AREA AND POWER EFFICIENT REGISTER ALLOCATION TECHNIQUE FOR THE IMPLEMENTATION OF PRINCIPAL COMPONENT ANALYSIS

A Dissertation Submitted in Partial Fulfillment of the Requirement for the Award of the Degree of

MASTER OF TECHNOLOGY

in VLSI DESIGN

Submitted By

SUKHMANI KAUR THETHI

601562025

Under Supervision of

Dr. RAVI KUMAR

Associate Professor (ECED)

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY, PATIALA, PUNJAB

JUNE, 2017

DECLARATION

I, Sukhmani Kaur Thethi hereby declare that the work presented in this thesis entitled "**Area and Power Efficient Register Allocation Technique for the Implementation of Principal Component Analysis**" in partial fulfillment of the requirement for the award of degree of Master of Technology submitted at Electronics and Communication Department, Thapar University, Patiala is an authentic record of work carried out under supervision of Dr. Ravi Kumar (Associate Professor, Thapar University, Patiala) from 2015-2017. The matter presented in this dissertation has not been submitted either in part or full to any other university or institute for the award of any other degree.

Date: 21/08/2017

Sukhmani Kaur Thethi
601562025

It is certified that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 21/08/2017

Dr. Ravi Kumar
Associate Professor, ECED

ACKNOWLEDGEMENT

I feel privileged to undertake this opportunity to acknowledge and extend my gratitude to the people who have helped me directly or indirectly in completion of this dissertation. First and foremost, enormous gratitude to my faculty guide, **Dr. Ravi Kumar,** who has the attitude and the substance of a genius. Without his guidance and persistent help this dissertation wouldn't have been possible.

Many thanks to **Dr. Alpana Agarwal**, Professor and Head, ECED, for providing the students with the adequate facilities and infrastructure for carrying out the work without hindrance. I am also thankful to **Dr. Anil Arora**, Assistant Professor & P.G. Coordinator, ECED, for the motivation and inspiration to work diligently.

Particular thanks and love to my parents who have been there with me throughout, and have been exceptionally tolerant and patient and motivating. I cannot thank them enough for their years of unyielding love and encouragement. A specific note of gratitude to my friends who did not make this path or road of 'research' as difficult as it looked in the beginning. Their constant support and cheers can never be thanked enough here.

Learning is a step by step process, and this research undertaken by me has indeed helped me in instilling the knowledge related to my topic within me. I am sure it will help me in my future endeavours.

Sukhmani Kaur Thethi

ABSTRACT

In majority of the cases, every variable in the input HDL behavioural description needs to be held in a register and a register can be shared by multiple variables if they have mutually disjointed lifetime intervals. This approach has been proven effective for various DSP algorithms. Efficient utilisation of registers used in various architectures is very important in all aspects for the improved performance and efficiency. In this report, lifetime analysis technique is used for register minimisation.

Today multi-dimensional data analysis required in almost every field such as neuroscience, meteorology, oceanography, gene expression, agricultural studies, climate studies, material sciences, ecological studies, in IC designing, etc., as real time operations are expected to handle data efficiently. Many applications like pattern recognition, signal estimation, fault diagnosis, yield diagnosis, and many more requires dimensionality reduction which is typically done using PCA.

This paper presents a novel register allocation technique i.e. semi-static allocation technique as well as the conventional technique i.e. forward-backward register allocation technique; for the implementation of PCA incorporating variable reuse technique. The purpose of this paper is to avoid register switching and hence reduction in dynamic power consumption as well as area during the implementation of PCA. Syntheses of verilog codes written for both the techniques were carried out in RC (cadence) tool. In case of generic synthesis, a substantial decrease of 56.867% in power and 56.66% in case of area was observed; whereas, in case of mapped synthesis, significant reduction of 86.145% in power and 74.79% in area was observed for the proposed technique in contrast to the conventional one.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 PRINCIPAL COMPONENT ANALYSIS (PCA)

The orthogonal decomposition of a positive semi-definite matrix or PSD is used in multivariate analysis, where in the sample covariance matrices are PSD. This orthogonal decomposition in statistics is known as Principal Component Analysis (PCA), also known as the Karhunen-Loeve transformation in communication theory. It is the most widely used and well known of the 'standard' multivariate methods invented by Pearson (1901) and Hotelling (1933). The flowchart of PCA is given in figure 1.1.

PCA, a mainstay of modern data analysis, is a well known statistical technique used in many signal processing applications which is used to identify the most meaningful basis to re-express a data set by undergoing a dimensionality reduction. It uses different varieties of domains to reduce the number of dimensions in input sets without truncating the "information" contained within them and projects a new set of axes which best suites the data set; and which represents a reduced number of effective features, yet retain most of the intrinsic information content of the original data.

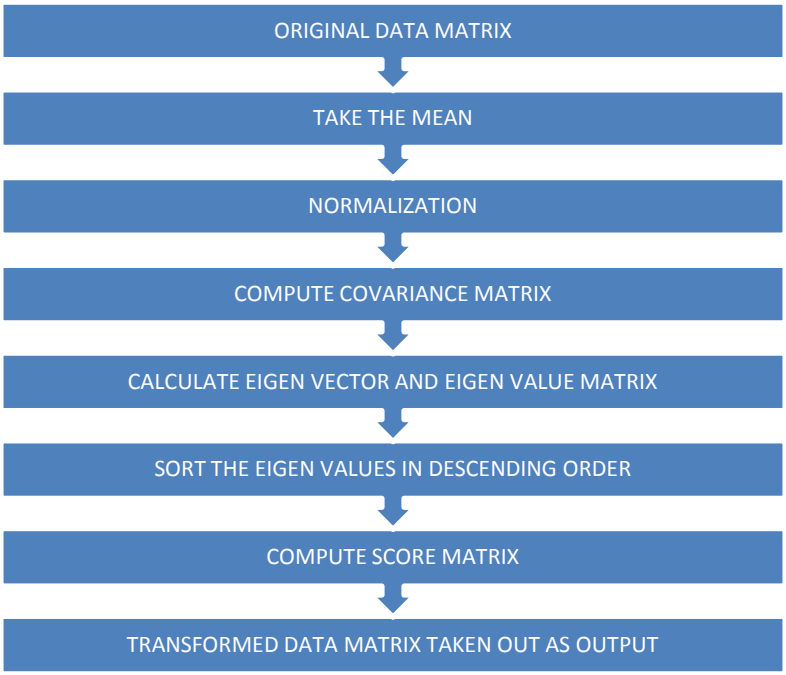

Figure 1.1: Flow chat for PCA Algorithm

1.1.1 STEPS TO CALCULATE PCA

a) Consider **X** = *m*-dimensional random vector representing the environment interest [1].

$$X = \begin{matrix} X_1 \\ \vdots \\ X_n \end{matrix} \quad (1.1.1)$$

b) Then find the mean of the random vector X

We assume that the random vector X has zero mean, or

$$\mathbb{E}[\mathbf{X}] = \begin{matrix} \mathbb{E}[\mathbf{X_1}] \\ \vdots \\ \mathbb{E}[\mathbf{X_2}] \end{matrix} \quad (1.1.2)$$

where $\mathbb{E}$ is the statistical expectation operator.

(If **X** has a nonzero mean, we subtract mean from it before proceeding with the analysis).

c) Random matrix is any matrix with random variables **Z**; where **Z** = $[Z_{ij}]$

Hence, $\mathbb{E}[\boldsymbol{Z}] => (E\left[\boldsymbol{Z_{ij}}\right])$ (1.1.3)

d) Then find the covariance of the matrix **X:**

$$\text{Cov [X]} = [\text{ Cov } (X_i, X_j)\ ] \quad (1.14)$$

$$\text{Cov [X]} = \begin{bmatrix} var\ (X_1) & \cdots & cov\ (X_1, X_n) \\ \vdots & \ddots & \vdots \\ cov\ (X_n, X_1) & \cdots & var\ (X_n) \end{bmatrix} \quad (1.1.5)$$

* If $X_1, X_2, \ldots, X_n\ are$ independent, then covariances are zero and hence,

$$\text{Covariance Matrix} = \text{Diagonal } [\ \sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2] \quad (1.1.6)$$

e) Then find the correlation **R** of matrix **X**

$$\text{Corr [X]} = [\text{Corr } (X_i, X_j)]$$

$$\text{Corr [X]} = \begin{bmatrix} 1 & \cdots & corr\ (X_1, X_n) \\ \vdots & \ddots & \vdots \\ corr\ (X_n, X_1) & \cdots & 1 \end{bmatrix} \quad (1.1.7)$$

f) PCA is obtained from eigen decomposition of a covariance matrix, and gives the least square estimate of original data matrix.

1.1.2 APPLICATIONS OF PRINCIPAL COMPONENT ANALYSIS

Earlier PCA involves computational complexity in performing manual calculations; hence its use was very limited until the dawn of computer era. Since then PCA has found its application in a wide range of multivariate problems pertaining to different fields such as, neuroscience, computer vision, meteorology, oceanography, gene

expression, material science, agricultural studies, pharmaceutics, ecological studies etc.

a) PCA is used for blind signal detection which uses eigen vectors for signal classification in multiple antenna systems [2], [3].

b) It is used in Modern Network Intrusion Detection System (NIDs) which is an anomaly detection to capture malicious attacks in the connections which are composed of large set of dimensions; to solve the time efficiency problem of processing these huge amounts of network data [4]. The PCA Framework for anomaly detection is shown in Figure 1.2.

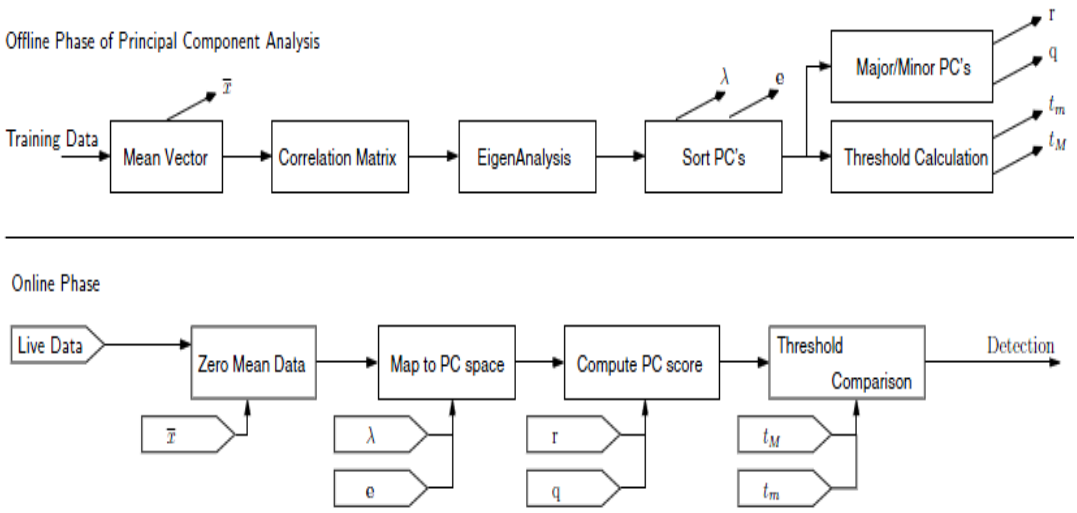

Figure 1.2: Principal Component Analysis for NIDs [4]

c) Principal Component Analysis is used for data analysis, where in a large data is provided for any type of survey [5-7]. One such example of data analysis can be climate studies; where in a series of values can represent mean air temperature in any month, over several years, for a particular location. The climate series formed in this way shows a statistical behaviour and is similar to white or red noise stochastic processes [8].

d) Subspace based techniques for face recognition such as Eigen-faces take advantage of large redundancy present in most images to compute a lower dimensional representation of their input data and stored patterns, and perform classification in reduced space which would substantially reduce the storage and computational requirements of the face recognition task [9-11]. Neural network for face recognition implements PCA for dimensionality reduction [12].

e) Internet of vehicles plays an important role in improving traffic situation aimed to identify road rage, fatigue driving or drunk driving and so on by recognising driver's

facial expression. PCA is used as a core algorithm of facial expression recognition [13].

f) PCA is used in analysing the information contained in the gene expression data by clustering the gene expression data [14].
g) PCA is used in neuroscience technologies like in MRI, EEG, and ECG [15-18].
h) PCA is used in multi-media chip for spike sorting [19].
i) Optimised implementation of PCA algorithm on spectral image analysis in real time [15].

1.1.3 CASE STUDY: Bivariate Data Set

Consider an example of a bivariate data set (2-D) to illustrate one of the applications of principal component analysis, as presented in figure 1.3 [1]. The horizontal and vertical axes (feature axes) of the diagram represent the original coordinates of the data set and are assumed to be of same scale. The rotated axes (labelled 1 and 2) are the resultant axes after PCA is applied to this data set. From figure 1.3, it can be observed that projecting the data set onto axis 1 captures the salient feature of the data- namely, the fact that the data set is bimodal (i.e. there are two clusters in its structure). A cloud of data points shown in 2-D, and the density plots formed by projecting this cloud onto each of two axes, 1 and 2, are indicated in the figure 1.3. The projection onto axis 1 has maximum variance and clearly depicts the bimodal/clustered character of the data. It is evident from the figure that the variance of the projections of the data points onto axis 1 is greater than that for any other projection axis in the figure. On the contrary, the inherent bimodal nature of the data set is completely uncertain when it is projected onto the orthogonal axis 2. Note that although the cluster structure of the data set is evident from the 2-D plot of the raw data displayed in the frame-work of the horizontal and vertical area, this is not always the case in practice. In the more general case of high dimensional data sets, it is quite possible to have the intrinsic cluster structure of the data concealed, and to analyse it we must perform a statistical analysis similar to PCA.

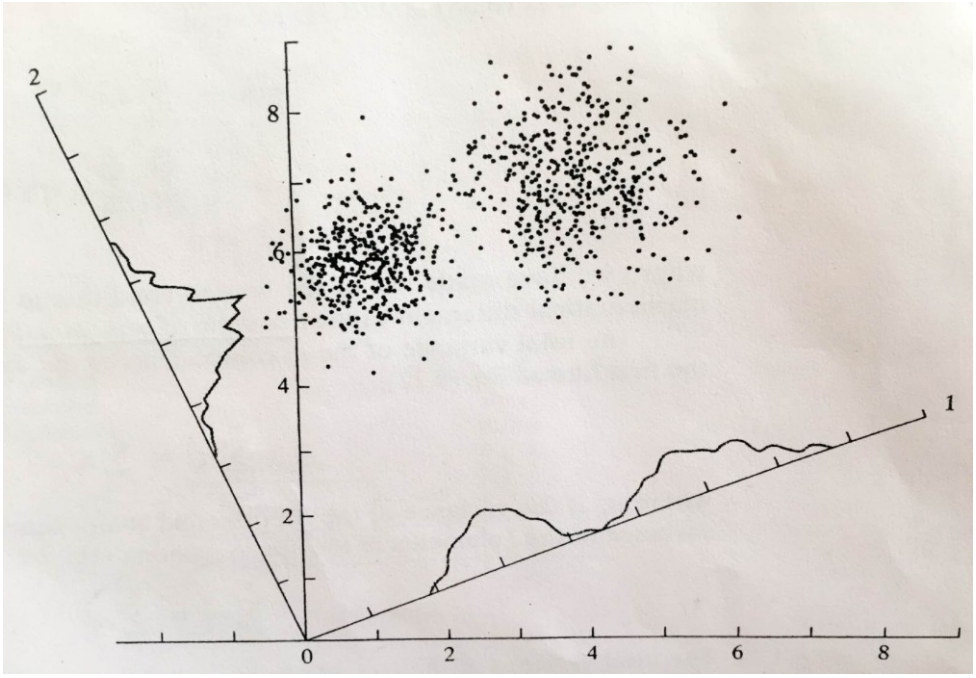

Figure 1.3: Example of Bivariate Data set showing Clustering and thus dimensional reduction [1]

For example in ECG machine, let the pulse rate be characterised into two sections viz. normal, and abnormal. It can be done with the help of clustering as explained above. When the axis of the result obtained is rotated at a certain angle so that cluster formation occurs, then the categorisation will be made clear reflecting upon amongst which category the result falls into (normal or abnormal).

1.2 RELEVANCE OF PCA IN VLSI

a) *Performance Optimisation of the effect of Nano-CMOS variability on ICs* [21]: **T**he "yield prediction" analysis of SRAM arrays is done with the use of PCA in achieving significant reduction in the computational cost of performing sufficient analyses to produce statistically reliable results. PCA transforms the Random variables (required to characterise a circuit) to a reduced number of statistically independent variables. The intrinsic variability of present VLSI technology (which is in nanometres) must be taken into account when analysing circuit designs to predict the outcome/yield. MonteCarlo (MC) and Quasi-Monte Carlo (QMC) based statistical techniques does this by analysing many randomised or quasi-randomised copies of the circuits. The various randomised models must form of the variations that occur in nano-CMOS technology, including short channel effects or even atomistic effects.

b) *Statistical Timing Analysis of* VLSI [22]: Statistical analysis is widely used in an IC design. Error estimation, Leakage analysis, gate delays, full chip statistical STA, dependent variation sources generation etc are some of the concerned areas in IC design. For this purpose, it is important to perform and study the analysis of the random

variables with non-linear dependence amongst them. With the CMOS technology scaling down to nanometres, process variations as well as the operating variations have become a limiting factor for IC design due to the uncertainty introduced for the circuit performance and leakage power. PCA is used here to remove the linear correlation between random variables, but fails to remove high order dependence. Non-linear PCA can decompose non-linear dependent variation sources to independent components, though the result is very complicated function of independent components.

c) *Minimizing circuit size (number of circuit elements)*: Complexity of the circuit depends upon the dimensionality of the inputs handled by the elements. Hence lesser the number of elements, lesser the complexity, as lesser the chances for error and improved performance is obtained. PCA helps in dimensionality reduction and hence minimizing number of elements in the circuits. For example, let the initial dimension of the inputs be 20, and corresponding to that let number of multipliers needed be equal to *m.* Let PCA reduces the 20 dimensional input to 3 dimensional, and let number of multipliers needed for the new value of the dimension be equal to *l,* and it Real is evident that *m>l.* Also number of elements required to implement n number of multipliers will be equal to n^2. Hence, clearly a large chip space is saved while implementing the circuit design when PCA is used [21].

d) *Face Recognition in biometric devices*: The face recognition task consists of assigning an identity to an unknown face of a subject by comparing it to a database of labelled face. However, due to high dimensionality of the input images, it becomes expensive and tedious to perform the classification on the original data. Fortunately, intrinsic dimensionality of human faces is much lower than that of their images. Hence PCA uses a linear transformation from high dimensional input data set/space to feature space, which preserves most of the information present in original vector [9-12].

1.3 IMPORTANCE OF REGISTER ALLOCATION FOR PCA

Real time operations are expected to handle data efficiently. For various applications such as pattern recognition, signal estimation, fault diagnosis etc, high dimensional data needs to be reduced to a low dimensional space which is typically done using Principal Component Analysis (PCA). However, steps involved in calculation of PCA are computationally very extensive. This has ramifications of a complex hardware with significant overheads in terms of silicon area, power consumption, and delay. Moreover, most of the above mentioned signal processing applications requires real-time hardware

implementation. Although many on-line PCA algorithms have been reported in the past such as Sanger's rule [22], the Rubner-Tavan model[23] and the APEX model, most of them suffer from convergence problems when multiple components need to be estimated. Rao and Principe [24] proposed a fact converging and stable sample based on-line PCA algorithm. Han et al. [25] have reported implementation of the on-line temporal PCA algorithm on a floating-point TMS 320 C32 chip.

One of the steps in calculating PCA is finding eigen values and eigen vectors from the covariance matrix achieved from the original high dimensional data. Our paper presents how compiler would implement or execute the function for finding the eigen values in the code of PCA by storing the variables in temporary registers whilst executing that polynomial function.

In register allocation a large number of target programs are assigned to variables onto a small number of CPU registers for compiler optimization. Modern RISC architectures have very large register sets with 32 general purpose integer registers and 32 floating point registers or more. Hence, register allocation is very important for modern RISC architectures as all operations in former occur between registers only. The effective use of the large register sets for the placement of variables is very important for achieving high performance due to long latencies of memory accesses in modern processors. When enough registers are not available, then some values need to be stored and retrieved from the stack more often. This makes it much less efficient than operating directly on registers.

Registers constitute a major part in the silicon area of a chip, hence it is necessary to minimize the number of registers being used for any process or operation [26]. Various methods of register allocation have been studied before. However, register allocation involving variables repeating at different clock cycles have not been done before. Two aspects of register allocation are concentrated in this paper, variable reuse and minimization of power consumption for PCA implementation.

We use lifetime analysis to calculate minimum number of registers required for our operation of finding Eigen values from a 3x3 covariance matrix. Viewing the complexity of PCA, it is shown later that a large number of variables are needed for register allocation, which can consume a large portion of the available resources. Optimised use of registers to store data and less power consumption are two important aspects in VLSI

architecture. Power consumption for any operation can be minimized by minimising the transitions (or switching) between the registers [27].

Interrupt is an important term when it comes to register allocation by compiler. Real time operating systems today are based on pre-emptive scheduling (task scheduling). In this paper, we are discussing about local register allocation technique which deals with the single function of the code at a time, and not the whole code as a block. If at all, a high priority interrupt occurs, it will be served after the completion of the execution of the current instruction.

Both forward-backward register allocation technique and our proposed semi-static register allocation technique uses same number of registers obtained from lifetime analysis but register switching occurs in every clock cycle in case of former while no register switching occurs in case of latter [28]. Hence our proposed technique is far superior when it comes to power efficiency.

1.4 WHY INSTRUCTION LEVEL REGISTER ALLOCATION TECHNIQUE?

Many recent researches and developments in the domain of implementation of PCA in real time applications were studied which can be summarised as follows:

Static Single Assignment (SSA) is the preferred form for register allocation in compiler design which aims to map temporaries to physical registers such that their live ranges do not interfere. Graph colouring based techniques which give rise to chordal interference graphs are the most popular ones and have been reported by several authors [29-31]. Pereira and Palsberg gave a general framework for register allocation inspired from puzzle solving methods [32]. In this scheme, the inputs are first transformed to elementary programs which led to increased complexity. In a subsequent paper, they proposed a different, spill-free way to perform SSA elimination after register colouring [33]. Hack & Goos reported register coalescing by graph coloring [34]. However, it has been proved analytically that optimal coalescing is NP-complete and hence we need to rely only on heuristics [35]. Braun & Hack presented an improved spilling algorithm in SSA form which is again however, computationally extensive [36]. Pereira & Palserg presented a linear time, scalable, locally optimised coalescing algorithm named Punctual Coelescing [37].

However, a fundamental limitation of graph coloring based register allocation is the number of coloring iterations required before a solution is identified. This has led to the shifting of focus towards biologically inspired search algorithms since there is a scope of substantially reducing the required iterations heuristically. Lintzmayer *et al*. have

introduced a new algorithm named Color Ant-RT which makes use of Ant-colony optimisation to improve traditional graph coloring algorithm for intra-procedural register [38]. Subsequently they reported another variant of the hybrid Color-Ant –RTA algorithm which uses tabus as local search [39].

Despite all these developments, one is compelled to ponder about the computational overhead associated with such meta-heuristic register allocation techniques. This has motivated the author to revisit the traditional forward backward allocation technique and address the implementation issues associated with it in the backdrop of a computationally intensive problem viz. PCA. Furthermore, a more efficient register allocation technique is proposed in this dissertation implementing a function of PCA.

Evidently, the importance of register allocation at instruction level can't be overlooked since even the global allocation techniques are eventually implemented at this very level. It is very likely that an improvement in the traditional forward backward allocation scheme would result in improved performance in terms of power and area with or without a global allocation technique.

1.5 OBJECTIVE OF THE PROPOSED WORK

This dissertation presents a novel register allocation technique for the implementation of Principal Component Analysis (PCA). PCA deals with a large dimensional data and is a computationally intensive technique. Out of both the techniques presented in this dissertation, Semi-Static Register Allocation Technique is suitable for this work, because PCA is computational intensive and not control dominated circuit (which may involve other efficient allocation techniques).

The purpose of this dissertation is to avoid register switching and hence reduction in dynamic power consumption as well as area during the implementation of PCA.

This dissertation attempts to achieve the following objectives:

a) To undertake performance analysis and comparison of both the register allocation techniques to implement PCA.
b) To describe the complete architectural flow of both the techniques used to implement PCA, by using hardware description language (verilog) and verify it by simulating on ISE Simulator (ISim).
c) To implement a function of PCA using both the register allocation techniques on RC Encounter (Cadence) and simulate to calculate the area and power required by both the techniques.
d) To obtain layout of both the register allocation techniques and compare their areas.

1.6 NOVEL ASPECTS OF THIS DISSERTATION

The work presented in this dissertation claims novelty on several grounds some of which are listed below:

a) This work proposes novel modification in the traditional register allocation scheme with an aim to reduce power consumption and area overhead.
b) A novel heuristic technique viz. semi-static register allocation is being proposed which has been found to be extremely power efficient.
c) The proposed technique avoids register switching at every clock cycle thereby reducing power dissipation significantly.
d) The technique reuses individual variable at different clock cycles leading to a reduction in total number of registers used for storage.

The proposed semi-static register allocation technique has been compared with the traditional forward backward register allocation technique for the implementation of PCA.

1.7 ORGANISATION OF DISSERTATION

As explained in the previous sections, the aim of the work is to implement a computational extensive technique i.e. Principal Component Analysis (PCA) with the help of Register Allocation Techniques. In order to achieve the same, the dissertation has been divided into the following six chapters:

Chapter 2 presents the Literature Survey done for the accomplishment of this work undertaken.

Chapter 3 describes the traditional Register allocation technique i.e. Forward-Backward Register Allocation; and the proposed technique i.e. Semi-Static Register Allocation Technique.

Chapter 4 describes the results and discussion section. It also outlines the basic description of all the tools used for simulation and implementation of the main work presented in this dissertation. A comparison is drawn in tabular form between both the register allocation techniques; and the simulation results in terms of power and area are attached along with the table.

Finally, Chapter 5 sums up the conclusion of the work and suggests some ideas for the future work along with the gaps in the present work.

CHAPTER 2
LITERATURE SURVEY

Parhi [43], proposed a novel forward-backward register allocation technique which addressed the systematic synthesis of DSP data format converter or DFC architectures with minimum number of registers. Systematic life time analysis is performed to calculate the minimum number of registers needed for implementation of converter which can then be synthesized using many possible register allocation schemes viz. Forward backward allocation technique, Graph coloring technique, Forward Circulate allocation technique and many more. The proposed allocation technique requires less area for the implementation of control circuits than a simpler forward-circulate allocation scheme.

Parhi [42], presented a general approach or technique to calculate the minimum number of registers used to implement any algorithm or function in a DSP circuit for any arbitrarily specified life time with random periodicity of computation. DSP operations are repetitive as well as periodic in nature. The life time chart defines the life period (birth time and dead time) of all variables used in the algorithm implementation in a single frame; and then subsequent frames are computed in a periodic manner. This paper presents the techniques for the calculation of minimum number of registers by using the life the chart and the circular life time graph.

Wu *et al.* [27], proposed an approach to reduce the register cost for a control dominated circuit. Traditionally, after performing lifetime analysis for the variables, register allocation for those variables is performed. Lifetime of a variable is defined as the time duration in which a variable is produced until it is consumed. Multiple variables can share a register provided their lifetimes do not overlap with one another. The register and its associated combinational circuit can be replaced with some simpler circuits based on the fact that a variable might be a function of some other variables, the controller state, some signal nets, or a mixture of above.

Stamkopoulos *et al.* [18], presented an algorithm developed for the ST segment feature extraction based on non linear Principal Component Analysis (NLPCA). Ischemic cardiac beats form a patient's ECG signal detection is based on the ST Segment, which is the characteristics of a specific part of the beat (duration or interval between the S and the T

segment). The correct classification (normal or abnormal heart beat) of the beats depends on the efficient and accurate extraction of the ST segment features through a statistical analysis technique. The NLPCA techniques are to classify each segment into one of the two classes that is normal or abnormal.

Srivatsan *et al.* [29], proposed a new register allocation technique called semi-static register allocation technique for low power data format converters (DFCs). This new technique not only minimises the number of registers used, but also minimises the power consumption in the DFCs by minimising the transitions/ switching and interconnections between the registers which are used to store data.

Yeung *et al.* [14], presented an analysis of clustering gene expression data in order to extract the information contained in the gene expression data for medical requirement. Clustering is an important and useful technique for the analysis of gene expression data (which can be done using PCA), because of the complexity of biological networks due to the presence of large number of genes. This paper proposed the effectiveness of PCs in capturing cluster structure and compared the quality of clusters obtained from original data and data after projecting onto subsets of principal component axes. The result showed that clustering with PCs often degrades the cluster quality and has different impact on different algorithms. Except in special cases, PCA is not recommended before clustering.

Han *et al.* [26], presented a real time PCA implementation on DSP chip. An on-line temporal PCA learning algorithm (which is coded in assembly language to optimise) is implemented on a floating point DSP for real-time applications. This algorithm can accurately estimate Principal Components (PCs) from input as well as can track those PCs from time varying input. A serial interrupt occurring at every sampling time from CODEC to DSP platform, is used here to implement PCA algorithm in real-time scenario. CODEC is used here for analog to digital data conversion; and this A/D converted data is transferred to DSP serial port. This triggers the interrupt cycle, and then PCA routine is performed sample by sample. Here all the features of a DSP chip are used, such as, circular buffer, internal memory inside the chip, and parallel instruction.

Bashir *et al* [13], presented a novel technique for model based recognition of complex object motion trajectories using Gaussian Mixture Models. These trajectories are segmented into

small units of similar pieces of motions (perceptually) with the help of PCA which can be called as sub-trajectories. Then these sub-trajectories are fitted with known mixture of Gaussians to estimate the underlying class probability distribution.

Yan [6], implemented a process failure analysis using PCA. After a process is finished in an IC process line, the process control module (PCM) is tested for process quality status. If a process failure occurs, then failure analysis is required to be performed to find out the root cause of the failure. Hence, this paper presents process failure diagnosing technique using PCA. First of all, all the correlated PCM parameters are analysed and transformed into a new set of independent parameters using PCA; followed by process failure cause identification from PCA eigen vectors. Moreover, the state space of process can be constructed with PCA eigenvectors used as a coordinate base and hence, this makes it possible to trace the IC processes or predict the process failure possible before hand.

El-Bakry *et al.* [12], presented new implementation of PCA in real time scenario, for face detection in a small amount of time i.e. fast face detection. PCA can be employed in many important applications especially in pattern detection such as face detection/ recognition due to its statistical nature. Such new implementation is achieved by cross correlation in the frequency domain between the input image and eigenvalues (weights). Simulation result shows that the proposed implementation of PCA is faster than conventional one.

Turk *et al*. [10], proposed a near real time system that can locate and track the subject's head; and then can identify the subject by comparing the characteristics of the face with the known individuals. Faces are normally straight or flat with featured embedded over it, hence they can be described by a small set of 2-D characteristic views instead of considering it to be in a 3-D view. This way, they have efficiently managed to treat the face recognition problem as an intrinsically 2-D recognition problem. The face images are projected onto a feature space that spans the significant variations among known face images. The significant features or "eigenfaces" which are the eigen vectors/PCs of the set of faces; do not necessarily correspond to features such as nose, ears, lips and eyes. The projection operation characterises an individual face by a weighted sum of the eigenface feature, and so to recognise a particular face, it is necessary only to compare these weights to those of known individual.

Das *et al.* [4], presented an efficient FPGA architecture for statistical analysis method i.e. PCA to reduce the dimensionality of the network data. Modern Network Intrusion Detection Systems (NIDSs) are used to scan for the suspicious network activity and seize harmful/malicious attacks within the connections which are described by large set of dimensions. PCA is employed to solve this time-efficiency problem in processing these huge amounts of network data, which otherwise makes it extremely slow.

Carvajal *et al.* [9], explained subspace based face recognition technique in analog and VLSI domains. Large redundancy is present in most of the images. Hence sub-space based techniques like Eigenfaces and Fisherfaces take advantage of this factor to compute a lower dimensional representation of the input data and stored patterns, for classification in reduced subspace which results in lower storage and less computational requirements for face recognition. Human faces exhibit regular statics, hence, their intrinsic dimensionality is much lower as compared to their images.

Hseih *et al.* [7], explained wafer sort Bitmap Data Analysis using PCA based approach for yield analysis and optimisation. Yield analysis is most important parameter for semiconductor processes. For a new process development during its initial stage, several learning cycle iterations are required to solve yield loss issues. Less number of iterations to solve the yield loss issues are desirable for enhanced efficiency which is presented in this paper. Firstly, the failure classification of bitmap data is transformed to new basis using PCA followed by the calculation of the defective rates. Then clustering of data according to the variances is done, which generates yield loss space by Cluster Analysis. Then physical failure analysis samples can be selected to solve yield loss issues.

Chang *et al.* [11], explained the popular issue of identity management that is user identification. Fingerprints, DNA and the human eye or face are distinct characteristics usually used in user identification and authentication. This paper presents the human face as the main subject for authentication which employs PCA as the main algorithm accomplished on ARM system. Through PCA, eigen faces of the training faces are obtained followed by the classification of the weighting patterns, and then update of eigen faces or weight patterns. If the same unknown face is seen several times, then its characteristic weight pattern is calculated and incorporated into the known faces category. Because the ARM system provides strong computation ability, various interfaces such as cameras, mass storage, etc was

used in this study.

Bhatti *et al.* [2], presented a technique for blind signal detection that uses PCA, and are tested in a real time with Software Defined Radios (SDRs). Two conditions should be fulfilled before PCA is used: First, the mean of the data should be equal to zero; and second, the data should follow normal distribution. Data considered here is composed of time domain complex baseband samples with normal distribution. PC algorithm can be effectively used for the detection of very low power signals without the knowledge of the source signal and channel.

Josth *et al.* [20], presented two optimisation implementations of the PCA algorithm for spectral image analysis in real time for medical imaging. For the implementation of PCA in real time, improvement in the speed of PCA; especially the speed of creation of the co-variance matrix is desirable. One of the implementation utilises the SSE instruction set of contemporary CPUs; and the other runs on graphic processors (or GPU) using CUDA environment. Human eye can perceive only three colours viz. RGB. But representation based on these three colours cannot capture whole information; hence spectral imaging and analysis must be used. If a spectral image contains 81 wavelength channels, then PCA could be employed for the reduction of spectral dimensionality to 6-11 without losing important information.

Xie *et al.* [21], presented the "yield prediction" analysis of SRAM arrays with the use of PCA in achieving significant reduction in the computational cost of carrying out sufficient analyses to produce statistically reliable results. PCA transforms the Random variables required to characterise a circuit to a reduced number of statistically independent variables. The intrinsic variability of nano-scale VLSI technology must be taken into consideration when analysing circuit designs to predict the outcome or the yield. MonteCarlo (MC) and Quasi-Monte Carlo (QMC) based statistical techniques do this by analysing many randomised or quasi-randomised copies of the circuits. The various randomised models must form of the variability that occur in nano-CMOS technology, including short channel effects or even atomistic effects.

Bhatti *et al.* [3], proposed a comparative analysis of eigen vector based signal detection algorithm, which is Principal Component based detection, is presented in this paper along

with a new eigen vector based modulation classifier which finds its application in multiple antenna cognitive radio. Here, Principal Component algorithm is used for blind signal detection using multiple antennas. For signal classification, it is assumed here that the features of the intended signals are known prior to the classifier.

Cheng *et al.* [22], proposed the analysis of the random variables with non-linear dependence amongst them. With the CMOS technology scaling down to nanometres, process variations as well as the operating variations have become a limiting factor for IC design due to the uncertainty introduced for the circuit performance and leakage power. PCA is used here to remove the linear correlation between random variables, but fails to remove high order dependence. Non-linear PCA can decompose non-linear dependent variation sources to independent components, though the result is very complicated function of independent components.

Chavan *et al.* [17], proposed a wavelet based multi scale PCA algorithm and demonstrated to enhance the classification performance in identifying EEG (Electro-Encephalogram) signals. Wavelet transform is used for the signal decomposition. Then PCA is used for de-correlation to achieve maximum compression, while simultaneously preserving the dominant modes of the signal and bad data rejection. The optimum decomposition scale of wavelet transform is selected on the basis of energy of wavelet coefficients in each scale. The proposed algorithm employing multi scale PCA computes the PCs of the wavelet coefficients at each scale, followed by combining the results at relevant scales. The wavelet coefficients of a particular scale corresponding to the dominant eigen values are retained for signal compression.

Bhuvaneshwari *et al.* [15], presented an intelligent image classification technique in Magnetic Resonance (MR) medical imaging i.e. PCA, in order to classify it as normal or abnormal slices of MRI brain image. Semantic with respect to the images, represents the association between the low level visual features and high level concepts that can be put in words. Similar MRI images and its feature is extracted for training the database using PCA technique in order to reduce the large dimensionality of the data, which in turn decreases the computational cost of analysing the data. Firstly, PCA and SFL (Semantic Feature Layers) are used to achieve the feature extraction; which are then fed to PNN (Probabilistic Neural Network) as input, which classifies the image as normal or abnormal.

2.1 GAPS IN PRESENT STUDY

It is a well known fact that dimensionality reduction serves as an important purpose of compression and feature extraction for various reasons as cited above. However, ubiquitous computing implements dimensionality reductions only at software platforms. This renders the whole process lacking in real time capabilities. Furthermore, implementation of popular algorithms on reconfigurable hardware platforms is a relevant research topic which requires careful analysis and error free implementation. Moreover, traditional study and work on PCA suggests that the main constraint of PCA is the execution time in terms of updating when new data is included.

Also, the register minimisation techniques cited above, altogether do not serve whole purpose of efficiency. Where on one side if number of register are minimised, then on the other hand number of clock cycles are increased for variables when live. Also the polynomial equation, whose register allocation is performed in our work, still requires a minimum of twelve registers which is a large count obtained through lifetime analysis. In future, some other algorithm is needed for register allocation at instruction level, which would render more efficiency and less number of sources.

Moreover, the literature is silent about computational intensive problem in which registers are reused.

CHAPTER 3

REGISTER ALLOCATION TECHNIQUES

This chapter focuses on allocating the minimum number of registers used in DSP architecture so that the silicon area due to registers remains small [40].

3.1 LIFETIME ANALYSIS

Lifetime analysis is a procedure used to compute the minimum number of registers required to implement a DSP algorithm in hardware. A data sample (also called a *variable*) is *live* from the time it is produced through the time it is consumed or *dead*. A variable occupies one register during each time unit it is live. In lifetime analysis, the number of live variables at each time unit is which is the minimum number of registers required to implement the DSP program computed and the maximum number of live variables at any time unit is determined [40-41].

For example, Let a DSP program produces 3 variables => w, *x, y*

Live durations of these three variables are:

$$w : n \in \{1,2,3,4\} \quad (3.1.1)$$

$$x : n \in \{2,3,4,5,6,7\} \quad (3.1.2)$$

$$y : n \in \{5,6,7\} \quad (3.1.3)$$

Assuming that the lifetimes of variables from the previous and subsequent iterations of the program do not overlap with the lifetimes of *w*, x, or y, the number of live variables during the time units 1,2,3,4,5,6,7 is 1,2,2,2,2,2,2, respectively.

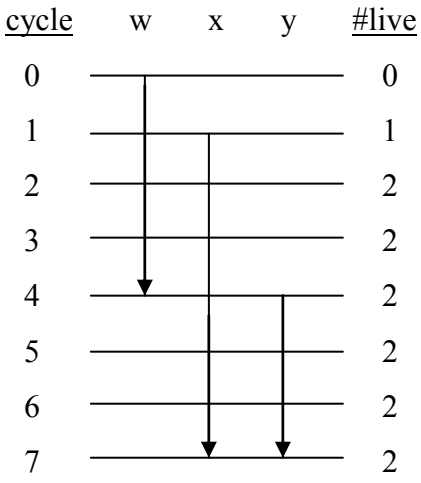

Figure 3.1: A Linear lifetime chart [40]

The minimum number of registers required to implement this DSP program is the maximum number of live variables at any time unit, which is

$$\max \{1,2,2,2,2,2,2\} = 2 \quad (3.1.4)$$

A linear lifetime chart is used to graphically represent the lifetime of each variable in a linear (as opposed to circular) fashion. The linear lifetime chart for our example is shown in the Figure 3.1. The horizontal line represent clock cycles (or time units), and the vertical lines represent the lifetimes of the variables. Note that this lifetime chart uses the convention that a variable is not live during the clock cycle in which it is produced, and the variable is live during the clock cycle in which it is consumed. For example, the variable *w* is produced during cycle 0 and is consumed during cycle 4, and this variable is live during the cycles 1,2,3,4.

The number of live variables during each cycle is shown at the right of the lifetime chart. The maximum number of live variables at any time step is 2, so the minimum number of registers that can be used to implement the underlying DSP program is 2.

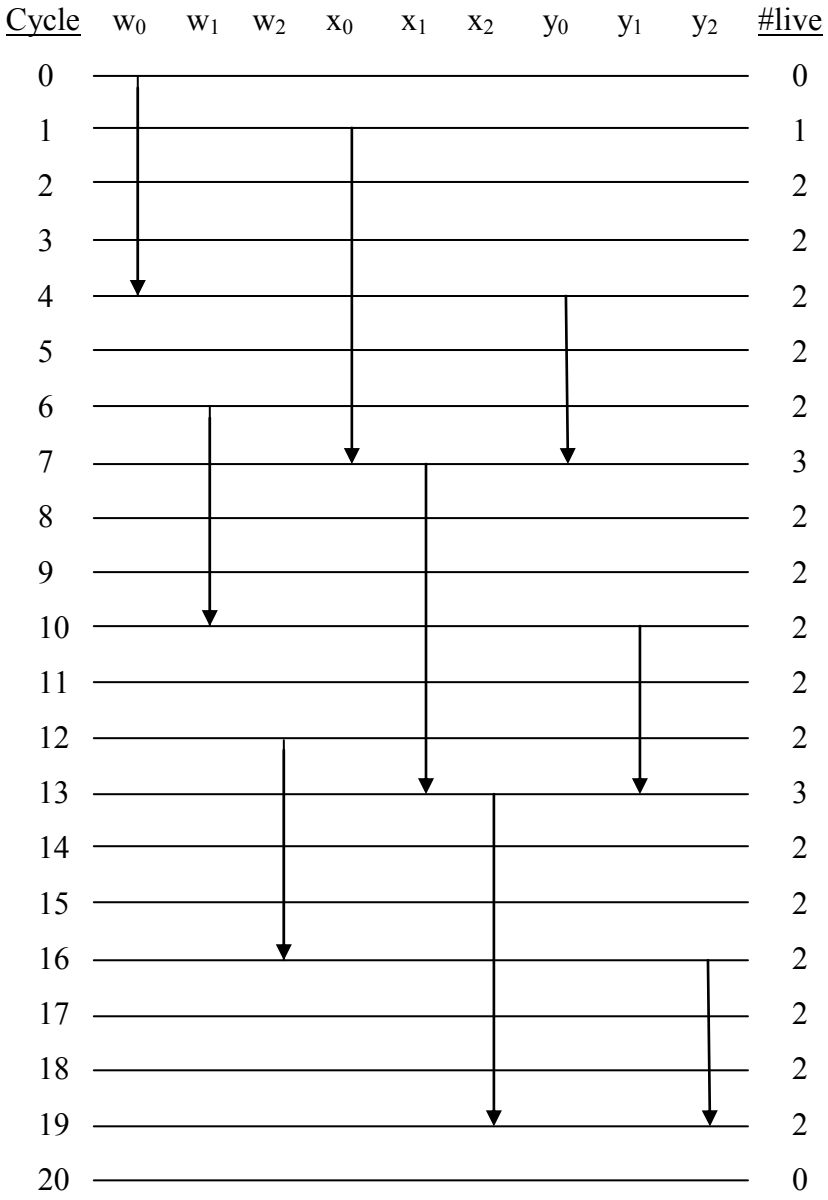

Figure 3.2: The linear lifetime chart explicitly showing 3 iterations of DSP program (N=6) [40]

In general, DSP programs are periodic. The lifetimes of *w*, *x*, and *y* from one iteration of

the underlying DSP program may overlap with the lifetimes of these same variables from other iterations. For example, let the iteration period be N=6, and let w_i, x_i, and y_i denote the variables *w, x,* and *y* resulting from iteration i of the DSP program. The lifetime chart for the three consecutive iterations of the DSP program is shown in Figure 3.2. Notice that the lifetime of w_1 overlaps with the lifetimes of x_0 and y_0, and as a result the DSP program requires 3 registers when the periodic nature of the DSP program was not taken into account. Recall that only 2 registers were required in Figure 3.1, where the periodic nature of the DSP programs must always be taken into account during lifetime analysis. Fortunately, this periodic nature can be taken into account without explicitly drawing the lifetimes of several iterations. This is done by drawing the lifetimes of the variables for the 0-th iteration and letting the number of live variables at the time partitions $n \geq$ N be the sum of the number of live variables due to the 0-th iteration at cycles n - kN for all integers $k \geq 0$.

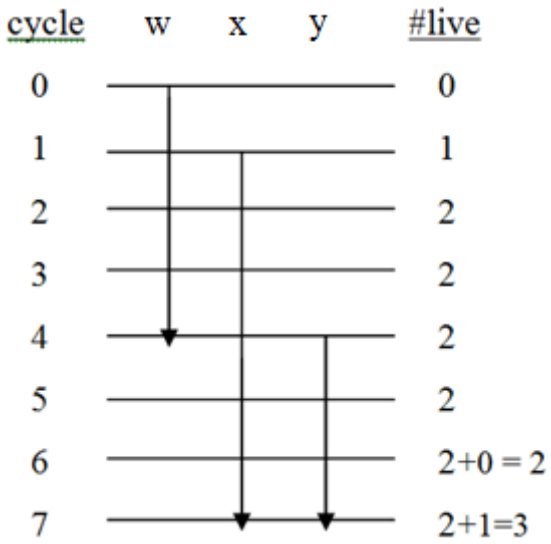

Figure 3.3: The linear lifetime chart implicitly taking into account the periodicity of the DSP program assuming period N=6 [40]

This is shown in Figure 3.3, where the number of live variables in cycle 7 is the sum of the number of live variables due to the 0-th iteration at cycles 7 and 1, which is 2+1 = 3.

In general, lifetime analysis begins with the construction of a lifetime table, such as the one in Table 3.1. This table represents the transpose operation of a 3 x 3 matrix which is as follows:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \tag{3.1.5}$$

The above matrix is reordered from row-by-row to column-by-column ordering.

SAMPLE	T_{input}	T_{zlout}	T_{diff}	T_{output}	Life Period $T_{input} \rightarrow T_{output}$
a	0	0	0	4	0 → 4
b	1	3	2	7	1 → 7
c	2	6	4	10	2 → 10
d	3	1	-2	5	3 → 5
e	4	4	0	8	4 → 8
f	5	7	2	11	5 → 11
g	6	2	-4	6	6 → 6
h	7	5	-2	9	7 → 9
i	8	8	0	12	8 → 12

Table 3.1: Lifetimes for 3x3 Matrix Transpose Operation [40]

Each variable in Table 1 has an input time, T_{input}, and a zero-latency output time, T_{zlout}, which is the output time of the variable ignoring the latency of the system. Each variable in the zero-latency system is live for $T_{diff} = T_{zlout} - T_{input}$ clock cycles; however, this violates causality if any of the life periods are negative.

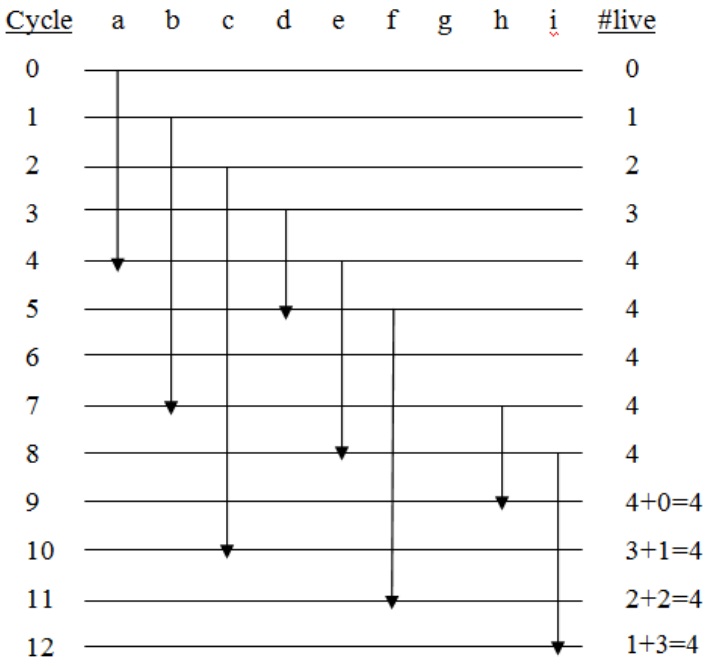

Figure 3.4: The linear lifetime chart for the 3x3 matrix transposer with period N=9 [40].

For example, the variable d is input at $T_{input} = 3$ and output at $T_{zlout} = 1$ and is live for 1-3 = -2 time units, which is not possible. To force the system to be causal latency is added to the system. The latency, T_{lat}, is the magnitude of the most negative value of T_{diff}, and $T_{lat} = -(-4) = 4$ in Table 3.1.

The actual output time of the variables is $T_{output} = T_{zlout} + T_{lat}$, and the life period of the variables is $T_{input} \rightarrow T_{output}$. Using these life periods, the lifetime chart can be drawn. The lifetime chart for Table 3.1 is shown in Figure 3.4. Note the period of this DSP program is N=9.

3.2 REGISTER ALLOCATION FOR PCA:

PCA is a statistical procedure for dimensionality reduction of input data without information loss. A crucial step towards PCA is finding the Eigen values and Eigen vectors of the data. It is assumed here, that the covariance matrix (A) of the input data set is already available. To proceed with PCA calculation, the next step is to find the Eigen values (λ_i) and Eigen vectors (X) from the covariance matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{3.2.1}$$

To find the eigen values, it is necessary to write the characteristic equation of the covariance matrix (A). The roots of the characteristic equation obtained after calculation are the eigen values.

$$AX = \lambda X \quad \text{or} \quad (A-\lambda I)\, X = 0 \tag{3.2.2}$$

$$(A-\lambda I) \rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{bmatrix} \tag{3.2.3}$$

Now to find Eigen values:

$$(A-\lambda I) = 0 \tag{3.2.4}$$

Therefore,

$$(a_{11}-\lambda)\ \{(a_{22}-\lambda)(a_{33}-\lambda) - a_{23}a_{32}\} - a_{12}\ \{a_{21}(a_{33}-\lambda) - a_{23}a_{31}\} + a_{13}\ \{a_{21}a_{32} - a_{31}\,(a_{22}-\lambda)\} = 0 \tag{3.2.5}$$

To obtain a result, compiler needs to compute the value of the polynomial written-in Eq. (3.2.5). Here comes the role of register allocation which determines that what values should be placed into which registers in individual clock cycles during to the execution of the program. This task of the compiler should be simple and optimized to yield maximum efficiency.

Now for register allocation, various entities in Eq. (3.2.5) are grouped and assigned to different variables as follows:

- $x_1 \{x_2x_3 - a_{23}a_{32}\} - a_{12} \{a_{21}x_3 - a_{23}a_{31}\} + a_{13} \{a_{21}a_{32} - a_{31} x_2\} = 0$
- $x_1 \{y_1 - y_2\} - a_{12} \{y_3 - y_4\} + a_{13} \{y_5 - y_6\}$
- $x_1u_1 - a_{12}u_2 + a_{13}u_3$
- $z_1 - z_2 + z_3$
- $m_1 + z_3$
- m_2

For instance, $(a_{11} - \lambda)$ variable is replaced by variable x_1 and $(a_{22} - \lambda)$ is replaced by variable x_2 so on. New variables replace the old ones and keep getting assigned in the temporary registers. This is where we can apply the concept of variable reuse as described in the following sections.

3.3 FORWARD-BACKWARD REGISTER ALLOCATION SCHEME:

In this method, variables are allocated all the way forward, and then in an appropriate backward register, thus the name "forward-backward" [40-42]. This scheme proceeds as follows:

Step 1: First step is to perform lifetime analysis (of equation (3.2.5)), and from that, calculate the minimum number of registers as shown in Table 3.2.

Sample	T_{in}	T_{zlout}	$T_{diff} = T_{zlout} - T_{in}$	T_{out}	Life Period $T_{input} \rightarrow T_{output}$
$a_{11}-\lambda = x_1$	6	9	3	20	6-20
a_{12}	0	8	8	19	0-19
a_{13}	5	10	5	21	5-21
a_{21}	1	4	3	15	1-15
$a_{22}-\lambda = x_2$	7	7	0	18	7-18
a_{23}	2	5	3	16	2-16
a_{31}	3	3	0	14	3-14
a_{32}	4	2	-2	13	4-13
$a_{33}-\lambda = x_3$	8	0	-8	11	8-11
$x_2 x_3 = y_1$	9	11	2	22	9-22
$a_{23}a_{32}=y_2$	10	6	-4	17	10-17
$a_{21} x_3 = y_3$	11	13	2	24	11-24
$a_{23}a_{31}=y_4$	13	12	-1	23	13-23
$a_{21}a_{32}=y_5$	12	1	-11	12	12-12
$x_2 a_{31} = y_6$	14	17	3	28	14-28
$y_1 - y_2 = u_1$	15	19	4	30	15-30
$y_3 - y_4 = u_2$	16	14	-2	25	16-25

y_5–y_6= u_3	17	15	-2	26	17-26
x_1 u_1=z_1	18	16	-2	27	18-27
a_{12} u_2= z_2	19	18	-1	29	19-29
a_{13} u_3= z_3	20	20	0	31	20-31
z_1-z_2= m_1	21	21	0	32	21-32
m_1-z_3= m_2	22	22	0	33	22-33

Table 3.2: Lifetime analysis of polynomial equation (3.2.5) derived from a 3x3 covariance matrix

Using these life periods, the linear lifetime chart can be drawn as shown in Figure 3.5

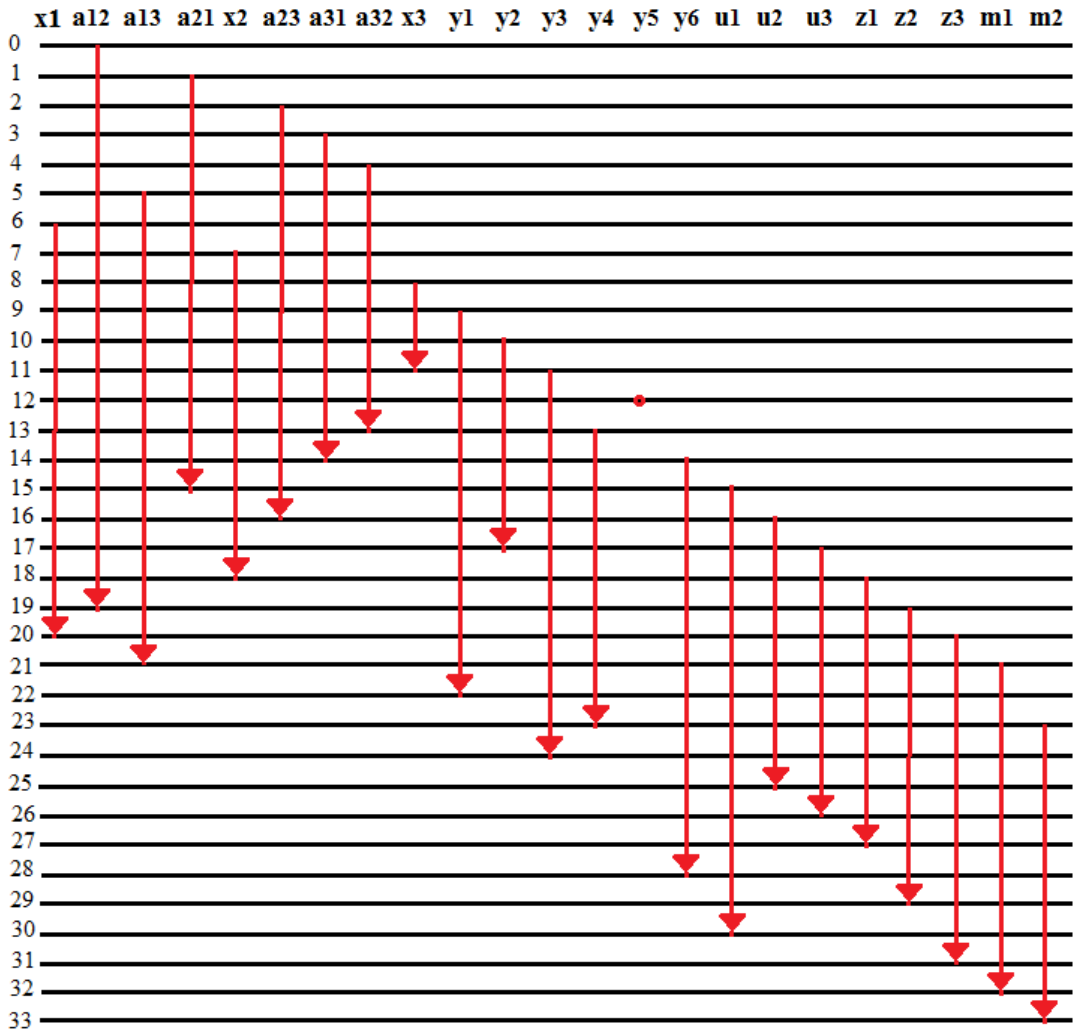

Figure 3.5: Linear Lifetime chart drawn for polynomial equation 5 derived from a 3x3 covariance matrix

The minimum number of registers required to implement the above lifetime chart is the maximum number of live variables at any time unit, which is: max{0,1,2,3,4,5,6,7,8,9,10,11,11,11,11,11,11,11,11,11,11,11,11,10,9,8,7,6,5,4,3,2,1} = 11.

The horizontal line in the lifetime chart represent clock cycles or time units, and vertical lines represent the lifetimes of the variables.

Step 2: Variables are allocated in forward manner until dead or till they reach last register. In forward allocation, if register *r* holds the variable in current cycle, then in next clock cycle variable is moved to *r+1* register. In case register *r+1* is unavailable, then the variable is allocated to the first available forward register.

Step 3: Then variables are allocated to register in a backward fashion after remaining life period is calculated. If multiple registers are available for backward allocation, then that register is chosen which is closer to the present register and has sufficient number of available forward registers to carry out the process in forward manner.

Step 4: If allocation is not yet complete, then steps 3 and 4 are repeated until the allocation is complete.

Forward Backward allocation scheme for a 3x3 matrix Transpose operation (Eq. 3.1.5); and from Table 3.1; can be shown in Figure 3.6

cycle	input	R1	R2	R3	R4	output
0	a					
1	b	a				
2	c	b	a			
3	d	c	b	a		
4	e	d	c	b	a	a
5	f	e	d	c	b	d
6	g	f	e	b	c	g
7	h	c	f	e	b	b
8	i	h	c	f	e	e
9		i	h	c	f	h
10			i	f	c	c
11				i	f	f
12					i	i

Figure 3.6: Forward Backward Allocation Technique for3x3 matrix transposer [40]

Fig. 3.6 shows the forward-backward register allocation table for the 3x3 matrix transposer whose lifetime table is given in Table 3.1. Step 1 has been performed in section 3.1, where it has been determined that the 3x3 matrix transposer requires 4 registers. In step 2, each variable is input at the cycle corresponding to the beginning of its lifetime. Note that no cycle has more than one input variable in this example. In step 3, the variables are all allocated in forward manner. In step 4, the hashing is performed to

avoid conflicts when backward allocation is performed. For example, the register R_2 is hashed in cycle 11 because this register is occupied by the variable *a* in the cycle 11-9 = 2. In step 5, the variables are allocated backward. For example, the variable *b* is allocated backward to R_3 because this is the only register available for backward allocation. After each of the variables *b*, *c*, and *f* have been allocated backward and then allocated forward until they are dead, the allocation is complete [37].

Figure 3.7 shows the forward backward allocation done for the Eq. (3.2.5). The minimum number of registers required for Eq. (3.2.5) is 11, which was calculated through lifetime analysis as shown in Table 3.1 and/or Figure 3.1. Register R_{12} is used in clock cycle 17 to input variable y_5 again as it's lifetime was already exhausted; but this variable is used again in this clock cycle to create another variable u3 (y5 – y6 = u3). All variables in the Eq. (3.2.5) are input in their respective clock cycles (as shown in Table 3.1), and then carried forward in the subsequent clock cycles to registers R_1 through R_{11} and/or moved backward to any register available for backward allocation until lifetime of that variable is exhausted. For instance, variable a_{12} is input in 0^{th} clock cycle, and is shifted to register R_1 in 1^{st} clock cycle with variable a_{21} input in 1^{st} clock cycle itself. Then a_{12} is shifted to different registers in subsequent clock cycles till it reaches R_{11} at 11^{th} clock. After reaching 11^{th} clock cycle, still life period of 8 is remaining which is to be allocated again. The closest register available in the 12^{th} clock cycle is R_4, hence variable a_{12} is shifted to R_4 in 12^{th} clock cycle. This variable is shifted again in forward fashion till clock 19 from where it is taken as output. Likewise, variable a_{21} is input at clock cycle 1, and it is shifted to R_{11} till 12^{th} clock cycle; life period of 3 is still remaining. Hence, it is allocated backward to the closest available register which is R_1 in this case, and output is taken at 15^{th} clock cycle. All the remaining variables are allocated in the similar fashion.

In Figure 3.7, Forward allocation is shown with black arrows and backward allocation is represented with the help of red arrows. Cells coloured blue represent that output is being taken in that clock cycle from that particular Register.

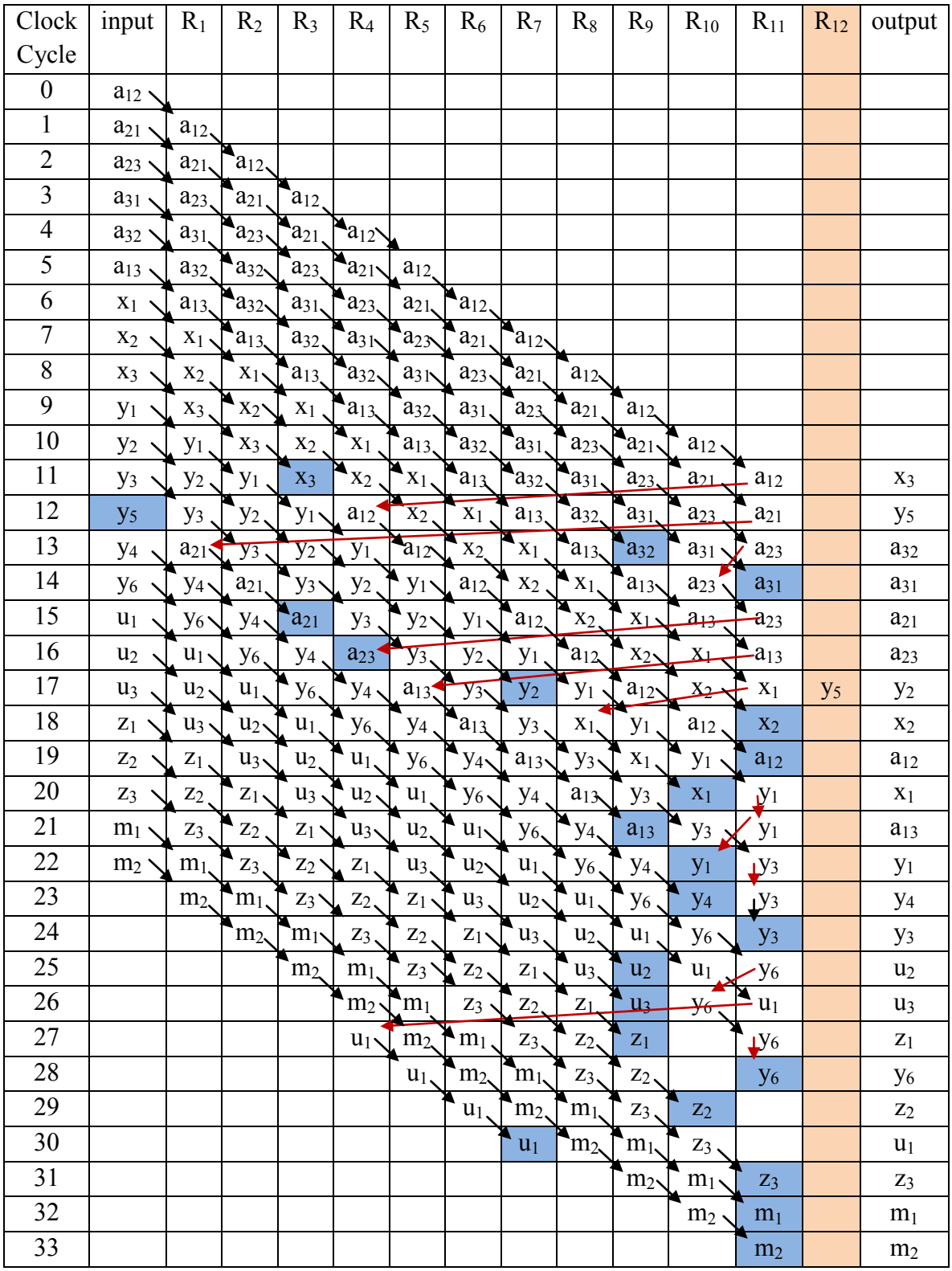

Figure 3.7: Forward-Backward Register Allocation for the polynomial in Eq. (3.2.5) derived for a 3x3 Covariance Matrix

3.4 SEMI-STATIC REGISTER ALLOCATION TECHNIQUE

Register switching and hence power consumption is minimised as the number of variables undergoing transitions from one register to another is reduced or minimised by

grouping the variables based on their lifetimes and then allocating the groups to registers. The proposed heuristic algorithm is described as follows [27-28]:

Let L_i and D_i be the live time and dead time of variable *i*.

Step 1: Find the minimum number of registers using lifetime analysis as done in Figure 3.1.

Step 2: Divide the variables into three categories:-

Group I→Variables with lifetime = P

Group II→Variables with lifetime < P

Group III→Variables with lifetime > P

where P is the Period or total number of variables. For example, in Table 3.1, Period is equal to total number of variables which is 9 (N=9=P). The Period is the number of time steps necessary to input all the variables for one data conversion.

Step 3: Different registers are to be assigned directly to different variables falling in Group I. Then update the available timeslot after this assignment.

Step 4: Variables in category III are to be split into two different variables, one with lifetime less than or equal to period P, and other with remaining lifetime. If any variable formed with lifetime equal to P, again follow Step 3. Unassigned variables will be assigned in next step. Again update the available time slot.

Step 5: Variables in category II, and unassigned variables from above Step 4 are required to be sorted in decreasing order of their lifetimes.

Step 6: From above sorted list, club the variables into further subgroups such that no two variables in one subgroup have overlapping lifetimes. Also the sum of the lifetimes of all variables in one subgroup should be less than period P.

Variables can be sub grouped ideally or non ideally. An ideal case for a subgroup would be when dead time of one variable is live time of another within a subgroup, else it would be a non ideal case.

Step 7: Now sort all the subgroups in decreasing order of their collective lifetimes of the variables clubbed into one subgroup. Allocate registers to these sorted subgroups.

Remaining variables that could not be clubbed into subgroups are allocated in next step. Update the available time slot.

Step 8: Assign the remaining variables in the available time slots in decreasing order of their lifetimes with time slots getting updated after each assignment. Repeat this step until all variables are allocated to registers.

Step 9: If more than one variable gets split, regroup the variables in different way and repeat all the steps from 4 to 8 until minimum number of variables are split.

Consider a 4x4 matrix as follows:

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \quad (3.4.1)$$

Variable	Live	Dead	Period	Step 4	Step 5,6
a	0	9	9	-	R4
b	1	13	12	-	R5
c	2	17	15	-	R2
d	3	21	18	d_1 R_1	d_2 R_6
e	4	10	6	-	R7
f	5	14	9	-	R8
g	6	18	12	-	R6
h	7	22	15	-	R3
i	8	11	3	-	R9
j	9	15	6	-	R4
k	10	19	9	-	R7
l	11	23	12	-	R9
m	12	12	0	-	-
n	13	16	3	-	R5
o	14	20	6	-	R8
p	15	24	9	-	R4

Table 3.3: Execution Trace of Semi-Static Register Allocation Technique for 4x4 Sequential Matrix Transposer [27]

Table 3.3 presents the semi-static register allocation technique for 4x4 sequential matrix transposer, for which the minimum number of registers can be calculated through

lifetime analysis and which comes out to be 9 and the period comes out to be 16 time units. No variable falls under category I, hence Step 3 is inapplicable. In step 4, variable d is split into two other variables viz. d_1 (with lifetime 5-21) and d_2 (with lifetime 3-5) such that it strictly follows condition mentioned in Step 4. The semi-static register allocation for the 4x4 sequential matrix is presented in the Figure 3.8 below.

Clock Cycles	R1	R2	R3	R4	R5	R6	R7	R8	R9	output
0				a						
1				a	b					
2		c		a	b					
3		c		a	b	d_2				
4		c		a	b	d_2	e			
5	d_1	c		a	b	d_2	e	f		d_2
6	d_1	c		a	b	g	e	f		
7	d_1	c	h	a	b	g	e	f		
8	d_1	c	h	a	b	g	e	f	i	
9	d_1	c	h	a, j	b	g	e	f	i	a
10	d_1	c	h	j	b	g	e, k	f	i	e
11	d_1	c	h	j	b	g	k	f	i, l	i
12	d_1	c	h	j	b	g	k	f	l	
13	d_1	c	h	j	b, n	g	k	f	l	b
14	d_1	c	h	j	n	g	k	f, o	l	f
15	d_1	c	h	j, p	n	g	k	o	l	j
16	d_1	c	h	p	n	g	k	o	l	n
17	d_1	c	h	p		g	k	o	l	c
18	d_1		h	p		g	k	o	l	g
19	d_1		h	p			k	o	l	k
20	d_1		h	p				o	l	o
21	d_1		h	p					l	d_1
22			h	p					l	h
23				p					l	l
24				p						p

Figure 3.8: Semi-Static Register Allocation Technique for 4x4 matrix transpose operation [27]

Figure 3.9 shows the register allocation of a characteristic polynomial Equation (3.2.5) of a 3x3 matrix based on proposed heuristic that is, semi-static register allocation technique. In this example, the minimum number of registers obtained from lifetime analysis in Figure 3.5 is 11 and the period is 23 time units. Also in this, variable reuse in different cycles is shown, due to which total number of registers used is 12.

There are no variables in group I and III as lifetimes of all variables in above example are less than 23.Hence step 3 and step 4 are not applicable. According to step 5, all the variables are sorted in decreasing order of their lifetimes. According to step 3 and 4, lifetimes of variables a_{12}, a_{13} being close to the period 23 are directly assigned to registers R_1 and R_2. Scanning the lifetimes of all variables with their birth time and dead time, other variables are directly allotted to registers.

For instance, variables y_1 has lifetime of 13 with birth and dead time of 9 and 22 respectively. No other variable can be accommodated between 0-9 or 22-33 as per the listed lifetime values. Hence it is allocated to register R_3.

According to step 6, following variables are sub grouped:-

'Ideally'→ (a_{21},u_1), (x_2,z_1), (x_3,y_3), (y_2,u_3), (y_1, m_2), (a_{23},u_2) in which dead time of the former variables are same as the birth time of the latter variables in all six subgroups.

'Non ideally'→ (a_{12},z_3), (a_{13},m_1), (a_{32},y_6), (a_{31},z_2) in which dead time of former variables are not same as the birth times of the latter variables in all four subgroups.

Though from lifetime analysis in Figure 3.5, minimum numbers of registers required are 11, but here 12 registers are required to accommodate all the lifetimes of the variables. In the 12^{th} register (R_{12}), variable y_5 is allotted at 12^{th} clock cycle. In R_{11}, at 17^{th} clock cycle, again variable y_5 is allotted because here (y_5-y_6=u_3), this variable is again being reused but accounting for no lifetime as birth time is same as dead time. Figure 3.9 shows the assignment of variables by the proposed method.

Clock	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	output
0	a_{12}												
1	a_{12}			a_{21}									
2	a_{12}			a_{21}	a_{23}								
3	a_{12}			a_{21}	a_{23}						a_{31}		
4	a_{12}			a_{21}	a_{23}	a_{32}					a_{31}		
5	a_{12}	a_{13}		a_{21}	a_{23}	a_{32}					a_{31}		
6	a_{12}	a_{13}		a_{21}	a_{23}	a_{32}				x_1	a_{31}		
7	a_{12}	a_{13}		a_{21}	a_{23}	a_{32}	x_2			x_1	a_{31}		
8	a_{12}	a_{13}		a_{21}	a_{23}	a_{32}	x_2	x_3		x_1	a_{31}		
9	a_{12}	a_{13}	y_1	a_{21}	a_{23}	a_{32}	x_2	x_3		x_1	a_{31}		
10	a_{12}	a_{13}	y_1	a_{21}	a_{23}	a_{32}	x_2	x_3	y_2	x_1	a_{31}		
11	a_{12}	a_{13}	y_1	a_{21}	a_{23}	a_{32}	x_2	x_3,y_3	y_2	x_1	a_{31}		x_3
12	a_{12}	a_{13}	y_1	a_{21}	a_{23}	a_{32}	x_2	y_3	y_2	x_1	a_{31}	y_5	y_5
13	a_{12}	a_{13}	y_1	a_{21}	a_{23}	a_{32}	x_2	y_3	y_2	x_1	a_{31}	y_4	a_{32}
14	a_{12}	a_{13}	y_1	a_{21}	a_{23}	y_6	x_2	y_3	y_2	x_1	a_{31}	y_4	a_{31}
15	a_{12}	a_{13}	y_1	a_{21},u_1	a_{23}	y_6	x_2	y_3	y_2	x_1		y_4	a_{21}
16	a_{12}	a_{13}	y_1	u_1	a_{23},u_2	y_6	x_2	y_3	y_2	x_1		y_4	a_{23}
17	a_{12}	a_{13}	y_1	u_1	u_2	y_6	x_2	y_3	y_2,u_3	x_1	y_5	y_4	y_2
18	a_{12}	a_{13}	y_1	u_1	u_2	y_6	x_2,z_1	y_3	u_3	x_1		y_4	x_2
19	a_{12}	a_{13}	y_1	u_1	u_2	y_6	z_1	y_3	u_3	x_1	z_2	y_4	a_{12}
20	z_3	a_{13}	y_1	u_1	u_2	y_6	z_1	y_3	u_3	x_1	z_2	y_4	x_1
21	z_3	a_{13}	y_1	u_1	u_2	y_6	z_1	y_3	u_3		z_2	y_4	a_{13}
22	z_3	m_1	y_1,m_2	u_1	u_2	y_6	z_1	y_3	u_3		z_2	y_4	y_1
23	z_3	m_1	m_2	u_1	u_2	y_6	z_1	y_3	u_3		z_2	y_4	y_4
24	z_3	m_1	m_2	u_1	u_2	y_6	z_1	y_3	u_3		z_2		y_3
25	z_3	m_1	m_2	u_1	u_2	y_6	z_1		u_3		z_2		u_2
26	z_3	m_1	m_2	u_1		y_6	z_1		u_3		z_2		u_3
27	z_3	m_1	m_2	u_1		y_6	z_1				z_2		z_1
28	z_3	m_1	m_2	u_1		y_6					z_2		y_6
29	z_3	m_1	m_2	u_1							z_2		z_2
30	z_3	m_1	m_2	u_1									u_1
31	z_3	m_1	m_2										z_3
32		m_1	m_2										m_1
33			m_2										m_2

Figure 3.9: Semi Static Register allocation Technique for the polynomial equation (3.2.5) derived from a 3x3 covariance matrix

CHAPTER 4

SIMULATION RESULTS AND DISCUSSIONS

Power consumption of any register depends on switched capacitance of that register.

$$C_{switched} = C_L \times \alpha \quad (4.1)$$

Where $C_{switched}$ is the switched capacitance, C_L is the Load Capacitance and α is the switching activity of driver.

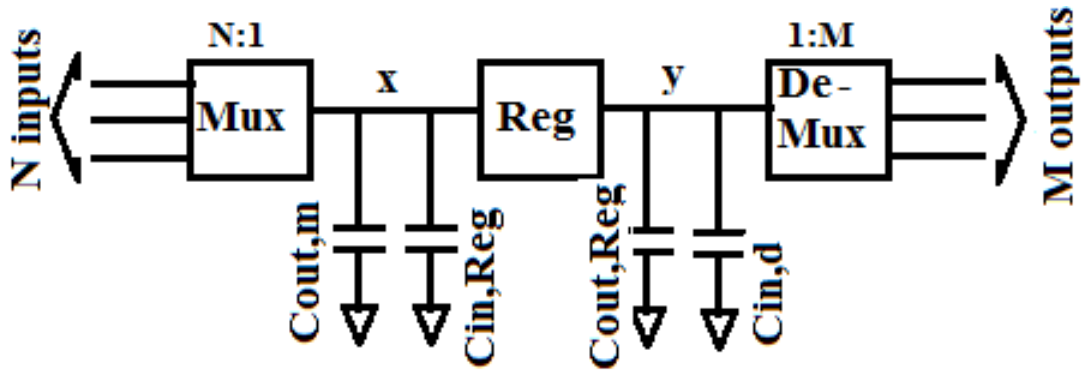

Figure 4.1 Demonstration of switching in a Register to calculate Power Dissipation

Figure 4.1 shows N:1 multiplexer placed at the input of the Register (Reg) which passes one value amongst 'N' number of inputs to node 'x'; while 1:M De-multiplexer is placed at output of the Register which receives output value of Register from node 'y' and yields output.

Power consumed by register 'Reg' depends on:

P(Reg) => {Switching(x)*(Cout, m + Cin, Reg)} + {Switching(y)*(Cout, Reg + Cin, d)}

(4.2)

where, (Cout, m) is output capacitance of multiplexer 'mux'

(Cin, Reg) is input capacitance of Register 'Reg'

(Cout, Reg) is the output capacitance of Register 'Reg'

(Cin, d) is input capacitance of De-multiplexer 'Demux'

Assuming: switching (x) = switching (y) (4.3)

Using equation (4.3), we can further solve equation (4.2) as follows:

$$P(Reg) = switching(y) \cdot C_{total} \quad (4.4)$$

where C_{total} is fixed for a given library.

From above equations we can conclude that minimising switching activity at the register output will minimise the power consumption regardless of the specific load seen at the register output [43].

4.1 SIMULATION PLATFORM AND IMPLEMENTATION TOOLS

In this chapter, various tools and platforms used for the analysis of the register allocation techniques for PCA along with their results have been discussed. As stated in the initial chapters, that hardware (ASIC/FPGA) implementation of PCA is an important task, which has not been undertaken yet. The only restriction of hardware implementation is the large dimensional input sequence generation, which differs with respect to different applications. In this dissertation, the author has worked on the PCA implementation with respect to both, the forward backward register allocation technique and Semi- Static Register allocation technique, and drew comparison between the both in terms of area occupied and dynamic power consumed. For that, initial step was to write a Verilog code for both the above stated techniques. Then to verify the functionality of both the codes, they were simulated through ISE Simulator (ISim). Then those codes were converted to '.v' extension for their use in RC Encounter tool of Cadence. In RC Adoption Kit (14.20 version), RC was invoked by setting various attributes of libraries; and those HDL codes were called. After elaborating the commands, generic synthesis and mapped synthesis were performed, and power and area reports were dumped out.

4.1.1 ISE SIMULATOR for XILINX

It is a verification and simulation tool for hardware description language (HDL) like VHDL, Verilog etc. In other words it can be said that it was used to verify and simulate HDL codes prepared by the programmer. The verification is done by performing timing simulations to indicate the results. Fig. 4.2 shows the basic steps for simulating a design using ISE simulator.

Xilinx ISE is a tool that was used to implement the design (HDL code) on the FPGA. In other words this tool interfaces and verifies the HDL codes for hardware implementation compatibility on FPGA devices. After the successful completion of the synthesis process, the RTL schematic and technology schematic of the synthesized code

can be generated. Physical Implementation (Mapping, Placing and Routing) is not necessary for our work.

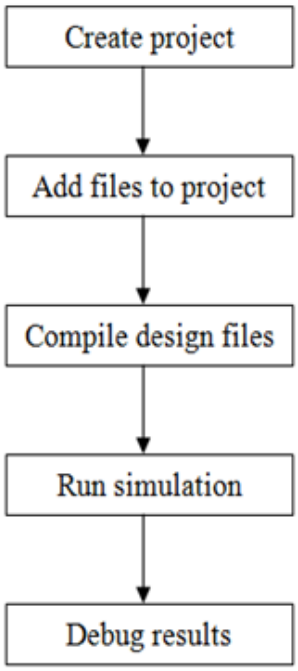

Figure 4.2: Design flow of ISE Simulator

4.1.2 RTL COMPILER (CADENCE)

With shrinking design nodes, a significant portion of the delays are contributed by the wires rather than the cells. Traditional synthesis tools use fan-out-based wire-load models to provide wire delay information, which has led to significant differences in quality of results (QoR) between the "synthesis" and "implementation" tools. RTL Compiler Physical (RCP) as a tool allows the user to integrate the "physical" information much earlier in the flow, and this provides a good level of down-stream predictability that is superior to using wire-load models. Predictability will enable the designer to gauge very early in the flow how the design will perform after place and route. This, in turn, will help the designer reduce front-end to back-end hand-off iterations.

After setting up of design constraints and design environments synthesis was done in RC tool (cadence). RTL Compiler (RC) Ultra is a powerful tool for logic synthesis and analysis for digital designs. It is fully compatible with all other Cadence Tools especially with Cadence Encounter which is mainly used for physical design automation (floor planning, placement and routing). The technology node on which the project was implemented was 180 nm technology. Direct verilog code was synthesised in RC tool instead of taking up the traditional method of creating a schematic in

Virtuoso and then moving ahead with the power and area calculations. This is because, the schematic is too complex to create and analyse which is given in next chapter; RTL views of the implemented techniques.

4.2 RESULT

Practically, to compare the power consumed in both the techniques, Verilog codes were written. After setting up of design constraints and design environments synthesis was done in RC tool (cadence) .Using report command, power and area report of both the techniques were dumped out as shown in Table 4.1.

GENERIC	**POWER (nW)**		**Cells**	**Leakage Power**	**Dynamic Power**	**Total Power**
		S.S.	305	23.593	76212.227	76235.820
		F.B.	335	104.263	176644.245	176748.508
	AREA		**Cells**	**Cell Area**	**Net Area**	**Total Area**
		S.S.	305	1098	0	1098
		F.B.	335	2633	0	2633
MAPPED	**POWER (nW)**		**Cells**	**Leakage Power**	**Dynamic Power**	**Total Power**
		S.S.	140	14.335	41920.274	41934.609
		F.B.	341	66.976	302609.513	302676.489
	AREA		**Cells**	**Cell Area**	**Net Area**	**Total Area**
		S.S.	140	300	0	300
		F.B.	341	1990	0	1990

Table 4.1: Power consumption and area occupied figures for both the techniques (SS: Semi Static Register Allocation Techniques; FB: Forward Backward Register Allocation Technique) in case of both generic and mapped synthesis.

In generic synthesis, RTL optimizations are done by converting the code into generic cells. These generic cells are a part of tool default library, which are not related to any specific technology, hence can be called as technology independent. In this, it can be observed from Table 4.1 that a substantial decrease in power of 56.867% can be seen. Where the cell count has decreased just by 30, the area is reduced by 56.66%. This is because cells comprised of various sets of logic gates, comparators etc and reduction in one cell leads to reduction in 3-4 components which that cell was comprised of.

In case of mapped synthesis, generic cells in the design are mapped to the technology dependent cells defined in the (180 nm) technology library after logic optimisation. Boolean optimisation (technology independent), Technology mapping, and Technology dependent gate optimisation are the functions performed in this mapped synthesis, and hence quite a large reduction of power and area is observed in this case. The cell count is decreased by 201. But the major achievement here is the power reduction which is of significant value of 86.145%.

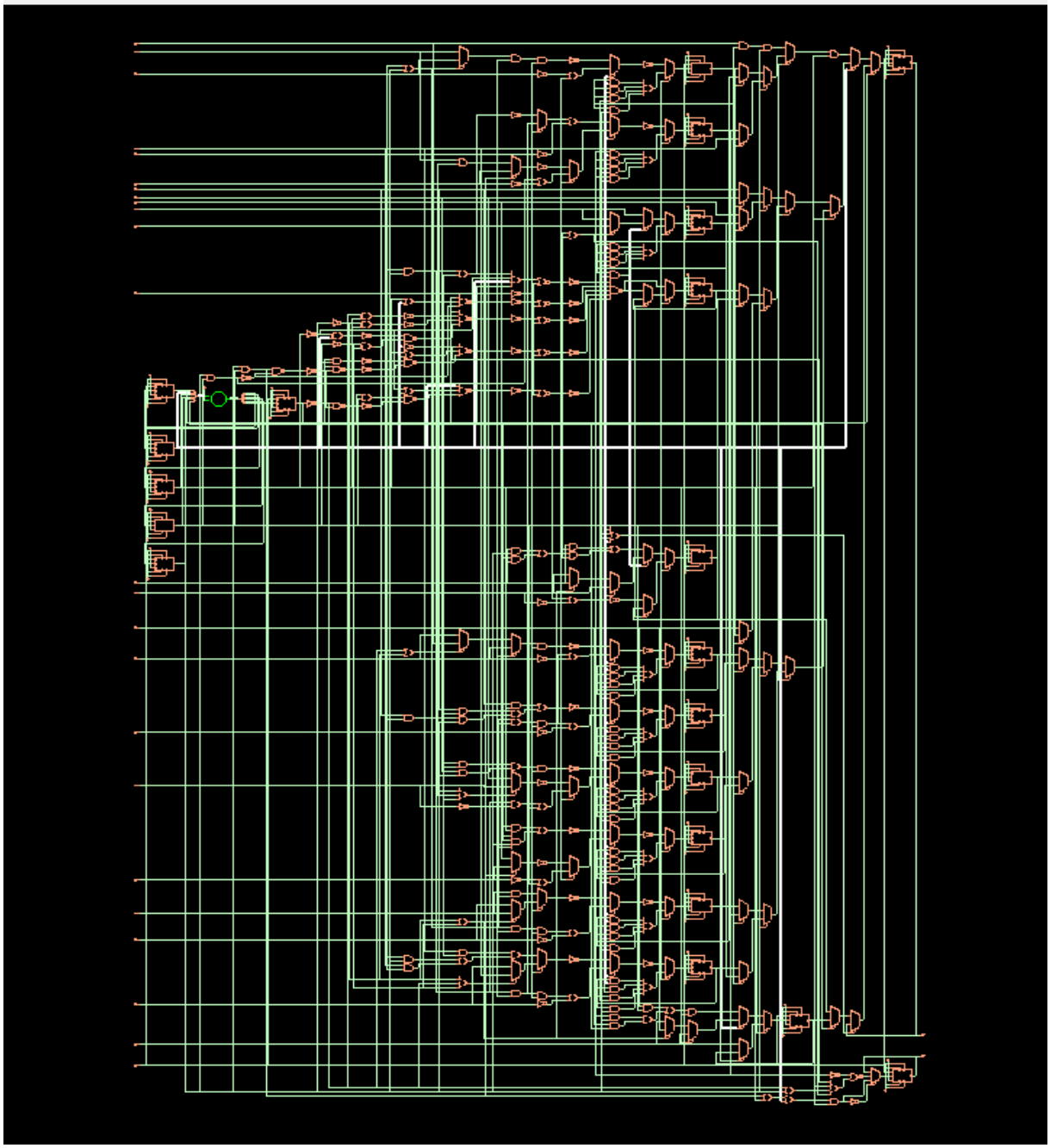

Figure 4.3: GUI View for Generic Synthesis of Semi-Static Register Allocation Technique

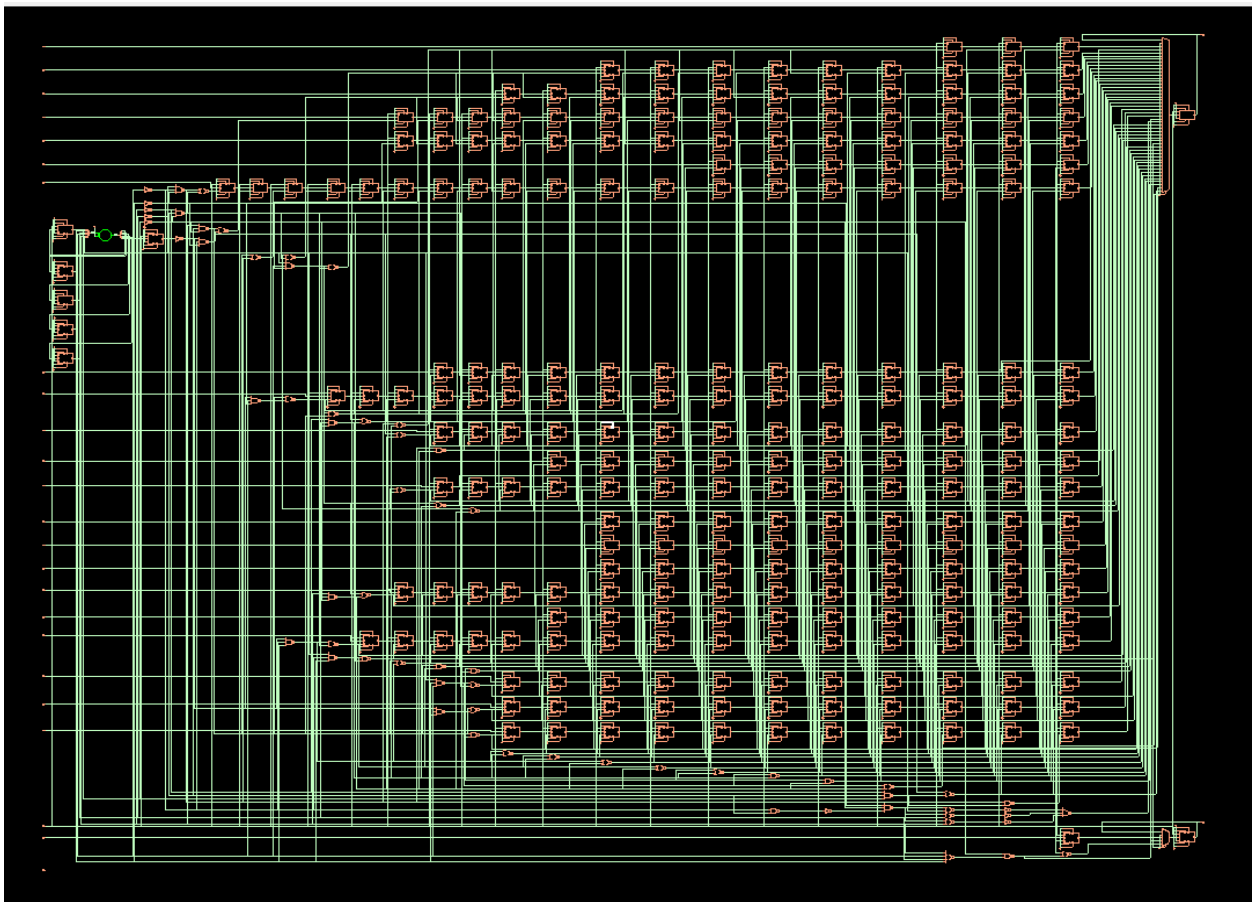

Figure 4.4: GUI View for Generic Synthesis of Forward-Backward Register Allocation Technique

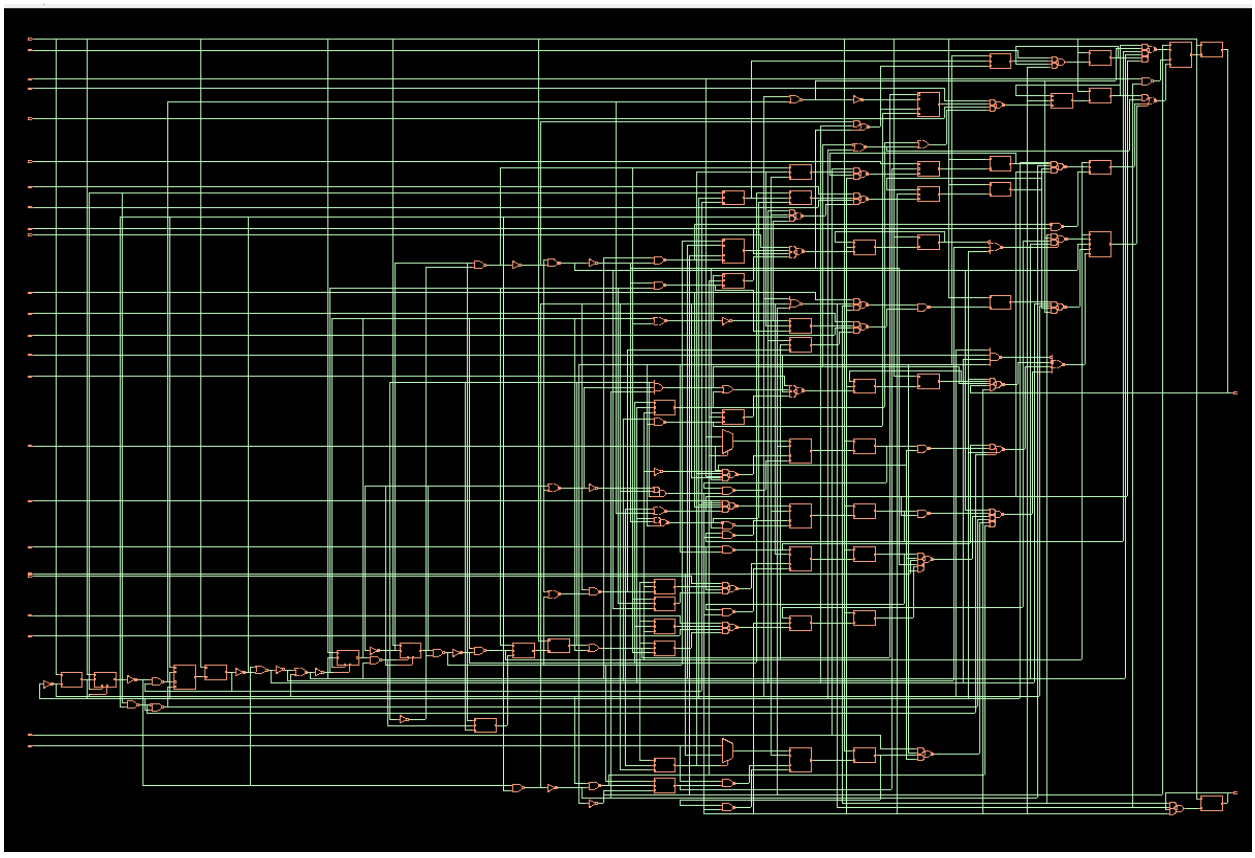

Figure 4.5: GUI View for Mapped Synthesis of Semi-Static Register Allocation Technique

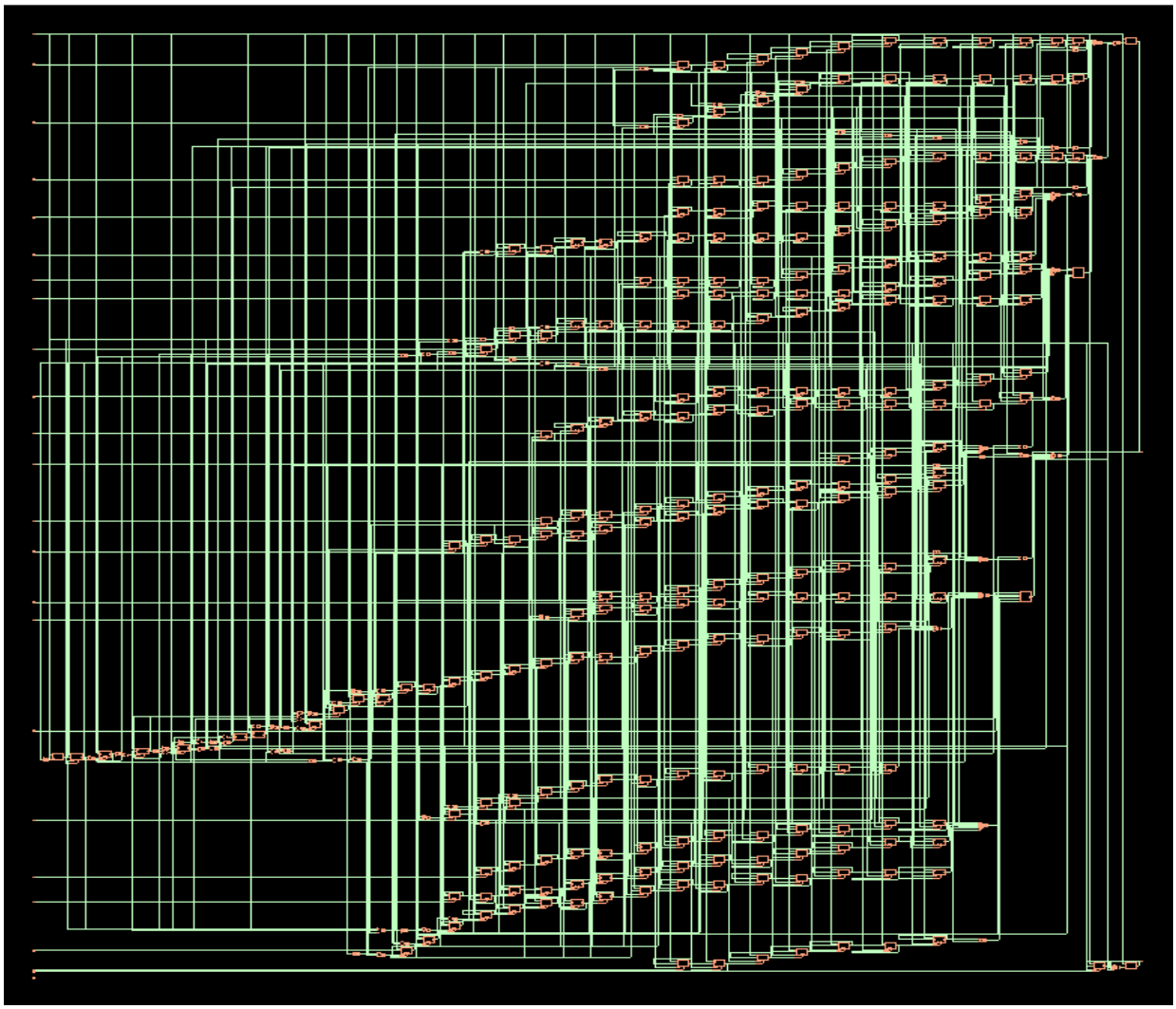

Figure 4.6: GUI View for Mapped Synthesis of Forward-Backward Register Allocation Technique

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

Modifications are required in register allocation schemes at the instruction level itself in order to save more power and area during implementation of PCA and other computationally intensive techniques. Hence our dissertation presented the instruction level register allocation techniques, which when combined with other efficient techniques in future, can altogether yield an efficient global level register allocation technique. It is thus imperative to minimise register switching which eventually leads to lower dynamic power consumption. The semi-static allocation technique proposed in this dissertation reuses variables in an efficient manner thereby significantly reducing power consumption by minimising the switching activity of the registers regardless of the load. The absence of register switching with each clock cycle amongst registers, resulted in a simpler verilog code for the proposed technique, where in a single dimensional array of register was used for register allocation unlike in the forward backward allocation technique, where a two-dimensional matrix of register was required. This difference was reflected in the form of a simpler RTL schematic with few number of cells and connecting wires which results in smaller silicon area; as compared to the RTL schematic obtained for the traditional technique, which relatively more number of cells and connecting wires, which ultimately led to larger silicon area occupancy.

5.2 FUTURE SCOPE

PCA being a versatile signal processing technique has immense applications in the field of multimedia signal processing. In current technological scenario, VLSI is also moving towards enhanced production of multimedia chip. Furthermore, a state of the art IP core requires inbuilt signal processing applications including PCA. The insights gained from this study can be further used for front end design of the multimedia chips. Future scope of PCA implementations is to implement PCA on real time hardware platform with improved performance, and less computational complexity.

Results obtained using both generic and mapped synthesis-confirm the efficacy of the proposed technique regardless of the implementation scheme and technology node. In future, this technique can be combined with some efficient global register allocation schemes so that two way benefits can be harvested.

REFERENCES

[1] Haykin S. *Neural Networks and Learning Machines*. New York: Prentice Hall/Pearson, 2009.

[2] Bhatti FA, Rowe GB and Sowerby KW (2002). Spectrum Sensing using Principal Component Analysis (PCA), *IEEE Wireless Communications and Networking Conference (WCNC)* [Paris: April 2002], pp. 725-730.

[3] Bhatti FA *et al*. (2014). On the use of eigenvectors for signal detection and classification in multiple antenna cognitive radios, *25th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)* [Washington, DC, USA: September 2014], pp. 753-757.

[4] Das A *et al*. (2008). An FPGA based Network Intrusion Detection Architecture, *IEEE Transactions on Information Forensics and Security,* 3(1), pp. 118-132.

[5] Sehgal S et al. (2014), "Data Analysis using Principal Component Analysis." in *International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom)* [Greater Noida, India: November 2014], pp. 45-48.

[6] Yan L (2006). A PCA-Based PCM Data Analyzing Method for Diagnosing Process Failure, *IEEE Transactions on Semiconductor Manufacturing,* 19(4), 404-410.

[7] Hsieh YL et al. (2010). Wafer Sort Bitmap Data Analysis using the PCA-based approach for Yield Analysis and Optimisation, *IEEE Transactions on Semiconductor Manufacturing,* 23(4), 493-502.

[8] Storch H and Zwiers F. *Statistical Analysis in Climate Research*. Cambridge: Cambridge University Press, 1999.

[9] Carvajal G, Valenzuela W and Figueroa, M (2008). Subspace-based face recognition in analog VLSI, *Advances in Neural Information Processing Systems,* pp. 225-232.

[10] Turk M and Pentland A (2007). Eigenfaces for Recognition, *Journal of Cognitive Neuroscience* , 3(1), 71-86.

[11] Chang CS *et al*. (2010). Human Face Recognition System using Modified PCA algorithm and ARM platform, *IEEE International Symposium on Computer, Communication, Control and Automation* [Tainan, Taiwan: May 2010], pp.294-297.

[12] El-Bakry HM and Zhao, Q (2006). Fast Neural Implementation of PCA for Face Detection, *IEEE International Joint Conference on Neural Networks* [Vancouver, BC, Canada: July 21, 2006], pp 806-811.

[13] Bashir F, Khokhar A and Schonfeld D (2005). Automatic object trajectory-based motion recognition using Gaussian mixture models, *IEEE International Conference on Multimedia and expo [*ICME, Amsterdam, Netherlands: July 2005]*,* pp. 1532-1535.

[14] Yeung KY and Ruzzo WL (2001). Principal Component Analysis for clustering gene expression data, *Bioinformatics*, 17(9), pp. 763-774.

[15] Bhuvaneswari KS and Geetha P (2016). Semantic feature based classification of Brain MRI using PCA and PNN, *IEEE International Conference on Electrical, Electronics, and Optimization Techniques* [ICEEOT, Chennai, India: March 2016]*,* pp. 2700-2706.

[16] Zamani J, Moghaddam AN and Rad HS (2012). MRI reconstruction through Compressed Sensing using Principle Component Analysis (PCA), *20th IEEE Iranian Conference on Electrical Engineering* [ICEE, Tehran, Iran: May 2012], pp. 1608-1611.

[17] Chavan A, Kolte M (2015). Improved EEG signal processing with wavelet based multiscale PCA algorithm, *IEEE International Conference on Industrial Instrumentation and Control* [ICIC, Pune, India: May 2015], pp. 1056-1059.

[18] Stamkopoulos T *et al*. (1998). ECG Analysis using Non Linear PCA Neural Networks for Ischemia Detection, *IEEE Transactions on Signal Processing,* 46(11), 3058-3067.

[19] Chen TC, Liu W and Chen LG (2008). VLSI architecture of leading eigen vector generation for on-chip PCA spike sorting system, *30th Annual conference of the IEEE Engineering in Medicine and Biology Society* [Vancouver, Canada: August 2008], pp. 3192-3195.

[20] Josth R *et al*. (2012). Real-time PCA calculation for spectral imaging (using SIMD and GP-GPU), *IEEE Journal Real Time Image Processing,* 7, 95-103.

[21] Xie Z and Edwards D (2013). Computational Performance Optimisation For Statistical Analysis of the Effect of Nano-CMOS Variability on Integrated Circuits, *Hindawi: VLSI Design,* article id: 984376*,* 1-22.

[22] Cheng L. *et al.* (2014). Statistical Timing and Power Analysis of VLSI considering non-linear dependence, *Integration: the VLSI Journal*, 1-12.

[23] Sanger TD (1989). Optimal Unsupervised learning in a Single-layer Linear Feedforward Neural Network, *Neural Networks*, 2(6), 459-473.

[24] Rubner J and Tavan P (1989). A self-Organizing Network for Principal Component Analysis, *Europhysics Letters*, 10(7), 693-698.

[25] Rao YN and Principe JC (2002). Robust on-line Principal Component Analysis based on a fixed-point approach, *IEEE International Conference on Acoustics, Speech, and Signal Processing* [ICASSP, Orlando, USA: May 2002], vol. 1, pp. I-981-I-984.

[26] Han D *et al*. (2004). Real-Time PCA (Principal Component Analysis) implementation on DSP, *Proceedings IEEE International Joint Conference on Neural Networks* [Budapest, Hungary, July 2004]*,* pp. 2159-2162.

[27] Wu TY and Lin YL (1996). Register minimisation beyond sharing among variables, *IEEE Transactions on Compiler-Aided Design of Integrated Circuits and Systems*, 15, 1583-1587.

[28] Srivatsan K, Chakrabarti C and Lucke L (1995). Low Power data format converter design using semi-static register allocation, *Proceedings IEEE International Conference on Computer Design (ICCD'95): VLSI in computer and Processors* [Austin, TX, USA: October 1995], pp. 460-465.

[29] Srivatsan K, Chakrabarti C and Lucke L (1999). A new register allocation scheme for low-power data format converters, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(9), 1250-1253.

[30] Pereira FMQ *et al*. Register allocation via coloring of chordal graphs. Yi, K. (ed.), *APLAS 2005 LNCS.* Springer, Heidelberg 2005,vol. 3780, pp.315-329.

[31] Brisk P *et al*. (2006). Optimal register sharing for high-level synthesis of SSA form programs, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25, 772-779.

[32] Brisk P, Verma AK and Lenne P (2010). An Optimal Linear-Time Algorithm in High Level Synthesis Using SSA Form*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29, 1096-1109.

[33] Pereira FMQ and Palsberg J (2008). Register allocation by puzzle solving, *PLDI '08 Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation* [Tucson, AZ, USA: June 2008], pp. 1-18.

[34] Pereira FMQ and Palsberg J (2009). SSA Elimination after Register Allocation, de Moor O., Schwartzbach M.I. (eds), *Compiler Construction CC 2009. Lecture Notes in Computer Science,* Springer, Berlin, Heidelberg, 2009, vol. 5501.

[35] Hack S and Goos G (2008). Copy coalescing by graph recoloring, *PLDI, ACM SIGPLAN Conference on Programming Language Design and Implementation* [Tucson, AZ, USA: June 2008], pp. 227-237.

[36] Pereira FMQ and Palsberg J (2006). Register allocation after classical SSA elimination is NP-complete, *FOSSACS'06 Proceedings of the 9th European joint conference on Foundations of Software Science and Computation Structures* [Vienna, Austria, March 2006], pp. 79-93.

[37] Braun M and Hack S (2009). Register Spilling and Live-Range splitting for SSA-Form Programs, *CC '09 Proc. of the 18th International Conf. on Compiler Construction: Held as Part of the joint European Conferences on Theory and Practice of Software* [ETAPS '09, York, UK: March 2009], pp. 174-189.

[38] Pereira FMQ and Palsberg J (2010). Punctual Coelescing, *CC'10 Proceedings of the 19th Joint European Conference on Theory and Practice of Software, International conference on compiler construction* [ETAPS'10, Cyprus: March 2010], pp. 165-184.

[39] Lintzmayer CN, Mulati MH and da Silva AF (2011). Register Allocation with graph coloring by Ant Colony Optimization, *30th International Conference of the Chilean Computer Science Society* [Curico, Chile: November 2011], pp. 247-255.

[40] Lintzmayer CN, Mulati MH and da Silva AF (2015). The hybrid ColorAnt-RT Algorithms and an Application to Register Allocation, *Inteligencia Artificial, [S.I.]*, 18(55), 81-111.

[41] Parhi KK. *VLSI Digital Signal Processing Systems: design and implementation*. John Wiley, 1999.

[42] Parhi KK (1994). Calculation of minimum number of registers in arbitrary life time

chart, *IEEE Trans. In Circuits and Systems-II,* 41(6), 434-436.

[43] Parhi KK (1992). Systematic synthesis of DSP data format converters using lifetime analysis and forward-backward register allocation, *IEEE Transactions On Circuits and Systems – II*, 39(7), 423-440.

[44] Chang JM and Pedram M (1995). Register allocation and binding for low power, *Proceedings of the 32nd annual ACM/IEEE Design Automation Conference* [San Fransico, USA: June 1995], pp. 29-35.

LIST OF PUBLICATIONS BY THE CANDIDATE

1. Sukhmani K. Thethi and Ravi Kumar, ”Area and Power Efficient Register Allocation Technique for the implementation of PCA,” in *IEEE International Conference on Signal Processing, Computing and Control (ISPCC)*, 2017. (Accepted)

2. Sukhmani K. Thethi and Ravi Kumar, ”A Power and Area Efficient Register Allocation Technique for the implementation of Principal Component Analysis,” in *HINDAWI: VLSI Design*. (Communicated)

Sukhmani2_13-07-2017

by Sukhmani Kaur

FILE	PLAGIARISM_2.DOCX (860.18K)		
TIME SUBMITTED	13-JUL-2017 04:36PM	WORD COUNT	11698
SUBMISSION ID	830623084	CHARACTER COUNT	57760

Sukhmani2_13-07-2017

ORIGINALITY REPORT

%21	%11	%17	%5
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	K.K. Parhi. "Systematic synthesis of DSP data format converters using life-time analysis and forward-backward register allocation", IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 1992 Publication	%1
2	Dongho Han, Y.N. Rao, J.C. Principe, K. Gugel. "Real-time PCA (principal component analysis) implementation on DSP", 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), 2004 Publication	%1
3	www.netlab.tkk.fi Internet Source	%1
4	Alok Choudhary. "An efficient FPGA implementation of principle component analysis based network intrusion detection system", Proceedings of the conference on Design automation and test in Europe - DATE 08 DATE 08, 2008 Publication	%1