

DIGITAL ELECTRONICS

**INVESTIGATIONS ON THE DEVELOPMENT OF AN ANN
MODEL & VISUAL MANIPULATION APPROACH FOR 2-D
DFT COMPUTATION IN IMAGE PROCESSING**

Thesis submitted by

R. GOPIKAKUMARI

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

DOCTOR OF PHILOSOPHY

Under the

FACULTY OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
KOCHI-682 022, KERALA, INDIA

AUGUST 1998

CERTIFICATE

This is to certify that the thesis entitled " **INVESTIGATIONS ON THE DEVELOPMENT OF AN ANN MODEL & VISUAL MANIPULATION APPROACH FOR 2-D DFT COMPUTATION IN IMAGE PROCESSING** " is a bonafide record of the research work carried out by *R.Gopikakumari* under my supervision and guidance in the *Department of Electronics, Cochin University of Science and Technology* and that no part there of has been presented for the award of any other degree.

C.S.S. Sridhar

Kochi 682022
31st July 1998

Dr. C. S. Sridhar
(Supervising Teacher)
Formerly Professor & Head
Department of Electronics
Cochin University of Science and Technology

DECLARATION

I hereby declare that the work presented in the thesis entitled " INVESTIGATIONS ON THE DEVELOPMENT OF AN ANN MODEL & VISUAL MANIPULATION APPROACH FOR 2-D DFT COMPUTATION IN IMAGE PROCESSING ", is based on the original research work carried out by me under the supervision and guidance of Dr. C. S. Sridhar in the Department of Electronics, Cochin University of Science and Technology and that no part there of has been presented for the award of any other degree.

Kochi 682 022
31st July 1998

Ropikakumari
(R.Gopikakumari)

ACKNOWLEDGEMENTS

First and foremost I would like to express my profound gratitude to my research guide Dr. C. S. Sridhar, Former Professor and Head, Department of Electronics, Cochin University of Science and Technology, under whose supervision and guidance I began my forays into the fields of Signal Processing and Neural Networks. This thesis is the outcome of education, able guidance, encouragement and inspiration I received from him. I wish to express my sincere gratitude to his wife and daughter for the patience in continueing Prof. Sridhar at Cochin till the research work is over.

Sincere thanks are due to Dr. K. G. Nair, Former Professor and Head, Department of Electronics and Director, Sophisticated Test and Instrumentation Centre, Cochin for permitting me to work in the department for the Doctoral studies and also for the valuable suggestions and encouragements during the tenuare of this work.

Sincere thanks are due to Mr. V. Chander, Director, NPOL, Cochin and Dr. A. Unnikrishnan & Dr. C. Karthikeyan, Scientists in NPOL, Cochin for the valuable suggestions and also for reviewing the thesis. Also, I would like to express my gratitude to all the Staff of NPOL who helped me directly or indirectly in completing the research work.

My sincere thanks are due to Dr. N. Balakrishnan, Chairman, SERC, IISc, Bangalore for the fruitful discussions and the helping hand offered me when I was stuck with research problems. Also, I would like to express my sincere gratitude to Dr. K. R. Ramakrishnan, Professor, Dept. of Electrical Engg., IISc, Bangalore for the valuable discussions and suggestions.

I like to express my sincere thanks to Mrs. A. P. Preethy, my colleague (on leave) and graduate student, School of Applied Sciences, Nanyang Technological University, Singapore as well as to Dr. Damu Radhakrishnan and Dr. Chong Man Nang , Faculty members of the School of Applied Sciences, Nanyang Technological University, Singapore for the valuable help offered me in doing part of the work there during the visit for participating in the conference ISIC-97.

Sincere thanks are due to Dr. K. Babu Joseph, Vice-Cancellor, Dr. V. P. N. Nampoori & Dr. Nandakumar, Professors of International School of Photonics for the valuable discussions and encouragement during the tenure of the research work.

Thanks are also due to Dr. A. Krishnan, Deputy Director, NAL, Bangalore, Dr. D. Sreenivasan, Eminent Professor, IIT, Madras for the discussions and encouragement in the course of the research work.

I take this opportunity to express my sincere thanks to all my colleagues, research scholars, students, authorities of the university and friends who helped me directly or indirectly in carrying out the research work to a fruitful level. Special thanks are due to Mr. Deepu Rajan, my colleague, for the valuable suggestions and timely help, offered me, during the period of the research work. I would like to express special thanks to Mrs. Prasanna kumari, Technical Assistant, and Mrs. Beena, Library Assistant for the timely help and encouragement.

Last and not the least, I express my profound gratitude to my husband, daughter, my parents, sister and the in-laws for the patience and support during the years when I was pursuing my studies.

I am indebted to one and all, who have enlightened me, in this crucial period of life.

R. Gopikakumari

ABSTRACT

This thesis is an outcome of the investigations carried out on the development of an Artificial Neural Network (ANN) model to implement 2-D DFT at high speed. A new definition of 2-D DFT relation is presented. This new definition enables DFT computation organized in stages involving only real addition except at the final stage of computation. The number of stages is always fixed at 4. Two different strategies are proposed. 1) A visual representation of 2-D DFT coefficients. 2) A neural network approach.

The visual representation scheme can be used to compute, analyze and manipulate 2-D signals such as images in the frequency domain in terms of symbols derived from 2x2 DFT. This, in turn, can be represented in terms of real data. This approach can help analyze signals in the frequency domain even without computing the DFT coefficients.

A hierarchical neural network model is developed to implement 2-D DFT. Presently, this model is capable of implementing 2-D DFT for a particular order N such that $(N)_4 = 2$. The model can be developed into one that can implement the 2-D DFT for any order N upto a set maximum limited by the hardware constraints. The reported method shows a potential in implementing the 2-D DFT in hardware as a VLSI / ASIC.

CONTENTS

Chapter 1	Introduction	1
1.1	Digital Signal Processing	1
1.1.1	Transforms	3
1.1.2	Discrete Fourier Transform	4
1.1.3	One dimensional DFT	4
1.1.4	Matrix representation of DFT	4
1.1.5	Twiddle factor	5
1.1.6	Two dimensional DFT	6
1.1.7	Algorithms to implement 2-D DFT	6
	1.1.7.1 Direct computation	6
	1.1.7.2 Row-column decomposition	7
	1.1.7.3 Vector-radix Fast Fourier Transform	7
1.2	Digital Image Processing	8
1.2.1	Image enhancement	9
1.2.2	Image restoration	10
1.2.3	Image compression	11

1.2.4	Image segmentation	11
1.2.5	Image description and representation	12
1.2.6	Image recognition and interpretation	12
1.2.7	Computed imaging	13
1.3	Neural Networks	13
1.3.1	Definition of Neural Networks	13
1.3.2	Neural Network Models	13
1.3.2.1	Perceptron Model	15
1.3.2.2	Multilayer Perceptron: Backpropagation Network	15
1.3.2.3	Hopfield Network Model	16
1.3.2.4	Adaptive Resonance Theory (ART) Model	17
1.3.2.5	Self Organizing Map (SOM)	18
1.3.2.6	Neocognitron Model	18
1.3.2.7	Modified Neocognitron Model	19
1.3.2.8	Cellular Neural Networks	20
1.4	Motivation for the present work	21
1.4.1	Why Neural Networks in Signal Processing ?	21
1.4.2	Problems in present day DFT computation	22
1.5	Brief sketch of the present work	23
Chapter 2	Review of past work in the field	24
2.1	Discrete Fourier Transform	25
2.2	Neural Networks	29
2.3	Applications in Image Processing	39
2.4	Conclusion	40
Chapter 3	Visual manipulation of symbols for DFT	41
3.1	Basic theory	42
3.2	Derivation of the primitive symbols	44
3.3	Pictorial representation	47
3.3.1	Direct method	47
3.3.2	Analysis of the pictorial representation	57

3.3.2.1	N= 4	57
3.3.2.2	N = 6	59
3.3.2.3	N = 8	62
3.3.2.4	N = 10	64
3.3.3	Results of the analysis	66
3.3.4	Indirect method	68
3.3.4.1	Construction of the pictorial representation for $((N))_4 = 2$	68
3.3.4.2	The Algorithm	69
3.4	Theorems & Definitions	71
3.5	Conclusion	74
Chapter 4	Neural Network Model for 2-D DFT Computation	75
4.1	Reasons for choosing the present Architecture	76
4.2	Basics	77
4.2.1	The derivation of the structure	77
4.2.2	Modifying patterns of connectivity as a function of experience	78
4.2.3	Structure of the network	79
4.3	Design of parallel distributed architecture for 6x6-point DFT	79
4.3.1	Outline of the model	80
4.3.2	The Algorithm	83
4.3.3	Example	86
4.4	Parallel Distributed Model for NxN-point DFT, $((N))_4 = 2$	88
4.5	Simulation results	95
4.6	Complexity of the model	100
4.6.1	Computational complexity	100
4.6.2	Hardware complexity	101
4.6.3	Memory Requirement	102
4.7	Conclusion	102
Chapter 5	Discussions and conclusions	103
5.1	Visual representation of 2-D DFT	103
5.1.1	Computation / representation	104

5.1.2	Ease of generation	104
5.1.3	Simplicity for users	104
5.1.4	Properties exhibited by the representation	104
5.1.5	Memory requirement	105
5.1.6	Computational complexity	105
5.1.7	Learning	106
5.1.8	Real-time processing	106
5.2	Neural Network model	106
5.2.1	Speed / complexity trade off	107
5.2.2	Computational merits	107
5.2.3	Learning	108
5.2.4	Hardware implementation	108
	5.2.4.1 VLSI implementation aspects	108
	5.2.4.2 Suggestions to hardware implementation of the Neural Network Model	109
5.3	Some practical applications of the present work	111
5.4	Scope of further work in the field	111
5.5	Conclusion	112
Appendix A		113
Appendix B		119
References		127
Index		138
List of Publications of the author		141

INTRODUCTION

When the only tool you have is a hammer, every problem you encounter tends to resemble a nail

---Source Unknown

1.1 DIGITAL SIGNAL PROCESSING

Simply stated, a signal is any medium for conveying information, and signal processing is concerned with the extraction of that information. Thus ensembles of time-varying voltages, the density of silver grains on a photographic emulsion, or list of numbers in the memory of a computer all represent examples of signals. A typical signal processing task involves the transfer of information from one signal to another. For example, a photograph might be scanned, sampled and stored in the memory of a computer. In this case, the information is transferred from available silver density, to a beam of visible light, to an electrical waveform, and finally to a sequence of numbers which, in turn, are represented by an arrangement of magnetic domains on a computer disk.

Whatever their form, signals are of interest only because of the information they contain. In general, we may say that signal processing is concerned with basic tasks information rearrangement and information reduction. Image scanning, image enhancement, image deblurring, spectral analysis are examples of information rearrangement. Information

reduction is concerned with removal of extraneous information - examples include, noise removal, feature extraction, parameter estimation.

Digital signal processing is concerned with the processing of signals which can be represented as sequences of numbers and multidimensional digital signal processing is concerned with the processing of signals which can be represented as multidimensional arrays, such as sampled images or sampled time waveforms which are received simultaneously from several sensors. The restriction to digital signals permits processing with digital hardware and it permits signal processing operators to be specified as algorithms or procedures.

Digital methods are simultaneously powerful and flexible. Digital systems can be designed to be adaptive and they can be made to be easily reconfigured. Digital algorithms can be readily transported from an equipment of one manufacturer to another or they can be implemented with special purpose hardware. They can be used equally well to process signals that originated as time functions or spatial functions and they interface naturally with logical operators such as pattern classifiers. Digital signals can be stored indefinitely without error. Many operations that we might want to perform on multidimensional sequences are also performed on 1-D sequences sampling, filtering and transform computation are examples. The multidimensional signal processing can be quite different from one dimensional processing due to the following reasons: 1) 2-D problems generally involve considerably more amount of data than 1-D signals; (2) the mathematics for handling multi-dimensional systems is less complete than that of 1-D systems; and (3) multidimensional systems have many more degrees of freedom, which give a system designer a flexibility not encountered in 1-D systems.

In the early 1960s, many of the methods of 1-D digital signal processing were developed with the intention of using digital systems to simulate analog systems. As a result, much of the theory was modeled after analog systems theory. In time, the awareness that digital systems can simulate analog systems very well and the strong push from the technology of digital hardware, the multi-dimensional signal processing field has blossomed and many of the methods in common use today have no analog equivalents. In late 1960s, most 2-D signal processing was performed using separable 2-D systems. The volume of data demanded by many 2-D applications and the absence of factorization theorem for 2-D

polynomials meant that many 1-D methods did not generalize well. The computer industry, by making components smaller and cheaper, has helped to solve the data volume problem[1].

Two new approaches are presented in this work for two Dimensional Discrete Fourier Transform (2-D DFT) computation. One through visual manipulation and the other based on the neural network concepts. The latter enables real time processing of 2-D signals in the frequency domain.

1.1.1 Transforms

Transforms come in many forms, depending on the particular application. However, the goal always the same: find an alternative domain where the processing or task at hand is easier to perform. To name just one example, if convolution needs to be performed, the Fourier domain is often preferred because of the complexity reduction so achieved [2]. Image transforms are useful for computation of convolution, correlation, image compression, noise reduction, edge detection and segmentation, image restoration and image fusion, template matching and object recognition, texture analysis and synthesis, motion estimation, object tracking etc.

Bulk of the applications in image processing is using linear transforms. Probably the most important, both for theoretical and practical reasons, is the Fourier transform [3]. The DFT is a fundamental tool in 1-D and multidimensional (m-D) signal processing. The DFT is used for computational reasons since it is discrete in time and frequency domains and can thus be implemented exactly. Moreover, there are fast algorithms that allow efficient computation.

A particular transform is selected depending upon the application and computational complexity. Joseph Fourier announced in 1807 that any 2π -periodic function could be represented as a series of sines and cosines. Since then, the spectral analysis of functions using Fourier series and integrals has been the source of numerous mathematical problems [4]. With the development of Fast Fourier Transform (FFT), Fourier analysis has been reduced to a readily available and practical procedure that can be applied without sophisticated training or years of experience. Different digital filtering techniques, which is one main application of signal processing, adopt different domains different to the input domain that requires transforms for the required conversion.

1.1.2 Discrete Fourier Transform

The Discrete Fourier Transform forms the basis for many digital signal processing techniques and is inherently a discrete time concept [5-8]. Fourier representation of finite duration sequences is referred to as Discrete Fourier Transform. The DFT is itself a sequence rather than a function of continuous variable, and it corresponds to samples, equally spaced in frequency of the Fourier Transform.

1.1.3 One Dimensional DFT

Consider a finite duration sequence $x(n)$ of length N samples so that $x(n) = 0$ outside the range $0 \leq n \leq N-1$. Then

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn}, 0 \leq k \leq N-1 \\ &= 0, \text{ otherwise.} \end{aligned} \quad \dots(1.1)$$

$$\begin{aligned} x(n) &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot W_N^{-kn}, 0 \leq n \leq N-1 \\ &= 0, \text{ otherwise} \end{aligned} \quad \dots(1.2)$$

Equations (1.1) & (1.2) form the 1-D DFT pair.

1.1.4 Matrix representation of DFT

Consider the DFT relation

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn}, k = 0, 1, 2, \dots, N-1 \quad \text{where } W_N^{nk} = e^{-j2\pi nk/N}$$

This describes the computation of N equations. For example, when $N=4$, $W_N = e^{-j(2\pi/4)}$, the equation can be written as

$$X(0) = x(0) W^0 + x(1) W^0 + x(2) W^0 + x(3) W^0$$

$$X(1) = x(0) W^0 + x(1) W^1 + x(2) W^2 + x(3) W^3$$

$$X(2) = x(0) W^0 + x(1) W^2 + x(2) W^4 + x(3) W^6$$

$$X(3) = x(0) W^0 + x(1) W^3 + x(2) W^6 + x(3) W^9$$

These equations can be more easily represented in matrix form:

$$\begin{aligned} X(0) &= W^0 & W^0 & W^0 & W^0 & x(0) \\ X(1) &= W^0 & W^1 & W^2 & W^3 & x(1) \\ X(2) &= W^0 & W^2 & W^4 & W^6 & x(2) \\ X(3) &= W^0 & W^3 & W^6 & W^9 & x(3) \end{aligned}$$

or more compactly as $X(k) = W^{nk} x(n)$

1.1.5 Twiddle factor

The DFT of a discrete-time signal $x(n)$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{nk}, k= 0,1,2,\dots,N-1$$

where $W_N = e^{-j2\pi/N}$ is called the twiddle factor of the DFT. The twiddle factors are periodic and define points on the unit circle in the complex plane. Fig. 1.1 shows the cyclic property of the twiddle factors for an 8-point DFT. The twiddle factors are equally spaced around the circle at frequency increments of $\frac{F_s}{N}$, where F_s is the sampling frequency of the input signal.

Therefore, the set of frequency samples, which defines the spectrum $X(k)$ are given on a frequency axis whose discrete frequency locations given by $F_k = k(\frac{F_s}{N})$, $k = 0,1,\dots, N-1$.

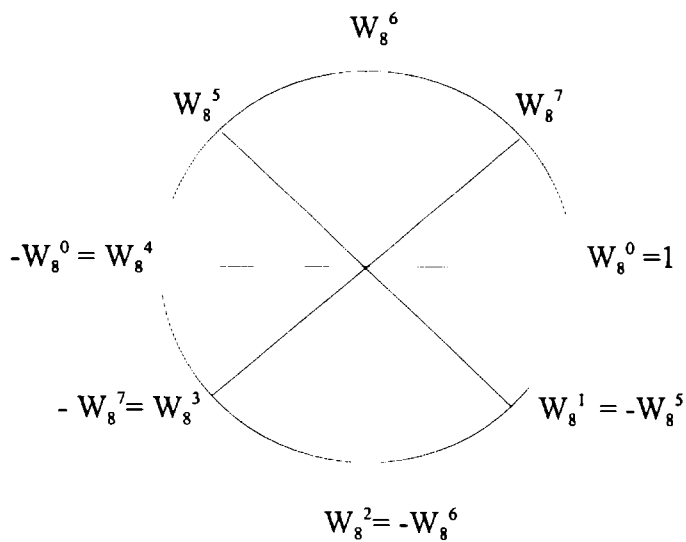


Fig. 1.1 The twiddle factors for $N = 8$

The frequency resolution of the Discrete Fourier Transform is equal to the frequency increment $\frac{F_s}{N}$ and is sometimes referred to as bin spacing of the Discrete Fourier Transform outputs. The frequency response of any discrete Fourier bin output is determined by applying a complex exponential input signal and evaluating the DFT bin output response as the frequency is varied.

1.1.6 Two Dimensional DFT

The Discrete Fourier Transform representation of two-dimensional sequences is of considerable computational importance in the digital processing of two-dimensional signals such as photographs and seismic array data, where the task is more complex and the volume of data is greater.

Consider a finite duration sequence $x(n_1, n_2)$ of size $N_1 \times N_2$ samples so that $x(n_1, n_2) = 0$ outside the range $0 \leq n_1 \leq N_1-1, 0 \leq n_2 \leq N_2-1$. Then the Discrete Fourier Transform [1,9-10] relations are given by

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2}, \quad 0 \leq k_1 \leq N_1-1, 0 \leq k_2 \leq N_2-1 \quad \dots(1.3)$$

$$x(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) W_N^{-n_1 k_1} W_N^{-n_2 k_2}, \quad 0 \leq n_1 \leq N_1-1, 0 \leq n_2 \leq N_2-1 \quad \dots(1.4)$$

1.1.7 Algorithms to implement 2 - D DFT

There are many algorithms for calculating 2-D DFT which vary considerably in their computational complexity [1]. As will be evident from the literature survey presented in chapter 2, there are a good number of approaches to implement the DFT. However, we shall examine three algorithms in the following sections, to bring out the salient aspects of 2-D DFT computation.

1.1.7.1 Direct computation

The direct calculation of 2-D DFT is simply the evaluation of the double sum

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) \cdot W_{N_1}^{n_1 k_1} \cdot W_{N_2}^{n_2 k_2} \quad \dots(1.5)$$

$$\text{for } 0 \leq k_1 \leq N_1-1 \ \& \ 0 \leq k_2 \leq N_2-1 \text{ where } W_N = e^{-j2\pi / N}$$

If we assume that the complex exponential in equation (1.5) have been pre-computed and stored in a table, then the direct evaluation of one sample of $X(k_1, k_2)$ requires $N_1 N_2$ complex multiplications and a like number of complex additions. Since the entire DFT involves $N_1 N_2$ output samples, the total number of complex multiplications and complex additions needed to evaluate the DFT by direct calculation is $N_1^2 N_2^2$

1.1.7.2 Row-column decomposition

The DFT relation can be rewritten as

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x(n_1, n_2) \cdot W_{N_2}^{n_2 k_2} \right] \cdot W_{N_1}^{n_1 k_1}$$

If we write
$$G(n_1, k_2) = \sum_{n_2=0}^{N_2-1} x(n_1, n_2) W_{N_2}^{n_2 k_2}$$

then,
$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} G(n_1, k_2) W_{N_1}^{n_1 k_1}$$

Each column of G is the 1-D DFT of the corresponding column of x. Each row of X is the 1-D DFT of the corresponding row of G. Thus we can compute a 2-D DFT by decomposing it into row and column DFTs. We first compute the DFT of each column of x, put the results into an intermediate array, then compute the DFT of each row of the intermediate array. Alternatively we could do the row DFTs first and the column DFTs second.

If a direct calculation is used to compute the 1-D DFTs in a row-column decomposition, then the evaluation of a 2-D DFT requires, $C_{r/c\text{direct}} = N_1 N_2 (N_1 + N_2)$ complex multiplications and additions. If each of N is a power of 2, so that 1-D FFTs can be used, the complex multiplications are further reduced to $C_{r/c\text{FFT}} = N_1 N_2 (\log N_1 N_2) / 2$. The number of complex additions needed is twice this number.

1.1.7.3 Vector-radix Fast Fourier Transform

The 1-D FFT algorithm achieves its computational efficiency through a 'divide and conquer' strategy. If the DFT length is, for example, a power of 2, the DFT can be expressed in turn as a combination of two quarter-length DFTs and so on. The 2-D vector-radix FFT algorithm is philosophically identical. A 2-D DFT is broken down into successively smaller 2-D DFTs until, ultimately, only trivial 2-D DFTs need be evaluated.

We can derive the decimation-in-time version of the algorithm by expressing an (N₁ × N₂)-point DFT in terms of four N/2 × N/2 DFTs (if N is divisible by 2). The DFT summation can be decomposed into four summations: one over those samples of x for which n₁ and n₂ is even, and one for which both n₁ and n₂ are odd. This gives us,

$$X(k_1, k_2) = S_{00}(k_1, k_2) + S_{01}(k_1, k_2) W_N^{k_2} + S_{10}(k_1, k_2) W_N^{k_1} + S_{11}(k_1, k_2) W_N^{k_2+k_1}, \text{ where}$$

$$S_{00}(k_1, k_2) = \sum_{m_1=0}^{N/2-1} \sum_{m_2=0}^{N/2-1} x(2m_1, 2m_2) W_N^{2m_1 k_1 + 2m_2 k_2}$$

$$S_{01}(k_1, k_2) = \sum_{m_1=0}^{N/2-1} \sum_{m_2=0}^{N/2-1} x(2m_1, 2m_2 + 1) W_N^{2m_1 k_1 + 2m_2 k_2}$$

$$S_{10}(k_1, k_2) = \sum_{m_1=0}^{N/2-1} \sum_{m_2=0}^{N/2-1} x(2m_1 + 1, 2m_2) W_N^{2m_1 k_1 + 2m_2 k_2}$$

$$S_{11}(k_1, k_2) = \sum_{m_1=0}^{N/2-1} \sum_{m_2=0}^{N/2-1} x(2m_1 + 1, 2m_2 + 1) W_N^{2m_1 k_1 + 2m_2 k_2}$$

The arrays S_{00} , S_{01} , S_{10} , S_{11} are each periodic in (k_1, k_2) with horizontal and vertical periods $N/2$.

The above said equations express the samples of the $N \times N$ DFT, $X(k_1, k_2)$, in terms of four $N/2 \times N/2$ DFTs. By analogy with the corresponding equations from the 1-D case, the computation is represented by a butterfly, or more properly a radix (2×2) butterfly.

Each butterfly requires that three complex multiplications and eight complex additions be performed. To compute all samples of X from S_{00} , S_{01} , S_{10} , S_{11} requires calculation of $N^2/4$ butterflies. This decimation process can be performed $\log_2 N$ times if N is a power of 2. The number of complex multiplications that need to be performed during the computation of a $(N \times N)$ point radix (2×2) FFT is

$$C_{\text{vr}(2 \times 2)} = \frac{3}{4} N^2 \log_2 N.$$

The foregoing discussion would have revealed that the 2-D DFT is computationally quite intensive. While research has been progressing on speeding up the 2-D DFT, the transform itself found many applications in image processing, which describes information in two dimensions. Some of the issues in digital image processing is worth considering to stress the diversity of the utility of 2-D DFT.

1.2 DIGITAL IMAGE PROCESSING

Digital image processing is a rapidly evolving field with growing applications in science and engineering. Image processing holds the possibility of developing the ultimate machine that could perform the visual functions of all living beings [10]. Interest in Digital Image Processing methods stems from two principal application areas: improvement of pictorial information for human interpretation, and processing of scene data for autonomous machine perception. One of the first applications of image processing techniques in the first category was in improving digitized newspaper pictures sent by submarine cable between

London & New York. Digital image processing has a broad spectrum of applications, such as remote sensing via satellites and other space crafts, image transmission and storage for business applications, medical processing, RADAR, SONAR, and acoustic image processing, robotics and automated inspection of industrial parts. Typical problems in machine perception that routinely utilize image processing techniques are automatic character recognition, industrial machine vision for product assembly and inspection, military recognizance, automatic processing of finger prints, etc [9].

A digital image is an image $x(n_1, n_2)$ that has been discretized in both spatial coordinates and brightness. A digital image can be considered as a matrix whose row and column indices identify a point in the image and the corresponding matrix element value identifies the gray level at that point. The elements of such a digital array are called pixels or pels.

Processing of digital images involves procedures that are usually expressed in algorithmic form. Thus with the exception of image acquisition and display, most image processing functions can be implemented in software. The reason for specialized image processing hardware is the need for speed in some applications or to overcome some fundamental computer limitations. For example, an important application of digital imaging is low-light microscopy. To reduce image noise requires performing image averaging over numerous images at frame rates (30 images per second in most cases). The bus architecture in all but a few high performance computers cannot handle the data rate required to perform this operation. Thus today's image processing systems are a blend of off-the shelf computers and specialized image processing hardware, with the overall operation being orchestrated by software running on the host computer [9]. Transforms are the fundamental tools that are used in most image processing applications. Probably the most important transform, both for theoretical and practical reasons is the Fourier Transform [11].

The different realms involved in Image Processing are briefly described below.

1.2.1 Image Enhancement

The principal objective of image enhancement techniques, classified as a preprocessing operation, is to process an image so that the result is more suitable than the original image for a specific application. Examples include contrast and edge enhancement pseudocoloring, noise filtering, sharpening and magnifying. Image enhancement is useful in

feature extraction, image analysis and visual information display. This enhancement process itself does not increase the inherent information content in the data. It simply emphasizes certain specified image characteristics.

The enhancement techniques fall into two broad categories: spatial domain methods and frequency domain methods. The spatial domain refers to the image plane itself, and the approaches in this category are based on the direct manipulation of pixels in an image. The image processing functions in the spatial domain may be expressed as

$$g(n_1, n_2) = T[f(n_1, n_2)]$$

where $f(n_1, n_2)$ is the input image, $g(n_1, n_2)$ is the processed image and T is an operator on f defined over some neighborhood of (n_1, n_2) .

Frequency domain processing techniques are based on modifying the Fourier transform of an image. The foundation of the frequency domain techniques is the convolution theorem [9]. Let $g(n_1, n_2)$ be an image formed by the convolution of an image $f(n_1, n_2)$ and a linear position invariant operator $h(n_1, n_2)$, that is,

$$g(n_1, n_2) = h(n_1, n_2) * f(n_1, n_2).$$

Then, from the convolution theorem, the following frequency domain relation holds:

$$G(k_1, k_2) = H(k_1, k_2)F(k_1, k_2)$$

where G , H , and F are the Fourier transforms of g , h and f respectively. Enhancement techniques based on various combinations of methods from these two categories are also used [9,11].

1.2.2 Image Restoration

Any image acquired by optical, electro-optical or electronic means is likely to be degraded by the sensing environment. Image restoration refers to removal or minimization of known degradations in an image. This includes deblurring of images degraded by the limitation of a sensor or its environment, noise filtering, and correction of geometric distortion or nonlinearities due to sensors. Image restoration differs from image enhancement in that the latter is concerned more with accentuation or extraction of image features rather than restoration of degradations [10].

1.2.3 Image compression

An enormous amount of data is produced when a 2-D light intensity function is sampled and quantized to create a digital image. In fact, the amount of data generated may be so great that it results in impractical storage, processing and communication requirements. For instance, more than 25 gigabytes of data are required to represent the *Encyclopaedia Britannica* in digital form [9].

Image compression addresses the problem of reducing the amount of data required to represent a digital image by removing redundant data. Image data compression methods fall into two common categories. The first, called predictive coding methods, that exploit redundancy in the data. Redundancy is a characteristic related to such factors as predictability, randomness and smoothness in the data. Techniques such as delta modulation and differential pulse code modulation fall into this category. In the second, called transform coding, the compression is achieved by transforming the given image into another array such that a large amount of information is packed into a small number of samples. Other image data compression methods exist that are generalizations or combinations of these methods. The compression process inevitably results in some distortion due to the rejection of some relatively insignificant information. Efficient compression techniques tend to minimize this distortion.

Applications of data compression are primarily in transmission and storage of information. Image transmission applications are in broadcast television; remote sensing via satellite, aircraft, radar or sonar; teleconferencing; computer communications; and facsimile transmission. Image storage is required most commonly for educational and business documents, medical images used in patient monitoring systems, and the like [10].

1.2.4 Image Segmentation

This area of processing comes under image analysis. The first step in image analysis is to segment the image. Segmentation subdivides an image into its constituent parts or objects. The level to which this subdivision is carried depends on the problem being solved. That is the segmentation should stop when the objects of interest in an application have been isolated. For example, in autonomous air-to-ground target acquisition applications, interest lies, among other things, in identifying vehicles on a road. The first step is to segment the road from the image and then to segment the contents of the road down to objects of a range

of sizes that correspond to potential vehicles. There is no point in carrying segmentation below this scale, nor is there any need to attempt segmentation of image components that lie outside the boundaries of the road.

In general, autonomous segmentation is one of the most difficult tasks in image processing. This step in the process determines the eventual success or failure of the analysis. In fact, effective segmentation rarely fails to lead to a successful solution. For this reason, considerable attention should be given to improve the probability of rugged segmentation. Segmentation algorithms for monochrome images generally are based on either discontinuity or similarity properties of gray-level values.

1.2.5 Image Description and Representation

After an image has been segmented into regions, the resulting aggregate of segmented pixels usually are represented and described in a form suitable for further computer processing. Basically, representing a region involves two choices: (1) represent the region in terms of its external characteristics (its boundary) or (2) represent it in terms of its internal characteristics (the pixel comprising the region). Choosing a representation scheme is only a part of the task making the data useful to a computer. The next task is to describe the region based on the chosen representation. For example, a region may be represented by its boundary with the boundary described by features such as its length, the orientation of the straight line joining the extreme points and the number of concavities in the boundary.

Generally, an external representation is chosen when the primary focus is on the shape characteristics. An internal representation is selected when the primary focus is on reflectivity properties, such as color and texture. An important approach to region description is to quantify its texture content. The three approaches to describe the texture of a region are statistical, structural and spectral.

1.2.6 Image Recognition and Interpretation

Image recognition and interpretation are related primarily to applications requiring automated image analysis. Image analysis is a process of discovering, identifying, and understanding patterns that are relevant to the performance of an image based task. One of the principal goals of image analysis by computer is to endow a machine with the capability to approximate, in some sense, a similar capacity in human beings. Thus an automated image

analysis system should be capable of exhibiting various degrees of intelligence. The concept of intelligence is somewhat vague, particularly with reference to a machine. However, conceptualizing various types of behavior generally associated with intelligence is not difficult. Several characteristics come immediately to mind: (1) the ability to extract pertinent information from a background of irrelevant details; (2) the capability to learn from examples and to generalize this knowledge so that it will apply in new and different circumstances; and (3) the ability to make inferences from incomplete information.

1.2.7 Computed Imaging

Advances in the design and fabrication of imaging detectors, combined with increasingly fast computers, has fueled the development of a spectacular variety of computed imaging systems. Such systems use numerical algorithms to transform measured data into an image. In most of these imaging systems, collect measurements of line integrals of the unknown image or samples of its Fourier transform [11]. The projection slice theorem is the fundamental result that is used to reconstruct images from their projections. This reconstruction is based on 2-D DFT.

The work described in this thesis is suitable for the areas of image processing described above.

1.3 NEURAL NETWORKS

1.3.1 Definition of Neural Networks

A neural network structure can be defined as a collection of parallel processors connected together in the form of a directed graph, organized such that the network structure tends itself to the problem being considered [12]. Now the term neural network, or more properly artificial neural network, has come to mean any computing architecture that consists of massively parallel interconnection of simple “neural” processors [13].

1.3.2 Neural Network Models

Neural Networks is a computational strategy that is radically different from the notions of ordinary, serial computing. It is a powerful tool for applications where the

processing is to be done in parallel. They are rough models of the human mental processes. These models are based on the use of large numbers of elemental processors interconnected in manners reminiscent of human neural networks and they exhibit powerful learning, memorization and associative recall capabilities of pattern formatted information. Because of their massive parallelism, neural networks can process information and carry out solutions almost simultaneously. They learn by being shown examples and expected results or they form associations without being promoted or reprimanded. Thus, they are good at solving many problems.

Neural networks offer the following specific advantages.

- a) Adaptive learning: This is learning to perform specific tasks by undergoing training with illustrated examples. Due to this feature, one does not need to have elaborate a priori models or a need to specify the probability distribution function.
- b) Self organization: Neural networks use self organizing capabilities to create representations of distinct features in the presented data. This leads to generalization of features.
- c) Fault tolerance: Neural network is the first computational method which are inherently fault tolerant. Networks can learn to recognize noisy and incomplete data and also exhibit graceful degradation when part of the network itself is destroyed.
- d) Real-time operation: This is due to its parallel distributed structure. For most networks operating in the real time environment, the need for training is minimal and is the only time consuming aspect of the operation.

The human brain has more than 10 million neural cells, which have complicated interconnections and constitute a large scale network. Hence, uncovering the neural mechanisms of the higher functions of the brain is not easy. Modeling neural networks [14] is useful in explaining the brain and also in Engineering applications. Researchers have reported various general purpose and special purpose models. But these are useful for classification type problems rather than computational problems. These have the function of self-organization and can learn to recognize patterns. Many are hierarchical networks consisting of layers of neuron-like cells. The ability to process information increases in proportion to the number of layers in the network.

Various paradigms are available. But, different paradigms are suitable for different kinds of tasks. The first step in matching an application and a neural network is to weigh the

characteristics, requirements, and drawbacks of each. Some of the important paradigms are briefly described below.

1.3.2.1 Perceptron Model

The perceptron was the first neural network to emerge. The original perceptron model was strictly feed forward. The learning algorithm for the perceptron allowed it to distinguish between classes of output if they were linearly separable in terms of decision space.

The neuron model was proposed by McCulloch and Pitts in 1943. Their model stemmed from their research into the behavior of the neurons in the brain. It specifically does not take any account of the complex patterns and timings of actual nervous activity in real neural systems, nor does it have any of the complicated features found in the body of biological neurons. This ensures its status as a model, and not a copy, of a real neuron, and makes it possible to implement on a digital computer. The model neurons, connected up in a simple fashion were given the name 'Perceptrons' by Frank Rosenblatt in 1962.

The perceptron model consists of a single layer of neurons [15,16]. The neurons are connected to the inputs through weights. The inputs can be binary or continuous. The output of each neuron is a binary value based on the weighted sum of inputs compared with a fixed threshold value.

A perceptron can learn anything it can represent [16]. The basic capability of the perceptron is that, it can distinguish between linearly separable decision spaces. However, an XOR function cannot be realized, since it is linearly inseparable. In spite of such drawbacks like linear separability requirement, indeterminacy of the number of steps required to train the network and the slow training algorithm, the Perceptron forms the foundation for many other models of neural networks. It is suitable for pattern recognition applications like classification of shapes, character recognition and robot vision systems.

1.3.2.2 Multilayer Perceptron : Backpropagation Network

As mentioned in the previous paragraph, the criterion that the function should be linearly separable, strictly limits the use of the perceptron model. Linearly inseparable functions can be implemented by using multilayer networks. This is also referred to as the backpropagating multilayer network or simply, backpropagation network. It is a feed

forward network - the first layer is called the input layer, the last one is the output layer and the layers in between the input and output layers are called the hidden layers [16-17].

The network is trained by the backpropagation algorithm [17]. This involves two passes, a forward pass and a backward pass. In the forward pass, the error is computed and in the backward pass, the error is minimized by modifying the weights. In the back propagation algorithm, the transfer function is required to be differentiable anywhere and usually a sigmoid function, instead of a step function, is used. This also provides automatic gain control. Along with the set of inputs, the corresponding desired outputs are also given and hence, the training is supervised.

Backpropagation networks have been applied successfully to a wide variety of applications. The main drawbacks of this method are the local minima and the long training time and the stability of the learned patterns. The backpropagation algorithm tries to adjust the weights to yield a minimum error with the desired outputs. But, the network can get trapped in a local minimum when there is a much deeper minimum nearby. This yields a less accurate solution. This algorithm also requires lots of supervised training and there is no guarantee that the system will converge. The backpropagation network has been successful in solving problems in the areas of feature extraction, pattern classification, and advanced control problems.

1.3.2.3 Hopfield Network Model

The Hopfield network is a recurrent network in which the output of each neuron is fed back to the input and thus modifying the states. The output is then re-calculated and the process is repeated again and again until a stable output is reached. Each neuron is connected to every other neuron and forms a fully interconnected network [12,16]. The learning is supervised.

The input pattern is read directly into the single network layer. A neuron is chosen at random and the weighted sum of inputs is thresholded to get the output. This procedure is continued till all the neurons are stable, that is, none of them change value when the procedure is repeated. The output will be the state of each neuron in the network. The weights in the network are symmetric.

Difficulties with the Hopfield network may arise if the patterns selected for storage do not form stable states in the iterative convergence procedure. Another property is that this

network is not adaptive - it cannot learn in real time. The maximum number of patterns that can be stored is about 15 percent of the total number of neurons in the network [18]. If the patterns are close to each other, the network will not converge to a stable state which distinctly represents each pattern. Finally, it is possible that a noisy pattern presented to the network may converge.

The Hopfield network has been useful in performing content addressable recall, solving optimization problems, robotic control systems, target identification and retrieving complete data from fragments.

1.3.2.4 Adaptive Resonance Theory (ART) Model

ART networks are radically different from other networks. These networks work on binary or real valued inputs. Their ability to create a new pattern classification when it observes a new type of pattern makes it highly attractive for applications such as automatic target recognition and other spatio-temporal and spatial pattern recognition tasks. The ART network solves the stability-plasticity dilemma encountered in other network models. A feedback mechanism is introduced between the two layers of the network and this facilitates learning of the new information without destroying existing ones, automatic switching between plastic and stable and stabilization of the encoding of the classes done by the neurons. In the ART network, information in the form of neuron inputs reverberates back and forth between layers. If proper outputs develop, a suitable oscillation ensues, which is the neural network equivalent of resonance and during this period, learning or adaptation can occur.

It is a two layer network, one being the input layer and the other the output layer [19-20]. The output layer has a finite number of neurons that are used only when needed. At any time, some output neurons will be in use while the others wait until needed. Once all the output neurons are used, the network stops adapting. If the network has previously learnt to recognize an input pattern, then a resonant state will be reached quickly, when the pattern is presented. During resonance, the adaptation process will re-reinforce the memory of the stored pattern. If no match to the input pattern is found, the network will enter a resonant state whereupon the input pattern will be stored for the first time.

Thus, the ART network is a self organizing network. It creates new categories when new patterns are presented. These can be useful when recognizing new patterns and

classifying them as such. However, in ART, the categories evolve over time. Thus, we do not have absolute control over category representation that we have when we ourselves choose the categories. Thus ART is useful in pattern recognition problems where the number of categories or pattern classes are not known in advance. Another drawback is that ART cannot handle shifted or distorted data.

Major applications of ART networks include speech recognition and generation, recognition of visual patterns and radar image detection.

1.3.2.5 Self Organizing Map (SOM)

The self organizing maps are the networks of choice for applications involving mapping distributed sensory information into a two dimensional or three dimensional representation. They may also be used for robotic arm control and optimization application. These mappings preserve topographic relations that may have existed among input data. This process can be used for reducing the dimensionality of complex input data and for pattern recognition.

The SOM is an auto-associative network having a single, highly interconnected layer and is a recurrent network [21,22]. Thus it uses the unsupervised learning method. Weights must be initialized and both weights and inputs must be normalized. Neurons compete for the privilege of learning. In a Winner-Take-All learning rule, the neuron with the highest response and its nearest neighbors all adjust their weights. As time passes, the size of the neighborhood reduces. Neighborhoods become similar in their response properties and a global organization begins to take place. The overall effect is to move the output neurons to positions that map the distribution of the training patterns. After training, each neuron's weights model the features that characterize a cluster in the data.

SOM finds natural clusters and similarities from unlabeled input data. Applications include data compression, feature extraction and pre-processing weights or data for other networks. Other uses are in speaker identification and signal processing.

1.3.2.6 Neocognitron Model

The Neocognitron model, proposed by Kuniyiko Fukushima of NHK Broadcasting Science Research Laboratories, Japan in 1982, is a powerful tool for visual pattern recognition [23-27]. The network is self organized by 'learning without a teacher', and

acquires an ability to recognize stimulus patterns based on the geometrical similarity of their shapes without being affected by their positions. After completion of self organization, the network has a structure similar to the hierarchy model of the visual nervous system. The network consists of an input layer (analogous to the photo-receptor array), followed by a cascade connection of a number of modular structures, each of which is composed of two layers of cells connected in cascade. The first layer of each module consists of 'S cells' which show characteristic simple cells, and the second layer consists of 'C cells' for complex cells. The feed forward synapses to each S cell have plasticity and are modifiable. After repetitive presentation of a set of input patterns, each stimulus pattern is able to elicit an output only from one of the C cells of the last layer, and conversely, this C cell has become selectively responsive only to that stimulus pattern. That is, none of the C cells of the last layer responds to more than one stimulus pattern. The response of these cells is not affected by the position of the input patterns. Neither is it affected by a small change in shape nor in size of the stimulus pattern.

But, when two or more patterns are shown simultaneously, the Neocognitron does not always correctly recognize them, because there was no provision for multiple recognition. Fukushima proposed the Modified Neocognitron model to circumvent this hurdle in 1984.

1.3.2.7 Modified Neocognitron Model

In this modified model, specially designed to accomplish multiple character recognition, backward connections are added to the conventional Neocognitron, which had only forward connections [28-29]. This model is enshrined with the ability of selective attention in visual pattern recognition. This can automatically segment and recognize individual patterns presented simultaneously. This can also restore imperfect patterns and eliminate noise from contaminated patterns. The process of recognition is achieved by the feed forward path same as that in the Neocognitron model. Backward path is used for switching attention and segmentation of input patterns.

When a composite figure consisting of two or more patterns are presented to the model that has finished learning, the model selectively focuses its attention on one pattern after another, segments the patterns from others and recognizes it separately. Even if noise or defects affect the pattern, the model can recognize it and recall the complete pattern in which the noise has been eliminated and the defects corrected. Perfect recall does not require that

the pattern be identical in shape to the training pattern the model learned. A pattern distorted in shape and changed in size can be correctly recognized and the missing portions restored. The stimulus pattern is presented to the lowest stage of the forward paths, the input layer, which consists of a two dimensional array of receptor cells. After the process of learning ends, cells of the recognition layer work as gnostic cells (grand mother cells).

The output of the recognition layer is sent to lower stages through the backward paths. The forward and backward signals interact with each other in the hierarchical network. The backward signals facilitate the forward signals, and at the same time, the forward signals gate the backward signal flow. This process resembles that of the adaptive resonance theory in the sense that the forward and backward signals interact with each other, but the method of interaction differs. The result of the associative recall appears at the lowest stage of the backward paths, the recall layer. We can also interpret the output of the recall layer as the result of segmentation. The response of the recall layer is fed back to the input layer. At each stage of the hierarchical network, several kinds of cells exist.

1.3.2.8 Cellular Neural Networks

The Cellular Neural Network (CNN) is a class of information processing systems proposed by L. O. Chua and L. Yang in 1988. Like neural network, it is a large-scale nonlinear analog circuit which processes signal in real-time. Like cellular automata it is made of a massive aggregate of regularly spaced circuit clones, called cells, which communicate with each other directly only through its neighbors. Cells not directly connected together may affect each other indirectly because of the propagation effects of the continuous-time dynamics of cellular neural networks. Each cell is made of a linear capacitor, a nonlinear voltage controlled current source and a few resistive linear circuit elements [30].

Cellular neural networks share the best features of both worlds; its continuous time feature allows real-time signal processing found wanting in the digital domain and its local interconnection feature makes it tailor made for VLSI implementation. [31] shows some of the applications of cellular neural networks in image processing and pattern recognition. For such applications, the network functions as a two-dimensional filter. However, unlike conventional two-dimensional filters, the cellular neural network uses parallel processing of the input image space and delivers the output in continuous time.

1.4. MOTIVATION FOR THE PRESENT WORK

1.4.1 Why Neural Networks in Signal Processing ?

The recent resurgence of interest in neural networks has its roots in the recognition that the brain performs computations in a different manner than do conventional digital computers. Computers are extremely fast and precise at executing sequences of instructions formulated for them. The human information processing system is composed of neurons switching at speeds about a million times slower than computer gates. Yet, humans are more efficient than computers at computationally complex tasks such as speech understanding, visual information processing etc. Unfortunately the understanding of biological neural systems is not developed enough to address the issues of functional similarity that may exist between the biological and man made neural systems [15].

During the past decade neural networks have begun to find wide applicability in many diverse aspects of signal processing, for example, filtering, parameter estimation, signal detection, system identification, pattern recognition, signal reconstruction, time series analysis, signal compression and signal transmission. The signals concerned include audio, video, speech, image, communication, geophysical, sonar, radar, medical, musical and others. The key features of neural network involved in signal processing are their asynchronous parallel and distributed processing, nonlinear dynamics, global interconnection of network elements, self organization, and high speed computational capability. With these features, neural networks can provide very powerful means for solving many problems encountered in signal processing [32].

Although neural networks can serve to further the understanding of brain functions, engineers are interested in neural networks for problem solving. As an engineering technique, neural networks have their own advantages, disadvantages and assumptions. The natural question is: What can neural networks do that traditional signal processing techniques can't do? Certainly speed of computation is a factor. In traditional Von Neumann computers, speed is limited by the propagation delay of the transistors. Neural networks on the other hand, because of the massively parallel distributed processing, can perform computations at a much higher rate. Because of their adaptive nature, neural networks can adapt to changes in the data and learn the characteristics of input signals [13].

1.4.2 Problems in Present Day DFT Computation

Speed of computation is very important in real-time applications. This can be improved by reducing the number of multiplications and/or by parallel processing. The DFT is a useful analytical tool and there are a number of techniques used for reduction of computation in DFT [5-7]. Literature shows that FFT is the most popular algorithm to implement DFT, which is highly efficient in the case of 1-D signals. The N-point sequence requires N^2 complex multiplications and $N(N-1)$ complex additions in the case of direct DFT where as $N\log_2 N$ complex operations in the case of FFT when N is a power of 2 [5]. The reduction in computation is more predominant when N is large.

In the DFT computation, the data will be real values where as DFT coefficients will be complex values. In present day DFT computation schemes, the data will also be converted to complex form and the computations will be in complex form, which increases the computation time and the memory requirement. One complex multiplication requires 4 real multiplications and two real additions. Two memory locations will be required to store one complex data. Also, the computation time will be more for multiplication than addition. Thus the speed of computation can be improved by reducing the number of complex multiplications. The speed of computation can also be increased by implementing the computations in parallel.

For 2-D signals, the FFT method is highly time consuming since it requires a large number of complex multiplications [1]. Due to the computational complexity of the Fourier Transform, it is not in much use in 2-D signal processing applications. Parallel computation also increases speed and may render real time applications possible.

2-D FFT computation is normally done using the Row-Column method, which uses 1-D FFT computation, or Vector Radix method. These methods require the length of the sequence N to be a power of 2. If N is not a power of 2 the sequence will be padded with zeros, but this will vary the frequency components and in many applications this is undesirable.

Also, in many applications, we may need only a few DFT coefficients rather than the complete set of DFT coefficients. In such cases also, the complete coefficients will be computed using FFT or similar other algorithms. So a new approach based on visual manipulation of the data / 2×2 DFT coefficients represented in the form of pictures is developed. This method is suitable when a few DFT coefficients need be computed. This will

help in visualizing the influence of data at different time / spatial index over a particular frequency.

The evaluation of DFT coefficients in the form of pictorial representation discussed in chapter 3 shows specific patterns that can easily be implemented in a parallel distributed scheme as explained in chapter 4 and can be extended to any order N . This approach will be highly useful in the analysis and processing of 2-D signals that exhibits repetition of blocks.

1.5. BRIEF SKETCH OF THE PRESENT WORK

The scheme of the work presented in this thesis is given below:

A review of the important research work done in the field of signal processing, neural networks and image processing relevant to the work are presented in chapter 2. Special emphasis is given to 2-D DFT and the different neural network models.

Chapter 3 describes the 2-D DFT computation through visual manipulation. The details of construction of the pictorial representations are presented in this chapter. The analysis of the pictorial representation for different order is carried out for the 2-D DFT computation and manipulation.

Chapter 4 gives details of the development of a neural network model for the 2-D DFT computation. The neural network approach is derived based on the results in chapter 3. The simulation results of the model are also included in this chapter.

The discussions and conclusions drawn from the investigations on the development of a neural network model and the visual manipulation approach for the 2-D DFT computation, their applications and the scope of further work in the field are discussed in chapter 5. The possible hardware implementation of the neural network model is also presented in this chapter.

The relation between 6×6 point DFT coefficients and 2×2 point DFT coefficients are presented in Appendix A. Appendix B deals with important results useful in the derivation of the algorithm used in the chapter 4.

REVIEW OF PAST WORK IN THE FIELD

Joseph Fourier announced in 1807 that any 2π -periodic function could be represented as a series of sines and cosines. Since then, the spectral analysis of functions using Fourier series and integrals has been the source of numerous mathematical problems [4]. A fundamental advancement in the field of signal processing occurred in the mid-1960s with the discovery of an efficient algorithm (FFT) for computing the sampled spectrum of a signal [33, 34]. There were further developments to improve the speed of computation in many applications, especially in 1-D signal processing, by modifying the algorithms or using parallel implementations [35-41]. But in many applications based on 2-D or m-D signals, the computational speed is a bottleneck. For instance, the image processing applications involve processing of bulk data in a short time and the existing algorithms for 2-D DFT computation are not quite suitable in many cases and hence the use of frequency domain processing is less popular compared to time domain processing. In this context, the introduction of the artificial neural networks in computational applications appears to be of great advantage since it is based on parallel and distributed processing in terms of very simple elements called neurons. The idea to develop an approach to compute 2-D DFT based the neural network concepts shows a good promise in improving the computation time. The work reported here thus evolves a new approach to implement 2-D DFT based on the neural network concepts. A review of the related literature is therefore in order.

2.1 DISCRETE FOURIER TRANSFORM

The literature shows that an enormous amount of work has been carried out on 1-D DFT where as that on 2-D or multidimensional DFT is less. One reason is that DFT is linearly separable and hence can be implemented using 1-D DFT. However the literature is rich with good publications on 1-D and 2-D DFT. This section presents a review on the work in this area of computation. The following are few of the publications related to Discrete Fourier Transform.

Winograd [42] developed an algorithm to compute 1-D DFT based on polynomial reduction, with the length of the sequence as a prime number.

In [43] Auslander et al. modified the 1- D Winograd algorithms to derive an algorithm for DFT ($p; k$), the DFT on a k -dimensional data set with p points along each array, where p is a prime.

Rajan [44] presented a parametric representation of a composite operation consisting of transformation, conjugation and modulation suitable for the study of the properties of multidimensional Fourier transform. A class of properties of m -D Fourier transform is established using these properties.

Gertner [45] presented an algorithm to compute 2-D DFT based on the geometric properties of the integers and with feasibility to parallel implementation.

In [46] Rayfield et al. presented an implementation of 2-D DFT on a loosely coupled parallel processing system made up of a large number of identical processing nodes with reconfigurable point - to - point communication network.

Rajan [47] modified the general Winograd algorithm for 3×3 size data to take into account of diagonal symmetry present in the data for the evaluation of DFT of diagonally symmetric 2-D data of size $N \times N$ where N is a prime number.

In [48] Sorensen et al. presented a method for computing only a few (P) output points of a length- N 1-D DFT. It permits the computation of any consecutive set of P points where P should be a power-of-two. It is based on a factorization of DFT where one part is computed using standard power-of-two FFT and the other uses a technique similar to Geortzel algorithm.

Babic et al. presented an interpolation technique for computing more DFT coefficients than the number samples of the sequence [49]. The conditions and solutions for perfect interpolation of the DFT are considered.

In pursuit of the methods to implement DFT efficiently, systolic implementation techniques were tried out.

Wang et al. presented a systolic implementation of the DFT algorithm by using a double decomposition method to create two 4-point systolic processors for a direct linear DFT implementation [50]. The entire hardware implementation is connected in a pipeline with $O(N)$ throughput.

Chen et al. [51] proposed a 2-D Systolic array for performing 2-D DFT based on the use of Goertzel algorithm in a row-column or column-row format, with the features of regularity, modularity and concurrency that are suited for VLSI implementation.

Subsequently, Yang [52] presented a decomposition in which the 2-D DFT can be converted to a series of the odd DFT using the discrete Radon transform (DRT). A fast DRT (FDRT) algorithm with reduced number of additions, greater regularity and parallel for computing DRT is also presented. In parallel implementation, the FDRT based algorithm increases the speed greatly.

In [53] Gertner et al. proposed a parallel algorithm for 2-D DFT computation which eliminates inter processor communications and uses only $O(N)$ processors for $N \times N$ DFT. They have discussed the mapping of the algorithm on to architectures with broadcast and report capabilities. The performance on these machines are expressed in terms of N , channel bandwidth C , time A for an addition, and time to perform a 1-D DFT.

Concentrating on the VLSI implementation of DFT, Julien et al. presented a systematic application of linear projection and scheduling to Dependence Graph (DG) algorithm representations are used to generate asymptotically optimum DFT layouts in a VLSI computing model [54]. Two stage prime factor maps are used to generate regular and minimum Area Period (AP) architectures. They found that systematic multi projection of six-dimensional 3-factor DGs and eight-dimensional 4-factor DGs generate successively faster AT^2 optimal and regular architectures, but with increasing wire area.

Further, in [55] Miyanaga et al. proposed a single chip 400 MFLOPS 2-D FFT processor VLSI architecture that is designed using $0.8\mu\text{m}$ CMOS technology. This processor integrates 3,80,000 transistors in an area of $11.58 \times 11.58 \text{ mm}^2$ with a typical machine cycle time of 25 ns. The 24 bit floating point processor executes $2^n \times 2^n$ point 2-D DFT in real time.

Exploring further into structure of DFT, Ma demonstrated an algorithm for computing $DFT(2^n; k)$ based on ring structure [56]. This uses the principle of permuting the DFT matrix

into block structured matrix with circulant blocks corresponding to the disjoint cosets of kernel group K being the direct product of a group of order 2 and cyclic group of order 2^{k-1} .

In [57] Rajarevivarma et al. presents a modification to the Vector-Radix 2-D DFT computation algorithm when real-valued sequences are used so that 50% saving in computation compared to the complex computation. They are saying that the saving will be of the order of 75% if the 2-D input sequence is centro-symmetric or anti-symmetric.

Ju [58] presented an equivalent relationship and unified indexing of FFT algorithms. The paper shows that the same vector matrix form can represent the multidimensional FFT as the 1-D FFT. Also, shows that the addressing sequences of the M-D FFT is the subset of the 1-D FFT. Also stated that both 256x256 2-D FFT and 64K 1-D FFT can be finished within 6.56 milliseconds by implementing on array processor chip set LH9124/LH9320.

In [59] Gupta et al. presented the scalability analysis of parallel (1-D) FFT algorithm on mesh and hypercube connected multi computers using the isoefficiency metric. They found that the hypercube architecture can obtain linearly increasing speed up with respect to the number of processors with a moderate increase in problem size. They also states that FFT can not make efficient use of large-scale mesh architectures unless the bandwidth of the communication channels is increased as a function of the number of processors.

In [60] Fragopoulou et al. presented parallel algorithm for the computation of DFT on n-star graph network. They are claiming that the algorithm requires $O(n^2)$ multiply - add steps for an input sequence of $n!$ elements.

In [61] Pihl presented a review of what trade-offs exist between bit-parallel and bit-serial architectures with special interest in VLSI designs for high performance digital signal processing. Parallel and serial architectures are compared in terms of area, delay and power conception and it is found that bit-serial and bit-parallel architectures are comparable by area-time measure but substantial difference exists in power conception. Bit-serial designs are not as efficient from a power conception point of view.

In [62] Shin et al. proposed a parallel architecture based on linear arrays for high performance implementation of FFT. They said that it offers constant I/O bandwidth and high parallel processing capability. It is based on half butterfly which maximizes the utilization of processing elements.

2.2 NEURAL NETWORKS

The work on artificial neural network models has a long history. Development of detailed mathematical models began more than 40 years ago with the work of McCulloch and Pitts [63], Hebb [64], Rosenblatt [65], Widrow [66] and others. More recent work by Hopfield [67-69], Rumelhart and McClelland [70], Sejnowski [71], Feldman [72], Grossberg [73] and others has led to a new resurgence of the field. Though the neural networks evolved from the field of pattern recognition and optimization, implementors were fascinated by the PDP capability of neural network.

A number of neural network models are proposed by different research groups and used in various applications. A review of the basic models of the neural network is presented below to evolve the utility of Parallel Distributed Processing (PDP). It shall be demonstrated that the PDP capability has found extensive use in the present work.

Neuron models

Hopfield [68] proposed a modification to the McCulloch-Pitts neuron model by introducing sigmoid input-output relation so as to mimic the real neurons. A continuous, deterministic neuron network of interconnected neurons with graded responses has been analyzed.

Hampson et al. [74] developed three different representations for a thresholded linear equation neurons. A training algorithm is also proposed which can be learned much faster compared to the perceptron algorithm

Habib et al. [75] proposed a neuron type processing element by cascading three elements, one called the cell body with its dendritic inputs and synaptic junctions, another representing the axon base and finally the axon circuit. The circuit performs input temporal and spatial summation as well as thresholding. The entire neuron circuit is simulated and a VLSI design is presented.

Habib et al. [76] proposed a digital neuron type processor using timed Petri net which performs input temporal and spatial summation as well as thresholding, operates asynchronously.

Meador et al. [77] describes CMOS electronic circuits, which emulate natural neurons at a more detailed level than that typically employed by the artificial neural network models. The short term neuron dynamics is realized by a pulse-firing circuit and long-term neuron

dynamics is realized by fixed & programmable synapse circuits, thus implementing a programmable impulse neural network.

Fukumi et al. [78] described a new neuron model and its learning algorithm for speeding up the convergence in the learning of layered neural networks. The neuron is called a saturating linear coupled neuron.

Glanz [79] presented the research news on the functioning of the brain and says that by applying concepts from mathematical physics, researchers hope to understand the collective dynamics of billions of neurons and perhaps control them in epilepsy. In this many researchers have presented their observations, still the researchers are far from pinning down the precise nature of the brain dynamics that lead to seizures.

Feed Forward Networks

Feed Forward neural networks have remained attractive to implementors due to the simplicity of organization and ease of training. Since the present work centers itself around a simple feed forward structure to build 2-D DFT, a review of feed forward neural network is presented below.

Lippmann [80] provides an introduction to the field of artificial Neural Nets by reviewing six neural net models that can be used for pattern classification. In addition to describing these nets, a major emphasis is placed on exploring how some existing classification and clustering algorithms can be performed using simple neuron-like components.

Widrow et al. [81] described the practical applications of the adaptive neuron called adaptive linear combiner (ALC) in signal processing and pattern recognition. They described the system in detail giving its features and the functioning as well as the theory behind it. Experimentally they verified the functioning of the system and found to be effective.

Touretzky et al. presented the importance of choosing the number of neurons in the hidden layer of a feed forward neural network in [82]. They also explains what is the function of the hidden layer neurons in learning.

Widrow et al. [83] reviewed fundamental developments in feed forward artificial neural networks during 30 years. It describes the history, origination, operating characteristics and basic theory of several supervised neural network training algorithms.

In [84] Amari showed, by mathematical treatments, the capabilities and limitations of information processing by various architectures of neural networks. It considered the capabilities of transformations by layered networks, statistical neurodynamics, the dynamical characteristics of associative memory, a general theory of neural learning, and self organization of networks.

In [85] Poggio et al. developed a theoretical framework for approximation based on regularization techniques that leads to a class of three layer networks called regularization network and include as a special case the radial basis function method. This generalizes the theory of regularization networks to a formulation that turns out to include task dependent clustering and dimensionality reduction.

In [86] Reeke et al. presented a synthetic neural modeling which is a multilevel theoretical approach to the problem of understanding the neuronal bases of adaptive behavior and uses simultaneous large-scale computer simulations of the nervous system, the phenotype, and an environment of a particular organism to study events and their interactions at these three levels. The automata discussed here deal first with certain abstract properties of pattern recognition (Darwin I) and then with categorization and association (Darwin II) and a description of an automation with sensory and motor systems and autonomous behavior (Darwin III).

In [87] Brahm et al. described an associative neural network whose architecture is greatly influenced by biological data, with high parallelism and modularity.

Cellular Neural Networks

The cellular neural network uses the features of analog domain and digital domain. It is largely based on the principles of cellular automata, where in the connections are made as boolean functions, the cellular neural networks have become handy in the reported work. In the basic computational model developed, the learning is proposed to be accomplished by making and breaking connections. So a review on this model is useful.

This class of neural networks, used for computational applications, was proposed by Chua and his group.

In [30] Chua et al proposed a novel class of information processing systems called cellular neural networks. Like neural networks, it is a large-scale nonlinear analog circuit which processes signals in real-time. Like cellular automata, it is made of a massive aggregate of

regularly spaced circuit clones, called cells, which communicate with each other directly only through its nearest neighbors. This model is suited for high-speed parallel signal processing. The theory of this model is dealt with in this article.

In [31] Chua et al. presented some of the applications of the cellular neural network model proposed in [30]. Some impressive applications of this model to areas such as image processing and pattern recognition are demonstrated.

Rodriguez-Vazquez et al. [88] presented a unified, comprehensive approach to the design of continuous-time and discrete-time cellular neural networks using CMOS current-mode analog techniques. Cell design relies on exploitation of current mirror properties for the efficient implementation of both linear and nonlinear analog operators.

In [89] Varrientos et al. presented a current-mode implementation for a cellular neural network using simple current mirrors and transconductors and discussed the strength and weaknesses of the implementation based on the experimental results.

In [90] Fruehauf et al. described the theoretical and practical aspects of an actual hardware realization to implement the cellular neural network optically. The theoretical and practical aspects of a Fourier optical CNN implementation with 25x25 neurons have been outlined.

Roska et al. [91] presented algorithmically programmable analog array computer having real time and supercomputer power on a single chip called the CNN universal machine and supercomputer. In this the universal machine is described emphasizing its programmability as well as global and distributed analog memory and logic, high throughput via electromagnetic waves and also the use of it as a multichip supercomputer. The analogic algorithms as well as the analogic software are explained.

Cimagalli et al. [92] described a generalized cellular neural network (GCNN) in which the input is a two-dimensional picture that is processed continuously in order to detect in real time trajectories of moving objects in a noisy environment. Also described the architecture, its equations and the method of design. Also this compared with other paradigms of ANN.

Seiler et al. [93] presented an implementation of winner-take-all behavior in inputless cellular neural networks which is defined as follows: the eventual output of the cell with the largest initial state is +1, while all other cells is -1. Exact parameters are derived for winner-take-all CNN's with an arbitrary number of cells, such that their robustness with respect to the simplified structure is maximum. It is found that accuracy requirements increase with the

number of cells, such that the largest winner-take-all CNN's that can be reliably implemented with current methods may consist of only about ten cells.

Harrer [94] presented a generalized architecture of cellular neural network that allows multiple layers of different architecture, which can be combined in several interconnection modes. Also this model permits time varying templates so that they can be defined as cyclic and applied periodically and important templates are proved to be convergent.

In [95] Baktir et al. presented an analog CMOS circuit realization of cellular neural network with transconductance elements, with the property that it can easily adapt to various applications in image processing by choosing appropriate transconductance parameters according to the predetermined coefficients.

In [96] Sziranyi et al. proposed a hardware implementation of a character recognition system based on cellular neural network architecture, called CNND. The CNND consists of one or more analog cellular neural networks and some digital logic incorporating advantages of fast analog Signal Processing and fast and easy decision capability of digital logics. It can recognize multifold printed or handwritten characters.

Neocognitron Model

The Neocognitron model presents the property of hierarchically building the neural network. The complex problem of 2-D DFT computation has been demonstrated to be tractable in a hierarchy of computations in the present work. Relevant literature that triggered the approach is reviewed below.

Fukushima and his group proposed a group of hierarchical neural network models that are applicable to handwritten character recognition.

Fukushima [97] proposed a neural network model named Cognitron which uses unsupervised learning, classified under competitive learning. It does not have the ability to correctly recognize position shifted or shape distorted patterns. It recognizes the same pattern presented at a different position as a different pattern.

Fukushima [23] proposed a hierarchical neural network model, with a nickname 'Neocognitron', for a mechanism of visual pattern recognition and uses self-organization with the ability to recognize patterns based on the geometrical similarity of their shapes without affected by their positions. This is a modification to the model 'Cognitron'

In [24] Fukushima et al. presented a new algorithm for the model given in [23] and having the ability to self-organize by unsupervised learning to recognize stimulus patterns according to the difference in their shapes.

Fukushima [25] modified the Neocognitron model by adding feedback connections so as to add the function of associative memory and to recognize patterns. Pattern recognition is performed hierarchically by integrating information by converging afferent paths in the network and the associative recall is incorporated by distributing the integrated information to the lower order cells by diverging efferent paths.

In [26] Fukushima et al. suggested a modification to the Cognitron model by adding feedback inhibition to increase the speed of self-organization. The network therefore quickly acquires favorable pattern selectivity by the mere repetitive presentation of a set of learning patterns.

In [14] Fukushima presented a modification to the Neocognitron model such that it can successively pay attention to each pattern, segmenting it from the rest and recognizing it separately when two or more patterns are presented simultaneously.

Fukushima [29] presents a modification to the neocognitron model such that it can recognize patterns with selective attention, segmentation, associative recall and learn any set of patterns.

In [27] Fukushima et al. discusses a pattern recognition system which works with the mechanism of the neocognitron for the application of handwritten alphanumeric character recognition. It offers techniques for selecting training patterns useful for deformation invariant recognition of large number of characters.

White et al. [98] described a digital implementation of Neocognitron model with the corresponding modifications in the algorithm. The performance of the new model is compared with analog model.

Nellis et al. presented a modification to the Neocognitron model by replacing the analog cells with digital processing elements [99]. They demand that it has reduced training time and increased character throughput characteristics.

Hatakeyama et al. proposed an expanded Neocognitron model which can recognize both shape and the location of an object [100]. They presented an improved mechanism of local feature extraction and competitive learning.

Adaptive Resonance Theory (ART)

The ART model has the basic feature of stability and plasticity in learning. Also the number of classes that can be trained is dependent on the number of output neurons. Similarly the number of cells in the planes of the neural network model, developed in this work, can be set to a particular value that determine the maximum size of the input / output. The learning algorithm is proposed to modify the connections in such a way that any order upto the set maximum can be implemented. Hence the relevant literature is reviewed below.

Gail A. Carpenter & Stephan Grossburg proposed a group of network models based on the resonance principle and they are of the self organizing type.

In [19] Carpenter et al. described a neural network architecture that self organizes and self-stabilizes its recognition codes in response to arbitrary ordering of arbitrarily many and arbitrarily complex binary input patterns for the learning of recognition categories. The architecture possesses a context-sensitive self-scaling property which enables its emergent critical feature patterns to form. A nonlinear matching law, nonlinear associative laws are used. The architecture circumvents the noise, saturation, capacity, orthogonality, and linear predictability constraints that limit the codes which can be stably learned by alternative recognition models.

Carpenter et al. [20] presented the features such as stability-plasticity, self-stabilized learning in an arbitrary environment, learning rules etc. of the models ART1 and ART2. They also described how ART systems provide a fertile ground for gaining anew understanding of biological intelligence. They also suggest novel computational theories and real-time adaptive neural network architectures with promising properties for tackling some of the outstanding problems.

Kane et al. [101] proposed and demonstrated a hybrid opto-electronic implementation of ART2-A (Algorithmic - ART) utilizing an optical joint transform correlator. They demand that the resultant opto-electronic system is able reduce the number of calculations compared to a strictly computer based approach.

For the sake of completion, the other relevant models are also reviewed below.

Self Organizing Map (SOM)

Kohonen [21] presented the features of Self Organizing Map and stated that it has the special property of effectively creating spatially organized internal representations of various features of input signals and their abstractions. One novel result is that the self organization process can also discover semantic relations hips in sentences.

Kohonen [22] described the use of neural network in building a phonetic typewriter from the fundamentals to the hardware implementation. He says that the system developed, forms a complete speech recognizer employing neural computing principles and has been brought to a commercial stage.

Hopfield Associative Memory

McEliece et al. [18] presented a rigorous analysis on the capacity of the Hopfield associative memory by applying techniques from coding theory.

In [102] Li et al. did a qualitative analysis of the behavior of the equilibrium points to investigate the dynamic properties of a class of neural networks including the Hopfield model. The result pertains to analysis and synthesis of the models.

In [103] Li et al. investigated qualitative properties of neural networks described by a system of first-order linear ordinary differential equations which are defined on a closed hypercube of the state space with solution extended to the boundary of the hypercube. They modified the Hopfield model and says that it is easy to implement in analog integrated circuits.

In [104] Palm presented the information storing capacity of certain associative and auto associative memories. He pointed out that the storage capacity of an associative memory increases proportionally to the number of storage elements. The usefulness of the associative memories as opposed to conventional memories is also discussed.

Chung et al. [105] proposed a digital multiplier based on unidirectional feedback type neural network and is VLSI implemented. The feedback connections are half that of the Hopfield model. They demands that the new multiplier is simpler than that designed by conventional techniques.

Learning

In [106] whitcomb et al. presented a hierarchical learning algorithm for feed forward neural networks that attempts to simulate structured learning. In structured learning, lower

concepts are acquired first and then used to uncover higher level concepts. The procedure dynamically builds a multilevel hierarchical structure of hidden nodes as concepts are learned in the supervised manner.

In [107] Nitta et al. presented a complex numbered version of the backpropagation algorithm that can be applied to neural networks whose weights, threshold values, input and output signals are all complex numbers. This can be used to transform geometrical figures.

Barber et al. [108] reported a study of learning and retrieval algorithm for Hopfield's pattern classifier network, multilayer backpropagation network, Kohonen's self organizing feature map and ART-1 models. The algorithms are mapped on to the sequential pipelined neuro emulator architecture.

Implementation

Lami et al. [109] presented an architectural level evaluation scheme by proposing some performance indexes on idealized architecture classes. Some conclusions on the advantages and disadvantages of the architectures are presented.

In [110] Masa et al. discussed a CMOS neural network integrated circuit designed for very high speed applications. This full-custom, mixed analog - digital chip implements a fully connected neural network with 70 inputs, 6 hidden layer neurons and one output neuron. This digitally programmable, analog neural network classifies upto 70 dimensional vectors within 20 nano seconds, performing 20 billion multiply - and - add operations per second. They demand that the circuit occupies $10 \times 9 \text{ mm}^2$ silicon area with $1.5 \text{ }\mu\text{m}$ CMOS process and dissipates only 1W at 5V supply.

Distante et al. [111] presented a solution for mapping neural nets onto massively parallel architectures, based on regular array structures, that can easily be implemented.

In [112] Watanabe et al. describes an implementation of error back-propagation on a massively parallel cellular array processor, AAP-2, considering allocation of processors, computing the activation value, with a table look-up method and communication between processors.

Kung et al. [113] proposed a digital VLSI architecture based on programmable systolic array for implementing a wide variety of artificial neural networks. The array is meant to be more general purpose and may be used for a variety of algorithms.

In [114] Graf et al. described the VLSI implementation of a connectionist model to study feasibility of electronic implementation of the neural network models. The network described provides a flexible tool because it can evaluate Boolean expressions and arithmetic equations. They found that this network looks promising for building powerful recognition systems.

Hutchinson et al. [115] showed that Optical flow can be computed by injecting currents into resistive networks and recording the stationary voltage distribution at each node and hence showed that appropriate computations map on to simple resistive networks. These networks do not require a system-wide clock and rely on many connections between simple computational nodes, converges rapidly and are quite robust to hardware errors and consume moderate power.

In [116] Mostafavi presented a method for use in VLSI design of specialized interconnection of multiprocessor architecture for implementing neural network models. A simple feed forward design of interconnecting processing elements in a single building block is presented. This building block is modular and can be used for building larger size neural network system.

Heileman et al. [117] presented a framework for simulating neural network as discrete event nonlinear dynamical systems. They considered the design and construction of concurrent object-oriented discrete event simulation environment for neural networks.

Torbey et al. [118] presented an automated architectural synthesis methodology for implementing digital neural networks. The functional units used in these architectures, their components and features are discussed.

Tam et al. [119] described a multi-chip analog neural network system capable of prototyping networks with as many as 81920 synaptic connections and 1024 neurons. The neural network architecture is reconfigurable by routing all neuron activation values through a host computer which can re-map the network connectivity by simply changing a look up table in memory.

In [120] Fu shows how to interpret neural network knowledge in symbolic form. In this the scheme is explained and the interpretation algorithm is formulated. Formulated a relationship between neural network and a rule-based system. Also, demonstrated that the neural network generates rules of better performance than the decision tree approach in noisy conditions.

Gomez et al. [121] discussed the theory, design and implementation of a digital neural network using ASIC technology. They have considered a Boolean network model to realize Boolean functions.

Kung et al. [122] proposed a generic iterative model for a wide variety of artificial neural networks: single layer feedback networks, multilayer feed forward networks, hierarchical competitive networks, as well as some probabilistic networks. A unified formulation is provided for both the retrieving and learning phases. Based on the formulation, a programmable ring systolic array is developed. The architecture maximizes the strength of VLSI in terms of intensive and pipelined computing. They demand that this can be used as a basic structure for universal neuro computer architecture.

Speckmann et al. [123] presented a system for automatic synthesis of special purpose hardware for neural networks. Only an algorithmic description of the behaviour of the hardware and a simulation environment have to be written by the designer using VHDL. This description can be automatically mapped to the hardware. By using FPGAs with this system it can be learned.

Vidal [124] presented networks of Boolean programmable logic modules as one purely digital class of artificial neural nets. The approach contrasts with the continuous analog framework. He found that programmable neural networks are capable of handling many neural net applications and some of the limitations of threshold logic networks are avoided.

2.3 APPLICATIONS IN IMAGE PROCESSING

Ramkumar et al. [125] presented a new FFT based technique of fractal image compression which can significantly speed up the process of matching domain and range blocks. In this method the isometric transformations of a domain block are chosen to permit block matching through cross correlation of range and domain blocks, which is implemented through the FFT.

In [126] Shalinie et al. proposed a neuro-fuzzy model for effective compression and decompression of data used for image recognition. Contrast enhancement coupled with fuzzy procedures is used to preprocess the image. Counter propagation network is used to train few image patterns. Using these image patterns, the neuro-fuzzy model compresses the unlearned image data and recovers the image to full extent.

Barros et al. [127] proposed an architecture for segmenting sequences of images in gray levels at video rate. This architecture implements, on FPGA, an algorithm of image segmentation by self-organizing feature maps. Performance, cost and reconfigurability of the architecture are discussed.

2.4 Conclusion

The literature reviewed here has triggered the basic approach to develop the central idea in the work reported. The chapters to follow shall illustrate this aspect.

VISUAL MANIPULATION OF SYMBOLS FOR DFT

The ability to see is one of the remarkable characteristics of living beings. It enables them to perceive and assimilate in a very short time an incredible amount of knowledge about the world around them. The scope and variety of information that can pass through the eye and be interpreted by the brain is nothing short of astounding. Mankind has increased this basic capability by inventing devices that can detect electromagnetic radiation at wavelengths far outside the range of normal vision and at energy levels orders of magnitude below what the eye is able to perceive by itself. By the use of X-rays or sound waves it is possible to see inside objects and into places that have been invisible to living beings since the dawn of creation. Ultra-fast photography can stop a speeding bullet or freeze a flying humming bird's wing [128].

For thousands of years mankind has been creating pictures that attempt to portray real or imagined scenes as perceived by human vision. Cave drawings, paintings and photographs are able to stimulate the visual system and conjure up thoughts of far away places, imagined situations or pleasant sensations. The art of motion picture has advanced to the point where viewers often undergo intense emotional experiences. On-the-spot news coverage gives the impression of actually witnessing events as they unfold. Thus the visual representation of a problem will be easier to understand compared to other methods.

The DFT computation is an approach for the analysis of signals and systems in the frequency domain with well established algorithms that are in use for a long time. But still it is not that efficient in the 2-D applications. Currently the software packages are also giving more importance to the visual representations. Hence, a visual approach is developed in this work to do the 2-D DFT computation and a pictorial representation of them so that the analysis of 2-D signals in the frequency domain will be easy.

3.1 BASIC THEORY

Let the $N \times N$ data matrix is

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & A_{0,1} & \dots & A_{0,N-1} \\ A_{1,0} & A_{1,1} & \dots & A_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N-1,0} & A_{N-1,1} & \dots & A_{N-1,N-1} \end{bmatrix}$$

Let the DFT of the matrix \mathbf{A} be the matrix \mathbf{Y} given by

$$\mathbf{Y} = \begin{bmatrix} Y_{0,0} & Y_{0,1} & \dots & Y_{0,N-1} \\ Y_{1,0} & Y_{1,1} & \dots & Y_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{N-1,0} & Y_{N-1,1} & \dots & Y_{N-1,N-1} \end{bmatrix}$$

$$\text{where } Y_{k_1, k_2} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} A_{n_1, n_2} W_N^{n_1 k_1 + n_2 k_2}, 0 \leq k_1, k_2 \leq N-1 \quad (3.1)$$

In direct DFT, the computation of each coefficient Y_{k_1, k_2} requires N^2 complex multiplications, $N(N-1)$ complex additions and there are N^2 coefficients. But the computation of 2×2 point DFT doesn't require any multiplications. Hence, if we can express the $N \times N$ point DFT in terms of 2×2 point DFTs, the complex multiplications can be reduced. Therefore, the matrix \mathbf{A} is partitioned into non-overlapping 2×2 matrices as

$$\mathbf{A}_{1_{0,0}} = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}, \mathbf{A}_{1_{0,1}} = \begin{bmatrix} A_{0,2} & A_{0,3} \\ A_{1,2} & A_{1,3} \end{bmatrix} \text{ etc.}$$

The DFT of $\mathbf{A}_{1_{0,0}}$ is

$$\begin{aligned} X_{0,0} &= A_{0,0} + A_{0,1} + A_{1,0} + A_{1,1} \\ X_{0,1} &= (A_{0,0} + A_{1,0}) - (A_{0,1} + A_{1,1}) \\ X_{1,0} &= (A_{0,0} + A_{0,1}) - (A_{1,0} + A_{1,1}) \\ X_{1,1} &= (A_{0,0} + A_{1,1}) - (A_{1,0} + A_{0,1}) \end{aligned}$$

Similarly, the DFT of other 2x2 matrices can be obtained. Then the DFT of the partitioned matrices can be written as \mathbf{X}

$$\mathbf{X} = \begin{bmatrix} X_{0,0} & X_{0,1} & X_{0,N-1} \\ X_{1,0} & X_{1,1} & X_{1,N-1} \\ X_{N-1,0} & X_{N-1,1} & X_{N-1,N-1} \end{bmatrix}$$

The $N \times N$ point DFT coefficients, Y_{k_1, k_2} , are expressed in terms of the elements of \mathbf{X} by using the relations for $N \times N$ point DFT & 2x2 point DFT as well as the property that $W_N^{p+N/2} = -W_N^p$. The relation between the matrices \mathbf{X} & \mathbf{Y} for $N = 6$ is given in appendix A. Thus, it is found that the $N \times N$ point DFT coefficients can be expressed as

$$Y_{k_1, k_2} = \sum_{p=0}^{N/2-1} Y_{k_1, k_2}^{(p)} W_N^p, \quad 0 \leq k_1, k_2 \leq N-1 \quad (3.2)$$

where $Y_{k_1, k_2}^{(p)}$ will be the sum of few 2x2 point DFT coefficients. Scaling factors of 1/4 or 1/2 respectively arise if 4 or 2 of the DFT coefficients from the respective 2x2 matrix are present in $Y_{k_1, k_2}^{(p)}$. Most of the coefficients $Y_{k_1, k_2}^{(p)}$ depend on all the 4 coefficients from a 2x2 matrix. This can be replaced with one data and hence the additions can be reduced. But for visual manipulation, the pictorial representation in terms of 2x2 DFT shows regular patterns better than the representation in terms of the data. So the pictorial representation is in terms of 2x2 DFTs and computation of DFT coefficients are in terms of the data itself.

From the equations (3.1) & (3.2), a relation between $Y_{k_1, k_2}^{(p)}$ and the data can be derived.

$$\begin{aligned} Y_{k_1, k_2} &= \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} A_{n_1, n_2} W_N^{n_1 k_1 + n_2 k_2} \\ &= \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} A_{n_1, n_2} W_N^{((n_1 k_1 + n_2 k_2))_N} \end{aligned}$$

Since W_N is periodic, with period N , $n_1 k_1 + n_2 k_2$ in the above equation can be replaced by $((n_1 k_1 + n_2 k_2))_N$. In equation (3.2) the factor W_N^p can be thought of as an $N/2$ dimensional vector with $Y_{k_1, k_2}^{(p)}$ as the respective scaling factor. Comparing equations (3.1) & (3.2) and using the periodicity and symmetry properties of the twiddle factor, $Y_{k_1, k_2}^{(p)}$ can be expressed as

$$Y_{k_1, k_2}^{(p)} = \sum_{\forall (n_1, n_2) z=p} A_{n_1, n_2} - \sum_{\forall (n_1, n_2) z=p+N/2} A_{n_1, n_2} \quad (3.3)$$

$$\text{where } z = ((n_1 k_1 + n_2 k_2))_N \quad (3.4).$$





Thus DFT computation using equations (3.2) & (3.3) can be interpreted as follows. The 2-D data in the spatial domain with index (n1, n2) is mapped to the frequency domain with index (k1, k2) in two steps. First, the data is mapped on to each of the vectors formed from the twiddle factors W_N^p with the mapping function as $((n1. k1 + n2. k2))_N = p$. Since $W_N^{p+N/2} = -W_N^p$, only N/2 dimensional vector is considered. The remaining N/2 fold over with a sign reversal. The algebraic sum of the data that is mapped over to the vector W_N^p will be the real value $Y_{k1,k2}^{(p)}$. Then the weighted sum of these vectors W_N^p and the corresponding $Y_{k1,k2}^{(p)}$ gives the DFT coefficient $Y_{k1,k2}$.

$Y_{k1,k2}^{(p)}$ can be represented pictorially using a set of primitive symbols derived from the relation between 2x2 data and 2x2 DFT coefficients. This gives a visual representation of the relation between NxN point DFT and 2x2 point DFTs. This can be easily converted to a visual representation in terms of NxN data by replacing the primitive symbols with the corresponding data. Also a parallel distributed approach can be derived for the DFT computation as explained in the next chapter.




3.2 DERIVATION OF THE PRIMITIVE SYMBOLS




Let $A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$, $X = \begin{bmatrix} X_{0,0} & X_{0,1} \\ X_{1,0} & X_{1,1} \end{bmatrix}$ be the 2x2 data and corresponding DFT matrices

respectively. Then

$A_{0,0} + A_{0,1} + A_{1,0} + A_{1,1} = X_{0,0}$	LA	
$(A_{0,0} + A_{1,0}) - (A_{0,1} + A_{1,1}) = X_{0,1}$	RA	
$(A_{0,0} + A_{0,1}) - (A_{1,0} + A_{1,1}) = X_{1,0}$	LB	
$(A_{0,0} + A_{1,1}) - (A_{1,0} + A_{0,1}) = X_{1,1}$	RB	

The corresponding mnemonic and the pictorial representations in terms of 2x2 DFT are shown on the right side of the above equations. From the above four data - DFT relations we can derive the following expressions.

$A_{0,0} + A_{0,1} = [X_{0,0} + X_{1,0}] / 2$	VLP	
$A_{0,0} + A_{1,0} = [X_{0,0} + X_{0,1}] / 2$	HAP	
$A_{0,0} + A_{1,1} = [X_{0,0} + X_{1,1}] / 2$	DP	

$A_{0,1} + A_{1,0} = [X_{0,0} - X_{1,1}] / 2$	DPA	
$A_{0,1} + A_{1,1} = [X_{0,0} - X_{0,1}] / 2$	HAPL	
$A_{1,0} + A_{1,1} = [X_{0,0} - X_{1,0}] / 2$	VLPA	
$A_{0,0} - A_{0,1} = [X_{0,1} + X_{1,1}] / 2$	VRP	
$A_{0,0} - A_{1,0} = [X_{1,0} + X_{1,1}] / 2$	HBP	
$A_{0,0} - A_{1,1} = [X_{1,0} + X_{0,1}] / 2$	CP	
$A_{0,1} - A_{1,0} = [X_{1,0} - X_{0,1}] / 2$	CPB	
$A_{0,1} - A_{1,1} = [X_{1,0} - X_{1,1}] / 2$	HBPL	
$A_{1,0} - A_{1,1} = [X_{0,1} - X_{1,1}] / 2$	VRPA	
$A_{0,0} = [X_{0,0} + X_{1,0} + X_{0,1} + X_{1,1}] / 4$	MP	
$A_{0,1} = [X_{0,0} + X_{1,0} - X_{0,1} - X_{1,1}] / 4$	MPL	
$A_{1,0} = [X_{0,0} + X_{0,1} - X_{1,0} - X_{1,1}] / 4$	MPA	
$A_{1,1} = [X_{0,0} + X_{1,1} - X_{0,1} - X_{1,0}] / 4$	MPD	



In order to understand the picture, in general,


“ indicates ‘+’ sign for the DFT coefficient.


“ ■ “ indicates ‘-’ sign for the DFT coefficient.


Continuous lines indicate positive sign and dotted lines for negative sign.


The meaning of typical symbols and mnemonics used in the pictorial representation are given below:

1. Symbol  named MP (for Matrix Positive) indicates that all the four DFT coefficients in the 2x2 DFT matrix are to be added to compute $Y_{k_1, k_2}^{(p)}$. Equivalently, the same result will be obtained if we consider the data at the top-left corner of the concerned 2x2 data matrix.
2. Symbol  named MPL (for Matrix Positive on the Left) indicates that the two DFT coefficients on the left (i.e. (0,0)th and (1,0)th position) are to be added from which the sum of DFT coefficients on the right (i.e. on the (0,1)th and (1,1)th position) of the 2x2 DFT matrix are to be subtracted. Equivalently, we may consider the data on the top right position, i.e. (0,1)th position, of the 2x2 data matrix to compute $Y_{k_1, k_2}^{(p)}$.

3. Symbol  is called by the name VLP (for Vertical Left Positive). VLP indicates that the two DFT coefficients on the left of the 2x2 DFT matrix are to be summed in the process of finding $Y_{k_1, k_2}^{(p)}$. Equivalently, we may consider the sum of the (0,0) and (0,1) data elements of the 2x2 data matrix.

4. The symbol  is called HAPL (Horizontal Above- Positive Left). HAPL indicates that in a cell, the DFT coefficients on the top right position (i.e. (0,1)th position) is subtracted from the DFT coefficient on the top left (i.e. (0,0)th position of 2x2 DFT) during the computation of $Y_{k_1, k_2}^{(p)}$. Equivalently, the data in the (0,1)th and (1,1)th position of the data matrix may be summed to get the same value.

5. The symbol  called DP (Diagonal Positive) when used indicates that in a cell, the diagonal element occupying the bottom position, i.e. the (1,1)th position is to be added to the diagonal element at the top, i.e., (0,0)th position while computing $Y_{k_1, k_2}^{(p)}$. Equivalently, the diagonal elements in the data cell may be added to get the same result.

6. The symbol  called LA, (Left Above) depending on its position in the 2x2 DFT matrix indicates that only the DFT coefficient at its position has to be considered while computing $Y_{k_1, k_2}^{(p)}$. Equivalently, to get the same result, all the 4 data elements have to be summed for LA.

Also, generally, for the DFT matrix

(a) Whenever all the four DFT coefficients of a cell are involved, a scaling factor of 1/4 is to be applied to the result of computation.

(b) If only two coefficients are involved, a scaling factor of 1/2 has to be applied.

(c) If only one coefficient is involved, no scaling factor has to be applied.

Similarly, other data-DFT relations are derived and named them using the nomenclature given below.

A - Above	B - Below	R - Right	L - Left
V - Vertical	H - Horizontal	D - Diagonal	C - Cross-diagonal
M - Matrix	P - Positive	N - Negative.	

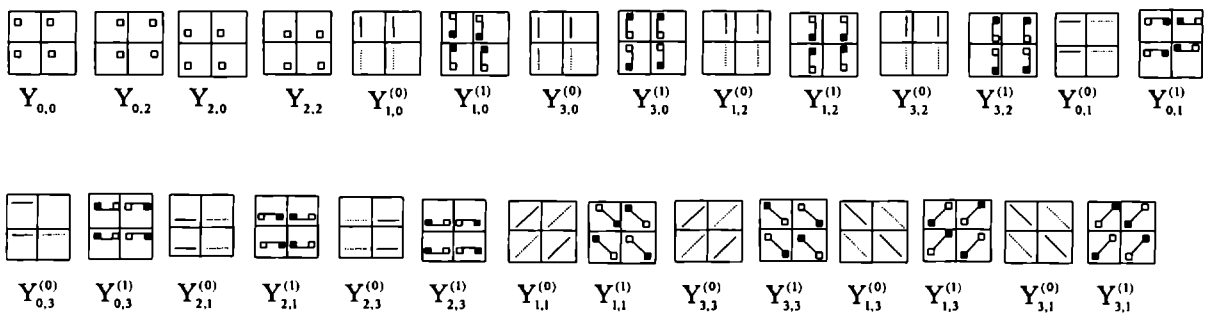
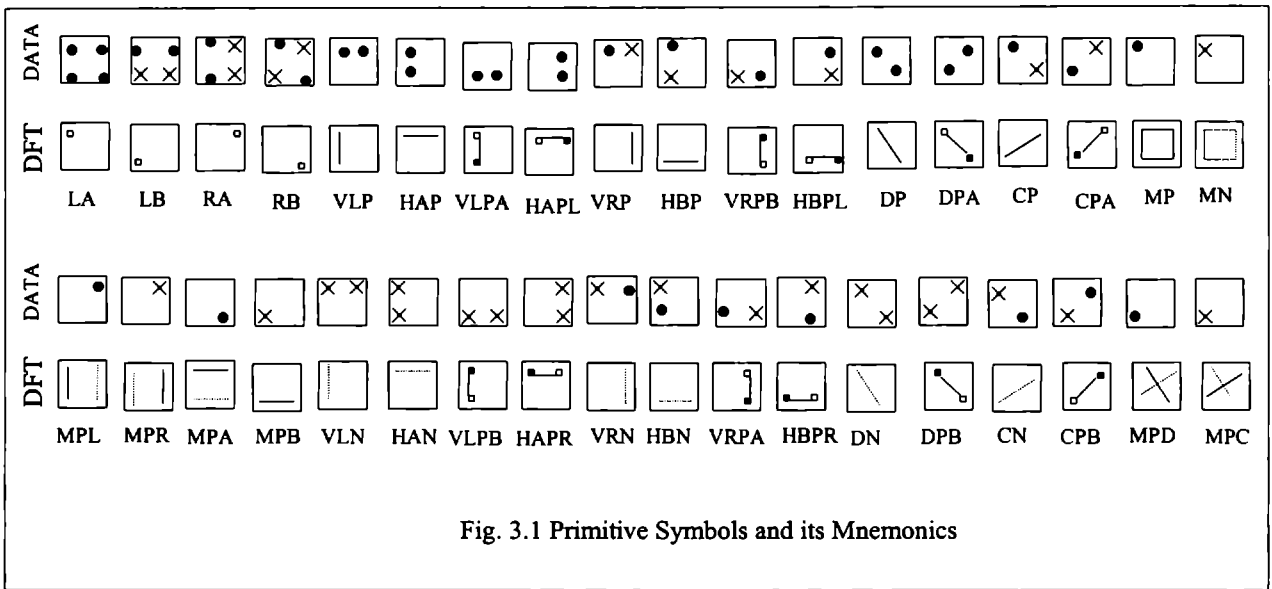
Using the above pictorial representation of 2x2 DFTs, the pictorial representation for NxN-point DFT coefficient Y_{k_1, k_2} can be obtained in terms of $Y_{k_1, k_2}^{(p)}$.

3.3 PICTORIAL REPRESENTATION

3.3.1 Direct Method

This method can be used to represent visually and compute the value of 2-D DFT of all or a few selected indices when the order of the data matrix is an even number N . Let $\mathbf{A} = [A_{i,j}]$ and $\mathbf{Y} = [Y_{i,j}]$, $0 \leq i,j \leq N-1$ be the $N \times N$ data and DFT matrices respectively related by (3.1). The coefficient Y_{k_1,k_2} can be expressed in terms of the real coefficients $Y_{k_1,k_2}^{(p)}$ using (3.2) where $Y_{k_1,k_2}^{(p)}$ is related to the data as in (3.3)

The real coefficient $Y_{k_1,k_2}^{(p)}$, for any (k_1, k_2) and $p = 0, 1, \dots, M-1$ where $M = N/2$, is represented pictorially, in terms of 2×2 point DFT coefficients, as explained below. The pictorial representation of $Y_{k_1,k_2}^{(p)}$ will have $M \times M$ cells with either a primitive symbol or a blank in each cell. Each cell represents a choice of mapping of 2×2 data into 2×2 DFT based on the value of z in (3.3). Hence, for each time index (n_1, n_2) , the value of $z = ((n_1 \cdot k_1 + n_2 \cdot k_2))_N$ is determined. If $z = p$, the data is to be added and if $z = p + M$, the data is to be subtracted. Other conditions are to be neglected. The primitive symbol to be used in each cell can be selected using the data-DFT relation, given in fig. 3.1, based on the number and sign of data used from that cell. For eg: for any even value of N , if $(k_1, k_2) = (0, 0)$ then $((n_1 \cdot k_1 + n_2 \cdot k_2))_N = 0$ for all values of (n_1, n_2) in the $M \times M$ cells. Thus all the four data in each cell will be involved with plus sign (equivalently one DFT coefficient at left & above position with mnemonic 'LA') if $p=0$. Thus the pictorial representation for $k_1 = k_2 = 0$ exists only for $p = 0$. Similarly, for $k_1 = 0$ & $k_2 = M$, $z = 0$ or M when n_2 is even or odd respectively. Thus in the columns of even n_2 , the data is to be added and for odd values of n_2 the data is to be subtracted. Equivalently, all the cells will have the symbol corresponding to the mnemonic 'RA' for $p=0$ alone. Similarly we can construct the pictorial representation for any (k_1, k_2) corresponding to any even value of N . The fig. 3.2, 3.3, 3.4, 3.5 & 3.6 show the pictorial representation for $N = 4, 6, 8, 10$ & 12 respectively. Similarly, the pictorial representation for other values of N can be derived. The fig. 3.2, 3.3 & 3.4 gives the complete set of coefficients for $N = 4, 6$ & 8 in which related coefficients are placed closer for ease of comparison. In fig. 3.5 multicolor representations are used so that a number of coefficients can be represented in one bin to enable easy analysis as well as to reduce the space consumption. The color codes used are self-explanatory. All the coefficients in



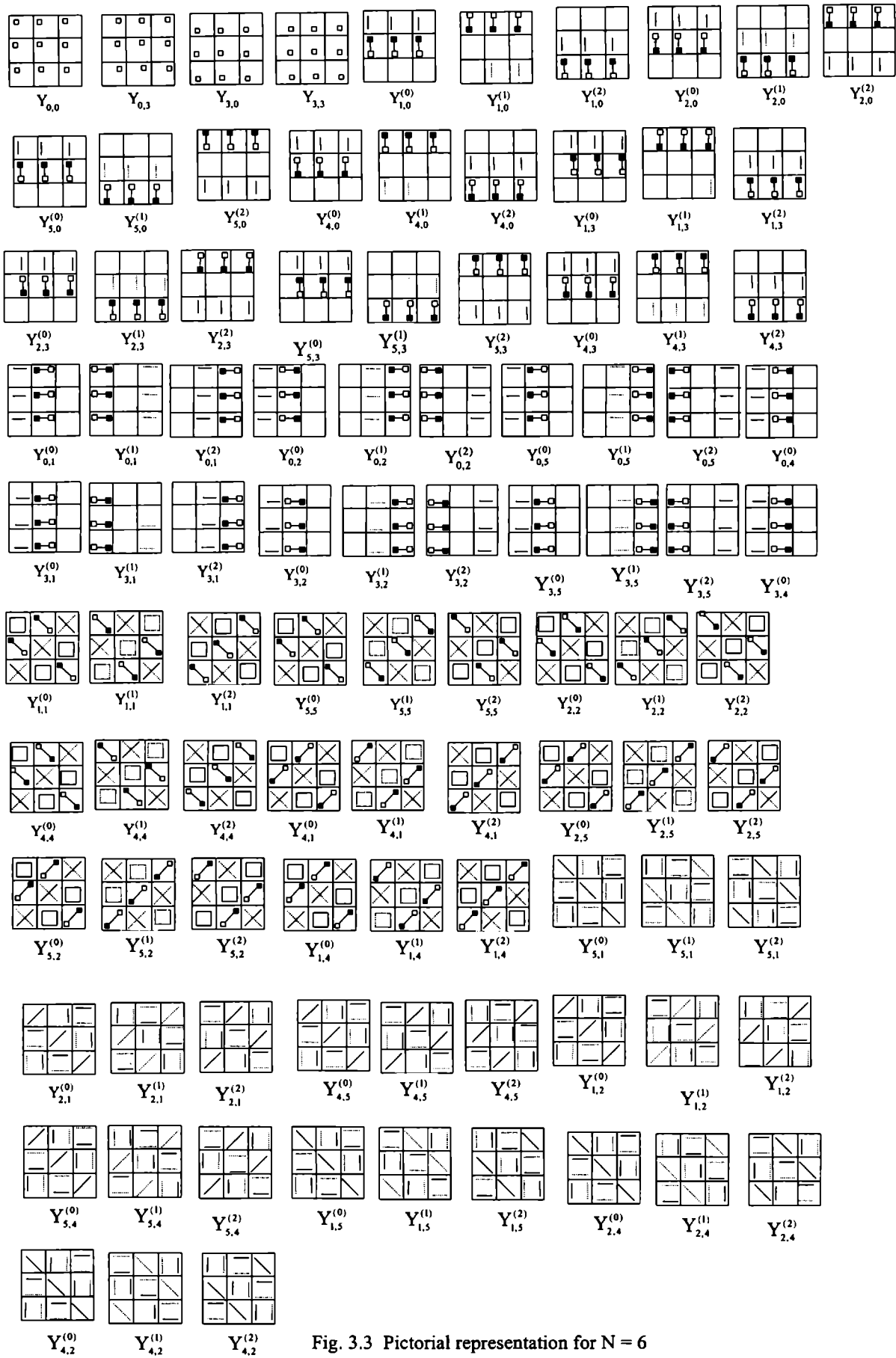


Fig. 3.3 Pictorial representation for $N = 6$

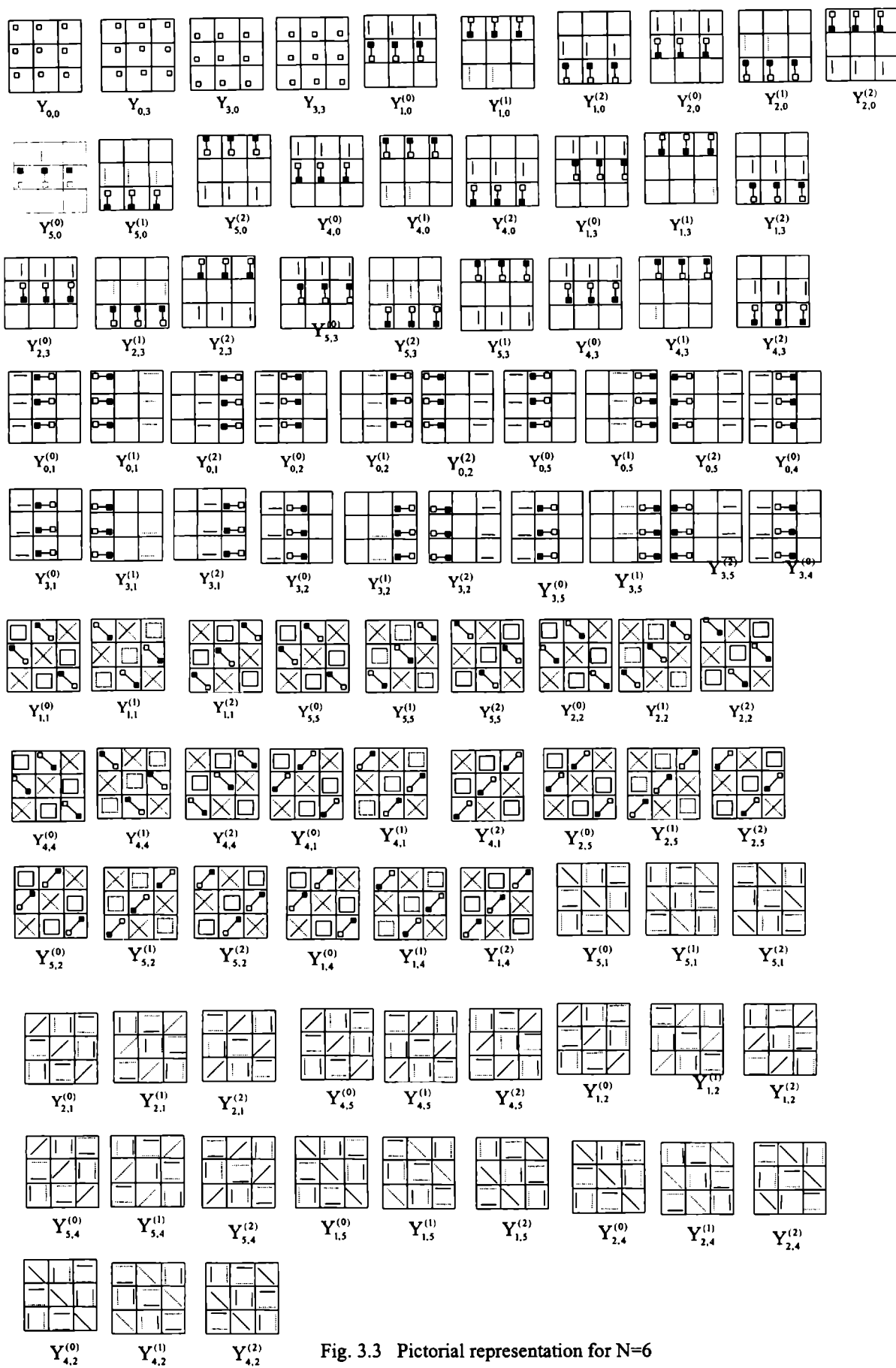


Fig. 3.3 Pictorial representation for $N=6$

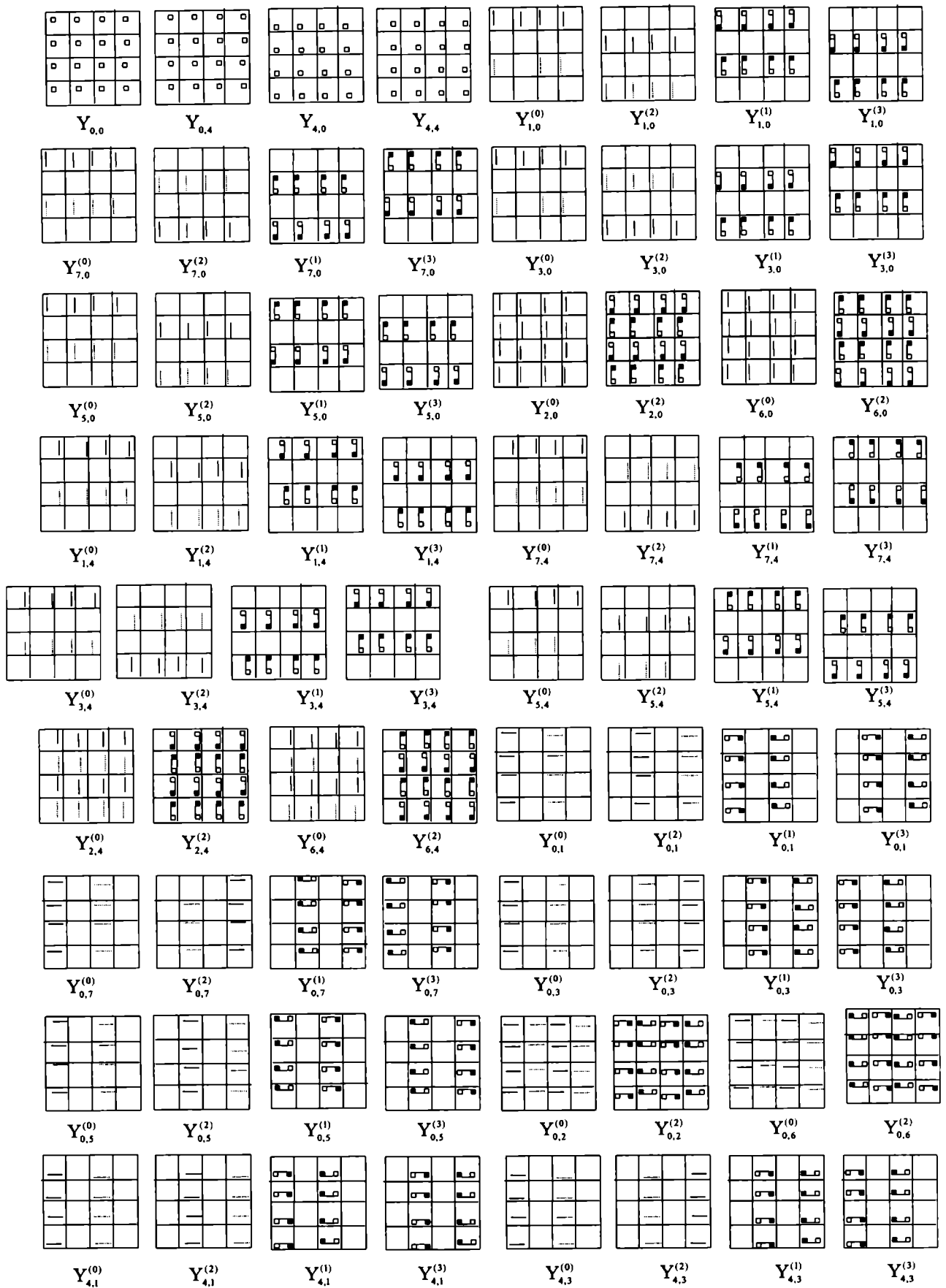


Fig. 3.4 Pictorial representation for N=8

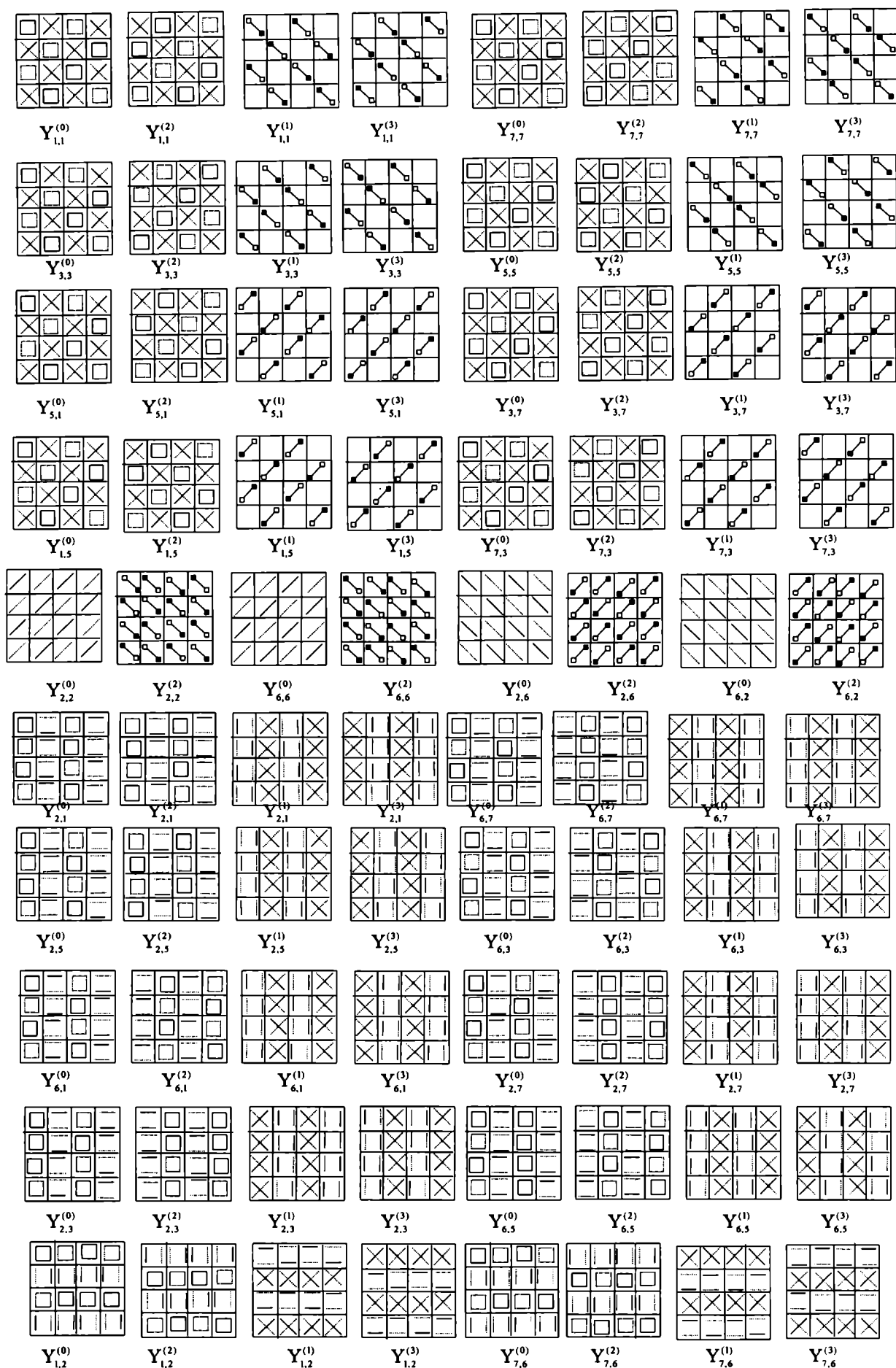


Fig. 3.4 Pictorial representation for $N=8$ (contd.)

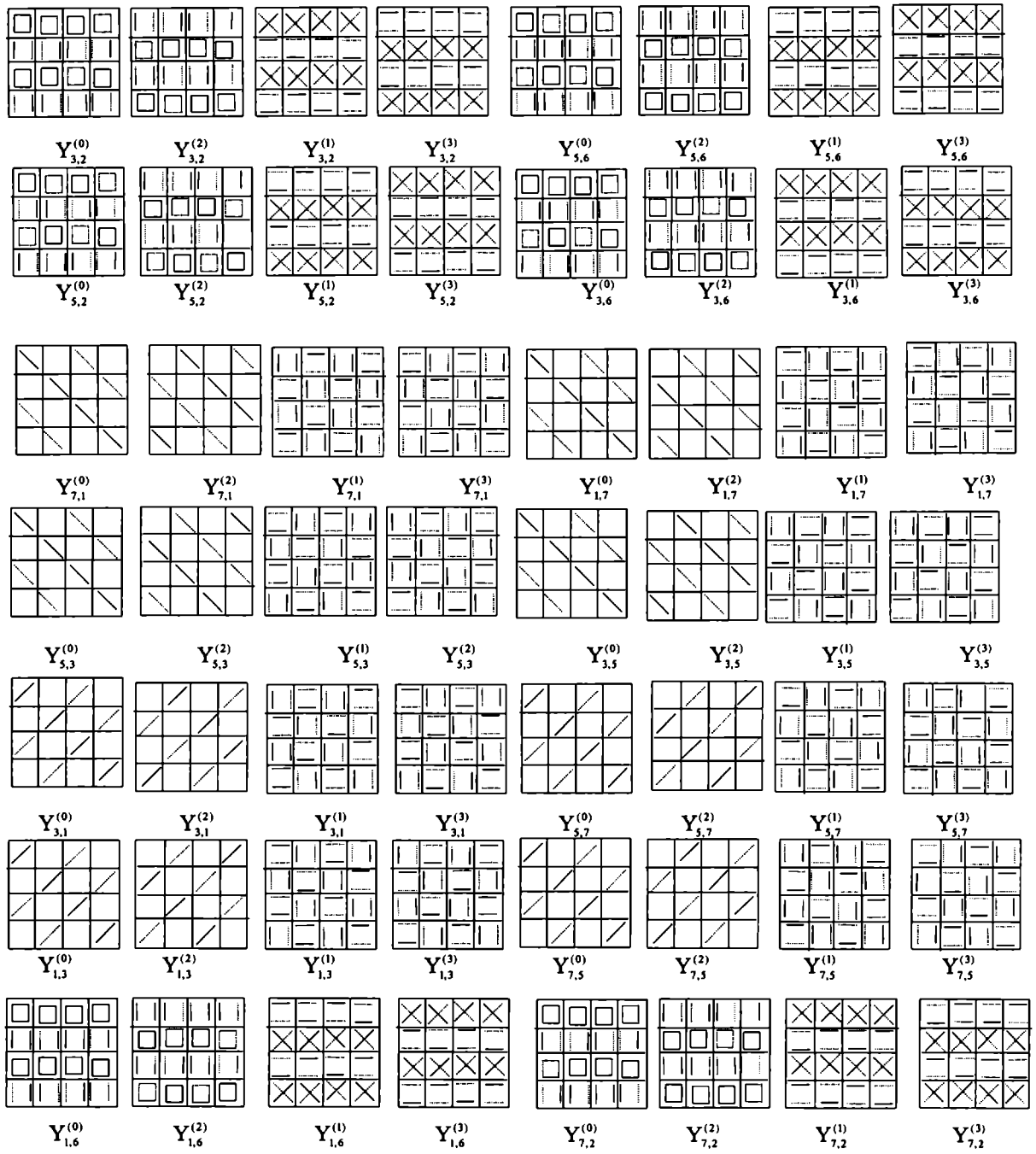


Fig. 3.4 Pictorial representation for N=8 (Contd.)

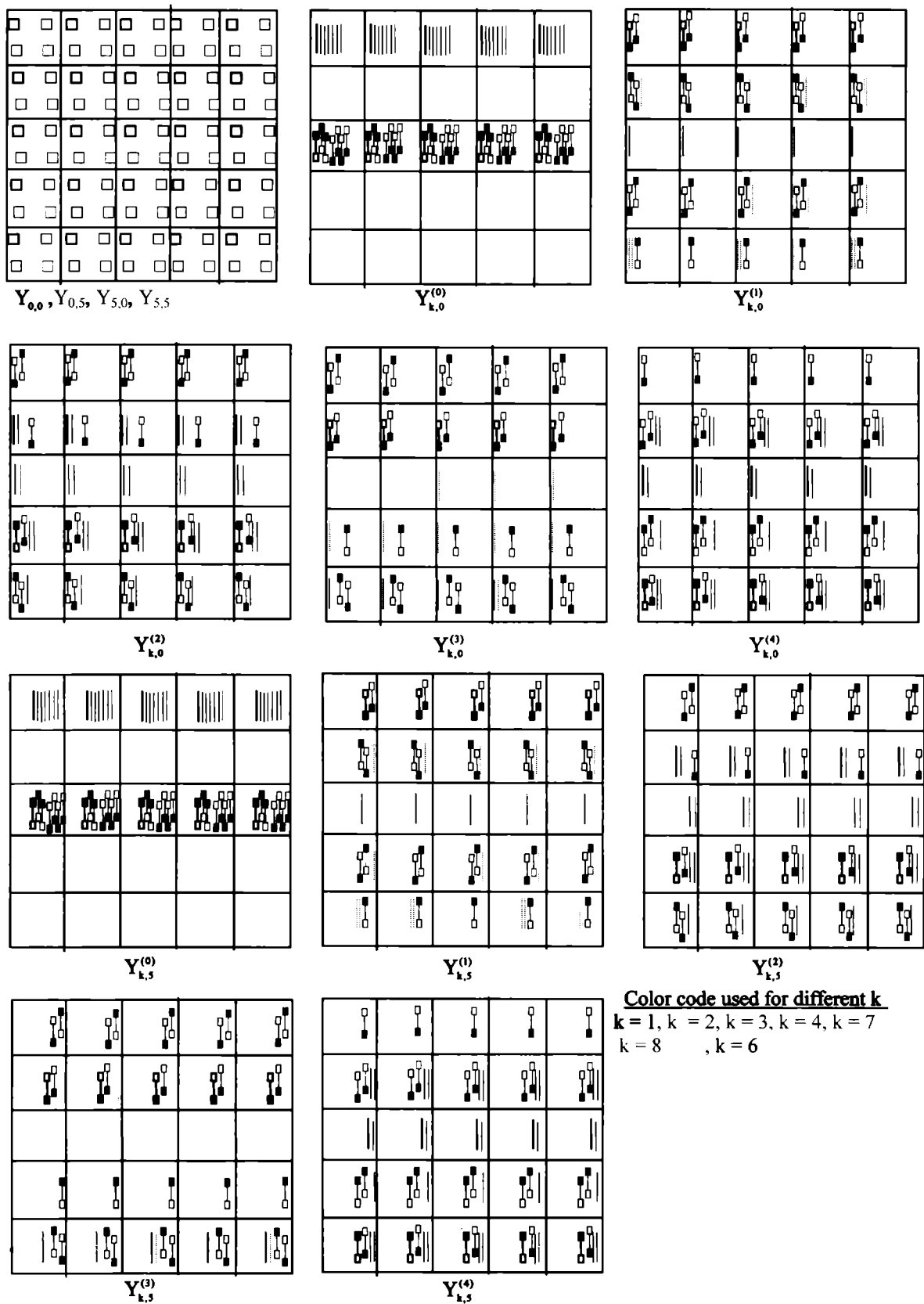
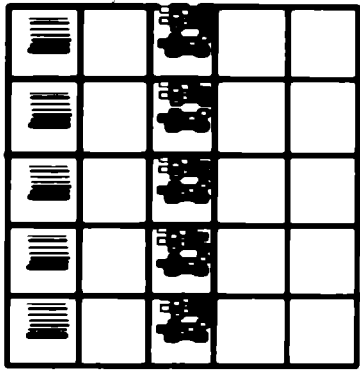
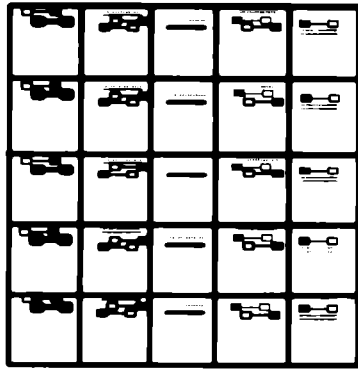


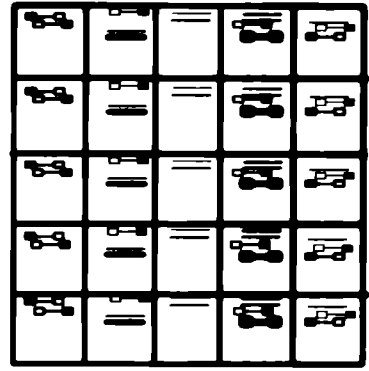
Fig. 3.5 Pictorial representation for $N=10$



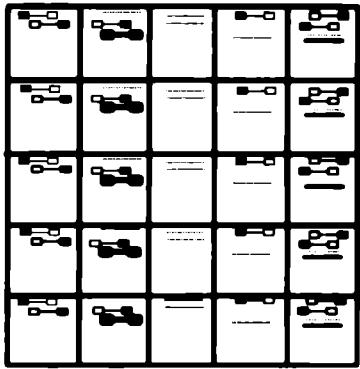
Y_{10}^3



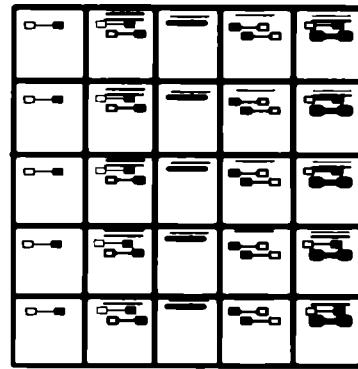
Y_{10}^2



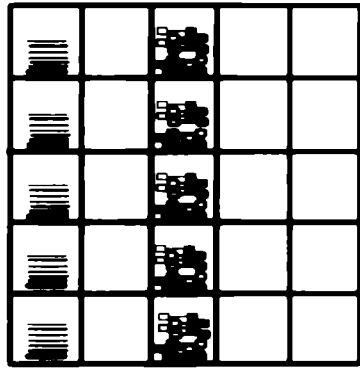
Y_{10}^1



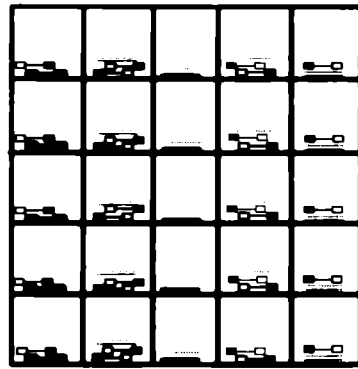
Y_{10}^4



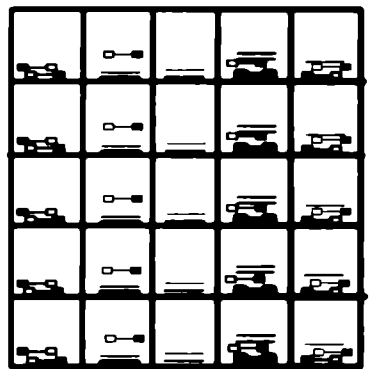
Y_{10}^5



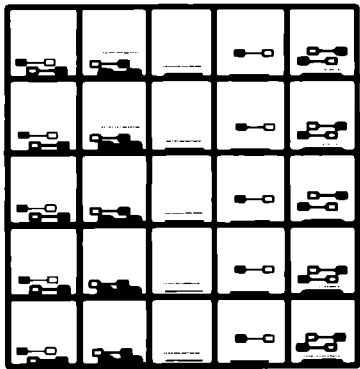
Y_{10}^6



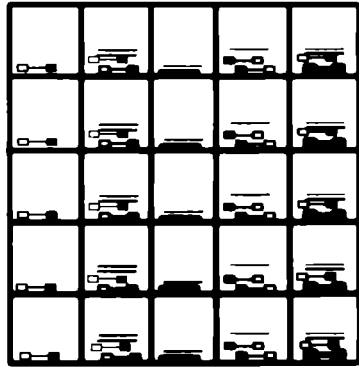
Y_{10}^7



Y_{10}^8



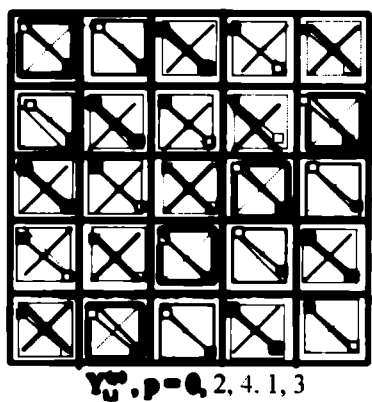
Y_{10}^9



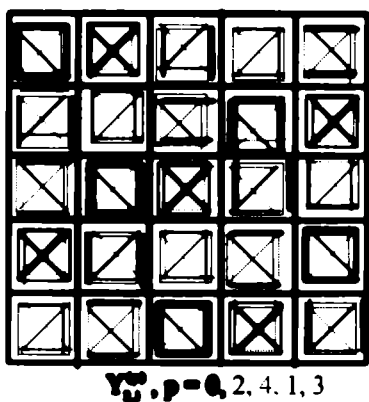
Y_{10}^{10}

For group II coefficients
 $k = 1, 2, 3, 4, 6, 7, 8.$

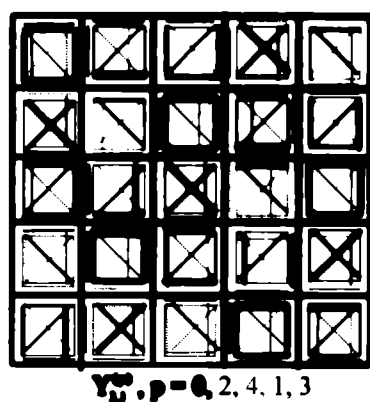
Fig. 3.5 Pictorial representation for N=10 (contd.)



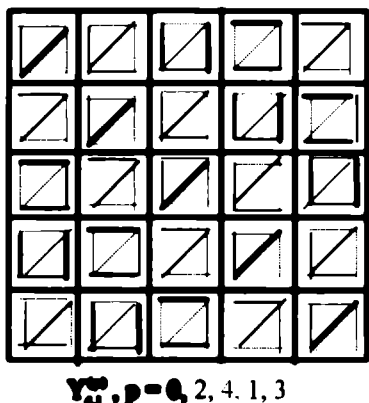
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



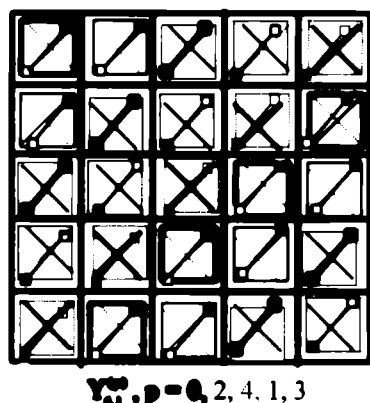
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



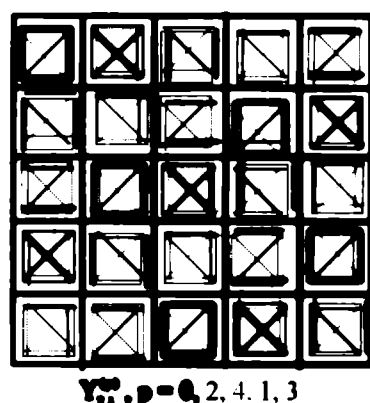
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



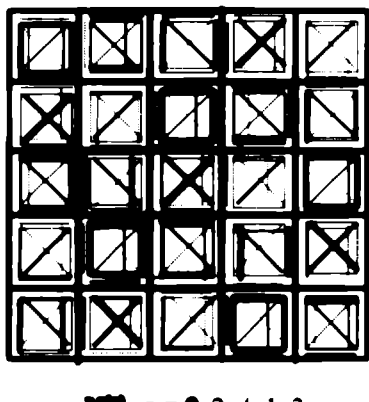
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



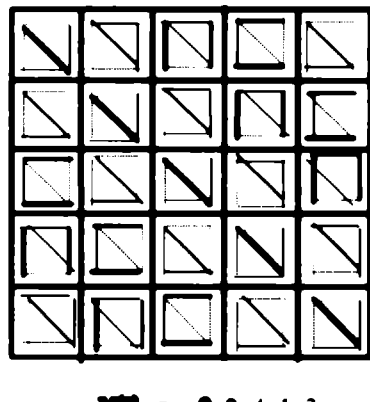
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



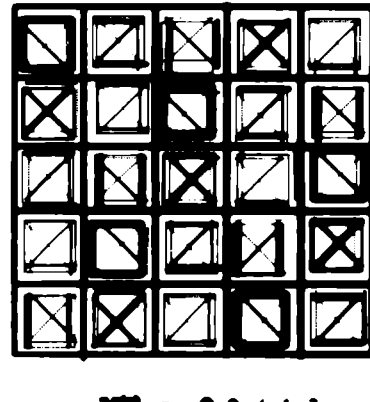
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



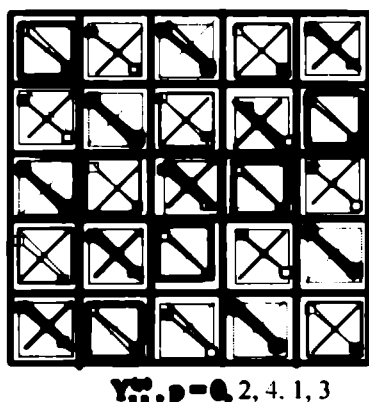
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



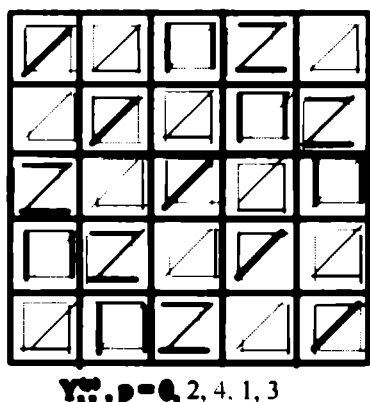
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



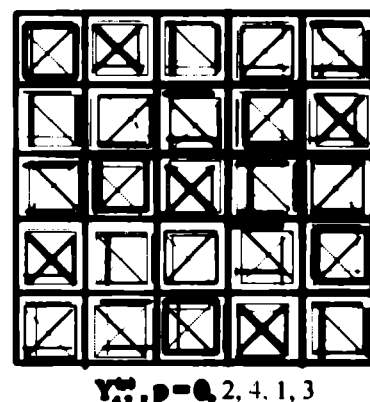
$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$



$Y_{10}^{\text{III}}, p=0, 2, 4, 1, 3$

Fig. 3.5 Pictorial representation for N=10 (Contd.)

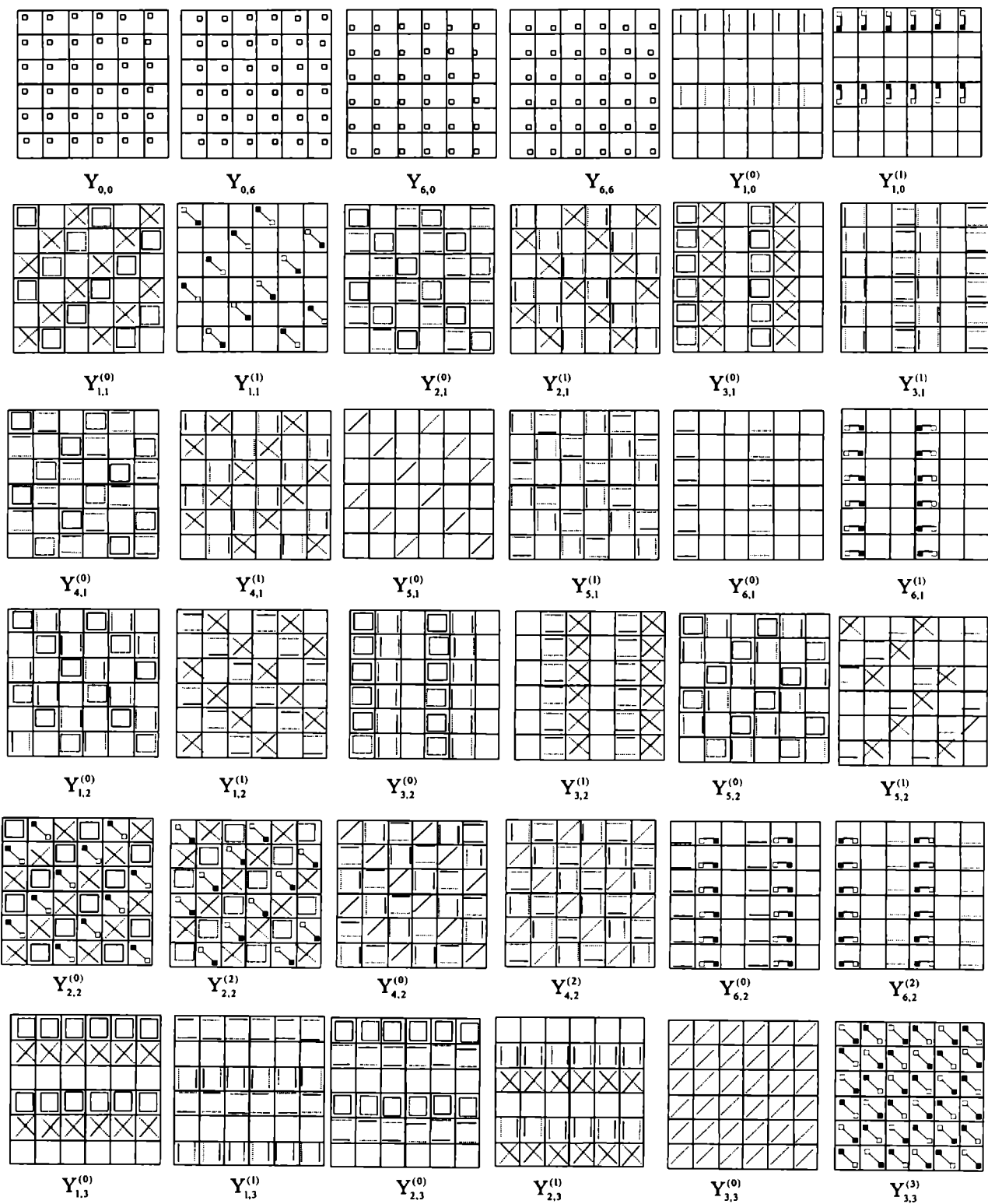


Fig. 3.6 Pictorial Representation for N=12

group 1 are represented in one bin. For group 2 coefficients, all the coefficients in one column / row are represented in one bin for each p. For example, $Y_{k,0}^{(p)}$ for $p = 0$ & $k = 1, 2, 3, 4, 6, 7, 8$ & 9 are shown in one bin with one color for each k. The basic set of coefficients from group 3 are also shown in fig. 3.5 with one coefficient for all values of p, differentiated by color as indicated in the fig. 3.5, in one bin. Similarly other coefficients can also be represented. A few coefficients corresponding to $N=12$ are shown in fig. 3.6 for the first two values of p for which the representations are different and not blank.

3.3.2 Analysis of the pictorial representation

3.3.2.1 $N=4$

$$Y = \begin{bmatrix} \textcircled{Y_{0,0}} & Y_{0,1} & \textcircled{Y_{0,2}} & Y_{0,3} \\ Y_{1,0} & Y_{1,1} & Y_{1,2} & Y_{1,3} \\ \textcircled{Y_{2,0}} & Y_{2,1} & \textcircled{Y_{2,2}} & Y_{2,3} \\ Y_{3,0} & Y_{3,1} & Y_{3,2} & Y_{3,3} \end{bmatrix}$$

The matrix above shows the grouping of coefficients corresponding to 4x4-point DFT and its pictorial representations are shown in fig. 3.2. The DFT coefficients are classified into three groups depending on the way the primitive symbols are present in them. The circled coefficients (ie; $Y_{0,0}$, $Y_{0,2}$, $Y_{2,0}$, $Y_{2,2}$) represent group 1 and the coefficients $Y_{1,0}$, $Y_{0,1}$, $Y_{3,0}$, $Y_{0,3}$, $Y_{1,2}$, $Y_{2,1}$, $Y_{3,2}$, $Y_{2,3}$ that are between two dotted lines represent group 2 and the remaining coefficients form group 3. The group 1 coefficients depend on one DFT coefficient from each 2x2 cell and it shows a positional relation with that of the index. Similarly, the group 2 coefficients also show a positional relation. That is, the coefficients ($Y_{1,0}$, $Y_{3,0}$) in the left column marked as group 2 will depend on the two coefficients on the left of the cells (VLP, VLN for even p & VLPA, VLPB for odd p), the coefficients in the right column ($Y_{1,2}$, $Y_{3,2}$) will depend on the two coefficients on the right of the cells (VRP, VRN for even p & VRPA, VRPB for odd p), the coefficients in the top row ($Y_{0,1}$, $Y_{0,3}$) will depend on the two coefficients on the top position of the cells (HAP, HAN for even p & HAPL, HAPR for odd p) and that in the lower row ($Y_{2,1}$, $Y_{2,3}$) will depend on the two coefficients on the lower position of the cells (HBP, HBN for even p & HBPL, HBPR for odd p). The group 3

coefficients also show a specific pattern depending on the position of the frequency index. That is, the coefficients on the diagonal position ($Y_{1,1}, Y_{3,3}$) depend on (CP, CN for even p & DPA, DPB for odd p) and that on the cross-diagonal position ($Y_{1,3}, Y_{3,1}$) depend on (DP, DN for even p & CPA, CPB for odd p). Thus, when N=4, the pictorial representation corresponding to even p will use one set of primitive symbols and that for odd p will use another set of primitive symbols in groups 2 & 3.

The fig. 3.2 shows that the pictorial representation of a number of coefficients can be derived from that of one coefficient. The group 1 coefficients can be derived from one cell, since all the cells are identical for one coefficient and the others in the group are related by the operation of flipping with respect to an axis. That is, $Y_{0,M}$ can be obtained from $Y_{0,0}$ by flip horizontal operation, $Y_{M,0}$ by flipping vertically and $Y_{M,M}$ by rotate right and flip horizontal operations.

The group 2 coefficients are related as shown in the following and they can be derived by simple manipulation of few coefficients.

$$Y_{1,0}^{(0)} = Y_{3,0}^{(0)}, Y_{1,0}^{(1)} = -Y_{3,0}^{(1)}, Y_{1,2}^{(0)} = Y_{3,2}^{(0)}, Y_{1,2}^{(1)} = -Y_{3,2}^{(1)}, Y_{0,1}^{(0)} = Y_{0,3}^{(0)}, Y_{0,1}^{(1)} = -Y_{0,3}^{(1)}, Y_{2,1}^{(0)} = Y_{2,3}^{(0)}, Y_{2,1}^{(1)} = -Y_{2,3}^{(1)}.$$

$Y_{k,0}^{(p)}$ can be derived from $Y_{0,k}^{(p)}$ by rotate right and flip horizontal operations. For converting $Y_{0,k}^{(p)}$ to $Y_{M,k}^{(p)}$ use flip vertical and for changing from $Y_{k,0}^{(p)}$ to $Y_{k,M}^{(p)}$ use flip horizontal operations. Thus we can derive the representation of group 2 coefficients from that of single coefficient.

The group 3 coefficients are related as $Y_{1,1}^{(0)} = Y_{3,3}^{(0)}, Y_{1,1}^{(1)} = -Y_{3,3}^{(1)}, Y_{1,3}^{(0)} = Y_{3,1}^{(0)}, Y_{1,3}^{(1)} = -Y_{3,1}^{(1)}$ and they can be derived by simple manipulation of few coefficients. The coefficients in this group can be derived from one another by rotate right or rotate left operations.

3.3.2.2 N = 6

$$\mathbf{Y} = \begin{array}{cccccc}
 \textcircled{Y_{0,0}} & Y_{0,1} & Y_{0,2} & \textcircled{Y_{0,3}} & Y_{0,4} & Y_{0,5} \\
 Y_{1,0} & Y_{1,1} & Y_{1,2} & Y_{1,3} & Y_{1,4} & Y_{1,5} \\
 Y_{2,0} & Y_{2,1} & Y_{2,2} & Y_{2,3} & Y_{2,4} & Y_{2,5} \\
 \textcircled{Y_{3,0}} & Y_{3,1} & Y_{3,2} & \textcircled{Y_{3,3}} & Y_{3,4} & Y_{3,5} \\
 Y_{4,0} & Y_{4,1} & Y_{4,2} & \textcircled{Y_{4,3}} & Y_{4,4} & Y_{4,5} \\
 Y_{5,0} & Y_{5,1} & Y_{5,2} & \textcircled{Y_{5,3}} & Y_{5,4} & Y_{5,5}
 \end{array}$$

The matrix above shows the indices corresponding to 6x6 DFT. Fig. 3.3 shows the pictorial representation for $N = 6$. The DFT coefficients are classified into three groups depending on the way the primitive symbols are present in them. The circled coefficients represent group 1, the coefficients between two dotted lines represent group 2 and the remaining coefficients form group 3. Group 1 corresponds to the four real coefficients with indices $(0,0)$, $(0,M)$, $(M,0)$, (M,M) , where $M = N/2 = 3$, having $Y_{k_1,k_2}^{(p)}$ for $p=0$ only. Group 2 with $4(N-2)=16$ coefficients having the indices $(k,0)$, (k, M) , $(0,k)$, (M,k) where $k=1,2,\dots, N-1$ and $k \neq M$. Group 3 consists of the remaining $N^2-4(N-1) = 16$ coefficients.

Fig. 3.3a shows the pictorial representation of group 1 coefficients in one bin. The group 1 coefficients depend on one 2x2 DFT coefficient in each cell and they show a positional relation with the frequency index. For example, $Y_{0,0}$ will depend on the 2x2 DFT coefficient at the top left corners of each cell (with mnemonic LA) represented by ‘♣’ whereas $Y_{0,3}$ will depend on that at the top right position(RA), $Y_{3,0}$ depends on 2x2 DFT coefficients at the left and bottom position(LB) and $Y_{3,3}$ depends on that at the right and below (RB) in each cell. In this group, all the cells are identical for a coefficient and hence only one cell will be sufficient to construct its visual representation. The group 1 coefficients for any even value of N will have a similar representation with $M \times M$ identical cells as that of $N=6$.

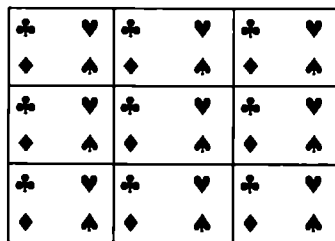


Fig. 3.3a Pictorial representation of group 1 coefficients for $N=6$
 ♣ - represents the 2x2 - point DFT coefficients for computing $Y_{0,0}$.

Similarly, ♦ - $Y_{3,0}$, ♥ - $Y_{0,3}$, ♠ - $Y_{3,3}$

Group 2 coefficients will have $Y_{k_1,k_2}^{(p)}$ for $p = 0, 1, \dots, M-1$. The Fig. 3.3b shows the simplified form of the pictorial representation of all the coefficients in column zero, column M , row zero and row M of group 2 coefficients. The numbers in the circles represent the index k and unfilled circles represent positive sign for the 2x2 DFT coefficient at that position whereas filled circles represent that with a negative sign. The fig. 3.3b shows that for even p , $Y_{k,0}^{(p)}$ depends on VLP and VLPA (ie., the two 2x2 DFT coefficients at the left of the cells)

12	45	12	45	12	45
45	12	45	12	45	12
12	45	12	45	12	45
15	12	15	12	15	12

$Y_{k,0}^{(0)}$

11	11	11
14	14	14
25	25	25
25	25	25
12	12	12
15	15	15
15	15	15
12	12	12

$Y_{k,0}^{(1)}$

2	2	2
5	5	5
14	14	14
14	14	14
12	12	12
45	45	45
15	15	15
12	12	12

$Y_{k,0}^{(2)}$

12	45	12	45	12	45
45	12	45	12	45	12
12	45	12	45	12	45
15	12	15	12	15	12
12		12		12	

$Y_{k,M}^{(0)}$

11	11	11
14	14	14
25	25	25
25	25	25
12	12	12
15	15	15
15	15	15
12	12	12

$Y_{k,M}^{(1)}$

25	25	25
52	52	52
14	14	14
14	14	14
12	12	12
45	45	45
15	15	15
12	12	12

$Y_{k,M}^{(2)}$

12	12	12	12	12	12
45	45	45	45	45	45
12	12	12	12	12	12
45	15	45	15	45	15

$Y_{0,k}^{(0)}$

11	14	11	14	11	14
25	25	25	25	25	25
12	12	12	12	12	12
15	15	15	15	15	15

$Y_{0,k}^{(1)}$

25	52	25	52	25	52
14	14	14	14	14	14
12	12	12	12	12	12
45	15	45	15	45	15

$Y_{0,k}^{(2)}$

Fig. 3.3b Pictorial representation of group 2 coefficients for N=6

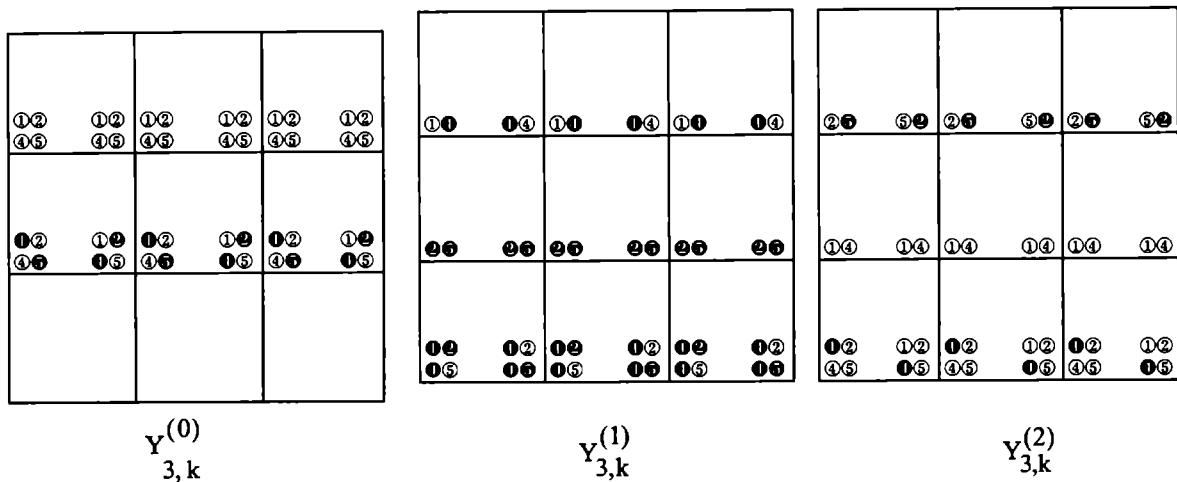


Fig. 3.3b Pictorial representation of group 2 coefficients for N=6 (cond)

and its sign reversed form for odd p (ie, $VLN=-VLP$ and $VLPB=-VLPA$). $Y_{k,3}^{(p)}$ depends on VRP and VRPA (the DFT coefficients at the right of the cells) for even p and its sign reversed form for odd values of p . Similarly, $Y_{0,k}^{(p)}$ & $Y_{3,k}^{(p)}$ will depend on HAP, HAPL and HBP, HBPL respectively for even p and corresponding sign reversed form for odd p . For group 2 coefficient, one column or row of cells will be sufficient to construct the complete coefficient. Other columns or rows will be identical for a particular value of p and the representation for different values of p can be derived by appropriate circular shift. When the index is changed by $M = N/2 = 3$, the 2×2 DFT coefficients to be considered will be from the other side of the cell depending on the change. For eg. $Y_{k,0}$ will depend on the coefficients on the left side of the cells where as $Y_{k,3}$ depends on that from right side of the cells. This movement of symbols can be represented by flip horizontal/vertical operations. Also many coefficients in this group can be derived from permutation, over p , of another $Y_{k_1,k_2}^{(p)}$. Thus the representation as well as the computation can be simplified using this property.

In group 3 also, the cells are related. The fig. 3.3 shows that in this group, for a particular (k_1, k_2) , the cells in a row are different, but different rows can be obtained from one row by circularly shifting it. For eg., in $Y_{1,1}^{(0)}$ the cells $C_{0,0} = MP$, $C_{0,1} = DPB$ and $C_{0,2} = MPD$. The other rows are also with the same symbol but in a different order depending on the index as in fig. 3.3. Similarly, the representation for other values of p are also shifted versions of that of $p=0$. Also a number of coefficients in the group can be derived using

permutations of $Y_{k_1, k_2}^{(p)}$, $0 \leq p \leq M-1$. For example, $Y_{5,5}^{(0)} = Y_{1,1}^{(0)}$, $Y_{5,5}^{(1)} = -Y_{1,1}^{(2)}$, $Y_{5,5}^{(2)} = Y_{1,1}^{(1)}$. The number of coefficients that can be derived from the permutations of $Y_{k_1, k_2}^{(p)}$ will increase with N . The indices for which the coefficients can be derived from the permutation of one another are shown in Table B.1 of Appendix B. Also, there are other coefficients in which the basic symbols are closely related to one another. For eg., in $Y_{2,2}^{(0)}$ the cells $C_{0,0} = MP$, $C_{0,1} = DPA = -DPB$ and $C_{0,2} = MPC = -MPD$ as compared to $Y_{1,1}^{(0)}$. Thus there is a closely related subgroup of coefficients with indices $[(1,1), (5,5)], [(2,2), (4,4)], [(4,1), (2,5)], [(5,2), (1,4)]$ and another with indices $[(5,1), (1,5)], [(4,2), (2,4)], [(2,1), (4,5)], [(1,2), (5,4)]$, in which indices in square brackets represents the coefficients related by permutation over p . Also, the other rows for $p=0$ are the same as that of the first row but a circularly shifted version depending on the index. The representation for other values of p can be derived by appropriate shift in the rows.

The position of different symbols show a close relation with (k_1, k_2) , p & N . The fig. 3.3c shows examples of the positional relation, present in group 3, in terms of the mnemonics of the symbols when the index is changing by M .

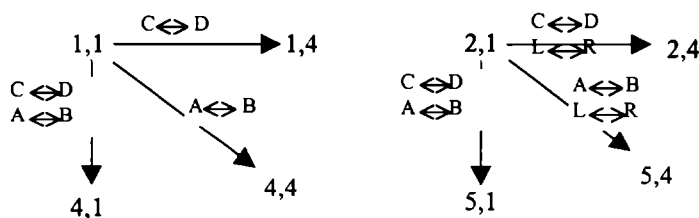


Fig. 3.3c Positional relation in group 3

3.3.2.3 $N=8$

The fig. 3.4 shows the pictorial representation of the DFT coefficients for $N=8$. The coefficients in one group are placed together while permutation related coefficients are placed adjacent to each other. The figure shows that the pictorial representation of a number of coefficients can be derived from that of one coefficient. The group 1 coefficients can be derived from one cell, since all the cells are identical for one coefficient and the others in the group are related by the operation of flipping with respect to an axis. That is, $Y_{0,M}$ can be obtained from $Y_{0,0}$ by flip horizontal operation, $Y_{M,0}$ by flipping vertically and $Y_{M,M}$ by rotate right and flip horizontal operations. Thus, in this case also the group 1 coefficients ($Y_{0,0}$, $Y_{M,0}$, $Y_{0,M}$, $Y_{M,M}$) have the same property as that of $N=4$ & 6. Similar to that of $N=4$, the

group 2 & 3 coefficients have different symbol combination for even p and odd p values.

The group 2 coefficients show a positional relation as that of N=4. That is, the coefficients in column 0, other than that in group 1, will depend on the two coefficients on the left of the cells (VLP, VLN for even p & VLPA, VLPB for odd p), the coefficients in column 4 or $Y_{k,4}$ will depend on the two coefficients on the right of the cells (VRP, VRN for even p & VRPA, VRPB for odd p), the coefficients in row 0 will depend on the two coefficients on the top position of the cells (HAP, HAN for even p & HAPL, HAPR for odd p) and that in row 4 will depend on the two coefficients on the lower position of the cells (HBP, HBN for even p & HBPL, HBPR for odd p). In this case

$$Y_{1,0}^{(0)} = Y_{7,0}^{(0)} = Y_{3,0}^{(0)} = Y_{5,0}^{(0)}, Y_{1,0}^{(2)} = -Y_{7,0}^{(2)} = -Y_{3,0}^{(2)} = Y_{5,0}^{(2)}, Y_{1,0}^{(1)} = -Y_{7,0}^{(3)} = Y_{3,0}^{(3)} = -Y_{5,0}^{(3)}, Y_{1,0}^{(3)} = -Y_{7,0}^{(1)} = Y_{3,0}^{(1)} = -Y_{5,0}^{(1)},$$

$$Y_{2,0}^{(0)} = Y_{6,0}^{(0)}, Y_{2,0}^{(2)} = -Y_{6,0}^{(2)} \text{ and } Y_{2,0}^{(1)} = Y_{2,0}^{(3)} = Y_{6,0}^{(1)} = Y_{6,0}^{(3)} = 0$$

Similar relations exist for $Y_{k,M}^{(p)}, Y_{0,k}^{(p)}, Y_{M,k}^{(p)}$ due to the positional relation and can be derived from one another using the flip operation as in the previous cases.

The group 3 coefficients also show a specific pattern depending on the position of the frequency index. That is, the coefficients on the diagonal position ($Y_{1,1}, Y_{3,3}, Y_{5,5}, Y_{7,7}$) depend on (MP, MN, MPD, MPC for even p & DPA, DPB for odd p), the coefficients ($Y_{5,1}, Y_{7,3}, Y_{1,5}, Y_{3,7}$) depend on (MP, MN, MPD, MPC for even p & CPA, CPB for odd p), the coefficients on the cross-diagonal position ($Y_{7,1}, Y_{5,3}, Y_{3,5}, Y_{1,7}$) depend on (DP, DN for even p & MPL, MPR, MPA, MPB for odd p), the coefficients ($Y_{3,1}, Y_{1,3}, Y_{7,5}, Y_{5,7}$) depend on (CP, CN for even p & MPL, MPR, MPA, MPB for odd p), ($Y_{2,2}$ & $Y_{6,6}$) depend on (CP, CN for p=0, DPA, DPB for p=2 & blank for odd p), ($Y_{2,6}, Y_{6,2}$) depend on (DP, DN for p=0, CPA, CPB for p=2 & blank for odd p). Thus, when N=8, the pictorial representation corresponding to even p will use one set of primitive symbols and that for odd p will use another set of primitive symbols in groups 2 & 3. The coefficients with one set of primitive symbols, form a permutation group as shown in the following examples.

$$Y_{1,1}^{(0)} = Y_{7,7}^{(0)} = Y_{3,3}^{(0)} = Y_{5,5}^{(0)}, Y_{1,1}^{(2)} = -Y_{7,7}^{(2)} = -Y_{3,3}^{(2)} = Y_{5,5}^{(2)}, Y_{1,1}^{(1)} = -Y_{7,7}^{(3)} = Y_{3,3}^{(3)} = -Y_{5,5}^{(3)}, Y_{1,1}^{(3)} = -Y_{7,7}^{(1)} = Y_{3,3}^{(1)} = -Y_{5,5}^{(1)},$$

$$Y_{2,2}^{(0)} = Y_{6,6}^{(0)}, Y_{2,2}^{(2)} = -Y_{6,6}^{(2)} \text{ and } Y_{2,2}^{(1)} = Y_{2,2}^{(3)} = Y_{6,6}^{(1)} = Y_{6,6}^{(3)} = 0$$

$$Y_{5,1}^{(0)} = Y_{3,7}^{(0)} = Y_{7,3}^{(0)} = Y_{1,5}^{(0)}, Y_{5,1}^{(2)} = -Y_{3,7}^{(2)} = -Y_{7,3}^{(2)} = Y_{1,5}^{(2)}, Y_{5,1}^{(1)} = -Y_{3,7}^{(3)} = Y_{7,3}^{(3)} = -Y_{1,5}^{(3)}, Y_{5,1}^{(3)} = -Y_{3,7}^{(1)} = Y_{7,3}^{(1)} = -Y_{1,5}^{(1)},$$

$$Y_{2,6}^{(0)} = Y_{6,2}^{(0)}, Y_{2,6}^{(2)} = -Y_{6,2}^{(2)} \text{ and } Y_{2,6}^{(1)} = Y_{2,6}^{(3)} = Y_{6,2}^{(1)} = Y_{6,2}^{(3)} = 0$$

The position of different symbols show a close relation with (k_1, k_2) , p & N . The fig. 3.4a shows examples of the positional relation in terms of the mnemonics of the symbols when the index is changing by M .

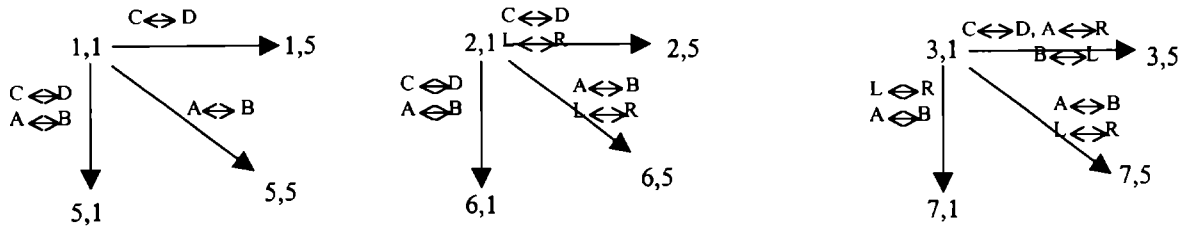


Fig. 3.4a Positional relation in group 3

3.3.2.4 $N=10$

The DFT coefficients are classified into three groups, as in previous cases, depending on the way the primitive symbols are present in them. Group 1 corresponds to the four real coefficients with indices $(0,0)$, $(0,M)$, $(M,0)$, (M,M) having $Y_{k_1, k_2}^{(p)}$ for $p=0$ only, where $M=5$. Group 2 is with $4(N-2) = 32$ coefficients having the indices $(k,0)$, (k, M) , $(0,k)$, (M,k) where $k=1,2,\dots, N-1$ and $k \neq M$. Group 3 consists of the remaining $N^2 - 4(N-1) = 64$ coefficients.

In Fig. 3.5, the first bin shows the pictorial representation of group 1 coefficients in one figure. The group 1 coefficients depend on one 2×2 DFT coefficient in each cell and they show a positional relation with the frequency index as in the previous cases. For example, $Y_{0,0}$ will depend on the 2×2 DFT coefficient at the top left corners of each cell (with mnemonic LA) represented by small rectangles with black color whereas $Y_{0,5}$ will depend on that at the top right position (RA), $Y_{5,0}$ depends on 2×2 DFT coefficients at the left below position (LB) and $Y_{5,5}$ depends on that at the right below (RB) in each cell. In this group, all the cells are identical for a coefficient and hence only one cell will be sufficient to construct its visual representation as in other cases. The group 1 coefficients for any even N will have similar representation with $M \times M$ identical cells as in the case of $N = 4, 6$ & 8 .

The group 2 consists of the coefficients $Y_{k,0}$, $Y_{k,M}$, $Y_{0,k}$, $Y_{M,k}$ where $k=1,2,3,4,6,7,8,9$. These coefficients will have $Y_{k_1, k_2}^{(p)}$ for $p = 0,1,\dots,M-1$. The Fig. 3.5 shows the simplified form of the pictorial representation of all the coefficients in column zero, column M , row zero and row M of group 2 coefficients using different colors. The figure shows that for even p , $Y_{k,0}^{(p)}$ depends on VLP and VLPA (ie., the two 2×2 DFT coefficients at the left of the cells) and its

sign reversed form for odd p (ie, $VLN=-VLP$ and $VLPB=-VLPA$). $Y_{k,5}^{(p)}$ depends on VRP and VRPA (the DFT coefficients at the right of the cells) for even p and its sign reversed form for odd values of p . Similarly, $Y_{0,k}^{(p)}$ & $Y_{5,k}^{(p)}$ will depend on HAP, HAPL and HBP, HBPL respectively for even p and corresponding sign reversed form for odd p . Hence, one column or row of cells will be sufficient to construct the complete coefficient. Other columns or rows will be identical for a particular value of p and the representation for different values of p can be derived by appropriate circular shift. When the index is changed by $M = N/2 = 5$, the 2×2 DFT coefficients to be considered will be from the other side of the cell depending on the change. For eg. $Y_{k,0}$ will depend on the coefficients on the left side of the cells where as $Y_{k,5}$ depends on that from right side of the cells. Thus the pattern of symbol organization for $N=10$ is the same as that of $N=6$ except the change in its position which is closely related to the value of N .

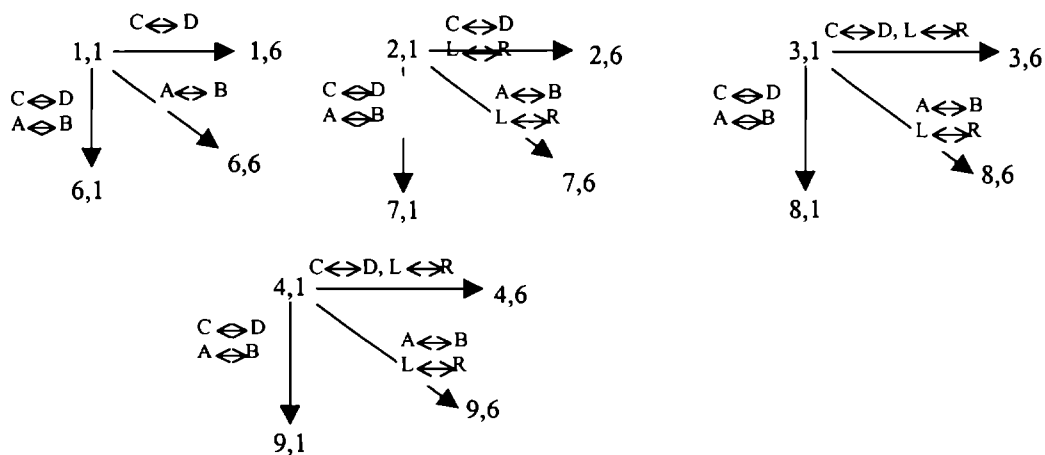


Fig. 3.5a Positional relation in group 3

The pictorial representation of the group 3 coefficients for $N = 10$ show a similar pattern as that of $N = 6$. The values of p in $Y_{k1,k2}^{(p)}$ are $p = 0, 1, 2, \dots, M-1=4$. These coefficients are represented, fig. 3.5, in one bin for all the values of p corresponding to each $(k1, k2)$ so that the comparison of the organization of symbols with variation in p and $(k1, k2)$ is easy. Since there are 64 coefficients in this group and each coefficient takes five different values of p , the space required for the representation of the complete set will be high, but they are closely related to a basic set, only the basic set are shown in the fig. 3.5. The fig. 3.5a shows examples of the positional relation in terms of the mnemonics of the symbols when the index changes by M .

3.3.3 Results of the analysis

The above analysis shows that the pictorial representation of the coefficients has useful patterns for deriving other computational schemes that are easy and efficient in processing 2-D signals. It shows a regular structure as N changes when $((N))_4 = 2$ and a slightly different regular structure for $((N))_4 = 0$ due to the difference in number of symmetries in the twiddle factors for the two cases. In both the cases the coefficients are grouped into three based on the similarities in the symbols used and the operation over it. It is found that group 1 consists of 4 real coefficients, group 2 has $4(N-2)$ coefficients and group 3 contains $N^2-4(N-1)$ coefficients. The group 1 have real coefficients with pictorial representation only for $p=0$ and have identical cells in both the cases. When $((N))_4 = 0$, the pictorial representation corresponding to even p will use one set of primitive symbols and that for odd p will use another set of primitive symbols in groups 2 & 3. Whereas, when $((N))_4 = 2$, the pictorial representation corresponding different p values will be with the same set of primitive symbols in groups 2 & 3.

The group 2 consists of the coefficients $Y_{k,0}$, $Y_{k,M}$, $Y_{0,k}$, $Y_{M,k}$ where $k=1, 2, 3, \dots, N-1$ and $k \neq M$. For group 2 coefficients when $((N))_4 = 2$, the symbols in one row/column of cells is sufficient to represent the complete coefficient. For a 'p' the other rows/columns are identical. The representation for different values of p can be derived by circularly shifting. Also a number of coefficients in the group can be derived using permutations of $Y_{k_1,k_2}^{(p)}$, $0 \leq p \leq M-1$. The number of coefficients that can be derived from the permutations of $Y_{k_1,k_2}^{(p)}$ will increase with N . Fig. 3.7 can be employed for the representation of the coefficients in different columns / rows.

In group 3 too, the cells are related. When $((N))_4 = 2$ for a particular (k_1, k_2) , the cells in a row are different, but different rows can be obtained from one row by circularly shifting it. The other rows also have the same symbols but in a different order depending on the index. Similarly, the representation for other values of p are also shifted versions of that of $p = 0$. Also a number of coefficients in the group can be derived using permutations of $Y_{k_1,k_2}^{(p)}$, $0 \leq p \leq M-1$. The number of coefficients that can be derived from the permutations of $Y_{k_1,k_2}^{(p)}$ will increase with N . The indices for which the coefficients can be derived from the permutation of one another are shown in Table B.1, B.2, B.3 of Appendix B for $N=6,10,14$ respectively.

In general, a graphical representation of the relationship between different coefficients in one group can be expressed as shown below in fig. 3.7. This is applicable when N is an even number.

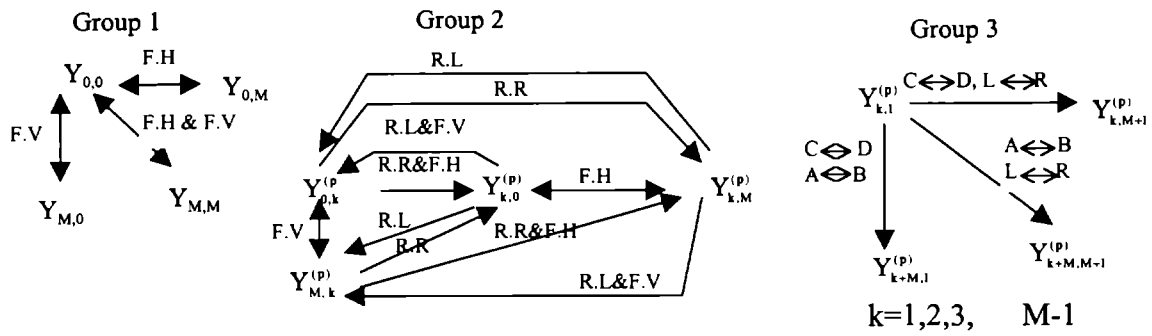


Fig. 3.7 Positional relation within a group

Thus, a semantic grammar can be derived to evolve a coefficient from another coefficient.

Comparing the representations for different values of N, their inter relationships can easily be derived. Some of the properties are the following.

1. For any even N, the group 1 coefficients (with index (0,0), (0,M), (M,0), (M,M)) will have the representation for p=0 only as given in theorem 2.
2. If $((N))_4 = 2$, all the coefficients other than in group 1 will have the representation for all the values of p = 0, 1, ..., M-1.
3. If $((N))_4 = 0$, then
 - a) The pictorial representation exists only for p = 0 and N/4, if k1 & k2 are 0, M, N/4 or 3N/4, and not in group 1 [i.e.; (0,N/4), (N/4,0), (N/4,N/4), (N/4,M), (M, N/4), (3N/4,0), (0,3N/4), (3N/4,M), (N/4,3N/4), (3N/4,N/4), (M, 3N/4), (3N/4,3N/4)].
 - b) The pictorial representation corresponding to even indexed (both k1 & k2 are even) coefficients does not have the representation for odd p.
 - c) If k1 or k2 is odd and not in (1) and (3.a) mentioned above, the pictorial representation for all p=0,1,..., M-1 exist.
4. The pictorial representation of the complete set of coefficients of N = 4 are available in that of N = 8. In general, the pictorial representations of the complete set of coefficients of all even factors of N are present in that of N as given in lemma 3.
5. If $N1 = k \cdot N$ and $k1' = k \cdot k1$, $k2' = k \cdot k2$ and $p' = k \cdot p$, then the pictorial representation of $Y_{k1',k2'}^{(p')}$ can be derived from that of $Y_{k1,k2}^{(p)}$ by repeating k times in rows and columns. For

eg. $Y_{2,2}^{(0)}$ of $N=8$ can be derived from $Y_{1,1}^{(0)}$ of $N = 4$ by repeating it 2 times in rows and columns.

By studying the properties of the pictorial representation, an indirect method for the pictorial representation is derived in the following section. A few theorems are also presented at the end of this chapter. Also this pictorial representation forms the basis for the design of a parallel distributed scheme of computation discussed in the next chapter.

3.3.4 Indirect method

In the direct method of pictorial representation, for each (k_1, k_2) pair and M values of p , it is required to compute $z = ((n_1.k_1+n_2.k_2))_N$ for all values of (n_1, n_2) and check whether it is p or $p + M$, then using the symbol table to select the appropriate symbols for each 2×2 cell. This can be simplified by utilizing the properties of the pictorial representation.

As explained earlier, it is found that the primitive symbols to be used are dependent on the index value (k_1, k_2) . Thus we can devise an algorithm to construct the pictorial representation without computing z for all (n_1, n_2) and searching the symbol table. The primitive symbols can be directly selected based on the index or obtained using direct method for a few cells and rearranging them to get the symbols for other cells.

3.3.4.1 Construction of the pictorial representation for $((N))_4 = 2$

The indices (k_1, k_2) are grouped into 3 as in section 3.3.

As explained earlier, the group 1 coefficients will have identical cells for a (k_1, k_2) and the symbol is closely related to the position of k_1, k_2 . The group 2 coefficients have two sets of symbols based on the position of the index. If the frequency index (k_1, k_2) is from a column then the 2×2 DFT coefficients will also be from the column (represented mnemonically by V (vertical)) and if (k_1, k_2) is from a row then the mnemonics will have H (Horizontal) as in previous subsections. The symbols that are present in $Y_{k,0}^{(p)}$ will be corresponding to the mnemonics VLP and VLPA if p is even and $VLN = -VLP$ and $VLPB = -VLPA$ when p is odd. Also in this case one row of cells will have the same symbols. The row number depends on the value of k and p . The row index 'n' of cells with VLP/VLN can be obtained from the relation $((2.n.k))_M = p$ and the cells VLPA/VLPB will be shifted from that of VLP/VLN by $(M-1)/2$ row of cells. The symbols that are in $Y_{0,k}^{(p)}$ will be HAP and HAPL

for even p and $HAN = -HAP$ & $HAPR = -HAPL$ for odd p . The symbols for $Y_{k,M}^{(p)}$ are VRP & VRPA for even p and VRN & VRPB for odd p . Similarly, the symbols for $Y_{M,k}^{(p)}$ are HBP & HBPL or HBN & HBPR depending on p .

The group 3 coefficients will have different sets of symbols in one row/column. It is found that the symbols in each row, for any value of p , are the repetition of symbols in the first row of $p=0$ with a proper circular shift. Thus, an algorithm can be derived for constructing the representation using the direct method for the first row of cells for $p=0$. Then applying proper circular shift depending on k_1 , k_2 & p , the complete picture can be derived. Using a similar approach as in group II, the order of circular shift to be given can be found out as given in the algorithm. The algorithm for construction of the pictorial representation when $((N))_4 = 2$ is as given below. This method is more suitable if a few selected DFT coefficients are to be represented pictorially.

3.3.4.2 *The Algorithm*

Choose values of N , (k_1, k_2) where $((N))_4 = 2$

Check whether (k_1, k_2) is in group G_1 , G_2 or G_3 and branch accordingly to derive the mnemonics corresponding to the $M \times M$ cells. Draw the $M \times M$ grid and place the symbols corresponding to the mnemonic 'Mnem' in each cell $C_{i,j}$. The remaining cells will be blank.

For G_1

- 1) If $k_1 = k_2 = 0$, Mnem = LA
- 2) If $k_1 = 0$ & $k_2 = M$, Mnem = RA
- 3) If $k_1 = M$ & $k_2 = 0$, Mnem = LB
- 4) If $k_1 = M$ & $k_2 = M$, Mnem = RB

For $i = 0, M-1$

For $j = 0, M-1$

$C_{i,j} = \text{Mnem}$

For G_2

- 1) If $k_2 = 0$, Mnem1 = VLP & Mnem2 = VLPA, $k=k_1$
- 2) If $k_2 = M$, Mnem1 = VRP & Mnem2 = VRPA, $k=k_1$
- 3) If $k_1 = 0$, Mnem1 = HAP & Mnem2 = HAPL, $k=k_2$
- 4) If $k_2 = M$, Mnem1 = HBP & Mnem2 = HBPL, $k=k_2$

For $p = 0, M-1$

Find n such that $((2nk))_M = p$

$i1 = n, i2 = ((i1 + (M-1)/2))_M$

For $j = 0, M-1$

If $k1 = 0$ or M

$$C_{i1,j} = (-1)^p M_{nem1}$$

$$C_{i2,j} = (-1)^p M_{nem2}$$

Else if $k2 = 0$ or M

$$C_{j,i1} = (-1)^p M_{nem1}$$

$$C_{j,i2} = (-1)^p M_{nem2}$$

For G3

1) Find the cells $C_{0,j}$ for the first row of cells using the direct method.

2) For $p = 0, M-1$

Find n such that $((2n.k2))_M = p$

3) For $i = 0, M-1$

For $j = 0, M-1$

$i1 = ((k2.i))_M, j1 = ((n-k1.i))_M$

$$C_{i1,((j+j1))_M} = C_{0,j}$$

The algorithm is tested and the pictures are compared with that of direct method.

3.4 THEOREMS & DEFINITIONS

Definition

The 2-D DFT coefficient corresponding to any $N \times N$ data can be expressed in terms of the real coefficient $Y_{k1,k2}^{(p)}$ as

$$Y_{k1,k2} = \sum_{p=0}^{N/2-1} Y_{k1,k2}^{(p)} W_N^p, \quad 0 \leq k1, k2 \leq N-1$$

where $Y_{k1,k2}^{(p)} = \sum_{\forall (n1,n2): z=p} A_{n1,n2} - \sum_{\forall (n1,n2): z=p+N/2} A_{n1,n2}$, $z = ((n1.k1 + n2.k2))_N$

Thus 2-D DFT computation through this can be interpreted as follows. The 2-D data in the spatial domain with index $(n1, n2)$ is mapped to the frequency domain with index $(k1, k2)$ in two steps. First the data is mapped on to each of the vectors formed from the twiddle factors W_N^p with the mapping function as $((n1 \cdot k1 + n2 \cdot k2))_N = p$. Since $W_N^{p+N/2} = -W_N^p$, only $N/2$ vectors are considered and the remaining $N/2$ will be folded over with a sign reversal. The algebraic sum of the data mapped over to a W_N^p will be the real value $Y_{k1,k2}^{(p)}$. Then the weighted sum of these vectors W_N^p and the corresponding $Y_{k1,k2}^{(p)}$ gives the DFT coefficient $Y_{k1,k2}$.

Theorem 1

Given a set of primitive symbols corresponding to 2×2 DFT, it is always possible to construct the pictorial representation for any $N \times N$ DFT if N is an even number.

Proof

The method explained in the construction of pictorial representations in section 3.4.1.

Lemma 1

The total number of primitive symbols, corresponding to 2×2 DFT, used for construction of pictorial representation is 37. These primitives can be defined based on the position of the DFT coefficient in the 2×2 cell.

1. Number of primitives with one DFT coefficient = 4

2. Number of combinations of two DFT coefficients out of 4 = $4C_2 = 6$

Number of combinations of two DFT coefficients out of 4 with \pm sign = $4 \cdot 6 = 24$.

3. Number of combinations with all 4 DFT coefficients = $4 \cdot 2 = 8$

4. Number of combinations with no DFT coefficient = 1

\therefore Total number of primitive symbols = $4 + 24 + 8 + 1 = 37$.

Lemma 2

The pictorial representation can also be derived in terms of primitive symbols obtained from 2×2 data in the same way as that of DFT. The number of primitive symbols will be the same with the same set of positional definition as in the case of 2×2 DFT.

Theorem 2

For any even N, the coefficients with index (0,0), (0,M), (M,0) & (M,M) will have the pictorial representation for p=0 only.

Proof

Let (n1, n2) and (k1, k2) be the time index and frequency index respectively and the order be N. As shown in definition, $Y_{k1,k2}^{(p)} = \sum_{\forall(n1,n2)|z=p} A_{n1,n2} - \sum_{\forall(n1,n2)|z=p+N/2} A_{n1,n2}$, $0 \leq p \leq M-1$, where $z = ((n1 \cdot k1 + n2 \cdot k2))_N$. When $k1 = k2 = 0$, for all values of (n1, n2) the value of z is zero and hence p=0 only. Similarly when $k1=0$ or M and $k2 = 0$ or M, for all values of (n1, n2), $z = 0$ or M. But as per the definition, $z = 0$ & $z = M$ correspond to $p=0$. Thus proved.

Theorem 3

The pictorial representation of the DFT coefficient corresponding to the index (k .k1, k .k2) can be derived from that of (k1, k2) by permuting over p by ((k .p))_M with a sign reversal if ((k .p))_N ≥ M when k is an odd integer.

Proof

Let (n1, n2) and (k1, k2) be the time index and frequency index respectively and the order be N. As shown in definition, $Y_{k1,k2}^{(p)} = \sum_{\forall(n1,n2)|z=p} A_{n1,n2} - \sum_{\forall(n1,n2)|z=p+N/2} A_{n1,n2}$, $0 \leq p \leq M-1$, where $z = ((n1 \cdot k1 + n2 \cdot k2))_N$. Let $k1' = k \cdot k1$, $k2' = k \cdot k2$ then $((n1 \cdot k1' + n2 \cdot k2'))_N = ((k(n1 \cdot k1 + n2 \cdot k2)))_N = ((k((n1 \cdot k1 + n2 \cdot k2)))_N)_N = ((k \cdot p))_N$ or $((k(p + N/2)))_N$. But $((k(p + N/2)))_N = ((k \cdot p))_N + N/2$, if k is odd. If $((k \cdot p))_N \geq M$ this will introduce a minus sign. Thus the permutation will be by ((k .p))_M. Thus proving the theorem.

Theorem 4

The pictorial representation of $Y_{k1,k2}^{(p)}$ for order N is contained in $Y_{k.k1,k.k2}^{(k.p)}$ of order N1 = k.N where k is a positive integer.

Proof

Let (n1, n2) and (k1, k2) be the time index and frequency index respectively when the order is N and (n1', n2') and (k1', k2') be the time index and frequency index respectively when N' = k N. As shown in definition, $Y_{k1,k2}^{(p)} = \sum_{\forall(n1,n2)|z=p} A_{n1,n2} - \sum_{\forall(n1,n2)|z=p+N/2} A_{n1,n2}$, $0 \leq p \leq M-1$, where $z = ((n1 \cdot k1 + n2 \cdot k2))_N$. Let $((n1 \cdot k1 + n2 \cdot k2))_N = p$ and $((n1' \cdot k1' + n2' \cdot k2'))_N$.

= p' where $k1' = k \cdot k1$, $k2' = k \cdot k2$ and $p' = k \cdot p$. Then $((n1' \cdot k \cdot k1 + n2' \cdot k \cdot k2))_{k,N} = ((k(n1' \cdot k1 + n2' \cdot k2)))_{k,N} = k \cdot p$ for $n1' = n1$ and $n2' = n2$. Thus proved.

Lemma 3

The pictorial representation of the DFT coefficients for an order N will contain the representation for all N1 such that N1 is an even factor of N.

eg. $((N))_4=0$	$((N))_4=2$
N=8 → N1 = 4	N=6
N=12 → N1 = 4,6	N=10
N=16 → N1 = 4,8	N=14
N=20 → N1 = 4,10	N=18 → N1 = 6
N=24 → N1 = 4,6,8,12	N=22
N=28 → N1 = 4,14	N=26
N=32 → N1 = 4,8,16	N=30 → N1 = 6,10
N=36 → N1 = 4,6,12,18	N=34
N=40 → N1 = 4,8,10,20	N=38
N=44 → N1 = 4,22	N=42 → N1 = 6,14
N=48 → N1 = 4,6,8,12,16,24	N=46
N=52 → N1 = 4,26	N=50 → N1 = 10
N=56 → N1 = 4,8,14,28	N=54 → N1 = 6,18
N=60 → N1 = 4,6,10,12,20,30	N=58
N=64 → N1 = 4,8,16,32	N=62
N=68 → N1 = 4,34	N=66 → N1 = 6,22
N=72 → N1 = 4,6,8,12,18,36	N=70 → N1 = 10,14
N=76 → N1 = 4,38	N=74
N=80 → N1 = 4,8,10,16,20,40	N=78
N=84 → N1 = 4,6,12,14,28,42	N=82
N=88 → N1 = 4,8,22,44	N=86
N=92 → N1 = 4,46	N=90 → N1 = 6,10
N=96 → N1 = 4,6,8,12,16,24,48	N=94
N=100 → N1 = 4,10,20,50	N=98 → N1 = 14

3.5 CONCLUSION

A new definition is suggested for the 2-D DFT relation and using this a pictorial representation of the DFT coefficients are formulated in this chapter. The representations are analyzed and the pictorial representations are simplified using the properties. The DFT computation and analysis of 2-D signals can be simplified using these pictorial representations. Also, this representation is the basis for the neural network model derived in chapter 4.

Studies of the brain show that a hierarchy of cells with increasing complex response characteristics exists in the visual cortex and other parts of the brain. It is not difficult to extrapolate this idea of hierarchy into one where further data abstraction takes place at successively higher levels. The neocognitron design adopts this hierarchical structure in a layered architecture. It is an example of how neurobiological results can be used to develop a new network architecture [12].

Speed of computation, important in real-time applications, can be improved by reducing the number of multiplications and/or by parallel processing. The DFT is a useful analytical tool in stationary signals and some quasi-stationary signals. There are a number of techniques used for reduction of computation. FFT is the most popular algorithm to implement DFT and is highly efficient in the case of 1-D signals. The N-point sequence requires N^2 complex multiplications and $N(N-1)$ complex additions in the case of direct DFT, whereas it needs only $N \log_2 N$ complex operations in the case of FFT when N is a power of 2 [5]. The reduction in computation is more predominant when N is large.

For 2-D signals, the FFT method is highly time consuming since it requires a large number of complex multiplications [1] and literature shows that it is not in much use in 2-D signal processing applications [129]. Parallel computation increases speed and may render real time applications possible.

The evaluation of DFT coefficients by employing pictorial representation described in chapter 3 shows the presence of specific patterns that can easily be exploited in implementing a parallel distributed scheme. From the analysis of the pictorial representation for the 2-D DFT computation and comparing it with the processing in neocognitron model, it is found that a hierarchical model can be developed. Consequently the speed of computation improves drastically. This approach will be extremely useful in the analysis and processing of 2-D signals that exhibit repetition in blocks. In this thesis, a parallel distributed model to implement $N \times N$ - point DFT for $(N)_4 = 2$ is proposed. This model can be extended to any order N .

4.1 REASONS FOR CHOOSING THE PRESENT ARCHITECTURE

Neural Network architectures such as backpropagation network have general applicability. We can use a single network type in many different applications by changing the network's size, parameters, and training sets [12]. But this model takes a long time for learning, and it should relearn if more examples are added to the training set and its convergence to global minima is difficult in many cases. These are serious drawbacks of the model [17]. Adaptive Resonance Theory (ART) model, also a general model, solves the drawbacks of backpropagation network. This model has real time learning capability with the number of different classes that can be learned limited only by the number of neurons provided in the network [130]. In contrast to this, there are special purpose neural network architectures such as the neocognitron model used for recognition of handwritten characters [12]. The neocognitron model uses a hierarchical structure in which there are a number of layers operating in a hierarchy and in each layer there are a number of planes operating in parallel. Each plane has a group of cells, the basic element in the structure, operating in parallel. But these models are popular for classification problems.

The advantage of a hierarchical neural network structure is that the processing elements of each layer have to concern themselves only with the processing of a limited

amount of information. The total global situation is then pieced together as one ascends from one hierarchical layer to the next. This approach has two major benefits.

1) The processing elements at each layer need only concern themselves with the relatively simple situation described by their input signals which make up only a tiny fraction of the total input coming to the layer from the previous layer.

2) Hierarchical networks can typically function with a spectacularly smaller number of processing elements than would be required by a network that would deal with the entire input as a single unit.

By processing input information one step at a time, hierarchical networks can provide capabilities that would be otherwise impossible. Naturally, hierarchical networks are appropriate in those situations where the inputs to the network have low level, intermediate level and high level structures that can be consistently related to one another as in images of hand written characters etc. Random data, for example, does not have such a structure.

The neocognitron model and its modifications are the only well studied neural network in this category. The pictorial representation of the DFT computation described in chapter 3 shows a good structure for the design of a hierarchical architecture for the computational application. Thus in this chapter, a hierarchical architecture for the computation of the 2-D DFT is proposed adopting concepts from the neocognitron model, ART model and the basics of parallel distributed processing as given in [17].

Also in all these models the learning is through modification of connection weights and the output of each neuron will be obtained through multiplication with weights. But a network can also learn through addition or deletion of connections [17] and is used in the proposed model. Since the main aim of this computational model is to increase the speed of computation, it uses one step learning to reduce the learning time.

4.2 BASICS

4.2.1 The derivation of the structure

The regular pattern present in the pictorial representation of the DFT coefficients can be used to derive a hierarchical computation scheme. Also, this shows a lot of repetitions in the computations that can be avoided by splitting into simple steps. Thus splitting the computation into different steps can form a hierarchy. The primitive symbols are computed

first. In the second step, different combinations of the primitive symbols are formed to represent one row/column and their circularly shifted versions. The third step is to combine the various row/column of symbols so that a set of the coefficients $Y_{k_1, k_2}^{(p)}$ can be obtained for a selected set of (k_1, k_2) . The last step is to combine $Y_{k_1, k_2}^{(p)}$ in proper order, scaled by the twiddle factors, to obtain the complete set of Y_{k_1, k_2} . Similar operations are grouped together to enable parallel distributed computation, as in the neocognitron model [23-29], so that the speed of computation can be improved considerably. Also, a maximum order for the input & output can be fixed using the principles of the ART model. Any order upto the set maximum can be implemented by choosing appropriate connections to make the network learn. This brings the requirement of learning by addition or deletion of connections which is equivalent to having the connection weight 0 or 1. Since speed of computation is more important in computational applications, care is taken to reduce the learning time as well as the processing time. The structure is derived in such a way that it should be possible to arrive at a compromise between the speed and complexity/cost by choosing sequential/parallel operation at different steps.

4.2.2 Modifying patterns of connectivity as a function of experience

Changing the processing or knowledge structure in a parallel distributed processing model involves modifying the patterns of interconnectivity. In principle this can involve three kinds of modifications [17]:

1. The development of new connections.
2. The loss of existing connections.
3. The modification of the strengths of connections that already exist.

D. E Rumelhart, G. E. Hinton and J. L. McClelland [17] say that very little work has been done on (1) & (2) above. However, (1) & (2) can be considered as a special case of (3). Whenever we change the strength of connection away from zero to some positive or negative value, it has the same effect as growing a new connection. Whenever we change the strength of a connection to zero, it has the same effect as losing an existing connection. Thus in most of the neural network models learning rules are based on modifying strengths of connections.

But, the model developed in this work being a computational model, learning is through addition or deletion of connections and the connection weights will be +1, -1 or 0. Also it is proposed to use single step learning so that the learning time will not be a problem.

4.2.3 Structure of the network

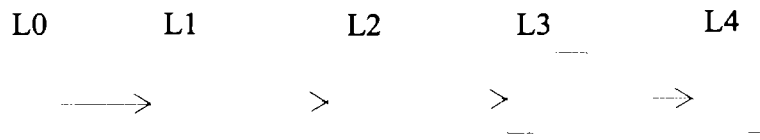


Fig. 4.1 Hierarchical structure of the model

The network model is designed using a hierarchical structure in a layered architecture as in fig. 4.1. It consists of a cascade connection of a number of modular structures preceded by an input layer L0. L0 is a 2-D array of input data, $x(i, j) \quad 0 \leq i, j \leq N-1$. It is divided into 2×2 matrices and the operations are in terms of these matrices.

The processing elements or cells of the network model are organized into modules called layers. There are four layers L1 to L4 other than the input layer. The cells in a layer are sorted into subgroups according to the features of their receptive fields and the operation over it. Each of these subgroups is set into a 2-D array called a 'cell-plane'. The cells in a single cell-plane have input connections of the same spatial distribution and only the positions of the input cells are shifted from cell to cell. Hence, all the cells in a single cell plane have receptive fields of the same function, but at different positions. Similar planes are categorized into one group and there are 3 groups of similar planes in each layer. Different groups in a layer may or may not have the same number of cell planes. The size of all the cell-planes in a group is the same.

4.3 DESIGN OF PARALLEL DISTRIBUTED ARCHITECTURE FOR 6x6-POINT DFT

The analysis of the pictorial representation, in chapter 3, shows that a hierarchy can be formed from the symbol organization. The DFT coefficients can be grouped into three classes, each having a common property. Each of these classes will depend on a particular organization of the primitive symbols. For example, $Y_{0,0}$ will depend on the DFT coefficients at the top left corner of each 2×2 matrix, or equivalently, the sum of all the four

data elements in each 2x2 matrix of data, for $p = 0$ only. $Y_{0,3}$ is obtained from the sum of the DFT coefficients at the right and above position (RA) in each 2x2 matrix, where RA = sum of data in the first column - sum of data in the second column of a 2x2 matrix. Similarly, the coefficients $Y_{3,0}$ & $Y_{3,3}$ depend on LB & RB respectively or the corresponding data relation as given in chapter 3. Thus, all the DFT coefficients in group 1 depend on one DFT coefficient from each 2x2 matrix.

The coefficients $Y_{k,0}$, $Y_{k,3}$, $Y_{0,k}$ & $Y_{3,k}$, $1 \leq k \leq N-1$ and $k \neq M$ where $N=6$ and $M = N/2$, form another group with common properties. The remaining coefficients are grouped into a third group. In all the three groups of coefficients, a hierarchy of computation can be derived using the structure available in the pictorial representation. As explained earlier, the pictorial representation is obtained in terms of a selected set of primitive symbols in a row/column and the different rows/columns are mere repetitions with a circular shift. Also, many coefficients will have the same combination of primitive symbols with different circular shift depending on the frequency index. Thus the DFT coefficients can be obtained in a hierarchy of four levels. In the first level the primitive symbols corresponding to all the 2x2 matrices are calculated. A few primitive symbols are chosen to form a combination representing a row / column of symbols corresponding to a selected set of $Y_{k_1,k_2}^{(p)}$ for $p = 0$ and all such combinations are computed in the second level. In the third level, proper combination of rows / columns of primitive symbols are chosen to obtain the selected set of $Y_{k_1,k_2}^{(p)}$, $0 \leq p \leq M-1$. The output of the fourth level will be the complete set of DFT coefficients, derived from the weighted sum of few $Y_{k_1,k_2}^{(p)}$ and a set of pre-computed twiddle factor values.

4.3.1 Outline of the model

The outline of the model is given in this section. The hierarchical structure of the network is illustrated in fig. 4.2. The names given in different rectangles represent the names of the cell-planes. In layer L1, all the cell-planes are of dimension $M \times M$ with $M = N/2$. In L2, the group L2G1 consists of 4 cell planes of dimension $M \times 1$, L2G2 with 6 planes of dimension $1 \times M$ and L2G3 has $2(N-2)$ planes of size $M \times M$ respectively. L3G1 consists of 6 cell planes of single cells, L3G2 and L3G3 are with 8 & $2(N-2)$ cell planes of dimension $1 \times M$. The outputs of layer L4 are the DFT coefficients that are grouped into 3. The output of

group L4G1 are the 4 real coefficients, L4G2 and L4G3 give $4(N-2)$ and $N^2 - 4(N-1)$ complex coefficients respectively. The group 2 has 8 cell-planes with $M-1$ cells representing related coefficients and group 3 has $2(N-2)$ cell planes, with $M-1$ cells each, representing the related coefficients. L4G1 has simple cells as in other layers whereas L4G2 & L4G3 have complex cells involving scalar multiplications of complex numbers.

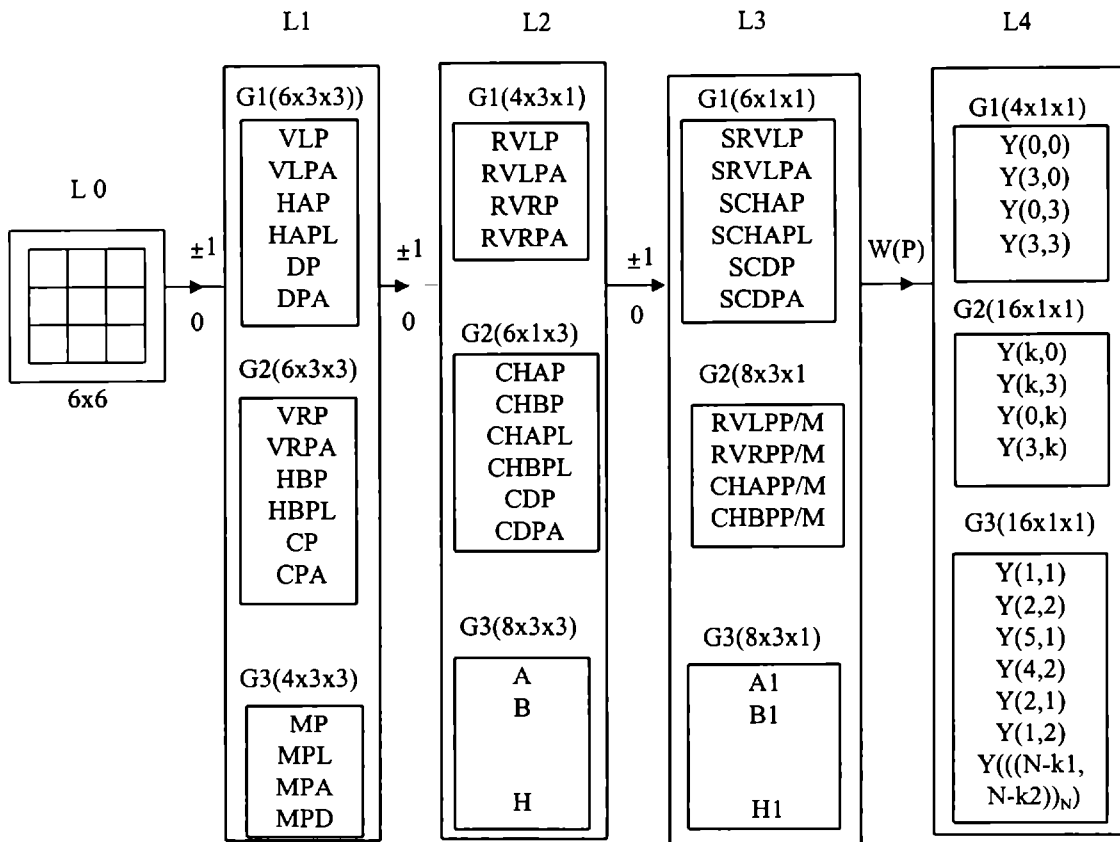


Fig. 4.2 Block Schematic of the Parallel Distributed model to implement 6x6 - point DFT

In this parallel distributed architecture, the primitive symbols chosen are VLP, VLPA, HAP, HAPL, DP & DPA forming L1G1, the second group (L1G2) with VRP, VRPA, HBP, HBPL, CP & CPA and the third group (L1G3) with MP, MPL, MPA & MPD. Corresponding to each primitive symbol, a plane is formed with the name of the plane indicating the primitive symbol. Each plane in layer 1 is a square array of $M \times M$ cells. The cells in each plane get the input data from the corresponding 2×2 input data matrix of the input layer. Also, the planes are grouped together based on the type of operation. In the first group of planes, the cell values are obtained as the sum of two data in a particular position in each 2×2 matrix. In G2, the cell values are obtained from the same set of data as that used in G1 but the operation will be subtraction. Similarly, the cell values of the planes in G3 are

obtained by copying the input from the 2x2 matrices at a particular position. Once the computation of the first layer is over, the input layer is no more required.

The second layer also consists of three groups. The planes in L2G1 are vectors of dimension M with the names indicating the input plane and the operation to be carried out. The planes are RVLP, RVLPA, RVRP & RVRPA in which, 'R' indicates that the cell values in that plane are obtained from the row sum. Hence, RVLP indicates that each cell value in this plane is obtained from the sum of the cell values of each row of the plane VLP. The planes in the second group (L2G2) are CHAP, CHBP, CHAPL, CHBPL, CDP, CDPA where 'C' represents column sum as in group 1. The planes in group 2 are also vectors of dimension M . There are 8 planes in L2G3, named as A, B, C, ..., H, with dimension $M \times M$. The cells in each plane of L2G3 depend on the cell values from a particular set of cell planes in L1 in a given order. For example, the $(0,0)^{\text{th}}$ cell of the plane A is obtained from the sum of $DP_{0,0}$, $MPL_{0,1}$ & $MPA_{0,2}$. The cell value for $(0,1)^{\text{th}}$ cell is obtained by increasing the column number by 1 in that of $(0,0)^{\text{th}}$ cell, i.e, the sum of $DP_{0,1}$, $MPL_{0,2}$ & $MPA_{0,0}$. Similar combinations from various cell planes of layer 1 give cell values in the other planes in this group.

The layer 3 also has 3 groups. The first group, L3G1, with cell planes having single cell in each plane are SRVLP, SRVLPA, SCHAP, SCDP, SCDPA where the 'S' indicates that the sum of the cell values in the respective cell planes from layer 2 are used to obtain the cell value in this group. For example, SRVLP indicates that the sum of cell values of RVLP gives the cell value in this plane.

The group 2 of layer 3 (L3G2) consists of 8 cell planes with M cells each. The cell planes are RVLPP, RVRPP, CHAPP, CHBPP, RVLPM, RVRPM, CHAPM, CHBPM where the cell values in RVLPP are obtained by selective addition of two cells from the plane RVLP and RVLPA in layer2 where as RVLPM uses subtraction of the same cell values. Thus 'P' indicates plus and 'M' indicates minus in naming these planes in this group.

The group L3G3 consists of 8 cell planes, M cells each, named as A1, B1, C1, ... H1 indicating that the inputs are from A, B, C, ... H respectively of the layer 2. The cell values in these planes are obtained by adding values of selected cells from the respective cell planes in L3G2.

The layer 4 output will be the N^2 DFT coefficients. The planes in layer 4 are also grouped into 3. The group 1 will have 4 planes, single cell in each, giving the real coefficients $Y_{0,0}$, $Y_{0,3}$, $Y_{3,0}$ and $Y_{3,3}$.

The group 2 consists of 8 cell planes with M-1 cells in each plane. The output from this group corresponds to the coefficients from row 0, row M, column 0 and column M excluding the coefficients of group 1 with odd indexed coefficients and even indexed coefficients from a row / column grouped in separate planes.

The group 3 consists of 2(N-2) cell planes with M-1 cells each. The output from this group will be the DFT coefficients other than that in group 1 & 2. The cells in L4G2 and L4G3 involve scalar multiplication of complex numbers where as the cells in all others need only real addition/subtraction.

Fig. 4.3 shows the detailed schematic diagram of parallel distributed computation of 6x6 - point DFT. In the fig. 4.3, the dots in each rectangle represent the cells and the order of each plane, represented by rectangles with continuous line, can easily be identified. The inputs to L2G2, L2G3, L3G1, L3G2 and L3G3 are dependent on the respective planes in the previous layer and the computations are the same as that of other planes in the group, hence all the connections are not shown in fig. 4.3. Similarly, in L4G2 & L4G3 the computations of the planes are similar to the one shown.

4.3.2 The Algorithm

The output of each cell in L1G1 is the sum of two inputs from L0. Similarly, the output of each cell in L1G2 is the difference between the output of two cells in L0. Output of cells in L1G3 is obtained from one cell in L0 at a specific position.

For 0 $i, j < M = N/2 = 3$.

$$\begin{aligned} \text{VLP}(i,j) &= x(2i, 2j) + x(2i, 2j + 1) & \text{VLPA}(i,j) &= x(2i + 1, 2j) + x(2i + 1, 2j + 1) \\ \text{HAP}(i,j) &= x(2i, 2j) + x(2i + 1, 2j) & \text{HAPL}(i,j) &= x(2i, 2j + 1) + x(2i + 1, 2j + 1) \\ \text{DP}(i,j) &= x(2i, 2j) + x(2i + 1, 2j + 1) & \text{DPA}(i,j) &= x(2i + 1, 2j) + x(2i, 2j + 1) \end{aligned}$$

The output of cells in L1G2 is obtained by replacing the plus sign in the equations of L1G1 with minus to obtain the corresponding cell value of each plane in the order given in the figure.

$$\begin{aligned} \text{VRP}(i,j) &= x(2i, 2j) - x(2i, 2j + 1) & \text{VRPA}(i,j) &= x(2i + 1, 2j) - x(2i + 1, 2j + 1) \\ \text{HBP}(i,j) &= x(2i, 2j) - x(2i + 1, 2j) & \text{HBPL}(i,j) &= x(2i, 2j + 1) - x(2i + 1, 2j + 1) \\ \text{CP}(i,j) &= x(2i, 2j) - x(2i + 1, 2j + 1) & \text{CPA}(i,j) &= x(2i + 1, 2j) - x(2i, 2j + 1) \end{aligned}$$

However, the cell values of the planes of L1G3 are:

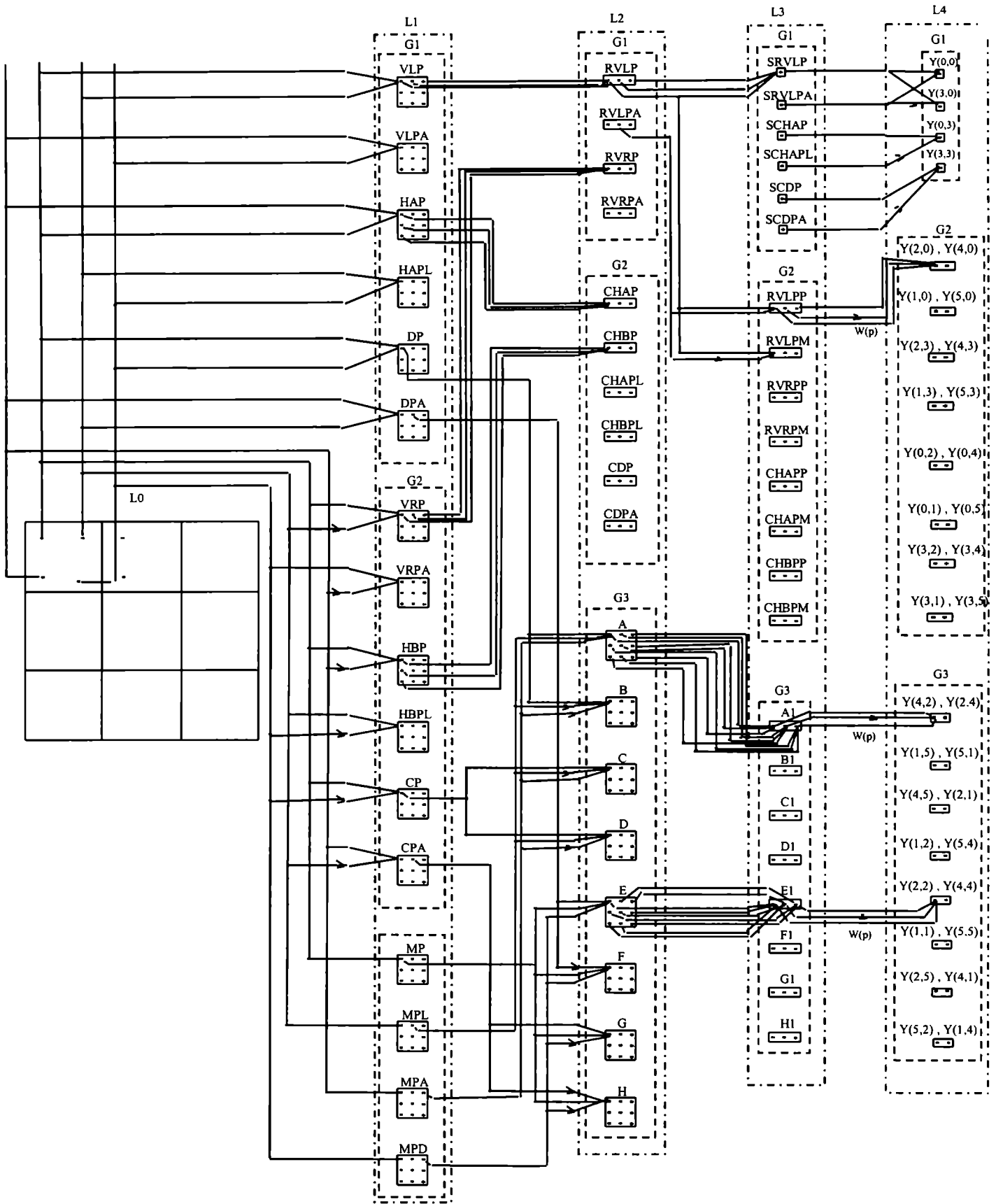


Fig.4.3 Schematic diagram for parallel distributed computation of 6x6 point DFT

For 0 $i, j < M$

$$\begin{aligned} MP(i,j) &= x(2i, 2j) & MPL(i, j) &= x(2i, 2j + 1) \\ MPA(i,j) &= x(2i + 1, 2j) & MPD(i, j) &= x(2i + 1, 2j + 1) \end{aligned}$$

The algorithm to find the output of cells in L2 is

i) For L2G1,

$$RVLP(i) = \sum_{j=0}^{M-1} VLP(i, j) \quad , 0 \leq i < M \quad \text{_____ (1)}$$

Similarly, the values of the arrays RVLPA, RVRP, RVRPA could be arrived at by replacing VLP with VLPA, VRP, VRPA respectively in equation (1).

ii) For L2G2,

$$CHAP(i) = \sum_{j=0}^{M-1} HAP(j, i) \quad , 0 \leq i < M \quad \text{_____ (2)}$$

Similarly, the values of the arrays CHBP, CHAPL, CHBPL, CDP, CDPA are obtained by replacing HAP with HBP, HAPL, HBPL, DP, DPA respectively in equation (2).

iii) For L2G3,

For 0 $i, j < M$

$$\begin{aligned} A(i,j) &= DP(i,j) + MPL(i,((j+1))_M) + MPA(i,((j+2))_M) \\ B(i,j) &= DP(i,j) - MPL(i,((j+1))_M) - MPA(i,((j+2))_M) \\ C(i,j) &= CP(i,j) - MPL(i,((j+1))_M) + MPA(i,((j+2))_M) \\ D(i,j) &= CP(i,j) + MPL(i,((j+1))_M) - MPA(i,((j+2))_M) \\ E(i,j) &= MP(i,j) + DPA(i,((j+1))_M) + MPD(i,((j+2))_M) \\ F(i,j) &= MP(i,j) - DPA(i,((j+1))_M) + MPD(i,((j+2))_M) \\ G(i,j) &= MP(i,j) + CPA(i,((j+1))_M) - MPD(i,((j+2))_M) \\ H(i,j) &= MP(i,j) - CPA(i,((j+1))_M) - MPD(i,((j+2))_M) \end{aligned}$$

In L3 the first group L3G1 consists of six planes, each with single cell only whereas both L3G2 and L3G3 consists of 8 planes having 3 cells in each. The output of the cells in L3 can be found out as follows:

ii) For L3G1,

$$SRVLP = \sum_{i=0}^{M-1} RVLP(i) \quad \text{_____ (3)}$$

Similarly, the values of the cells SRVLPA, SCHAP, SCHAPL, SCDP, SCDPA are derived by replacing RVLP with RVLPA, CHAP, CHAPL, CDP, CDPA respectively in equation (3).

iii) For L3G2,

For $0 \leq i < M$

$$\text{RVLPP}(i) = \text{RVLP}(i) + \text{RVLPA}(((i+1))_M)$$

$$\text{RVLPM}(i) = \text{RVLP}(i) - \text{RVLPA}(((i+1))_M)$$

$$\text{RVRPP}(i) = \text{RVRP}(i) + \text{RVRPA}(((i+1))_M)$$

$$\text{RVRPM}(i) = \text{RVRP}(i) - \text{RVRPA}(((i+1))_M)$$

$$\text{CHAPP}(i) = \text{CHAP}(i) + \text{CHAPL}(((i+1))_M)$$

$$\text{CHAPM}(i) = \text{CHAP}(i) - \text{CHAPL}(((i+1))_M)$$

$$\text{CHBPP}(i) = \text{CHBP}(i) + \text{CHBPL}(((i+1))_M)$$

$$\text{CHBPM}(i) = \text{CHBP}(i) - \text{CHBPL}(((i+1))_M)$$

iv) For L3G3,

$$A1(i) = \sum_{j=0}^{M-1} A(j, i+j) \quad 0 \leq i < M$$

Similarly, the arrays B1, C1, D1 are obtained by replacing A in the above equation with B, C, D respectively.

$$E1(i) = \sum_{j=0}^{M-1} E(j, i-j) \quad , 0 \leq i < M$$

Similarly, the arrays F1, G1, H1 are obtained by replacing E in the above equation with F, G, H respectively.

The DFT coefficients $Y_{k1,k2}$ will be obtained from the layer L4. The group L4G1 gives the real coefficients $Y_{0,0}, Y_{0,3}, Y_{3,0}, Y_{3,3}$ for which no multiplications are required and the input is

from L3G1. The complex coefficients $Y_{k1,k2}$ can be expressed as $Y_{k1,k2} = \sum_{p=0}^{M-1} Y_{k1,k2}^{(p)} \cdot W(p)$

and $Y_{(((N-k1))_N, ((N-k2))_N)} = Y_{k1,k2}^{(0)} - \sum_{p=1}^{M-1} Y_{k1,k2}^{(((M-p))_M)} W(p)$ where $Y_{k1,k2}^{(p)}$ corresponds to the

output of the p^{th} cell in a cell-plane of L3G2 or L3G3 and $W(p) = \exp((-j2\pi/N)p)$.

The algorithm for L4 is

i) For L4G1

$$Y_{0,0} = \text{SRVLP} + \text{SRVLPA}$$

$$Y_{3,0} = \text{SRVLP} - \text{SRVLPA}$$

$$Y_{0,3} = \text{SCHAP} - \text{SCHAPL}$$

$$Y_{3,3} = \text{SCDP} - \text{SCDPA}$$

ii) For L4G2

$$\begin{aligned}
 Y_{1,0} &= RVLPM(0) - RVLPM(1) \cdot W1 + RVLPM(2) \cdot W2 \\
 Y_{2,0} &= RVLPP(0) - RVLPP(1) \cdot W1 + RVLPP(2) \cdot W2 \\
 Y_{1,3} &= RVRPM(0) - RVRPM(1) \cdot W1 + RVRPM(2) \cdot W2 \\
 Y_{2,3} &= RVRPP(0) - RVRPP(1) \cdot W1 + RVRPP(2) \cdot W2 \\
 Y_{0,1} &= CHAPM(0) - CHAPM(1) \cdot W1 + CHAPM(2) \cdot W2 \\
 Y_{0,2} &= CHAPP(0) - CHAPP(1) \cdot W1 + CHAPP(2) \cdot W2 \\
 Y_{3,1} &= CHBPM(0) - CHBPM(1) \cdot W1 + CHBPM(2) \cdot W2 \\
 Y_{3,2} &= CHBPP(0) - CHBPP(1) \cdot W1 + CHBPP(2) \cdot W2
 \end{aligned}$$

iii) For L4G3

$$\begin{aligned}
 Y_{1,1} &= F1(0) - F1(1) \cdot W1 + F1(2) \cdot W2 \\
 Y_{2,2} &= E1(0) - E1(1) \cdot W1 + E1(2) \cdot W2 \\
 Y_{5,1} &= B1(0) - B1(1) \cdot W1 + B1(2) \cdot W2 \\
 Y_{4,2} &= A1(0) - A1(1) \cdot W1 + A1(2) \cdot W2 \\
 Y_{2,1} &= C1(0) - C1(1) \cdot W1 + C1(2) \cdot W2 \\
 Y_{1,2} &= D1(0) - D1(1) \cdot W1 + D1(2) \cdot W2
 \end{aligned}$$

Where $W1 = \exp(-j2\pi/N)$ and $W2 = \exp(-j4\pi/N)$

4.3.3 Example

An example for computation of 6x6 point DFT is shown below. The data matrix chosen is

$$[A] = \begin{bmatrix}
 1 & 2 & 2 & 3 & 1 & 3 \\
 3 & 4 & 4 & 1 & 2 & 4 \\
 1 & 1 & 0 & 1 & 1 & 2 \\
 1 & 1 & 2 & 3 & 3 & 4 \\
 1 & 3 & 1 & 2 & 2 & 3 \\
 2 & 4 & 3 & 4 & 4 & 1
 \end{bmatrix}$$

L1G1

VLP	VLPA	HAP	HAPL	DP	DPA
3 5 4	7 5 6	4 6 3	6 4 7	5 3 5	5 7 5
2 1 3	2 5 7	2 2 4	2 4 6	2 3 5	2 3 5
4 3 5	6 7 5	3 4 6	7 6 4	5 5 3	5 5 7

L1G2

VRP	VRPA	HBP	HBPL	CP	CPA
-1 -1 -2	-1 3 -2	-2 -2 -1	-2 2 -1	-3 1 -3	1 1 -1
0 -1 -1	0 -1 -1	0 -2 -2	0 -2 -2	0 -3 -3	0 1 1
-2 -1 -1	-2 -1 -3	-1 -2 -2	-1 -2 2	-3 -3 1	-1 1 1

L1G3

MP	MPL	MPA	MPD
1 2 1	2 3 3	3 4 2	4 1 4
1 0 1	1 1 2	1 2 3	1 3 4
1 1 2	3 2 3	2 3 4	4 4 1

L2G1

RVLP	RVLPA	RVRP	RVRPA
12	18	-4	0
6	14	-2	-2
12	18	-4	0

L2G2

CHAP	CHBP	CHAPL	CBPL	CDP	CDPA
9 12 13	-3 -6 -5	15 14 17	-3 -2 -1	12 11 13	12 15 17

L2G3

A	B	C	D	E	F	G	H
10 9 11	0 -3 -1	-4 1 -1	-2 1 -5	12 11 7	-2 1 -3	-2 -3 1	-4 -1 -1
6 6 8	-2 0 2	2 -4 -2	-2 -2 -4	8 6 6	2 -4 2	-2 0 -2	-4 -2 -2
11 10 9	-1 0 -3	-1 -4 1	-5 -2 1	7 12 11	-3 -2 1	1 -2 -3	-1 -4 -1

L3G1

SRVLP	SRVLPA	SCHAP	SCHAPL	SCDP	SCDPA
30	50	34	46	36	44

L3G2

RVLPP	RVRPP	CHAPP	CHBPP	RVLPM	RVRPM	CHAPM	CHBPM
26	-6	23	-5	-2	-2	-5	-1
24	-2	29	-7	-12	-2	-5	-5
30	-4	28	-8	-6	-4	-2	-2

				<u>L3G3</u>			
A1	B1	C1	D1	E1	F1	G1	H1
25	-3	-7	-3	30	-2	-6	-10
28	-2	-2	-8	30	-10	-8	-6
27	-3	-3	-9	20	4	2	-4

<u>L4G1</u>			
$Y_{0,0}$	$Y_{0,3}$	$Y_{3,0}$	$Y_{3,3}$
80	-12	-20	-8

<u>L4G2</u>				<u>L4G3</u>			
$Y_{1,0}$, $Y_{5,0}$	$Y_{2,0}$, $Y_{4,0}$	$Y_{1,1}$, $Y_{5,5}$	$Y_{2,2}$, $Y_{4,4}$	$Y_{2,3}$, $Y_{4,3}$	$Y_{1,3}$, $Y_{5,3}$	$Y_{2,5}$, $Y_{4,1}$	$Y_{5,2}$, $Y_{1,4}$
7+j5.2, 7-j5.2	-1-j5.2, -1+j5.2	1+j12.12, 1-j12.12	5+j8.66, 5-j8.66	-3+j1.73, -3-j1.73	1-j1.73, 1+j1.73	-3-j8.66, -3+j8.66	-5-j1.73, -5+j1.73
$Y_{0,2}$, $Y_{0,4}$	$Y_{0,1}$, $Y_{0,5}$	$Y_{4,2}$, $Y_{2,4}$		-5.5+j0.866, -5.5-j0.866	-1.5+j2.6, -1.5-j2.6	-2.5+j0.866, -2.5-j0.866	
$Y_{3,2}$, $Y_{3,4}$	$Y_{3,1}$, $Y_{3,5}$	$Y_{5,1}$, $Y_{1,5}$		2.5+j0.866, 2.5-j0.866	2.5+j2.6, 2.5-j2.6	-0.5-j0.866, -0.5+j0.866	
		$Y_{2,1}$, $Y_{4,5}$	$Y_{1,2}$, $Y_{5,4}$				
		-4.5-j0.866, -4.5+j0.866	5.5+j0.866, 5.5-j0.866				

4.4 PARALLEL DISTRIBUTED MODEL FOR NxN-POINT DFT, ((N))₄=2

The fig. 4.4 shows the block diagram of the hierarchical network model to implement NxN-point DFT when ((N))₄ = 2. The structure of the model used, for N=6, in the previous section is slightly modified to reduce the complexity by eliminating a few planes as explained below. It consists of a cascade of four layers with 3 groups of cell planes each. The layer one consists of 4 planes, each of size MxM, in all the groups. The planes in L1G1 are named VLP, VLPA, HAP & HAPL, in L1G2 are VRP, VRPA, HBP, HBPL and that in L1G3 are MP, MPL, MPA & MPD. The planes in L2G1 are RVLP, RVLPA, RVRP, RVRPA with M cells each. The second group in L2 has the planes CHAP, CHBP, CHAPL, CHBPL, each with M cells. The third group consists of 2(N-2) cell planes, each of dimension MxM, with the names indicating the index of basic coefficients such as A(1,1), A(2,1)... A(N-1,2), i.e; indices of columns one and two with the row index k1≠M. The planes in L3G1 are SRVLP,

SRVLP, SRVRP, SRVRPA with single cell. L3G2 consists of 8 cell planes, M cells in each, with name B(k,0), B(k, M), B(0,k) & B(M,k) where k = 1, 2 and L3G3 consists of 2(N-2) cell planes, with M cells each, having the name B(k1, k2) where k2 = 1, 2 and k1 = 1, 2, N-1 & k1 ≠ M.

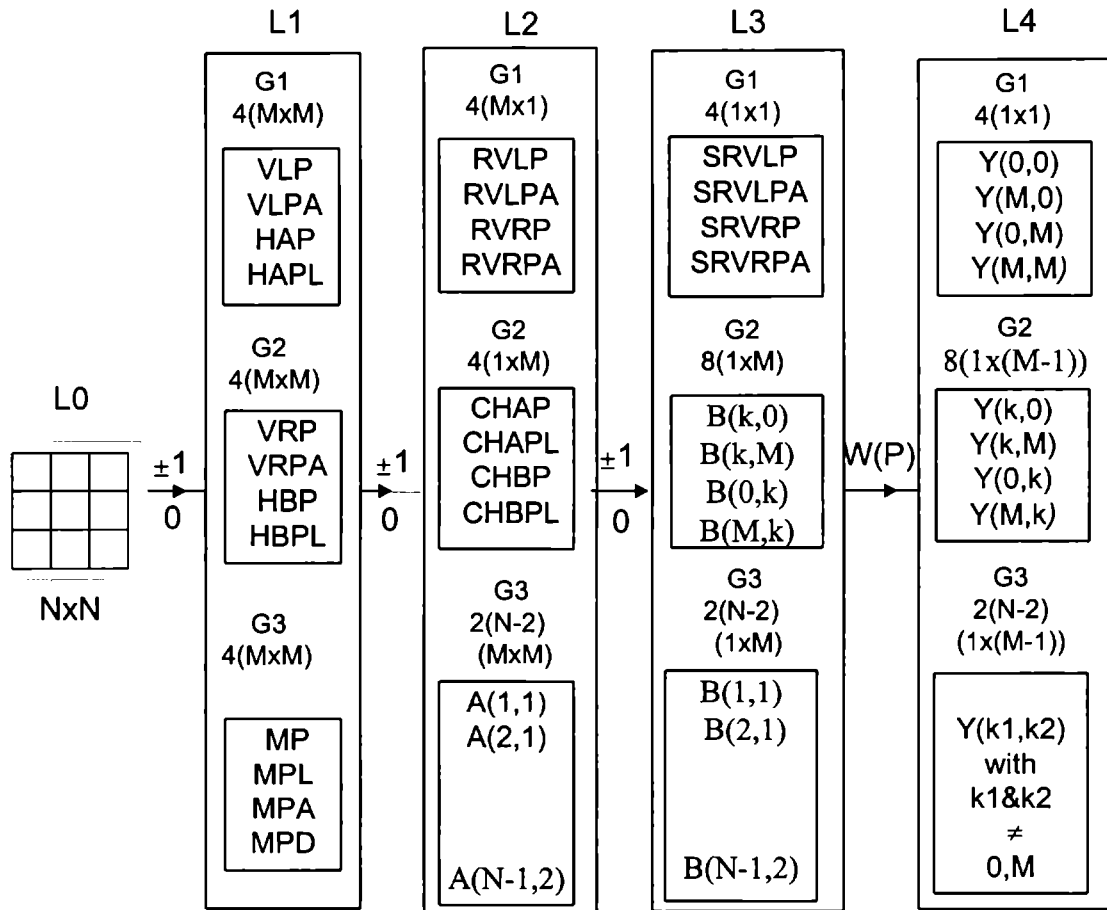


Fig. 4.4 Block diagram of the network architecture for $((N))_4 = 2$

The algorithm for the layer one is similar to that for N=6, only the size of the group varies. The cell planes DP, DPA, CP, CPA are eliminated since they can easily be avoided by modifying the algorithm so that the complexity reduces. The output of each cell in L1G1 is the sum of two inputs from L0. Similarly, the output of each cell in L1G2 is the difference between the output of two cells in L0. The output of cells in L1G3 is obtained from one cell in L0 at a specific position.

The input-output relation for each cell in L1 is as follows.

For 0 $i, j < M = N/2$

For L1G1

$$\text{VLP}(i,j) = x(2i, 2j) + x(2i, 2j + 1) \quad \text{VLPA}(i,j) = x(2i + 1, 2j) + x(2i + 1, 2j + 1)$$

$$\text{HAP}(i,j) = x(2i, 2j) + x(2i + 1, 2j) \quad \text{HAPL}(i,j) = x(2i, 2j + 1) + x(2i + 1, 2j + 1)$$

For L1G2

$$\text{VRP}(i,j) = x(2i, 2j) - x(2i, 2j + 1) \quad \text{VRPA}(i,j) = x(2i + 1, 2j) - x(2i + 1, 2j + 1)$$

$$\text{HBP}(i,j) = x(2i, 2j) - x(2i + 1, 2j) \quad \text{HBPL}(i,j) = x(2i, 2j + 1) - x(2i + 1, 2j + 1)$$

For L1G3

$$\text{MP}(i,j) = x(2i, 2j) \quad \text{MPL}(i,j) = x(2i, 2j + 1)$$

$$\text{MPA}(i,j) = x(2i + 1, 2j) \quad \text{MPD}(i,j) = x(2i + 1, 2j + 1)$$

The layer L2 also consists of three groups of cell planes. The group L2G1 consists of 4 cell- planes of dimension $M \times 1$. The algorithm to find the output of cells in L2 is:

i) For L2G1

For 0 $i < M$

$$\text{RVLP}(i) = \sum_{j=0}^{M-1} \text{VLP}(i, j) \quad \text{RVRP}(i) = \sum_{j=0}^{M-1} \text{VRP}(i, j)$$

$$\text{RVLPA}(i) = \sum_{j=0}^{M-1} \text{VLPA}(i, j) \quad \text{RVRPA}(i) = \sum_{j=0}^{M-1} \text{VRPA}(i, j)$$

In this the cell values of RVLP represents sum of data on each row with even index and RVRP represents that on rows of odd index. Similarly, the cell values of RVLPA represents sum of data on each row with even index where the data on odd column position are sign reversed and RVRPA represents that on rows of odd index.

ii) For L2G2

For 0 $i < M$

$$\text{CHAP}(i) = \sum_{j=0}^{M-1} \text{HAP}(j, i) \quad \text{CHBP}(i) = \sum_{j=0}^{M-1} \text{HBP}(j, i)$$

$$\text{CHAPL}(i) = \sum_{j=0}^{M-1} \text{HAPL}(j, i) \quad \text{CHBPL}(i) = \sum_{j=0}^{M-1} \text{HBPL}(j, i)$$

Derivation of algorithm for L2G3

The primitive symbol combination at the first row of cells in $Y_{k1,k2}^{(0)}$ for $N=6$ are as follows:

$$N=6$$

		<u>k1,k2</u>
1)	$MP_{0,0} + DPB_{0,1} + MPD_{0,2}$	1,1
2)	$CP_{0,0} + MPR_{0,1} + MPA_{0,2}$	2,1
3)	$MP_{0,0} + CPA_{0,1} + MPC_{0,2}$	4,1
4)	$DP_{0,0} + MPR_{0,1} + MPB_{0,2}$	5,1
5)	$CP_{0,0} + MPL_{0,1} + MPB_{0,2}$	1,2
6)	$MP_{0,0} + DPA_{0,1} + MPD_{0,2}$	2,2
7)	$DP_{0,0} + MPL_{0,1} + MPA_{0,2}$	4,2
8)	$MP_{0,0} + CPB_{0,1} + MPC_{0,2}$	5,2

Similarly, the primitive symbol combination at the first row of cells in $Y_{k1,k2}^{(0)}$ for $N = 10, 14$ and 18 (of L2G3) are given in Appendix B. Since $DP = MP + MPD$, $CP = MP + MPC$, $DPA = MPL + MPA$, $CPA = MPA + MPR$, $DPB = -DPA$, $MPR = -MPL$, $MPC = -MPD$, $MPB = -MPA$, $MN = -MP$, $CPB = -CPA$, we can express the symbol combinations in terms of the four symbols MP , MPL , MPA , MPD and derive an expression for finding the cell values of the cell planes in L2G3. Thus the symbol in $(0, 0)^{th}$ cell of the pictorial representation, for any $(k1, k2)$ of group 3 with $p = 0$, will be MP and that of $(0, M2)^{th}$ cell, $M2 = (M-1)/2$, will be MPL if $k2$ is even and MPR when $k2$ is odd. Thus $(0, M2)^{th}$ cell has MPL with positive sign if $k2$ is even and negative if $k2$ is odd. Similarly, MPA will be positive if $k1$ is even and negative if $k1$ is odd. Also MPD is positive if both $k1$ & $k2$ are odd or even otherwise negative. The position of the symbol MPA moves in steps of $((k1 \cdot M2))_M$ as $k1$ increases till $k1 = M$ and then reversing the sign. Similarly, the position of MPD is changing with increase in $k1$ in a similar fashion in steps of $((k1+1)M2))_M$. Thus the expression for finding the cell values of L2G3 are derived.

The algorithm for L2G3 is:

For $k2 = 1, 2$

For $j = 0, 1$

For $i = 1, M-1$

$k1 = ((k2 \cdot i + j \cdot M))_N$, $M2 = (M-1)/2$

For $0 \leq m < M$

$A(k1, k2)(l, m) = MP_{l,m} + (-1)^{k1} MPA_{l,m1} + (-1)^{k2} MPL_{l,m2} + (-1)^{k1+k2} MPD_{l,m3}$, Where
 $m1 = ((m + ((i.M2))_M))_M$, $m2 = ((m + M2))_M$, $m3 = ((m + (((i + 1)M2))_M))_M$

In the above equation $A(k1, k2)$ represents the address of a cell plane and (l, m) represents the index of the cell in that plane. There will be $2(N-2)$ cell planes of size $M \times M$ in L2G3.

The output of the cells in L3 can be found out as follows:

i) For L3G1

$$\begin{aligned} SRVLP &= \sum_{i=0}^{M-1} RVLP(i) & SRVRP &= \sum_{i=0}^{M-1} RVRP(i) \\ SRVLPA &= \sum_{i=0}^{M-1} RVLPA(i) & SRVRPA &= \sum_{i=0}^{M-1} RVRPA(i) \end{aligned}$$

The value of SRVLP represents the sum of data of all rows with even index and SRVLPA represents that of odd index. Similarly, SRVRP represents the sum of data of all rows with even index where the data with odd column index is sign reversed and SRVRPA represents the sum of data of all odd indexed rows where the data with odd column index is sign reversed.

Derivation of algorithm for L3G2

The pictorial representation shows that the group 2 coefficients will have the identical rows/columns depending on the position of the frequency index. Only two symbols will be used in the column/row. Thus in this group, the cell values of each plane are formed by combining the row/column sum of two categories of cell planes. Different cell values are obtained by circularly changing the index value. In this group each plane represents the value of $Y_{k1,k2}^{(p)}$ for different values of p . The following relations can be derived by analyzing the pictorial representation for the basic set of coefficients, for $p=0$, corresponding to group 2.

$$\begin{aligned} Y_{1,0}^{(0)} &= \sum_{m=0}^{M-1} VLP_{0,m} - \sum_{m=0}^{M-1} VLPA_{M2,m} & Y_{1,M}^{(0)} &= \sum_{m=0}^{M-1} VRP_{0,m} - \sum_{m=0}^{M-1} VRPA_{M2,m} \\ Y_{2,0}^{(0)} &= \sum_{m=0}^{M-1} VLP_{0,m} + \sum_{m=0}^{M-1} VLPA_{M2,m} & Y_{2,M}^{(0)} &= \sum_{m=0}^{M-1} VRP_{0,m} + \sum_{m=0}^{M-1} VRPA_{M2,m} \\ Y_{0,1}^{(0)} &= \sum_{m=0}^{M-1} HAP_{m,0} - \sum_{m=0}^{M-1} HAPL_{m,M2} & Y_{M,1}^{(0)} &= \sum_{m=0}^{M-1} HBP_{m,0} - \sum_{m=0}^{M-1} HBPL_{m,M2} \\ Y_{0,2}^{(0)} &= \sum_{m=0}^{M-1} HAP_{m,0} + \sum_{m=0}^{M-1} HAPL_{m,M2} & Y_{M,2}^{(0)} &= \sum_{m=0}^{M-1} HBP_{m,0} + \sum_{m=0}^{M-1} HBPL_{m,M2} \end{aligned}$$

The relation for other values of p can also be derived by changing the row index and column index in a circular fashion as seen in the pictorial representation. Other (k_1, k_2) pairs in this group are also related to these symbols in a similar way. Hence, to derive different Y_{k_1, k_2} of L4G2, all combinations in the current group of planes will be computed. From the relations given above and the pictorial representation we can observe that the first row/column will be with VLP/VRP or HAP/HBP and the M_2^{th} row/column will be with VLPA/ VRPA or HAPL/HBPL depending on the position of index (k_1, k_2) . The position of the rows/columns of symbols will be circularly shifted as shown in the pictorial representation but with the spacing between the two rows/columns of different symbols equal to M_2 . Thus we can find the index of row/column in which VLP/VRP or HAP/HBP is present from the value of n such that $((2.n.k))_M = p$ for different values of p where $k = k_1$ or k_2 that is not zero or M .

ii) For L3G2

For $k = 1, 2$

For $0 \leq p < M$

Find n such that $((2.n.k))_M = p, 0 \leq n \leq M-1$

$$B(k, 0)(p) = RVLP(n) + (-1)^k RVLPA(((n + M_2))_M), M_2 = (M-1)/2$$

$$B(k, M)(p) = RVRP(n) + (-1)^k RVRPA(((n + M_2))_M)$$

$$B(0, k)(p) = CHAP(n) + (-1)^k CHAPL(((n + M_2))_M)$$

$$B(M, k)(p) = CHBP(n) + (-1)^k CHBPL(((n + M_2))_M)$$

Derivation of algorithm for L3G3

The algorithm for the group L3G3 is derived by analyzing the pictorial representation, described in chapter 3, of group 3 coefficients for different values of N . First we can find the position of the symbol MP for $p=0$ which depends on k_2 . As the row number increases the symbol MP will be shifting to the left by k_1 times the row number in a circular fashion and k_2 times the row number down the rows in a circular fashion. Using this principle we can derive the algorithm for this group as given below.

iii) For L3G3

For $k_2 = 1, 2$

For $j = 0, 1$

For $i = 1, M-1$

$$k_1 = ((k_2 \cdot i + j \cdot M))_N, M_2 = (M-1)/2$$

For $0 \leq p < M$

Find n such that $((2.k2.n))_M = p, 0 \leq n \leq M-1$

$$B(k1,k2)(p) = \sum_{l=0}^{M-1} A(k1,k2)((k2.l))_M, ((n - k1.l))_M$$

Derivation of algorithm for the layer L4

The DFT coefficients $Y_{k1,k2}$ will be obtained from the layer L4. The connection weights from L3 to L4 will be the twiddle factors. The group L4G1 gives the real coefficients $Y_{0,0}, Y_{0,M}, Y_{M,0}, Y_{M,M}$ for which the connection weights will be ± 1

$$\begin{aligned} \text{where } Y_{0,0} &= \text{SRVLP} + \text{SRVLP} & Y_{M,0} &= \text{SRVLP} - \text{SRVLP} \\ Y_{0,M} &= \text{SRVRP} + \text{SRVRP} & Y_{M,M} &= \text{SRVRP} - \text{SRVRP} \end{aligned}$$

The complex coefficients $Y_{k1,k2}$ can be expressed as

$$Y_{k1,k2} = \sum_{p=0}^{M-1} Y_{k1,k2}^{(p)} \cdot W(p)$$

where $Y_{k1,k2}^{(p)}$ corresponds to the output of the p^{th} cell in a cell-plane of L3G2 or L3G3 and $W(p) = \exp(-j2\pi p / N)$. But the analysis of the pictorial representation shows that a number of DFT coefficients can be derived from the permutations of $Y_{k1,k2}^{(p)}, 0 \leq p \leq M-1$ corresponding to a given $(k1, k2)$. The table B.1, B.2, B.3 show the set of indices of the coefficients that are related by the permutation of $Y_{k1,k2}^{(p)}$, over p , for $N = 6, 10, 14$ respectively. In these tables the indices with '*' are the conjugate of the corresponding index given in the bracket, ie. $(k1, k2)^*$ is the complex conjugate of the coefficient with index $(k1, k2)$. By analyzing these tables, it is seen that the values of $Y_{k1,k2}^{(p)}$ for all p need be computed for $2N+8$ different set of $(k1, k2)$ only as given table B.4. Also a set of relations are derived in terms of $Y_{k1,k2}^{(p)}$ as permutation over p for different values of $(k1, k2)$ as given in Appendix B. From the equations for $N = 6, 10, 14$ we can see that in general the conjugates are related by the relation $Y_{(k1,k2)^*}^{(0)} = Y_{k1,k2}^{(0)}$ & $Y_{(k1,k2)^*}^{(p)} = -Y_{k1,k2}^{(M-p)}, 1 \leq p \leq M-1$ where $(k1, k2)^* = (((N-k1))_N, ((N-k2))_N)$.

The DFT coefficients of the group L4G2 are obtained by using the following algorithm:

For $k_1 = 1, 2$

For $k_2 = 0, M$

For $k = 1, 3, 5, \dots M-2$

$$Y_{(((k.k_1))_N, k_2)} = \sum_{p=0}^{M-1} (-1)^p B_{k_1, k_2}^{(((k.p))_M)} \cdot W(p)$$

$$Y_{(k_2, ((k.k_1))_N)} = \sum_{p=0}^{M-1} (-1)^p B_{k_2, k_1}^{(((k.p))_M)} \cdot W(p)$$

All the symbols in the pictorial representation of $Y_{k_1, k_2}^{(p)}$ for odd values of p are the sign reversed form of that of the even p . But that change is not considered in the previous layers and hence $(-1)^p$ is used in the expressions for the final DFT relations of L4G2 & L4G3.

The DFT coefficients of L4G3 are obtained using the algorithm given below:

For $k_2 = 1, 2$

For $j = 0, 1$

For $i = 1, M-1$

$$k_1 = ((k_2 \cdot i + j \cdot M))_N$$

$$Y_{(((k.k_1))_N, ((k.k_2))_N)} = \sum_{p=0}^{M-1} (-1)^p B_{k_1, k_2}^{((k.p))_M} \cdot W(p), \quad k = 1, 3, \dots M-2$$

The remaining DFT coefficients in L4G2 & L4G3 can be found out by using the relation

$$Y_{(k_1, k_2)} = \sum_{p=0}^{M-1} (-1)^p \cdot B_{k_1, k_2}^{(((M-p))_M)} \cdot W(p)$$

4.5 SIMULATION RESULTS

The model developed in this chapter is basically using a parallel distributed approach for the computational application. So the simulation of the model requires a number of processes to be running simultaneously. Also it is aiming at implementing with very simple units (adders) to perform the complex task of computing the 2-D DFT for variable size of the data matrix. Thus the simulation of the model on a sequential machine with single processor will not give a proper evaluation of the model. Since the algorithm is for parallel and distributed operation, it is difficult to implement on a sequential machine that uses a sequential programming. But the JAVA programming language can simulate parallel processing functions due to the multithread design feature available. For each process to be executed

sequentially, individual threads were created. Hence, the model is simulated using the JAVA language.

The parallel distributed model developed in the previous section is simulated on a Pentium DX machine, 120 MHz clock, 16MB RAM, using JAVA programming language under the windows 95 environment.

The model is simulated and the results of computation are compared with that of the direct DFT. It is found that the results of computation are one and the same. The model can be implemented in four different ways depending on the requirement of speed and complexity.

1. Fully parallel implementation in which each cell will be operating in parallel so that the time taken for the computation in one layer is same as the time required for the computation in one cell. The realization will have the maximum speed but the complexity will also be high.
2. Plane sequential implementation where the planes are operating in parallel and within a plane the operations are sequential. Since all the cells in a plane are identical and doing the same operation on different data that are circularly shifted in position, the implementation of this will be very easy.
3. Group sequential implementation in which the groups are operating in parallel and within a group the operations are sequential. This can reduce the complexity, but this is not a good choice due to the difference in complexity of various groups.
4. Fully sequential implementation in which the complete operations within a layer are performed sequentially.

The table 4.1 shows the computation time in the four different schemes mentioned above for different order of the data matrix for a typical example. The table also shows the time taken for computing the DFT using row column method for the same data. Even though the values obtained are not the absolute values, a comparison can be made from the results available. The following inferences can be drawn from the table 4.1.

By comparing the timings for the fully sequential implementation of the PDP method with direct DFT, shows the reduction in computation time due to the new definition of the DFT relation alone. The complexity for the implementation of this scheme will be the minimum. Still there is a good reduction in computation time. The reduction in computation time is more predominant as the order increases.

Table 4.1: The computation time of PDP model for different degrees of parallelism and that of the Direct DFT

N	DFT (msec)	PDP Fully sequential (msec)	PDP Group sequential (msec)	PDP Plane sequential (msec)	PDP Fully parallel (msec)
6	82	14	10.21	1.358	0.3558
10	648	62	30.93	3.357	0.4654
14	2310	173	131.04	3.827	0.53415
18	6100	378	302.59	9.9365	0.70515
22	14440	674	577.8	15.3137	0.86525
26	27960	1154	956.71	21.027	0.977
30	48830	1761	1529.63	28.833	1.195
34	80850	2627	2224.5	36.5859	1.3187
38	125070	3542	3184.02	46.448	1.4984
42	187660	4797	4272.61	56.2125	1.6356
46	278360	6489	5958.86	70.926	1.8343
50	369160	8177	7254.33	79.3825	1.9278

The group sequential method is not showing much improvement in computation time compared to the fully sequential method. This is due to the uneven grouping of coefficients. That is the first group has the four simplest coefficients, second group has $4(N-2)$ coefficients and the third group has the remaining coefficients which increases as N increases. But this can be solved by splitting the third group into more groups, depending on the order N , so that all the groups will require approximately equal computation time. Then the group sequential method will be advantageous in reducing the complexity of the system with improvement in speed. Presently, group sequential approach is not an advantageous method compared to other schemes.

The plane sequential method shows a good reduction in computation time compared to the group sequential and fully sequential methods. There will be a considerable saving in implementing the system also. The increase in order will have a greater effect on the number of cells in one plane compared to the number planes in a layer. Thus the complexity will not

increase much as the order is increased. As seen in table 4.1, the increase in computation time compared to direct DFT, fully sequential PDP or group sequential PDP methods is less.

The fully parallel method shows the maximum speed of computation compared to other schemes. But the arithmetic units to be used will be the maximum and increases in terms of M^2 . But the hierarchical architecture permits many simplifications in the implementation.

The table 4.2 shows a comparison of the time taken by various groups in implementing the model in the group sequential method for different order of the data matrix. The table shows that the group 3 in L2 and L4 are taking the maximum time compared to other groups. In L2 to L4, the timing is uneven due to the uneven grouping of coefficients. Thus for the present grouping of coefficients, the group sequential method is not as good as others. But splitting the group 3 coefficients into more groups, depending on N, can solve this.

Table 4.2 (Group Sequential): The table shows the time taken for computation of each group for different order of the matrix. The time is in milliseconds

N	6	10	14	18	22	26	30	34	38	42	46	50
Group	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)
L1G1	0.66	1.26	1.76	2.52	3.52	4.78	5.99	7.36	9.72	11.81	13.4	15.93
L1G2	0.5	0.82	1.43	2.2	3.95	4.29	6.54	7.8	9.5	11.37	13.46	15.76
L1G3	0.33	0.66	0.99	1.54	2.09	2.91	3.95	5.54	6.59	7.96	8.46	10.76
L2G1	0.39	0.6	0.99	1.42	2.04	2.74	3.8	4.45	5.44	6.6	7.96	9.11
L2G2	0.55	0.71	1.05	1.65	2.25	3.07	4.33	4.99	6.2	7.97	9.77	11.3
L2G3	4.56	21.71	58.2	129.7	243.8	418	636	932	1326	1781.3	2582	3036.3
L3G1	0.285	0.286	0.291	0.313	0.335	0.357	0.379	0.401	0.428	0.45	0.467	0.5
L3G2	0.94	0.99	1.37	2.14	2.28	2.41	2.74	3.35	3.95	4.23	4.72	5.7
L3G3	1.81	7.09	18.07	39.05	72.94	112	187.3	277.9	393.9	524.5	688.2	884.9
L4G1	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22
L4G2	2.69	7.63	16.92	28.61	44.14	62.95	85.9	111.1	142.64	172.3	205.29	240.52
L4G3	3.18	19.83	53.01	131.2	257.1	414	699.8	1007	145.44	1955	2675	3317.2

The table 4.3 shows the time taken for computation of cell values in one plane of each group when implemented in plane sequential method for different order of the matrix. The first row represents the order N and first column represents the group from which the plane is considered. The time is in milliseconds. The different planes in a group will take the same amount of time as all the planes in the group are doing the same operation on a similar set of data and each plane in a group are of the same dimension.

Table 4.3 (Plane Sequential); The time taken for computation of cell values in one plane of each group for different order of the matrix.

N	6	10	14	18	22	26	30	34	38	42	46	50
Group	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)	Time (msec)
L1G1	0.165	0.315	0.44	0.562	0.88	1.195	1.497	1.84	2.43	2.952	3.35	3.982
L1G2	0.125	0.205	0.357	0.55	0.9875	1.072	1.635	1.95	2.375	2.8425	3.365	3.94
L1G3	0.082	0.165	0.247	0.385	0.522	0.727	0.9875	1.385	1.6475	1.99	2.115	2.69
L2G1	0.097	0.15	0.275	0.355	0.51	0.685	0.95	1.1125	1.36	1.65	1.99	2.277
L2G2	0.1375	0.177	0.2625	0.4125	0.5625	0.767	1.0825	1.247	1.55	1.992	2.442	2.825
L2G3	0.57	1.359	2.425	4.053	6.095	8.708	11.36	14.562	18.347	22.266	29.342	31.63
L3G1	0.0712	0.0715	0.072	0.0782	0.084	0.089	0.095	0.1	0.107	0.1125	0.1167	0.125
L3G2	0.1175	0.1237	0.171	0.267	0.285	0.301	0.342	0.419	0.494	0.529	0.57	0.712
L3G3	0.2263	0.443	0.753	1.22	1.803	2.499	3.344	4.342	5.47	6.556	7.82	9.218
L4G1	0.055	0.055	0.055	0.055	0.055	0.055	0.055	0.055	0.055	0.055	0.055	0.055
L4G2	0.336	0.954	2.115	3.576	5.267	7.868	10.737	13.88	17.825	21.54	25.66	30.06
L4G3	0.397	1.239	2.209	4.1	6.428	8.624	12.5	15.73	20.2	24.43	30.4	34.55

The table 4.4 shows the time taken for computation of one cell in each group when implemented in fully parallel method for different order of the matrix. The first row represents the order N and first column represents the group from which the cell is considered. The time is in microseconds. But this will not be the actual time. Since the computation time for one cell value is very small, the influence of the other tasks that are performed by the windows on the measured time will be more. Thus if this model is implemented in hardware, the speed of computation can be improved well.

Table 4.4 (Fully Parallel): The time taken for computation of one cell in each group for different order of the matrix.

N	6	10	14	18	22	26	30	34	38	42	46	50
Group	Time μsec	Time μsec	Time μsec	Time μsec	Time μsec	Time μsec	Time μsec	Time μsec	Time μsec	Time μsec	Time μsec	Time μsec
L1G1	18.33	12.6	8.98	6.944	7.27	7.07	6.655	6.367	6.73	6.695	6.33	6.304
L1G2	13.89	8.2	7.29	6.79	8.16	6.35	7.266	6.74	6.58	6.445	6.361	6.304
L1G3	9.166	6.6	5.05	4.321	4.31	4.3	4.39	4.79	4.563	4.512	3.998	4.304
L2G1	32.5	30	39.29	39.44	46.36	52.69	63.3	65.44	71.58	78.57	86.52	91.1
L2G2	45.83	35.5	32.14	45.83	51.13	59.04	72.1	73.38	81.58	94.88	106.2	113
L2G3	63.33	54.37	49.49	50.04	50.37	51.5	50.47	50.39	50.82	50.49	55.467	50.6
L3G1	71.25	71.5	72.75	78.25	83.75	89.25	94.75	100.25	107	112.5	116.7	125
L3G2	39.167	24.75	24.46	29.7	25.9	23.17	22.8	24.63	25.99	25.18	24.78	28.5
L3G3	75.42	88.62	107.56	135.6	163.9	192.24	222.96	255.4	287.94	312.2	340	368.7
L4G1	550	550	550	550	550	550	550	550	550	550	550	550
L4G2	168.1	238.4	352.5	447	526.7	655.7	767	867.6	990.3	1076.9	1166.4	1252.7
L4G3	198.7	309.8	368.12	512.6	642.8	718.7	892.6	983.2	1122.2	1221.9	1381.8	1439.7

4.6 COMPLEXITY OF THE MODEL

4.6.1 Computational complexity

The total number of computations involved in the computation of NxN point DFT coefficients is given below

a) Total number of real additions

The number of additions in each group in terms of 2 input real additions is as follows.

$$L1G1 - 4M^2 \quad L2G1 - 4M(M-1) \quad L3G1 - 4(M-1) \quad L4G1 - 4$$

$$L1G2 - 4M^2 \quad L2G2 - 4M(M-1) \quad L3G2 - 8M \quad L4G2 - 8M^2 - 16M + 8$$

$$L1G3 - \text{No} \quad L2G3 - 12M^2(M-1) \quad L3G3 - 4(M^3 - 2M^2 + M) \quad L4G3 - 4(M^3 - 3M^2 + 3M + 2)$$

Thus the total number of real additions = $2.5N^3 - 2N^2 + 2N - 4$.

b) Total number of real multiplications

The multiplication operation is required only in L4G2 and L4G3. The outputs of L3 are real and the output of L4 is the sum of the scalar multiples of the twiddle factors.

The total number of real multiplications = $(N-2)(N^2-4)$

Since the computations will be carried out in parallel and distributed manner the speed of computation will be very high.

4.6.2 Hardware complexity

The table 4.5 shows a comparison of the arithmetic units required in each layer and each group for implementing the model to compute a NxN point DFT in any one of the four realizations. The first three layers involves only real addition and the fourth layer involves multiplication of complex numbers, pre-computed values of the twiddle factors, with real numbers. It shows that the number of adders required in the first layer increases as the square of $M = N/2$ to realize the model in fully parallel implementation. But, since these are with identical units and considering the improvement in speed, the implementation will not be a severe problem.

Table 4.5: The number of arithmetic processors required in different realizations.

Group	Component	Fully parallel	Plane sequential	Group sequential	Fully sequential
L1G1	2 input adders	$4M^2$	4	1	Only one adder for each layer
L1G2	2 input adders	$4M^2$	4	1	
L1G3	adders	Nil	Nil	Nil	
L2G1	M input adders	$4M$	4	1	
L2G2	M input adders	$4M$	4	1	
L2G3	4 input adders	$4M^2(M-1)$	$4(M-1)$	1	
L3G1	M input adders	4	4	1	
L3G2	2 input adders	$8M$	8	1	
L3G3	M input adders	$4M(M-1)$	$4(M-1)$	1	
L4G1	2 input adders	4	4	1	
L4G2	M input adders	$16(M-1)$	8	1	
	Multipliers	$16(M-1)$	8	1	
L4G3	4 input adders	$8(M-1)^2$	$4(M-1)$	1	
	Multipliers	$8(M-1)^2$	$4(M-1)$	1	

4.6.3 Memory Requirement

Table 4.6: The Maximum memory requirement

Layer	Memory
L0	$4M^2$
L1	$12M^2$
L2	$4M(M^2-M+2)$
L3	$4(M^2-M+1)$
L4	$4(2M^2-1)$

The table 4.6 shows the maximum storage requirement in each layer of the model. Different layers, due to the hierarchical structure of the model, can use the same memory. The table shows that the storage requirement is maximum in layer 2. Hence the total memory required will be same as that of Layer 2. Theoretically speaking, the fully parallel implementation of the model does not require any memory area and the fully sequential implementation will require the maximum memory. But due to the hierarchical structure of the model, the memory requirement is not a problem.

4.7 CONCLUSION

In this chapter an investigation on the development of a neural network model is carried out. Presently a parallel distributed model to compute $N \times N$ point DFT, $((N))_4=2$, is completed. It is found that the speed of computation can be improved well. Also there is a scope for having a choice between speed and complexity that can be permitted for a particular application. Being a hierarchical model, it permits simplification in realization.

DISCUSSIONS AND CONCLUSIONS

A new definition of 2-D DFT relation is presented (in chapter 3). This new approach enables DFT computation organized in stages involving only real addition except at the final stage of computation. The number of stages is always fixed at 4. Two different strategies are proposed.

- 1) A visual representation of 2-D DFT coefficients (described in chapter 3)
- 2) A neural network approach (explained in chapter 4).

5.1 VISUAL REPRESENTATION OF 2-D DFT

The visual representation scheme, introduced in chapter 3, can be used to compute, analyze and manipulate 2-D signals such as images in the frequency domain in terms of symbols derived from 2x2 DFT. This, in turn, can be represented in terms of real data. This approach can help analyze signals in the frequency domain even without computing the DFT coefficients.

The various aspects brought forth in the thesis on visual representation are given in the succeeding paragraphs.

5.1.1 Computation/representation

The visual manipulation approach described in chapter 3 is a novel approach. It is mainly intended for signal analysis in the frequency domain in terms of the time domain data. This analysis is shown to be possible through the development of a pictorial representation that relates the DFT coefficients and the data. Since the pictorial representation is data independent and is fixed for a particular order N and index (k_1, k_2) , it can also be used as a look up table to evaluate the coefficients for any index.

5.1.2 Ease of generation

The pictorial representation of many coefficients can be derived from that of a few coefficients by simple manipulations such as circular shift, flipping horizontally / vertically, rotate right / left operations. Also the representation of lower orders are contained in that of higher orders. Hence, the lower order representations can be easily derived from higher orders or vice versa. By using various properties inherent in these representations, their generation can be made simple. Similarly, by developing a semantic grammar, the generation of the pictorial representation can be made easy and user friendly.

5.1.3 Simplicity for users

The visual representation of any information has more advantages in understanding compared to other methods of representation. Software can be developed in which the user can point to the desired frequency index and obtain the visual output. The users can also compute the corresponding value of the DFT coefficient. They can also display the related coefficients and corresponding relations so that the signal analysis will be easy. The same representation can be used for the analysis of other data sets/images. This facility will be extremely useful in image data compression, scene analysis and block coding.

5.1.4 Properties exhibited by the representation

A number of interesting and useful properties exist in the pictorial representation of the DFT coefficients. If the representation is available for a particular frequency index, it is possible to obtain the representation of other coefficients in the same group by operations like

flipping, rotation, permutation over p , negation etc. These properties are very useful in simplifying the representations as well as reducing the memory requirements.

5.1.5 Memory requirement

Similar to any visual representation scheme, this approach is also memory consuming. If the data is of dimension $N \times N$ the DFT will also be of dimension $N \times N$. Therefore, N^2 DFT coefficients are to be represented pictorially and, in general, each coefficient has M components. Thus there are approximately $M \cdot N^2$ pictures that are to be coded for the purpose of storage. Therefore the memory requirement is too high.

However, the complete set of coefficients need not be stored as explained in the following. Many coefficients are repetitions by virtue of permutation over the index p . Some coefficients are related by a specific pattern to a basic coefficient and hence can be easily derived and need not be stored. Also for a (k_1, k_2) the representation for different values of p can be derived by proper circular shift on that of $p=0$. In addition, for $p=0$ and a given index (k_1, k_2) , different row/column of symbols are either identical to one cell or one row/column of cells are circularly rotated version of a row/column of cells. Thus depending on the speed or memory constraint, the storage mechanism can be modified. Also, the representations of lower orders are contained in higher order coefficients. Thus the representations for lower order coefficients can be derived from that of higher orders and vice versa which enables reduction in memory requirement.

5.1.6 Computational complexity

The pictorial representation is order dependent and is not data dependent. Thus, once the pictorial representation for a particular order is developed and stored it need not be developed again for another data. Hence this will be advantageous for block processing of signals with fixed size blocks. Once the pictorial representation is developed and stored, a coefficient can be evaluated by adding the relevant data using the representation in the form of a look up table. Thus the computation of the DFT coefficients using this method can be simplified. The DFT coefficients will be computed in terms of real additions rather than complex operations as used in conventional methods. The complex operations that are involved in this approach are only scaling of the twiddle factors. Once the properties of the DFT coefficients in terms of $Y_{k_1, k_2}^{(p)}$ are studied, the frequency domain analysis of 2-D signals can be carried out without

doing even single complex operation. Thus the computational complexity will be very less in this approach.

5.1.7 Learning

This approach is amenable to supervised learning for the computation of DFT coefficients of any index and of any order limited only by implementation. The visual representation can be used to train a general purpose neural network so that the frequency domain processing of 2-D signals can be made intelligent. Since the pictorial representation for a given order is fixed and it is with a regular pattern for different orders, dependant on the order, is an advantage in deriving the learning algorithm.

5.1.8 Real-time processing

The real time processing of 2-D signals in the frequency domain can be made possible by training a neural network to compute the DFT coefficients based on the visual manipulation approach.

5.2 NEURAL NETWORK MODEL

In chapter 4, a hierarchical neural network model is developed to implement 2-D DFT. Presently, this model is capable of implementing 2-D DFT for a particular order N such that $((N))_4 = 2$. The theoretical studies based on the analysis of the pictorial representation in chapter 3 and the analysis of different neural network models such as Neocognitron and ART models, the following can be carried out to make this model useful in implementing 2-D DFT of any order. Using a similar approach as developed in chapter 4, a model to implementation 2-D DFT can be derived for $((N))_4 = 0$. A generalized model can be derived from these two models to implement the 2-D DFT for any even N . This can be easily extended to odd values of N by padding with zeroes, as in the FFT implementations. Thus the model can be developed into one that can implement the 2-D DFT for any order N upto a set maximum limited by the hardware constraints. The learning algorithm should be modified in such a way that it can implement the 2-D DFT computation of a given data matrix of any order upto the maximum value possible in the model.

The following are some of the features of the new approach described in chapter 4.

5.2.1 Speed / complexity trade off

As explained in chapter 4, the network model can be configured in four different ways depending on the importance of speed or complexity consideration of the application.

1. Fully parallel implementation in which each cell will be operating in parallel so that the time taken for the computation in one layer is same as the time required for the computation in one cell. This realization will have the maximum speed but the complexity will also be high as shown in table 4.5.
2. Plane sequential implementation where the planes are operating in parallel and within a plane the operations are sequential. Since all the cells in a plane are identical and doing the same operation on different data that are circularly shifted in position, the implementation of this will be very easy. This scheme of realization will be less costly compared to fully parallel implementation. Also the complexity will not be increasing as in the parallel implementation when N is increased. But the speed of processing will also be less compared to the parallel implementation.
3. Group sequential implementation in which the groups are operating in parallel and within a group the operations are sequential. This can reduce the complexity, but this is not a good choice due to the difference in complexity of various groups.
4. Fully sequential implementation in which the complete operations within a layer are performed sequentially. This realization will use minimum arithmetic processors as shown in table 4.5. But the speed of processing will also be minimum compared to the other schemes explained above.

Hence, the implementation scheme can be chosen based on the speed and cost requirements. The speed and cost are inversely related. Thus there is flexibility in the implementation of the system.

5.2.2 Computational merits

The model developed in chapter 4 is using a parallel distributed scheme of computation in which the processing is in terms of real additions. Hence the speed of computation will be very high. In this model there are 4 layers of computational units, but the first 3 layers are using only real additions for the computation of DFT. The last layer alone involves complex operation, that is also scalar multiplication of pre-computed twiddle factor values. Thus in this scheme of DFT computation the complex operations are converted

into real additions and they are computed in a parallel and distributed manner. Also, this scheme of computation exploits the reduction of redundancies present in the computation of different coefficients. This makes the computation faster compared to other schemes of computation.

Even though the model developed in chapter 4 is capable of implementing 2-D DFT of $N \times N$ data for $(N)_4 = 2$ only, the studies show that it can be easily extended to any value of N . Also, currently the model is capable of computing the 2-D DFT of a particular order only, but studies are made to train the network to implement any order upto a set maximum. Presently FFT is the most commonly used tool for the DFT computation, and it requires that N should be a power of 2. This is solved in this approach.

5.2.3 Learning

The network model is designed in such a way that the learning time can be minimized using one step learning. The network model can be built for a maximum order determined by the hardware constraints. The network has to find the order N of the data matrix and based on the value of N the network connections are to be reconfigured. The learning is through addition or deletion of connections rather than changing the connection weights. Now, the learning algorithm is not developed but the theoretical studies are carried out for the derivation of the algorithm.

5.2.4 Hardware implementation

The reported method shows a potential in implementing the 2-D DFT in hardware. The basic computational blocks available in the modular form can be repeated at different levels of hierarchy. This leads us to explore the possibility of implementing the DFT computation as a VLSI / ASIC.

5.2.4.1 VLSI implementation aspects

Fig. 4.3 shows the detailed schematic diagram of parallel distributed computation of 6x6 - point DFT shown in fig. 4.2. This model can be implemented in four different ways considering the speed/complexity requirement - 1) Fully parallel - all the cells in a layer are operating in parallel, 2) Plane sequential where the operations in a plane are sequential and the planes are operated in parallel, 3) Group sequential - one cell will be doing the operations

in a group, 4) Fully sequential where one cell does the operations of a layer. The simulations of the algorithm and the theoretical analysis of complexity show that the plane sequential method is a better solution. The connections to one cell in a plane are shown since the others are repetitions of the one shown with a proper shift to the input as given in the algorithm. The cell, in all the planes except that in L4G2 & L4G3, is a simple adder with a selected set of inputs from a particular position in the input plane. All the cells in a plane have the same number of inputs with a regular change in position as in the algorithm. The cell in L4G2 & L4G3 are as shown in fig. 5.1. In the figure, the register $Y_{k_1, k_2}^{(p)}$ is the output from the layer L3 and the Twiddle Factor Register (TFR) is with the twiddle factors, both the registers are of dimension $M = N/2$. PCS (permutation control signal) and FOS (flush out signal) are used to control the data flow. The PCS can be incorporated by using queue registers or LIFO/FIFO organization for the register.

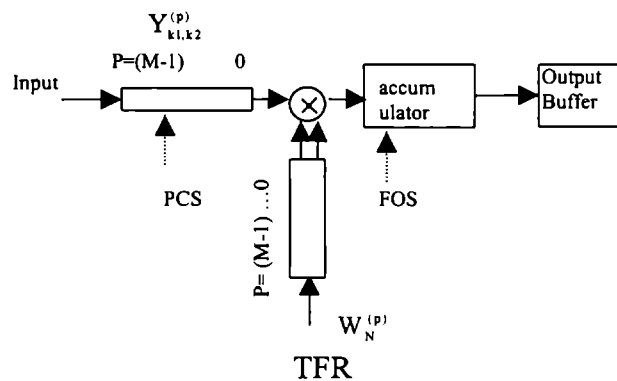


Fig. 5.1 Structure of a cell in L4G2 & L4G3

5.2.4.2 Suggestions to hardware implementation of the Neural Network Model

Since the network model has a hierarchical structure, the computation at one layer will be starting only after finishing that at the previous layer. Thus only one layer will be under use at one time. Hence, only one layer need be implemented if all the layers have the same architecture. In the model shown in Fig. 4.4, there are four layers that are doing the computations. The first three layers are doing only real additions. The last layer L4 alone needs complex operations. The Fig. 5.2 shows the block diagram that can be used for the hardware implementation of the network model for 2-D DFT computation. Fig. 5.3 shows the structure of a neuron model (cell) where the connection weights will be $\pm 1 / 0$ for the first 3 layers and $W_N^P / 0$ in the last layer. The neuron model can operate on complex values similar

to real values since the multiplications are in the form of scaling. Thus the model will be simple and also the complexity will be less. But the complex operations can be completely eliminated once the properties of the DFT coefficients in terms of $Y_{k_1, k_2}^{(p)}$ are explored and hence the model can be made simple.

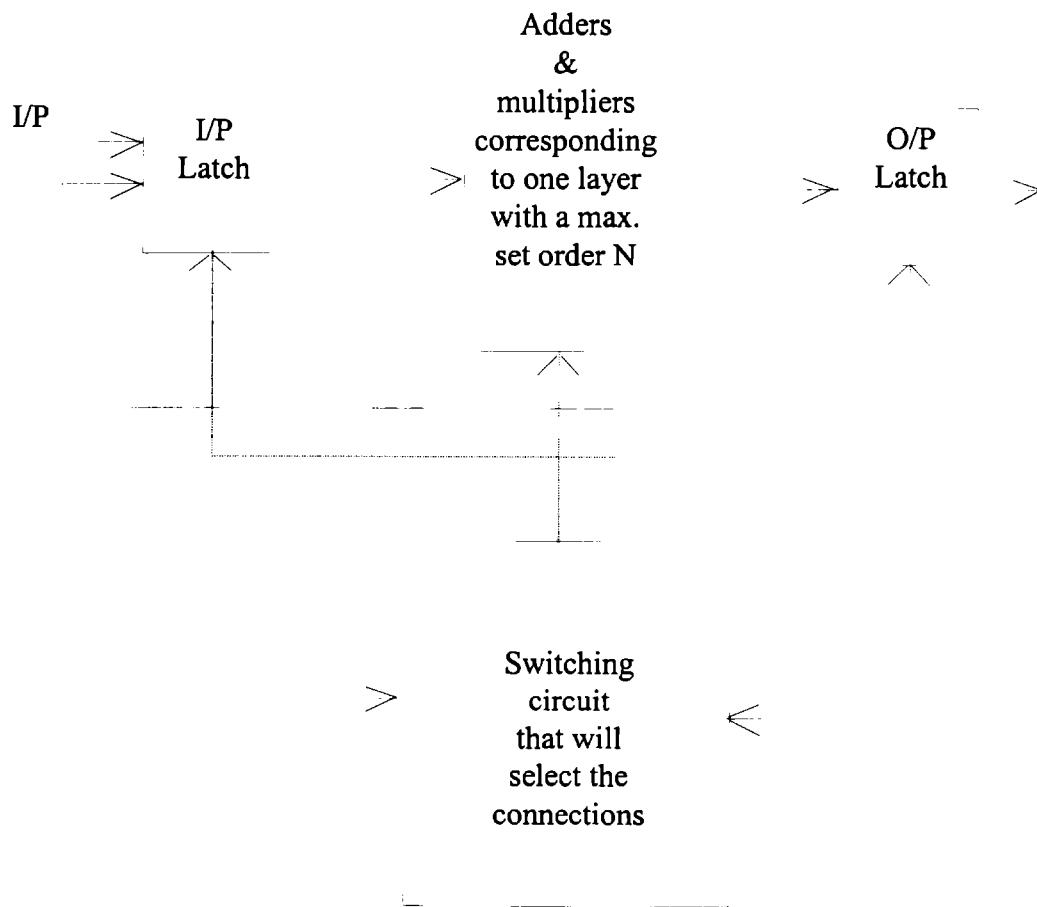


Fig. 5.2 Block diagram for the hardware implementation of the model

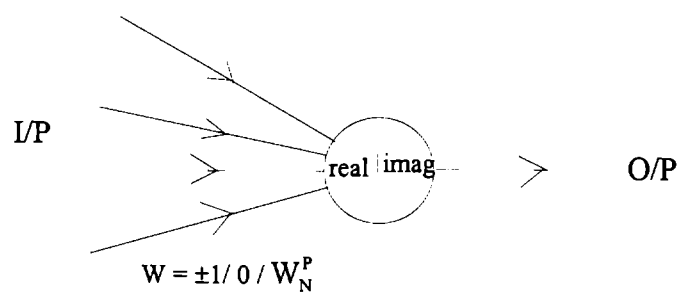


Fig. 5.3 Structure of a cell

5.3 SOME PRACTICAL APPLICATIONS OF THE PRESENT WORK

DFT being a general purpose tool, numerous applications of the present work can be suggested. The frequency domain processing of 2-D signals are limited due to the computational complexity. Some of the problems in image processing that can use DFT are mentioned in chapter 1. Since convolution is an important operation in many signal processing applications and is computationally intensive, the DFT is used. Thus if the IDFT can also be computed using this approach, this method can become a powerful tool in many 2-D applications where convolution is used. In problems like image recognition, scene analysis etc., the processing is done on blocks of data, with block size in powers of 2. In many applications this block size has serious drawbacks and can be solved by using this method of computation.

5.4 SCOPE OF FURTHER WORK IN THE FIELD

The studies on 2-D DFT reported in this thesis, opened two different approaches for the computation as well as signal analysis for 2-D signal processing applications. A number of problems are remaining to be solved in both the approaches.

The following are some of the problems in connection with visual manipulation approach.

1. The visual manipulation approach can be simplified for any order by studying various properties of the representation.
2. A semantic language can be developed to represent the DFT coefficients using the visual representation.
3. Study the properties of signals in the frequency domain in terms of new definition of 2-D DFT (i.e; in terms of $Y_{k_1, k_2}^{(p)}$) so that the complex operations can be eliminated for signal analysis in the frequency domain. Then the frequency domain processing of signals can be carried out in terms of real additions alone. This will simplify the implementation of the hierarchical neural network model also.
4. Train a neural network to use the visual representation in the form of a look up table to analyze 2-D signals in the frequency domain.

5. DFT being a general purpose tool, it can be applied to any 2-D signal processing application. This approach will be highly useful in image processing applications.
6. This approach can be extended to multidimensional DFT also.
7. This approach can be modified to implement inverse DFT also.
8. This approach can be extended to other transforms also.

Also, in connection with neural network model developed in chapter 4, some of the problems are listed below.

1. Modify the neural network model to implement the 2-D DFT of 2-D data of any size.
2. Derive a generalized algorithm for the network model in (1) above.
3. Try to use a hybrid network that uses analog signals for computational purposes and digital signals for control purposes.
4. Hardware implementation, in ASIC, of the neural network model is an important problem since this model is using very simple operations in a highly parallel and distributed manner.
5. This neural network model can be modified to implement multidimensional DFT.
6. This neural network model can be modified to implement many image processing applications such as image segmentation, image compression, motion analysis, image processing using fractals etc.
7. Extend the model to implement inverse DFT.

Thus the work reported here is shown to have sufficient potential to track a large set of problems in 2-D signal processing.

APPENDIX A

A relation for NxN DFT (\mathbf{Y}) can be derived in terms of 2x2 DFTs (\mathbf{X}) of the partitioned data matrix. The relation between the matrices \mathbf{X} & \mathbf{Y} for N=6, referred in section 3.1, is given below.

$$\text{I. 1) } Y_{0,0} = X_{0,0} + X_{0,2} + X_{0,4} + X_{2,0} + X_{2,2} + X_{2,4} + X_{4,0} + X_{4,2} + X_{4,4}$$

$$2) Y_{0,3} = X_{0,1} + X_{0,3} + X_{0,5} + X_{2,1} + X_{2,3} + X_{2,5} + X_{4,1} + X_{4,3} + X_{4,5}$$

$$3) Y_{3,0} = X_{1,0} + X_{1,2} + X_{1,4} + X_{3,0} + X_{3,2} + X_{3,4} + X_{5,0} + X_{5,2} + X_{5,4}$$

$$4) Y_{3,3} = X_{1,1} + X_{1,3} + X_{1,5} + X_{3,1} + X_{3,3} + X_{3,5} + X_{5,1} + X_{5,3} + X_{5,5}$$

$$\text{II. 1) } Y_{1,0} = 0.5 * [(X_{0,0} + X_{0,2} + X_{0,4} + X_{1,0} + X_{1,2} + X_{1,4} + X_{3,0} + X_{3,2} + X_{3,4}) - (X_{2,0} + X_{2,2} + X_{2,4}) + W_6^{-1} [(X_{0,0} + X_{0,2} + X_{0,4}) - (X_{1,0} + X_{1,2} + X_{1,4} + X_{4,0} + X_{4,2} + X_{4,4} + X_{5,0} + X_{5,2} + X_{5,4})] + W_6^2 [(X_{2,0} + X_{2,2} + X_{2,4} + X_{3,0} + X_{3,2} + X_{3,4} + X_{5,0} + X_{5,2} + X_{5,4}) - (X_{4,0} + X_{4,2} + X_{4,4})]]$$

$$2) Y_{2,0} = 0.5 * [(X_{0,0} + X_{0,2} + X_{0,4} + X_{1,0} + X_{1,2} + X_{1,4} + X_{2,0} + X_{2,2} + X_{2,4}) - (X_{3,0} + X_{3,2} + X_{3,4}) + W_6^{-1} [(X_{5,0} + X_{5,2} + X_{5,4}) - (X_{2,0} + X_{2,2} + X_{2,4} + X_{3,0} + X_{3,2} + X_{3,4} + X_{4,0} + X_{4,2} + X_{4,4})] + W_6^2 [(X_{0,0} + X_{0,2} + X_{0,4} + X_{4,0} + X_{4,2} + X_{4,4} + X_{5,0} + X_{5,2} + X_{5,4}) - (X_{1,0} + X_{1,2} + X_{1,4})]]$$

$$3) Y_{4,0} = 0.5 * [[(X_{0,0} + X_{0,2} + X_{0,4} + X_{1,0} + X_{1,2} + X_{1,4} + X_{2,0} + X_{2,2} + X_{2,4}) - (X_{3,0} + X_{3,2} + X_{3,4})] + W_6^{-1}[(X_{1,0} + X_{1,2} + X_{1,4}) - (X_{0,0} + X_{0,2} + X_{0,4} + X_{4,0} + X_{4,2} + X_{4,4} + X_{5,0} + X_{5,2} + X_{5,4})] + W_6^2 [(X_{2,0} + X_{2,2} + X_{2,4} + X_{3,0} + X_{3,2} + X_{3,4} + X_{4,0} + X_{4,2} + X_{4,4}) - (X_{5,0} + X_{5,2} + X_{5,4})]$$

$$4) Y_{5,0} = 0.5 * [[(X_{0,0} + X_{0,2} + X_{0,4} + X_{1,0} + X_{1,2} + X_{1,4} + X_{3,0} + X_{3,2} + X_{3,4}) - (X_{2,0} + X_{2,2} + X_{2,4}) + W_6^{-1}[(X_{4,0} + X_{4,2} + X_{4,4}) - (X_{2,0} + X_{2,2} + X_{2,4} + X_{3,0} + X_{3,2} + X_{3,4} + X_{5,0} + X_{5,2} + X_{5,4})] + W_6^2 [(X_{1,0} + X_{1,2} + X_{1,4} + X_{4,0} + X_{4,2} + X_{4,4} + X_{5,0} + X_{5,2} + X_{5,4}) - (X_{0,0} + X_{0,2} + X_{0,4})]]]$$

Similarly the coefficients $Y_{0,k}, Y_{k,3}, Y_{3,k}$, where $k=1,2,4,5$, can be expressed from the above expressions $Y_{k,0}$ by using $X_{j,i}, X_{i,j+1}, X_{i+1,j}$ respectively instead of $X_{i,j}$. From these equations,

$Y_{k1,k2}$ can be expressed as $Y_{k1,k2} = \sum_{p=0}^{M-1} Y_{k1,k2}^{(p)} \cdot W_6^p$ where $W_N = e^{-j2\pi/N}$, $M = N/2$. In the

following expressions $Y_{k1,k2}^{(p)}$ are expressed as $Y_{k1,k2}(p)$.

$$1) Y_{1,1}(0) = 0.25 * [(X_{0,0} + X_{0,1} + X_{1,0} + X_{1,1}) + (X_{2,4} + X_{2,5} + X_{3,4} + X_{3,5}) + (X_{4,2} + X_{4,3} + X_{5,3} + X_{5,2}) + (X_{0,4} + X_{1,5} + X_{4,0} + X_{5,1} + X_{2,2} + X_{3,3}) - (X_{1,4} + X_{0,5} + X_{4,1} + X_{5,0} + X_{2,3} + X_{3,2})] + 0.5 * [(X_{1,3} + X_{3,1} + X_{5,5}) - (X_{0,2} + X_{2,0} + X_{4,4})]$$

$$Y_{1,1}(1) = 0.5 * [(X_{0,0} + X_{2,4} + X_{4,2}) - (X_{1,1} + X_{3,5} + X_{5,3})] + 0.25 * [(X_{0,3} + X_{1,2} + X_{4,5} + X_{5,4} + X_{2,1} + X_{3,0}) - ((X_{0,4} + X_{0,5} + X_{1,4} + X_{1,5}) + (X_{2,2} + X_{2,3} + X_{3,2} + X_{3,3}) + (X_{4,0} + X_{4,1} + X_{5,0} + X_{5,1}) + (X_{0,2} + X_{1,3} + X_{4,4} + X_{5,5} + X_{2,0} + X_{3,1}))]$$

$$Y_{1,1}(2) = 0.5 * [(X_{1,5} + X_{3,3} + X_{5,1}) - (X_{0,4} + X_{4,0} + X_{2,2})] + 0.25 * [(X_{0,0} + X_{1,1} + X_{4,2} + X_{5,3} + X_{2,4} + X_{3,5}) + (X_{0,2} + X_{0,3} + X_{1,2} + X_{1,3}) + (X_{2,0} + X_{2,1} + X_{3,0} + X_{3,1}) + (X_{4,4} + X_{4,5} + X_{5,4} + X_{5,5}) - (X_{0,1} + X_{1,0} + X_{4,3} + X_{5,2} + X_{2,5} + X_{3,4}))]$$

$$2) Y_{2,2}(0) = 0.25 * [(X_{0,0} + X_{0,1} + X_{1,0} + X_{1,1}) + (X_{2,4} + X_{2,5} + X_{3,4} + X_{3,5}) + (X_{4,2} + X_{4,3} + X_{5,3} + X_{5,2}) + (X_{0,4} + X_{1,5} + X_{4,0} + X_{5,1} + X_{2,2} + X_{3,3}) - (X_{1,4} + X_{0,5} + X_{4,1} + X_{5,0} + X_{2,3} + X_{3,2})] + 0.5 * [-(X_{1,3} + X_{3,1} + X_{5,5}) + (X_{0,2} + X_{2,0} + X_{4,4})]$$

$$Y_{2,2}(1) = 0.5 * [(X_{1,5} + X_{3,3} + X_{5,1}) - (X_{0,4} + X_{4,0} + X_{2,2})] + 0.25 * [-(X_{0,0} + X_{1,1} + X_{4,2} + X_{5,3} + X_{2,4} + X_{3,5}) + (X_{0,2} + X_{0,3} + X_{1,2} + X_{1,3}) + (X_{2,0} + X_{2,1} + X_{3,0} + X_{3,1}) + (X_{4,4} + X_{4,5} + X_{5,4} + X_{5,5})] + (X_{0,1} + X_{1,0} + X_{4,3} + X_{5,2} + X_{2,5} + X_{3,4}))]$$

$$Y_{2,2}(2) = 0.5 * [(X_{0,0} + X_{2,4} + X_{4,2}) - (X_{1,1} + X_{3,5} + X_{5,3})] + 0.25 * [-(X_{0,3} + X_{1,2} + X_{4,5} + X_{5,4} + X_{2,1} + X_{3,0}) + (X_{0,4} + X_{0,5} + X_{1,4} + X_{1,5}) + (X_{2,2} + X_{2,3} + X_{3,2} + X_{3,3}) + (X_{4,0} + X_{4,1} + X_{5,0} + X_{5,1}) + (X_{0,2} + X_{1,3} + X_{4,4} + X_{5,5} + X_{2,0} + X_{3,1})]$$

$$3) Y_{4,4}(0) = Y_{2,2}(0), Y_{4,4}(1) = -Y_{2,2}(2), Y_{4,4}(2) = -Y_{2,2}(1)$$

$$4) Y_{5,5}(0) = Y_{1,1}(0), Y_{5,5}(1) = -Y_{1,1}(2), Y_{5,5}(2) = -Y_{1,1}(1)$$

$$5) Y_{4,1}(0) = 0.25 * [(X_{0,0} + X_{0,1} + X_{1,0} + X_{1,1}) + (X_{2,4} + X_{2,5} + X_{3,4} + X_{3,5}) + (X_{4,2} + X_{4,3} + X_{5,3} + X_{5,2}) + (X_{0,5} + X_{1,4} + X_{4,1} + X_{5,0} + X_{2,3} + X_{3,2}) - (X_{0,4} + X_{1,5} + X_{4,0} + X_{5,1} + X_{2,2} + X_{3,3})] + 0.5 * [(X_{0,3} + X_{2,1} + X_{4,5}) - (X_{1,2} + X_{3,0} + X_{5,4})]$$

$$Y_{4,1}(1) = 0.5 * [(X_{1,0} + X_{3,4} + X_{5,2}) - (X_{0,1} + X_{4,3} + X_{2,5})] + 0.25 * [-(X_{0,3} + X_{1,2} + X_{4,5} + X_{5,4} + X_{2,1} + X_{3,0}) + (X_{0,4} + X_{0,5} + X_{1,4} + X_{1,5}) + (X_{2,2} + X_{2,3} + X_{3,2} + X_{3,3}) + (X_{4,0} + X_{4,1} + X_{5,0} + X_{5,1}) + (X_{0,2} + X_{1,3} + X_{4,4} + X_{5,5} + X_{2,0} + X_{3,1})]$$

$$Y_{4,1}(2) = 0.5 * [(X_{0,5} + X_{2,3} + X_{4,1}) - (X_{1,4} + X_{3,2} + X_{5,0})] + 0.25 * [-(X_{0,0} + X_{1,1} + X_{4,2} + X_{5,3} + X_{2,4} + X_{3,5}) + (X_{0,2} + X_{0,3} + X_{1,2} + X_{1,3}) + (X_{2,0} + X_{2,1} + X_{3,0} + X_{3,1}) + (X_{4,4} + X_{4,5} + X_{5,4} + X_{5,5}) + (X_{0,1} + X_{1,0} + X_{4,3} + X_{5,2} + X_{2,5} + X_{3,4})]$$

$$6) Y_{5,2}(0) = 0.25 * [(X_{0,0} + X_{0,1} + X_{1,0} + X_{1,1}) + (X_{2,4} + X_{2,5} + X_{3,4} + X_{3,5}) + (X_{4,2} + X_{4,3} + X_{5,3} + X_{5,2}) + (X_{0,5} + X_{1,4} + X_{4,1} + X_{5,0} + X_{2,3} + X_{3,2}) - (X_{0,4} + X_{1,5} + X_{4,0} + X_{5,1} + X_{2,2} + X_{3,3})] + 0.5 * [(X_{1,2} + X_{3,0} + X_{5,4}) - (X_{0,3} + X_{2,1} + X_{4,5})]$$

$$Y_{5,2}(1) = 0.5 * [(X_{0,5} + X_{2,3} + X_{4,1}) - (X_{1,4} + X_{3,2} + X_{5,0})] + 0.25 * [(X_{0,0} + X_{1,1} + X_{4,2} + X_{5,3} + X_{2,4} + X_{3,5}) - (X_{0,2} + X_{0,3} + X_{1,2} + X_{1,3}) + (X_{2,0} + X_{2,1} + X_{3,0} + X_{3,1}) + (X_{4,4} + X_{4,5} + X_{5,4} + X_{5,5}) + (X_{0,1} + X_{1,0} + X_{4,3} + X_{5,2} + X_{2,5} + X_{3,4})]$$

$$Y_{5,2}(2) = 0.5 * [(X_{1,0} + X_{3,4} + X_{5,2}) - (X_{0,1} + X_{4,3} + X_{2,5})] + 0.25 * [(X_{0,3} + X_{1,2} + X_{4,5} + X_{5,4} + X_{2,1} + X_{3,0}) + (X_{0,4} + X_{0,5} + X_{1,4} + X_{1,5}) + (X_{2,2} + X_{2,3} + X_{3,2} + X_{3,3}) + (X_{4,0} + X_{4,1} + X_{5,0} + X_{5,1}) + (X_{0,2} + X_{1,3} + X_{4,4} + X_{5,5} + X_{2,0} + X_{3,1})]$$

$$7) Y_{5,1}(0) = 0.5 * [X_{0,0} + X_{1,1} + X_{2,2} + X_{3,3} + X_{4,4} + X_{5,5}] + 0.25 * [(X_{0,3} + X_{1,3} + X_{1,4} + X_{1,5} + X_{2,5} + X_{3,5} + X_{3,0} + X_{3,1} + X_{4,1} + X_{5,1} + X_{5,2} + X_{5,3}) - (X_{0,2} + X_{1,2} + X_{0,4} + X_{0,5} + X_{2,4} + X_{3,4} + X_{2,0} + X_{2,1} + X_{4,0} + X_{5,0} + X_{4,2} + X_{4,3})]$$

$$Y_{5,1}(1) = 0.25 * [(X_{0,0} + X_{1,0} + X_{0,2} + X_{0,3} + X_{2,2} + X_{3,2} + X_{2,4} + X_{2,5} + X_{4,4} + X_{5,4} + X_{4,0} + X_{4,1}) - (X_{0,1} + X_{1,1} + X_{1,2} + X_{1,3} + X_{2,3} + X_{3,3} + X_{3,4} + X_{3,5} + X_{4,5} + X_{5,5} + X_{5,0} + X_{5,1})] - 0.5 * [X_{0,4} + X_{1,5} + X_{2,0} + X_{3,1} + X_{4,2} + X_{5,3}]$$

$$Y_{5,1}(2) = 0.5 * [X_{0,2} + X_{1,3} + X_{2,4} + X_{3,5} + X_{4,0} + X_{5,1}] + 0.25 * [(X_{0,5} + X_{1,5} + X_{1,0} + X_{1,1} + X_{2,1} + X_{3,1} + X_{3,2} + X_{3,3} + X_{4,3} + X_{5,3} + X_{5,4} + X_{5,5}) - (X_{0,4} + X_{1,4} + X_{0,0} + X_{0,1} + X_{2,0} + X_{3,0} + X_{2,2} + X_{2,3} + X_{4,2} + X_{5,2} + X_{4,4} + X_{4,5})]$$

$$8) \quad Y_{4,2}(0) = 0.5 * [X_{0,0} + X_{1,1} + X_{2,2} + X_{3,3} + X_{4,4} + X_{5,5}] + 0.25 * [-(X_{0,3} + X_{1,3} + X_{1,4} + X_{1,5} + X_{2,5} + X_{3,5} + X_{3,0} + X_{3,1} + X_{4,1} + X_{5,1} + X_{5,2} + X_{5,3}) + (X_{0,2} + X_{1,2} + X_{0,4} + X_{0,5} + X_{2,4} + X_{3,4} + X_{2,0} + X_{2,1} + X_{4,0} + X_{5,0} + X_{4,2} + X_{4,3})]$$

$$Y_{4,2}(1) = 0.25 * [(X_{0,5} + X_{1,5} + X_{1,0} + X_{1,1} + X_{2,1} + X_{3,1} + X_{3,2} + X_{3,3} + X_{4,3} + X_{5,3} + X_{5,4} + X_{5,5}) - (X_{0,4} + X_{1,4} + X_{0,0} + X_{0,1} + X_{2,0} + X_{3,0} + X_{2,2} + X_{2,3} + X_{4,2} + X_{5,2} + X_{4,4} + X_{4,5})] - 0.5 * [X_{0,2} + X_{1,3} + X_{2,4} + X_{3,5} + X_{4,0} + X_{5,1}]$$

$$Y_{4,2}(2) = 0.5 * [X_{0,4} + X_{1,5} + X_{2,0} + X_{3,1} + X_{4,2} + X_{5,3}] + 0.25 * [(X_{0,0} + X_{1,0} + X_{0,2} + X_{0,3} + X_{2,2} + X_{3,2} + X_{2,4} + X_{2,5} + X_{4,4} + X_{5,4} + X_{4,0} + X_{4,1}) - (X_{0,1} + X_{1,1} + X_{1,2} + X_{1,3} + X_{2,3} + X_{3,3} + X_{3,4} + X_{3,5} + X_{4,5} + X_{5,5} + X_{5,0} + X_{5,1})]$$

$$9) \quad Y_{2,1}(0) = 0.5 * [X_{0,1} + X_{1,0} + X_{2,3} + X_{3,2} + X_{4,5} + X_{5,4}] + 0.25 * [(X_{0,4} + X_{0,5} + X_{0,3} + X_{1,3} + X_{2,5} + X_{3,5} + X_{2,0} + X_{2,1} + X_{4,1} + X_{5,1} + X_{4,2} + X_{4,3}) - (X_{0,2} + X_{1,2} + X_{1,4} + X_{1,5} + X_{2,4} + X_{3,4} + X_{3,0} + X_{3,1} + X_{4,0} + X_{5,0} + X_{5,2} + X_{5,3})]$$

$$Y_{2,1}(1) = 0.25 * [(X_{0,0} + X_{1,0} + X_{1,2} + X_{1,3} + X_{2,2} + X_{3,2} + X_{3,4} + X_{3,5} + X_{4,4} + X_{5,4} + X_{5,0} + X_{5,1}) - (X_{0,1} + X_{1,1} + X_{0,2} + X_{0,3} + X_{2,3} + X_{3,3} + X_{2,4} + X_{2,5} + X_{4,5} + X_{5,5} + X_{4,0} + X_{4,1})] - 0.5 * [X_{1,4} + X_{0,5} + X_{2,1} + X_{3,0} + X_{4,3} + X_{5,2}]$$

$$Y_{2,1}(2) = 0.5 * [X_{0,3} + X_{1,2} + X_{2,5} + X_{3,4} + X_{4,1} + X_{5,0}] + 0.25 * [(X_{0,5} + X_{1,5} + X_{0,0} + X_{0,1} + X_{2,1} + X_{3,1} + X_{2,2} + X_{2,3} + X_{4,3} + X_{5,3} + X_{4,4} + X_{4,5}) - (X_{0,4} + X_{1,4} + X_{1,0} + X_{1,1} + X_{2,0} + X_{3,0} + X_{3,2} + X_{3,3} + X_{4,2} + X_{5,2} + X_{5,4} + X_{5,5})]$$

$$10) \quad Y_{1,2}(0) = 0.5 * [X_{0,1} + X_{1,0} + X_{2,3} + X_{3,2} + X_{4,5} + X_{5,4}] + 0.25 * [-(X_{0,4} + X_{0,5} + X_{0,3} + X_{1,3} + X_{2,5} + X_{3,5} + X_{2,0} + X_{2,1} + X_{4,1} + X_{5,1} + X_{4,2} + X_{4,3}) + (X_{0,2} + X_{1,2} + X_{1,4} + X_{1,5} + X_{2,4} + X_{3,4} + X_{3,0} + X_{3,1} + X_{4,0} + X_{5,0} + X_{5,2} + X_{5,3})]$$

$$Y_{1,2}(1) = 0.25 * [-(X_{0,0} + X_{1,0} + X_{1,2} + X_{1,3} + X_{2,2} + X_{3,2} + X_{3,4} + X_{3,5} + X_{4,4} + X_{5,4} + X_{5,0} + X_{5,1}) + (X_{0,1} + X_{1,1} + X_{0,2} + X_{0,3} + X_{2,3} + X_{3,3} + X_{2,4} + X_{2,5} + X_{4,5} + X_{5,5} + X_{4,0} + X_{4,1})] - 0.5 * [X_{1,4} + X_{0,5} + X_{2,1} + X_{3,0} + X_{4,3} + X_{5,2}]$$

$$Y_{1,2}(2) = 0.5 * [X_{0,3} + X_{1,2} + X_{2,5} + X_{3,4} + X_{4,1} + X_{5,0}] + 0.25 * [-(X_{0,5} + X_{1,5} + X_{0,0} + X_{0,1} + X_{2,1} + X_{3,1} + X_{2,2} + X_{2,3} + X_{4,3} + X_{5,3} + X_{4,4} + X_{4,5}) + (X_{0,4} + X_{1,4} + X_{1,0} + X_{1,1} + X_{2,0} + X_{3,0} + X_{3,2} + X_{3,3} + X_{4,2} + X_{5,2} + X_{5,4} + X_{5,5})]$$

$$11) \quad Y_{5,4}(0) = 0.5 * [X_{0,1} + X_{1,0} + X_{2,3} + X_{3,2} + X_{4,5} + X_{5,4}] + 0.25 * [- (X_{0,4} + X_{0,5} + X_{0,3} + X_{1,3} + X_{2,5} + X_{3,5} + X_{2,0} + X_{2,1} + X_{4,1} + X_{5,1} + X_{4,2} + X_{4,3}) + (X_{0,2} + X_{1,2} + X_{1,4} + X_{1,5} + X_{2,4} + X_{3,4} + X_{3,0} + X_{3,1} + X_{4,0} + X_{5,0} + X_{5,2} + X_{5,3})]$$

$$Y_{5,4}(1) = 0.25 * [- (X_{0,0} + X_{1,0} + X_{1,2} + X_{1,3} + X_{2,2} + X_{3,2} + X_{3,4} + X_{3,5} + X_{4,4} + X_{5,4} + X_{5,0} + X_{5,1}) + (X_{0,1} + X_{1,1} + X_{0,2} + X_{0,3} + X_{2,3} + X_{3,3} + X_{2,4} + X_{2,5} + X_{4,5} + X_{5,5} + X_{4,0} + X_{4,1})] - 0.5 * [X_{1,4} + X_{0,5} + X_{2,1} + X_{3,0} + X_{4,3} + X_{5,2}]$$

$$Y_{5,4}(2) = 0.5 * [X_{0,3} + X_{1,2} + X_{2,5} + X_{3,4} + X_{4,1} + X_{5,0}] + 0.25 * [- (X_{0,5} + X_{1,5} + X_{0,0} + X_{0,1} + X_{2,1} + X_{3,1} + X_{2,2} + X_{2,3} + X_{4,3} + X_{5,3} + X_{4,4} + X_{4,5}) + (X_{0,4} + X_{1,4} + X_{1,0} + X_{1,1} + X_{2,0} + X_{3,0} + X_{3,2} + X_{3,3} + X_{4,2} + X_{5,2} + X_{5,4} + X_{5,5})]$$

$$12) \quad Y_{4,5}(0) = 0.5 * [X_{0,1} + X_{1,0} + X_{2,3} + X_{3,2} + X_{4,5} + X_{5,4}] + 0.25 * [(X_{0,4} + X_{0,5} + X_{0,3} + X_{1,3} + X_{2,5} + X_{3,5} + X_{2,0} + X_{2,1} + X_{4,1} + X_{5,1} + X_{4,2} + X_{4,3}) - (X_{0,2} + X_{1,2} + X_{1,4} + X_{1,5} + X_{2,4} + X_{3,4} + X_{3,0} + X_{3,1} + X_{4,0} + X_{5,0} + X_{5,2} + X_{5,3})]$$

$$Y_{5,4}(1) = 0.25 * [(X_{0,0} + X_{1,0} + X_{1,2} + X_{1,3} + X_{2,2} + X_{3,2} + X_{3,4} + X_{3,5} + X_{4,4} + X_{5,4} + X_{5,0} + X_{5,1}) - (X_{0,1} + X_{1,1} + X_{0,2} + X_{0,3} + X_{2,3} + X_{3,3} + X_{2,4} + X_{2,5} + X_{4,5} + X_{5,5} + X_{4,0} + X_{4,1})] - 0.5 * [X_{1,4} + X_{0,5} + X_{2,1} + X_{3,0} + X_{4,3} + X_{5,2}]$$

$$Y_{5,4}(2) = 0.5 * [X_{0,3} + X_{1,2} + X_{2,5} + X_{3,4} + X_{4,1} + X_{5,0}] + 0.25 * [(X_{0,5} + X_{1,5} + X_{0,0} + X_{0,1} + X_{2,1} + X_{3,1} + X_{2,2} + X_{2,3} + X_{4,3} + X_{5,3} + X_{4,4} + X_{4,5}) - (X_{0,4} + X_{1,4} + X_{1,0} + X_{1,1} + X_{2,0} + X_{3,0} + X_{3,2} + X_{3,3} + X_{4,2} + X_{5,2} + X_{5,4} + X_{5,5})]$$

$$13) \quad Y_{2,5}(0) = 0.25 * [(X_{0,0} + X_{0,1} + X_{1,0} + X_{1,1}) + (X_{2,4} + X_{2,5} + X_{3,4} + X_{3,5}) + (X_{4,2} + X_{4,3} + X_{5,3} + X_{5,2}) + (X_{0,5} + X_{1,4} + X_{4,1} + X_{5,0} + X_{2,3} + X_{3,2}) - (X_{0,4} + X_{1,5} + X_{4,0} + X_{5,1} + X_{2,2} + X_{3,3})] + 0.5 * [- (X_{1,2} + X_{3,0} + X_{5,4}) + (X_{0,3} + X_{2,1} + X_{4,5})]$$

$$Y_{2,5}(1) = 0.5 * [- (X_{0,5} + X_{2,3} + X_{4,1}) + (X_{1,4} + X_{3,2} + X_{5,0})] + 0.25 * [(X_{0,0} + X_{1,1} + X_{4,2} + X_{5,3} + X_{2,4} + X_{3,5}) - [(X_{0,2} + X_{0,3} + X_{1,2} + X_{1,3}) + (X_{2,0} + X_{2,1} + X_{3,0} + X_{3,1}) + (X_{4,4} + X_{4,5} + X_{5,4} + X_{5,5})] + (X_{0,1} + X_{1,0} + X_{4,3} + X_{5,2} + X_{2,5} + X_{3,4})]$$

$$Y_{2,5}(2) = 0.5 * [- (X_{1,0} + X_{3,4} + X_{5,2}) + (X_{0,1} + X_{4,3} + X_{2,5})] + 0.25 * [(X_{0,3} + X_{1,2} + X_{4,5} + X_{5,4} + X_{2,1} + X_{3,0}) + (X_{0,4} + X_{0,5} + X_{1,4} + X_{1,5}) + (X_{2,2} + X_{2,3} + X_{3,2} + X_{3,3}) + (X_{4,0} + X_{4,1} + X_{5,0} + X_{5,1}) - (X_{0,2} + X_{1,3} + X_{4,4} + X_{5,5} + X_{2,0} + X_{3,1})]$$

$$14) \quad Y_{1,4}(0) = 0.25 * [(X_{0,0} + X_{0,1} + X_{1,0} + X_{1,1}) + (X_{2,4} + X_{2,5} + X_{3,4} + X_{3,5}) + (X_{4,2} + X_{4,3} + X_{5,3} + X_{5,2}) + (X_{0,5} + X_{1,4} + X_{4,1} + X_{5,0} + X_{2,3} + X_{3,2}) - (X_{0,4} + X_{1,5} + X_{4,0} + X_{5,1} + X_{2,2} + X_{3,3})] + 0.5 * [- (X_{0,3} + X_{2,1} + X_{4,5}) + (X_{1,2} + X_{3,0} + X_{5,4})]$$

$$Y_{1,4}(1) = 0.5 * [- (X_{1,0} + X_{3,4} + X_{5,2}) + (X_{0,1} + X_{4,3} + X_{2,5})] + 0.25 * [- ((X_{0,3} + X_{1,2} + X_{4,5} + X_{5,4} + X_{2,1} + X_{3,0}) + (X_{0,4} + X_{0,5} + X_{1,4} + X_{1,5}) + (X_{2,2} + X_{2,3} + X_{3,2} + X_{3,3}) + (X_{4,0} + X_{4,1} + X_{5,0} + X_{5,1})) + (X_{0,2} + X_{1,3} + X_{4,4} + X_{5,5} + X_{2,0} + X_{3,1})]$$

$$Y_{1,4}(2) = 0.5 * [- (X_{0,5} + X_{2,3} + X_{4,1}) + (X_{1,4} + X_{3,2} + X_{5,0})] + 0.25 * [- ((X_{0,0} + X_{1,1} + X_{4,2} + X_{5,3} + X_{2,4} + X_{3,5}) + (X_{0,2} + X_{0,3} + X_{1,2} + X_{1,3}) + (X_{2,0} + X_{2,1} + X_{3,0} + X_{3,1}) + (X_{4,4} + X_{4,5} + X_{5,4} + X_{5,5})) + (X_{0,1} + X_{1,0} + X_{4,3} + X_{5,2} + X_{2,5} + X_{3,4})]$$

$$15) \quad Y_{1,5}(0) = Y_{5,1}(0), Y_{1,5}(1) = -Y_{5,1}(2), Y_{1,5}(2) = -Y_{5,1}(1)$$

$$16) \quad Y_{2,4}(0) = Y_{4,2}(0), Y_{2,4}(1) = -Y_{4,2}(2), Y_{2,4}(2) = -Y_{4,2}(1),$$

These equations were the basis for the new definition of the DFT relation and the pictorial representation.

APPENDIX B

The primitive symbol combination at the first row of cells in the pictorial representation of $Y_{k_1, k_2}^{(0)}$ in group 3 for $N=10, 14$ & 18 are given below (for L2G3) as referred in section 4.4. The frequency index (k_1, k_2) of the basic coefficient is also given.

N = 10

- 1) $DP_{0,0} + MPL_{0,2} + MPA_{0,3}$ (8,2)
- 2) $DP_{0,0} + MPR_{0,2} + MPB_{0,3}$ (9,1)
- 3) $CP_{0,0} + MPR_{0,2} + MPA_{0,3}$ (4,1)
- 4) $CP_{0,0} + MPL_{0,2} + MPB_{0,3}$ (3,2)
- 5) $MP_{0,0} + DPA_{0,2} + MPD_{0,4}$ (2,2)
- 6) $MP_{0,0} + DPB_{0,2} + MPD_{0,4}$ (1,1)
- 7) $MP_{0,0} + CPA_{0,2} + MPC_{0,4}$ (6,1)
- 8) $MP_{0,0} + CPB_{0,2} + MPC_{0,4}$ (7,2)

- 9) $MP_{0,0} + MPC_{0,1} + MPR_{0,2} + MPA_{0,4}$ (2,1)
- 10) $MP_{0,0} + MPD_{0,1} + MPR_{0,2} + MPB_{0,4}$ (7,1)
- 11) $MP_{0,0} + MPD_{0,1} + MPL_{0,2} + MPA_{0,4}$ (4,2)
- 12) $MP_{0,0} + MPC_{0,1} + MPL_{0,2} + MPB_{0,4}$ (9,2)
- 13) $MP_{0,0} + MPA_{0,1} + MPR_{0,2} + MPC_{0,3}$ (8,1)
- 14) $MP_{0,0} + MPB_{0,1} + MPL_{0,2} + MPC_{0,3}$ (1,2)
- 15) $MP_{0,0} + MPB_{0,1} + MPR_{0,2} + MPD_{0,3}$ (3,1)
- 16) $MP_{0,0} + MPA_{0,1} + MPL_{0,2} + MPD_{0,3}$ (6,2)

N=14

- 1) $DP_{0,0} + MPL_{0,3} + MPA_{0,4}$ 12,2
- 2) $DP_{0,0} + MPR_{0,3} + MPB_{0,4}$ 13,1
- 3) $CP_{0,0} + MPR_{0,3} + MPA_{0,4}$ 6,1
- 4) $CP_{0,0} + MPL_{0,3} + MPB_{0,4}$ 5,2
- 5) $MP_{0,0} + DPA_{0,3} + MPD_{0,6}$ 2,2
- 6) $MP_{0,0} + DPB_{0,3} + MPD_{0,6}$ 1,1
- 7) $MP_{0,0} + CPA_{0,3} + MPC_{0,6}$ 8,1
- 8) $MP_{0,0} + CPB_{0,3} + MPC_{0,6}$ 9,2
- 9) $MP_{0,0} + MPC_{0,2} + MPR_{0,3} + MPA_{0,6}$ 2,1
- 10) $MP_{0,0} + MPD_{0,2} + MPR_{0,3} + MPB_{0,6}$ 9,1
- 11) $MP_{0,0} + MPD_{0,2} + MPL_{0,3} + MPA_{0,6}$ 4,2
- 12) $MP_{0,0} + MPC_{0,2} + MPL_{0,3} + MPB_{0,6}$ 11,2
- 13) $MP_{0,0} + MPA_{0,1} + MPR_{0,3} + MPC_{0,4}$ 12,1
- 14) $MP_{0,0} + MPB_{0,1} + MPL_{0,3} + MPC_{0,4}$ 3,2

15)	$MP_{0,0} + MPB_{0,1} + MPR_{0,3} + MPD_{0,4}$	5,1
16)	$MP_{0,0} + MPA_{0,1} + MPL_{0,3} + MPD_{0,4}$	10,2
17)	$MP_{0,0} + MPA_{0,2} + MPR_{0,3} + MPC_{0,5}$	10,1
18)	$MP_{0,0} + MPB_{0,2} + MPR_{0,3} + MPD_{0,5}$	3,1
19)	$MP_{0,0} + MPA_{0,2} + MPL_{0,3} + MPD_{0,5}$	6,2
20)	$MP_{0,0} + MPB_{0,2} + MPL_{0,3} + MPC_{0,5}$	13,2
21)	$MP_{0,0} + MPD_{0,1} + MPR_{0,3} + MPB_{0,5}$	11,1
22)	$MP_{0,0} + MPC_{0,1} + MPR_{0,3} + MPA_{0,5}$	4,1
23)	$MP_{0,0} + MPC_{0,1} + MPL_{0,3} + MPB_{0,5}$	1,2
24)	$MP_{0,0} + MPD_{0,1} + MPL_{0,3} + MPA_{0,5}$	8,2

N=18

k1,k2

1)	$DP_{0,0} + MPL_{0,4} + MPA_{0,5}$	16,2
2)	$DP_{0,0} + MPR_{0,4} + MPB_{0,5}$	17,1
3)	$CP_{0,0} + MPR_{0,4} + MPA_{0,5}$	8,1
4)	$CP_{0,0} + MPL_{0,4} + MPB_{0,5}$	7,2
5)	$MP_{0,0} + DPA_{0,4} + MPD_{0,8}$	2,2
6)	$MP_{0,0} + DPB_{0,4} + MPD_{0,8}$	1,1
7)	$MP_{0,0} + CPA_{0,4} + MPC_{0,8}$	10,1
8)	$MP_{0,0} + CPB_{0,4} + MPC_{0,8}$	11,2
9)	$MP_{0,0} + MPC_{0,3} + MPR_{0,4} + MPA_{0,8}$	2,1
10)	$MP_{0,0} + MPD_{0,3} + MPR_{0,4} + MPB_{0,8}$	11,1
11)	$MP_{0,0} + MPD_{0,3} + MPL_{0,4} + MPA_{0,8}$	4,2
12)	$MP_{0,0} + MPC_{0,3} + MPL_{0,4} + MPB_{0,8}$	13,2

13)	$MP_{0,0} + MPA_{0,3} + MPR_{0,4} + MPC_{0,7}$	12,1
14)	$MP_{0,0} + MPB_{0,3} + MPL_{0,4} + MPC_{0,7}$	15,2
15)	$MP_{0,0} + MPB_{0,3} + MPR_{0,4} + MPD_{0,7}$	3,1
16)	$MP_{0,0} + MPA_{0,3} + MPL_{0,4} + MPD_{0,7}$	6,2
17)	$MP_{0,0} + MPA_{0,2} + MPR_{0,4} + MPC_{0,6}$	14,1
18)	$MP_{0,0} + MPB_{0,2} + MPR_{0,4} + MPD_{0,6}$	5,1
19)	$MP_{0,0} + MPA_{0,2} + MPL_{0,4} + MPD_{0,6}$	10,2
20)	$MP_{0,0} + MPB_{0,2} + MPL_{0,4} + MPC_{0,6}$	1,2
21)	$MP_{0,0} + MPD_{0,2} + MPR_{0,4} + MPB_{0,7}$	13,1
22)	$MP_{0,0} + MPC_{0,2} + MPR_{0,4} + MPA_{0,7}$	4,1
23)	$MP_{0,0} + MPC_{0,2} + MPL_{0,4} + MPB_{0,7}$	17,2
24)	$MP_{0,0} + MPD_{0,2} + MPL_{0,4} + MPA_{0,7}$	8,2
25)	$MP_{0,0} + MPA_{0,1} + MPR_{0,4} + MPC_{0,5}$	16,1
26)	$MP_{0,0} + MPB_{0,1} + MPL_{0,4} + MPC_{0,5}$	5,2
27)	$MP_{0,0} + MPA_{0,1} + MPL_{0,4} + MPD_{0,5}$	14,2
28)	$MP_{0,0} + MPB_{0,1} + MPR_{0,4} + MPD_{0,5}$	7,1
29)	$MP_{0,0} + MPD_{0,1} + MPR_{0,4} + MPB_{0,6}$	15,1
30)	$MP_{0,0} + MPC_{0,1} + MPR_{0,4} + MPA_{0,6}$	6,1
31)	$MP_{0,0} + MPC_{0,1} + MPL_{0,4} + MPB_{0,6}$	3,2
32)	$MP_{0,0} + MPD_{0,1} + MPL_{0,4} + MPA_{0,6}$	12,2

The tables B.1, B.2 & B.3 show the indices of the DFT coefficients that can be derived from one another for $N = 6, 10, 14$ respectively and table B.4 shows the number coefficients for which the complete set of $Y_{k_1, k_2}^{(p)}$ need be computed for different N . These tables are used section 4.4 for the derivation of the algorithm for layer 4.

N=6	
k1,k2	(k1,k2)*
0,1	0,5
1,1	5,5
2,1	4,5
3,1	3,5
4,1	2,5
5,1	1,5
0,2	0,4
1,2	5,4
2,2	4,4
3,2	3,4
4,2	2,4
5,2	1,4
1,0	5,0
2,0	4,0
1,3	5,3
2,3	4,3
0,0	
3,0	
0,3	
3,3	

Table B.1

N=10			
k1,k2	(k1,k2)*	k1',k2'	(k1',k2')*
0,1	0,9	0,3	0,7
1,1	9,9	3,3	7,7
2,1	8,9	6,3	4,7
3,1	7,9	9,3	1,7
4,1	6,9	2,3	8,7
5,1	5,9	5,3	5,7
6,1	4,9	8,3	2,7
7,1	3,9	1,3	9,7
8,1	2,9	4,3	6,7
9,1	1,9	7,3	3,7
0,2	0,8	0,6	0,4
1,2	9,8	3,6	7,4
2,2	8,8	6,6	4,4
3,2	7,8	9,6	1,4
4,2	6,8	2,6	8,4
5,2	5,8	5,6	5,4
6,2	4,8	8,6	2,4
7,2	3,8	1,6	9,4
8,2	2,8	4,6	6,4
9,2	1,8	7,6	3,4
1,0	9,0	3,0	7,0
2,0	8,0	6,0	4,0
1,5	9,5	3,5	7,5
2,5	8,5	6,5	4,5
0,0			
5,0			
0,5			
5,5			

Table B.2

N=14					
k1,k2	(k1,k2)*	k1',k2'	(k1',k2')*	k1'',k2''	(k1'',k2'')*
0,1	0,13	0,11	0,3	0,5	0,9
1,1	13,13	11,11	3,3	5,5	9,9
2,1	12,13	8,11	6,3	10,5	4,9
3,1	11,13	5,11	9,3	15,5	13,9
4,1	10,13	2,11	12,3	6,5	8,9
5,1	9,13	13,11	1,3	11,5	3,9
6,1	8,13	10,11	4,3	2,5	12,9
7,1	7,13	7,11	7,3	7,5	7,9
8,1	6,13	4,11	10,3	12,5	2,9
9,1	5,13	1,11	13,3	3,5	11,9
10,1	4,13	12,11	2,3	8,5	6,9
11,1	3,13	9,11	5,3	13,5	1,9
12,1	2,13	6,11	8,3	4,5	10,9
13,1	1,13	3,11	11,3	9,5	5,9
0,2	0,12	0,8	0,6	0,10	0,4
1,2	13,12	11,8	3,6	5,10	9,4
2,2	12,12	8,8	6,6	10,10	4,4
3,2	11,12	5,8	9,6	15,10	13,4
4,2	10,12	2,8	12,6	6,10	8,4
5,2	9,12	13,8	1,6	11,10	3,4
6,2	8,12	10,8	4,6	2,10	12,4
7,2	7,12	7,8	7,6	7,10	7,4
8,2	6,12	4,8	10,6	12,10	2,4
9,2	5,12	1,8	13,6	3,10	11,4
10,2	4,12	12,8	2,6	8,10	6,4
11,2	3,12	9,8	5,6	13,10	1,4
12,2	2,12	6,8	8,6	4,10	10,4
13,2	1,12	3,8	11,6	9,10	5,4
1,0	13,0	11,0	3,0	5,0	9,0
2,0	12,0	8,0	6,0	10,0	4,0
1,7	13,7	11,7	3,7	5,7	9,7
2,7	12,7	8,7	6,7	10,7	4,7
0,0					
7,0					
0,7					
7,7					

Table B.3

N	Total N ²	No of coefficients to be computed (2N+8)
6	36	20
10	100	28
14	196	36
18	324	44

Table B.4

A set of relations are derived, for the coefficients with index given in tables B.1, B.2, B.3, in terms of $Y_{k_1, k_2}^{(p)}$ as permutation over p for different values of (k1, k2). The permutation relations for N = 6, 10 & 14 are given below. These relations are used in section 4.4 for the derivation of algorithm for the layer 4.

N=6

$$Y_{(k_1, k_2)}^{(0)*} = Y_{k_1, k_2}^{(0)}, \quad Y_{(k_1, k_2)}^{(1)*} = -Y_{k_1, k_2}^{(2)}, \quad Y_{(k_1, k_2)}^{(2)*} = -Y_{k_1, k_2}^{(1)}$$

N=10

$$1) \quad Y_{(k_1, k_2)}^{(0)*} = Y_{k_1, k_2}^{(0)}, \quad Y_{(k_1, k_2)}^{(1)*} = -Y_{k_1, k_2}^{(4)}, \quad Y_{(k_1, k_2)}^{(2)*} = -Y_{k_1, k_2}^{(3)},$$

$$Y_{(k_1, k_2)}^{(3)*} = -Y_{k_1, k_2}^{(2)}, \quad Y_{(k_1, k_2)}^{(4)*} = -Y_{k_1, k_2}^{(1)}$$

$$2) \quad Y_{k_1', k_2'}^{(0)*} = Y_{k_1, k_2}^{(0)}, \quad Y_{k_1', k_2'}^{(1)*} = -Y_{k_1, k_2}^{(2)}, \quad Y_{k_1', k_2'}^{(2)*} = Y_{k_1, k_2}^{(4)}, \quad Y_{k_1', k_2'}^{(3)*} = Y_{k_1, k_2}^{(1)},$$

$$Y_{k_1', k_2'}^{(4)*} = -Y_{k_1, k_2}^{(3)}$$

$$3) \quad Y_{(k_1', k_2')}^{(0)*} = Y_{k_1', k_2'}^{(0)}, \quad Y_{(k_1', k_2')}^{(1)*} = -Y_{k_1', k_2'}^{(4)}, \quad Y_{(k_1', k_2')}^{(2)*} = -Y_{k_1', k_2'}^{(3)},$$

$$Y_{(k_1', k_2')}^{(3)*} = -Y_{k_1', k_2'}^{(2)}, \quad Y_{(k_1', k_2')}^{(4)*} = -Y_{k_1', k_2'}^{(1)}$$

N=14

$$1) \quad Y_{(k_1, k_2)}^{(0)*} = Y_{k_1, k_2}^{(0)}, \quad Y_{(k_1, k_2)}^{(1)*} = -Y_{k_1, k_2}^{(6)}, \quad Y_{(k_1, k_2)}^{(2)*} = -Y_{k_1, k_2}^{(5)}, \quad Y_{(k_1, k_2)}^{(3)*} = -Y_{k_1, k_2}^{(4)},$$

$$Y_{(k_1, k_2)}^{(4)*} = -Y_{k_1, k_2}^{(3)}, \quad Y_{(k_1, k_2)}^{(5)*} = -Y_{k_1, k_2}^{(2)}, \quad Y_{(k_1, k_2)}^{(6)*} = -Y_{k_1, k_2}^{(1)}$$

$$2) \quad Y_{k_1', k_2'}^{(0)*} = Y_{k_1, k_2}^{(0)}, \quad Y_{k_1', k_2'}^{(1)*} = -Y_{k_1, k_2}^{(2)}, \quad Y_{k_1', k_2'}^{(2)*} = Y_{k_1, k_2}^{(4)},$$

$$Y_{k_1', k_2'}^{(3)*} = -Y_{k_1, k_2}^{(6)}, \quad Y_{k_1', k_2'}^{(4)*} = -Y_{k_1, k_2}^{(1)}, \quad Y_{k_1', k_2'}^{(5)*} = Y_{k_1, k_2}^{(3)}, \quad Y_{k_1', k_2'}^{(6)*} = -Y_{k_1, k_2}^{(5)}$$

$$3) \quad Y_{(k1',k2)'}^{(0)} = Y_{k1',k2'}^{(0)}, \quad Y_{(k1',k2)'}^{(1)} = -Y_{k1',k2'}^{(6)}, \quad Y_{(k1',k2)'}^{(2)} = -Y_{k1',k2'}^{(5)}, \quad Y_{(k1',k2)'}^{(3)} = -Y_{k1',k2'}^{(4)},$$

$$Y_{(k1',k2)'}^{(4)} = -Y_{k1',k2'}^{(3)}, \quad Y_{(k1',k2)'}^{(5)} = -Y_{k1',k2'}^{(2)}, \quad Y_{(k1',k2)'}^{(6)} = -Y_{k1',k2'}^{(1)}$$

$$4) \quad Y_{k1',k2'}^{(0)} = Y_{k1,k2}^{(0)}, \quad Y_{k1',k2'}^{(1)} = Y_{k1,k2}^{(3)}, \quad Y_{k1',k2'}^{(2)} = Y_{k1,k2}^{(6)}, \quad Y_{k1',k2''}^{(3)} = -Y_{k1,k2}^{(2)},$$

$$Y_{k1',k2'}^{(4)} = -Y_{k1,k2}^{(5)}, \quad Y_{k1',k2'}^{(5)} = Y_{k1,k2}^{(1)}, \quad Y_{k1',k2'}^{(6)} = -Y_{k1,k2}^{(4)}$$

$$5) \quad Y_{(k1',k2'')}^{(0)} = Y_{k1',k2''}^{(0)}, \quad Y_{(k1',k2'')}^{(1)} = -Y_{k1',k2'}^{(6)}, \quad Y_{(k1',k2'')}^{(2)} = -Y_{k1',k2''}^{(5)}$$

$$Y_{(k1',k2'')}^{(3)} = -Y_{k1',k2'}^{(4)}, \quad Y_{(k1',k2'')}^{(4)} = -Y_{k1',k2'}^{(3)}, \quad Y_{(k1',k2'')}^{(5)} = -Y_{k1',k2'}^{(2)},$$

$$Y_{(k1',k2'')}^{(6)} = -Y_{k1',k2'}^{(1)}$$

REFERENCES

1. D. E. Dudgeon, R. M. Mersereau, "Multidimensional Signal Processing", Prentice - Hall Inc., Englewood Cliffs, New Jersey, 1984.
2. C. S. Burrus, T. W. Parks, DFT/FFT and convolution algorithms, Wiley, New York, 1985.
3. R. M. Gray, J. W. Goodman, Fourier Transform: An introduction for Engineers, Kluwer Academic Publishers, Boston, MA, 1995.
4. Albert Cohen, Robert D. Ryan, "Wavelets and Multiscale Signal Processing", Chapman & Hall, UK, 1995.
5. E. O. Brigham, "The Fast Fourier Transform and its Applications", Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1988.
6. R. E. Blahut, "Fast Algorithms for Digital Signal Processing", Addison Wesley Publishing Co., New York, 1985.
7. D. F. Elliot, K. R. Rao, "Fast Transforms Algorithms, Analyses, Applications", Academic Press, New York, 1982.
8. J. G. Proakis, D. G. Manolakis, Digital Signal Processing: Principles, Algorithms and Applications, Prentice- Hall of India, New Delhi, 1995.

9. R. C. Gonzalez, R. E. Woods, "Digital Image Processing", Addison-Wesley Publishing Company, 1992.
10. A. K. Jain, "Fundamentals of Digital Image Processing", Prentice-Hall of India, New Delhi, 1995.
11. T. Chen (Ed.), "The Past, Present, and Future of Image and Multidimensional Signal Processing", IEEE Signal Processing Magazine, pp 21-58, March 1998.
12. J. A. Freeman, D. M Skapura, Neural Networks: Algorithms, Applications and Programming Techniques, Addison - Wesley Publishing Co., New York, 1992.
13. C. G. Y. Lau, B. Widrow, "Neural Networks in IEEE Proceedings", IEEE Trans. on Neural Networks, vol. 1, no. 3, pp 276-278, September 1990.
14. K. Fukushima, "Neural network model for selective attention in visual pattern recognition and associative recall", Applied Optics, vol. 26, no. 23, December 1987.
15. J. M. Zurada, Introduction to Artificial Neural Systems, Jaico Publishing House, Bombay, 1994.
16. P. D. Wassermann, Neural Computing: Theory & Practice, Van Nostrand Reinhold, New York, 1989.
17. D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. I: Foundations, MIT Press, London, 1986.
18. R. J. McEliece, E. C. Posner, E. R. Rodemich, S. S. Venkatesh, "The Capacity of the Hopfield Associative Memory", IEEE Trans. on Information Theory, vol: IT33, No. 4, July 1987.
19. G. A. Carpenter, S. Grossberg, "A Massively Parallel Architecture for a self-Organizing Neural Pattern Recognition Machine", Computer, Vision, Graphics and Image Processing, vol. 37, pp 54-115, 1987.
20. G. A. Carpenter, S. Grossberg, "The ART of Adaptive Pattern Recognition by a Self Organizing Neural Network", IEEE Computer, pp 77- 88, March 1988.
21. T. Kohonen, "The Self-Organizing Map", Proceedings of the IEEE, vol. 78, no. 9, pp 1464-1480, September 1990.
22. T. Kohonen, "The 'Neural' Phonetic Typewriter", IEEE Computer, pp 11-22, March 1988.

23. K. Fukushima, "Neocognitron: A Self-organizing Neural network model for a Mechanism of Pattern Recognition Unaffected by Shift in Position", *Biological Cybernetics*, 36 193-202,1980.
24. K. Fukushima, S. Miyake "Neocognitron: A New Algorithm for Pattern Recognition Tolerent of Deformations and Shift in Position", *Pattern Recognition*, vol. 15, no. 6, pp 455-469, 1982.
25. K. Fukushima, "A Hierarchical Neural Network Model for Associative Memory", *Biol. Cybernetics*, 50, 105-113, 1984.
26. S. Miyake, K. Fukushima, "A Neural Network Model for the Mechanism of Feature-Extraction", *Biol. Cybernetics*, 50, 377-384, 1984.
27. K. Fukushima, N. Wake, "Handwritten Alphanumeric Character Recognition by Neocognitron", *IEEE Trans. on Neural Networks*, vol.2, no.3, pp 355-365, May 1991.
28. K. Fukushima, "Neural network model for selective attention in visual pattern recognition ", *Biological Cybernetics*, 55 5-15,1986.
29. K. Fukushima, "A Neural Network for Visual Pattern Recognition", *IEEE Computer*, March 1988.
30. L. O. Chua, L.Yang, " Cellular Neural Networks: Theory", *IEEE Trans. on Circuits and Systems*, vol. 35, No. 10, pp 1257-1272, 1988.
31. L. O. Chua, L.Yang, " Cellular Neural Networks: Applications", *IEEE Trans. on Circuits and Systems*, vol. 35, No. 10, pp 1273-1290, 1988.
32. F.-L Luo, R. Unbehauen, *Applied neural networks for signal processing*, Cambridge University Press, UK, 1997.
33. J. W. Cooley, J. W. Tukey, " An algorithm for the machine calculation of complex Fourier Series", *Mathematics of computation*, 19, no. 90, pp 297-301, 1965.
34. J. W. Cooley, P. W. A. Lewis, P. D. Welch, "Historical notes on the Fast Fourier Transform", *IEEE Trans. on Audio Electroacoustics*, AU_15, no.2, pp 76-79, June 1967.
35. J. H. McClellan, C. M. Rader (Eds), *Number Theory in Digital Signal Processing*, Prentice-Hall Signal Processing series, 1979.
36. V. Kumar, A. Grama, A. Gupta, G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, The Benjamin / Cummings Publishing Co. Inc., California, 1994.

37. I. Pitas (Ed.), *Parallel Algorithms for Digital Image Processing*, Computer Vision, Neural Networks, John Wiley & Sons, Singapore, 1993.
38. W. W. Smith, J. M. Smith, *Handbook of Real-Time Fast Fourier Transform: Algorithms to Product Testing*", IEEE Press, New York, 1995.
39. M. C. Pease, "An Adaptation of the Fast Fourier Transform for Parallel Processing", *J. Ass. Comput. Mach.* Vol. 15 pp 252 - 264, April 1968.
40. R. C. Singleton, "A Method for Computing the Fast Fourier Transform with Auxiliary Memory and Limited High-Speed Storage", *IEEE Trans. Audio Electroacoust.*, vol. AU-15, pp 91-97, June 1967.
41. R. C. Singleton, "An Algorithm for Computing the Mixed Radix Fast Fourier Transform", *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp 93-103, June 1969.
42. S. Winograd, "On computing the Discrete Fourier Transform", *Mathematics of Computation*, vol. 32, no. 141, January 1978, pp 175-199, in J. H. McClellan, C. M. Rader (Eds), 'Number Theory in Digital Signal Processing', Prentice-Hall Signal Processing series, 1979.
43. L. Auslander, E. Feig, S. Winograd, "New algorithms for the multidimensional discrete Fourier transform", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-31, no. 2, pp 388-403, April 1983.
44. P. K. Rajan, "A Study on the Properties of Multi-Dimensional Fourier Transform", *IEEE Trans. on Circuits and Systems*, vol. CAS-31, no. 8, pp 748-750, August 1984.
45. I. Gertner, "A new efficient algorithm to compute the two-dimensional discrete Fourier transform", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 36, no. 7, pp 1036-1050 July 1988.
46. J. T. Rayfield, H. F. Silverman, "Implementation of 2-D algorithms on a loosely-coupled parallel system", *Proc. of IEEE ICASSP'88*, pp 1997-2000, 1988.
47. P. K. Rajan, "Fast DFT algorithms diagonally symmetric 2-D data", *Proc. of IEEE ICASSP'88*, pp 1411-1414, 1988.
48. H. V. Sorensen, C. S. Burrus, D. L. Jones, "A new efficient algorithm for computing a few DFT points", *Proc. of IEEE ISCAS'88*, 1915-1918, 1988.
49. H. Babic, S. Loncaric, "A perfect interpolation of discrete Fourier transform", *Proc. of IEEE ISCAS'88*, pp 1911-1914, 1988.

50. L. Wang, I. Hartimo, T. Laakso, "A Novel double decomposition method for systolic implementation of DFT", Proc. of IEEE ISCAS'92, 1085-1088, 1992.
51. C.-Y. Chen, C.-L. Wang, "A new efficient systolic architecture for the 2-D discrete cosine transform", Proc. of IEEE ISCAS'92, vol. 2, 689-692, 1992.
52. D. Yang, "Fast computation of two-dimensional discrete Fourier transform using fast discrete Radon transform", Proc. of IEEE TENCON'90, pp 207-210, 1990.
53. I. Gertner, M. Rofheart, "A parallel algorithm for 2-D DFT computation with no inter processor communication", IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 3, pp 377 - 382, July 1990.
54. A. W. Julien, W. G. Bliss, "The systematic design of area efficient VLSI architecture for the discrete Fourier transform", Proc. of IEEE 25ACSSC'91, pp121-125, 1991.
55. H. Miyanaga, H. Yamauchi, K. Matsuda, "A Real Time 256x256 Point Two Dimensional FFT Single Chip Processor", Proc. of IEEE ICASSP'91, pp1193-1196, 1991.
56. W. Ma, "Ring structure and multi-dimensional discrete Fourier transform on a power of 2", Proc. of IEEE ISCAS'92, vol.3, 1510-1512, 1992.
57. R. Rajarevivarma, P. K. Rajan, "Fast computation of 2-D DFT of real valued nonsymmetric and centro symmetric sequences", Proc. of IEEE ISCAS'92, vol.3, 2449-2452, 1992.
58. C. J. Ju, "Equivalent relationship and unified indexing of FFT algorithms", Proc. of IEEE ISCAS'93, pp742-745, 1993.
59. A. Gupta, V. Kumar, "The scalability of FFT on parallel computers", IEEE Transactions on Parallel and Distributed Systems, vol.4, no.7, pp 922 - 931, August 1993.
60. P. Fragopoulou, S. G. Akl, "A parallel algorithm for computing Fourier Transforms on the Star Graph", IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 5, pp 525 - 531, May 1994.
61. J. Pihl, "Tradeoffs between parallel and serial architectures in high performance digital signal processing", Proc. of IEEE ISIC'97, Singapore, 10-12 September 1997.
62. K.-W. Shin, M.-K. Lee, "A VLSI architecture for parallel computation of FFT" in Systolic array processors, Edited by J. McCanny, J. McWriter, E. Swartzlander Jr., Prentice Hall, 1989.

63. W. S. McCulloch, W. Pitts, "A logical calculus of the ideas imminent in nervous activity", *Bulletin of Mathematical Biophysics*, 5, 115-133, 1943.
64. D. O. Hebb, "The Organization of Behavior", John Wiley & Sons, New York, 1949.
65. R. Rosenblatt, *Principles of Neurodynamics*, New York, Spartan books, 1959.
66. B. Widrow, M. E. Hoff, "Adaptive Switching Circuits", 1960 IRE WESCON Conv. Record, Part 4, 96-104, August 1960.
67. J. J. Hopfield, "Neural Networks and Physical Systems with Emergent collective Computational Abilities", *Proc. Natl. Acad. Sci. USA*, vol. 79, 2554-2558, April 1982.
68. J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", *Proc. of Natl. Acad. Sci. USA*, *Biophysics*, vol. 81, pp 3088-3092, May 1984.
69. J. J. Hopfield, D. W. Tank, "Computing with Neural Circuits: A Model", *Science*, vol. 233, 625-633, August 1986.
70. D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning Internal Representations by Error Propagation" in D. E. Rumelhart, J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1 Foundations*, MIT Press, London, 1986.
71. T. Sejnowski, C. R. Rosenberg, "NETtalk: A parallel Network That Learns to read Aloud", Johns Hopkins Univ. Technical Report JHU/EECS-86/01, 1986.
72. J. A. Feldman, D. H. Ballard, "Connectionist Models and Their Properties", *Cognitive Science*, vol. 6, 205-254, 1982.
73. S. Grossberg, *The Adaptive Brain I Cognition, Learning, Reinforcement, and Rythm and The Adaptive Brain II Vision, Speech, Language, and Motor Control*, Elsevier/North -Holland, Amsterdam, 1986.
74. S. E. Hampson, D. J. Volper, "Linear Function Neurons: Structure and Training", *Biological Cybernetics*, vol. 53, pp 203-217, 1986.
75. M. K. Habib, R. W. Newcomb, "A Digital Neuron Type Processor and its VLSI Design", *IEEE Trans. on Circuits and Systems*, vol. 36, pp 739-746, May 1989.
76. M. K. Habib, R. W. Newcomb, "Neuron Type Processor Modeling Using Timed Petri Net", *IEEE Trans. on Neural Networks*, vol. 1, no. 4, pp 282-289, December 1990.

77. J. L. Meador, A. W. Clint Cole, N. Nintunze, P. Chintrakulchai, "Programmable Impulse Neural Circuits", IEEE Trans. on Neural Networks, vol. 2, no. 1, pp 101-109, January 1991.
78. M. Fukumi, S. Omatu, "A New Back-propagation Algorithm with coupled Neuron", IEEE Trans. on Neural Networks, Vol. 2, No. 5, pp 535-538, September 1991.
79. J. Glanz, "Mastering the Nonlinear Brain", Research News, Science, vol. 277, 19 September 1997.
80. R. P. Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp 4-22, April 1987.
81. B. Widrow, R. Winter, "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition", IEEE Computer, pp 25-39, March 1988.
82. D. S. Touretzky, D. A. Pomerleau, "What is hidden in the hidden layer?", BYTE, pp 227-233, August 1989.
83. B. Widrow, M. A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", Proceedings of the IEEE, vol.78, N0.9, pp 1415-1442, September 1990.
84. S.-I. Amari, "Mathematical Foundations of Neurocomputing", Proceedings of the IEEE, vol.78, N0.9, pp 1443-1462, September 1990.
85. T. Poggio, F. Girosi, "Networks for approximation and Learning", Proceedings of the IEEE, vol. 78, N0. 9, pp 1481-1497, September 1990.
86. G. N. Reeke, O. Sporns, G. M. Edelman, "Synthetic Neural Modeling: The 'Darwin' Series of Recognition Automata, Proceedings of the IEEE, vol. 78, no. 9, pp 1498-1530, September 1990.
87. R. Braham, J. O. Hamblen, "The Design of a Neural Network with a Biologically Motivated Architecture", IEEE Trans. On Neural Networks, vol. 1, no. 3, pp 251-262, September 1990.
88. A. Rodriguez-Vazquez, S. Espejo, R. Dominguez-Castro, J. L. Huertas, E. Sanchez-Sinencio, "Current-Mode Techniques for the Implementation of Continuous- and Discrete-Time Cellular Neural Networks", IEEE Trans. on Circuits and Systems - II Analog and Digital Signal Processing, vol. 40, no. 3, pp 132-146, March 1993.

89. J. E. Varrientos, E. Sanchez-Sinencio, J. Ramirez-Angulo, "A Current-Mode Cellular Neural Network Implementation", *IEEE Trans. on Circuits and Systems - II : Analog and Digital Signal Processing*, vol. 40, No. 3, pp 147-155, March 1993.
90. N. Fruehauf, E. Lueder, G. Bader, "Fourier Optical Realization of Cellular Neural Networks", *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 3, pp 156-162, March 1993.
91. T. Roska, L. O. Chua, "The CNN Universal Machine: An Analogic array Computer", *IEEE Trans. on Circuits and Systems - II Analog and Digital Signal Processing*, vol. 40, No. 3, pp 163-173, March 1993.
92. V. Cimagalli, M. Bobbi, M. Balsi, "MODA: Moving Object Detecting Architecture", *IEEE Trans. on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 40, No. 3, pp 174-183, March 1993.
93. G. Seiler, J. A. Nossek, "Winner-Take-All Cellular Neural Networks", *IEEE Trans. on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 40, no. 3, pp 174-183, March 1993.
94. H. Harrer, "Multiple-Layer Discrete-Time Cellular Neural Network Using Time-Variant Templates", *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 3, pp 191-199, March 1993.
95. I. A. Baktir, M. A. Tan, "Analog CMOS Implementation of Cellular Neural Networks", *IEEE Trans. on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 40, no. 3, pp 200-206, March 1993.
96. T. Sziranyi, J. Csicsvari, "High Speed Character Recognition Using a Dual Cellular Neural Network Architecture (CNND)", *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, No. 3, pp 223-231, March 1993.
97. K. Fukushima, "Cognitron: A Self-organizing Multilayered Neural Network", *Biological Cybernetics*, vol. 20, No. 3/4 pp 121-136, November 1975.
98. B. A. White, M. I. Elmasry, "The Digi-Neocognitron - A Digital Neocognitron Neural Network model for VLSI", *IEEE Trans. on Neural Networks*, vol.3, no.1, pp 73-84, January 1992.
99. J. Nellis, T. J. Stonham, "The Boolean Neocognitron", in *Intelligent Engg. Systems through Artificial Neural Networks*, Proc. of ANN in Engg. (ANNIE'92) Ed. C. H. Dgli, L. I. Burke, Y. C. Shin, pp 385-390, 1992.

100. Y. Hatakeyama, Y. Kakazu, "Detecting a target using an expanded Neocognitron", Proc. of ANN in Engg. (ANNIE'92) Ed. C. H. Dgli, L. I. Burke, Y. C. Shin, pp 391-396,1992.
101. J. S. Kane, M. J. Paquin, "POPART Partial Optical Implementation of ART2", IEEE Trans. On Neural Networks, vol. 4, no. 4, pp 695-702, July 1993.
102. J. H. Li, A. N. Michael, W. Porod, " Qualitative Analysis and Synthesis of a class of Neural Networks", IEEE Trans. on Circuits and Systems, vol. 35, no. 8, pp 976-986, August 1988.
103. J. H. Li, A. N. Michael, W. Porod, "Analysis and Synthesis of a class of Neural Networks: Linear systems operating on a closed Hypercube", IEEE Trans. on Circuits and Systems, vol. 36, no. 11, pp 1405-22, November 1989.
104. G. Palm, "On Associative Memory", Biological Cybernetics, 36, pp 19-31, 1980.
105. H. S. Chung , J. K. Ryeu, W. I. Lee, " A novel digital multiplier chip based on the neural network", Proc. of IEEE ISCAS'90, pp1046-1049, 1990.
106. M. J. Whitcomb, M. F. Augusteijn, " Hierarchical learning in neural nets a new paradigm", Proc. of ANN in Engg. (ANNIE'91) Ed. C. H. Dgli, L. I. Burke, Y. C. Shin, pp 107-112, 1991.
107. T. Nitta, "A Backpropagation algorithm for complex numbered neural networks", Proc. of IEEE IJCNN'93, pp 1649-1652, 1993.
108. S. M. Barber, J. G. Delgado-Frias, S.Vassiliadis, G. G. Pechanek, "SPIN-L Sequential pipelined neuroemulator with learning capabilities", Proc. of IEEE IJCNN'93, pp1927-1930, 1993
109. Lami, Mantra, " Quantitative comparison of architectures for digital neuro-computers", Proc. of IEEE IJCNN'93, pp 1987-1990, 1993.
110. P. Masa, K. Hoen, H. Wallinga, " 70 input, 20 nanosecond pattern classifier", Proc. of IEEE IJCNN'94, pp 1854-1859, 1994.
111. F. Distanto, M. Sami, R. Stefanelli, G. Storti-Gajani, " Mapping Neural Nets onto a Massively Parallel Architecture: A Defect - Tolerance Solution", Proceedings of the IEEE, vol. 79, No. 4, pp 444-459, April 1991.
112. T. Watanabe, Y. Sugiyama, T. Kondo, Y. Kitamura,"Neural Network Simulation on a Massively Parallel Cellular Array Processor AAP-2", Proc. of IEEE IJCNN'89 vol. 2, pp155-161, 1989.

113. S. Y. Kung, J. N. Hwang, "Parallel Architectures for Artificial Neural Nets", Proc. of IEEE ICNN'88, vol. 2, pp165-172, 1988.
114. H. P. Graf, L. D. Jackel, W. E. Hubbard, " VLSI Implementation of a Neural Network Model", IEEE Computer, pp 41- 49, March 1988.
115. J. Hutchinson, C. Koch, J. Luo, C. Mead, "Computing Motion Using Analog and Binary Resistive Networks", IEEE Computer, pp 52-63, March 1988.
116. M. T. Mostafavi, " Specialized interconnection architecture for VLSI implementation of neural network models", Proc. of ANN in Engg. (ANNIE'92) Ed. C. H. Dgli, L. I. Burke, Y. C. Shin, pp 385-390, 1992.
117. G. L. Heilman, M. Georgiopoulos, W. D. Roome, " Concurrent Object-Oriented Simulation of Neural Network Models", Proc. of IEEE IJCNN'92, vol. 2, pp 553-559, 1992.
118. E. Torbey, B. Haroun, " Architectural synthesis for digital neural networks", Proc. of IEEE IJCNN'92, vol. 2, pp 601-606, 1992.
119. S. Tam, M. Holler, J. Brauch, A. Pine, A. Peterson, S. Anderson, S. Deiss, " A reconfigurable multi-chip analog neural network; recognition and backpropagation training", Proc. of IEEE IJCNN'92, vol. 2, pp 625-630, 1992.
120. L. Fu, "Rule Generation from Neural Networks", IEEE Trans. on Systems, Man and Cybernetics, vol. 24, no. 8, August 1994.
121. M. I. Gomez, M. R. Kabuka, " Implementation of a new digital neural network using ASIC technology", Proc. of ANN in Engg. (ANNIE'91) Ed. C. H. Dgli, L. I. Burke, Y. C. Shin, pp 53-60,1991.
122. S. Y. Kung, J. N. Hwang, " Digitl VLSI architectures for neural networks", Proc. of IEEE ISCAS'89, pp445-448, 1989.
123. H. Speckman, P. Thole, W. Rosenstiel, " Hardware synthesis for neural networks from a behavioral description with VHDL", Proc. of IEEE IJCNN'93, pp1983-1986, 1993.
124. J. J. Vidal, " Implementing Neural Nets with Programmable Logic", IEEE Trans. on Acoustics Speech and Signal Processing, vol. 36, no.7, July 1988.
125. M. Ramkumar, G. V. Anand, "An FFT-Based Technique for Fast Fractal Image Compression", Proceedings of Communications '98, Bangalore, 1998, pp 204-209.
126. S. M. Shalinie, R. Rajaram, "Neuro-Fuzzy Data Compression Model for Image Recognition", Proceedings of Communications '98, Bangalore, 1998, pp197-203.

127. M. A. Barros, M. Akil, R. Natowicz, "A reconfigurable architecture for real time segmentation of image sequences using self-organizing feature maps", Proc. of IEEE IJCNN'93, pp 197-202, 1993.
128. A. N. Netravali, B. G. Haskell, "Digital Pictures: Representation and Compression", Plenum Press, New York, 1988.
129. P. M. Athanas, A. L. Abott, "Real-time image processing on a custom platform", IEEE Computer magazine, vol. 28, no.2, February 1995.
130. G. A. Carpenter, S. Grossberg, Pattern Recognition by Self Organizing Neural Networks, MIT Press, London, 1991.

INDEX

- 2-D DFT computation 6, 24, 25, 27, 28,
34, 43, 70, 75, 108, 111, 133
- 2-D FFT 23, 28
- Architecture
 - Layered 74, 78
 - Massively parallel 38
- ASIC 39, 110, 113, 138
- Brain 15, 21, 22, 30, 42, 74
- Cellular Neural Network 21, 32, 131, 136
- CHAP 81, 84, 85, 87, 88, 94
- CHAPL 81, 83, 84, 85, 87, 89, 94
- CHBP 81, 83, 85, 87, 88, 94
- CHBPL 81, 83, 85, 89, 94
- CN 57, 62
- Complexity 3, 6, 23, 77, 88, 89, 96, 97, 98,
101, 103, 106, 107, 108, 110, 111, 112
- Computed imaging 13
- Convolution 3, 10, 112, 129
- CP 46, 57, 62, 80, 83, 84, 86, 89, 91
- CPA 57, 62, 80, 83, 84, 86, 89, 91
- CPB 46, 57, 62, 91
- Digital signal processing 1, 2
- Discrete Fourier Transform 3, 4, 6, 26, 132
- Discrete Radon Transform (DRT) 27
- DN 57, 62
- DP 46, 47, 57, 62, 80, 83, 84, 86, 89, 91
- DPA 46, 57, 61, 62, 80, 83, 84, 86, 89, 91
- DPB 57, 60, 62, 91
- Feature extraction 2, 10, 17, 19, 35
- Feed forward 15, 16, 19, 20, 30, 31, 37,
39, 40
- Feldman 29, 134
- Flip horizontal 57, 60, 61
- Flipping vertical 57, 61
- FPGA 40
- Fractal 40

Frequency domain processing	3, 10, 25, 43, 45, 70, 104, 105, 107, 112, 113	Joseph Fourier	3, 25
Grossberg	29, 130, 134, 139	LA	45, 47, 48, 58, 63, 68
HAN	56, 62, 68	LB	45, 58, 63, 69, 79
HAP	46, 56, 60, 62, 64, 68, 69, 80, 83, 86, 88, 90, 93	Learning	
HAPL	46, 47, 56, 60, 62, 64, 68, 69, 80, 83, 86, 88, 90, 93	supervised	107
HAPR	56, 62, 68	unsupervised	19, 34
Hardware	2, 9, 24, 27, 32, 33, 36, 39, 40, 100, 107, 109, 110, 111	McClelland	29, 77, 130, 134
HBN	57, 62, 68	McCulloch and Pitts	15, 29
HBP	46, 57, 60, 62, 64, 68, 69, 80, 83, 86, 88, 90, 93	Memory	1, 18, 23, 31, 33, 34, 37, 39, 102, 106
HBPL	46, 57, 60, 62, 64, 68, 69, 80, 83, 86, 88, 90, 93	MN	62, 91
HBPR	57, 62, 68	Modified Neocognitron	20
Hebb	29, 134	MP	46, 60, 62, 80, 83, 84, 87, 88, 90, 91, 94
Hopfield	17, 29, 37, 38, 130, 134	MPA	46, 62, 80, 83, 84, 87, 88, 90, 91
Hopfield network	17	MPB	62, 91
Image compression	11	MPC	61, 62, 91
Image deblurring	1	MPD	46, 60, 62, 80, 83, 84, 87, 88, 90, 91
Image enhancement	1, 10, 11	MPL	46, 47, 62, 80, 83, 84, 87, 88, 90, 91
Image Recognition	40, 112	MPR	62, 91
Image scanning	1	Multidimensional DFT	26, 113
Implementation		Multilayer Perceptron	16
Fully parallel	96, 102, 103, 108	Multithread	96
Fully sequential	97, 103, 108	Neocognitron	19, 20, 34, 35, 107, 131, 137
Group sequential	97, 108	Neural network model	24, 29, 30, 32, 34, 35, 38, 39, 73, 78, 103, 107, 113, , 138
Plane sequential	97, 108	Neuron model	15, 29, 30, 111
VLSI	21, 27, 38, 110, 138	Noise removal	2
Information rearrangement	1	Parallel distributed processing	22, 76, 77
Information reduction	1	Parameter estimation	2, 22
JAVA	96	Perceptron model	15, 16
		periodic	5, 8, 44
		Primitive	45, 48, 56, 58, 62, 63, 65, 67,

	70, 71, 77, 79, 80, 91, 121	Threshold	15, 30, 37, 40
RA	45, 49, 58, 63, 68, 79	Transforms	3, 9, 129, 133
RB	45, 58, 63, 69, 79	Twiddle factor	5, 45, 65, 70, 77, 79, 94, 101, 106, 109, 110
Real-time	21, 22, 32, 36, 74	VHDL	40, 139
Rumelhart	29, 77, 130, 134	Visual Manipulation	42
RVLP	81, 84, 87, 88, 90, 93	VLN	56, 60, 62, 64, 68
RVLPA	81, 83, 84, 87, 88, 90, 93	VLP	46, 47, 56, 59, 62, 64, 68, 69, 80, 81, 82, 83, 86, 88, 90, 93
RVRP	81, 83, 84, 85, 87, 88, 90, 94	VLPA	46, 56, 59, 62, 64, 68, 69, 80, 82, 83, 86, 88, 90, 93
RVRPA	81, 83, 84, 85, 87, 88, 90, 94	VLPB	56, 60, 62, 64, 68
Sejnowski	29, 134	VRN	56, 62, 68
Self organizing	14, 18, 19, 35, 38	VRP	46, 56, 60, 62, 64, 68, 69, 80, 83, 86, 88, 90, 93
Self organizing map	18	VRPA	46, 56, 60, 62, 64, 68, 69, 80, 83, 86, 88, 90, 93
Simulation	24, 30, 39, 40, 96	VRPB	56, 62, 68
Spectral analysis	2, 3, 25	Weight	15, 16, 17, 19, 37, 76, 77, 78, 94, 109, 111
SRVLP	81, 85, 87, 89, 92, 94	Widrow	29, 31, 130, 134, 135
SRVLPA	81, 84, 85, 87, 89, 92, 94	Winner-Take-All learning	19
SRVRP	89, 92, 94		
SRVRPA	89, 92, 94		
Symbol	42, 45		

LIST OF PUBLICATIONS OF THE AUTHOR

1. "Influence of the number of Hidden Layer Neurons on the Recognition Capability of BPN", Proc. of SRC '95, Cochin, 4-6 May 1995
2. "Handwritten Numeral Recognition using BPN and Neocognitron", Proc. of SRC '95, Cochin, 4-6 May 1995.
3. "A Pictorial Representation of 6x6 - point DFT in terms of 2x2 - point DFT", Proc. of NCBME '97, Chennai, 28 - 29 March 1997.
4. "A new Modulo Arithmetic based Hierarchical Neural Network (MAHNN) model to implement NxN - point DFT for $((N))_4 = 2$ ", Proc. of NET-X'97, CSI, Cochin, 16-17 May 1997
5. "Performance Evaluation of MAHNN model to implement NxN point DFT for $((N))_4 = 2$ ", Proc. of NET-X '97, CSI, Cochin, 16-17 May 1997.

6. "A parallel distributed implementation of 6x6 point DFT", Proc. of IEEE international conference ISIC-97, Nanyang Technological University, Singapore, 10-12 September 1997, pp 106-108.
7. "Visual Manipulation of Data for Analysis - A DFT Example", Proc. of BECON-97, Cochin, 4-6 September 1997.
8. "An application of Parallel Distributed computation of 6x6-point DFT in the determination of Ray Paths", Proc. of the National Symposium SYMPOL-'97, Cochin, 16-17 Dec. 1997.
9. "Visual manipulation of symbols to implement 2-D DFT", Accepted for presentation at the National Conference ANCS-98, Cochin, September 1998.
10. "A new model for parallel distributed computation of 6x6 - point DFT suitable for VLSI implementation", Communicated to the IEEE International Conference VLSI '99, January 1999.
11. "A comparative study of Neocognitron & Modified Neocognitron models in Recognizing Handwritten Characters", Accepted for presentation at the National Conference ANCS-98, Cochin, September 1998.