International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 2 | Issue – 4

Design of Systolic Architecture Using Evolutionary Computation

Shilpa V, Suma V Shetty
Assistant Professor, Department of ECE, Sapthagiri College of Engineering,
Bengaluru, Karnataka, India

ABSTRACT
This work presents a new concept for finding the optimal values for the entire three fundamental design vectors namely: scheduling, projection and processor so that not only architecture design could be feasible along with that maximum hardware utilizing efficiency could be achieved. This Approach also having the focus to minimize the total delay involved with systolic architecture design. Evolutionary programming has applied to find the optimal solution. Presented work and result will provide facility to designer without any involvement to find out best suited architecture for a particular application. The Proposed method having capability to find the large number of optimal vectors for any algorithm which can be implemented in systolic architecture. The increasing demands of speed and performance in modern signal and image processing applications necessitate a revolutionary super –computing technology. The proposed method is coded in MATLAB editor and simulation environment.

Keywords: *Scheduling; Projection; Systolic Array*

1. INTRODUCTION

The essential goal of developing new computer architectures and efficient use of existing modern systems is to run larger and more complicated applications faster over time. The continued demand for increased computing power led in the late 1980's to the development of high parallel scalable multiprocessing systems. Parallel computing is a form of computation which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel")[1]. The most effective way to improve the computer performance in terms of computational speed is to use parallel processing architectures, which employ multiple processors to perform a computation task. When multiple processors working together, an appropriate architecture is very important to achieve the maximum performance in a cost- effective manner. Systolic arrays are ideally qualified for computationally intensive applications with inherent massive parallelism because they capitalize on regular, modular, rhythmic, synchronous, concurrent processes that require intensive, repetitive computation. There is a necessity of an essential tool which maps all the DSP algorithms or high level computations in to hardware architecture which maximizes the hardware utilization efficiency [2]. Systolic Architecture is a general methodology for mapping high-level computations into hardware structures.

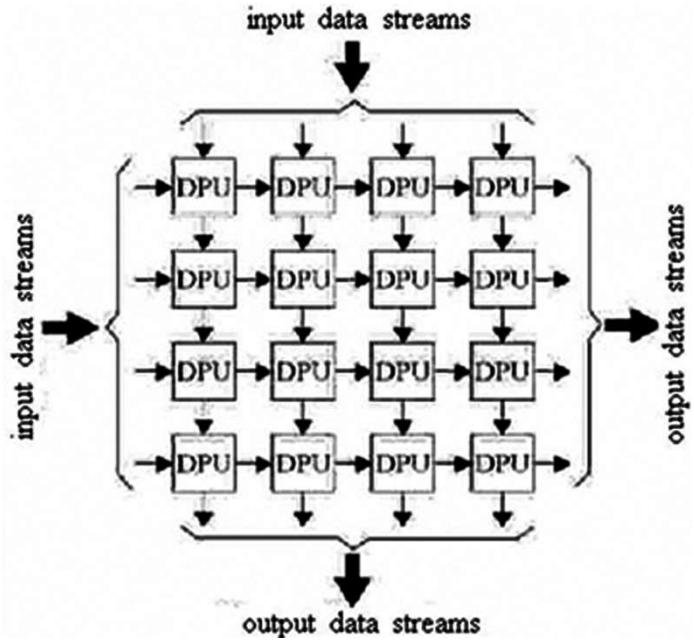

Fig. 1 General model of systolic array

In a systolic system, data flows from the computer memory in a rhythmic fashion, passing through many processing elements before it returns to memory much as blood circulates to and from the heart. This can be rectangular, triangular or hexagonal to make use of higher degrees of parallelism. Moreover to implement a variety of computations, data flow in a systolic system may be at multiple speeds in multiple directions-both inputs and (partial) results flow, whereas only results flow in classical pipelined systems. The Fig.1 presents the General Model of a Systolic Array.

The rest of the work in the paper is arranged as; Section 2 briefs out the basic knowledge of systolic array and its architecture. Section 3 explains the implementation of systolic array using evolutionary computation; also define the meaning of evolutionary computation and how we can use it in our design. Section 4 addresses about the designing of systolic array, there we have taken example of 3 tap FIR filter and 3x3 matrix multiplication to understand the design. Section 5 we have discussed results which are obtained after implementing the systolic array for FIR filter and matrix multiplication using MATLAB tool.

2. SYSTOLIC ARRAY

A systolic array is an arrangement of processors in an array where data flows synchronously across the array between neighbors, usually with different data flowing in different directions. Each processor at each step takes in data from one or more neighbors, processes it and, in the next step, outputs results in the opposite direction. H. T. Kung and Charles Leiserson were the first to publish a paper on systolic arrays in 1978, and coined the name. Systolic array is a specialized form of parallel computing with multiple processors connected by short wires. Each unit is an independent processor. Cells (processors) compute data and store it independently of each other. Every processor has some registers and an ALU [3].

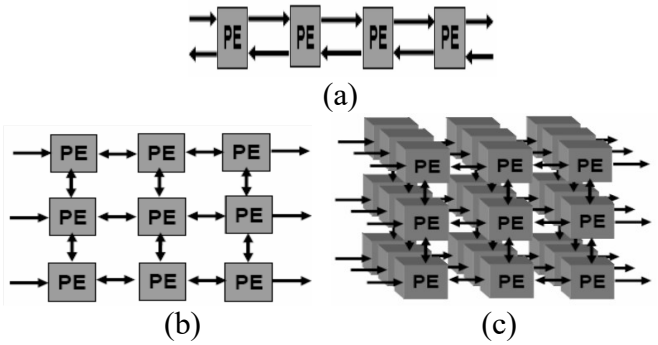

Fig. 2 systolic Array Models; (a) 1D; (b) 2D; (c) 3D

The cells share information with their neighbors, after performing the needed operations on the data. Systolic system is easy to implement because of its regularity and easy to reconfigure. This architecture can result in cost-effective, high-performance special-purpose systems for a wide range of problems. Some simple examples of systolic array models are shown in the Fig. 2.

A systolic array is composed of matrix-like rows of data processing units called cells (DPU). Each cell shares the information with its neighbors immediately after processing. The systolic array is often rectangular where data flows across the array between neighbor DPUs, often with different data flowing in different directions. Systolic architectures, an architectural concept originally proposed for VLSI implementation of some matrix operations. A systolic system consists of a set of interconnected cells each capable of performing some simple operation. As simple, regular communication and controlled structures have substantial advantages over complicated ones in design and implementation; cells in a systolic system are typically interconnected to form a systolic array or systolic tree. Information in systolic systems flows between cells in a pipelined fashion and communication with the outside world occurs only at the “boundary cells”. By replacing the single processing element with an array of PE’s as shown in Fig. 2 , a higher computation throughput can be achieved without increasing memory bandwidth Being able to use each input data item a number of times (and thus achieving high computation throughput with only modest memory bandwidth) is one of the many advantages of systolic approach.

3. EVOLUTIONARY COMPUTATION

Evolutionary computation uses computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems. There are a variety of evolutionary computational models that have been proposed and studied which we will refer to as evolutionary algorithms. They share a common conceptual base of simulating the evolution of individual structures via processes of selection and reproduction. These processes depend on the perceived performance (fitness) of the individual structures as defined by an environment. More precisely, evolutionary algorithms maintain a population of structures that evolve according to rules of selection and other operators, such as

recombination and mutation. Each individual in the population receives a measure of its fitness in the environment. Selection focuses attention on high fitness individuals, thus exploiting the available fitness information. Recombination and mutation perturb those individuals, providing general heuristics for exploration. Although simplistic from a biologist's viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

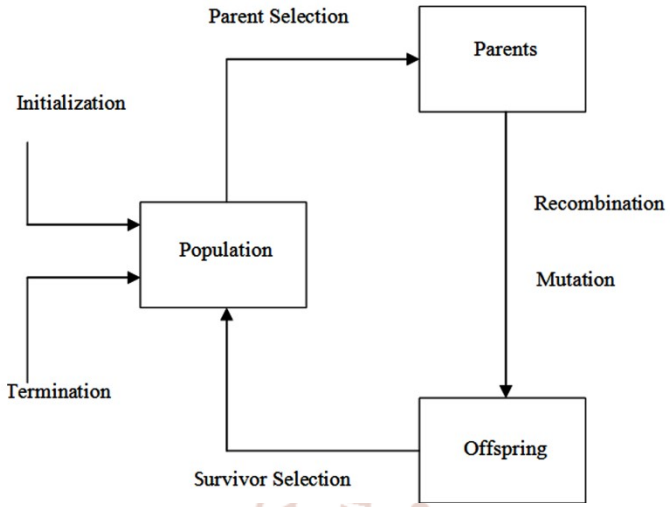

Fig. 3 General Scheme of Evolutionary Computation.

The Fig. 3 outlines a typical evolutionary algorithm (EA). A population of individual structures is initialized and then evolved from generation to generation by repeated applications of evaluation, selection, recombination, and mutation. The population size N is generally constant in an evolutionary algorithm, although there is no a priori reason (other than convenience) to make this assumption.

3.1 *Proposed Algorithm*

The basic steps of algorithm are given as follows:

1. The problem is defined as finding the real-valued n dimensional vector x that is associated with the extremum of a functional $F(x)$: $R^n \longrightarrow R$.
2. An initial population of parent vectors, x_i, $i = 1 \ldots P$ is selected at random from a feasible range in each dimension.
3. An offspring vector, x'_i, $i = 1, \ldots, P$, is created from each parent x_i by adding a Gaussian random variable with zero mean and preselected standard deviation to each component of x_i.
4. Selection then determines which of these vectors to maintain by comparing the errors $F(x_i)$ and F (x'_i), $i= 1 \ldots P$. the P vectors that possess the least error become the new parents for the next generation.
5. The process of generating new trials and selecting those with least error continues until a sufficient solution is reached or the available computation is exhausted.

4. SYSTOLIC ARCHITECTURE DESIGN

Systolic array architectures are designed by using linear mapping techniques on regular dependence graphs (DG). The DG corresponds to space representation, where no time instance is assigned to any computation (t=0). A DG is said to be regular, if a presence of an edge in a certain direction at any node in the DG represents presence of an edge in the same direction at all nodes in the DG[4].

Basic vectors involved in the systolic array design are:

1. Projection vector (also called as iteration vector)

$$d = \begin{pmatrix} d1 \\ d2 \end{pmatrix} \qquad (1)$$

Where d is the projection vector.

Two nodes that are displaced by'd 'or multiples of 'd 'are executed by same processor.

2. Processor space vector:

PT= (p1, p2), Any node with index IT= (i, j) would be executed by processor in space time representation by:

$$PTI = (p1, p2)\begin{pmatrix} i \\ j \end{pmatrix} \qquad (2)$$

3. Scheduling Vector ST= (s1, s2)

Any node with index I would be executed by:

$$STI = (s1, s2)\begin{pmatrix} i \\ j \end{pmatrix} \qquad (3)$$

4. Hardware Utilization Efficiency

$$HUE = 1 / |ST\ d| \qquad (4)$$

This is because two tasks executed by the same processor are spaced $|ST\ d|$ time units apart. Much systolic architecture can be designed for a given

problem by selecting different projection, processor space and scheduling vectors. These vectors must satisfy the feasibility constraints.

The feasibility constraints are

1. If points A and B differ by the projection vector, i.e. (IA-IB) is same as d, then they must be executed by the same processor. In other words:

$$P^T IA = P^T IB \ ; P^T (IA-IB) = 0 \ ; P^T d = 0 \quad (5)$$

2. If A and B are mapped to the same processor, then they cannot be executed at the same time. i.e.

$$STIA \neq STIB \ ; \quad ST (IA-IB) \neq 0 \ ; STd \neq 0 \quad (6)$$

Once the above two constraints are satisfied, edge mapping is done.

5. Edge mapping

If an edge E exists in a space representation or DG then an edge PT e is introduced in a systolic array with ST e delays. Once edge mapping is done, then we construct low-level implementation of the particular design. The Fig.4 displays the flow chart of the proposed method[5]. The proposed algorithm is applied to two designs and the Space-Time Representation of FIR filter is show in the Fig.5. The low level architecture for the systolic array is shown in the Fig.6. The Design 1 deals with the Broadcast inputs, Move Results and Weights Stay.

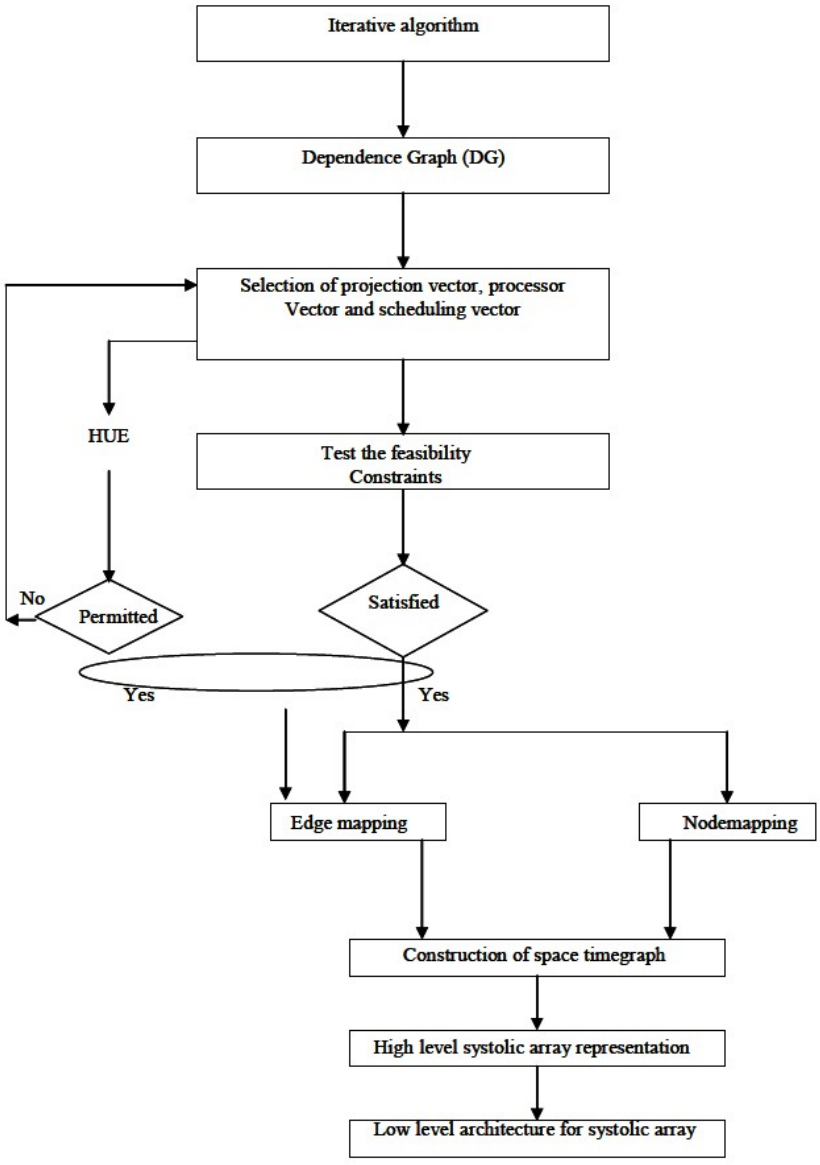

Fig. 4 Model of the Proposed Methodology

The vector for the systolic arrays has been assumed as Projection vector d = (1,0), Processor vector P^T = (0 1), Scheduling vector S^T = (1 0) and the space representation for the FIR filter is displayed in Fig. 5.

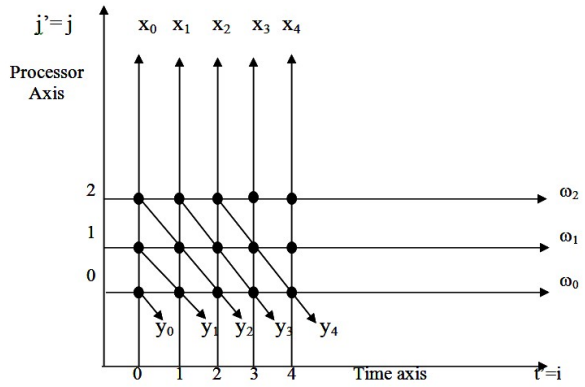

Fig. 5 Space-Time Representation of FIR filter

The Low level architecture for the above FIR filter is presented in the Fig.6. The coordinates for the FIR filter has been calculated from the basic vectors in the systolic array and finally the Fig.5 that's the space representation for the considered FIR filter has been presented.

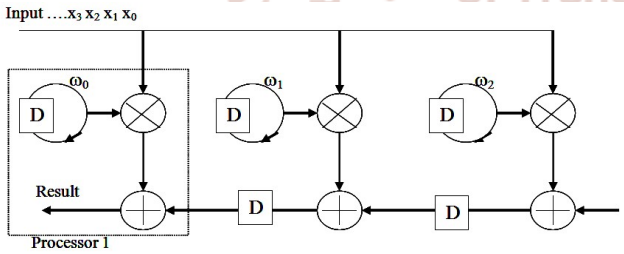

Fig. 6. Low level architecture for systolic array for design I

The Design 2 deals with Fan-in results, Move inputs and Weights Stay. The projection vector, processor vector and scheduling vector for the design 2 assumed as Projection vector d (1,0) , Processor vector PT = (0 1) and Scheduling vector ST = (1 1). The Space Time Representation of the above given design is shown in Fig. 7. The selection of scheduling vector: For any specified projection vector, processor space vector and scheduling vector, the systolic array can be designed using linear mapping technique. We can select the feasible scheduling vector using scheduling inequalities. Based on selected scheduling vector sT , the projection vector d and the processor space vector p^T can be selected according to scheduling inequalities: p^T d=0 And s^T d ≠ 0.and hence the systolic array can be obtained.

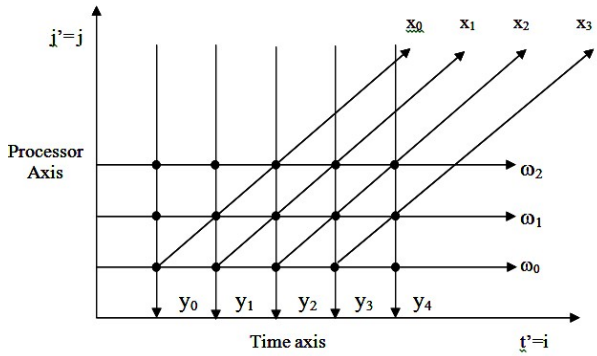

Fig. 7 Space-Time Representation of the design II.

The Low level architecture for the above design II is presented in the Fig.8. The coordinates for the FIR filter has been calculated from the basic vectors in the systolic array and finally the Fig.7 that's the space representation for the considered design I have been presented.

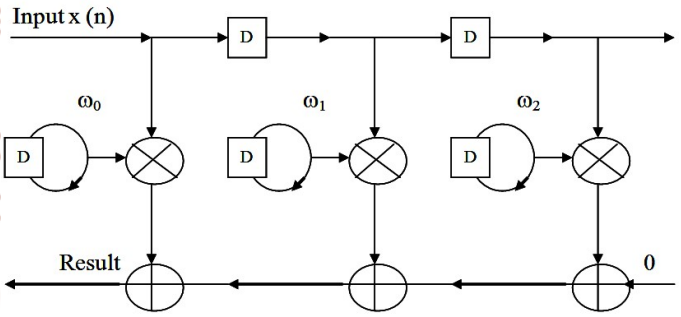

Fig. 8 Low level architecture for systolic array for design II.

6. EXPERIMENTAL RESULTS

When the projection vector, processor vector and scheduling vectors are known for the proposed design I and II, the proposed method has been designed using matlab code and the design have been executed in the Matlab Lab Tool box. The result windows are displayed from Fig.9. The figure displays the input sequence for the design I and the Fig.9 shows Edge mapping result window for the design I. If the projection vector, processor vector and scheduling vectors are not known to prior, then using DG it is possible to find these vectors by the following design methodology. Selection of schedule vector using scheduling inequalities. The vector coordinates were considered as mentioned earlier.

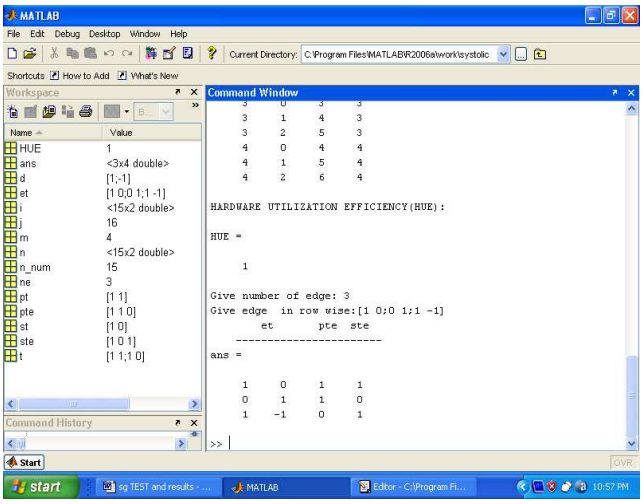

Fig. 9 Edge mapping result window for the design I

The Fig.9 displays the Edge mapping result window for the design I. The Output Window to find the Processor vector when s^T=[1 1 1] and d=[1 –1 1] for design II is displayed in the Fig.10.

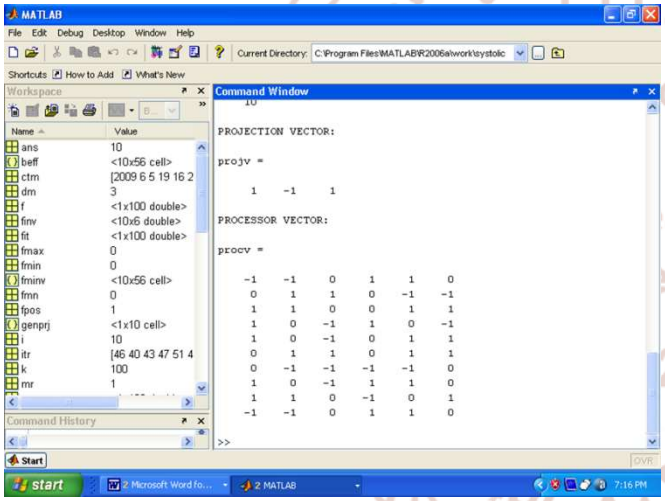

Fig. 10. Output Window to find the Processor vector.

The Advantages of Systolic Arrays are simple and regular data and control flows use of simple and uniform cells ; Extremely fast ; Easily scalable architecture; Can do many tasks single processor machines cannot attain ;Turns some exponential problems into linear or polynomial time. The applications of Systolic Arrays in general, systolic designs apply to any compute bound problem that is regular that is, one where repetitive computations are performed on a large set of data. In the field of Signal and Image processing: FIR, IIR filtering, and 1-D convolution; 2-D convolution and correlation; Discrete Fourier Transform;1-D and 2-D median Filtering; Artificial Neural networks; Matrix Multiplication: Matrix-Vector multiplication; Matrix-Matrix multiplication ; Matrix triangularization (solution of linear systems, matrix inversion) ; Graph algorithms-Transitive closure, minimum spanning trees, and connected components.

CONCLUSION

The CAD tool designed will help us to design the systolic array according to the different strategies. Design I When the projection vector, processor vector and scheduling vector are given. Here we are able to find the node mapping and edge mapping. For Design II, When the vectors mentioned in design 1 are not known. In this using EP we are able to find different design patterns. Where in the user can choose the design depending upon his application. In this work, we have designed SA for 3-tap FIR filter and Matrix Multiplication. Once the systolic array is designed, and then further we have to go for retiming, folding, unfolding where: Retiming refers to Retiming is the technique of moving the structural location of latches or registers in a digital circuit to improve its performance, area, and/or power characteristics in such a way that preserves its functional behavior at its outputs.

REFERENCE

1. Kalle Tammemäe, system on chip architecture, Dept. of CE, Tallinn Technical University 2000/02.
2. Mahendra Vucha, MANIT, “Design and FPGA Implementation of Systolic Array Architecture for Matrix Multiplication” International Journal of Computer Applications (0975 – 8887),Volume 26– No.3, July 2011
3. Jason HandUber, Systolic Arrays Presentation at UCF, Feb 12, 2003.
4. Richard Hughey, Programming Systolic Arrays, Proc. Int. Conf. Application-Specific Array Processors, IEEE Computer Society, Aug. 4-7, 1992.
5. Lan-DaVan, “Systolic Architecture Design”, Ph. D. Dissertation, Department of Computer Science, National ChiaoTung University, Taiwan, R. O. C., fall, 2010).
6. Rudra Pratap, Getting Started with Matlab 7 (A Quick Introduction for Scientists and Engineers).
7. Ingle & Proakis, Digital Signal Processing Using Matlab V4.