bandwidth with the anchor. With the comparably low bandwidth of existing I/O systems, to accomplish a rapid computing rate, it is essential to achieve numerous computing per I/O course. Nonetheless, the constant use of a data point needs it to be reserved inward the system for an enough span of time. On the other hand, the I/O issue impacts not only the needed I/O bandwidth but also the needed domestic memory. The essential algorithmic array structure problem then is how to organize a computing, associate with an applicable memory design and I/O bandwidth, so that computing period is evened with I/O period Lin et al (2011).

Then I/O issue becomes essentially relentless when the computing of a trouble of huge dimension is achieved on a brief array. Necessarily, it comprises a fractioning issue, i.e., the computing must be crumbled. In exercise, this is always the example, and hence, queries such as how a computing can be decayed and how bolster memory can be aligned to reduce I/O are serious to the experience structure of an array processor method.

2.3.4 Clock Distribution Schemes

Integrations of the movements in a high integrating system, is ruled by a method broad clock signal. The clock signal provides two intentions; as a

sequence resource and also as a time resource. In a sequence resource, the clock progresses provide the objective of describing overthrowing currents at which system capacity variations may happen. In a time resource, the time between clock progresses, reports for electrifying and unit lags in ways create the output to input of clocked units. “The twin act of clock signal devotes to some calm of structure of analogy systems; nonetheless, associating progressing and timing so firmly concludes in devising being the cause of various troubles in the structure, regulation, alteration, and accuracy of consonant methods” Kung (1985).

As the clock signal dictums the movements of the complete system, clock scattering is very essential for systolic arrays. There are few problems in integrating a huge array. The issues come primarily from clock bias, i.e., every PE in the array may not accept the clock signal at a single time. This may be because of the various line intensity from the clock producer to every PE. There may be also other logics, such as process differences for various clock paths. To beat clock path issues, an H-tree strategy can be applied to scatter the clock signal to common arrays so that each PE has the equal length from the clock producer. H-tree designs can also be applied for horizontal arrays, square arrays and semestral arrays Fisher (1985).

2.4 TYPES OF PARALLEL COMPUTERS

Unalike the type of the uniprocessor method, for which definite patterns of computing occur (such as RAM) there are several disputing patterns for lateral computation. A lateral computer defines various matters to various people. There are numerous various constructions have been suggested for lateral devices must really stay in hundreds.

Many trials have been performed at division of lateral computers, but there are paper devices that spurn all of the divisions. Hence, the definite uniprocessor is a single instruction single-data stream (SISD) device. An array processor, such as ILLIAC lV,where a firm of equivalent, processors functions the similar application in a lockstep pattern on various data, is known as single-instruction flow, multiple-data flow (SIMD) device. Despite the processing units accomplish every description in lateral, single units may be computed to avoid any specific guidance. This capability to blind out processing units permits integration to be regulated over different ways of restricted designs, such as in the section of an if-then-else description. In the article, always the word SIMD arid array processors are applied identically. Nonetheless, the word array processors have also been applied for algorithmic array processor or peripheral array processor, known as Data west 400, or Floating Point System AP120B, which are not SIMD devices. SIMD patterns can also be categorized on the foot of their mutual dependent, if or not they have a united memory, and if the sum of processors is settled or unlimited Deo (2014).

A third kind of computer is the multiple-instruction flow single-data flow (MISD) device. Pipeline design inclines in this division.

At last, a multiple-processor system subsisting of a majority of completely programmable, autonomous processors, each efficient of functioning a guidance variant from others, is known as MIMD (multiple-instruction stream, multiple-data stream) computer. An MIMD device can be again divided by the grade of coupling-firmly or coupling-lightly, or by memory-processor network scheme-crossbar switch, a bus, or an admixture, if or not there is a master-slave affiliation between the processors, and so on. Flynn's categorization is rude, and many elegant categorization designs for

lateral computers have finally been suggested by Enslow, Higbie, Hobbs et al & Murtha & Beadles, Hockney, and others Deo (2014).

2.4.1 VLSI Array Processors

The experience of all DSP algorithms will basically be obstinate by its computing workability. Actual-time alert and image processing bet on recklessly on the speed and quantity abilities produced by the state-of-the-art lateral processing computers. Owing to critical system beyond, regular-objective supercomputers are always not very applicable for actual-time alert/image processing. Therefore, a new access positioned on VLSI array processors is acceptable enhancing aggressive. Figure that the basic aspects refined by the above-stated algorithms can be abused in a certain kind of VISI array processor structure. These arrays increase the power of VLSI in points of accelerated and pipelined computation and still evade its primary constraint on interaction. This offers the analytical basement for the structure of narrowly mutually connected VLSI arrays like systolic arrays and wave front arrays. The colossal adequacy in systolic/wave front is adopted from pipeline processing, lateral processing, or mixture. This is shown in Figure 2.1. Lateral processing can be defined that all processes described in points of the data D and the instructions i can straightly approaches the m processors in lateral and maintain all processors engaged. Pipeline processing is that process is crumbled into many alternate processes, which are pipelined over m processors regulated in a network, and each alternate process is processed one behind the other. For every alternate process entering out of the array, there will be a processor need and primed to accept and regulate its alternate process instantly. Hence, all m processors can be maintained engaged all the period by using the pipeline method Mertzios (1995).

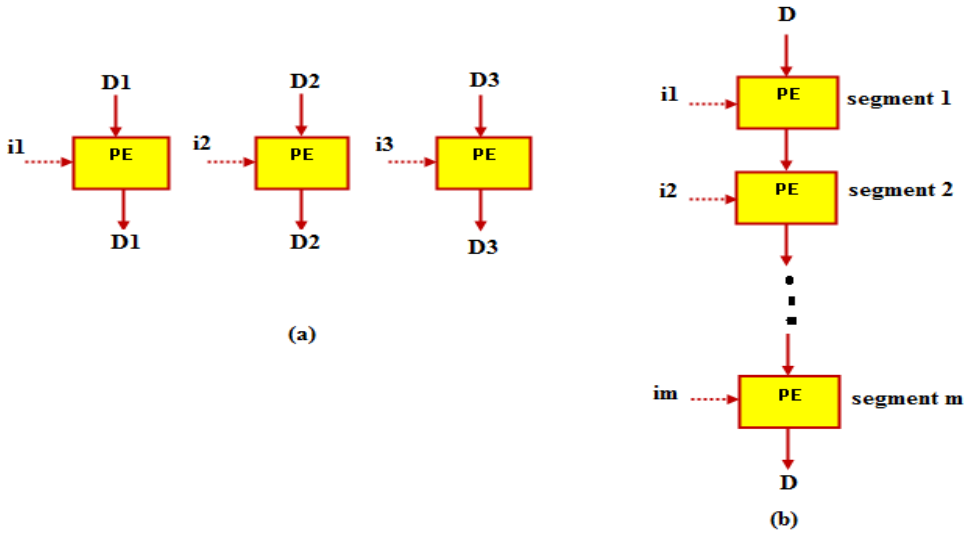


Figure 2.1 Parallel and pipelined processing

Note: Array processors derive a massive concurrency for both (a) parallel processing and (b) pipelined processing

2.5 FEATURES OF SYSTOLIC ARRAYS

A Systolic array is a computation structure occupying the below lineaments:

- Synchronous defines that the data is delicately gauged (Scheduled by a universal clock) and flow over the system.
- Compatibility defines that the array (Limited/Unlimited) comprise of modular or compatible processing elements.
- Uniformity defines that the compatible processing units are mutually connected with uniformly.

- Structural locality defines that the units has a domestic network interconnection.
- Sensual locality defines that the units broadcast the signals from one unit to other which need at least a single unit time lag.

- Pipeline capabilitydefines that the array can perform a huge speed.
- Superior parallelism defines huge troubles can always classified into shorter ones, which are then worked out synchronously (in lateral) Himani & Sidhu (2013).

2.6 ADVANTAGES AND DISADVANTAGES OF SYSTOLIC ARRAYS

Advantages:

It can be applied for certain objective processing structure as of

1. Lucid and Normal Structure

   - Profitable,
   - Array is compatible (i.e.) flexible to different achievement tasks ,
   - Huge number of processors act well-balanced,
   - Local transmission in systolic array is beneficial for communication to be rapid De (2008).

2. Readiness and Transmission

   - Huge number of lucid PEs
   - Regulate without guidance fount

3. Stabilizing Computing with I/O

   - Can’t move rapid than the data appears
   - Decrease bandwidth demands
   - Appoint functions fine!
   - Appoint algorithms accurately

4. Uniformity and modular structure (Excellent for VLSI application).

5. Domestic interconnections (Appliance algorithms domain).

6. Vast grade of pipelining.

7. Extremely integrated multiprocessing.

8. Easy I/O subsystem.

9. Very capable application for good collection of algorithms.

10. Vast speed and reduced cost.

11. Destruction of universal transmitting and modular immensity.

12. Adequately apply VLSI

    - Duplicate simple units
    - Domestic Communication ==>routing expense manages: power, space, and period
    - Concise cables
        - brief lag
        - less clock slant
        - limited drivers
        - fewer space
    - Adaptable
    - Less count of I/Os

13. Decrease “Von Neumann Bottleneck”

    - Progress every input several times.
    - Maintain fractional outcomes in the PEs.

14. Focus compute-accelerated operations(De et al 2008).

Disadvantages:

The major challenges or disadvantages of systolic arrays are:

- Universal synchronism restraints because of signal lags.
- Steep bandwidth demands both for boundary (RAM) and among PEs.

Scanty execute-time defect resistance because of absence of linkage protocol De et al (2008).

2.7 EXISTING STUDIES

Abellard & Abellard (2008) have displayed an outline philosophy of systolic clusters in light of Petri nets. Right now customary techniques, it takes favourable circumstances of utilizing a model exceptionally valuable as a part of various mechanical and examination applications. After a short helping on particular properties to remember essential systolic architectures, the distinctive strides of the strategy are displayed. Compositional Petri nets empower the displaying of a systolic construction modelling, as well as nature interfaces.

Togelius et al (2008) Geometric molecule swarm advancement (GPSO) is a normal speculation of customary molecule particle swarm improvement (PSO) that gives actually to both nonstop and combinatorial spaces. GPSO has connected to the space of hereditary projects spoke to right now, uniting the ideal models of hereditary programming and molecule swarm advancement. The outcome is a molecule swarm flying through the space of hereditary projects.

Soudan & Saad (2008) have investigated two dynamic populace size changes for traditional PSO with the point of diminishing execution time. Extending Population PSO (EP-PSO) began with a LESS number of particles and iteratively builds the swarm size. Decreasing Population PSO (DP-PSO) begins with an expansive number of particles and iteratively diminishes the swarm size.

Yanteng et al (2009) have established systolic exhibit based structural planning with a parallel information giving unit can abuse most extreme parallelism of the full form of P7Viterbi. The anticipated structural engineering theoretically keeps running with completely parallelism expecting that the criticism circle does not occur. If in case, that the uncommon input case really happens, a rollback component is utilized to guarantee rightness.

Toshiaki (2009) has created a plan, utilizing a voracious requesting, to accomplish an execution similar to that of V-BLAST with ideal requesting, while its computational multifaceted nature is lower than a direct detector. In different info various yield frameworks, a requested progressive obstruction canceller, termed the vertical Bell laboratories layered space-time (V-BLAST) calculation, offers great execution. Vertical Bell labs layered space-time displays a low-unpredictability V-BLAST plan suited for parallel execution.

Chuanpeng Chen & Zhongping Qin (2009) have anticipated an equipment implementable calculation and its systolic structural engineering for the longest common subsequence (LCS) issue. On the other hand, most oblige too expansive memory space altogether, bringing about being not suitable for equipment usage proposed versatile systolic structural planning with straight space many-sided quality for the LCS issue is suitable for

equipment usage, and the incorporated results demonstrate that the modelling is more effective.

Hegen Xiong et al (2009) have portrayed a new quantum hereditary calculation called variable-limit coded quantum genetic calculation (vbQGA) in which qubit chromosomes are fell into variable-limit coded chromosomes rather than paired coded chromosomes. Along these lines acquire much shorter chromosome strings. The technique for encoding and unravelling of chromosome is initially portrayed before another versatile choice plan for point parameters utilized for turn entryway is set forward in view of the centre thoughts and standards of quantum calculation.

GuoLiang Chen et al (2011) have provided an overview the flow status of coordinated examination technique for parallel figuring and by joining the effect of multi-centre frameworks, distributed computing and individual superior PC, the study introduce an attitude toward the future advancement of parallel registering.

Wolf et al (2009) have developed an axon division calculation, taking into account a fluffy controlled framework. The issues that show up, is that a right setting of the guideline set can barely be known, so they demonstrated to streamline the standard set with transformative calculations.

Julien et al (2010) have elaborated a work with the outline of a two-dimensional (2D) systolic exhibit for picture preparing. This part is intended to work on a multi-processor framework on-chip. In similar with other 2D systolic-exhibit architectures and numerous other equipment quickening tools, they have examined the appropriateness of executing different errands in a period interleaved form on the Systolic Array (SA). This prompts a lower outer memory transmission capacity and better load adjusting

of the works on the diverse preparing tiles. To empower the interleaving of errands, the researcher includes a shadow-state list for quick errand exchanging. To decrease the quantity of contact to the outside memory, plans the thought of offer the correspondence help between successive works. According to the work by Liang et al (2010) given an examination concerning systolic exhibit plan in QCA innovation. A contextual analysis of the first systolic network multiplier is planned and dissected. The outcomes demonstrate that by applying the systolic cluster structure to QCA plans, critical advantages can be accomplished specific with extensive systolic exhibit size, considerably more so than when connected to CMOS-based innovation. QCA has noteworthy benefits regarding the speed and place over CMOS innovation.

Jiang (2010) have elaborated the thoughts to upgrade the nature of hereditary calculation by fusion of calculation of an inclination calculation joining with the hereditary calculation be offered against to the deformities, for example, the prematurity, moderate on the meeting rate, feeble in the capacity of nearby pursuit, each one of those showed up on the advancement of the hereditary calculation emphasis. The specialty's innovation be drafted because of the neighbourhood ideal arrangement can undoubtedly show up on enhancement of the various peak quality.

Salih & Arshad (2010) have anticipated the construction modelling and timing strategy to upgrade the execution of systems by expanding the preparing control. This new structural planning outline called Embedded Parallel Systolic Filters (EPSF) can transform information assembled from sensors and milestones are planned in the study by utilizing a high-thickness reconfigurable gadget (FPGA chip). The outcomes demonstrate that EPSF structural planning and bit-banner with a gleam check perform altogether

better in numerous info sensors motions under both ceaseless and intruded on conditions.

Crocker et al (2010) have formulated the Nano magnet rationale based systolic design. Nanomagnet Logic (NML) is a gadget building design that uses the polarization of nano-scale magnets to execute sensible operations. NML has been tentatively shown and works at normal temperature. Since the nanomagnets are stable, as information courses through a circuit, it is characteristically pipelined. This component makes NML an amazing fit for systolic architectures that could empower low-control, high-throughput frameworks that can address an assortment of utilization level assignments.

Okamoto & Hirata (2010) have given a worldwide advancement system in that the concurrent bother inclination estimate is brought into a multi-point sort riotous enhancement strategy so as to figure slope give or take so that the confused improvement technique can be connected to a class of issues whose target capacity values just can be registered.

Pedro et al (2011) have anticipated a computational method for two-scale topology streamlining issue utilizing parallel figuring methods. The objective is to get at the same time the best structure and element, decreasing the auxiliary consistence. An algorithmic methodology is introduced in a suitable manner for parallelization.

From a study of Bekakos et al (2011) the evaluated issue of selecting the most appropriate calculation technique for systolic usage and outlined this issue on the illustration of network increase.

According to Sudha et al (2011) an upgraded mapping of fine-grained systolized sign flow charts (SFGs) for every individual stride of the calculation on to a solitary self-configurable straight systolic exhibit by suitable converging of the processing relating to diverse hubs of distinctive SFGs. The building design has the adaptability of handling face pictures and databases of any size and it is effectively adaptable with the quantity of Eigen faces to be processed.

Jinfeng Zhao (2011) has connected the disorganized enhancement with PSO to stay away from PSO getting into neighbourhood best and seeming untimely union. Due to the inconvenience of moderate meeting and nearby best of particle swarm optimization (PSO), in light of the ergodicity, randomization and authoritarian of disorder and additionally the upsides of Tent mapping, Tent drawing was utilized right now advancement looking and brought into PSO. This adjusted and new PSO was known as chaotic particle swarm optimization algorithm (CPSO). This calculation is connected to unravel the most extreme entropy model, evaluating framework from activity connection streams.

Fernandez and Garcia (2011) have pointed out the stochastic examination of the direct consistent and summed up PSO models for the instance of a stochastic middle of fascination. Examination of the GPSO second request directions is performed and clears up the parts of the PSO parameters and that of the expense work through the calculation execution. While the PSO parameters principally direct the eigen-values of the dynamical frameworks included, the mean direction of the focal point of fascination and its covariance capacities with the directions and their subsidiaries (or the directions in the close past) go about as compelling terms to redesign first and second request directions.

Tsujimoto et al (2011) have given an sanctioned deterministic PSO (abbr. Album PSO) frameworks that does not have any stochastic elements, and it's arrange of the stage space is standardized. They discovered worldwide best data impacts the motion. This circumstance can be viewed as the right now association state. Then again, there is the situation where the best data in a restricted populace. Those data is called as the Ibest. Step by step instructions to get the Ibest data from any populace is identical to a system structure. Such system structure impacts the execution of looking capacity.

As Feng Chen et al (2011) have anticipated an enhancement in PSO by Inspiration of the trade-off methodology in the middle of investigation and misuse in fortification learning,. The sigmoid capacity is fused into the speed redesign mathematical statement of PSO to handle these downsides of PSO.

Lopez-Parrado & Velasco-Medina(2012)have displayed a systolic construction modelling for taking care of the Hermitian eigenvalue issue that is a typical issue in the range detecting capacity of subjective radio frameworks like WRANs. The composed structural engineering uses a new upper triangular structure to enhance execution regarding the BLV cluster. The handling components are taking into account of the successive CORDIC processors to perform two complex Jacobi revolutions.

Xue et al (2012) have anticipated an enhanced self-adaptive evolutionary algorithm (ESEA) to conquer the negative marks above. In the ESEA, four transformative administrators are intended to upgrade the developmental structure. Moreover, the ESEA utilizes four viable inquiry procedures under the system of the self-versatile learning. Four gatherings of

the investigations are carried out to figure out the most appropriate parameter standards for the ESEA. .

Krzysztof Lichy (2012) exhibitedsolution to the issue of outlining an ideal systolic exhibit for morphological preparing and given a proposition of the novel systolic structural engineering with the samples of utilizing new elements.

Zhen Wang & Shuqin Fan (2012) have planned a multiplier one in view of anticipated Gaussian normal basis Montgomery (GNBM) illustration, a semi-systolic even-sort GNBM multiplier. Additionally, because of properties of consistency and measured quality, the proposed multiplier is extremely suitable for VLSI usage.

Renteria Mejia et al (2012) have given the configuration of a 8192-bit RSA crypto-processor utilizing a radix two Montgomery multiplier in view of a systolic building design. For this situation, the Montgomery multiplier all the while performs two duplications, and the crypto-processor does the measured exponentiation by utilizing the twofold exponentiation calculation.

Sharifi et al (2012) have given a synergistic adaptation of cell PSO, named Two Phased cell PSO to speak to a dynamic advancement issues. The planned calculation presented two inquiry stages so as to make a more proficient harmony in the middle of investigation and misuse in cell PSO. The routine PSO in cell PSO is supplanted by a planned PSO to expand the investigation ability and a misuse stage is added to build misuse in the promising cells. In addition, the cell limit edge that is a vital part parameter of cell PSO is not included because of these adjustments.

Liang et al (2011) have given the thought of systolic structural engineering in QCA technology. Quantum-dot Cellular Automata (QCA) innovation is a promising probable distinct option for CMOS innovation. To investigate the attributes of QCA and suitable configuration strategies, advanced circuit plan methodologies have been examined. Because of the innate wire postpone in QCA, architectures that are pipelined give off an impression of being an especially suitable configuration method. Likewise, due to the pipeline way of QCA innovation, it is not suited for a confounded control framework outline. Systolic clusters exploit parallelism, pipelining and basic neighbourhood control.

Forte et al (2013) have given the outline of two systolic designs to assess polynomials of degree in taking the Horner's tenet. The systolic clusters were in light of handling components that procedure standardized and non-standardized information.

Buzdalov et al (2013) have dissected the already proposed EA + RL strategy that improves single-target optimization by choosing the proficient secondary suitable capacities. Decisively, Random Mutation Hill Climber balanced along with Q-learning utilizing insatiable investigation methodology is taken. They have acquired both lower and upper headed for the quantity of sound capacity assessments required for the EA + RL execution to tackle an adjusted one max issue. EA + RL with ineffective secondary sound capacity actions comparable to a routine developmental calculation.

Yamazaki et al (2014) have planned and execute a 3D virtual systolic exhibit to process a tile QR deterioration of a tall-and-thin thick grid. Usage was taking into account a best in class calculation that factorizes a board in view of a tree-decrease. This is a helpful commitment because such

QR corrosion is utilized, for instance, to process the east squares arrangement of the over-determined framework that emerges in numerous experimental and building issues.

Hajar Asgari et al (2014) have anticipated systolic construction modelling for productive actualizing of advanced Hopfield neural systems for taking care of most limited way issue on Field-Programmable-Gate-Array (FPGA). In recent times, the Hopfield Neural Network (HNN) is utilized right now device to tackle briefest way issue in correspondence systems. The equipment usage of computerized Hopfield neural system is an essential assignment.

Schouten et al (2014) have presented a matrix component strategy utilizing design preparing units as a part of the utilization of molecule physical science. The framework component technique uses abinitio estimations of likelihood densities as capable discriminants for procedures of enthusiasm for exploratory molecule physical science. The system has as of now been utilized effectively at past and current collider analyses. Nonetheless, the computational intricacy of this strategy for last states with numerous particles and degrees of opportunity sets it a drawback contrasted with directed order techniques like KK closest neighbour, decision trees or neural systems.

Kevin et al (2014) have exhibited an adaptable configuration for quickening the issue of understanding a thick straight arrangement of comparisons utilizing LU corrosion. A new systolic cluster building design can be utilized as a block that is used as investigative applications was depicted.

Bhanu & Chilambuchelvan (2014) have planned a productive way to deal with outline a 2-D systolic exhibit for high velocity execution of square based elating lossy 9/7 wavelet channel is anticipated. The characteristic favourable position of the set up processing of the lifting-based isolated wavelet change over the customary convolution technique makes it appropriate for effective equipment usage with lower computational intricacy. The column processor comprises of handling components placed in a systolic way, and for section preparing, the elevating steps are registered simultaneously, by drawing the coefficients to the same systolic clusters, utilizing the cyclic symmetry source of the piece of info picture coefficients. The benefit of the structural engineering is that it do not require extra memory for keeping the transitional coefficients. The usefulness of the preparing component in the systolic exhibit enhances the velocity, by having the basic way defer of one multiplier along with the two adders computational point.

Jie Zeng & Wei Nie (2014) have planned a pointer based multi-target improvement calculation, in particular, the multi-goal rearranged frog jumping calculation in light of the e marker. This calculation embraces a memetic meta-heuristic, specifically, the SFLA that is portrayed by the intense ability of worldwide inquiry and speedy union methodology and a basic and viable marker presently task plan to lead the pursuit strategy.

Yan Pei et al (2014) have dissected and examined the connection between enhancement execution of chaotic evolution (CE) calculation and appropriation feature for disorderly parameter. CE is a trans-formative calculation that recreates chaotic movement of a chaotic framework in an inquiry space for actualizing enhancement. Yet, it is streamlining execution, interior procedure instrument and improvement standard are not very much

concentrated on. They have examined appropriation qualities of chaotic frameworks that motivates chaotic factor in CE calculation.

Ke Hou et al (2015) have given investigation of the structural engineering of DIC stage identified with parallel processing stage and, the configuration prerequisites of DIC stage were dissected, the coordinated exploration system for DIC were talked about and after that a building design of DIC stage with seven layers were given.

Mamatha et al (2015) have planned a structural planning for Discrete Fourier Transform that is broadly utilized as a part of sign preparing for spectral investigation, separating, picture improvement, OFDM and so on. Methodology based on Cyclic convolution is one of the systems utilized for figuring DFT. Utilizing this approach a N point DFT can be figured utilizing four sets of [(M-1)/2]-point cyclic convolution in where M is an odd number and N=4M. This study planned a design for complication based DFT and its FPGA usage. Anticipated structural planning includes a pre-handling component, systolic exhibit and a post preparing part. Handling component of systolic cluster uses a label bit to choose the kind of process on the information signals.

Azarderakhsh & Kermani (2015) have introduced idea of Finite field number-statistical operations that have been generally utilized as a part of diverse applications extending two-dimensional corrosion systolic-arranged calculations to create systolic designs for digit-level Gaussian ordinary premise increase and exponentiation over GF(2m). The proposed elite architectures were suitable for various applications, e.g., designs for elliptic bend Diffie–Hellman key harmony plot in cryptography.

Chia-Hsiang et al (2015) have given generalized triangular deterioration processor structure design with a similar algorithm. GTD (Generalized Triangular Decomposition) has been identified to be helpful in the side of sign preparing, however the attainability of the related equipment has not yet been built up. The planned parallel GTD calculation accomplishes an increment in velocity, contrasted with the velocity of its ordinary successive part for a 8 ×8 grid.

Saha et al (2015) have given a versatile transformation is proposed for multi- objective PSO and named it as AMPSO. In AMPSO, change is connected on the position and speed of the elements taking into account on the wellness estimations of the elements.

Erlei et al (2015) have proposed a variant of the concurrent orthogonal coordinating interest to take care of the previously stated issue due to its advancement with strong merging assurance and proficiency. Besides, to further enhancements the grouping execution, the researcher fuse relevant neighbourhood data of the picture into every sort of highlight. Contrasted with the cutting edge calculations, it has been demonstrated that the proposed calculation with significantly less memory prerequisites performs many times quicker than those on true HSIs, while giving the same (or far and superior) exactness.

Li et al (2013) have proposed a target with fuzzy-multi enhancement model with linked requirements to minimize the aggregate monetary cost and system failure of microgrid. Disorderly micro sources are taken as negative pact, and stochastic net burden situations were created for considering the instability of their yield load and power. Collaborating with capacity tools of the ideal limit manageable micro sources are dealt with the improvement process with the deliberation of their initial and final procedure.

Into the binary element swarm optimization (BPSO), chaos optimization algorithm is introduced to proposed disorder BPSO .

2.8 CURRENT PROBLEM

Systolic arrays generally have high rate of I/O and it is well suited for the intensive parallel operations. Systolic arrays are used in various applications including language recognition, signal processing, relational database operations, matrix arithmetic, character string manipulation, data structure manipulation and image processing. In general, Systolic arrays' high I/O requirements made system integration as one of the significant problems. Systolic array are designed based on the dependence graph (DG). Then the dependence graph will be transformed to space – time representation. Here, the transformation techniques are mainly based on three vectors such as Projection Vector, scheduling Vector, and Processor Vector (Faizulla and Venkateswarlu, 2014). Edge mapping is the next process in designing the systolic array and finally it is possible to construct the systolic architecture. Thus, it is clearly understand, design methodology of systolic architecture needs to determine three critical vectors such as projection vector, scheduling vector, and processor vector. So, there is a strong need for developing an automated and efficient approach for designing systolic array.

2.9 NEED FOR SOLUTION:

Bhanu & Chilambuchelvan (2014) also studies about the productive way to deal with outline a 2-D systolic exhibit for high velocity execution of square based elating lossy 9/7 wavelet channel. The systolic exhibit enhances the velocity, by having one multiplier along with the two adder computational point. Liang et al (2013) conducted a study on systolic structural engineering with respect to QCA technology. Author identified that, pipeline way of QCA innovation is not suited for confounded control framework outline. Systolic clusters exploit basic neighborhood

control, parallelism, and pipelining. Thus, several authors focused towards the systolic array but they failed to develop an approach enhancing the performance of systolic architecture by using the computational intelligence. Thus, this research developed an approach for enhancing the performance of systolic architecture by using the computational intelligence

2.10 SUMMARY

This literature review has clearly brought about the fundamental overview to the study on systolic array from different dimensions like importance and properties of systolic architecture,different designingpossibilities, challenges associated with current trend of development procedure and explored the possibilities for further research .Undoubtedly in past several researchers have conducted their research in order to develop efficient systolic array with various means for different applications. At the same time, there is a strong need for efficient and effective techniques for enhancing the performance of systolic array. In general, increasing demands for highly scalable and efficient computational architecture makes researchers to propose various new techniques. The gaps identified in previous researches were addressed in this study and explained clearly. Analyzing the existing researches, this research aims to propose techniques for enhancing the performance of systolic architecture based on computational intelligence approach

CHAPTER 3

EVOLUTIONARY PROGRAMMING

3.1 INTRODUCTION TO EVOLUTIONARY COMPUTING

Evolutionary computing has accomplished a fantastic growth in the recent decagon in both analytical research and industrial functions. Its purview has emerged above its primary definition of biological growth. Approaching a broad collection of nature excited computing algorithms and methods, counting evolutionary, general, ecological, communal and economical computing, etc. in a combined structure, EC are the research of computational method which use concepts and get influences from nature growth and accommodation. It is a rapid developing associative analysis region in which a collection of techniques and systems are learned for handling with large complicated and effective troubles Xu (2005).

The major goals of EC are to recognize the structure of such computational methods and tobuild highly potent, adaptable and dynamic algorithms for resolving true global issues that are commonly are very complex for traditional computational systems. EC was basically classified into four classes: evolutionary strategy (ES), evolutionary programming (EP), genetic algorithm (GA) and genetic programming (GP). In present days all the ways applied in EC engage a community oriented search engine with disruption (e.g. crossover and metamorphosis) and denotation (selection and replication) to the greater solutions related to traditional optimization systems. The primary benefits of EC access counts: theoretical and computational integrity, wide application, skilful true global issue dealers, capable to use domain intelligence and infuse with other systems, parallelism, powerful to

effective environments, efficiency for self –development, ready to solve troubles with unaware solutions etc. there are still few other benefits with EC advents e.g. no requirement for analytic interpretation of the trouble, derivatives, etc Yao (2006).

3.2 HOW DO GENETIC ALGORITHMS WORK

3.2.1 Primary Concept

Genetic algorithms were initially advanced by Holland (1975) and are primarily used as search and optimization systems. Provided a huge solution area, one would decide to choose out the mark which progresses an object activity while still satisfying a firm of pressures. In system outlining, a result point could be certain link topography, a routing line design, or a thorough quantity avocation with low costs. Genetic algorithms are positioned on the concept of natural enactment. In general, the resources of an organism are arbitrated by its genes. Outset from an odd first genesis with all types of feasible gene systems, natural selection advises that during the period, humans with "fine" genes exist while "poor" genes are denied. Genetic algorithms attempt to carbon this basis by coding the probable solution substitutes of an issue as a genetic strand. The genes can be chips, integers, or any other kind from which a particular result can be presumed. It is needed that all result points can be characterized by at least one strand. On the other aspect, a certain gene string points to literally one result. A firm of a table number of gene strand, each representing one individual, is known as a generation. As the various strings have to be calculated and related to one another, the concept of fitness is made known. The fitness value corresponds to the nature of a specific solution. In this case, it is reversely proportional to the value of a system. Contrary to struggling with the true result itself, genetic

algorithms function on the relevant string characterization Riedl (1998). The below three common operators are used:

- Reproduction
- Crossover
- Mutation

The replication or reproduction system forms a new genesis. Starting from a current generation, strings are replicated with a possibility corresponding to their vigour rate. Strings which mean explications with good effects have a greater fortunate to sustain than strings representing solution marks with poor representations. This basis is also named as "survival of the fittest". The crossover executor transfers genetic messages among two strands. The strands of two regardless picked solutions are busted up at an also regardless picked situation, and units of the strings are transformed. One prospect is that two explications with fine effects form an even superior one. Latest genetic objective is imported by the mutation executor. The codes of single genes are reversed and, thus, new explications are picked. Mutation turns into essential when next few generations the count of various stands reduces as robust individuals initiate asserting. In a position of robust dominance of some strands, the crossover executor alone would not import any variations and the hunt for a flawless solution would be completed. To moderately transfer the hunt to new position in the solution area, the mutation executor regardless shifts genes Mehta & Midhuna (2014).

3.2.2 Application of Genetic Algorithms

The flow chart in Figure 3.1 displays the string of the essential executors applied in genetic algorithms. We initiate out with a regardless picked early generation. Each string in this generation is calculated respective

to its character, and a fitness rate is appointed. Later, a new generation is formed by using the reproduction executor. Couple of strings of the modern generation is picked and crossover is achieved. With a specific possibility, genes are modified before all explications are calculated again. This practice is redone until increased counts of generations are attained. Despite performing this, the everlasting best explication is saved and revolved at the edge of the algorithm.

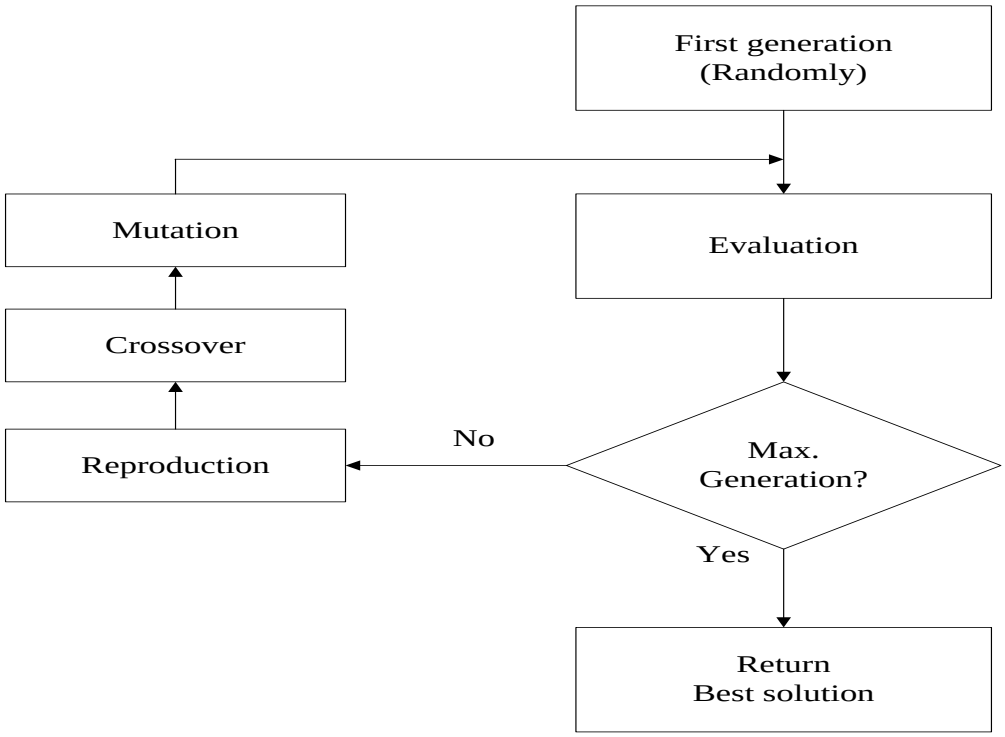


Figure 3.1 Genetic algorithm flow chart

Since one marks from the flow chart, the genetic algorithm provides as a structure which gives the exterior cycle of the hunt or development courses. An essential part of the hook is the calculation exercise which completes the fitness rate of a certain string. Within this system, the string has to be charted to a pragmatic solution, and the object operation has to be calculated. To attain this, interrogative systems might be essential Stephen et al (2010).

3.3 EVOLUTIONARY COMPUTATION AND ITS USES

Evolution is in core of two-step actions of random changes and election. A community of individuals is defined to an environment and behaves with a number of attitudes. Few of these attitudes are fine fitted to acquire the needs of the environment compared to others. Choosing tends to ignore those individuals that manifest improper attitudes. The residues replicate, and the genetics concealed their behavioural habits are passed on to their progenies. But this reproduction is never left out lapse, nor can single genotypes stay free of aimless mutations. The initiation of aimless genetic changes consecutively leads to unusual behavioural peculiarities, and the course of evolution repeats. Over subsequent generations, progressively relevant behaviours acquire within deriving genetic lines. Evolution develops attitudes (i.e., the phenotype), not the basic genetics per itself, as choosing can move only in the aspect of phenotypic difference. The presence in which operational adaptations are concealed in genetics is clear to election; only the recognized behaviours appearing from the synergy of the genotype with the ambiance can be determined by aggressive picking. Benefit changes have the great scope of being conserved in the battle for life, noted to a process of endless advancement Fogel (1994).

Evolution may literally form "organs of intense perfection and complexity", but must often move within the pressures of physical growth and the ancient disasters of life that guide the existent population. Evolution is completely strategic and can only act within the differences present in actual entity. The course of evolution can be shaped algorithmically and imitated on a system.

In the most underlying of patterns, it may be abstracted as a variance

$$x\,[t + 1] = s\,(V(x[t]))$$

where the denizens at time, t, expressed as *x[t]*, is functioned on by aimless variation, v, and selection, *s*, to give growth to a new denizens *x[t* + 1]. Natural evolution does not appear in irregular time layoffs, but the utilization of an analog computer needs various actions. Over subsequent emphasis of variation and selection, a metamorphic algorithm can spin a denizen toward specific vertex on a feedback surface that means the perceptible benefit of each feasible individual that might dwell in a population. Evolutionary computing is the area that researches the effects of these algorithms and related strategies for replicating evolution on a computer.

Despite the term evolutionary computation was established as newly as 1991, the area has a story that extents four decades. Several autonomous efforts to replicate evolution on a system were given in the 1950s and 1960s. Three widely identical avenues of examination in replicated evolution have subsisted as primary fosters inner the field: evolution strategies (ES), evolutionary programming (EP), and genetic algorithms (GA). Each starts with a population of confronting testing solutions led to a goal at hand. Late solutions are formed by nevertheless varying the current solutions. An equitable measure of achievement is applied to determine the "fitness" of each testing solution, and a selection method arbitrates which solutions to recognize as "parents" for the successive generation. The variances among the strategies are represented by the common data illustrations, the kinds of variations that are appointed on solutions to form offspring, and the systems applied for choosing new parents. Supplementary, nonetheless, these variances have become progressively obscured, and will inclined become of only ancient passion Back et al (1997).

Evolutionary computation uses computing patterns of evolutionary strategies as vital aspects in the structure and application of computer-oriented issue solving methods. There are a collection of evolutionary computing models that have been scheduled and analysed which we will cite to as evolutionary algorithms. They contribute a usual visionary base of replicating the evolution of human designs via strategies of selection and replication. These strategies depend on the recognized performance (fitness) of the human structures as described by an ambiance. More accurately, evolutionary algorithms regulate a population of designs that emerge subsequently to orders of selection and other executors, such as reunion and mutation. Each human in the population acquires an amount of its fitness in the ambiance. Selection targets diligence on large fitness humans, thus manipulating the possible fitness messages. Reunion and mutation agitate those individuals, giving common searching for analysis. Despite condensed from a biologist's aspect, these algorithms are adequately difficult to give strong and capable flexible search appliances. Figure 3.1 drafts a common evolutionary algorithm (EA). A denizen of individual designs is started and then emerged from genesis to genesis by imitated operations of assessment, selection, reunion, and mutation. The denizen size N is commonly stable in an evolutionary algorithm, despite there is not a priori logic (other than comfort) to form this acceptance Brazdil (1993).

An evolutionary algorithm basically starts its population nevertheless, despite domain particular awareness can also be utilized to tilt the hunt. Assessment codes the fitness of each personal subsequent to its benefit in some ambience.

```
procedure EA; {
    t = 0;
    initialize population P(t);
    evaluate P(t);
    until (done) {
t = t + 1;
Parent selection P(t);
recombine P(t);
mutate P(t);
evaluate P(t);
survive P(t);
  } }
```

Figure 3.2 A typical evolutionary algorithm

Assessment may be as easy as measuring a fitness operation or as difficult as working a detailed replication. Selection is always achieved in two methods, parent selection and survival. Parent selection chooses who emerges a parent and how many offspring the parents get. Children are formed via reunion, which transforms messages among parents, and mutation, which more agitates the children. The children are then assessed. At last, the survival process chooses who sustains in the denizen (Spears et al 1993).

3.3.1 The Darwinian Root of Evolutionary Algorithms

Evolutionary algorithms are intent of mechanisms activated by the natural evolution strategy, more particularly, by Darwin's theory of evolution by natural selection. In “Origin of Species”, Darwin formed the below perceptions on the natural evolution strategy:

- A large number of species of creatures are prevailing on the earth. Each species has an excessive count of individuals –the denizen.
- Substance in a provided environment is finite and so only a finite number of creatures can be sheltered, prominent to aggressiveness for durability—the selection strategy.
- Surviving creatures expand by asexual or sexual replication, over which aimless mutations always appear, and maximum of the attributes of the parent(s) are rooted – bequest with alteration.

Natural selection as a common basis concealed evolution has been fine invented. The basic algorithmic aspect of the natural evolutionary strategy has activated several of the founders of evolutionary computing to arrange their unreal evolutionary algorithms, counting, but not finite to, the broadly accepted genetic algorithms, evolution process, evolutionary programming, and genetic programming. In short, all the EAs can be outlined with the algorithm displayed in Figure 3.2. Although the general source of Darwinian evolution theory, anciently, current evolutionary algorithms appear with utterly various analysis histories of their founders. These variances lead to their particular attributes and their desires respecting the delegation, variation executors, and selection/replication executors. It is fair that each of them gives some peerless lineaments that may devote to the growth of

continual evolutionary algorithms. Evolutionary algorithms are debatable search systems that copycat the analogy of natural biological growth. Evolutionary algorithms function on a denizen of capable solutions using the code of survival of the fittest to give greater and greater proximities to a solution. At all generation, a new firm of proximities is formed by the method of choosing individuals respect to their range of fitness in the difficult domain and culturing them unruffled using executors rented from regular genetics. This method advances to the evolution of denizens of individuals that are greater fitted to their ambiance than the individuals that they were formed from, just as in regular adoption Hu (2004).

Evolutionary algorithms model regular methods, such as selection, reunion, mutation, migration, environment and region. Figure 3.3 and Figure 3.4 displays the difficult solution applying evolutionary algorithms and the design of an easy evolutionary algorithm subsequently. Evolutionary algorithms act on denizen of individuals rather of specific solutions. In this method the research is achieved in a lateral aspect.

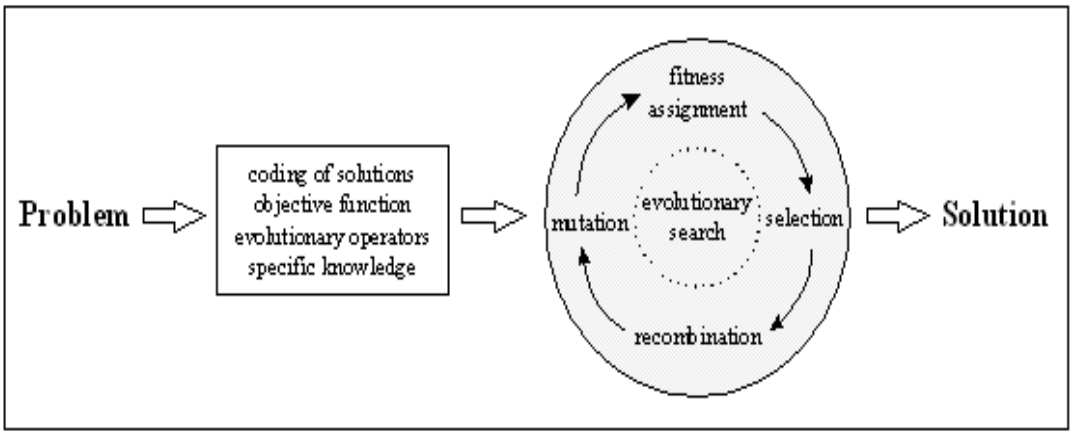


Figure 3.3 Problem solution usingevolutionary algorithms

At the onset of the computing a statistic of individuals (the denizen) are nevertheless computed. The unbiased operation is then measured for these individuals. The first/starting generation is formed. If the development principles are not face the formation of a new generation initiates. Individuals are picked depending on their fitness for the creation of children or offspring. Parents are reunited to give offspring. All offspring will be altered with a specific prospect.

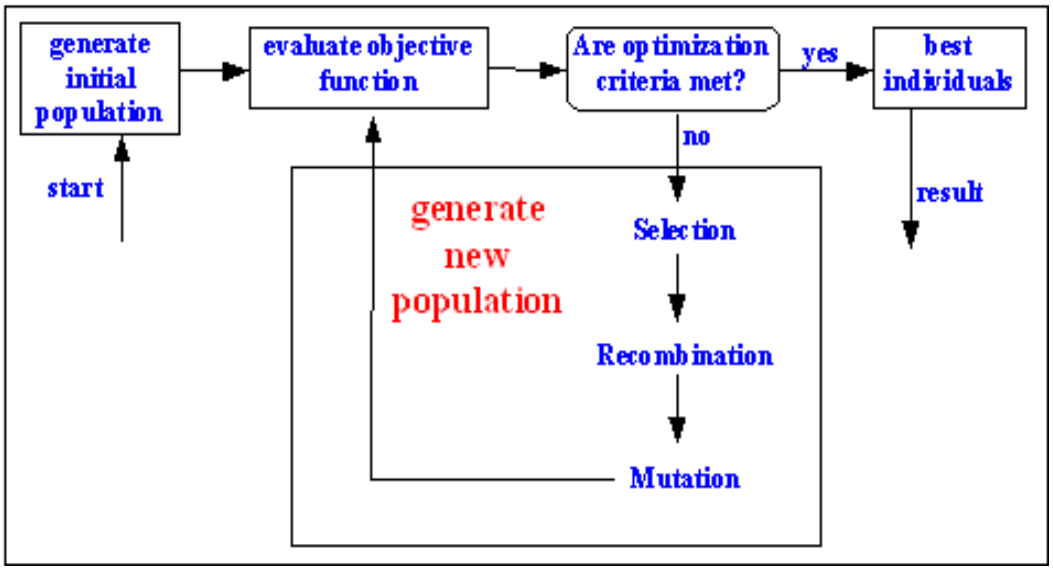


Figure 3.4 Structure of a single population evolutionary algorithm

The robustness of the offspring is then measured. The offspring are added into the population restoring the parents, forming a new genesis. This process is executed until the developmental norms are attained. Such an individual denizen evolutionary algorithm is strong and achieves fine on a broad collection of troubles. Nonetheless, greater results can be acquired by inventing numerous sub denizens. Every sub denizen emerges over some generations detached (alike the single denizen evolutionary algorithm) before one or more personals are transformed among the sub denizen. The multi-denizen evolutionary algorithm models the emerging of breeds in a form more

identical to nature than the individual denizen evolutionary algorithm. Figure 3.5 displays the design of such a prolonged multi-denizen evolutionary algorithm Jones (1998).

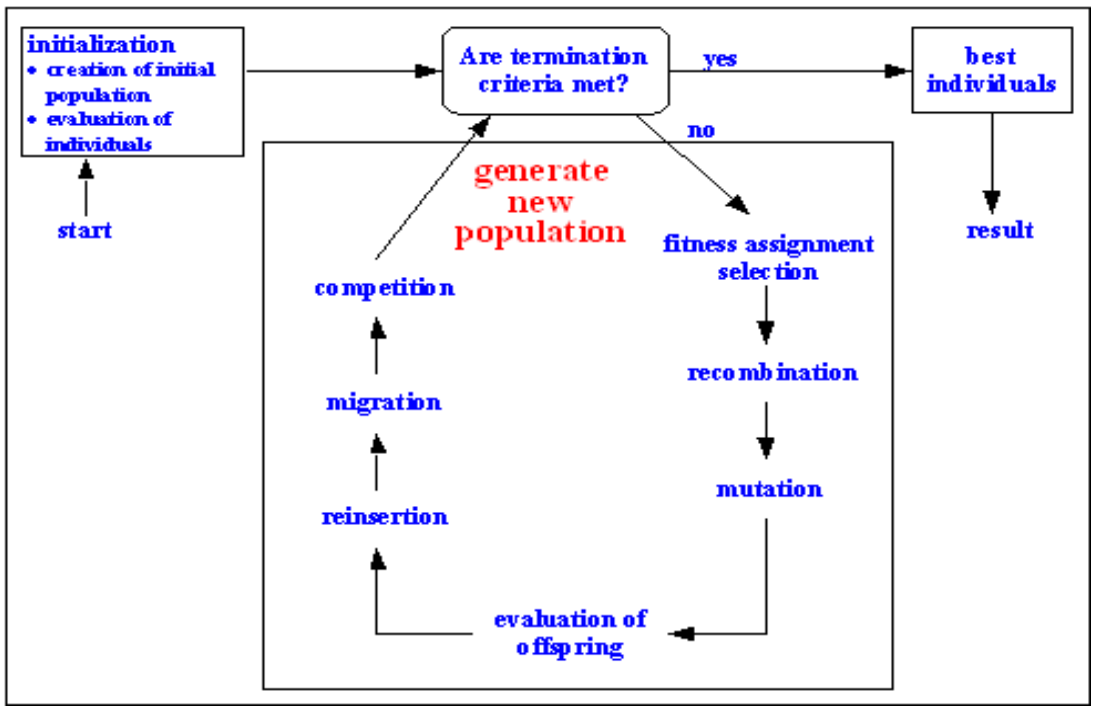


Figure 3.5 Structure of an extended multi population evolutionaryalgorithm

From the over analysis, it can be detect that evolutionary algorithms vary extensively from more universal analysis and developmental process. The most important variances are:

- Evolutionary algorithms analyze a denizen of counts in lateral, not just an individual point.
- Evolutionary algorithms do not need acquired message or other ancillary awareness; only the detached operation and analogous fitness ranges impact the indications of analysis.
- Evolutionary algorithms use feasibility progress rules, not

arbitrating ones.

- Evolutionary algorithms are commonly more unequivocal to use, as no limitations for the description of the equitable operation occur.
- Evolutionary algorithms can give a sum of capable solutions to a produced issue. The end option is given up to the user. (Hence, in terms where the specific issue does not have one single solution, for instance a family of pareto-ideal solutions, as in the event of multi-detached development and programming issues, then the evolutionary algorithm is efficiently supportive for recognizing these substitute solutions concurrently Dianati et al (2002). Figure 3.6 displays the flow chart illustration of an Evolutionary Algorithm:

3.4 TYPES OF EVOLUTIONARY ALGORITHMS

The influence of evolutionary algorithms can be chased to slightly the 1950's. For the welfare of economy we will not focus on this previous work but will review in some elaborate three techniques that have appeared in the recent few decapods: "evolutionary programming", "evolution strategies" Rechenberg (1973) & "genetic algorithms" Holland (1975).

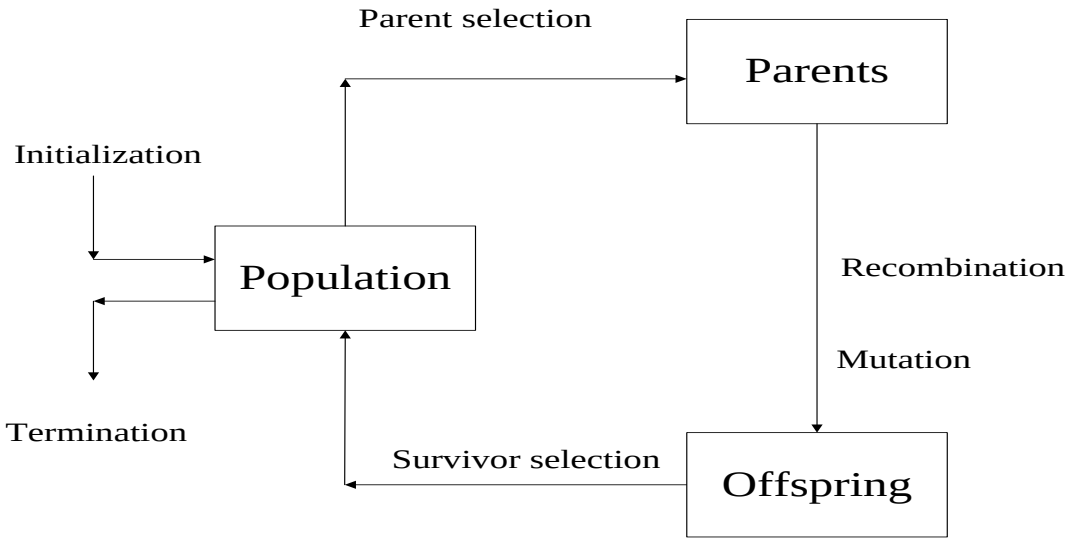


Figure 3.6 The general scheme of an evolutionary algorithm as a flow chart

Despite identical at the maximum level, each of these variations implements an evolutionary algorithm in a various form. The divergences impact above almost all forms of evolutionary algorithms, counting the options of illustrations for the individual designs, kinds of selection method applied aspects of genetic executors, and degrees of achievement. Let us high spot the significant variances (and affinities), by considering some of the collection defined by the existent family of evolutionary algorithms.

These methods in turn have influenced the optimization of further evolutionary algorithms such as "classifier methods" , the LS methods Smith (1983), "adaptive operator" methods Davis (1989) Genitor Whitley & Starkweather (1989), Samuel Grefenstette (1989), "enetic programming" Garis (1990) & Koza (1991), "messy GAs" Goldberg & Deb et al (1991) and the CHC method Eshelman (1991). Let us not trial to analyse this wide range of methods here.

3.5 EVOLUTIONARY PROGRAMMING

Evolutionary programming (EP), established by basically has applied delegations that are bespoke to the issue domain. For instance, in true valuable optimization issues, the individuals inner the denizen are true-valued agents. Also, organized lists are applied for moving marketers issues, and graphs for operations with limited state devices. EP is always applied as a developer; despite it emerges from the motive to produce machine perception. After computing, all N individuals are picked to be parents, and then are modified, giving N offspring. These children are measured and N legacies are picked from the $2N$ individuals, applying a probabilistic operation positioned on fitness. In other aspect, individuals with a better fitness have a greater possibility of survival. The method of mutation is positioned on the delegation applied, and is always flexible. For instance, when applying a true-valued agent, all variable inside an individual may have a flexible mutant rate that is regularly scattered with a zero assumption. Reunion is not commonly operated as the methods of mutation applied are really adaptive and can form disruptions related to reunion, if aimed. One of the pleasing and transparent problems is the intensity to which an EA is damaged by its option of the executors applied to form instability and creation in emerging denizens Chipperfield & Flemming (2002).

3.6 GENERIC SYSTEM IN GENETIC ALGORITHMS

Genetic algorithms comprises of Chromosomal delegation, first population, fitness assessment, selection and replication (i.e.,crossover and mutant). Lucid summary of a genetic algorithm is displayed in Figure 3.7.

3.6.1 Initial Population

This is formed from an aimless or random selection of results (which are comparable to chromosomes). This is contrary in an emblematic unreal brilliance where the first state in an issue is formed rather.

3.6.2 Fitness Evaluation

A idolize of fitness is designated to every solution (chromosome) based on how proximate it really is to determining the difficult, thus emerging to the response of the structured issue. The solutions are speculated of as probable attributes that the method would apply in order to attain the result.

3.6.3 Selection

The natural selection is defined as transmission survival of individuals where few entities alive and others demise. There are several various methods, which a genetic algorithm can apply to choose the entities to be duplicated upon into the next genesis. Some of the techniques mentioned below are commonly unique, yet others can be and always applied in sequence Adedeji (2007).

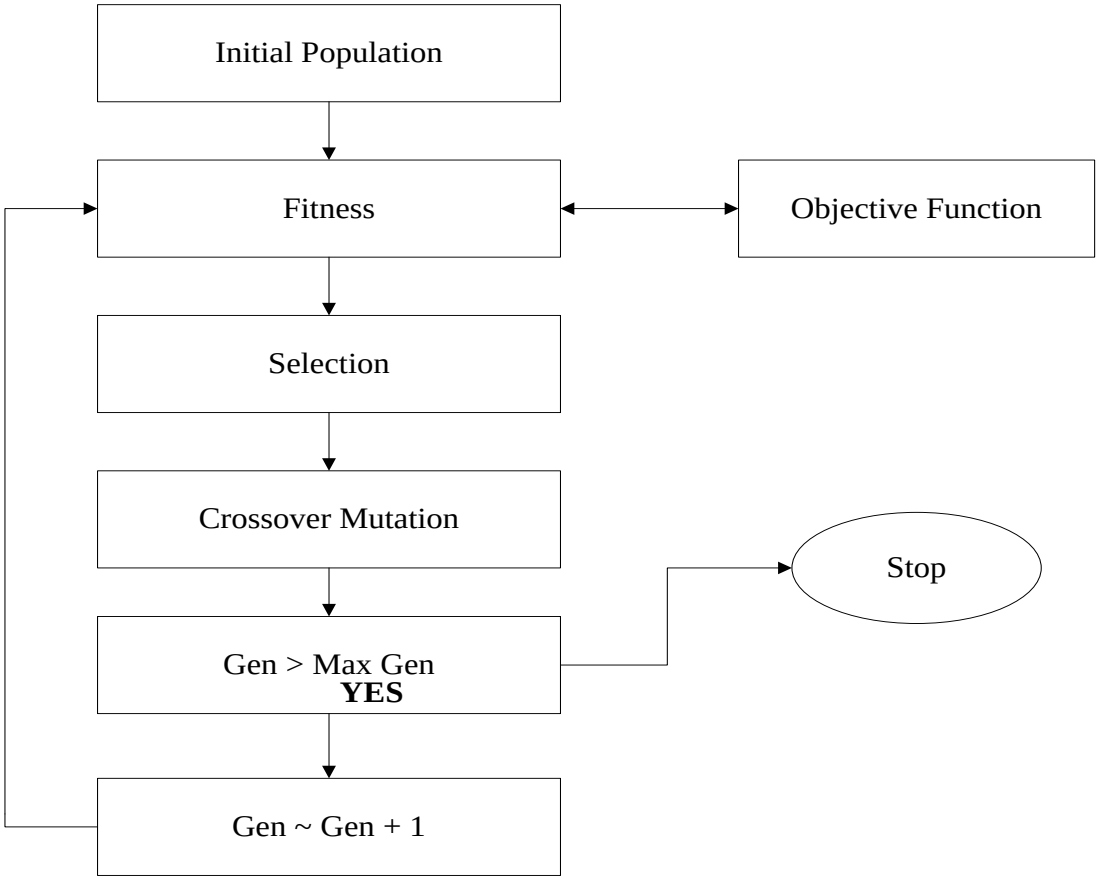


Figure 3.7 Outline of genetic algorithm

3.6.4 Reproduction

Natural selection refers transmission survival of individuals where few entities alive and others demise. For this to appear there must be a denizen of individual potential of replication.

- Crossover: Crossover is a GA executor that requires selecting two entities to change divisions of their code, whereby forming new progeny that are sequence of their parents. This method is expected to replicate the related process of reunion that appears to chromosomes at the time of sexual reproduction. In other aspect crossover sources genotypes (firm of genes that forms a

life style or phenotype) to be cleft and joined.

- Mutation: This executor produces a new chromosome by creating (typically small) mutations to the codes of genes in a carbon of an individual parent chromosome, just as alterations in living creatures varies one gene to another, so also in a genetic algorithm.

Crossover and mutation are two fundamental executors of GA. Execution of GA based on them mere adequate. The kinds and application of these executors based on the concealing and also on the issue Adedeji (2007).

3.7 SOME SPECIFIC APPLICATIONS OF GA IN ENGINEERING TECHNOLOGY

3.7.1 Design of Hardware

In a huge of hardware machines, rules of evolution has been applied to form an archetype voice-identification circuit that can categorize among and answer to voiced instructions using only 37 logic gates- a target that would have been studied impractical for any human designer. Random bit strands have been produced and used as outlines for the field programmable gate array (FPGA), choosing the fittest entities from every generation, replicating and randomly mutating them, changing divisions of their code and fleeting them on to another cycle of election.

3.7.2 Acoustic

GA is applied to form a recital hall with excellent acoustic characters increasing the sound nature for the crowd Sato et al (2002).

GA was applied to course neural systems to separate among sonar ruminations from various kinds of the substance: human-created metal spheres, plant-life, etc. Porto Porto et al (Porto et al (1995).

Tang et al (1996) analysed the benefits of GAs inside the area of acoustics and signal refining, such as the structure of effective noise control method.

3.7.3 Aerospace

Different objective GA was applied to form the wing frame for a rapid aircraft, by reducing simplified drag at rapid navigating speeds, and modernized load (the curving efforts on the wing) Obayashi et al (2000). Chromosome in this aspect is a strand of 66 units, each of which resembles to a particular character of the wing for its frame, its density, its bend, etc.

Reducing the curving point (extra fitness detached) of the aircraft wing – A familiar capable issue for arrow-wing formed. Here, further restraint points Sasaki et al (2001) for density are joined to the array of frame volatiles. The solutions, when related with the Japanese National Aerospace Laboratory's wing structure for NEXST-1 empirical rapid airplane, were formed to be physically logical and greater to the NAL's structure.

Satellites in huge Earth orbit, around 35,420 km up, can view high divisions of the planet promptly and be in stable with earth stations, but they are distant more costly to project and more sensitive to cosmic emission. It is hence more cost-effective to insert satellites in small orbits. But as of the arc frame of the Earth, it is imminent that satellites will occasionally drop line-of-vision approach to surface recipients and hence be ineffective. Also intention of many satellites practice necessary knockouts and drops of description for

this logic. This is a multi-detached issue where there is the demand to organize the satellites orbits to reduce this intermission. GA was then used to this issue, the emerged solutions for the intentions were incredible, largely unbalanced orbit structures, with the satellites dispersed by varying huge and little spaces instead than levelled spaces as accepted method would form Adedeji (2007).

3.7.4 Astronomy and Astrophysics

GA (PIKAIA) has been applied genesis, fitness-equitable grading selection and superiority to assuring that a one best entity is duplicated upon already into the next genesis without alteration, with a cross figure of 0.65 and a fickle mutation figure of 0.003 firstly and steadily expanded later on. In the cosmic rotation-twist issue, the GA formed two bends, both of which were very well apt to the data. In resolving for the six analytical criterions of the solar wind, the GA profitably regulate the grade of three of them to an efficiency of within 0.1% and the halting three to veracities of among 1 to 10% Coello (2000).

3.8 DIFFERENT METHODS AND ALGORITHMS USED IN THIS STUDY:

This study uses evolutionary computing and its algorithms along with the computational intelligence in order to design the proposed method. Here, efficient automated solution environment was developed for designing efficient systolic architecture which uses genetic algorithm, evolutionary programming, swarm intelligence and chaotic optimization. This study uses evolutionary computing in order to build highly adaptable, potent, and dynamic algorithms. Apart from these, genetic algorithm is also used in this study. This study uses Computational Intelligence as one of the important

concepts in the proposed technique. In this research, multiple automatic design vector generation for efficient systolic architecture is developed by using NIC and systolic architecture is developed by using the hybridization of GA and local chaotic optimization.

3.9 TRADE-OFF METHODS IN THIS STUDY:

This study considers evolutionary computing as most important and this is because it helps to find out the structure of computational methods and also to build highly adaptable, potent, and dynamic algorithms which are basically difficult for the traditional computational systems. Evolutionary computation uses computing patterns of evolutionary strategies in the structure and application of computer-oriented issue solving methods. Thus, it will help to design the proposed system. First, multiple automatic design vector generation for efficient systolic architecture is developed by using NIC. Secondly, systolic architecture is developed by using the hybridization of GA and local chaotic optimization. The evolutionary computing and computational intelligence have ability to support for designing the proposed solutions.

3.10 SUMMARY

Detail discussion has presented to understand the evolutionary computation from level of requirement to stage of implementation design. Various part of evolutionary computation has explored to understand the benefit and inherent power in their solution development. Even though there are some fundamental similarities exist in all different method, their functional differences were also part of discussion .This helps to develop some new and unique operators in their function domain to further enhance

the solution quality. Number of evolutionary operators have discussed in details which help in how to design better evolutionary algorithm for a particular application. Several applications ,where in past evolutionary computation have been applied successfully , also discussed in details .With this deep understanding ,it is possible to integrate the computational intelligence method, which based on evolutionary computation, to design of systolic architecture so that performance can be improved.

CHAPTER 4

FLOW FOR DESIGNING SYSTOLIC ARRAYARCHTECTURE

4.1 INTRODUCTION

Reconfigurable programming is used on millions of logic gates. Appropriate methodology needs to be evolved to use the large scale programmable devices in most efficient way. In this chapter a methodology that is used to design efficient systolic array for iterative algorithms is discussed.

The increase in large number of programmable logic per FPGA increases the need of parallel hardware architecture like Systolic array. Most iterative algorithms need efficient memory access to utilize parallel processing environment. The flow is for mapping nested loops seen in numerical algorithms to efficiently do parallel processing. This flow exploits inherent parallelism and pipelining. Data dependency and the importance of projection vector, processor vector and scheduling vector are used in designing the flow.

Nested loop algorithms are seen in numerous intensive computing algorithms. Designs using systolic array efficiently exploits parallelism in the nested loop algorithm and thus reduces the processing time.

This chapter gives methodology for mapping iterative algorithms and designing a systolic array by using three vectors projection vector, processor vector and scheduling vectors.

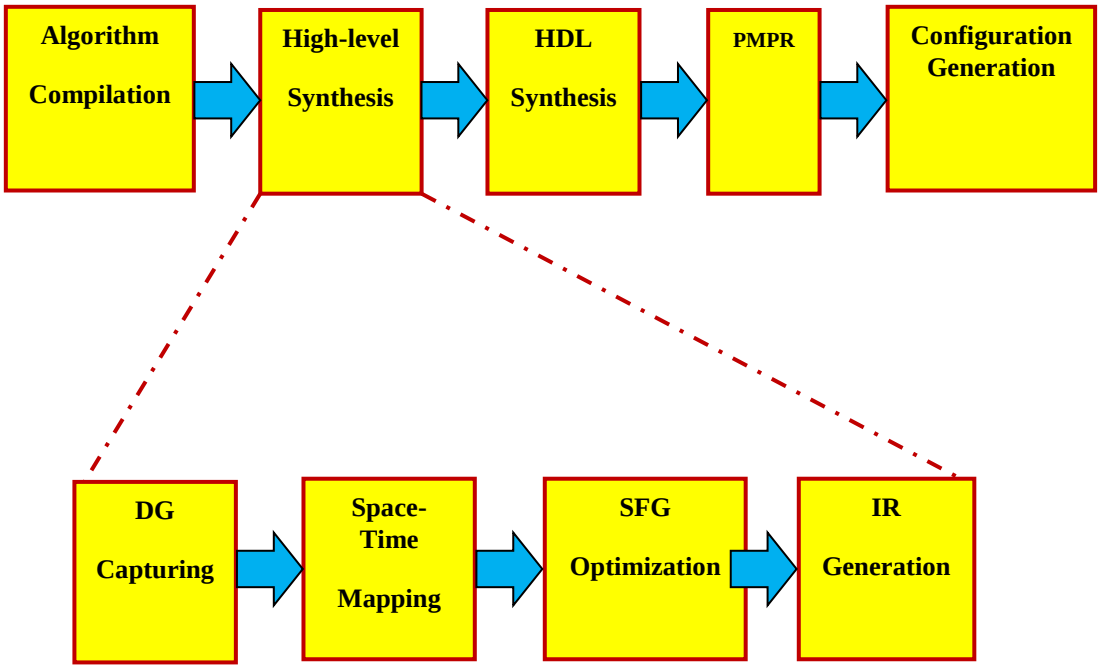


Figure4.1 Design flow algorithm to hardware mapping

- Algorithm Compilation
  - Converts algorithm to graph-based representation
- High-Level Synthesis
  - Converts graph to data path (space) and control(time) to HDL
- HDL (Hardware Description Language) Synthesis
  - Convert behavioral or RTL HDL(VHDL, Verilog) to netlist of gates and flip/flops
- Partitioning, Mapping, Placing, Routing
  - A netlist to a given architecture
  - Partitioning is required if the design doesn't fit.

- Configuration Generation
  - Equivalent to code generation of compiler
- Dependence Graph Capturing
  - *n*-D Dependence Graph (DG) is obtained from a compiler or user.
- Space-Time Mapping
  - -*n*-D is mapped to *k*-D signal flow graph (SFG) and (*n*-*k*)-D time schedule, 1<= *k*<*n.*
- SFG Optimization
  - -*k*-D SFG is optimized mostly by human designer.
- IR Generation
  - -Intermediate representation (IR) e.g. VHDL, Verilog HDL

4.2 HIGH LEVEL SYSTHESIS

The algorithm to be produced by the synthesis of high-level system should be defined in a way. The typical way of achieving this is in form of text is by a way of a language formally. This can be in a language of programming conventionally or a depiction language of hardware. Investigators in the arena have diverse opinions about the language widely apt for synthesis at high-levels. VHDL, silage and Verilog are instances of such languages, just to point few popular ones. It is though, outside the range of this text to pay consideration to this point. Whatsoever, language is utilized, the textual form is not suitable for the depiction of the algorithms during the course of synthesis. Obviously, text is merely an extended string of symbols and the string depiction does not encompass every explicit structure. One will particularly be concerned in the depiction of the parallelism existing in the algorithm. It is thus essential to parse the text and alter it into a structured

internal depiction. It is widely in general agreed that like a depiction must be graph based. The graph that is utilized to depict an algorithm is termed a data-flow graph (DFG).

4.3 DEPENDENCE GRAPH

A dependence graph (DG) is a graph of directed nature that portrays the computations' dependence in an algorithm. The nodes in a DG depict computations (intended as nodes) and the edges depict precedence restraints between nodes. In a DG, a fresh node is formed whenever a fresh computation is called for in the algorithm (program) and no node in DG is ever reutilized on a lone basis of computation. DG depiction is similar to DFG depiction as it obviously displays the requirement of node on different nodes in the graph. The variance is which the nodes in DFG only include the computations in one iteration of the conforming algorithm and these are implemented continually from iteration to iteration, while DG includes all iterations' computations in an algorithm. DFGs comprise of delay features which are stored and pass data from present iteration to succeeding iteration, though DG does not have elements of delay.

Dependence graphs (DG) are utilized for designing systolic arrays; here numerous implementations (DFGs or block diagrams) could be attained from a lone DG by taking advantage of the parallelism offered in DG in diverse ways.

To attain the parallelism maximally in an algorithm, we should carefully analyse the dependencies of data in the computations. In the exceptional case here,the sequential algorithm operation has no dependencies of data amidst each of them; these could be implemented at similar times in a parallel computer.

4.4 DATA-FLOW GRAPH (DFG)

Before explaining the characteristics of a DFG, it is essential to mention which numerous definitions of a graph occur. Though, all of them have approximate amount, which are similar. The diverse definitions may or may not be designated by a precise name.

The distinction made in the hardware amidst data path and controlhas made few investigators describe two graphs: control-flow and data-flow graphs. Though, the term “data-flow graph” is also generally utilized when information on control is incorporated in the graph. The latter will be the case in this text.

Dataflow graphs are meticulously related to signal-flow graphs that have been traditionally utilized in the domain of digital signal processing (DSP). The special need of DSP, like the repetitive application of the similar algorithm to data incoming at fixed intervals, has led to the development of particular DFGs for algorithms synthesis of DSP, key classes of DSP algorithms, described by the lack of computations which are controlled by data-dependent circumstances (like while and if-then-else constructs).

In data flow graph (DFG) depiction, the node represents computations (or functions or subtasks) and the directed edges depict data paths (communication amidst nodes) and every edge should be a nonnegative number of delays linked with it. For instance, figure 1 is a data-flow graph of the computation y(n)=ay(n-1)+x(n), here node A depicts addition and node B depicts multiplication, the edge from A to B (represented as A→B)includes one delay and the edge from B to A (B→A) has no delay linked with every node that is its implementation time in the way of normalized time units (u.t.; units of time)..

The data-flow graph internments the data-driven characteristics of DSP algorithms. Here every node could fire (achieve its computation) every time all the data inputs are obtainable. This implies, when a node with no input edges can fire time; and so many nodes could be fired concurrently, leading to concurrency. On the other hand, a node with numerous input edges could only fire after all its nodes in precedence have fired. The latter scenario enforces the precedence restraints on a DFG, here every edge defines precedence restraints amidst two nodes. This precedence restraint is an intra-iteration precedence restraints if the edge has zero delays or inter-iteration precedence restraint if the edge should have one or more delays. Organized, the inter-iteration and intra-iteration precedence restraints stipulate the order in that the nodes in the DFG could be implemented.

Numerous DFGs deduced for one algorithm could be gained from every different through transformations of high-level. DFGs are usually utilized for high-level synthesis to arrive at concurrent execution of DSP applications onto parallel hardware, here subtask allocation of resource and scheduling.The node in a DFG could be as modest as operations that are elementary and indivisible like as operations of addition or basic logic. These DFGs are termed as atomic. When the granularity is at the level of processing if signal subtasks like filtering, the DFG is data-flow graph of coarse-grain kind.

4.5 SYSTOLIC ARRAY DESIGN METHODOLOGY

Systolic architecture is designed by using linear mapping techniques on regular dependency graph. The edge in dependency graphs represents precedence constraints. A dependency graph (DG) is said to be regular if the existence of edge in a definite direction at any node in the DG represents presence of edge in the same direction at all nodes in the DG.

4.5.1 Definitions

- Projection vector (also called iteration vector)$\boldsymbol{d} = \begin{pmatrix} \boldsymbol{d_1} \\ \boldsymbol{d_2} \end{pmatrix}$
  - Two nodes that are displaced by d or multiples of d are performed by the same processor.
- Processor space vector, $p^T = (p1\ p2)$ Any node with index I^T=(i, j) would be performed by processor; $p^T I = (p_1 p_2)\begin{pmatrix} \boldsymbol{i} \\ \boldsymbol{j} \end{pmatrix}$
- Scheduling vector, s^T = (s1 s2). Any node with index I would be executed at time, $s^T I$.
- Hardware Utilization Efficiency, HUE =$\frac{\mathbf{1}}{|\boldsymbol{S^T d}|}$. This is owing to two tasks implemented by the similar processor that are spaced $|S^T d|$ time units apart.
- Many systolic architecture can be designed for a given problem by selecting different projection, processor space and scheduling vectors. These vectors must satisfy the feasibility constraints derived below.
- Processor space vector and projection vector must be orthogonal to each other. If points A and B differ by the projection vector, i.e., I_A-I_B is same as d, then must be executed by the same processor, in other words, $P^T I_A = P^T I_B$. This leads to $P^T (I_A - I_B) = 0 \Rightarrow p^T d = 0$.
- If A and B are mapped to the same processor, then they cannot be executed at the same time, i.e., $S^T I_A \neq S^T I_B$, i.e., $S^T d \neq 0$.
- Edge mapping: If an edge e exists in the space representation or DG, then an edge $p^T e$ is introduced in the systolic array with $s^T e$ delays.

4.5.2 Selection of Scheduling Vector

For any specified projection vector, processor space vector and scheduling vector, the systolic array can be designed using linear mapping technique. The method of selecting feasible scheduling vectors using scheduling inequalities is described. Based on the selected scheduling vector S^T, the projection vector and the processor space vector P^T can be selected accordingly to equations $p^T d = 0$ and $S^T d \neq 0$. . Hence the desired systolic array can be obtained.

4.6 SELECTION OF ST BASED ON SCHEDULING INEQUALITITIS

Selection of sT based on scheduling inequalities:

For a dependence relation X →Y,

$$X: I_x = \begin{pmatrix} i_x \\ j_x \end{pmatrix} \rightarrow Y: I_y = \begin{pmatrix} i_y \\ j_y \end{pmatrix}$$

Where I_x and I_y are the indices of node X and node Y, respectively. The scheduling inequality for this dependence is given by,

$$S_y \geq S_x + T_x$$

Where T_x is the computation time of node X. The scheduling equations can be classified into the following two types:

- Linear scheduling,

  Where

  $$S_x = s^T I_x = (s_1\ s_2)\begin{pmatrix} i_x \\ j_x \end{pmatrix}$$

$$S_y = s^T I_y = (s1\ s2)\begin{pmatrix} i_y \\ j_y \end{pmatrix}$$

- Affine Scheduling,

  Where

$$S_x = s^T I_x + \gamma_x = (s1\ s2)\begin{pmatrix} i_x \\ j_x \end{pmatrix} + \gamma_x$$

$$Sx = s^T I_x + \gamma_y = (s1\ s2)\begin{pmatrix} i_y \\ j_y \end{pmatrix} + \gamma_y$$

Using the forgoing definition, we can rewrite the scheduling equation for affine scheduling as $s^T I_x + \gamma y \geq s^T I_x + \gamma_x + T_x$

Note that the scheduling equation for linear scheduling can be obtained by setting γ_x and γ_y equal to zero. Define the edge from node X to node Y as $e_{x-y} = I_y - I_x$. Then the scheduling inequality for an edge is described as follows.

$$S^T e_{x-y} + \gamma_y - \gamma_x \geq T_x.$$

Therefore, one scheduling inequalities can be obtained for each fundamental edge in the dependency graph and the scheduling vector S^T can be obtained by solving these inequalities. Hence the selection of scheduling vector consists of 2 steps.

- Capture all the fundamental edges. The reduced dependence graph (RDG) is used to capture the fundamental edges and the regular iteration algorithm (RIA) description of the corresponding problem is used to construct RDGs.
- Construct the scheduling inequalities according to $S^T e_{x-y} + \gamma_y - \gamma_x \geq T_x$. and solve them for feasible S^T.

4.7 RIA DESCRIPTION

The regular iterative algorithm (RIA) is introduced and the method for construction reduced dependency graphs (RDGs) using RIA is illustrated.

- The RIA is in standard input RIA form if the indexes of the inputs are the same for all equations.
- The RIA is in standard output RIA form if all output indices, i.e. indices on the left side, are the same.

4.8 MOTIVATION

Systolic array has inherent pipelining and parallelism which can be used to achieve throughput rates for high computation on FPGA. VLSI fabrication technology inspired significant architectural development for systolic array.

By pumping data through a regular network of hundreds or thousands of simple processing elements, computationally intensive problems can be solved many times faster on systolic arrays than on traditional machines.

To develop techniques for analysing algorithms, dependency analysis and mapping to enormous parallel array structures is the great challenge. The repeated sets of patters of operations are seen in computationally intensive algorithms like deeply iterative loop algorithms. They are good candidates for massive parallel implementation Lee & Kedem (1988) & Kung (1985) & Liang & Jean (2003).

In the process of evolving the designs criteria is to maximize throughput by using maximum number of input ports with maximum data reuse.

This chapter gives methodology for mapping iterative algorithms and designing a systolic array.

4.9 FLOW FOR DESIGNING SYSTOLIC ARRAY

Designing a systolic array by using three vectors projection vector, processor vector and scheduling vectors is shown in Figure4.1. Dependence graph (DG) is drawn for regular iterative algorithm and it is used for designing systolic array architecture, where various implementations can be derived from a single DG by exploiting the parallelism presented in DG in various ways Parhi (2007).

Three vectors namely projection vector, processor vector and scheduled vector should be selected to meet the constraints and to achieve better HUE. Selection of these three vectors plays a very important role in designing efficient systolic architecture and currently it is a manual process and time consuming one. Our work uses evolutionary computation based methodology for selecting projection vector, processor vector and scheduled vector to meet the constraints with better HUE.

Constraints in synthesis refer to I/O constraints and implementation constraints. Interface constraints are to make sure that the circuit can be embedded in a given environment. They describe format and timing of the I/O data transfers. The timing constraints on I/Os is to ensure proper I/O operation toprecedes/follows another one in given number of cycles.Constraints for architecture design are it must be highly regular for

nested loop algorithms; the basic operation requirement is communications must be of near-neighbour type.

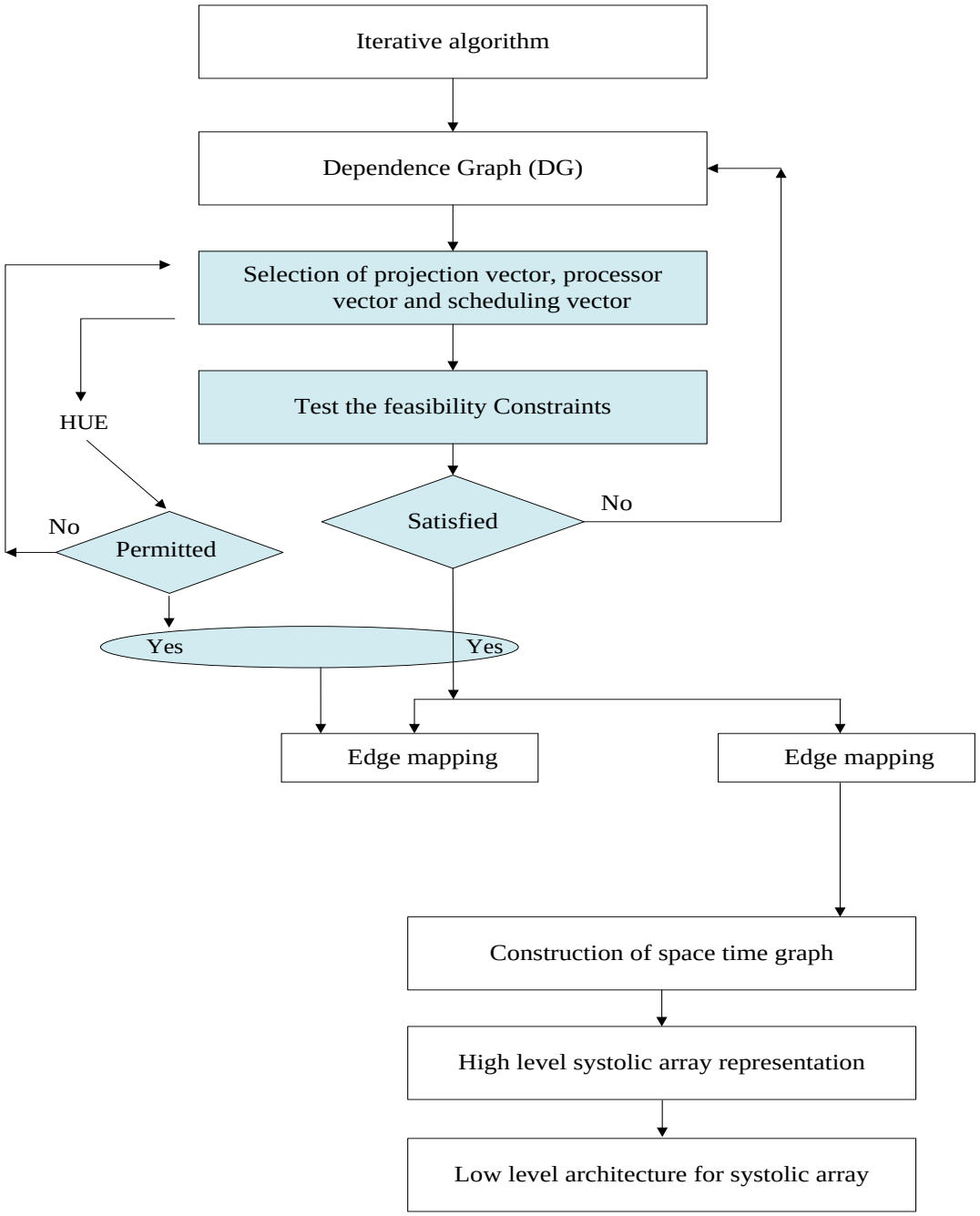


Figure 4.2 Designflow for SA

Testing the feasibility constraints represents validation of constraints with the fundamental vectors to generate the feasible solution. Selection of projection vector, processor vector and scheduled vector are carried out until constraints are met and HUE is with-in permitted one. Flow for testing the feasibility of meeting constraints are shown in Figure.4.2.Flow represents validation of constraintsavailable with fundamental vectors to generate feasible solutions.

In Figure.4.3‘d’is the Projection vector, $p^T$is the Processor Vector ans^{T}is the Scheduling vector. Selected three vectors are valid when $p^Td = 0$ *and* $S^Td \neq 0$.Otherwise it is not possible to develop systolic array from selected three vectors.

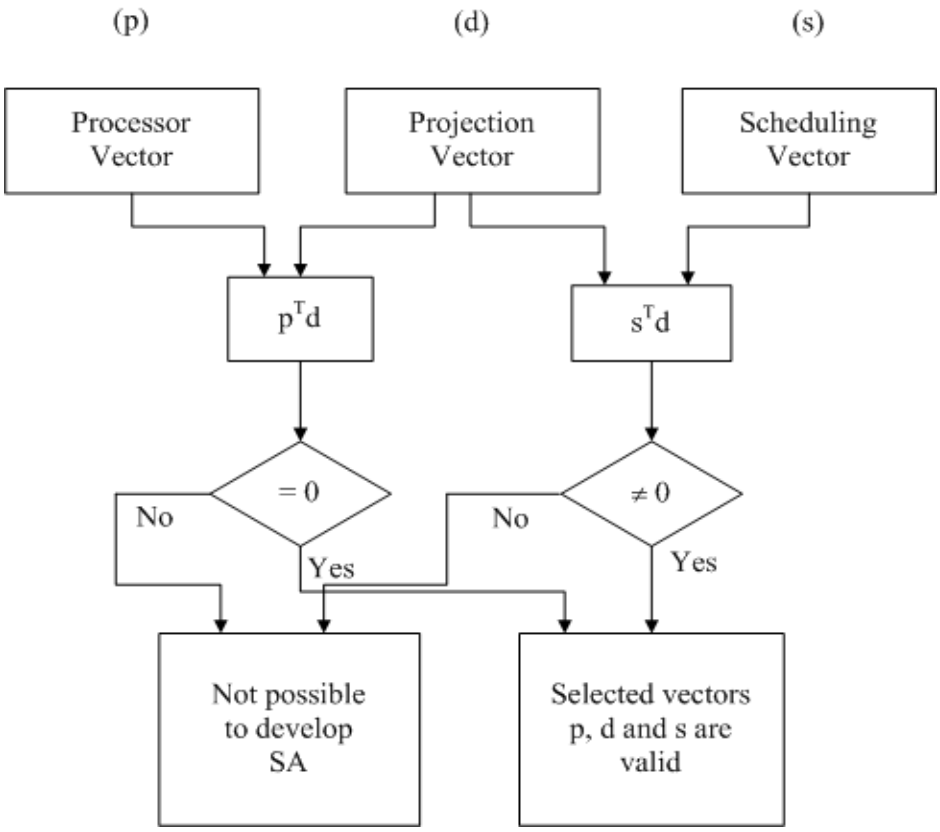


Figure 4.3 Flowchart for testing thefeasibility constraints

4.9.1 Hardware Utilization Efficiency

The flowchart for HUE is shown in Figure 4.4. It is given by: HUE = $1/|s^T d|$

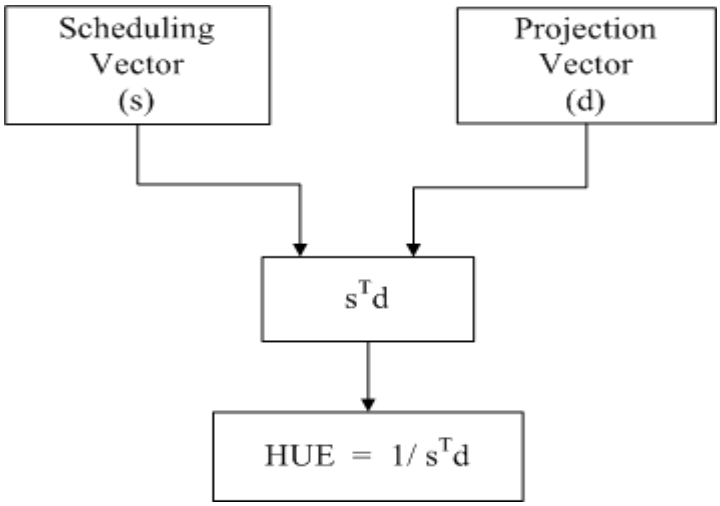


Figure 4.4 Flowchart for HUE

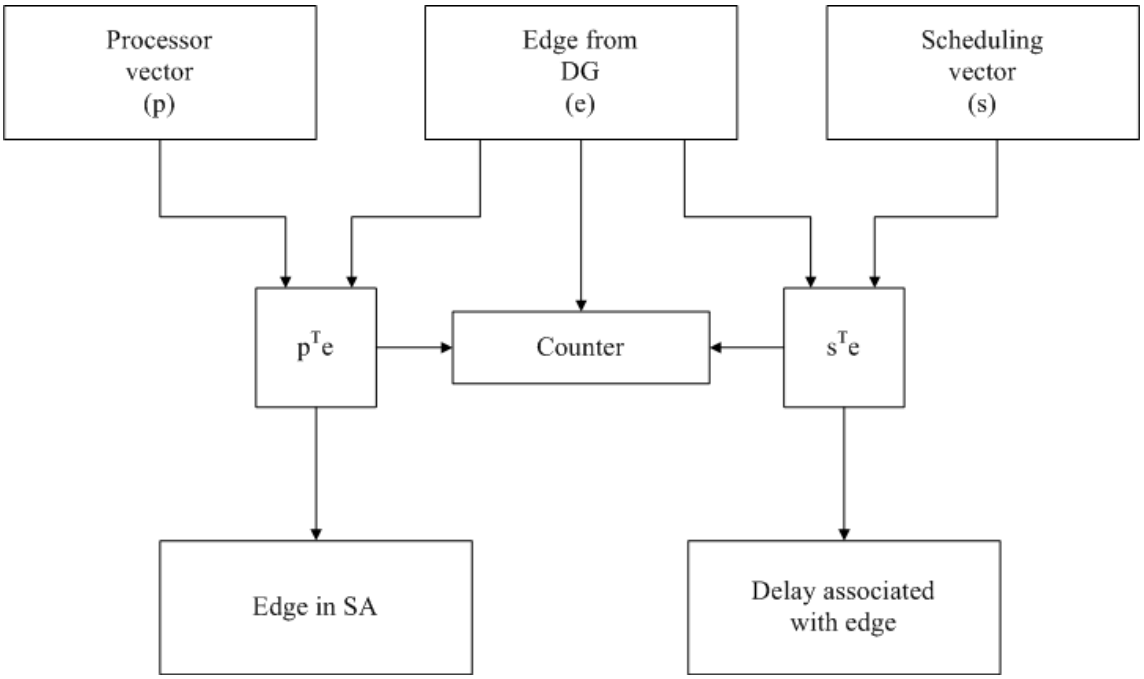


Figure 4.5 Methodology for edge mapping

4.9.2 Edge Mapping

Edges related to each node are determined from the DG and then the processor associated to that node in the systolic array is determined by calculating $p^T e$ and the delay associated with each node is then calculated by $s^T e$. Methodology for edge mapping is shown in Figure 4.5

4.10 DESIGN MTYHODOLOGY FOR SELECTIONG SCHEDULING VECTOR FROM RIA DESCRIPTION

When the projection vector, processor vector and scheduling vectors are not known to prior, then using DG it is possible to find these vectors by the following design methodology.Design methodology for selecting schedule vector using scheduling inequalities is shown in Figure 4.7

DG represents all individual possible operations in the algorithm or process. RegularIteration Algorithm (RIA) description is to capture the fundamental edge from the DG. Reduced Dependence Graph (RDG) gives the compact DG.Find the fundamental edges over which systolic architecture has to be designed. There are number of inequalities equations that will appear which is function of fundamental edges and their dependency along with computational time involvement. Inequalities are automatically solved.

4.10.1 Node Mapping

It is the technique to map each node from DG into space time representation graph. Methodology for node mapping is shown in Figure 4.6.

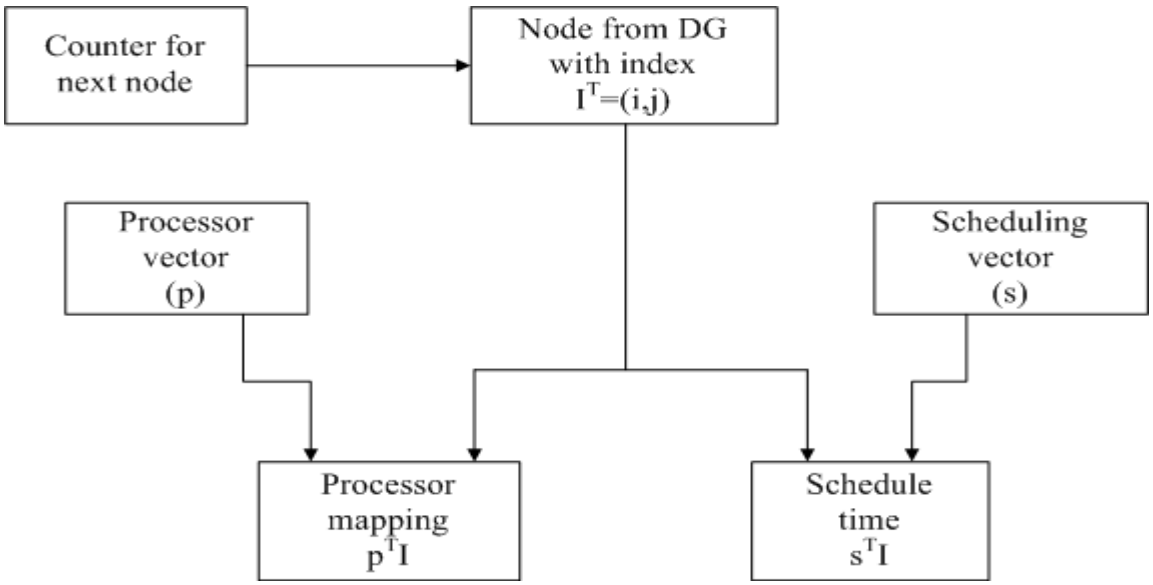


Figure 4.6 Methodologyfor node mapping

4.11 MOTIVATION FOR USING EVOLUTIONARY PROGRAMMING

To design different types of systolic architecture for an algorithum multiplie projection,processor & scheduling vectors are need. There is no unique solution which gives a chance to find multiplie these vectors for designing different types of systolic architecture with high HUE. To remove trial & error approach for finding SDP vectors it is necessary to automate the process which provides number of possible vectors and to design possible architectures with different characterstics.

Among the huge possibility of generating design vectors with the use of evolutionary concept.our methods gives the trials to find out number of efficient design vectors with hign HUE using evolutionary computaion

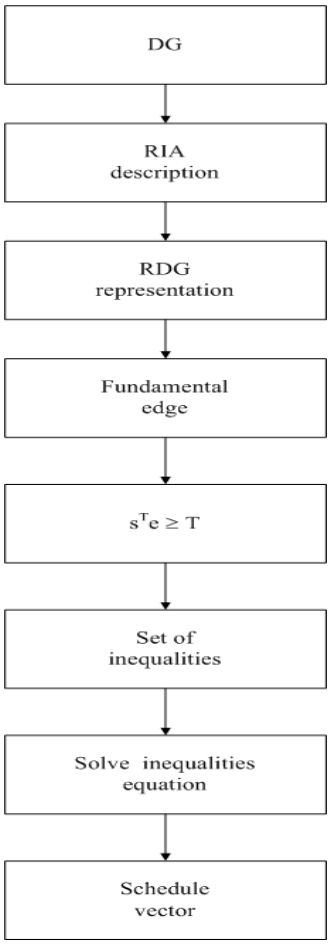


Figure 4.7 Flowchart for selection of the scheduling vector

4.12 SUMMARY

This chapter presented the concept of designing systolic array from projection,processor & scheduling vectors. With the possibility of finding the large number of vectors, it will help the designer not only to meet the requirement by performance but also by the nature of computational structure demanded by the surrounding environment where systolic architecture is a subsystem. There is a very easy and effective approach presented to meet the throughput of individual processing elements and achieving the appropriate latency by modifying the objective function.

CHAPTER 5

DESIGN EXPLORATION FOR SYSTOLIC ARRAY USING HYBRIDIZATION OF GA AND LOCAL CHAOTIC OPTIMIZATION

5.1 INTRODUCTION

In this chapter, mapping of computationally intensive applications to hardware is focused, with an aim of increasing the speed of computations. In order to meet higher performance computing speeds, an important option is to use highly parallel architectures. Additionally, for these systems, the target architecture could be a systolic array (SA) - a locally interconnected grid of elementary homogeneous processors termed processing elements (PEs). The evolved design is defined to be good with respect to PE, cycle time and utilization criterion. Array architectures leading to systolic array design is a widely researched area wherein many SA designs and their adaptations and variations have been developed. Hence the development of inexpensive special-purposes parallel architecture has led to a widespread research on linear arrays as parallel hardware architectures for parallelization of applications, by exploiting the potential of pipelining and multiprocessing Cappello (1992).

A novel methodology that can be used to design efficient systolic array for iterative algorithms is proposed in this chapter. Specifically, proposed design flow maps the DG of iterative algorithm to an array of processing elements connected locally to exploit parallelism. Due to the regular structure of this array of processors, it can be easily implemented on a

FPGA. This method is can be used for designing conventional systolic array architecture for any RIA.

5.2 IMPLEMENTING OF EVOLUTIONARY ALGORITHMS

The method of parent selection, mutation & crossover and control parameters like population size and number of generations are necessary to specify while implementing evolutionary algorithm Mohammad et al (2012).

Parent selection needs a method for identifying good parents to select for mating to produce the next generation. Tournament selection method is used in our work. In tournament selection, a random number of individuals are chosen from the population (with or without replacement) and they compete against each other. Best individual from this group is chosen as a parent for the next generation Njini & & Ekabua (2014).

Once selection has chosen fit individuals, they must be randomly altered in hopes of improving their fitness for the next generation. There are two basic strategies to accomplish this. The first and simplest is called mutation. Just as mutation in living things changes one gene to another, so mutation in a genetic algorithm causes small alterations at single points in an individual's code.

The second method is called crossover, and entails choosing two individuals to swap segments of their code, producing artificial "offspring" that are combinations of their parents.

Common forms of crossover include single-point crossover, in which a point of exchange is set at a random location in the two individuals' genomes, and one individual contributes all its code from before that point

and the other contributes all its code from after that point to produce an offspring, and uniform crossover, in which the value at any given location in the offspring's genome is either the value of one parent's genome at that location or the value of the other parent's genome at that location, chosen with 50/50 probability.

5.2.1 Control Parameters

The fundamental issue is discovering a situated of control parameters which ideally adjusts the investigation and misuse: if hybrid and transformation rates are high, a significant part of the space will be investigated, however there is a high likelihood of losing great arrangements and neglecting to adventure existing construction. The normal execution is discovered delicate to populace size, change and hybrid rates, and the quantity of hybrid focuses utilized as a part of mating.

One territory of concern utilizing EAs is untimely union of arrangements. Untimely joining alludes to a circumstance where a large portion of the populace individuals has comparative bit strings without coming to the ideal point in the space. Restricted of keeping the loss of assorted qualities is to not permit new strings that have a littler Hamming separation than a predefined worth; in any case, this confines the looking of the EA. the routines to maintain a strategic distance from untimely union are genuinely basic and include utilizing two hybrid focuses rather than one, presenting hybrid more deliberately so that it really presents variety (it is not valuable to cross from two folks which are indistinguishable), and changing the hybrid rate progressively to make up for the investigation/use transaction.

5.2.2 Functional Flow for Evolutionary Computation

Functional flow for evolutionary computation is shown in Figure 5.1 One population is selected through the uniform distribution in the range of (0-1) randomly.Taken this as the current population with repesect to the each member one offspring will be created by the gaussian distribution mutaion based offsprings and other offspring is created though cauchy distribution mutation. Like this the number offspring will be created by each member to create the final offspring population. Fitness values are estimated with use of fitness function with respect to parent offspring operation.

5.2.3 Tournament Selection

Tournament selectionis one of numerous routines for choice in hereditary calculations, which runs a "competition" among a couple of people picked indiscriminately from the populace and chooses the victor (the one with the best wellness) for hybrid effectively balanced by changing the competition size. In the event that the competition size is bigger, feeble people have a less opportunity to be chosen. Subgroups of people are looked over the bigger populace, and individuals from every subgroup go up against one another. One and only individual from every subgroup are decided to repeat. A common method is to choose every one of the people in the populace to be the N folks, to transform every guardian to frame N posterity, and to probabilistically choose, based upon wellness, N survivors from the aggregate 2N people to shape the coming up generation. Competition choice has a few advantages: it is proficient to code, chips away at parallel architectures and permits the choice weight to be effectively balanced Ekstrand et al (2008).Tournament selection applied to generate the next generation members as shown in Figure-5.2.Termination criteria is to check is it satisfy the final solution and it is the best member of the last generation.

Otherwise new generation member will be the current population and process will be repeated further until best solution is achieved from population.

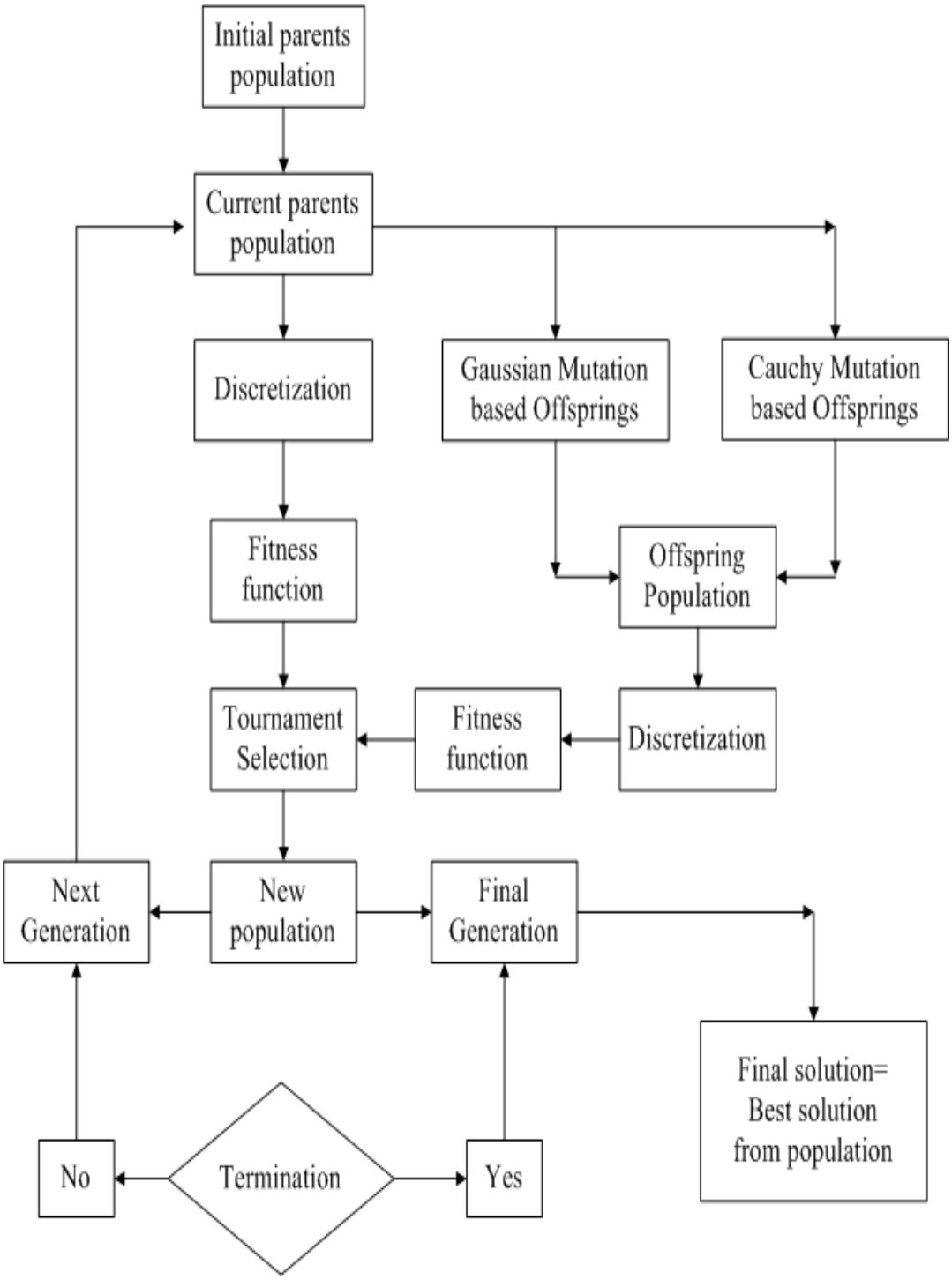


Figure 5.1 Functional flow of evolutionary computation

Natural system selection operator is ultimate which define the quality of the next generation with respect to whatever has been achieved. If selection criteria is good there will be proper pressure over the exploitation otherwise even with the quality of the exploration convergence will be poor.

Amount the possibility of the selection tournament selection provides the fairchance to prove its solutions about their capability.In result the next generation members are having better quality.With the given population, for each member ‘N’ number of opponents (10%) are selected randomly.

Comparison of the fitness takesplace for a member from ‘M’ population with ‘N’ number of opponents selected randonmly. ‘’’ is the fitness of’M’and ‘F_{Ni}’is the fittness of ‘N’.When ‘$F_{M >= F_{Ni}}$’ Tournament selected T_{Mscr} will get one extra score ($T_{Mscr} = T_{Mscr} + 1$)otherwise loss will not improve the score ($T_{Mscr} = T_{Mscr} + 0$).Finally Total Tournament Score TT_{Mscr} is checked and when all memebers have score,t his scores are sorted and the best half of the score is taken out. Corresponding to this score value available members are taken as next generation members.

Rather than selecting the member just because of fittness values each member get a several chance to prove himself, because of this there is a proper selection of right candidate.

Tournament selection has several benefits: it is efficient to code, works on parallel architectures and allows the selection pressure to be easily adjusted.

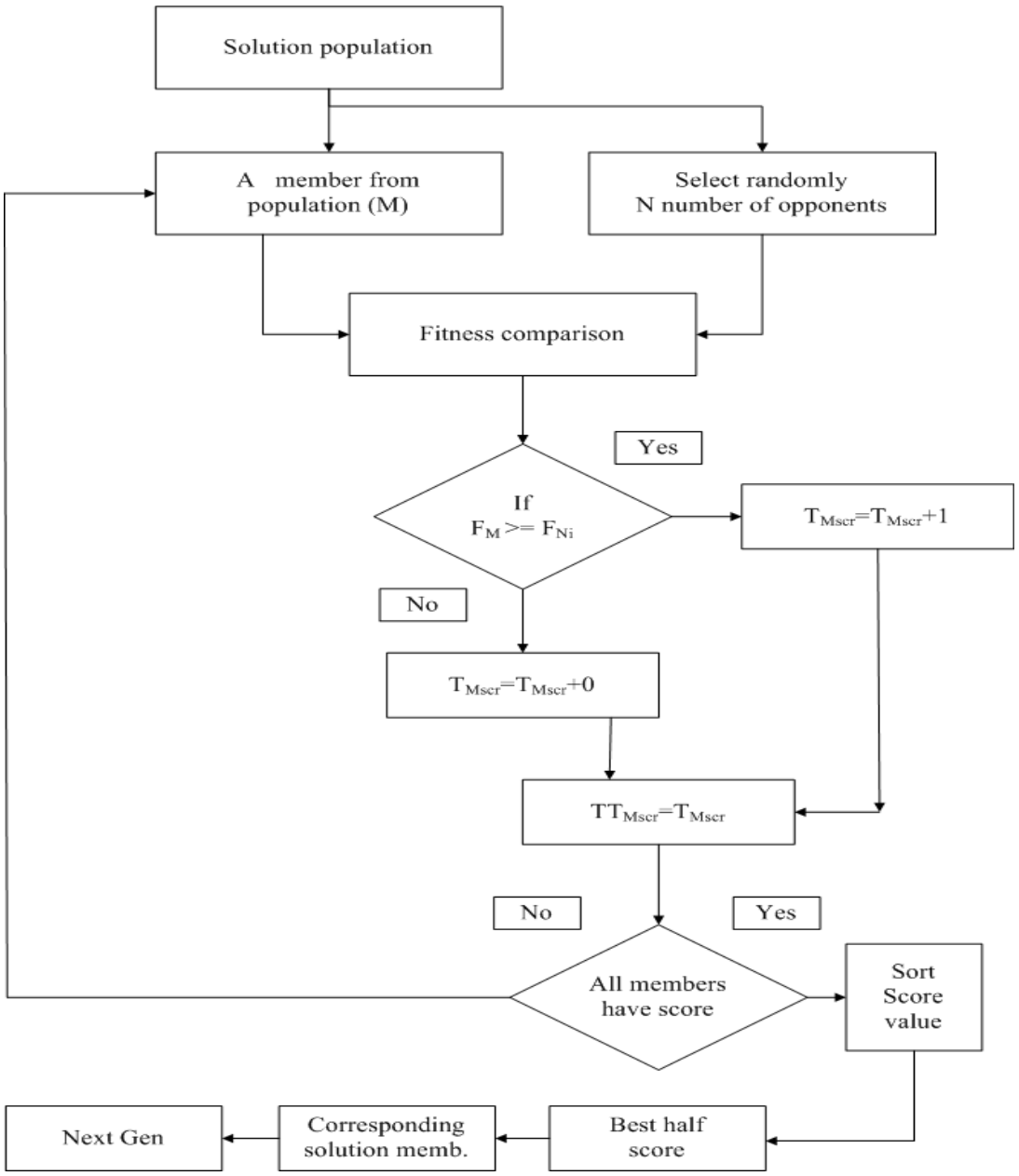


Figure 5.2 Flow for tournament selection

5.2.4 How EP Works

The essential algorithm is as given below:

- A inhabitants of μ parent outcomes x_i, i = 1,2,… μ, is

initialized over an area $M \varepsilon R^n$.

- Each parent explanation x_i, i = 1……μ, is scored because of the idea purpose f(x).
- Each parent result produces an progeny x'_i, i =1……μ, where j^{th} section of $x'_I$is determined by Equation 5.1

  $x'_{ij} = x_{ij} + N(0,\sigma_j)$, j = 1,…,n (5.1)

  In where N $(0,\sigma_j)$ is a zero mean Gaussian random element with standard deviation σ_j.
- Each progeny key x'_i, i =1……μ, is scored because of the purpose role f(x).
- Each outcome x_i and x'_i ,i =1….μ, is exhibited next to ten other arbitrarily selected solutions from the inhabitants for each assessment, a 'win' is allocated if the solution's achieve is less than or equivalent to that of its rival.
- The μ outcomes with the largest number of wins are remained to be parents of the coming up generation.
- If the obtainable calculate time is going beyond, halt. Otherwise, continue to step (iii).

The fundamental methodology has number of varieties for instance make one posterity from each guardian, the quantity of posterity can be an administrator set parameter. Additionally the quantity of rivalries in step (V) can be fluctuated to give an outside stringency of choice.

Every developing trial arrangement encoded not just the vector of item variables x to be enhanced, additionally the vector of step sizes σ that to a limited extent decide how x and σ are changed into x' and σ'. Particularly for all segments i =1

$$x_i' = x_i + \sigma_i N(0,1) \quad (5.2)$$

$$\sigma_i' = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \quad (5.3)$$

In where Ni (0, 1) is a standard Gaussian, random element re-sampled for all the i^{th} elements.

The wide factor $\tau' \cdot N(0, 1)$ permits for an in general change of the variability, whereas

$T \cdot N_i(0, 1)$ gives for people's modifications of step sizes.

5.2.5 Cauchy or Gaussian Based Mutation for Selecting Offspring

The one dimensional Cauchy density function cantered at origin is defined by Equitation 5.4

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2} \quad (5.4)$$

$-\infty < x < \infty$ where $t > 0$ is a scalar parameter.

The corresponding distribution function is defined by (2)

$$F_1(x) = \frac{1}{2} + \frac{1}{\pi}\left(\arctan\left(\frac{x}{t}\right)\right) \quad (5.5)$$

The shape of $f_t(x)$ resembles that of the Gaussian density function but approaches the axis so slowly that an expectation does not exist. As a result, the variance of Cauchy distribution is infinite as shown in Figure 5.3.

5.3 OPTIMIZATION BY EVOLUTIONARY COMPUTATION

A complex system can be studied in many different ways. The obvious way to model a complex system is to look at it from the outside, observing various types of systems behaviour and trying to summarize those using models. Assumptions about unknown mechanisms have to be made in order to get the process started, given some observable behaviour of the desired system.

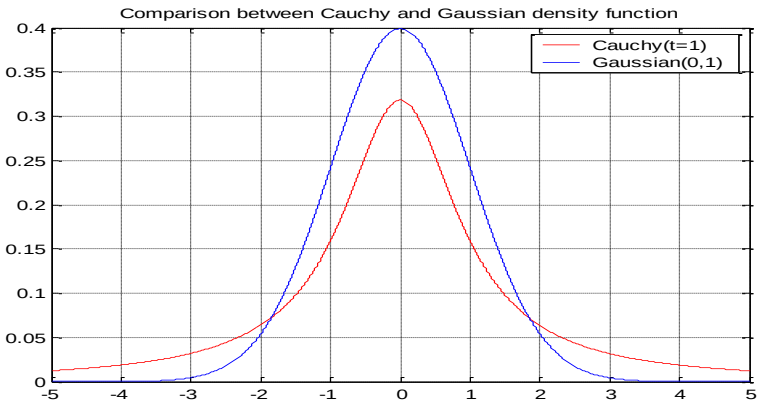


Figure 5.3 Cauchy and gaussian density function

In this paper two different models having similarity in design and computations are taken. One model is based on natural evolution called evolutionary programming (EP) and another is based on social behaviour of birds, fish schooling called particle swarm optimization (PSO). Method with which change generated from one population to another population has defined in section 5.3.1.

5.3.1 Particle Swarm Optimization Basic Algorithm

PSO's forerunner was a test system of social conduct that was utilized to picture the development of a flying creatures' herd. A few variants of the re-enactment model were produced fusing ideas, for example, closest

neighbour speed coordinating and increasing speed by separation. When it was understood that the re-enactment could be utilized as a streamlining agent, a few parameters were precluded, through an experimentation procedure, bringing about the first straightforward rendition of PSO. PSO is as EC methods in that, a populace of potential answers for the issue under thought is utilized to test the inquiry space. In any case, in PSO, every person of the populace has a versatile speed (position change), as indicated by which it moves in the inquiry place Abed & Siferd (2000).

$$V(n+1)id = \chi [w\ Vnid + C1\ r1\ (Pnid - Xnid) + C\ 2\ r2\ (Pngd - Xnid)] \quad (5.6)$$

$$X(n+1)id = X\ nid + V(n+1)id \quad (5.7)$$

In addition, every individual has a memory, recalling the best position of the pursuit space it has ever gone to. Along these lines, its development is a total quickening towards its best beforehand gone by position and towards the best individual of a topological neighbourhood. Two variations of the PSO calculation were created, one with a worldwide neighbourhood, and one with a nearby neighbourhood. As per the worldwide variation, every molecule moves towards its best past position and towards the best molecule in the entire swarm. Then again, as indicated by the nearby variation, every molecule moves towards its best past position and towards the best molecule in its limited neighbourhood. In the accompanying passages, the worldwide variation is exhibited (the nearby variation can be effectively inferred through small modifications). Assume that the pursuit space is D dimensional, then the "i^{th}" molecule of the swarm can be demonstrated to by a D-dimensional vector,

$$Xi = [xi1, xi2, \ldots\ldots xiD]. \quad (5.8)$$

The force (place change) of this element can be symbolized by another D-dimensional force Vi = [vi1, vi2, . . . viD].

The top before visited place of the i-th element is mentioned as

Pi =[pi1, pi2, . . . piD]. (5.9)

Characterizing "g" as the record of the best molecule in the swarm (i.e., the "g^{th}" molecule is the best),'n' is the best seen by that specific molecule and let the superscripts indicate the emphasis number, then the swarm is controlled by taking after two mathematical statements

In where,w is called inertia weight;
c1, c2 are two positive constants,
c1 is called cognitive parameter.
c2 is called social parameter.
χ is a constriction factor.

The part of these parameters is talked about in the following area. In the nearby variation of PSO, every molecule moves towards the best molecule of its neighbourhood. In reality, the swarm in PSO performs space computations for a few time steps. It reacts to the quality components inferred by every molecule's best position and the best molecule in the swarm, distributing the reactions in a manner that guarantees the assorted qualities. In addition, the swarm modifies its conduct (state) just when the best molecule in the swarm (or in the area, in the nearby variation of PSO) modifications accordingly, it is both versatile and steady (Mclaren 2003).

5.3.1.1 Rule of population upgradation in discrete PSO

Velocity population

$$V[j]' = \chi \times [w \times V[j] + C1 \times r1 \times (P_{GL}[j] - P[j]) C2 \times r2 \times (P_{SB}[j] - P[j])] \quad (5.10)$$

Solution population

$$P[j]' = \text{Integer}(P[j] + V[j]'); \quad (5.11)$$

Where abbreviation of notations are, V: velocity, P: individual solution, P_{GL}:Global best solution, P_{SB}: self-best solution, [j]: j^{th} parameter of a member, χ: constriction factor, w: inertia weight, C1 & C2 are positive constants, r1 & r2 are random number in the range [0 1].

5.3.1.2 Rule of Population Upgradation in Discrete EP

- Offspring population generated by Cauchy mutation

$$p'(j) = \text{Integer}(p(j) + \eta(j).\, Cj\ ) \quad (5.12)$$

by Gaussian mutation

$$p'(j) = \text{Integer}(p(j) + \eta(j).N(0,1)) \quad (5.13)$$

- Standard deviation population

$$\eta'(j) = \eta(j) \exp(\tau' N(0,1) + \tau\, Nj(0,1)) \quad (5.14)$$

Notation definition are given as, Cj is a new Cauchy random number for each j, N(0,1) is a Gaussian random variable with zero mean and

unity standard deviation and Nj(0,1) is generated a new random for each value of j. η(j) is the standard deviation for jth parameter. τ and τ' are constants and commonly set to $(\sqrt{(2\sqrt{n})})^{-1}$ and $(\sqrt{2n})^{-1}$. Where n is number of parameter which has to evolve.

A suitability capacity with the utilization of punishment system has connected to handle the requirements alongside target work as given in (7)

$$F = f + S\left[\sum_{K=1}^{r}\left(g_k^{+}(x_i)\right)^2 + \sum_{m=1}^{n}\left(h(x_i)^2\right]\right. \tag{5.15}$$

Where S is punishment consider more often than not an extensive worth and g_k and h speak to the infringement of limitations in ($\precsim 0$) and (= 0) cases correspondingly.

5.4 HYBRIDIZING OF GENETIC ALGORITHM AND LOCAL CHAOTIC OPTIMIZATION

Another advancement technique has created by hybridizing genetic algorithm and local chaotic optimization (HGACH) as indicated in Figure 5.4 to acquire the ideal arrangement in quicker and worldwide union. Qualities of riotous guide non-periodicity and beginning condition affectability convey the likelihood of more vigorous pursuit particularly in neighbourhood locale. Consequently, subsequent to applying the crossover and change operation in hereditary calculation nearby place is further sought by chaotic optimization. Lozi disorganized guide has connected to characterize the disordered example under riotous advancement. Advantage of proposed system has likewise checked with numerical benchmark issue and it is watched that proposed technique has conveyed the better exhibitions Rowe et al (2005).

5.4.1 Local Search Using Chaotic Optimization

According to Crnkovic & Larsson (2002) disarray hypothesis contains nonlinear numerical demonstrating and examination to portray the conduct of temperamental and periodic element framework or procedure. There are number of critical qualities accessible in disorder like (i) disorderly framework is progressive in nature; it implies it has time shifting characteristics. (ii) Framework conduct is not occasional and precarious. (iii) Chaotic conduct is mind boggling additionally it can have straightforward causes through deterministic procedure. (iv) It demonstrates touchy to starting conditions; it implies that for two diverse beginning conditions same procedure conveys imaginatively uncorrelated yields. (v) Because the framework is portrayed through deterministic procedure, confused conduct cannot considered arbitrary phenomena.

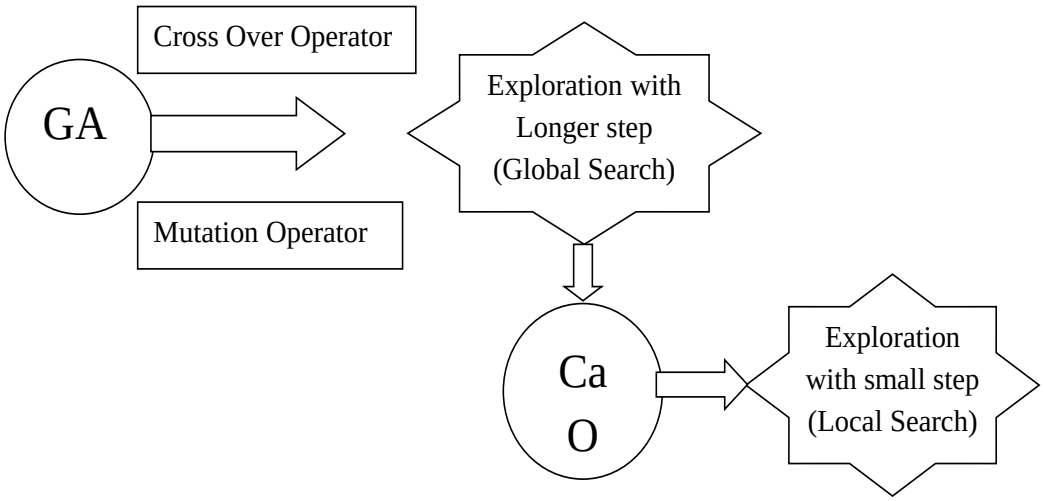


Figure 5.4 Hybridization of GA and chaotic optimization

Then again, due to the precariousness, periodicity and affectability to beginning conditions, the conduct of disorganized frameworks is not

unsurprising despite the fact that it is deterministic. (vi) There is an input component. Frameworks may show both confused and non-clamorous conduct contingent upon the control elements utilized.

5.4.2 Lozi Map

In numerous science and designing applications disarray can be connected. A vital component of clamorous frameworks is towards its starting condition sensitivity. This touchy reliance on beginning conditions is widely displayed by frameworks containing different components with nonlinear collaborations, especially when the framework is constrained and dissipative. Delicate reliance on introductory conditions can likewise be getting by straightforward mathematical statement and produced disorder example can be used to seek the arrangement space in quicker way in light of the non-reiteration in contrast with probabilistic stochastic ergodic based quests.Johnston et al (2004) says that the configuration of ways to deal with enhance the merging of disorderly improvement is a testing issue. Lozi guide is an extremely proficient strategy to produce the intense riotous pattern. The Lozi's piecewise liner model is a disentanglement of the He'non guide and it concedes unusual attractors. This disordered guide includes additionally non-differentiable capacities that troublesome the demonstrating of the related time arrangement. The Lozi guide can be characterized by Equations (5.16) and (5.17).

$$S(t+1) = 1 - P.|S(t)| + y(t) \tag{5.16}$$

$$y(t) = Q.S(t) \tag{5.17}$$

Where "t" is the emphasis number. In this work, the estimations of “y” are standardized in the reach [0, 1] to every choice variable in n-

dimensional space of enhancement issue. This change is mentioned in Equation (5.18)

$$Z(k) = \frac{y(k) - \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}} \tag{5.18}$$

In where [- 0.6418, 0.6716] and [λmax, λmin] equivalents to [0.6418, - 0.6716]. The parameters utilized as a part of this work are P=1.7and Q=0.5, these qualities demonstrate the affectability regarding beginning condition as proposed in. The investigator has done analyses for two diverse setting of introductory condition and consequences of produced tumultuous arrangements are indicated in the Figure 5.5.

Disordered arrangement are produced for two distinctive setting of parameters that are having extremely slight change in beginning condition to demonstrate the varieties in created yield. In the Figure 5.5 Data1 is created with beginning estimation of parameters S=0.1999, y=0.2, while data2 is produced with introductory estimation of parameter S=0.2, y=0.2. Indeed, even there is next to no distinction in introductory condition the produced, variety between two disorderly arrangements are demonstrated in Figure 5.5 for 50 examples. With the perception of Figure 5.5 it is clear there is an exceptionally noteworthy contrast between these two clamorous successions. This will make truly an exceptionally troublesome environment for aggressor to figure the parameters of in-statement, in the result and more powerful arrangement Zhang & Asari (2007).

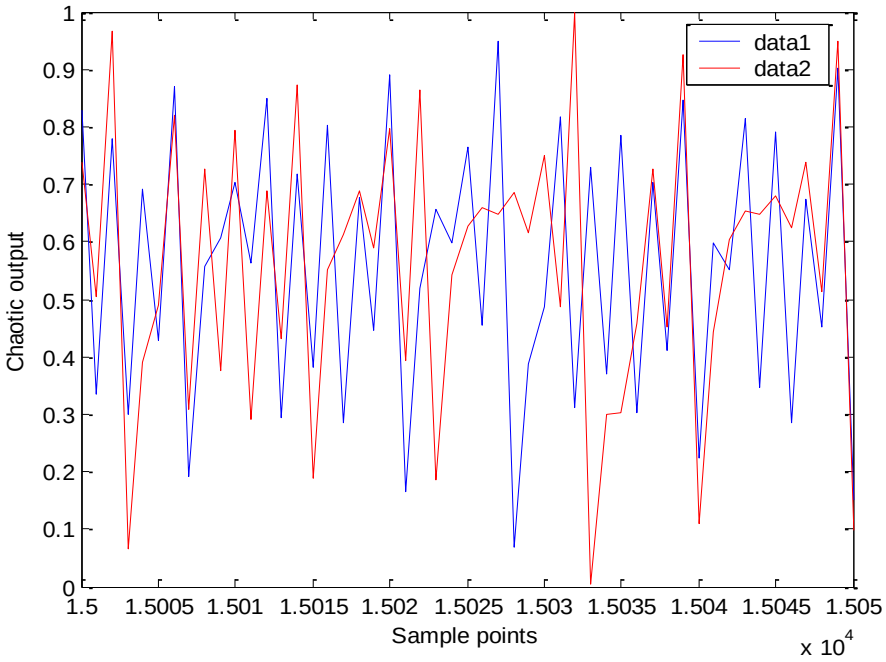


Figure 5.5 Sensitivity of chaotic sequence withtwo different initial conditions in Lozi map

5.4.3 Chaotic Optimization

Amid of the riotous neighbourhood inquiry; the stride size λ is a vital parameter in joining conduct of advancement system that confirms little sensual ranges around current arrangement.

$$V(n+1)id = \chi [w\ Vnid + C1\ r1\ (Pnid - Xnid) + C\ 2\ r2\ (Pngd - Xnid)] \quad (5.19)$$

$$X(n+1)id = X\ nid + V(n+1)id \quad (5.20)$$

Local search algorithm

Initialize λ and β

For (m2=1,……..M2) {

If r <0.5 {

(Where r consistently produced random number in variety [0 ,1])

Let $\boldsymbol{X}c = \boldsymbol{X}n + \lambda\beta|$ U- $\boldsymbol{X}n|$

else

Let $\boldsymbol{X}c = \boldsymbol{X}n - \lambda\beta|$ $\boldsymbol{X}n$-L| }

Let $\boldsymbol{\Delta} f=(\boldsymbol{X}c) - f(\boldsymbol{X}n)$

If $(\boldsymbol{\Delta} f<0)$ then $(\boldsymbol{X}n=\boldsymbol{X}c)$

$\beta = Z(\beta)$ }

The stride size λ is utilized to control the effect of the present best arrangement on the creating of another trial arrangement. A less value λ has a tendency to perform operation to purify comes about by neighbourhood pursuit, while a substantial one has a tendency to encourage a worldwide investigation.

5.4.4 Functional Flow of HGACH for Generating Vectors

Exploration is used to find out the process mechanism which could be fundamentally worked over binary or integer domain. Genetic algorithm is considered as it appears as one of the finest solution for this. One initial population is defined randomly through uniform distribution to create the offspring's two member from the population selected randomly Cross over and mutation have defined over these selected parents, in the result of two offspring's created.

To explore the surroundings area available near to offspring solution, local variant of chaotic optimization is applied (if better solution is

observed it will replace the offspring) like this a set of offspring population is created. With estimation of fitness value (parent's offspring population) tournament selection is applied to get the next generation. Depends upon outcome of termination criteria decision will be taken either to create a next generation or to get final solution.

Conventional GA fundamentally applied over cross over mutation which may not be the optimal in the surrounding regions. In result either it takes long time to create the better solution or may trap in local solution.

In proposed approach shown in Figure 5.6 chaotic search has given very high quality of exploration in the local region to make sure obtained offspring's are finest in that particular region. In result with the less a number of iteration there is a good possibility to achieve the global solution.

5.5 VECTORS GENERATION USING DPSO AND DEP

Discrete Particular Swarm Optimization and Discrete Evolutionary Programming are experimented on matrix multiplication for automatic generation of three vectors. Computations are amongst the most widely and often needed intensive computational tasks. These forms of computations form the foundation for majority of the key applications, like the processing of images and videos, digital signal, computer graphics, numerical analysis, and vision, etc. The characteristics of algorithms of matrix multiplication are like the one that are perfectly applicable for parallel processing Tselepis et al (2007).

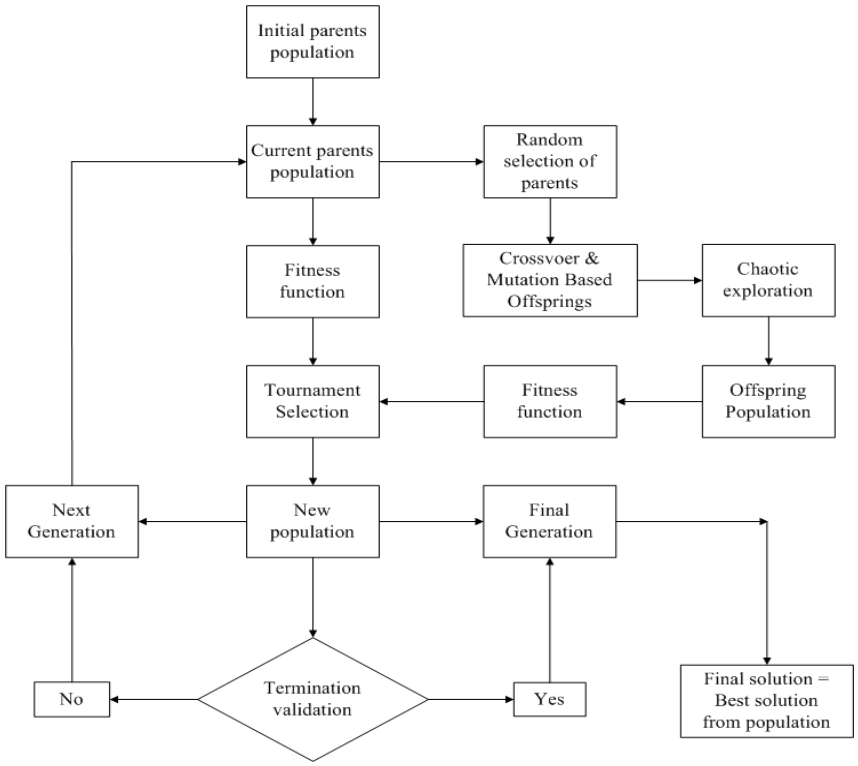


Figure 5.6 Functional flow of GA and chaotic optimization

The Dependence graph and reduced dependence graph (RDG) for matrix multiplication is shown in Figure 5.7.

Edge matrix of RIA and time taken for execution of dependent node (T_x) are the required input parameters for generating three vectors. The depended node execution time and edge matrix is shown in Figure 5.7 by assumption that consumed time for multiplication and addition equal to one unit of time and communication time equal to zero unit of time

$$EMAT = [010;100;000;000;001]$$

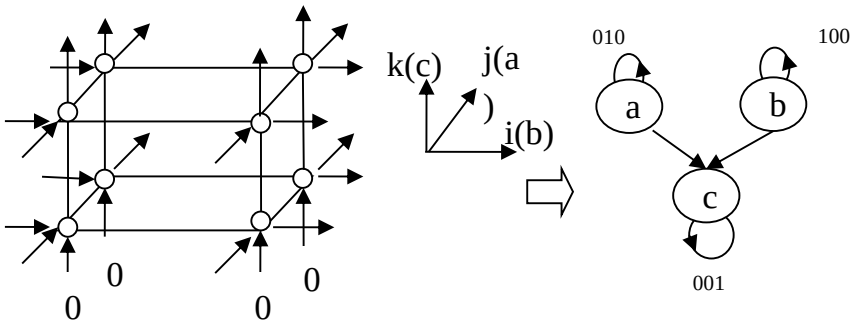


Figure 5.7 DG and its corresponding RDG ofmatrix multiplication (2×2)

Population size equal to 20 with an initialization of population as uniform random number in the discrete range of range of [-2 2] is considered for both Discrete PSO (DPSO) and in Discrete EP (DEP).Random number in the range of [0 1] uniformly is taken as velocity population for PSO and constant χ is 0.72, c1=c2=0.5, whereas '*w*' value decrease from 1.2 to 0.1 for maximum 500 iterations. η values in EP for all member is taken as Initialization of mutation vector values: 0.01.

Terminating criteria is defined as 10, for continues generation best chromosomes fitness difference should be less than 0.00001. Selection process in DEP defined as tournament selection with challenger population size equal to 10% of parent population. Penalty factor S is equal to 1000.Total 20 independent repetitions of process have given to get the different design vectors and obtained unique results are presented in Table 5.1 and in Table 5.2.

Table 5.1 Design vectors & associated performance generated by DPSO

Schedule	Projection	Processor vector	HUE(%)	Delay	Gen.
0 0 1	0 0 -1	-1 -1 0 ; 1 1 0	100	1	24
1 0 1	-1 0 0	0 1 -1 ; 0 0 -1	100	2	28
0 0 1	0 0 1	0 -1 0 ; 1 -1 0	100	1	36
0 0 1	0 1 1	-1 -1 1 ; -1 0 0	100	1	27
1 0 1	0 0 -1	1 1 0 ; 0 -1 0	100	2	24
0 0 1	0 0 -1	1 1 0 ; -1 0 0	100	1	48
0 0 1	0 0 1	1 0 0 ; -1 0 0	100	1	21
0 0 1	0 0 1	0 1 0 ; -1 0 0	100	1	30
0 0 1	0 0 1	-1 0 0 ; 1 1 0	100	1	40
0 0 1	1 0 -1	0 -1 0 ; 0 -1 0	100	1	37
1 0 1	0 0 -1	-1 0 0 ; -1 0 0	100	2	23
0 0 1	0 0 1	0 1 0 ; 0 1 0	100	1	20
1 0 1	1 0 0	0 1 0 ; 0 -1 0	100	2	34
0 0 1	0 0 1	1 0 0 ; 1 -1 0	100	1	33
0 0 1	0 0 -1	1 0 0 ; 1 0 0	100	1	32
0 0 1	0 0 -1	0 1 0 ; -1 0 0	100	1	31
0 0 1	0 0 -1	1 0 0 ; 1 1 0	100	1	20
0 0 1	0 0 -1	0 -1 0 ; -1 0 0	100	1	29
1 0 1	0 0 1	0 1 0 ; -1 1 0	100	2	20

Table 5.2 Design vectors & associated performance generated by DEP

Schedule	Projection	Processor vector	HUE(%)	Delay	Gen.
0 0 1	0 -1 1	0 -1 -1 ; -1 0 0	100	1	11
0 0 1	0 0 1	0 -1 0 ; -1 0 0	100	1	16
0 0 1	0 -1 -1	1 0 0 ; 1 0 0	100	1	11
0 0 1	0 0 -1	-1 -1 0 ; -1 0 0	100	1	11
0 0 1	0 -1 1	1 0 0 ; -1 0 0	100	1	11
0 0 1	1 -1 -1	0 1 -1 ; 0 -1 1	100	1	11
0 0 1	0 -1 1	1 -1 -1 ; -1 1 1	100	1	11
1 0 1	0 -1 -1	1 -1 1 ; -1 0 0	100	2	12
0 1 1	0 0 -1	1 1 0 ; -1 1 0	100	2	11
0 0 1	-1 -1 -1	-1 1 0 ; 1 0 -1	100	1	11
0 0 1	0 0 1	0 1 0 ; 1 0 0	100	1	11
0 0 1	0 0 1	0 1 0 ; -1 0 0	100	1	11
0 0 1	0 0 -1	1 0 0 ; 1 0 0	100	1	11
0 0 1	-1 1 1	0 1 -1 ; 1 0 1	100	1	11
1 0 1	0 0 1	1 1 0 ; -1 0 0	100	2	14
0 0 1	-1 0 1	-1 0 -1 ; 1 0 1	100	1	11
1 0 1	0 1 1	0 1 -1 ; 0 -1 1	100	2	11
0 0 1	1 1 1	1 -1 0 ; 0 -1 1	100	1	11
1 0 1	-1 0 0	0 -1 1 ; 0 1 -1	100	2	12
0 0 1	0 0 -1	0 1 0 ; 1 -1 0	100	1	36

Algorithm for automatic generation of three vectors and its solution is developed in the environment of MATLAB. Comparison between DPSO and DEP have given with respect to performance parameters like average value, hardware utilization efficiency (HUE), Total number of delay and number of generation required to converge the process. Figure 5.8 shows the performance plot for the vectors generated using DPSO.

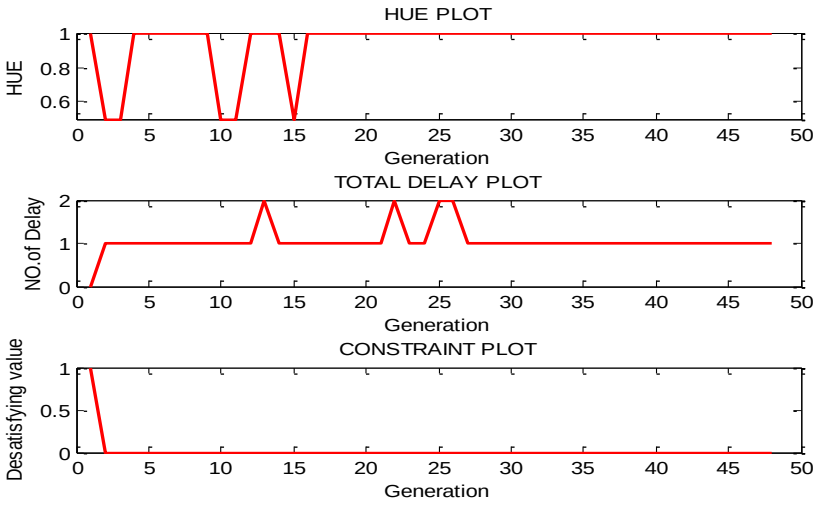


Figure 5.8 Performance plot with generation in DPSO (a) HUE plot (b) Delay plot (c) Constraint plot

Figure 5.9 shows the performance plot for execution of DEP. In DEP, from one parent, two offspring are generated by mutation operation, one from Cauchy distribution another from Gaussian distribution. Longer jump in Cauchy distribution has advantage over small jump in Gaussian distribution sometimes, which gives a better exploration.

As from the graph it's very clear, designed evolutionary programming algorithm satisfies the constraints within very little number of generations (approximately around 10 generation) and contains the optimal

solution. Plot in Figure 5.5 (c) and in Figure 5.56 (d) signify the contribution of parent population, and offspring's created by Cauchy and Gaussian distribution in the formation of the new generation. Graphs justify the importance of hybridization of mutation mechanism. As it was expected there is no clear winner in the generation of better chromosomes.

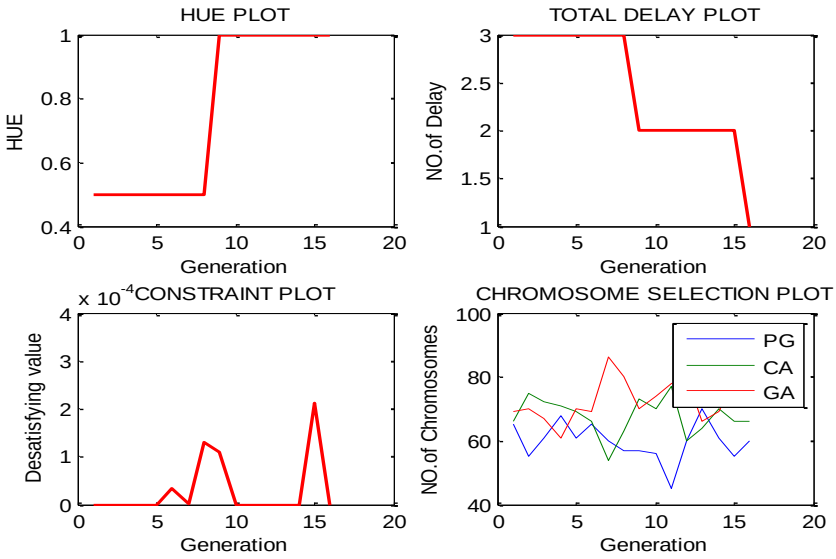


Figure 5.9 Performance plot with generation for DEP (a) HUE plot (b) constraint plot(c) delay plot (d) chromosome selection for next generation

Hence experiment shows the support for hybridization rather than having a single mutation strategy for generating next generation. From performance results it is clear that in terms of HUE and delay, both algorithms have perform equally well whereas DEP outperform the DPSO in terms of convergence time as shown in Table 5.3.

Table 5.3 Average Performance comparison between DPSO and DEP

Algorithm	Avg.HUE (%)	Delay	Generation Convergence rate (%)	
DPSO	100	1.25	29.65	100
DEP	100	1.26	11.52	100

5.5.1 Vector Generation using HGACH and comparison to GA

The Goldstein-Price & Six-hump camel back test functions shown in Equation (5.21) and in Equation (5.22) are taken as a problem of minimization to define the experiments with dimension size equal to 2. F1 and F2 are both multimodal problems and contain number of local minima. Maximum numbers of allowed iterations in both algorithms are equal to50. In case of HGACH, associated local chaotic optimization has given 50 iterations for each member of the offspring population. Size of population in all cases has taken as 20.In all test cases 10 independent trials have given to analyse the performance in terms of best achieved, worst possible seen, mean and standard deviation. Probability of mutation has taken as 0.1.

- The Goldstein-Price function

$$f(x) = (1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) * (30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2)). \quad (5.21)$$

-2 <= x1 <= 2, -2 <= x2 <= 2.

Global minimum $f(x_1,x_2)=3$; $(x_1,x_2)=(0,-1)$.

- Six-hump camel back function

$$F2(x,y) = (4 - 2.1\ x + \ x^4/3)\ x^2 + xy + (- 4 \ + \ 4y^2)\ y^2; \qquad (5.22)$$

$$-3 <= x <= 3, -2 <= y <= 2.$$

Global minimumf (x, y) = - 1.0316;

(x, y) = (-0.0898,0.7126), (0.0898,-0.7126).

Table 5.4 Performance comparison between GA &HGACH for Goldsteinprice test function

	GA			HGACH		
Trail No.	P1	P2	Function value	P1	P2	Function value
1	0	0.9404	4.5402	0.0016	0.9994	3.0006
2	0	0.9492	4.1078	0	1.0000	3.0000
3	0	0.9384	4.6489	0.0003	1.0004	3.0001
4	-0.2165	-1.2240	52.3808	0.0006	1.0010	3.0004
5	0	-1.0395	3.7445	-0.0003	-1.0011	3.0005
6	-0.0822	-0.8351	16.6464	-0.0024	-1.0001	3.0015
7	-0.5898	-0.4223	30.2877	0.0009	-1.0003	3.0003
8	-0.3636	-0.6807	36.1564	-0.0001	-0.9991	3.0004
9	-0.1004	-0.9656	6.7336	0.0003	-1.0000	3.0000
10	-0.0553	-0.7878	28.4166	0	-0.9996	3.0001

Performances obtained for Goldstein price function is shown in Table 5.4. HGACH has delivered the most appreciable result and very close to global solution whereas GA in all trails could not converge properly.

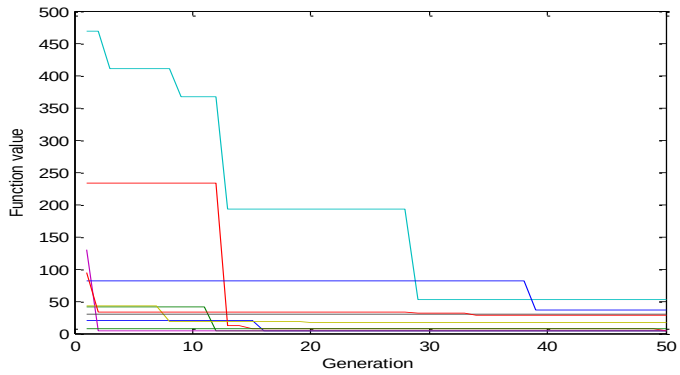


Figure 5.10 Convergence characteristics of GA with respect to Golden stein price function in 10 independent trails

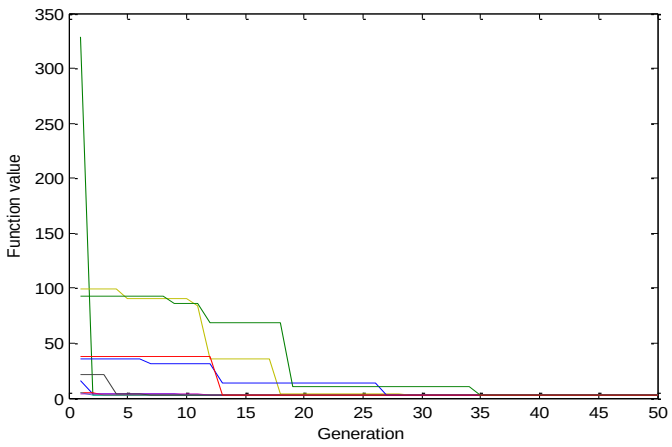


Figure 5.11 Convergence characteristics of HGACH with respect to Golden stein price function

Convergence characteristics of HGACH and GA are shown in Figure 5.10 and Figure 5.11 for all the 10 independent trails. Each different colour represents the convergence characteristics of independent trail. It can

be observed very clearly that HGACH has the fastest rate of convergence in comparison to GA. Each colourrepresents the convergence behaviour of each trail.

In second test Six-hump camel back function in which within the bounded region there are six local minima located performances as shown in Table5.5.

Table 5.5 Performancesfor six hump camel back test function

Trail No.	GA			HGACH		
	P1	P2	Function value	P1	P2	Function value
1	0.0654	-0.6804	-1.0219	0.0882	-0.7127	-1.0316
2	-0.1233	0.6734	-1.0140	-0.0889	0.7134	-1.0316
3	0	0.7223	-0.9981	0.0905	-0.7139	-1.0316
4	0	0.4562	-0.6593	0.0900	-0.7129	-1.0316
5	0	0.6596	-0.9831	- 0.0906	0.7127	-1.0316
6	0	-0.5952	-0.9150	0.0901	-0.7136	-1.0316
7	0	-0.8475	-0.8093	0.0897	-0.7118	-1.0316
8	0	0.6274	-0.9547	- 0.0889	0.7135	-1.0316
9	0	0.6978	-0.9993	- 0.0899	0.7133	-1.0316
10	-0.0843	0.7898	0.9769	0.0882	-0.7127	-1.0316

Convergence characteristics of HGACH and GA for all ten independent trails are shown in Figure 5.12 and Figure 5.13.HGACH has obtained the global solution in all trails whereas GA fails to obtain for many functions values.

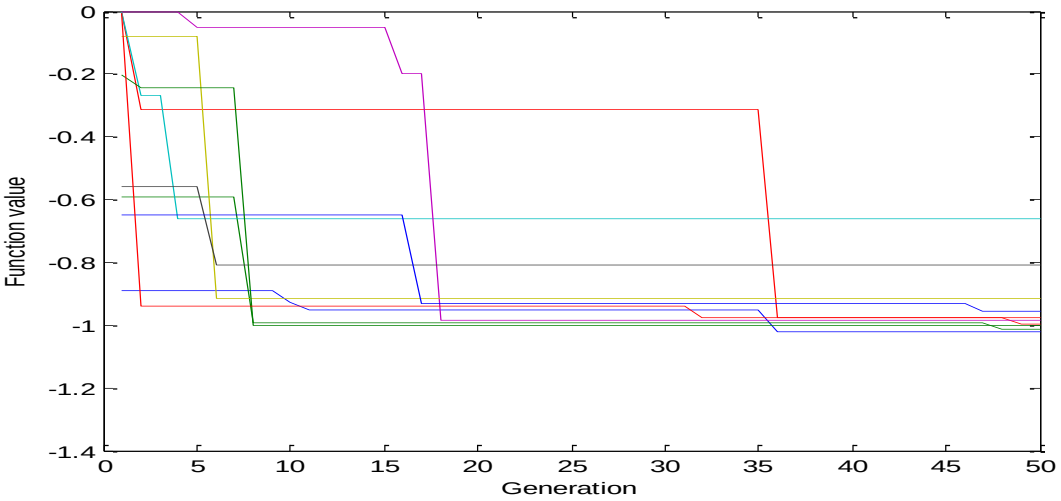


Figure 5.12 Convergence characteristics of GA with respect to six hump camel back function

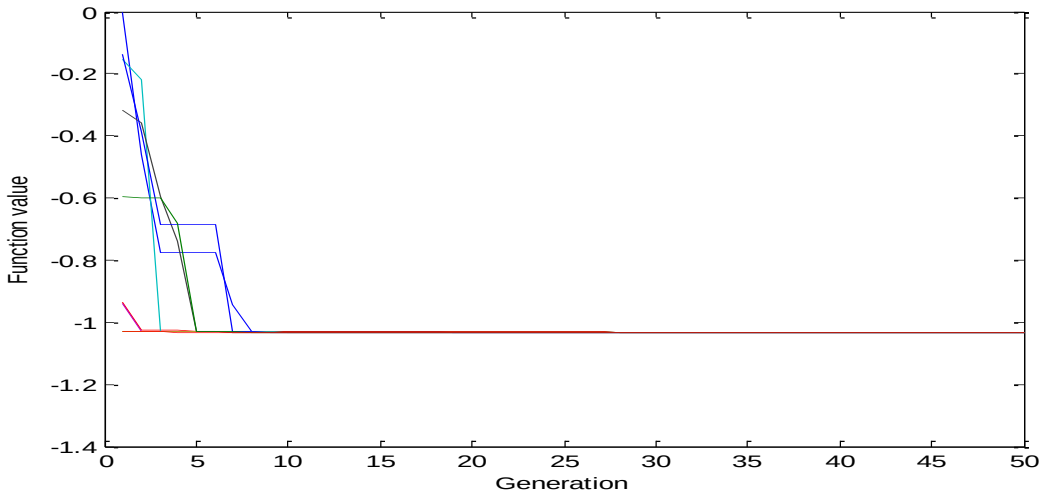


Figure 5.13 Convergence characteristics of HGACH with respect to six hump camel back function

It can be concluded from Table 5.6 that fastest convergence in all trails has been observed with HGACH and it is best suitable for generating three vectors for designing efficient systolic array.

Table 5.6 Comparative performances for F1 and F2test function

	$f1$		$f2$	
Perf	GA	HGACH	GA	HGACH
Best	3.7445	3.0000	-1.0219	-1.0316
Worst	52.3808	3.0015	-0.6593	-1.0316
Mean	18.7663	3.0004	-0.9332	-1.0316
Std. Dev	17.1584	0.0004	0.1148	0.0000

5.6 DESIGNING OF SYSTOLIC ARRAY FOR RIA

In this section different systolic array architecture is designed using Projection vector, processor vector and schedule vector obtained using HGACH method for matrix multiplications, Convolution, Vector Quantization and Digital Correlator.

5.6.1 Matrix Multiplication

Design 1: Systolic array design for matrix multiplication using vectors generated by HGACH

Projection vector, processor vector and schedule vector obtained by HGACH is shown in Table 5.7 and HUE for design is obtained using Equation 4.3

Table 5.7 Design vectors generated by HGACH and corresponding HUE for design 1

Projection Vector	Processor Vector	Schedule Vector	HUE (%)
[0 1 1]	[0 - 1 1 ; 1 0 0]	[1 0 1]	1

Table 5.8 shows the edge mapping corresponding to design vectors from Table 5.7

Table 5.8 Edge mapping in systolic array for design 1

RIA Edge	Mapped Edged in SA	Delay with Mapped Edge in SA
a(0, 1, 0)	(-1,0)	0
b(1, 0, 0)	(0,1)	1
c(0, 0, 1)	(1,0)	1

Performance plot for HUE, vector's generation and constraint plot for design 1 are also shown in Figure 5.14. Constraints are met at 2nd generation but HUE reaches maximum and delay is better after 9th generation .The corresponding 2D systolic architecture of this design is shown in Figure 5.15.Data 'a' flows from left to right and data 'b' flows from top to bottom and data 'c' flow is from right to left.

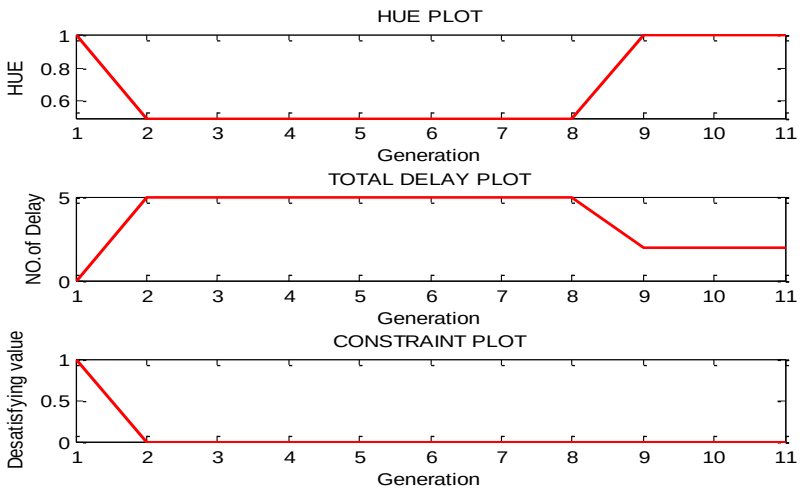


Figure 5.14 HGACH Performance plot with generation for design 1

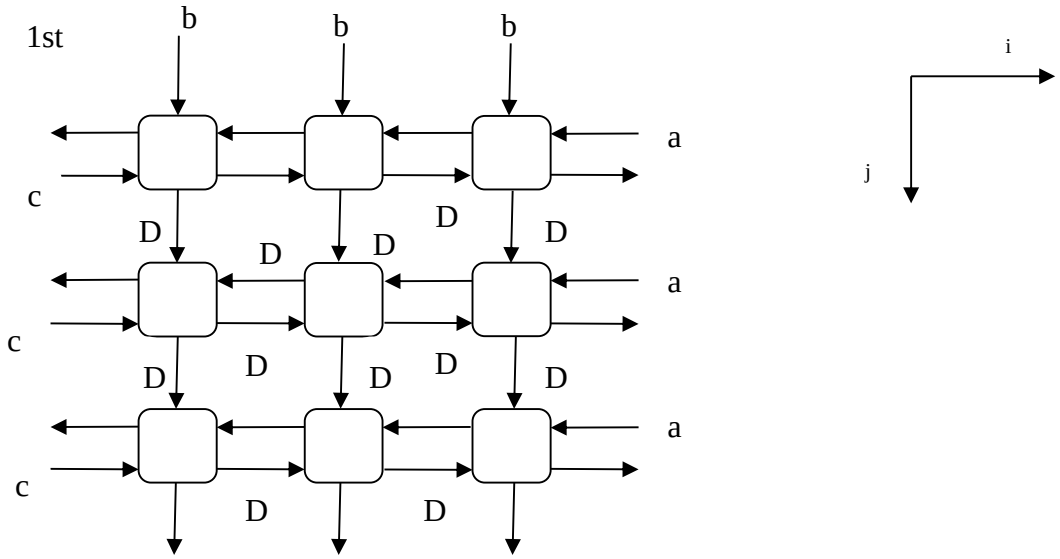


Figure 5.15 Systolic architecture for design 1

Design 2:

Table 5.9 Design vectors and corresponding HUE generated by HGACH for design 2

Projection Vector	Processor Vector	Schedule Vector	HUE(%)
[0 1 0]	[-1 0 0 ; 0 0 -1]	[0 1 1]	1

Table 5.10 Edge mapping in systolic array for design 2

Edge	Mapped Edged in SA	Delay with Mapped Edge in SA
a(0, 1, 0)	(0,0)	1
b(1, 0, 0)	(-1,0)	0
c(0, 0, 1)	(0,-1)	1

Various design vectors have obtained by HGACH as shown in Table 5.10 and corresponding edge mapping has shown in Table 5.11. The corresponding 2D systolic architecture of this design is shown in Figure 5.16

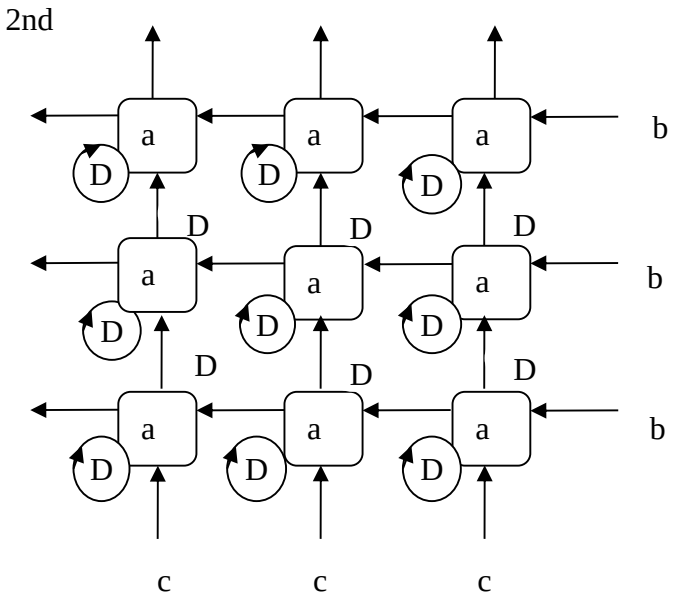


Figure 5.16 Systolic architecture for design 2

Design 3:

Table 5.11 Design vectors and corresponding HUE generated by HGACH for design 3

Projection Vector	Processor Vector	Schedule Vector	HUE(%)
[0 1 0]	[0 0 1 ;-1 0 1]	[0 1 1]	1

Table 5.12 Edge mapping in systolic array for design 3

RIA Edge	Mapped Edged in SA	Delay with Mapped Edge in SA
a(0, 1, 0)	(0,0)	1
b(1, 0, 0)	(0,-1)	0
c(0, 0, 1)	(1,1)	1

Various design vectors have obtained by HGACH as shown in Table 5.12 and corresponding edge mapping has shown in Table 5.13. The corresponding 2D systolic architecture of this design is shown in Figure 5.17

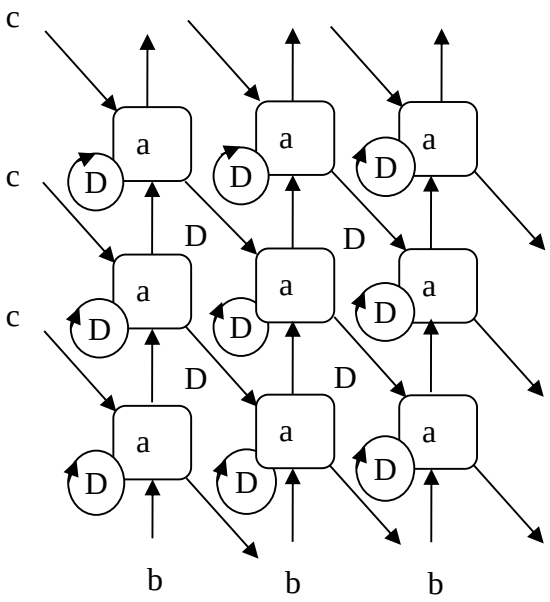


Figure 5.17Systolic architecture for design 3

Design 4:

Table 5.13 Design vectors and corresponding HUE generated by HGACH for design 4

Projection Vector	Processor Vector	Schedule Vector	HUE(%)
[1 0 0]	[0 1 -1 ; 0 1 1]	[1 0 1]	1

Table 5.14 Edge mapping in systolic array for design 4

RIA Edge	Mapped Edged in SA	Delay with Mapped Edge in SA
a(0, 1, 0)	(1,1)	0
b(1, 0, 0)	(0,0)	1
c(0, 0, 1)	(-1,1)	1

Various design vectors have obtained by HGACH as shown in Table 5.14 and corresponding edge mapping has shown in Table 5.15. The corresponding 2D systolic architecture of this design is shown in Figure 5.18

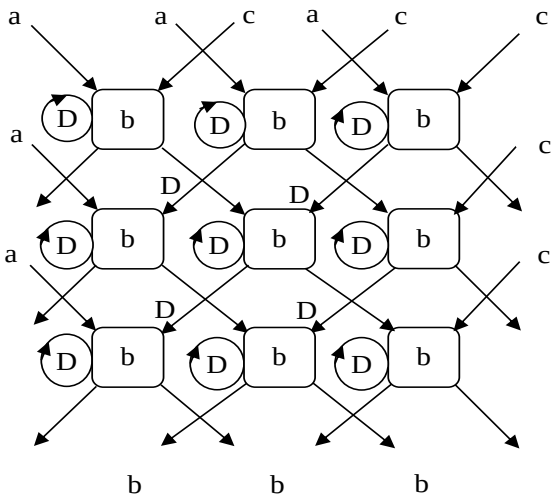


Figure 5.18 Systolic architecture for design 4

Design 5:

Table 5.15 Design vectors and corresponding HUE generated by HGACH for design 5

Projection Vector	Processor Vector	Schedule Vector	HUE(%)
[1 -1 0]	[-1 -1 0 ; 0 0 -1]	[1 0 1]	1

Table 5.16 Edge mapping in systolic array for design 5

RIA Edge	Mapped Edge in SA	Delay with Mapped Edge in SA
a(0, 1, 0)	(-1,0)	0
b(1, 0, 0)	(-1,0)	1
c(0, 0, 1)	(0,-1)	1

Various design vectors have obtained by HGACH as shown in Table 5.15 and corresponding edge mapping has shown in Table 5.16. The corresponding 2D systolic architecture of this design is shown in Figure 5.19

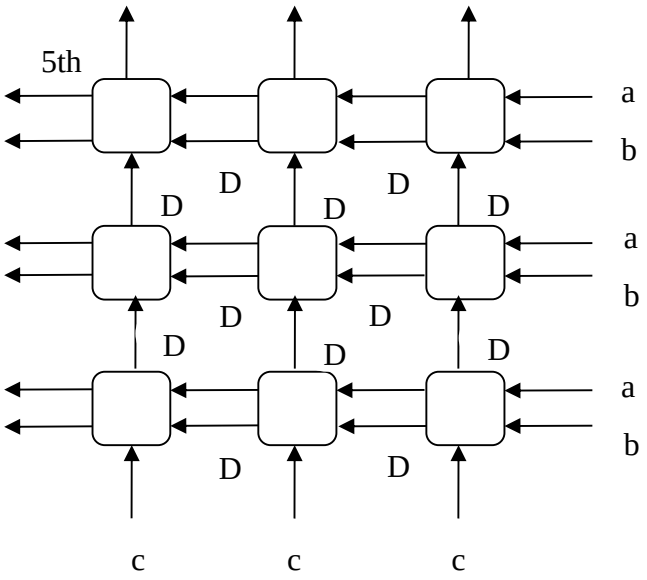


Figure 5.19 Systolic architecture for design 5

There is also possible to design fully pipelined systolic architecture in with all edges having at least one delay unit, with placing this as a constraint in solution development. Two architectures systolic architecture which contains fully pipelined facility is in Design 6 and Design 7.

Design 6:

Table 5.17 Design vectors and corresponding HUE generated by HGACH for design 6

Projection Vector	Processor Vector	Schedule Vector	HUE(%)
[0 0 1]	[0 -1 0 ; 1 0 0]	[1 1 1]	1

Table 5.18 Edge mapping in systolic array for design 6

RIA Edge	Mapped Edged in SA	Delay with Mapped Edge in SA
a(0, 1, 0)	(-1,0)	1
b(1, 0, 0)	(0,1)	1
c(0, 0, 1)	(0,0)	1

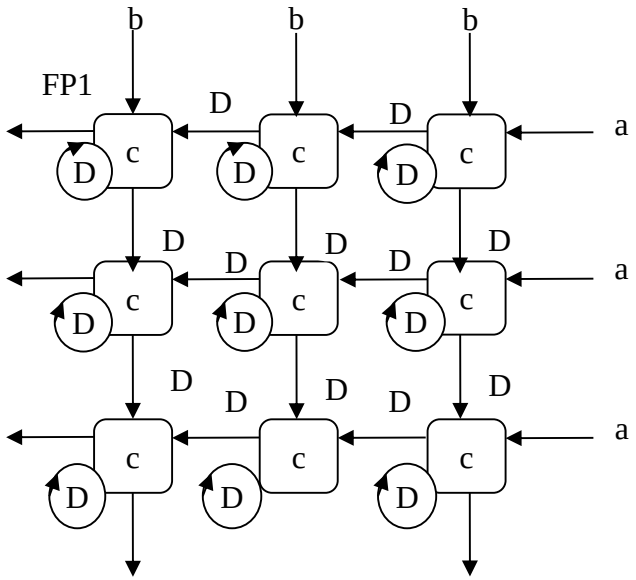


Figure 5.20 Systolic architecture for design 6

Design 7:

Table 5.19 Design vectors and corresponding HUE generated by HGACH for design 7

Projection Vector	Processor Vector	Schedule Vector	HUE(%)
[-1 0 0]	[0 1 0 ; 0 0 1]	[1 1 1]	1

Table 5.20 Edge mapping in systolic array for design 7

RIA Edge	Mapped Edged in SA	Delay with Mapped Edge in SA
a(0, 1, 0)	(1,0)	1
b(1, 0, 0)	(0,0)	1
c(0, 0, 1)	(0,1)	1

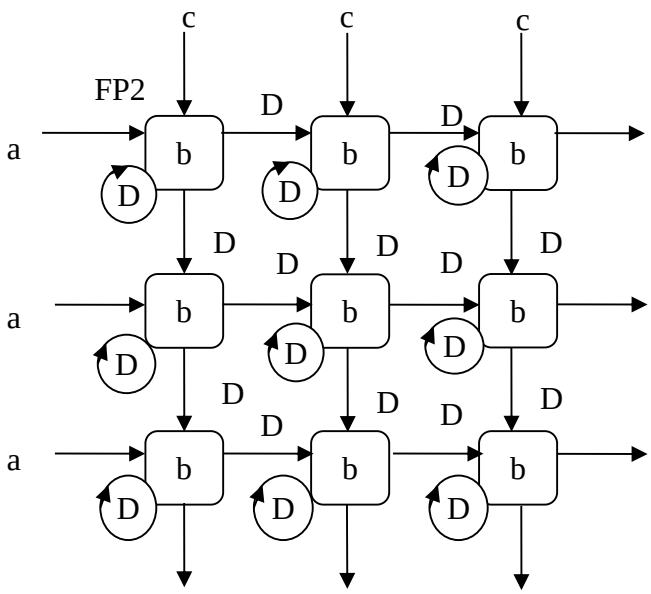


Figure 5.21 Systolic architecture for design 7

5.6.2 Systolic Array Design for Convolution Algorithm

One of the first applications of the Cooley-Tukey fast Fourier transform (FFT) algorithm was to implement convolution faster than the usual direct method Sockham (1996) & Helms (1967) & Arunachalam et al (2013).

Convolution of two discrete sequence h(n) and x(n) is defined as

$$y(n)=x(n)*h(n) \tag{5.23}$$

$$Y(n) = \sum_{k=-\infty}^{\infty} X(n)H(n-k) \tag{5.24}$$

where, H(n)is a length-N sequence of numbers considered to be the input signal,

X(n) is a sequence of numbers considered to be the filter coefficients, and Y(n) is the filtered output.

The output at time instance n, Y(n) can be viewed as the inner product between X(n) and h(-k+n) summed over $-\infty < k < \infty$.

Let

$$X= \{x_1, x_2, x_3 \ldots\ldots\ldots\ldots x_k\}$$

$$W= \{w_1, w_2, w_3 \ldots\ldots\ldots w_k\} \tag{5.25}$$

$$Y= \{y_1, y_2 , y_3 \ldots\ldots\ldots\ldots y_k\}$$

Where $y_i = x_i w_1 + x_{i+1} w_2 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots x_{i+l-1} w_l$

Let

$$W = \{w_1, w_2, w_3, w_4\}$$

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

$$Y = \{y_1, y_2, y_3\}$$

$$Y_1= x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$

$$Y_2=x_2 w_1+ x_3 w_2+ x_4 w_3+ x_5 w_4$$

$$Y_3=x_3\ w_1+ x_4\ w_2+ x_5\ w_3+ x_6\ w_4 \quad (5.26)$$

From Equation (5.26) dependence graph of Figure 5.22 is derived. Each node in the graph performs a multiplication and an addition

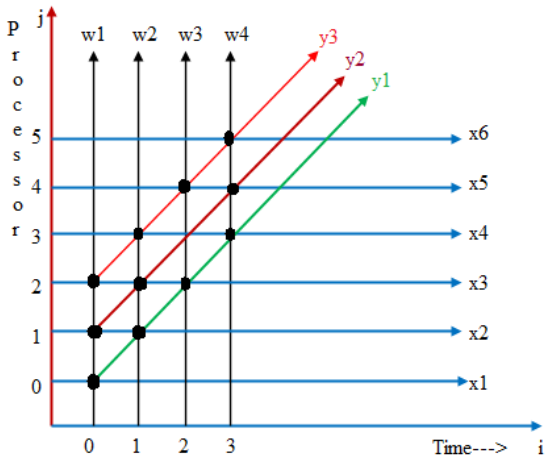


Figure 5.22 Dependence graph representation of convolution

Table 5.21 Design vectors and corresponding HUE generated by HGACH for convolution

Projection Vector	Processor Vector	Schedule Vector	HUE(%)
[1 0]	[0 1]	[1 1]	1

5.6.2.1 Edge mapping

Edge mapping concept is applied to transform the edge available from DG to space diagram where each edge from the dependence graph is accepted by the process (P^TI) with the associated delay S^TI

A DG can be transformed to a space-time representation by interpreting one of the spatial dimensions as temporal dimension.

In Table 5.23 “Wt” is the edge from the DG diagram, “i/p” is the signal input and the result edges are mapped over the processor with the corresponding associate delay.

Table 5.22 Edge mapping

e^T	P^TI	S^TI
Wt(1 0)	**0**	**1**
i/p(0, 1)	**1**	**1**
Result(1, 1)	**1**	**2**

5.6.2.2 Space-time diagram

Space diagram is mapping representation of the operations available in the DG to the processor and represents their execution time. The shape of the space time diagram is depending up on the character of mapping given by the edge mapping and it is shown in Figure 5.23.

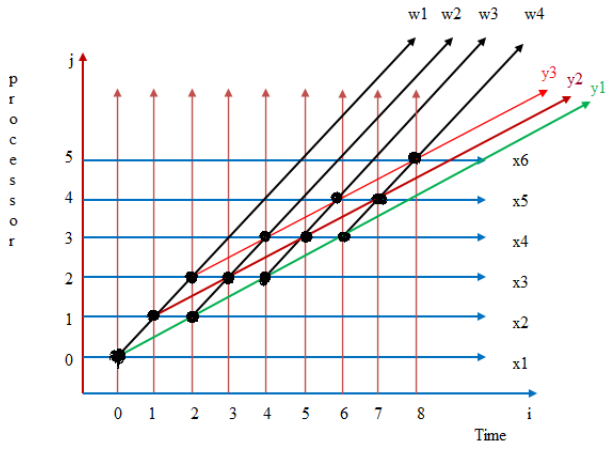


Figure 5.23 Spacediagram for convolution algorithm

Table 5.24 explains the processing completed at particular time unit. As shown at time '0' processing element 'P_1' executes inputs "X_1* W_1" .At time "1" weight input "W_1" is shifted to processing element 'P_2" and "X_2* W_2" operation takes place and this operation is continued for all weights (W_1 to W_4) and inputs ($X_1$to X4).

Table 5.23 Analysis of space-time diagram

P	0	1	2	3	4	5	6	7	8
P_1	X_1		X_3	X_3	X_3		X_5	X_5	
P_2		X_2	X_2		X_4	X_4	X_4		X_6
P_1	W_1		W_1	W_2	W_3		W_3	W_4	
P_2		W_1	W_2		W_2	W_3	W_4		W_4

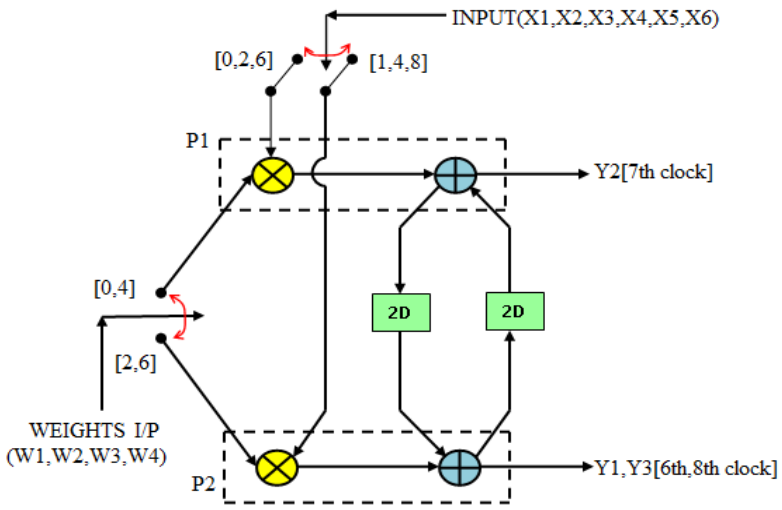


Figure 5.24 Architecture for convolution algorithm and its switching activity

Figure 5.24 is the architecture designed for Convolution Algorithm from Figure 5.23 Space diagram. The switching key activity depends upon available time slot of the particular processing element. There are two PEs **P_1&P_2** and each processing element has one multiplier and an adder.

As shown in table 5.25 at "**P_1**" executes "X_1* W_1" and at first unit of time PE "**P_2**" executes "X2*W_1" and at second unit of time "**P_1**"& "**P_2**" "$x_{3*}w_1$" and "$x_2w_2+x_1w_1$" respectively. Output "Y_1" is available at 6^{th} unit of time, "Y_2" at 7^{th} unit of time and "Y_3" at 8^{th} unit of time. Same is shown in figure 5.24.

Table 5.24 Switchingactivities for convolution architecture

T	P_1	P_2
0	x_1 w_1	0
1	0	x_2w_1
2	x_3w_1	$x_2w_2+x_1w_1$
3	$x_3w_2+x_2w_1$	0
4	$x_2w_2+x_1w_{1+}x_3w_3$	x_3w_{1+} x_4w_2
5	0	$x_4w_3+x_3w_2+x_2w_1$
6	x_3w_{1+} x_4w_{2+} x_5w_3	$x_4w_{4+}x_3w_{3+}x_2w_2+x_1w_1$
7	$x_5w_4+x_4w_3+x_3w_2+x_2w_1$	0
8	0	$x6w4+x_3w_{1+}$ x_4w_{2+} x_5w_3

5.6.3 Systolic Array Design for Vector Quantizer

Vector Quantization (VQ) that originated as a pattern matching scheme is now commonly utilized for data compression in speech, image and video coding and speech recognition. It is a Lossy compression technique which exploits the spatial correlation existing amidst the neighbouring signal samples. In VQ a group, the samples are quantized together rather than individually. A vector quantizer could operate directly on image blocks to achieve spatial information compression and quantization at the same time. In a VQ system, identical code books containing code word vectors are obtainable both in the receiver and the transmitter sides. The vector quantizer transmits the index of the code word rather than the code word itself Parhi (1999).

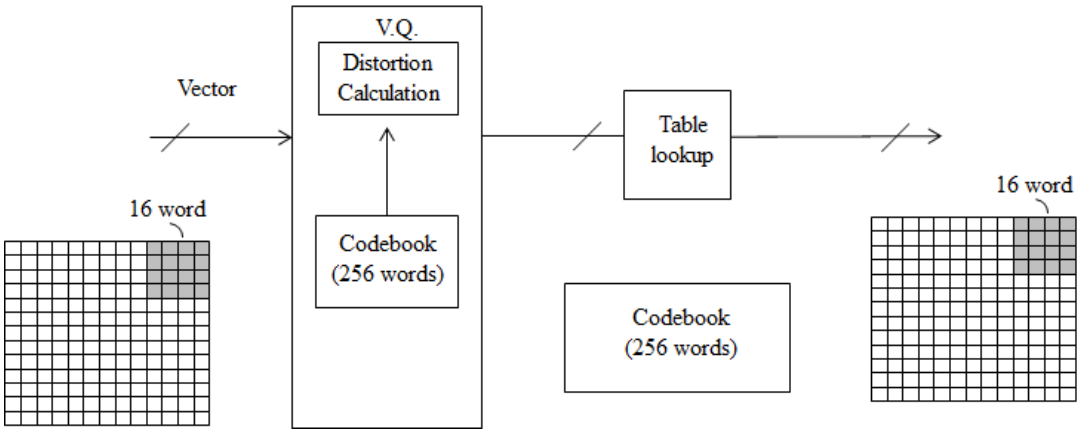


Figure 5.25 Vector quantization encoding and decoding process

Encoder side the vector quantizer takes a group of input sample (pixels in the scenario of compression of image), associates this input vector to the code words in the code book and choses the code word with least distortion. Accepting that vectors are K dimensional and the code book size is N. If the word length of the vector elements is W and N=2m, then the m bit address of the code book is transmitted as divergent to KW bit. This forms a

compression factor of m/KW. The decoder only receives the m bit index as the address of the code book and recovers the best code word to rebuild the input vector. In the Figure 5.25, each vector contains K=16 pixels of word length W=8. Code book contain N=856 code words, hence, m=8. Therefore, vector quantizer achieves a compression factor of 1/16.

The encoding algorithm in vector quantizer can be viewed as an exhaustive search algorithm, where the computation of distortion is performed sequentially on every code word vector in the code book, keeping track of the minimum distortion so far and continuing until every code word vector has been tested.

Two vectors (also called square error) are used as distortion measurement.

$$d(X,Y) = \| X - Y \|^2 = \sum_{i=0}^{k-1} (x_i - y_i)^2 \quad (5.26)$$

In practical implementation, the distortion between input vector "X" and the j^{th} code vector "Cj" (0 <= j < =N-1) is computed based on their inner product, instead of direct squaring operation.

From the above expression,

$$d\ (X, C_j) = \| X \|^2 - 2(X.\ C_j + e_j) \tag{5.27}$$

$$e_j = -\frac{1}{2} \| C_j \|^2 = -\frac{1}{2} \sum_{i=0}^{K-1} C_{2ji} \tag{5.28}$$

$$d(X, C_j) = \| X \|^2 - 2(X \cdot C_j + e_j) \tag{5.29}$$

The inner product is given by

$$X.\ C_j = \sum_{i=0}^{K-1} x_i\, c_{ji} \tag{5.30}$$

Since e_j depends only on the code word vector C_j and is a constant, it can be pre-computed and treated as an additional component of the vector C_j.

Therefore, for a fixed input vector x, minimizing the distortion

$$X_i\, c_j + e_j \tag{5.31}$$

where $0 \leq j \leq N - 1$

$$d(X, C_j) = \| X \|^2 - 2(X.c_j + e_j) \tag{5.32}$$

Min → Max

Therefore the search process in VQ can be described as follows:

$$ind_n = \left(Min_{o \leq j \leq N-1}{}^{m}\ dj\right) \equiv Max_{o \leq j \leq N-1}^{k-1} \sum_{i=0}^{k-1} \left(x_i^m c_{ji} e_j\right) \tag{5.33}$$

where ind_n → output index, n → represent time instance.

5.6.3.1 Matrix representation

The search process can also be described equivalently in a matrix-vector multiplication formulation followed by comparison as shown in Equation 5.34

$$D = [d_0, d_1, d_2, \ldots, d_{N-1}] = CX + e \quad (5.34)$$

$$ind_n = \max (d_i)$$

where $C = \{C_{ji}\} \rightarrow N * k$ matrix, with j^{th} code vector C_j as its j^{th} row.

$X \rightarrow$ input vector of dimension K.

$e = [e_0, e_1, \ldots, e_{N-1}]$

The above searching algorithm is a brute force approach where the distortion between the input vector and every entry in the code book is computed and is called fall-search vector quantization.

Every full search operation requires number of distortion computations " N" and each distortion computation involve "K" multiply - add operation.

Computing the index for one K-dimensional input vector requires "NK" multiply add operations and N-1 comparisons, without including memory access operation. This algorithm may be a bottleneck for high performance for large N. For such cases, a true-structured vector quantization scheme can be used whose complexity is proportional to $\log_2 N$.

Design 1:

Three vectors projection, processor and scheduling vectors are generated using HGACH FORdesigning systolic architecture for VG is shown in Table 5.26

Table 5.25 Design vectors and HUE generated by HGACH for vector quantization

Projection Vector	Processor Vector	Schedule Vector	HUE (%)
[1 0]	[0 1]	[1 1]	1

Dependence graph:

$D = [d_0\ d_1 \ldots d_{N-1}] = YX + e$ (5.35)

$Y = [y_{ji}]$ is NxK matrix with j^{th} code vector J_j as its j^{th} row.

X is the input vector

$e = [e_0\ e_1 \ldots e_{N-1}]$

From Equation 5.35 DG is derived, each node is the graph performs a multiplication and addition. The input vector X is propagated along the i_{th} direction and elements in the code vector Y_{ji} are preloaded to the (j, i) node.

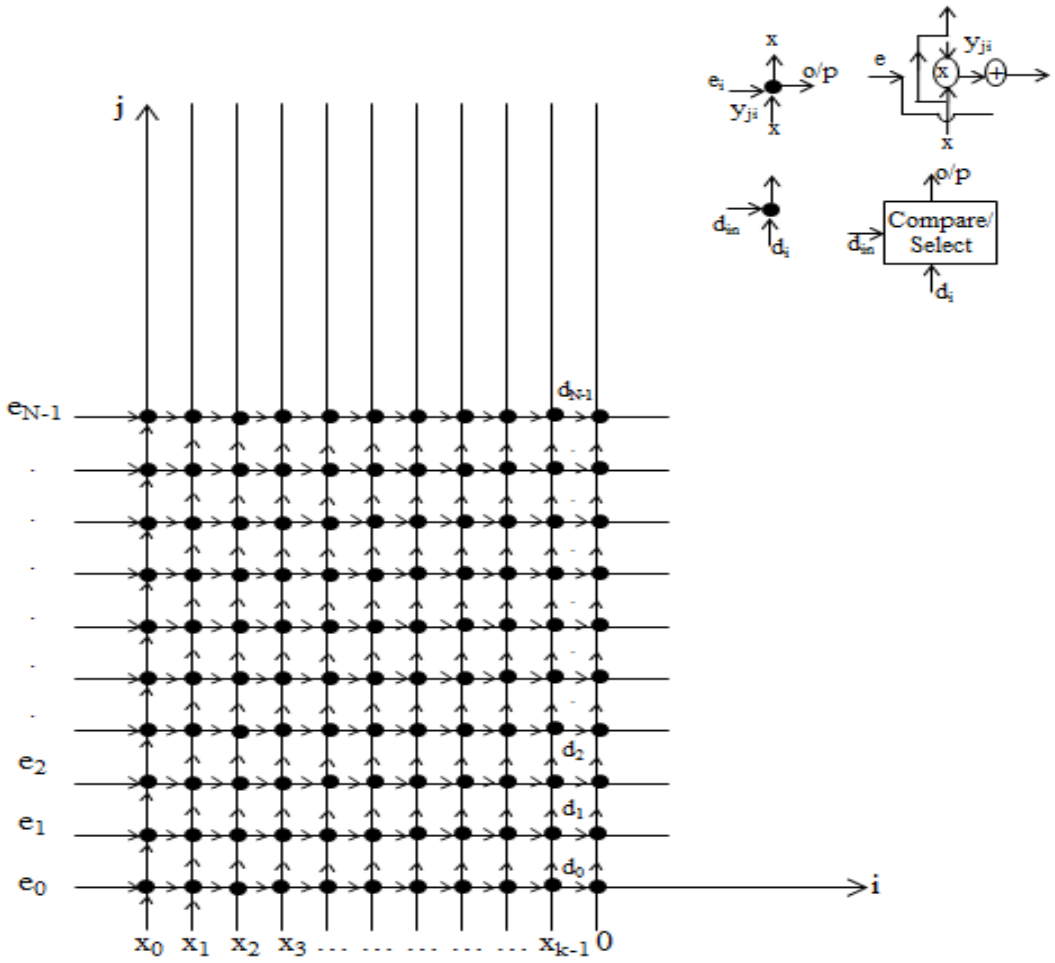


Figure 5.26 Dependence graph representation of VQ

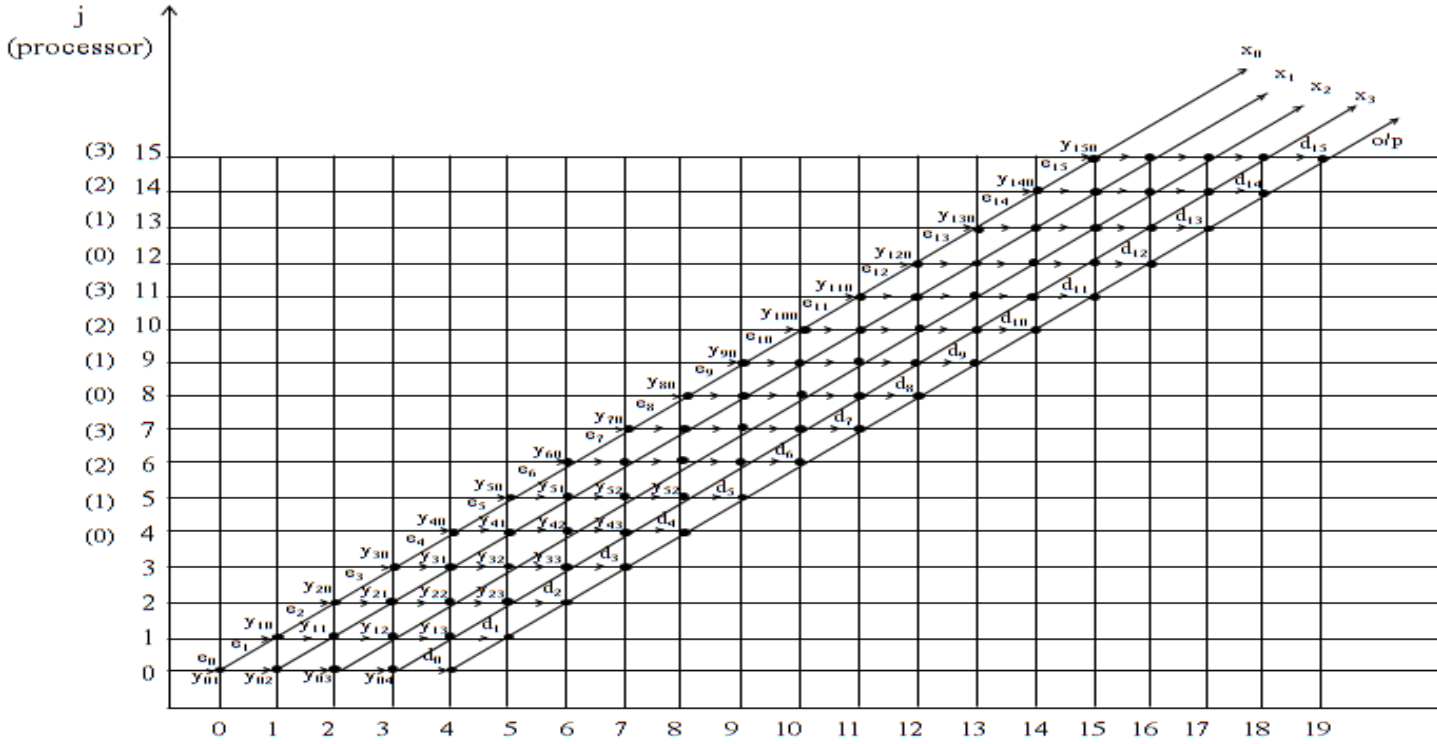


Table 5.26 Analysis of space-time diagram for vector quantizing

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P_0	“	x_1	x_2	x_3	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	0	0	0
P_1	0	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	0	0
P_2	0	0	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	0

P_3	0	0	0	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3	x_0	x_1	x_2	x_3
P_0	y_{00}	y_{01}	y_{02}	y_{03}	y_{40}	y_{41}	y_{42}	y_{43}	y_{80}	y_{81}	y_{82}	y_{83}	y_{120}	y_{121}	y_{122}	y_{123}			
P_1	0	y_{10}	y_{11}	y_{12}	y_{13}	y_{50}	y_{51}	y_{52}	y_{53}	y_{90}	y_{91}	y_{92}	y_{93}	y_{130}	y_{131}	y_{132}	y_{133}		
P_2	0	0	y_{20}	y_{21}	y_{22}	y_{23}	y_{60}	y_{61}	y_{62}	y_{63}	y_{100}	y_{101}	y_{102}	y_{103}	y_{140}	y_{141}	y_{142}	y_{143}	
P_3	0	0	0	y_{30}	y_{31}	y_{32}	y_{33}	y_{70}	y_{71}	y_{72}	y_{73}	y_{110}	y_{111}	y_{112}	y_{113}	y_{150}	y_{151}	y_{152}	y_{153}
P_0	e_0	e_0	e_0	e_0	e_4	e_4	e_4	e_4	e_8	e_8	e_8	e_8	e_{12}	e_{12}	e_{12}	e_{12}			
P_1	0	e_1	e_1	e_1	e_1	e_5	e_5	e_5	e_5	e_9	e_9	e_9	e_9	e_{13}	e_{13}	e_{13}	e_{13}		
P_2	0	0	e_2	e_2	e_2	e_2	e_6	e_6	e_6	e_6	e_{10}	e_{10}	e_{10}	e_{10}	e_{14}	e_{14}	e_{14}	e_{14}	
P_3	0	0	0	e_3	e_3	e_3	e_3	e_7	e_7	e_7	e_7	e_{11}	e_{11}	e_{11}	e_{11}	e_{15}	e_{15}	e_{15}	e_{15}

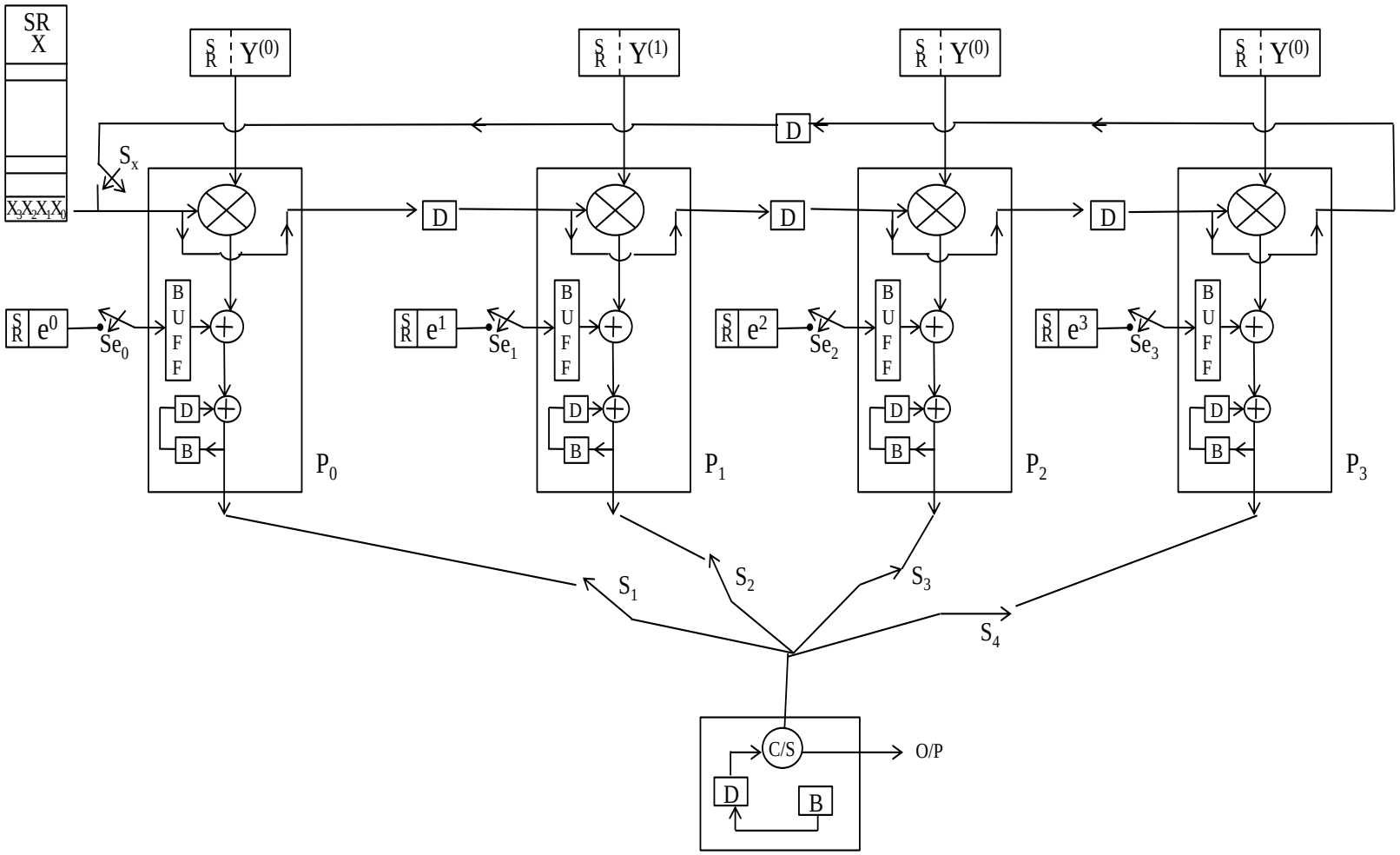


Figure 5.28 Architecture for vector quantization and its switching activity

Table 5.27 shows the processing at particular time unit for each PEs. As shown in time "0" Processing element "P_0" executes multiplication of "X_0" and "y_{00}" and it is added with "e_0".Four processing elements (P_0, P_1, P_2 P_3)are needed for completing the operation.

Architecture designed for vector quantization with k-4 & N=16 from space time diagram is shown in Figure 5.28. Four PEs are used for designing, "D" is a delay element typically a Flop, "B" is buffer in data flow path, "SR" is shift register for data inputs. "C/S" is comparator and selector for controlling the switching activity

The four switches (S_1, S_2, S3, S_4) controlled by comparator and selector for generating the output. In first connection switch "S_1" is connected at 3rd unit of time, "S_2" at 4th , "S_3" at 5th and "S_4" at 6th unit of time. Each switch is closed for one time unit and then opened. Every switching activity (connection) is repeated after every third unit of time after opening of corresponding switches.

The switches for "e" are "Se_0, Se_1 , Se_2 and Se_3". First connection for "Se_0 " is at 0th unit of time, "Se_1" at 1st, "Se_2 " at 2nd and "Se_3" at 3rd unit of time. Each switch is closed for one unit of time and then opened after every 3 units of time.

The switches used for controlling "X" inputs are "Sx_1 andSx_2". "Sx_1" is opened after 3 units of time and connected after 19th unit of time only. "Sx_s" is closed after 3 units of time and remain closed till 15th unit of time and then it is opened. Output is available from P_4 at 19th unit of time.

Design 2:

Second set of vectors projection, processor and scheduling vectors generated using HGACH for designing vector Quantization is shown in Table 5.28.

Table 5.27 Design vectors and HUE generated by HGACH for vector quantization

Projection Vector	Processor Vector	Schedule Vector	HUE (%)
[0 1]	[1 0]	[1 1]	1

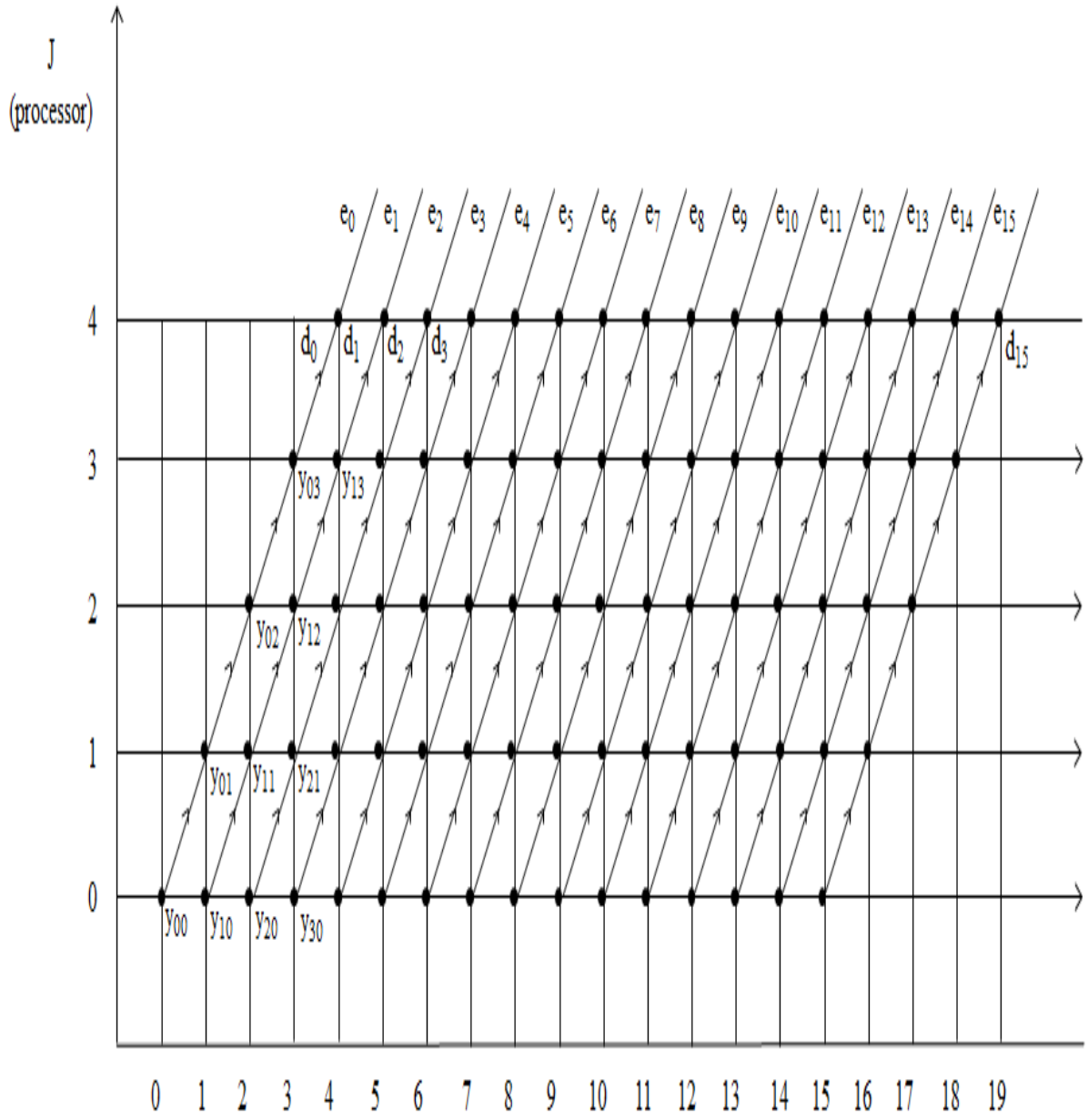


Figure 5.29 Spacediagram for vector quantizer- design 2

Table 5.28 Analysisof space-time diagram for vector quantizing design-2

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P_0	x_0	y_{00}	y_{10}	y_{20}	y_{30}	y_{40}	y_{50}	y_{60}	y_{70}	y_{80}	y_{90}	y_{100}	y_{110}	y_{120}	y_{130}	y_{140}	y_{150}			
P_1	x_1	0	y_{01}	y_{11}	y_{21}	y_{31}	y_{41}	y_{51}	y_{61}	y_{71}	y_{81}	y_{91}	y_{101}	y_{111}	y_{121}	y_{131}	y_{141}	y_{151}		
P_2	x_2	0	0	y_{02}	y_{12}	y_{22}	y_{32}	y_{42}	y_{52}	y_{62}	y_{72}	y_{82}	y_{92}	y_{102}	y_{112}	y_{122}	y_{132}	y_{142}	y_{152}	
P_3	x_3	0	0	0	y_{03}	y_{13}	y_{23}	y_{33}	y_{43}	y_{53}	y_{63}	y_{73}	y_{83}	y_{93}	y_{103}	y_{113}	y_{123}	y_{133}	y_{143}	y_{153}

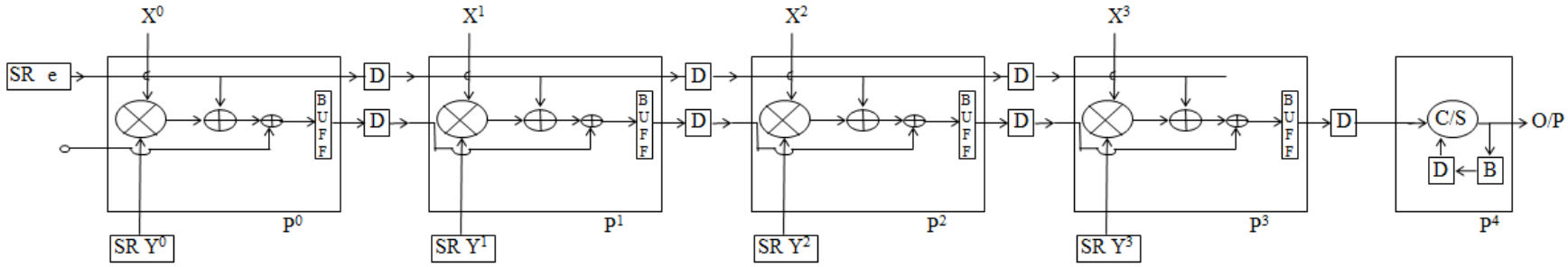


Figure 5.30 Architecture for vector quantization and its switching activity - design 2

Architecture designed for vector quantization using new set of vectors is shown in Figure 5.30.Four PEs is used and each PE has a multiplier and two adders. In design1 input "X" is passed serially and there are switches to control the operation. In design twoinputs "X" are given parallel to each PE. Input "e" is passed serially whereas in designs "e" is passed with the control of four switches.

5.6.4 Design of Systolic Array for Digital Correlator

The Digital Correlation operation is given by

$$Y_n = \sum_{K=0}^{N-1} x_K \, G_{n+k} \tag{5.36}$$

for n = 0, 1, 2, … N-1.

It can also be written in terms of matrix-vector multiplication, where the matrix is Toeplitz i.e. all the elements along the diagonal direction are the same.

For example four level digital correlator can be as in Equation 5.37. There are only 7 distinct elements in the A matrix.

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} G_3 & G_2 & G_1 & G_0 \\ G_4 & G_3 & G_2 & G_1 \\ G_5 & G_4 & G_3 & G_2 \\ G_6 & G_5 & G_4 & G_3 \end{bmatrix} \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} \tag{5.37}$$

Design 1 :

Three vectors projection, processor and scheduling vectors are generated using HGACH for designing systolic architecture for digital correlator is shown in Table 5.28

Table 5.29 Design vectors and HUE generated by HGACH for digital correlator

Projection Vector	Processor Vector	Schedule Vector	HUE (%)
[1 1]	[1 -1]	[1 1]	1

Space Time Diagram Analysis

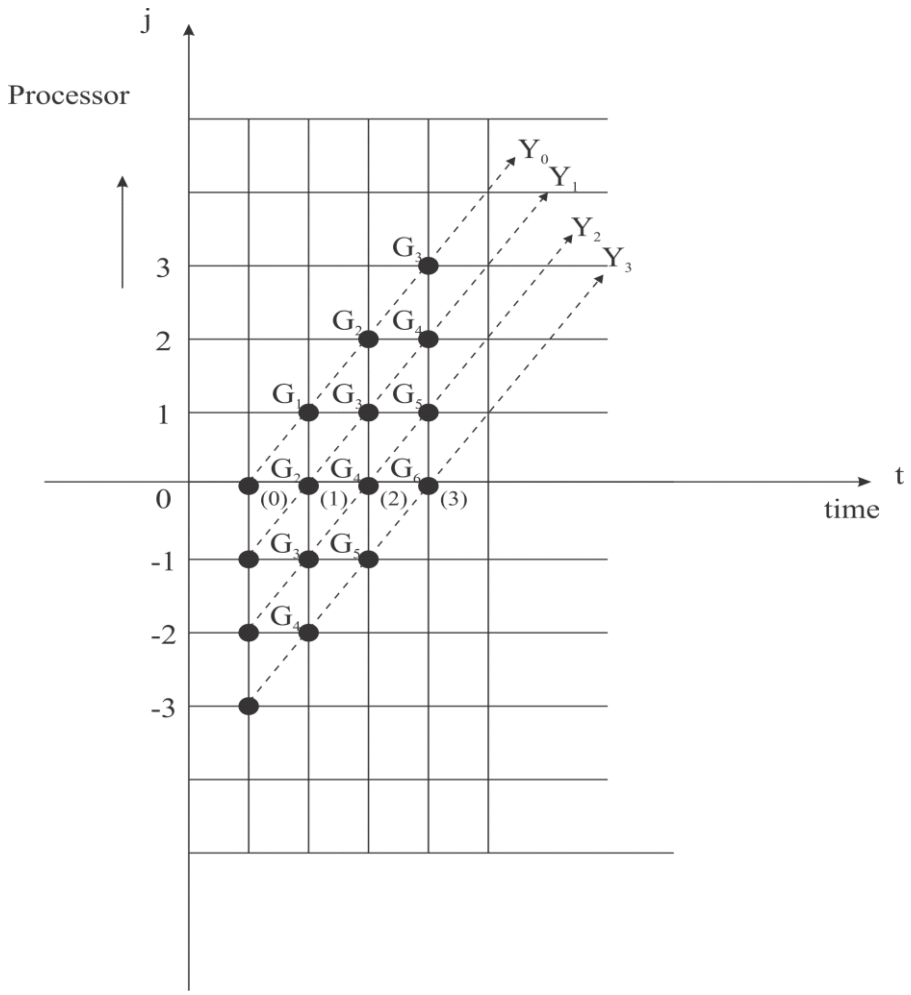


Figure 5.31 Space diagram for design-1 of digital correlator

Equation 5.38 shows the outputs for digital correlator. As Shown in Figure 5.31 at unit of time "0" PE_0 executes G_0X_0, PE_1 executes G_1X_0, PE_2 executes G_2X_0 and PE_s executes G_3X_0. Architecture designed for digital correlator-1 is shown in Figure 5.32. It has four PEs and each PE has two multipliers and two adders. "SR" represents shift registers for feeding "G" input serially to the multiplier.

$$Y_0 = G_3X_3 + G_2X_2 + G_1X_1 + G_0X_0$$
$$Y_1 = G_4X_3 + G_3X_2 + G_2X_1 + G_1X_0$$
$$Y_2 = G_6X_3 + G_4X_2 + G_3X_1 + G_2X_0$$
$$Y_3 = G_6X_3 + G_5X_2 + G_4X_1 + G_3X_0 \quad (5.38)$$

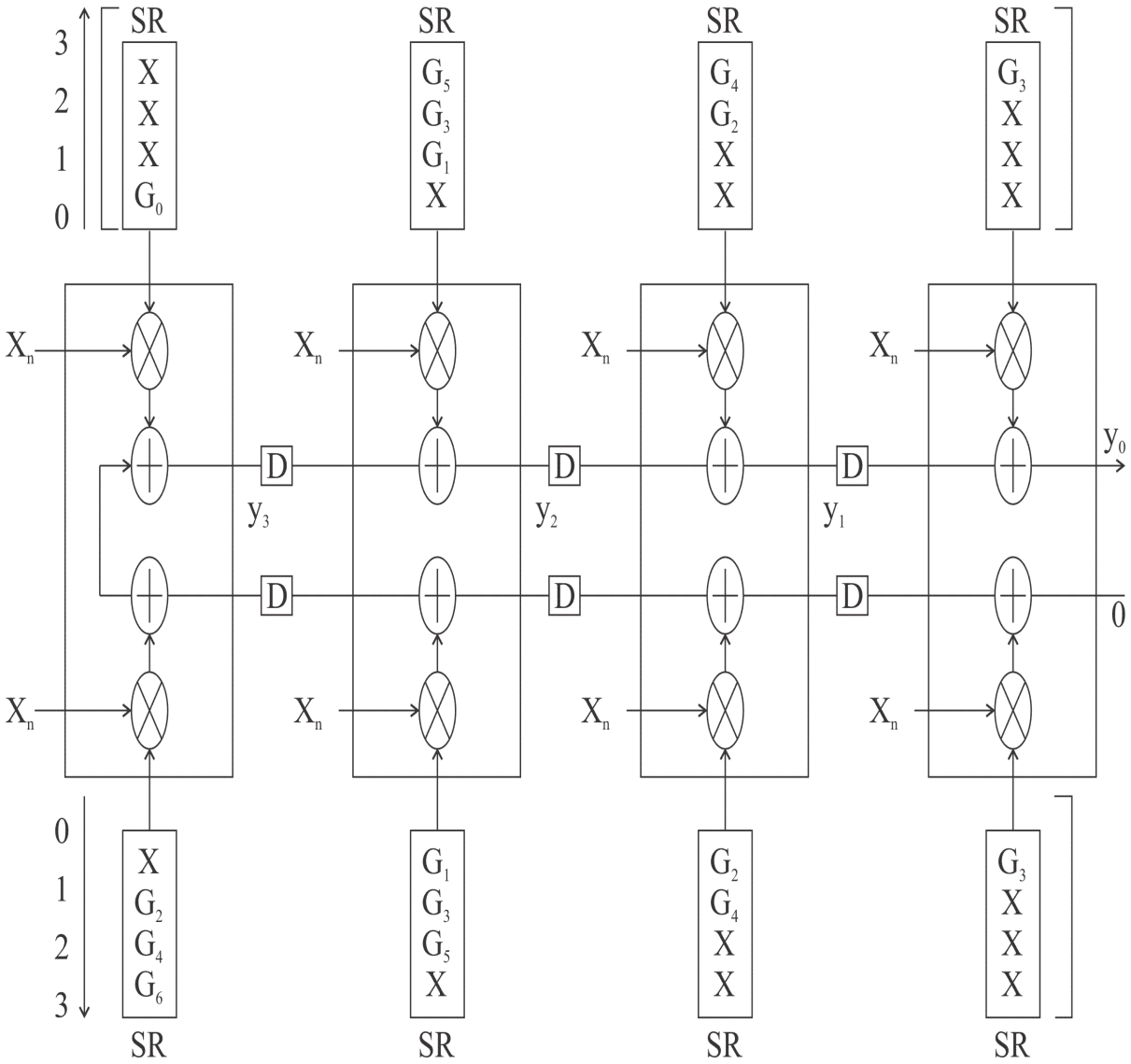
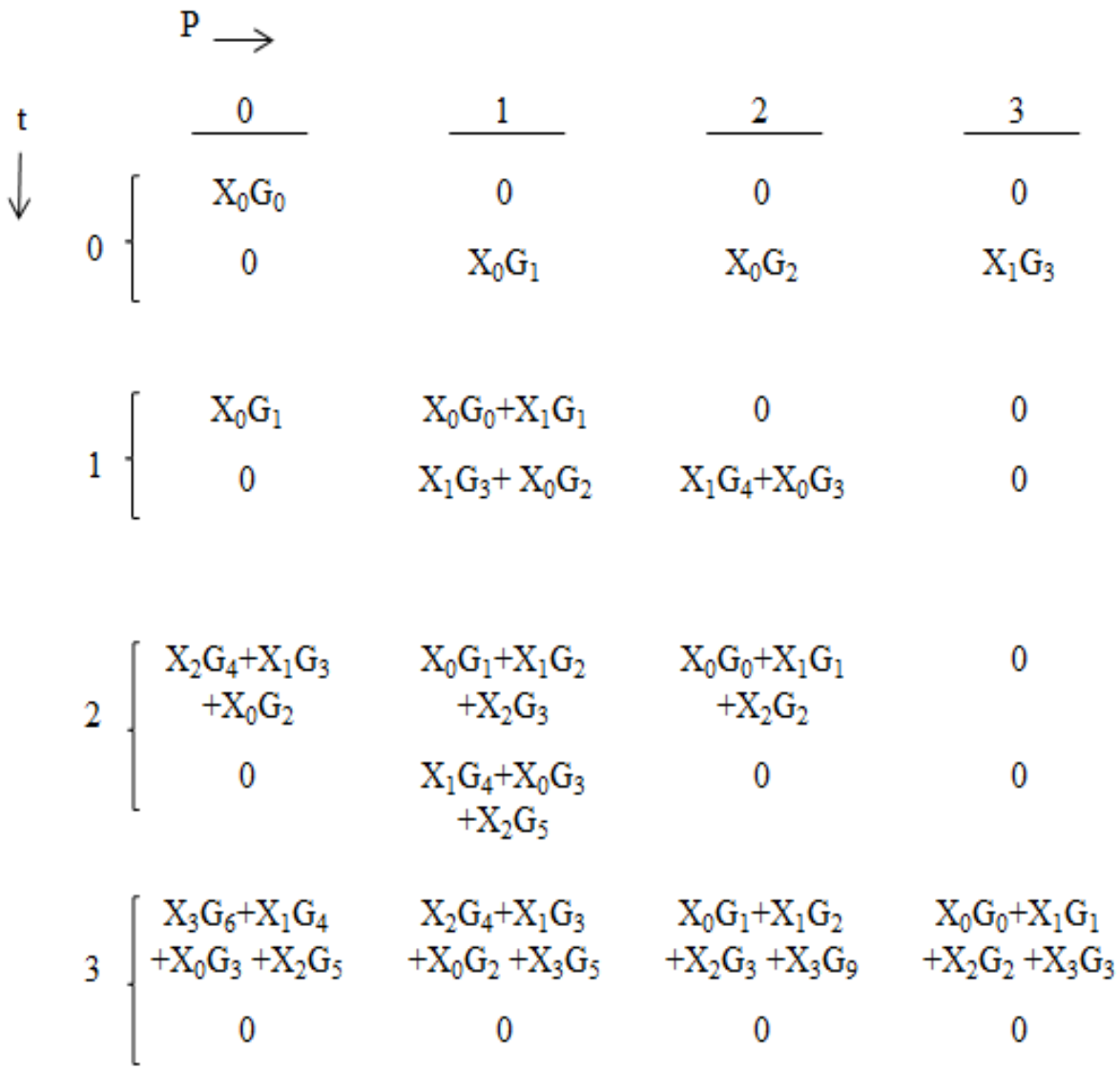


Figure 5.32 Architecture for design-1 of digital correlator

Analysis of architecture in Figure 5.32 is shown below.At time "0" only multiplier in PEs is used.



Design 2

Three vectors projection, processor and scheduling vectors are generated using HGACH for designing systolic architecture for digital correlator is shown in Table 5.29 and its corresponding space time diagram is shown in Figure 5.33 and the architecture designed is in Figure-5.34

Table 5.30 Design vectors and HUE generated by HGACH for digital correlator

Projection Vector	Processor Vector	Schedule Vector	HUE (%)
[1 1]	[1 -1]	[0 1]	1

Space Time Diagram

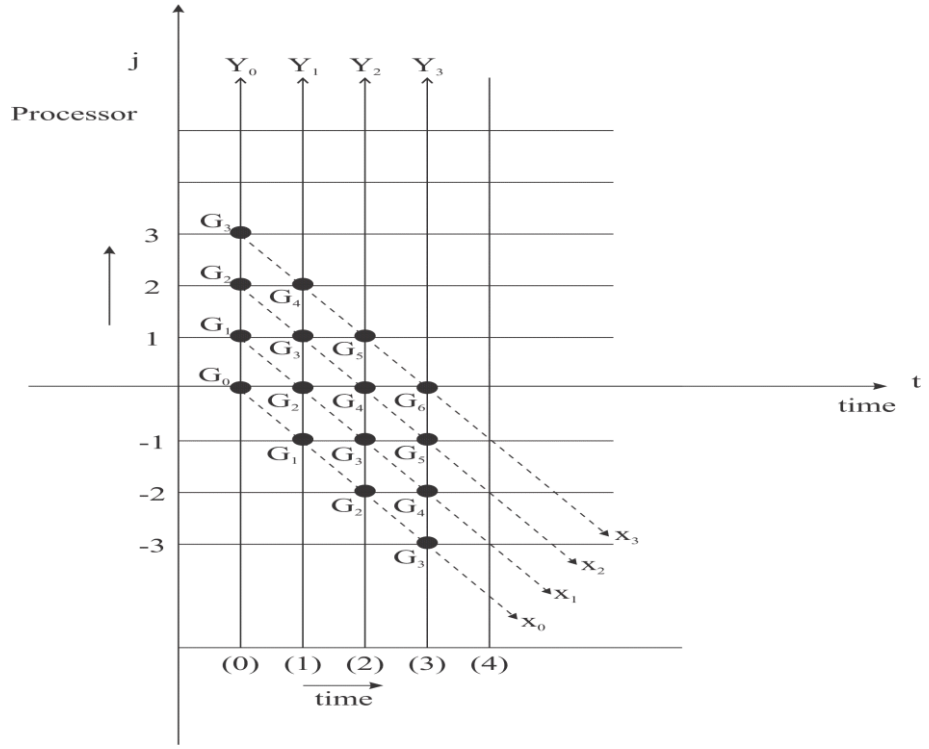


Figure 5.33 Spacediagram for design-2 of digital correlator

Low Level Diagram

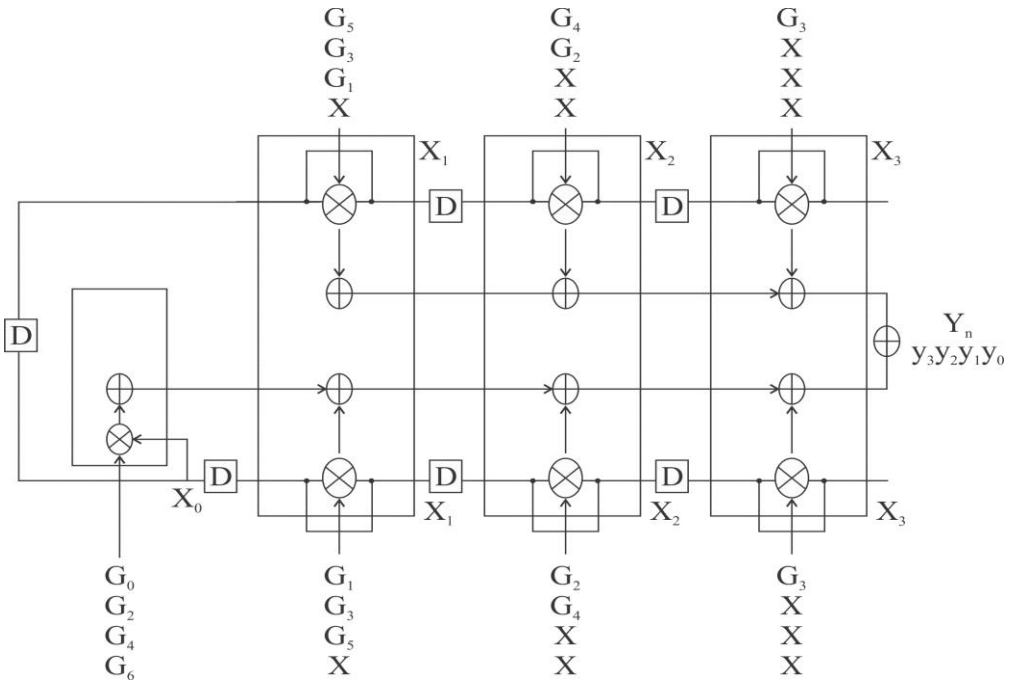


Figure 5.34 Architecturefor design-1 of digital correlator

5.7 SUMMARY

Natural computing systems have enormous capabilities but mathematical model to represents natural phenomenon are equally difficult. A major challenge associated with this is, designing an algorithm to deliver the optimal solution so that architecture efficiency could takes places in proper way. In this regard a hybrid approach between genetic algorithm and chaotic optimization has developed in this paper. As in case of natural system integrity of modules is much more robust concept to develop the optimal solution. Rather than having one solution, a set of solutions has developed. In comparison to single solution which may impose the constraint with surrounding, the proposed solution gives better performance to satisfy the surrounding as well desired objectives. Parametric quality evaluation of systolic architecture is a challenging task and mathematical model for hardware utilizing efficiency has been applied and associated number of delays also minimizes to reduce cost without compromising the quality. Various architecture for matrix multiplication under two different possibilities of systolic architecture called partial pipelined and fully pipelined have developed. Also systolic architecture is designed for convolution, vector quantizer and digital correlator using the vectors obtained using HGACH method. It is possible to apply the same approach for other algorithm to design number of different and efficient systolic array.

This chapter presented the concept and design of how different possible vectors with very high performance characteristics can be achieved automatically It is hope that, proposed method will provide a very effective means to design the optimal systolic architectures having high efficiency and minimum delay elements for different algorithms.

CHAPTER 6

CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSIONS

In this research designer requirement to develop the computing architecture compactable to surrounding environment to achieve the objectives in optimum manner has been taken care using computational intelligence.This research was needed to help the designer instead of opting the approach of trial and error method to find the compactible solution which is not only very time consuming process but also in most of the case suboptimal solution accepted at the end. Hence the need to find the automated approach which could deliver the number of possible solutions quickly and efficiently with best possibilitieswas the objectives. This objective has fulfilled by use of computational intelligence for exploration of solution space. In result, to develop an efficient automated solution environment for designing efficient systolic architecture which using nature inspired Computations like, Genetic algorithm, Evolutionary Programming, Swarm intelligence and chaotic optimization have considered to our solution automation.

Literature review has been done in detail to get the current status of research in the design of systolic array which has given possibilities to analyse systolic array from different dimensions. Detail discussion along with applications of evolutionary computation has presented which has given support to fit the systolic array design problem in domain of natural computing.

Regular iterative algorithm description of dependence graph have considered for finding the optimal values for fundamental design to design efficient architecture along with that maximum hardware utilizing efficiency. Approach is also having the focus to minimize the total number of delay involved with systolic architecture design. A mathematical formulation has done to consider as fitness

function for exploration of heuristic method which could defined as estimation of vectors as problem of constraint optimization, where objectives are to maximize hard ware utilising efficiency and minimize the total delay associated with each edge of processing element. Both objectives have formulated to form a single objective problem in form of problem of minimization.

Evolutionary programming, which is a very efficient heuristic under evolutionary computation and work for real value domain, a discrete version of evolutionary programming has developed to find the several optimal design vectors set for matrix multiplication systolic architecture. In evolutionary programming where mutation is fundamental operator to develop the offspring, a hybrid approach which is based on Gaussian and Cauchy distribution have developed. Another very successful swarm intelligence based concept, Particle swarm optimization have applied in discrete form to find the optimal solution and comparative performance analysis has made and it is observed that hybrid evolutionary programming has more efficient in development of solution .

A automated method to estimate and explore the systolic array design vectors using help of hybridization of genetic algorithm and chaos based chaotic optimization is proposed. This hybridization has applied to achieve the global solution in faster manner. Number of different systolic architecture which have the characteristics of partial pipeline or fully pipelined can be easily developed with proposed solution. Characteristics of chaotic map non periodicity and initial condition sensitivity deliver the possibility of more robust search especially in local region. Hence after applying the cross-over and mutation operation in genetic algorithm local region is further searched by chaotic optimization. To get the more detail study of proposed solution, numerical benchmark problems have tested and outcomes are appreciable. With proposed solution different architecture of systolic array corresponding to various computational intensive problems like matrix multiplication, digital correlator, vector quantizor have proposed.

Various architecture for different algorithms which are heavily applied in the area of signal processing have implemented over hard ware platform to understand the physical usefulness of proposed solution.

6.2 SCOPE FOR FUTURE WORK

Every research has opened the path for further exploration and this research is no exception.Fundamental restriction of this research was not to have user friendly compiler which could transform the proposed solution in modular representation quickly and in attractive manner. This is a challenge for further researchers to think about this.

Even though evolutionary computation based solution have designed efficiently in this research but as it obvious further possibilities to explore the domain of natural computing to have better solution is justified. There is further possibility to find out all solution of systolic array corresponding to an algorithm in a single trial of evolutionary computation rather than repeating the trials to obtain the different architecture. This is possible with applying the concept of distance based subpopulation creation under a population in a run, where each subpopulation will responsible to find out a unique solution.

It is hope that proposed concept of designing the systolic architecture will not only provide the better and efficient architecture but it also fulfilled the requirement of customized design requirement cost effectively.

REFERENCES

1. Abed, KH & Siferd, R 2000, 'CMOS VLSI implementation of 16-bit logarithm and antilogarithm converters', Proceedings of the IEEE Midwest Symposium on Circuits and Systems, vol.2, pp.776–779.

2. Abellard, A & Abellard, P 2008, 'A design methodology of systolic architectures based on a petri net extension: application to a stereovision hardware/software processing improvement', Software Engineering Advances, ICSEA '08. The Third International Conference, pp.77-82.

3. Adedeji, AA 2007, 'Genetic (evolutionary) algorithm: introduction and its use as an engineering design tool', Department of Civil Engineering, University of Ilorin, pp.978-811.

4. Arunachalam, S, Khairnar, SM & Desale, BS 2013, 'Application of fast fourier transform (FFT), Algorithm in Finite Impulse Respo NSE (FIR) Linear Filtering using MATLAB', IJERT, vol.2, no.11.

5. Asgari, H ,Kavian, YS & Mahani, A 2013, 'A systolic architecture for Hopfield neural networks'.

6. Azarderakhsh, R & Kermani, MM 2015, 'High-performance two-dimensional finite field multiplication and exponentiation for cryptographic applications', Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, vol.5,pp.1–1.

7. Back, T, Hammel, U & Schwefel, HP 1997, 'Evolutionary computation: comments on the history and current state', IEEE Trans. Evolutionary Computation, vol.1, no.1, pp.3-17.

8. Banaiyan A , Esmaeelzadeh, H & Saafari, S 2006, 'Co evolutionary scheduling and mapping for high level test synthesis', IEEE ICEIS' 06, Islamabad, Pakistan.

9. Bekakos, Milovanovic´, TI, Tokic´ Dolic´anin , CB & Milovanovic, EI 2011, 'Selecting mathematical method for systolic processing', SER. A: APPL. MATH. INFORM. AND MECH, vol.3, pp.53-58.

10. Beyer, HG & Schwefel, HP 2002, 'Evolution strategies–a comprehensive introduction, Natural Computing', vol.1, pp.3-52.

11. Bezdek, JC 1994, ‘What is computational intelligence?Brazdil, BP 1993, ‘Machine learning: ECML-93’, European conference on machine learning, Proceedings, Springer Science & Business Media,,Computational Intelligence Imitating Life, IEEE Press, New York.

12. Bhanu, UN & Chilambuchelvan, A 2014, ‘High-speed systolic VLSI architecture for 2-D forward lifting-based DWT’, Arabian Journal for Science and Engineering, vol.39, no.8, pp.6125-6135.

13. Brazdil, BP 1993, ‘Machine learning and knowledge discovery databaseBrazdil, BP 1993, ‘Machine learning: ECML-93’, European conference on machine learning, Proceedings, Springer Science & Business Media,ECML-93’, European conference on machine learning, Proceedings, Springer Science & Business Media.

14. Bryant, K & Benjamin, A 2000, ‘Genetic algorithms and the traveling salesman problem’, Proceedings of 1st GNT Regional Conference on Mathematics, Statistics and Applications.

15. Buzdalov, M, Buzdalova, A & Shalyto, A 2013, ‘First step towards the runtime analysis of evolutionary algorithm adjusted with reinforcement learning’, Machine Learning and Applications (ICMLA), 12th International Conference, vol.1, pp.203–208.

16. Cappello, P 1992, ‘A processor-time minimal systolic array for cubical mesh algorithms’, IEEE Transactions on Parallel Distributed Systems, vol.3, no.1, pp.4-13.

17. Chia-Hsiang, Y , Chun-Wei, C , Chia-Shen, H & Chiao-En, C 2015, ‘A systolic array based gtd processor with a parallel algorithm’, Circuits and Systems I: Regular Papers, IEEE Transactions, vol.62, no.4, pp.1099–1108.

18. Chipperfield, A & Fleming, P 2002, ‘An overview of evolutionary algorithms for control systems engineering’, Evolutionary Computation Research.

19. Chuanpeng Chen & Zhongping Qin 2009, ‘A systolic architecture with linear space complexity for longest common subsequence problem’, ASIC, ASICON, IEEE 8th International Conference.

20. Coello, C 2000, ‘An updated survey of GA-based multiobjective optimization techniques’, ACM Computing Surveys, vol.32, no.2, pp.109-143.

21. Crnkovic, I & Larsson, M 2002, ‘Building reliable component based software systems’, Artech House Inc., Norwood, MA, USA.

22. Crocker, M , Sharon Hu , X & Niemier, M 2010, ‘Design and comparison of NML systolic architectures’, Nanoarch '10, Proceedings of the 2010 IEEE/ACM International Symposium on Nanoscale Architectures,pp.29-34.

23. Crowley, MA 2005, ‘Computational intelligence and knowled', vol. 5.

24. Davidor, Y 1991, ‘Genetic algorithms and robotics: a heuristic strategy for the optimization’, vol.6.

25. Davis, RHD 1984, ‘Systolic array chip matches the pace of high speed processing’, Electronic Design.

26. Davis 1989, ‘Adapting operator probabilities in genetic algorithms’, Available from : http://citeseerx.ist.psu.edu/viewdoc/summary.

27. De, M , De, S & Bhattacharya, AB 2008, ‘Parallel architecture and algorithms for space weather prediction–a review’, Indian Journal of Radio and Space Physics, vol.37, no., pp.157-173.

28. Deo, N 2014, ‘Parallel computers in signal processing', Defence Science Journal, vol.35,pp.375-382.

29. Dianati, M , Song, I &Treiber, M 2002, ‘An introduction to genetic algorithms and evolution strategies’, Technical Report, University of Waterloo, Ontario, N2L 3G1, Canada.

30. Ekstrand, F, Lidholm, J & Asplund, L 2008, ‘Robotics for SMES- 3D vision in real-time for navigation and object recognition’, in: 39th International Symposium on Robotics, ISR, pp.70–75.

31. Erlei, Z, Xiangrong, Z, Hongying, L & Licheng, J 2015, ‘Fast multifeature joint sparse representation for hyperspectral image classification’, Geoscience and Remote Sensing Letters, IEEE, vol.12, no.7, pp.1397–1401.

32. Eshelman, S 1991, ‘On crossover as an evolutionarily viable strategy’, Available from : http://citeseerx.ist.psu. edu/viewdoc/ summary.

33. FarlandMC , Parker, AC & Camposano, R 1990, ‘The high level synthesis of digital systems’, Proc. of the IEEE.

34. FarlandMC , Parker, AC & Camposano, R 1990, 'The high level synthesis of digital systems', Procedure of the IEEE.

35. FengChen, Xinxin Sun, Dali Wei & Yongning Tang 2011, 'Tradeoff strategy between exploration and exploitation for PSO', Natural Computation Seventh International Conference, vol.3, pp.1216–1222.

36. Fernandez-Martinez, JL & Garcia-Gonzalo, E 2011, 'Stochastic stability analysis of the linear continuous and discrete pso models', Evolutionary Computation, IEEE Transactions, vol.15, no.3, pp.405–423.

37. Fisher AL , HT 1985, 'Special purpose VLSI architectures: general discussions and a case study',VLSI and Modern Signal Proccessing, Prentice-Hall Inc.

38. Fisher AL , HT 1985, 'Synchronizing large VLSI processor arrays', IEEE Transactions on Computers, vol.100, no.8, pp.734-740.

39. Fogel, DB 1994, 'An introduction to simulated evolutionary optimization', IEEE Trans. Neural Networks, vol.5, no.1, pp.3-14.

40. Fogel, DB 2006, 'Evolutionary computation: toward a new philosophy of machine intelligence', IEEE Press Series on Computational Intelligence, John Wiley & Sons, vol.1,pp.384.

41. Forte, G, Espinosa-Duran, JM & Velasco-Medina, J 2013, 'Systolic architectures to evaluate polynomials of degree n using the Horner's rule', Circuits and Systems (LASCAS), IEEE Fourth Latin American Symposium, pp.1-4.

42. Foster MJ , HT 1980, 'The design of special-purpose VLSI chips', computer, vol.13, no.1, pp.26-40.

43. Garis 1990, 'Genetic programming: modular evolution for darwin machines', Available from : http://citeseerx.ist.psu.edu/viewdoc/summary? doi=10.1.1.1.

44. Gentleman, WM & Kung, HT 1982, 'Matrix triangularization by systolic arrays',25th Annual Technical Symposium, International Society for Optics and Photonics,pp.19-26.

45. Goldberg & Deb et al 1991, 'Don't worry, be messy', Available from : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.378 [July 2015].

46. Grefenstette 1989, ‘A system for learning control strategies with genetic algorithms’, Available from : http://citeseerx.ist.psu.edu/viewdoc/summary.

47. GuoLiangChen, GuangZhong Sun, Yun Xu & Bai Long 2011, ‘Integrated research of parallel computing: Status and future’, Springer,Chinese Science Bulletin, vol54, no.11, pp.1845-1853.

48. Hajar Asgari , Yousef, S, Kavian & Ali Mahani 2014, ‘A systolic architecture for hopfield neural networks’, Elsevier, Procedia Technology, vol.17, pp.736–741.

49. Hegen Xiong, Kai Xiong & Qiuhua Tang 2009, ‘A novel variable-boundary-coded quantum genetic algorithm for function optimization’, Dependable, Autonomic and Secure Computing, Eighth IEEE International Conference,pp.279-285.

50. Helms, HD 1967, ‘Fast fourier transform method of computing difference equations and simulating filters’, IEEE Trans. Audio Electroacoust, pp.85-90.

51. Himani&Sidhu, HS 2013, ‘Design and implementation modified booth algorithm and systolic multiplier using FPGA’, International Journal of Engineering Research & Technology, vol.2, no.11.

52. Holland, J 1975,’ Adaptation in natural and artificial systems’, Ann Arbor, The University of Michigan Press.

53. Hu, J 2004, ‘Sustainable evolutionary algorithms and scalable evolutionary synthesis of dynamic systems’, Doctoral Dissertation, Michigan State University.

54. Jha, CK, Suresh, M & Panda, AK 2009, ‘The IUP Journal of Telecommunications’, Science & Technology.

55. Jiang Feng-Guo 2010, ‘The hybrid genetic algorithm based on the niche’s technology’, Control Conference (CCC).

56. Jinfeng Zhao 2011,‘Chaotic particle swarm optimization algorithm based on tent mapping for dynamic origin-destination matrix estimation’, Electric Information and Control Engineering (ICEICE), pp.221–224.

57. Johnson, KT, Hurson, AR & Shirazi, B 1993, 'General-purpose systolic array's', computer, vol.26, no.11, pp.20-31.

58. Johnston, CT, Gribbon, KT & Bailey, DG 2004, 'Implementing image processing algorithms on FPGAs', Proceedings of the Eleventh Electronics, New Zealand Conference, Palmerston North, pp.118–123.

59. Jones, G 1998, 'Genetic and evolutionary algorithms', Encyclopedia of Computational Chemistry.

60. Julien, A, Vijverberg, Peter, HN & De With 2010, 'Two-dimensional systolic-array architecture for pixel-level vision tasks', Procedure Real-Time Image and Video Processing.

61. KeHou , Jing Zhang & Xing Fang 2015, 'Research on the Architecture of Data-Intensive Computing Platform', Journal of Software Engineering,pp.686-701.

62. Kevin SH Ong ,Suhaib A Fahmy & Keck-Voon Ling 2014, 'A scalable and compact systolic architecture for linear solvers', IEEE 25th International Conference on Application-specific Systems, Architectures and Processors (ASAP).

63. Koza 1991, 'Evolving a computer program to generate random numbers using the genetic programming paradigm', Available from : http://citeseerx. ist.psu. edu/viewdoc/summary?doi=10.1.1.18.378.

64. Krzysztof Lichy 2012, 'Proposal of new systolic architecture for mathematical morphology algorithms', Springer,Towards Modern Collaborative Knowledge Sharing Systems Studies in Computational Intelligence, pp.119-132.

65. Kung, SY 1984, 'On supercomputing with systolic/wavefront array proccessors', Proceedings of IEEE, vol.72, no.7.

66. Kung, SY 1985, 'VLSI Array Processors', IEEE ASSP Magazine, vol.2, no.3, pp.4-22.

67. Kung HT, Sproull, R, Steele, G 2012, 'VLSI systems and computations', Springer Science & Business Media, pp.415.

68. Lee, CC 1990, 'Fuzzy logic in control systems: fuzzy logic controllers – parts I, II', IEEE Transactions on Systems, Man, and Cybernetics, vol.20,pp.404-435.

69. Lee, PZ & Kedem, ZM 1988, 'Synthesizing linear array algorithms from nested for loop algorithms', IEEE Transactions on Computers, vol.37, pp.1578–1598.

70. Lewis FW, Jagannathan, S &Yesildirak, A 1998, 'Neural network control of robot manipulators and non-linear systems'.

71. Liang, X & Jean, J 2003, 'Mapping of generalized template matching onto reconfigurable computers', IEEE Transaction on Very Large Scale Integration (VLSI) Systems, vol.11, no.3, pp.485-498.

72. Liang, Lu, Liu, W, O'Neill, M & Swartzlander, EE 2010, 'QCA systolic matrix multiplier', IEEE Annual Symposium on VLSI, pp.149-154.

73. Liang, Lu, Liu, W, O'Neill, M & Swartzlander EE 2010, 'QCA systolic matrix multiplier', ISVLSI, IEEE Annual Symposium on VLSI, pp.149-154.

74. Liang, Lu , Liu, W , O'Neill, M &Swartzlander EE 2011, 'QCA systolic array design', IEEE Transactions on Computers,pp.14.

75. Lin H , Ma, X, Feng, W & Samatova, NF 2011, 'Coordinating computation and I/O in massively parallel sequence search', IEEE Transactions on Parallel & Distributed Systems, vol.22, no.4, pp.529-543.

76. Li, P, Xu, D, Zhou, Z & Lee, WJ 2013,'Stochastic optimal operation of microgrid based on chaotic binary particle swarm optimization', Smart Grid, IEEE Transactions.

77. Lopez-Parrado, A & Velasco-Medina, J 2012, 'Efficient systolic architecture for Hermitian eigenvalue problem', Circuits and Systems (CWCAS), IEEE 4th Colombian Workshop, pp.1–6.

78. Mamatha, I, Raj, JN, Tripathi, S & Sudarshan 2015, 'Systolic architecture implementation of 1D DFT and 1D DCT', Signal Processing, Informatics, Communication and Energy Systems (SPICES), IEEE International Conference,pp.1–5.

79. Mclaren, DJ 2003, 'Improved Mitchell-based logarithmic multiplier for low-power DSP applications', SOC Conference, Proceedings of the IEEE International Systems-on-Chip,pp.53–56.

80. Mehta, S & Midhuna, R 2014, 'Hybridization of neural network using genetic algorithm for heart disease detection', International Journal of Application or Innovation in Engineering & Management (IJAIEM), vol.3, no.1.

81. Mertzios, BG 1995, 'Implementation of matrix decomposition structures of 2-D digital filters via VLSI array processors', Circuits, Systems and Signal Processing, vol.14, no.1, pp.39-55.

82. Miller, WT, Sutton, RS & Werbos, PJ 1990, 'Neural networks for control'.

83. Mohammad, AK, Mohammad, AK & Abdul, QA 2012, 'Handbook of research on industrial informatics and manufacturing intelligence: innovations and solutions', IGI Publishing Hershey, PA, USA.

84. Njini, I & Ekabua, OO 2014, 'Genetic algorithm based energy efficient optimization strategies in wireless sensor networks', ACSIJ, vol.3, no.5.

85. Obayashi, Shigeru, Sasaki, D, Takeguchi, Y & Hirose, N 2000, 'Multiobjective evolutionary computation for supersonic wing-shape optimization', IEEE Transactions on Evolutionary Computation, vol.4, no.2, pp.182-187.

86. Okamoto, T & Hirata, H 2010, 'Global optimization using a multi-point type quasi-chaotic optimization method with the simultaneous perturbation gradient approximation', Systems Man and Cybernetics (SMC), IEEE International Conference, pp.804–809.

87. Parhi, KK 1999, 'VLSI digital signal processing systems', vol.6.

88. Parhi, KK 2007, 'VLSI digital signal processing systems: design and implementation', John Wiley & Sons, pp.808.

89. Pasca, BM 2011, 'High-performance floating-point computing on reconfigurable circuits', Doctoral dissertation, Ecolenormalesupérieure de lyon-ENS LYON.

90. Pedro, G, Coelho João B Cardoso , Paulo R Fernandes & Hélder C Rodrigues 2011, 'Parallel computing techniques applied to the simultaneous design of structure and material', Elsevier, Advances in Engineering Software, vol.42, no.5, pp.219–227.

91. Porto, V, Fogel, D & Fogel, L 1995, 'Alternative neural network training methods', IEEE Expert, vol.10, no.3, pp.16-22.

92. Rechenberg 1973, ‘Evolutions strategies: optimieru ngtechni schersystem enachprinzipien der biologischen’, Available from : http://citeseerx. ist.psu. edu / viewdoc/ summary?doi=10.1.1.18.378.

93. Renteria-Mejia, CP, Trujillo-Olaya, V & Velasco-Medina, J 2012, ‘Design of an 8192-bit RSA cryptoprocessor based on systolic architecture’, Programmable Logic (SPL), VIII Southern Conference, pp.1–6.

94. Riedl, A 1998, ‘A versatile genetic algorithm for network planning’, Proceedings of EUNICE, vol.98,pp.97-103.

95. Rowe, A, Rosenberg, C & Nourbakhsh, I 2005, ‘A second generation low cost embedded color vision system’,Computer Vision and Pattern Recognition - Workshops, IEEE Computer Society Conference on CVPR Workshops, pp.136.

96. Saha, D, Banerjee, S & Jana, ND 2015, ‘Multi-objective particle swarm optimization based on adaptive mutation', Computer, Communicatio’n, Control and Information Technology (C3IT), pp.1–5.

97. Salih, MH & Arshad, MR 2010,’ Embedded parallel systolic architecture for multi-filtering techniques using FPGA’, IEEE Electronic Computer Technology (ICECT) International Conference, pp.122-127.

98. Sato, S, Otori, K, Takizawa, A, Sakai, H, Ando, Y & Kawamura, H 2002, ‘Applying genetic algorithms to the optimum design of a concert hall’, Journal of Sound and Vibration, vol.258, no.3, pp.517-526.

99. Selvakumar, R et al 2015, ‘Design exploration for systolic architecture using hybridization of GA and local chaotic optimization’, International Journal of Applied Engineering Research, ISSN:0973-4562, vol.10, no.3, pp.7009-7031.

100. Sharifi, A, Noroozi, V , Bashiri, M , Hashemi, AB & Meybodi, MR 2012, ‘Two phased cellular PSO: a new collaborative cellular algorithm for optimization in dynamic environments’, Evolutionary Computation (CEC), IEEE Congress,pp.1–8.

101. Smith 1983, ‘Flexible learning of problem solving heuristics through adaptive search’, Available from : http://citeseerx.ist.psu.edu/viewdoc/summary? doi=10.1.1.18.378.

102. Soudan, B & Saad, M 2008, ‘An evolutionary dynamic population size pso implementation’, Information and Communication Technologies: From Theory to Applications, 3rd International Conference, pp.1–5.

103. Spears, WM, De Jong , KA, Bäck, T, Fogel, DB & De Garis, H 1993, 'An overview of evolutionary computation', in: Machine Learning: ECML-93, pp.442-459.

104. Stephen, J, Das, VV & Thankachan, N 2010, 'Power electronics and instrumentation engineering, Proceedings', Communications in Computer and Information Science, International Conference, PEIE,pp.114.

105. Stockham, TG 1966, 'High speed convolution and correlation in AFIPS', Conf. Proc., Spring Joint Computer Conference, vol.28, pp.229-233.

106. Sudha, N, Mohan, AR & Meher, PK 2011, 'A self-configurable systolic architecture for face recognition system based on principal component neural network', Circuits and Systems for Video Technology, IEEE Transactions, vol.21, no.8.

107. SwartzlanderE, JR 2012, 'Application specific processors, The Springer International Series in Engineering and Computer Science', Springer Science & Business Media.

108. Tang, KS , Man, KF, Kwong, S & He, Q 1996, 'Genetic algorithms and their applications', IEEE Signal Processing Magazine, vol.13, no.6, pp.22-37.

109. Togelius, J, De Nardi , R & Moraglio, A 2008, 'Geometric PSO + GP = particle swarm programming', Evolutionary Computation, IEEE World Congress on Computational Intelligence, IEEE Congress, pp.3594-3600.

110. Toshiaki 2009, 'Low-complexity systolic v-blast architecture', Wireless Communications, IEEE Transactions, vol.8 , no.5.

111. Tselepis, IN, Bekakos, MP, Milovanović, IŽ & Milovanović, EI 2007, 'FPGA implementation of optimal planar systolic arrays for orthogonal matrix multiplication', SETIT.

112. Tsujimoto, T, Shindo, T & Jin'no, K 2011, 'The neighborhood of canonical deterministic PSO', Evolutionary Computation (CEC), IEEE Congress, pp.1811–1817.

113. Whitley & Starkweather 1989, 'Scheduling problems and traveling salesmen: the genetic edge recombination operator', Available from : http://citeseerx. ist. psu.edu/viewdoc/summary?doi=10.1.1.18.378.

114. Wolf, A, Herzog, A, Westerholz, S , Michaelis, B & Voigt, T 2009, 'Improving fuzzy-based axon segmentation with genetic algorithms', The IEEE Congress on Evolutionary Computation Evolutionary Computation, IEEE Congress, pp.1025–1031.

115. Xu, Y 2005, 'Recent advances in evolutionary computation', The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), vol.21,pp.1-18.

116. Xue, Yu, Zhuang, Yi , Tianquan, Ni, Ouyang & Wang 2012, 'Enhanced self-adaptive evolutionary algorithm for numerical optimization', Systems Engineering and Electronics, Journal, vol.23, no.6, pp.921–928.

117. Yamazaki, I, Kurzak, J, Luszczek, P & Dongarra, J 2014, 'Design and implementation of a large scale tree-based QR decomposition using a 3D virtual systolic array and a lightweight runtime', Parallel & Distributed Processing Symposium Workshops (IPDPSW), pp.1495–1504.

118. Yanteng, Peng, Li, Guochang, Gu, Yuan Wen, Yuan Liu & Dong Liu 2009, 'Accelerating HMMer on FPGAs using systolic array based architecture', Parallel & Distributed Processing, IEEE International Symposium, pp.1–8.

119. Yao, X 2006, 'Recent advances in evolutionary computation', nature inspired computation and appliances laboratory, The University of Science and Technology of China, vol.21, no.1, pp.1-18.

120. Zhang, MZ & Asari, KV 2007, 'An efficient multiplier-less architecture for 2D convolution with quadrant symmetric kernels, Integration', the VLSI Journal, vol.40, no.4, pp.490–502.

121. Zhen Wang & Shuqin Fan 2012, 'Efficient montgomery-based semi-systolic multiplier for even-type GNB of GF(2^m)', IEEE Transactions on Computers, vol.61, no.3, pp.415-419.

LIST OF PUBLICATIONS

1. **Selvakumar, R**, Dharmishtan K Varughese & Manoj Kumar Singh 2013, 'Multiple Automatic Design Vector Generation for Efficient Systolic Architecture using NIC', ElSEVIER Proceedings of International Conference on Advances in Communication, Network and Computing.

2. **Selvakumar, R** & Dharmishtan K Varughese 2014, 'Design of Systolic Architecture for FSMBA', International Journal of Applied Engineering Research, ISSN: 0973-4562, vol.9, no.24, pp.29181-29191.

3. **Selvakumar, R** & Dharmishtan K Varughese 2015, 'Design and FPGA Implementation of Systolic Array Architecture for Vector Computation', Australian Journal of Basic and Applied Sciences ISSN:1991-8178, vol.9, no.2, pp.133-142.

4. **Selvakumar, R** Dharmishtan K Varughese & Manoj Kumar 2015, 'Design Exploration for Systolic Architecture Using Hybridization of GA and Local Chaotic Optimization', International Journal of Applied Engineering Research ISSN: 0973-4562, vol.10, no.3, pp.7009-7031.