

An Empirical Model Based Optimal Architecture for N-Bits
M-Points Fast Fourier Transform



Author

Muhammad Ghashan Ali

00000206067

Supervisor

Dr. Shoab Ahmed Khan

Co-Supervisor

Dr. Sajid Gul Khawaja

DEPARTMENT OF COMPUTER ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

July 2019

An Empirical Model Based Optimal Architecture for N-Bits
M-Points Fast Fourier Transform

Author

Muhammad Ghashan Ali

00000206067

A thesis submitted in partial fulfillment of the requirements for the degree of
MS Computer Engineering

Supervisor

Dr. Shoab Ahmed Khan

Co-Supervisor

Dr. Sajid Gul Khawaja

Thesis Supervisor's Signature: _____

Thesis Supervisor's Signature: _____

DEPARTMENT OF COMPUTER ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD

FEBRUARY 2019

Declaration

I certify that this research work titled “*An Empirical Model Based Optimal Architecture for N-Bits M-Points Fast Fourier Transform*” is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Muhammad Ghashan Ali

00000206067

Language Correctness Certificate

This thesis has been read by an English expert and is free of most typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

Muhammad Ghashan Ali

00000206067

Signature of Supervisor

Dr. Shoab Ahmed Khan

Signature of Co-Supervisor

Dr. Sajid Gul Khawaja

Copyright Statement

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of E&ME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.
- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of E&ME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of E&ME, Rawalpindi.

Acknowledgements

By the grace of the ALLAH Almighty, my Master's thesis is finally complete. It was quite an arduous exercise that could not have been completed without the help of ALLAH Almighty and the strength that he bestowed upon me.

Moreover, I owe my deepest gratitude to my incredible supervisor ***Professor Dr. Shoab Ahmed Khan*** for his patient guidance, motivation and appreciation throughout this the course of this thesis. I am very thankful to him that he showed faith in me and allowed me with one of my life's greatest opportunity to work with him.

I am also highly obliged and grateful to my thesis co-supervisor ***Dr. Sajid Gul Khawaja*** who always encouraged me in my entire thesis work, irrespective of whether its thesis report, research publication, presentations, so on and so forth. Without him this incredible journey would be pretty difficult.

Furthermore, a special thanks to ***Dr. Muhammad Usman Akram*** for his valueable feedback in this research work. Many times he became a helping hand in execution of this work.

I am also eternally grateful to my mother and sisters, Sharmeen and Ishmal, who supported me. In addition to my friends and family, especially Engr. M. Waqas, Dr. Shoaib & Dr. Johar who were a helping hand to me in every step of my journey. Their support was what made me accomplish this herculean task.

Without their support – in every sense of the word – and constant guidance, this thesis would never have been conceived or completed (and definitely not in time). I cannot thank them enough for their role in the completion of this thesis and report. Additionally, I would like to express my gratitude towards everyone who provided me assistance in any stage of this thesis accomplishment.

In dedication

*To my **Father Mr. Waqar Akhtar (Late)***

&

*To my **Mother Mrs. Samia Firasat:***

for encouraging and supporting me to achieve this daunting task.

*To my **Siblings, Cousins and Friends:***

for their patience, continuous care and support during my hard times.

Abstract

The use and vast implementation of Discrete Fourier Transform has revolutionized the world and allowed the researchers to think of the modern world from a different perspective. The discovery of Fast Fourier Transform has laid the foundation of an entirely new dimension to the modern world. Keeping in view its utmost importance in the future industry researchers tried to design its hardware architecture as per the requirement of the application. Several architectures have been proposed time to time with new inventions in the previous designs. Some architectures consider clock rate, some take architectural area into consideration, some focuses on parallel execution of the algorithm, so on and so forth. Considering all these inputs to the industry that has been a part to modern world time to time, this research presents an empirical model based upon the optimal architectures for Fast Fourier Transform algorithm for n-bits m-points input. This empirical model is obtained by making several architectures and their respective characteristics are obtained. The data obtained is then passed through a machine learning algorithm known as Regression Algorithm. Linear, quadratic and cubic regression technique is applied to achieve the hierarchy of the designed architectural parameters and this intern will provide us with the empirical models of the architecture. This model will provide us with the specifications of the futuristic architecture that mainly depends upon the one's requirement i.e. either one considers a single parameter or a tradeoff between different hardware parameters. The parameters that are mainly considered are number of Slice LUT's, LUT FF Pairs, clock rate, number of processing elements and number of clock cycles required. This proposed methodology can be applied to any hardware architectural designs for analysis and generation of empirical models.

Key Words: *Discrete Fourier Transform, Fast Fourier Transform, Processing Element, Butterfly Architecture, n Radix FFT, Permutation Matrix, Kronecker Product.*

Table of Contents

DECLARATION	I
LANGUAGE CORRECTNESS CERTIFICATE.....	II
COPYRIGHT STATEMENT	III
ACKNOWLEDGEMENTS	IV
ABSTRACT.....	VI
TABLE OF CONTENTS.....	VII
LIST OF FIGURES	IX
LIST OF TABLES.....	X
CHAPTER 1 : INTRODUCTION.....	1
1.1 MOTIVATION.....	1
1.2 PROBLEM STATEMENT.....	2
1.3 AIMS AND OBJECTIVES	3
1.4 STRUCTURE OF THESIS.....	3
CHAPTER 2 : LITERATURE REVIEW.....	4
2.1 INTRODUCTION TO FAST FOURIER TRANSFORM.....	4
2.2 RADIX 2 FAST FOURIER TRANSFORM	5
2.2 PROPOSED FFT ARCHITECTURES BY RESEARCHERS	6
CHAPTER 3 : METHODOLOGY.....	13
3.1 EMPIRICAL MODEL FOR FFT.....	13
3.1 PROPOSED HARDWARE ARCHITECTURE	14
3.1.1 <i>Basic Building Block / Butterfly Processing Element</i>	16
3.1.2 <i>Butterfly Processing Element Architecture</i>	17
3.1.3 <i>Field Programmable Gate Array (FPGA) Implementation</i>	18
3.1.4 <i>Select Lines for Proposed Architecture</i>	20
3.2 WORKING OF EMPIRICAL MODEL.....	22
CHAPTER 4 : EXPERIMENTAL RESULTS.....	25
4.1 RESULTS AND DISCUSSION	25
4.2 EXECUTION, WORKING AND IMPLEMENTATION OF EMPIRICAL MODEL	32
CHAPTER 5 : CONCLUSION AND FUTURE WORK.....	37
5.1 CONCLUSION	37
5.2 CONTRIBUTION.....	38
5.3 FUTURE WORK.....	38
CHAPTER 6 : REFERENCES.....	40
ANNEXURE : FINDING THE UNIQUE PERMUTATION MATRIX FOR REVERSE ORDER KRONECKER PRODUCT INTUITIVELY FOR FAST FOURIER TRANSFORM.....	42
1 INTRODUCTION	42
2 PERMUTATION MATRIX.....	43
2.1 <i>Rows and Column Permuted Matrix</i>	43
3 KRONECKER PRODUCT AND PERMUTATION MATRIX.....	44
4 PROPOSED METHODOLOGY	45

5	OBSERVATION AND RESULTS.....	48
6	CONCLUSION.....	50
7	REFERENCES.....	ERROR! BOOKMARK NOT DEFINED.

List of Figures

FIGURE 2-1: RADIX 2- 8 POINT FAST FOURIER TRANSFORM ARCHITECTURE.....	5
FIGURE 3-1: INPUTS/OUTPUTS OF EMPIRICAL MODEL	14
FIGURE 3-2: BLOCK DIAGRAM OF PROPOSED ARCHITECTURE.....	15
FIGURE 3-3: HARDWARE IMPLEMENTATION OF 4 POINT FFT WITH 1 PROCESSING ELEMENT.....	16
FIGURE 3-4: BUILDING BLOCK FOR FFT (BUTTERFLY PROCESSING ELEMENT).....	17
FIGURE 3-5: BUTTERFLY ELEMENT ARCHITECTURE DESIGN	17
FIGURE 3-6: FPGA DESIGN FOR 4 POINT 1 PE	18
FIGURE 3-7: DETAILED RTL SCHEMATIC FOR 4 POINT 1 PE ARCHITECTURE	18
FIGURE 3-8: FPGA DESIGN FOR 4 POINT 2 PE	19
FIGURE 3-9: DETAILED RTL SCHEMATIC FOR 4 POINT 2 PE ARCHITECTURE	19
FIGURE 3-10: APPLICATION OF LINEAR, QUADRATIC AND CUBIC REGRESSION ON A RANDOM DATASET [29]	22
FIGURE 4-1: MATLAB FFT COMPLEX SIGNAL EXECUTION.....	25
FIGURE 4-2: GENERATED WAVE FORM FOR 4 POINT FFT USING 1 PE	26
FIGURE 4-3: GENERATED WAVE FORM FOR 4 POINT FFT USING 2 PE	26
FIGURE 4-4: COMPARISON OF 8 BIT INPUT FOR M-POINT FFT ON BASIS OF FIGURE (A): SLICE LUT's ; FIGURE (B) LUT FF PAIRS; FIGURE (C)CLOCK RATE.....	27
FIGURE 4-5: COMPARISON OF 16 BIT INPUT FOR M-POINT FFT ON BASIS OF FIGURE (A): SLICE LUT's ; FIGURE (B) LUT FF PAIRS; FIGURE (C)CLOCK RATE.....	28
FIGURE 4-6: COMPARISON OF 32 BIT INPUT FOR M-POINT FFT ON BASIS OF FIGURE (A): SLICE LUT's ; FIGURE (B) LUT FF PAIRS; FIGURE (C)CLOCK RATE.....	29
FIGURE 4-7: REGRESSION TECHNIQUE FOR 8 BIT INPUT AND 1 PROCESSING ELEMENT FOR M-POINT FFT ON FIGURE (A): SLICE LUT's ; FIGURE (B) LUT FF PAIRS; FIGURE (C)CLOCK RATE.....	30
FIGURE 4-8: COMPARISON OF 32 BIT INPUT FOR CLOCK RATE:	32
FIGURE ANNEX-0-1: GENERATION OF 4! PERMUTATIONS OF 4x4 IDENTITY	45

List of Tables

TABLE 1: SELECT LINES FOR 4 POINT 1 PROCESSING ELEMENT	20
TABLE 2: SELECT LINES FOR 4 POINT 2 PROCESSING ELEMENT	20
TABLE 3: SELECT LINES FOR 8 POINT 1 PROCESSING ELEMENT	21
TABLE 4: SELECT LINES FOR 8 POINT 2 PROCESSING ELEMENT.....	21
TABLE 5: SELECT LINES FOR 8 POINT 4 PROCESSING ELEMENT	21
TABLE 6: EMPIRICAL MODELS FOR N-BIT {8, 16 AND 32}, M-POINT \in {4, 8, 16 AND 32} FOR PROCESSING ELEMENTS \in {1, 2, 4, AND 8}	31
TABLE 7: REQUIREMENTS FOR GENERATION OF HARDWARE PARAMETERS:.....	32
TABLE 8: ARITHMETIC CALCULATIONS BASED ON EMPIRICAL MODEL FOR HARDWARE PARAMETERS STATED IN TABLE 7	33
TABLE 9: SQUARED ERROR FOR DIFFERENT PROCESSING ELEMENTS AS PER REQUIREMENT IN TABLE 7.....	33
TABLE 10: REQUIRED CYCLES FOR GENERATED HARDWARE FOR 4 PROCESSING ELEMENTS.....	34
TABLE 11: ARITHMETIC CALCULATIONS BASED ON EMPIRICAL MODEL FOR HARDWARE PARAMETERS STATED IN TABLE 7 FOR 16 BIT INPUT	34
TABLE 12: REQUIRED CYCLES FOR GENERATED HARDWARE FOR 2 PROCESSING ELEMENTS.....	35
TABLE 13: ARITHMETIC CALCULATIONS BASED ON EMPIRICAL MODEL FOR HARDWARE PARAMETERS STATED IN TABLE 7 FOR 32 BIT INPUT	35
TABLE 14: REQUIRED CYCLES FOR GENERATED HARDWARE FOR 1 PROCESSING ELEMENT	36
TABLE 15: SUMMARIZED SQUARED ERRORS FOR 8, 16 AND 32 BITS INPUT	36

Chapter 1 : INTRODUCTION

With an adverse research in field of science and technology, Fast Fourier Transform emerges out as one of the main basic tool for signal analysis and signal processing. The actual implementation and usage of FFT in major applications in modern era, for instance in implementation of 5G MIMO OFDM system [1], hearing devices, MRI, classical mechanics, military purposes and much more [2], that depicts its utmost importance in daily life of an individual.

In modern world, the data is expanding adversely. The amount of data we produce on daily basis is truly mind boggling. Approximately 2.5 quintillion bytes of data [3] of data is generated on daily basis. Almost 90% of the data currently present, has been generated over the last two years [4]. With the increase in data generation the analytical tools are also burdened to improve the efficiency and performance of the current systems. Taking software as research platform for analysis of such big data is generally a time taking task that can be catered for using state of the art hardware, capable of performing such an immense task in the meantime.

Considering the applications in of FFT in real life, the implementation of this algorithm varies as per the requirement of the applications. Some applications consider the hardware execution area to be minimized, some take frequency constraints into consideration, few applications focuses on the parallelization of this algorithm to lessen the execution time and many such requirements for several applications are addressed by researchers at different times.

In this report we present an empirical model based optimal architecture for FFT algorithm that focuses on the requirement of the application and depicts its nearly possible architecture based upon the proposed model, for n-Bits m-Points input considering hardware parameters as Slice LUT's, LUT FF Pairs, clock rate and clock cycle.

1.1 Motivation

It's quiet difficult to believe, but in 1986, we had as little as 1% of today's total media storage capacity in digital form. By 2007, this number came up to 94%, which best illustrates the speed at which digitalization came about now [5].

With an advent of digital world, signals and systems have made its way to the technological path. Understanding, exploring and improving the world has always been a part of human nature. During the last decade, several mind blowing researches on signal processing have been made that entirely changed the thinking perspective of the digital world that may consider discovery of Fast Fourier Transform by Cooley and Turkey [6] in 1965 on the top of the list. With the passage of FFT has proved its utmost importance in the daily life of an individual because of its immense use in major applications.

Since its usage varies as per its need in any application, in order to design a hardware for its use in any area / field, we must design a generalized optimal hardware based upon real time models / architectures, that is capable of performing all the desired task to a certain level, so that one can meet his technological requirement as per his need.

We need to specialize hardware because in every aspect of life in modern world everything is in its hardware form. Specializing the hardware is very much necessary to meet up the day to day requirements in a way to meet the software requirement.

In the early stages of the digital technology, the algorithms were not that much extensive and hence doesn't need that much hardware execution, but in the modern world this is currently not the case, now there is a special need to specialize the hardware so that we can meet up with the software requirements and can excel in state of the art technology.

As the required specs to execute the intensive algorithms are increased, the state of the art hardware may not meet up the software requirements. This needs the hardware researchers to get the way out of it. As any hardware specs is altered, there should have been any trade off in some other specs, and hence it's currently the need of the modern world to excel the hardware to meet up the day to day requirements.

1.2 Problem Statement

FFT is being used in the modern era as a key to many bottlenecks faced in signal processing. Its wide hardware applications forced the researchers to put their input into it, to meet the requirements of the modern world using this technique. The purpose of this report is to provide with an empirical model that is based upon optimal FFT architecture n-bits m-Points using Field Programmable Gate Array (FPGA) as hardware platform. This empirical model is obtained using a renowned machine learning algorithm i.e. regression algorithm.

This model will help the users in a way that they can predict their FFT architecture constraints as per their need/requirement before its actual implementation.

Furthermore, the technique used in this report can be used in with any algorithm to design its empirical model.

1.3 Aims and Objectives

Major objectives of the research are as follow:

- Formation of several optimal Architecture for Fast Fourier Transform.
- Architectural analysis of FFT architecture.
- Acquiring parameters for all the possible architectures for n-Bits m-points
- Generation of Empirical Model using Regression Algorithm on all the required parameters.
- Optimal architecture generation as per one's requirement by using empirical model.

1.4 Structure of Thesis

This work is structured as follows:

Chapter 2 gives review of the literature and the significant work done by researchers in past few years for classification of heavenly entities using light curves.

Chapter 3 consists of the proposed methodology in detail. It includes the details about the proposed FFT hardware.

Chapter 4 includes all the experimental results accompanied by relevant figures. In addition to this, execution and working of the empirical model is also discussed in this section.

Chapter 5 concludes the thesis and reveals future scope of this research

Chapter 6 contains the references used in this thesis report.

Annexure briefly describes an intuitive technique to achieve Permutation Matrix for finding reverse order Kronecker Product to execute FFT.

Chapter 2 : LITERATURE REVIEW

FFT acts as a basic fundamental tool for almost all the up gradation that exists in the modern era. This tool is considered as one of the most powerful tool that Signal Processing have with itself. The actual implementation of anything that is used by people ends up in its hardware form. As we are discussing here the use, implementation, execution and implementations of FFT, many hardware implementations are proposed by researchers as per the requirement and need of the application.

The domain of the stated report mainly centers the actual architectural implementation of Fast Fourier Transform algorithm. Researchers have proposed many architectures based upon their requirement and specification of different platforms e.g. Field Programmable Gate Array (FPGA), Graphical Processing Unit (GPU), Central Processing Unit (CPU) etc. Some of the proposed architectural designs of FFT on different platforms are discussed below.

2.1 Introduction to Fast Fourier Transform

Fourier transform is considered as one of the fundamental tools in any signal processing and analysis [7] [8] that allows us to bifurcate individual frequency components of any digital signal. Fourier Transform is one of the most well established transformers to study a signal from frequency prospective, it's analysis and filtering. It primarily concerned with the representation of a signal by estimation of trigonometric functions or more precisely by a series of periodic functions i.e. sinusoids. For a given sequence $x(n)$, an n -point Discrete Fourier Transform (DFT) can be calculated as

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

W_N^{nk} is known as twiddle factor that can be calculated as,

$$W_N^{nk} = e^{-j\frac{2\pi nk}{N}} = \cos\left(\frac{2\pi nk}{N}\right) - j \sin\left(\frac{2\pi nk}{N}\right) \quad (2)$$

The computational cost of direct implementation of DFT as stated in equation 1 is $O(N^2)$. By using the symmetry and periodicity properties of the twiddle factors, the FFT algorithm can reduce the complexity to $O(N \log_2 N)$. The importance of this transform cannot be denied especially considering about its importance from application and analysis perspective in Engineering, Technology and research.

2.2 Radix 2 Fast Fourier Transform

FFT is considered as an elementary tool for conversion of time domain signal into frequency domain that has wide applications in real life. The use of FFT in different applications that requires its hardware implementation, cannot be denied. Several hardware architectures of FFT algorithm by many researchers, implement and use this transform as per their need of use, as its implementation technique and usage may vary, as some application primarily concerns about frequency constraints, some with throughput, few towards hardware efficient implementation whereas some take processing time into consideration. All such type of architectures are precisely discussed. In addition to this, here we present an optimal architecture for n-Bit m-Point Fast Fourier Transform that is mainly based upon the empirical models for different hardware constraints that give the expected requirements of the architecture as per the one's need i.e. either one wants his hardware to be area efficient, time efficient, a tradeoff between area and time or it may depend upon one's requirement/priority for implementation of his desired architecture.

Here radix 2 implementation of Fast Fourier Transform is taken into consideration for implementation Figure 2-1 and generation of an empirical model.

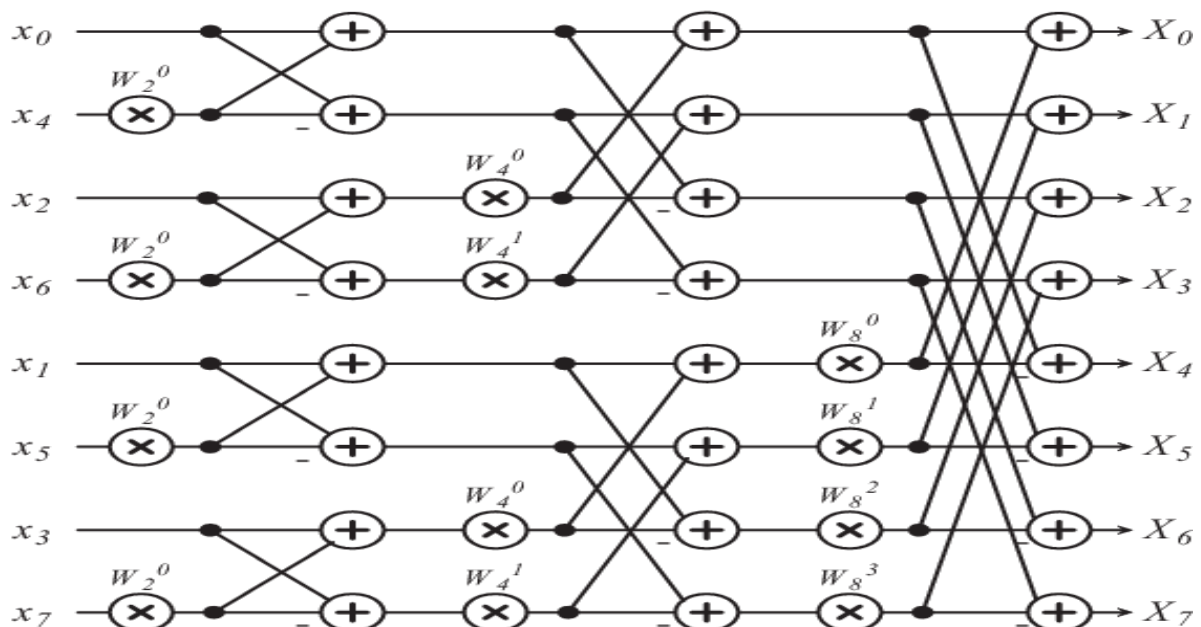


Figure 2-1: Radix 2- 8 Point Fast Fourier Transform Architecture

2.2 Proposed FFT Architectures by Researchers

Feifei shen et al. [9] demonstrates the advantages and validity of using GPUs for FFT over CPUs especially when large input size is targeted. The proposed design has two types of input textures: image data textures containing rectangular texture (RGBA) for $M \times N$ image whereas other one secondary texture i.e. $\log(N)$ and N columns where each row stores the real and imaginary part of primary rotating factor. In this way the FFT butterfly calculation requires $\log(M) \times \log(N)$ times rendering. GPU used is Nvidia Geforce 8600 video card as hardware of GPU. Experimental comparison for different input size for open CV time, GPU time and GPU maximum error were demonstrated by the author.

Mokhtar A. Aboleaze et al. [10] concentrated his research on energy consumption in memory access for FFT calculation. It was also investigated the effect of number of registers in CPU to lessen the energy consumption in accessing the memory. It was also investigated that the number memory access depends upon machine code, compiler, and writing the memory. It was assumed in the paper that compiler uses available register to access in store the data. The FFT algorithm that were used for comparison were radix two DIF FFT radix 4 and a Twiddle factor based FFT algorithm with reduced memory access.

Peter A. Milder et al. [11] used a well-known tool i.e. spiral, that automatically generates corresponding hardware implementations, for DFT with non-power of two input size. According to him most previous work on hardware implementations of non-power of two sized input for DFT focuses on producing a solution for a specific requirement/situation/application, i.e. a given problem size and performance requirement for specified hardware. The author discusses 4 FFT algorithms including pease FFT, iterative FFT, mixed radix FFT and Bluestein FFT.

Kota soloman Raju et al [12] proposed a DFT and IDFT hardware architecture based on floating point numbers to achieve accuracy and precision. General purpose arithmetic modules based on 32 bit single precision IEEE.754 standard are design firstly and then hardware architecture for DFT and IDFT based on radix two butterfly computation was perform. Hardware sharing scheme was also introduced to lessen the hardware cost. To design the architecture Verilog Hardware Description Language (VHDL) was used, simulated on ModelSim 6.6E on Xilinx Virtex.5 LX110T board.

K Ganesoan et al. [13] proposed a general purpose sequential FFT processor for 1024 points. The processor is capable of performing 1024 point FFT execution in 26.3 msec. the processor is configured in distributed processing system with Intel 8086 microprocessor.

Radix 2 hardware implementation was performed on ATRW (1003J) 12x12 bit multiplier cum accumulator LSI chip was used as an arithmetic unit. The proposed processor computes each butterfly operation in 64 basic clock cycles taking 5.12 μ sec. the processor works on 20 KHz sampling frequency.

Gouhui et al. [14] presented a novel design for configurable FFT/IFFT module to provide scalability and reconfigurability. Here unified radix structure for radix 2,3,4,5 & 7 is proposed. Furthermore issues for designing high performance Fourier algorithm for cognitive radio communication and network systems are discussed. The clock period for the proposed architecture is 1.988×10^{-9} sec, whereas the f_{max} is 500 MHz.

Muniandi Kannan et al. [15] proposed DIT FFT pipelined hardware architecture for low power multiplier less radix 4, single path delay combinator pipelined FFT processor for 16, 64 & 256 points for fixed points inputs. The proposed multiplier less architecture uses common sub expression sharing so that it may replace complex multiplications by simple shifting and addition operations. In this way a low power butterfly architecture is achieved. 59% and 43% power reduction is achieved for 16 points and 64 points radix 4 FFT when the proposed architecture is compared with conventional FFT architecture based on non booth coded Wallace tree multiplier. The parameterization impact on power speed and performance is also compared.

K. Indira Priyadarsini et al. [16] proposed a pipeline VLSI implementation for FFT, that adopts a single path delay feedback. A reconfigurable complex multiplier and bit parallel multipliers are used to store the twiddle factors and eliminating ROM (Read only Memory) and achieving a ROM less FFT processor, thus the architecture may consume low power according to the author. This ROM less low power FFT processor can be used for OFDM applications. The author has designed a reconfigurable complex constant multiplier such that the size of ROM for twiddle factors can be considerably shrunk.

Rozita Teymourzadeh et al. [17] proposed an FFT architecture considering floating points to achieve high precision FFT. Since floating point architecture limits maximum clock frequency and increases the power consumption, the author focuses on improving the speed, area, resolution, power consumption and latency for FFT architecture. The proposed architecture illustrates VLSI for floating points parallel pipelined (FPP) 1024 radix 2 processor making use of single butterfly element incorporated for intelligent controller. The proposed radix two FPP-FFT was optimized in AISC under Silterra 0.18 μ m and Mimos 0.35 μ m technology libraries. 32 bit data was processed and synthesized using Xilinx ISE platform. Maximum clock frequency for FPP-FFT processor was obtained as 227 MHz. The

latency for 1024 points input is 22 μ s. The Estimated power consumption for Silterra and Mimos was 640 μ W and 1.198 μ W respectively.

Archana Fande at el.[18] developed a low power complex multiplier design to reduce the hardware required to implement the FFT algorithm for Radix 4. The aim to implement a complex multiplier is to offer high speed, low power consumption and lesser area. In this way this proposed architecture would be suitable for various high speed, low power VLSI architectures. These three parameters i.e. power, area and speed are always tradeoff. The implementation of the hardware architecture was done on Spartan 6 Trainer Kit and the hardware characteristics were compared with Spartan 3 using VHDL. The maximum path delay to implement the complex multiplier in Radix 4 was obtained to be 11.656 nsec.

Yazan Samir Algnabi at el.[19] proposed a novel multiplier less pipelined architecture for Radix 22 SDF FFT based on using digital slicing technique to meet the requirement for high speed wireless communications system standards. An optimal constant multiplication arithmetic architectural design for multiplication of a particular input with specified twiddle factor is also proposed by the author. The proposed architecture was simulated on MATLAB and FPGA Virtex 4. The hardware design was tested on TLA5201 logic analyzer and a high speed of 669.277 MHz was achieved. The author claimed his proposed architecture to be 3.35 times faster compared with the conventional architectures and it only consumes 20% of the conventional butterfly area. The proposed architecture comprises of twiddle factors that are saved in ROM, Digital slicing complex multiplier, processing elements that are generally butterflies and $N \log_2 N$ counter

Yifan Bo at el.[20] proposed an FFT processor for low power applications for variable length input. The author employed a modified fast scaling scheme and trading method to improve SQNR (Signal to Noise Ratio) performance. Memory based architecture is proposed to support variable length FFT processing. To reduce the power dissipation, a tailored constant multiplier array is introduced in the data path. The author claims to perform 64 to 8192 input FFT at 100MHz processing speed. The SQNR of 55.4dB and 33.3dB are achieved for 64 point and 8192 point FFT respectively.

Atin Mukerjee at el.[21] proposed an area efficient Radix 2 FFT architecture that reuses the same butterfly element several times, that intern rescues the required area. To reuse the butterfly element many times and forwarding the input to the same processing element, a routing network is used that routes the input at a specified time. The proposed FFT processor is simulated using VHDL and the results are simulated on Virtex 4 FPGA. The

author claims that this architecture outperforms the conventional architecture for N-Point FFT Processor in terms of area that is reduces by a factor of $\text{Log}N_2$ with +

Yousri Ourhani at el. [22] present an architecture based on radix 4 FFT algorithm consisting of a novel memory sharing and dividing technique with processing elements having parallel in parallel out capability. The proposed architecture is capable to perform N Point FFT with $4/3$ delay elements and involves a latency of $N/4$ cycles. The author compared his architecture with R4SDC, R22SDC, RX4-B1, RX4-B2, RX4-B5 and XILINX IP on basis of throughput by slice ratio. The analysis of the proposed design shows the execution time to be 56% lower than obtained with Xilinx IP core and increase in 19% of throughput by area ratio for 256 Point FFT.

Dariusz Puchala at el.[23] compared the effectiveness of selected variants of Radix 2 FFT on GPU's and CPU's. The algorithm that are taken for consideration differ in memory consumption and data flow path arrangement that may affect the global memory coalescing and cache memory exploitation. The author claimed that we can achieve 30 times more acceleration in performing FFT on GPU's compared with CPU's for sufficiently large sized inputs. It was also claimed that FFT phase coefficients calculation and bit reversal permutation stages for GPU implementation highly outperforms the standard CPU implementation. Another observation shared by the author is that the algorithms categorized by the unified structures i.e. having identical stages are equally suited for both CPU's and GPU's.

Praveen Kumar Jhariya at el. [24] compared two FFT architectures simulated on FPGA. The designs are compared for 8 point input. The first design comprises of a butterfly unit and a complex multiplier that is used several times in execution of FFT. Whereas in the other design, 4 butterfly units along with 2 multipliers are used thrice in 8 point FFT execution. It was concluded in the paper that the area required by the first design is less but the latency of this design is comparatively higher as it took 12 butterfly cycles to compute 8 point FFT. On the other hand, second design shows comparatively higher performance but consumes a larger architectural area.

The summary of the above stated case study is as follows:

Sr No.	Researcher	Platform Used	Contribution/ Results Acquired/ Conclusions
I.	Feifei shen	GPU, CPU	<ul style="list-style-type: none"> • Demonstrates the advantages and validity of using GPUs for FFT over CUPs using large input size • GPU used is nvidia Gefores 8600

			<ul style="list-style-type: none"> • Experimental comparison for different input size were demonstrated for : <ul style="list-style-type: none"> ▪ open CV time, ▪ GPU time ▪ GPU maximum error
II.	Mokhtar A.Aboleaze	CPU	<ul style="list-style-type: none"> • Effect of number of registers in CPU to lessen the energy consumption in excessing the memory was investigated • It was concluded that number memory access depends upon: <ul style="list-style-type: none"> ▪ machine code ▪ compiler ▪ writing the memory.
III.	Peter A. Milder	Spiral, a tool that automatically generates corresponding hardware	<ul style="list-style-type: none"> • A generalized DFT with non-power of two input size was implemented using spiral, that is not application oriented • 4 FFT algorithms were discussed including: <ul style="list-style-type: none"> ▪ Pease FFT ▪ Iterative FFT ▪ Mixed radix FFT ▪ Bluestein FFT
IV.	Kota soloman Raju	FPGA	<ul style="list-style-type: none"> • Proposed a DFT and IDFT hardware architecture based on fluting point numbers to achieve accuracy and precision. • Hardware sharing scheme was introduced
V.	K Ganesoan	Intel 8086 microprocessor	<ul style="list-style-type: none"> • sequential FFT processor for 1024 points in 26.3msec. • Processor computes each butterfly operation in 64 basic clock cycles taking 5.12 μ sec • Works on 20KHz clock Rate
VI.	Gouhui		<ul style="list-style-type: none"> • Novel design for configurable FFT/IFFT module to provide scalability and reconfigurability. • unified radix structure for radix 2,3,4,5 & 7 is proposed. • This Fourier algorithm is proposed for cognitive radio communication and network systems
VII.	Muniandi Kannan		<ul style="list-style-type: none"> • Low power multiplier less radix 4, single path delay comutator pipelined FFT processor for Fixed point input. • complex multiplications are replaced by simple shifting and addition operations hence achieving low power butterfly architecture • 59% and 43% power reduction is achieved

			for 16 points and 64 points radix 4 FFT
VIII.	K. Indira Priyadarsini	VLSI	<ul style="list-style-type: none"> • Reconfigurable complex multiplier and bit parallel multipliers are used to store the twiddle factors and eliminating ROM • ROM less FFT processor • Processor is applicable for OFDM applications
IX.	Rozita Teymourzadeh	VLSI	<ul style="list-style-type: none"> • Floating Point FFT architecture is proposed • Proposed architecture is focused on speed, area, resolution, power consumption and latency • Proposed radix two floating points parallel pipelined FPP-FFT was optimized in AISC under Silterra and Mimos technology libraries • Clock Rate for the processor was calculated to be 227 MHz, whereas the latency was calculated to be 22 μs
X.	Archana Fande	VHDL	<ul style="list-style-type: none"> • Proposed high speed low power VLSI architecture and developed a low power complex multiplier for Radix 4 FFT and intern reduces the required hardware.
XI.	Yazan Samir Algnabi	MATLAB, FPGA	<ul style="list-style-type: none"> • Proposed a novel multiplier less pipelined architecture for Radix 22 SDF FFT • Digital slicing technique was used. • optimal constant multiplication arithmetic architectural design for multiplication is also proposed. • Proposed architecture is 3.35 times faster compared with the conventional architectures as it consumes only 20% of the conventional butterfly are
XII.	Yifan Bo		<ul style="list-style-type: none"> • Proposed variable length FFT processor for low power applications • Memory based architecture • tailored constant multiplier array is introduced to reduce power dissipation. • 64 to 8192 input FFT at 100MHz processing speed
XIII.	Atin Mukerjee	VHDL FPGA	<ul style="list-style-type: none"> • Radix 2 FFT architecture is introduced that reuses the same butterfly element several times • Area reduces by a factor of $\log_2 N$ for N point FFT • There is also a negligible increase in processing time for execution of the algorithm.
XIV.	Yousri Ourhani	FPGA	<ul style="list-style-type: none"> • An architecture with a novel memory sharing and dividing technique with

			<p>processing elements having parallel in parallel out capability is proposed</p> <ul style="list-style-type: none"> • Perform N Point FFT with 4/3 delay elements with a latency of N/4 cycles • Proposed design has execution time to be 56% lower compared with Xilinx IP core • increase in 19% of throughput by area ratio for 256 Point FFT
XV.	Dariusz Puchala	GPU, CPU	<ul style="list-style-type: none"> • Compared the effectiveness of selected variants of Radix 2 FFT on different platforms • Claimed to achieve 30 times more acceleration in performing FFT on GPU's compared with CPU's for sufficiently large sized inputs.
XVI.	Praveen Kumar Jhariya	FPGA	<ul style="list-style-type: none"> • Compared two FFT architectures for 8 bit input, simulated on FPGA • The cons and pros of both the architectures are discussed in detail by the author

After considering summarized inputs by different researchers of their time, we can say that the advancement in the modern technology was possible only because of the hindrances and bottlenecks faced by the world. These barriers laid the foundation for an advent of the new technology to the modern era. Different platforms were used that mainly depend upon its use in the application e.g. FPGA, GPU etc. Furthermore, with the passage of time as the technology gets inflated, the execution time using the same technique also became an issue that is addressed accordingly by the researchers of that time. Same is the case with the hardware area for the architecture and many other aspects of the same nature.

The main issue that was faced by several developers, analysts, scientists and researchers was that any up gradation in technology was more or less application limited. As any verity is added to the application there needs to reassemble all the architectural design. In other words we can say that the hardware is entirely reshaped as the application for which it was designed. In this thesis report we have presented an empirical model for signal processing Fast Fourier transform algorithm that can be reshaped as per the requirement of the user and application, or in other words we can say that the architecture can be reshaped as the requirements get changed that mainly depends upon the number of Processing Elements (PE) used. Using this technique for hardware designing, we can achieve parallelism, lesser hardware area, or any other efficiency in our hardware that designed with the same architectural blocks.

Chapter 3 : METHODOLOGY

This part of the thesis presents the steps for developing an optimal architecture for Fast Fourier Transform for an n-bits m-points input using p-processing elements. The proposed architecture is briefly explained in this section. Furthermore the execution, working and architectural design of the processing element will also be discussed. With the help of several hardware architectures and its analysis, an empirical model based upon the several hardware parameters for FFT will be developed using machine learning algorithm. This model will be our empirical model for n-bits m-points FFT. With the help of this generated model we will be able to predict the futuristic parameters of the desired FFT hardware design and feasibility of its implementation, its characteristics and other parameters before actual implementing it.

3.1 Empirical Model for FFT

An empirical model is a generalized way for representation of a designed prototype for the activities on basis of observation and experiment. It is generally used to represent hierarchy of the results of the performed experiments. Furthermore, it is used for predicting the futuristic results from the previously calculated dataset. Output requirements of the hardware will be calculated by a decision tree i.e. either it has to consider any one of the hardware parameter for optimization or a generalized hardware having a trade of for all the parameters for generalized implementation as per ones requirement.

In this report we are considering following hardware parameters for making our empirical model i.e. number of Slice LUT's, LUT FF Pairs, clock rate The inputs of the system will be 'm' i.e. point of FFT, desired Slice LUT's, LUT FF Pairs, clock rate and clock cycles in which one wants the hardware to perform the FFT and the output will include the actual number of Processing Elements, Slice LUT's, LUT FF Pairs, clock rate and clock cycles required to perform that m-point FFT for that architecture.

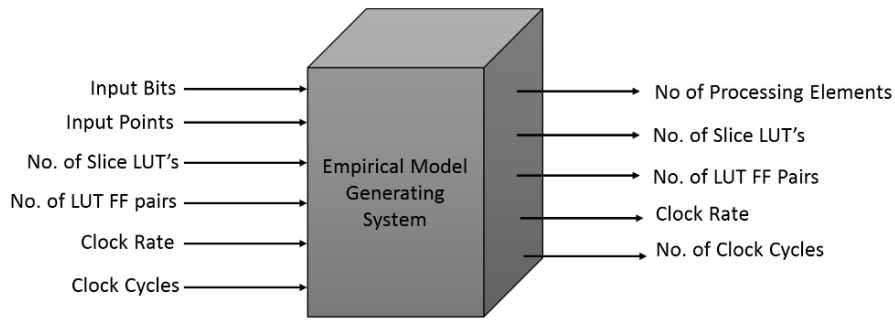


Figure 3-1: Inputs/Outputs of Empirical Model

The model will provide the user with the nearest possible actual hardware specifications. Most probably, it is quite possible to have a difference in the actual and required specifications, but the model will provide the user with the error of the actual and required specifications. After that it will be upto the user/designer of the architecture that how much he can trade off on any hardware parameter that is either required hardware area, number of clock cycles required to perform the desired task or clock rate or something else.

3.1 Proposed Hardware Architecture

The proposed hardware architecture is based upon multiprocessor architecture (p-processing elements), of homogenous PEs. These PEs are connected by a crossbar switch that acts as a back bone for our architecture, shown in Figure 3-2.

A crossbar switch is generally an assembly of switches between inputs and outputs. The switches are arranged in a matrix. If a crossbar switch has M inputs and N outputs, then it has $M \times N$ matrix cross-points where the connections cross. It is a matrix where each crossbar switch runs between two points, in a design that is intended to hook up each part of an architecture to every other part. A crossbar switch finds its applications in various disciplines as on network and system on chips (NOC & SOC) [25,26], in network processors that uses rotating round robin algorithm [27], reconfigurable crossbar switches in network processors to increase the performance and flexibility for multiprocessors and computer clusters[28], integrated designs and much more.

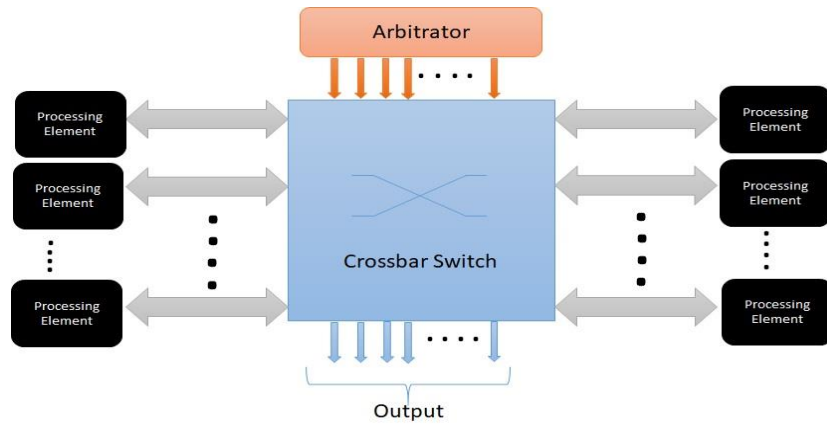


Figure 3-2: Block Diagram of Proposed Architecture

The inputs for the processing element vary for every next cycle. In the stated architecture a crossbar switch routes the input towards particular processing element, either coming from original input (as in stage 1 of Figure 2-1) or some output that is treated as input in the later cycles (Stage 2 and later of Figure 2-1). The routing of particular inputs towards a specified processing element is achieved using an arbitrator (Figure: 3-2). This arbitrator generates the control signals that controls the of data from one processing element to another. These control signals and its working will be discussed later on (Table:1-5).

From Figure 2-1 it can be observed that maximum number of Processing elements that can be used for parallel execution for m -point FFT are $\frac{m}{2}$, as for radix 2 FFT every PE has 2 inputs and after that, wastage of hardware resources will occur in every executing cycle, as in every clock cycle, there will be some part of architecture without any assigned task. On the other hand, by using number of P.E. $\leq \frac{m}{2}$, all the hardware resources are used. For FFT implementation, the architecture should be capable to feedback the generated outputs. To reuse the output of the system again as input, demultiplexers are used so that the output generated can be routed towards the particular input line for further usage if required/needed using demux select line.

To understand the working of the model let us suppose a 4 point input with 1 PE. In the first cycle, original inputs will be forwarded towards original muxes from back muxes (Figure 3-3) by using all the select lines as 0. The back mux has three inputs, one is the original input, whereas other two are routed back inputs from the respective demultiplexers. The original muxes have all the concerned inputs that will take part in FFT execution. The input at 0 and 2 are firstly put forward using the select lines. In the next cycle of execution, other two inputs (1 & 3) are given to the PE from original muxes (using select lines as 1 and 3). The outputs obtained in the respective cycles, are feedback through demultiplexers by

using the select lines same as used for muxes (labelled as original muxes in Figure 3-3). In this way the first stage of 4 point FFT with 1 PE is executed. After the completion of the first stage, in the second stage, system has three types of inputs in which the first is that of original input, that will not take part in further execution of FFT, whereas the other two are routed back inputs from the respective demultiplexers. For processing the 2nd stage, three type of inputs are taken into account in back mux. To select the concerned input for processing i.e. either original input or previously generated outputs (out1 or out2), select lines are used. These select lines put forward the input to the original multiplexer. In the first half, the select lines will be 0 & 1 whereas in the next half it will be 2 & 3. As stated the select lines for demuxes are same as original muxes in the respective cycles to put forward the generated output to its initial place. The described architecture is shown in Figure 3-3. In this way a FFT for 4 Point input with 1 PE is processed.

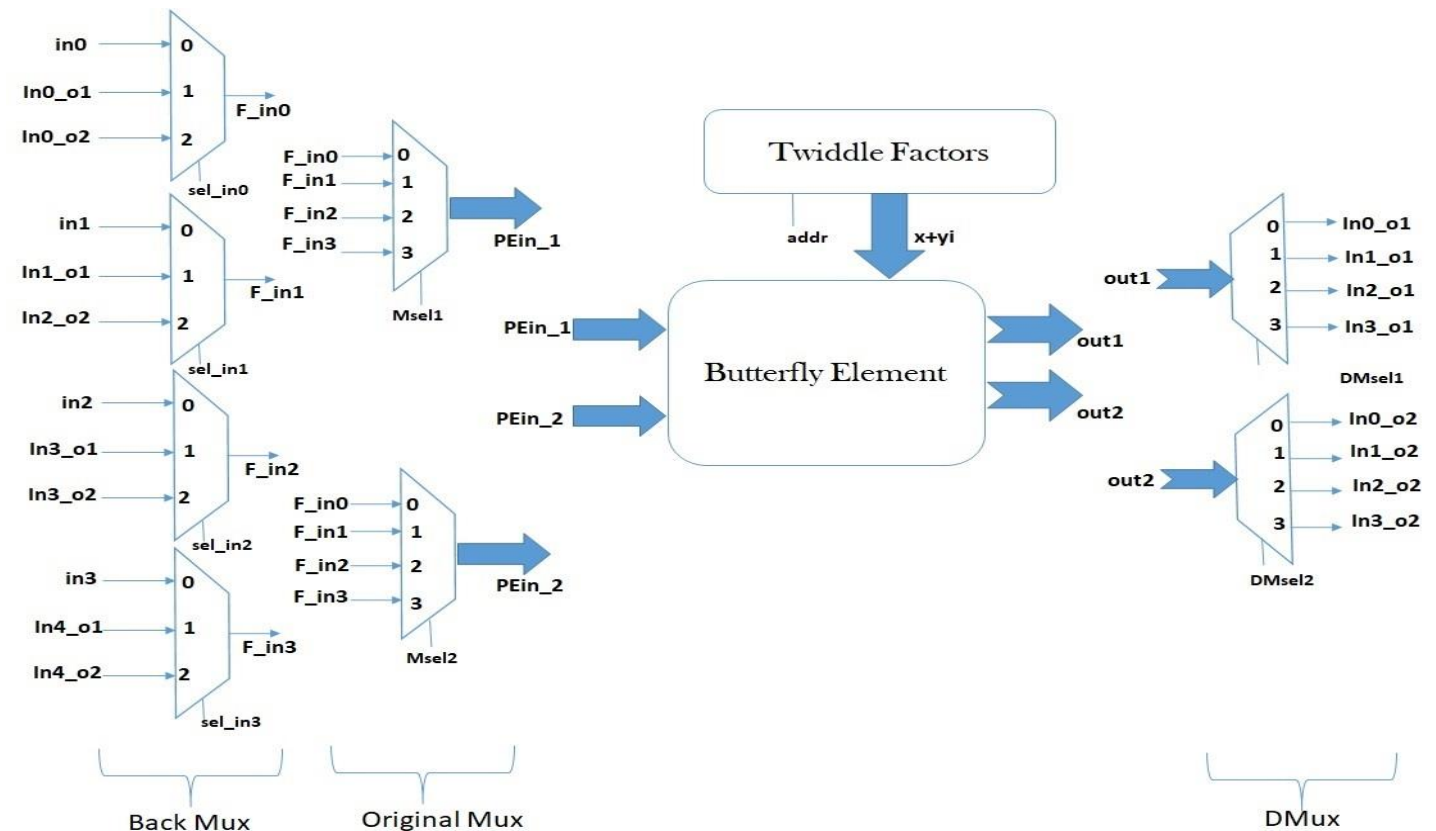


Figure 3-3: Hardware Implementation of 4 Point FFT with 1 Processing Element

3.1.1 Basic Building Block / Butterfly Processing Element

From the Figure 2-1, it can be noticed the main processing element for radix 2 hardware implementation of FFT illustrated are two adders and a multiplier. This is shown in Figure 3-4, that acts as a building block for FFT execution, known as butterfly processing element.

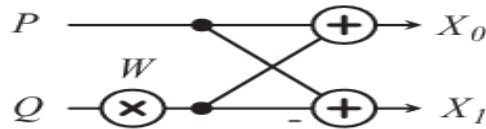


Figure 3-4: Building Block for FFT (Butterfly Processing Element)

It can be noted here that the system has two inputs P & Q , a twiddle factor W and two outputs X_0 and X_1 . In FFT the inputs and twiddle factors play the major (Pivotal) role in processing of butterfly element, this basic building block of its architecture that can be implemented as

$$X_0 = \text{out1} = \text{in1} + (\text{in2} * \text{wn}) \dots (3A)$$

$$X_1 = \text{out2} = \text{in1} + (\text{in2} * -\text{wn}) \dots (3B)$$

Let $\text{out1} = x_1 + y_1i$ & $\text{out2} = x_2 + y_2i$

And $\text{in1} = a + bi$, $\text{in2} = c + di$ & $\text{wn} = e + fi$

From (3A) & (3B)

$$\begin{aligned} x_1 + y_1i &= (a + bi) + ((c + di) * (e + fi)) \\ &= (a + ce - df) + (b + cf + de)i \dots (4A) \end{aligned}$$

$$\begin{aligned} x_2 + y_2i &= (a + bi) + ((c + di) * (-e - fi)) \\ &= (a - ce + df) + (b - cf - de)i \dots (4B) \end{aligned}$$

3.1.2 Butterfly Processing Element Architecture

The outcome of equations (4A) & (4B) are the two outputs of the PE, acting as building block (butterfly processing element) for FFT. The hardware implementation of butterfly element (4A-B) is shown in Figure 3-5:

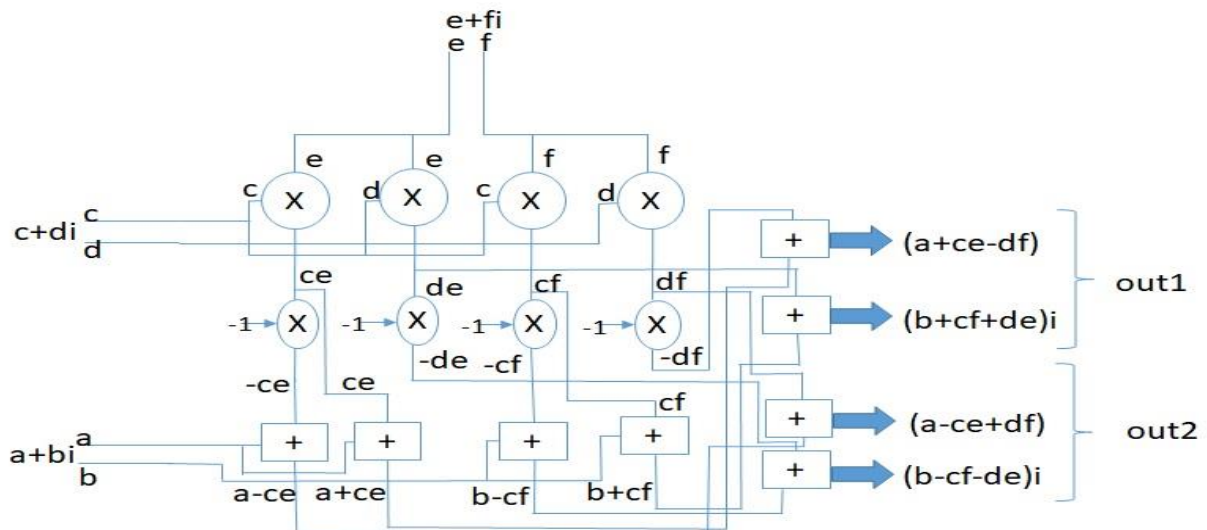


Figure 3-5: Butterfly Element Architecture Design

In the stated butterfly element $a + bi$ and $c + di$ are the two inputs that are participating in execution of FFT whereas $e + fi$ is the specific twiddle factor.

3.1.3 Field Programmable Gate Array (FPGA) Implementation

The FPGA architecture design after implementing the above model for 4 point FFT using 1 processing element is depicted below in Figure 3-6. The stated hardware design has signal to perform FFT and select lines as input and four output lines, as there is only one processing element to perform FFT.

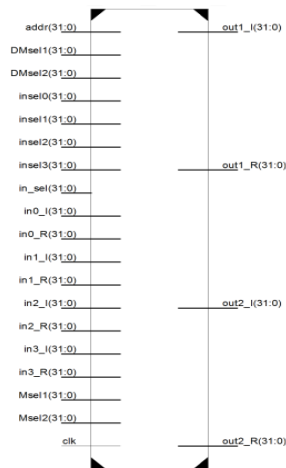


Figure 3-6: FPGA design for 4 point 1 PE

The detailed Register Transfer Level (RTL) schematic for the above architecture is shown in the figure below. Here the muxes, demuxes and processing elements can be seen properly that forms the basis of our architectural model.

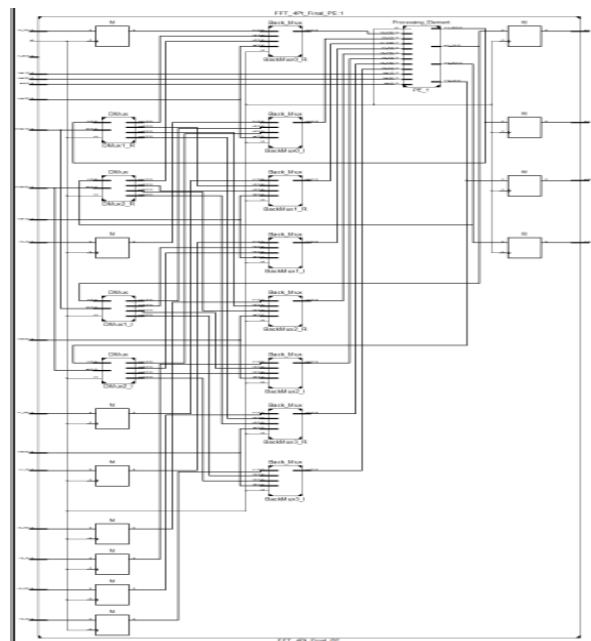


Figure 3-7: Detailed RTL Schematic for 4 Point 1 PE Architecture

In the same way, using 2 processing elements for 4 point FFT, the FPGA design is as follows:

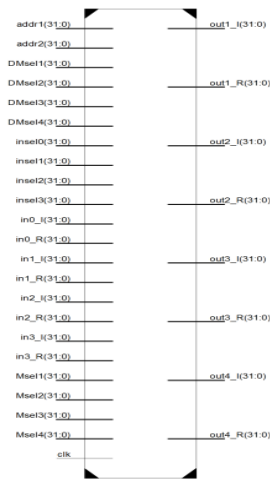


Figure 3-8: FPGA design for 4 point 2 PE

The only difference that can be observed in performing 4 point FFT using 1 and 2 processing elements are, increase in number of muxes and demuxes resulting in increasing the number of select lines. Whereas on the output side, as the number of processing elements are doubles the output lines are increased accordingly. The detailed RTL Schematic for 4 point architecture using 2 processing elements is shown in the figure below:

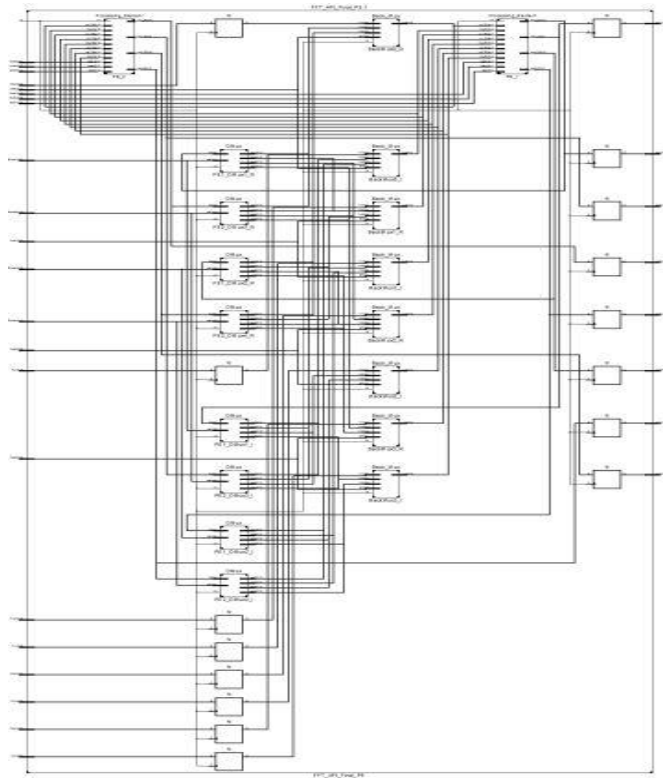


Figure 3-9: Detailed RTL Schematic for 4 Point 2 PE Architecture

Both the processing elements are shown in the above diagram. Furthermore, the increase in number of select and output lines can be observed clearly as the number of processing elements are increased.

Same methodology has been followed for n-Bits input of m-Points input with p-Processing Elements. The select lines used for 4 and 8 Point FFT using p-Processing Elements are shown in Table(1-5).

3.1.4 Select Lines for Proposed Architecture

The select lines for 4 Point FFT with 1 PE for back mux, original mux, memory element and demux for the architecture in (Figure 3-3 & Figure 3-7) is shown in Table (1)

Clk Cycle	sel_in0	sel_in1	sel_in2	sel_in3	Msel1	Msel2	addr	DMsel1	DMsel2
1	0	0	0	0	0	2	1	X	X
2	0	0	0	0	1	3	2	0	2
3	1	1	2	2	0	1	3	1	3
4	1	1	2	2	2	3	4	0	1
XXX	Z	Z	Z	Z	Z	Z	Z	2	4

Table 1: Select Lines for 4 Point 1 Processing Element

By increasing the processing elements from 1 to 2, another twiddle factor select line i.e. ‘addr’, two ‘Msel’ lines for multiplexers and two ‘DMsel’ select lines for demultiplexers will be added in the architecture as shown in Figure 3-9, is depicted in Table(2)

Clk Cyc	sel_in 0	sel_in 1	sel_in2	sel_in 3	Msel 1	Msel 2	Ms el3	Ms el4	add r	addr 1	DMsel 1	DMsel 2	DMsel 3	DMsel 4
1	0	0	0	0	0	2	1	3	1	2	X	X	X	X
2	1	1	2	2	0	1	2	3	3	4	0	2	1	3
X	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	0	1	2	3

Table 2: Select Lines for 4 Point 2 Processing Element

It should be noted here that in first cycle demux select lines are not initiated, being the input that is in processing stage. When the processing of the first cycle is successfully done, then it will be feed backward in the next cycle to its initial place. After the completion of the cycle new inputs are forwarded with the help of select lines and the outputs are moved back to their initial places using the previously used mux lines. In other words we can also say that the current select lines are of input multiplexers, after passing through processing element, will be select lines for demux in the next cycle

For 8 point FFT the select lines using one, two and four processing elements are shown in Table3, Table4 and Table 5 respectively. The select lines for the twiddle factor ‘addr’, two ‘Msel’ Lines for multiplexers and two ‘DMsel’ select lines would be added in the architecture, in a similar way as in Table(1-2).

Clk. Cyc	sel_i n0	sel_i n1	sel_i n2	sel_i n3	sel_i n4	sel_i n5	sel_i n6	sel_i n7	Ms el1	Ms el2	ad dr	DMs el1	DMs el2
1	0	0	0	0	0	0	0	0	0	4	1	X	X
2	0	0	0	0	0	0	0	0	2	6	2	0	4
3	0	0	0	0	0	0	0	0	1	5	3	2	6
4	0	0	0	0	0	0	0	0	3	7	4	1	5
5	1	1	1	1	2	2	2	2	0	2	5	3	7
6	1	1	1	1	2	2	2	2	4	6	6	0	2
7	1	1	1	1	2	2	2	2	1	3	7	4	6
8	1	1	1	1	2	2	2	2	5	7	8	1	3
9	1	1	2	2	1	1	2	2	0	1	9	5	7
10	1	1	2	2	1	1	2	2	2	3	10	0	1
11	1	1	2	2	1	1	2	2	4	5	11	2	3
12	1	1	2	2	1	1	2	2	6	7	12	4	5
xxx	Z	z	z	z	z	z	z	z	z	z	z	6	7

Table 3: Select Lines for 8 Point 1 Processing Element

By increasing the number of P.E. the number of clock cycles required to perform FFT will lessen to half (Equation (5)), but the select lines will increase accordingly

Clk Cyc	sel_in0	sel_in1	sel_in2	sel_in3	sel_in4	sel_in5	sel_in6	sel_in7	Ms el1	Ms el2	Ms el3	Ms el4	ad dr1	ad dr2	DM sel1	DM sel2	DM sel3	DM sel4
1	0	0	0	0	0	0	0	0	0	4	2	6	1	2	X	X	X	X
2	0	0	0	0	0	0	0	0	1	5	3	7	3	4	0	4	2	6
3	1	1	1	1	2	2	2	2	0	2	4	6	5	6	1	5	3	7
4	1	1	1	1	2	2	2	2	1	3	5	7	7	8	0	2	4	6
5	1	1	2	2	1	1	2	2	0	1	2	3	9	10	1	3	5	7
6	1	1	2	2	1	1	2	2	4	5	6	7	11	12	0	1	2	3
xxx	z	z	z	z	z	z	z	z	z	z	Z	Z	z	z	4	5	6	7

Table 4: Select Lines for 8 Point 2 Processing Element

Clk Cy c	sel_in 0	sel_in 1	sel_in 2	sel_in 3	sel_in 4	sel_in 5	sel_in 6	sel_in 7	M sel 1	M sel 2	M sel 3	M sel 4	M sel 5	M sel 6	M sel 7	M sel 8	ad dr 1	ad dr 2	Ad dr 3	ad dr 4
1	0	0	0	0	0	0	0	0	0	4	2	6	1	5	3	7	1	2	3	4
2	1	1	1	1	2	2	2	2	0	2	4	6	1	3	5	7	5	6	7	8
3	1	1	2	2	1	1	2	2	0	1	2	3	4	5	6	7	9	10	11	12

Table 5: Select lines for 8 Point 4 Processing Element

The DMux select lines are in same hierarchy as in previous tables i.e. same as Msel, having number input value in the first cycle but in the second cycle, it would be same as previously used Msel select line. Keeping in view the above stated select lines in the above mentioned tables, same hierarchy will be followed for 16 point, 32 point, up till m-point FFT. All these (Table1-5) acts as the select lines that are used by the arbitrator that act as state machine. These distribution and routing of data for the architecture is managed via control signals received from control unit that acts as state machine for the stated hardware architectures.

3.2 Working of Empirical Model

To make the empirical model, different architectures similar to the architecture stated in Figure: 3-3 for $n\text{-Bits} \in \{8, 16 \text{ and } 32\}$, $m\text{-Point} \in \{4, 8, 16 \text{ and } 32\}$ with $p\text{-processing elements} \in \{1, 2, 4 \text{ and } 8\}$ were modeled and their comparative parameters were taken into consideration for the modelling of the empirical model. The acquired results, are accumulated and an algorithm of machine learning known as regression algorithm of order 1, 2 and 3 was implemented.

Regression analysis is a form of predictive modelling technique which calculates the relationship between a target (i.e. dependent variable) and predictor (i.e. independent variable) in form of a function. This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables and much more. Regression technique actually generates a function of order $1, 2, 3, \dots, n^{\text{th}}$, which is a linear curve in case of 1st order, quadratic in case of 2nd order and cubic in case of order 3, and this goes up till n^{th} order generation. The implementation of linear, quadratic and cubic regression on a random dataset is shown in figure below, however, the use of this technique in case of making the empirical model will be discussed shortly in the upcoming chapter.

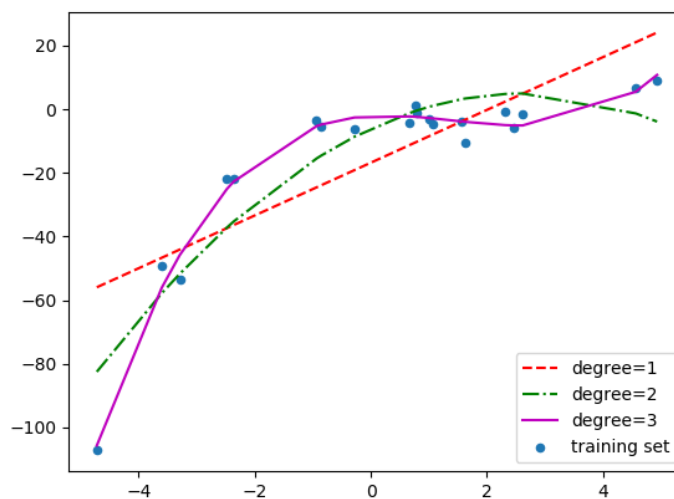


Figure 3-10: Application of Linear, Quadratic and Cubic Regression on a Random Dataset [29]

The inputs will be passed through the empirical model and will predict the requirement of for the futuristic parameters if only one parameter is taken into consideration. Whereas on the other hand if two or more parameters are taken for granted, the hardware will be calculated for all the set parameters and an optimized specifications of hardware will be generated, having tradeoff for all the set parameters.

As discussed earlier, the clock cycles required by the proposed architecture depends upon the number of processing elements. With the increase in number of processing elements the clock cycles will lessen accordingly or vice versa. The clock cycles required to perform FFT for m-point using p-processing elements can be calculated using eq. 5 as below:

$$\text{For } \frac{m}{2 * x} \text{ Processing Elements } \rightarrow (\log_2 m) * x \text{ "Clock Cycles" ; } x = 1, 2, \dots, \frac{m}{2} \quad (5)$$

Eq. 5 is a generalized representation for clock cycles required to perform m-Point FFT having $\frac{m}{2 * x}$ number of Processing Elements, it would take $(\log_2 m) * x$ clock cycles, where: $x = 1, 2, \dots, \frac{m}{2}$. As we are catering here Radix 2 FFT architecture that why we use \log_2 , if we propose an architecture of Radix n, then similarly \log_n will be used in equation 5.

To execute the empirical model for actual implementation and prediction of the parameters of the futuristic architecture either taking into account a single parameter or all the stated parameters, following steps are to be followed:

1. Take number of input bits and points for FFT (Input)
2. Take desired parameters Slice LUT's, LUT FF Pars, clock rate and clock cycle (Input: Take either 1 or more as required)
3. If (ii) has only 1 parameter: Pass the number of points of FFT from (i) through the specific model (Table 6) and deliver the output.
4. Pass the number of Processing Elements from (iii) and calculate clock cycles using equation (5) of the hardware.
5. If (ii) has 2 or more parameters: Pass the number of points for FFT from (i) through all the specific models (Linear Regression model for Slice LUT and LUT-FF Pairs, Quadratic Regression for Clock Rate) for n-bit input (Table 6).
6. Repeat step (iv) for all the models generated in (v) to calculate the clock cycles.
7. Calculate the percentage error between the outputs from (v& vi) with desired parameters from (ii).
8. Output the model with the least percentage error.
9. Pass the number of Processing Elements from (vii) and calculate clock cycles using equation (5).

In this section of the thesis we have presented an architecture for m-points with p-processing elements. In this architecture, the processing element that acts as the basis of the hardware is actually the butterfly architecture of the FFT that is briefly elaborated in the given section. The architectural design of the processing element that is the main building block of the hardware design, is also shown in this section.

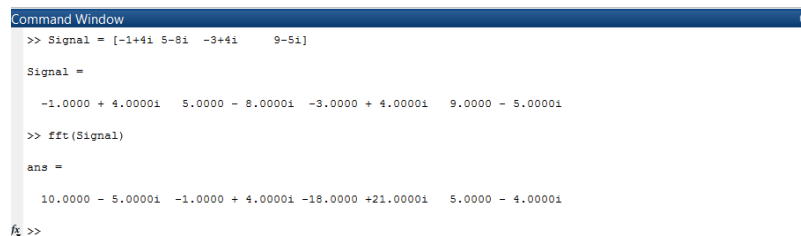
In addition to this, the select lines that are the main fundamentals of the proposed architecture which helps in routing and selection of the selected input is also shown in this section. Furthermore the stepwise processing and execution of the empirical model is also encountered. In addition to this a generalized formula to calculate the number of clock cycles required to perform FFT with variable number of processing elements is also proposed in this section. However experimental results, empirical model and its working is elaborated in the next chapters.

Chapter 4 : EXPERIMENTAL RESULTS

4.1 Results and Discussion

In order to make our empirical model for specified hardware parameters for proposed FFT architecture, several experimentation for the implementation of hardware architecture were performed similar to architecture stated in the Figure (3-3, 3-7 & 3-9) using n-bit $\in \{8, 16 \text{ and } 32\}$, m-Points for FFT $\in \{4, 8, 16 \text{ and } 32\}$ with p-processing elements $\in \{1, 2, 4 \text{ and } 8 \text{ PE's}\}$. All the possible combinations of n, m and p were designed. The platform used for designing and implementation of all the architectures is on Xilinx using HDL Verilog coding. The FPGA used for this purpose is XC7A100T that mainly belongs to Artix family of FTG256 package.

The hardware designs for 4 point input FFT using 1 and 2 processing elements are discussed in the previous section. For confirming the feasibility of our architecture, we have performed FFT algorithm on our model. The bench mark to decide, whether our model is working properly or not, we have executed FFT on a complex signal on MATLAB and then confirmed it on our model. The 4 point complex signal FFT on MATLAB is shown in Figure 4-1. Signal to perform FFT is $[-1+4i \quad 5-8i \quad -3+4i \quad 9-5i]$.



```
Command Window
>> Signal = [-1+4i 5-8i -3+4i 9-5i]

Signal =

-1.0000 + 4.0000i  5.0000 - 8.0000i  -3.0000 + 4.0000i  9.0000 - 5.0000i

>> fft(Signal)

ans =

10.0000 - 5.0000i  -1.0000 + 4.0000i  -18.0000 +21.0000i  5.0000 - 4.0000i

fx >>
```

Figure 4-1: MATLAB FFT Complex Signal Execution

To wave form achieved using 1 processing element for 4 points input for the same signal using the hardware design stated in Figure 3-7 is shown below:

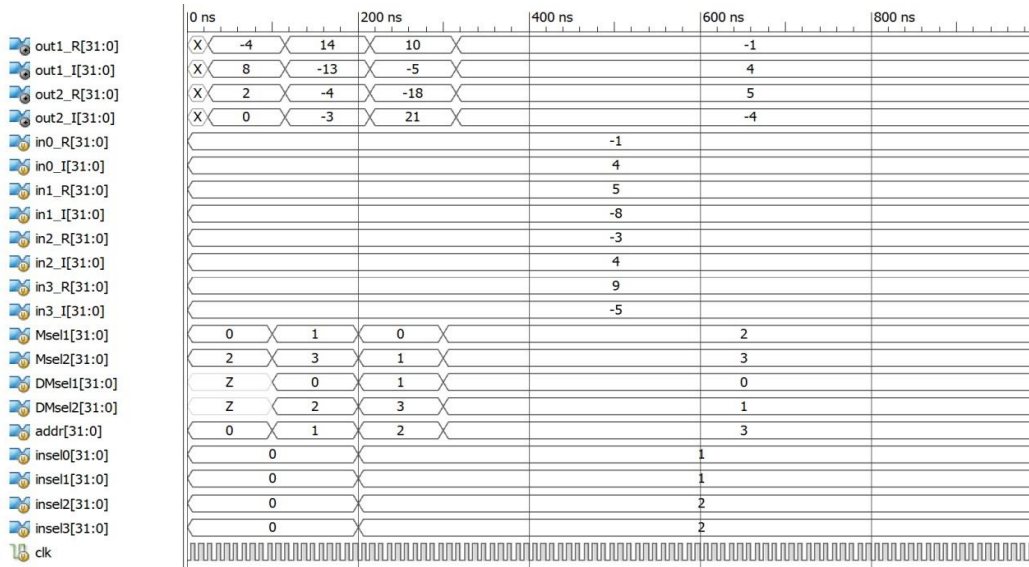


Figure 4-2: Generated Wave Form for 4 Point FFT using 1 PE

In addition to this, using 2 processing elements, the output wave form from the architectural design stated in Figure 3-9 is shown below.

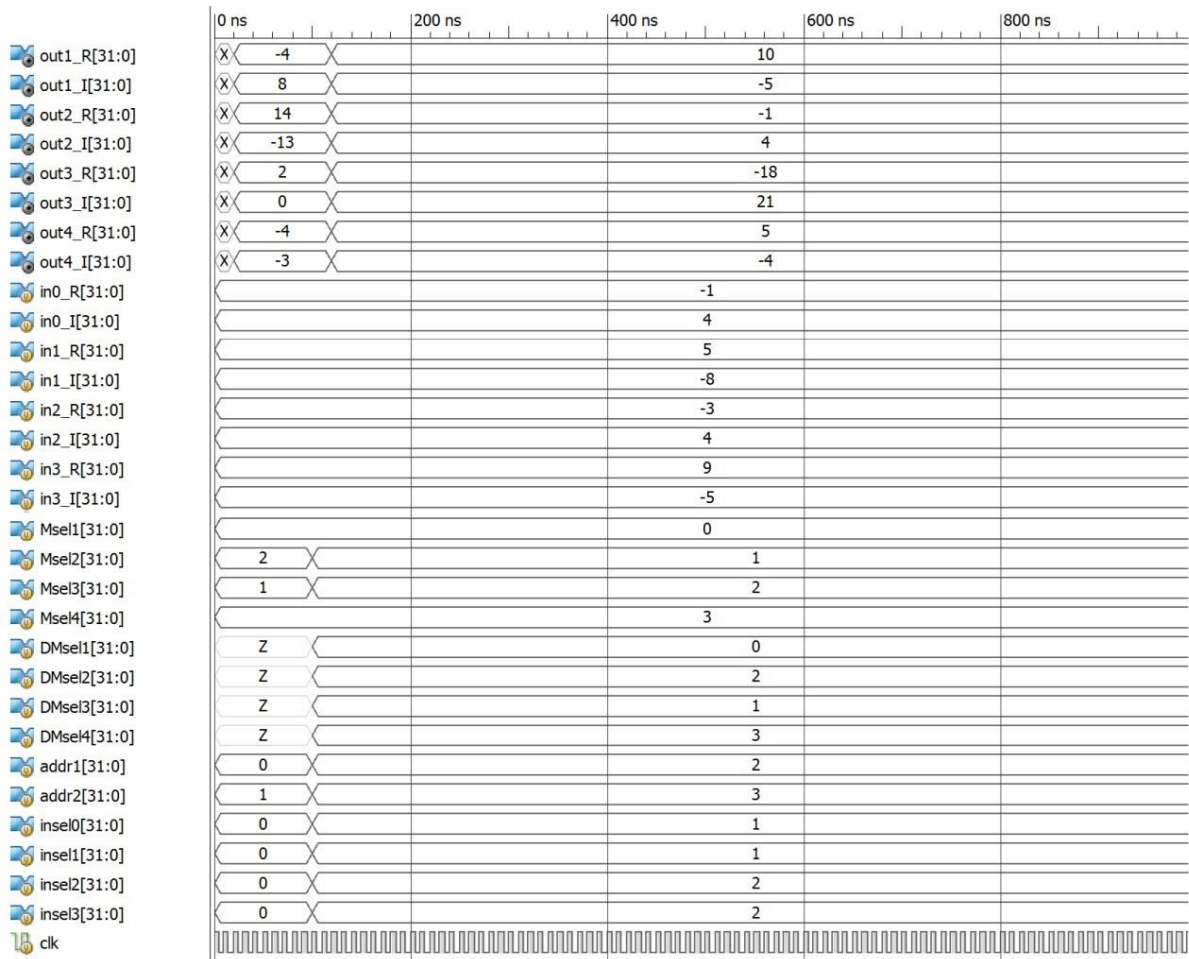


Figure 4-3: Generated Wave Form for 4 Point FFT using 2 PE

From both the above Figures (4-2 & 4-3), the output signal is same. The only difference that lies in between the two wave forms is that, if we use 1 PE, the output will be achieved in two cycles whereas using 2 PE, the output is received in a single cycle.

The performance of all the implemented architectures were evaluated on the basis of hardware parameters i.e. number of slice LUT's, LUT FF Pairs and clock frequency of that particular specified characteristic architecture. These three parameters were analyzed for all the possible architectures for all the possible combinations of n, m and p.

For 8 bit input of $m\text{-point} \in \{ 4, 8, 16 \text{ and } 32 \}$ FFT using $p\text{-Processing Elements} \in \{ 1, 2, 4 \text{ and } 8 \text{ PE's} \}$. Making constant input as 8 bit and variable m-points and p-processing elements we came up with the results as shown in Figure 4-4.

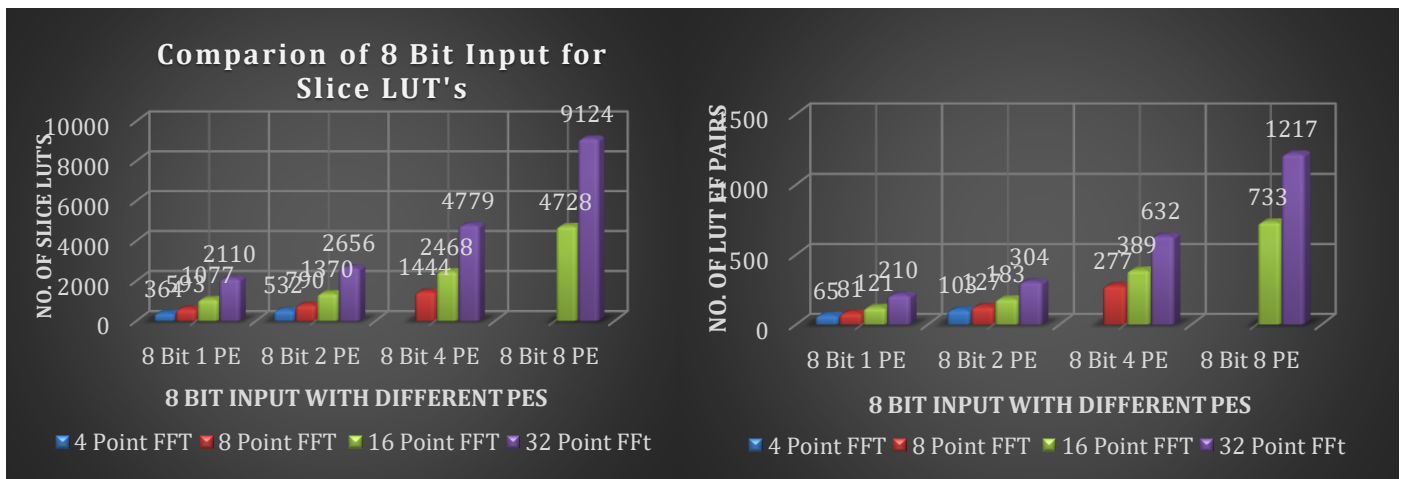


Figure (A)

Figure (B)

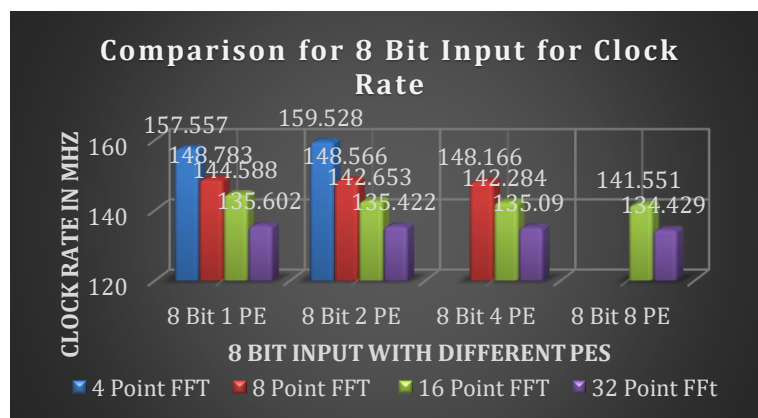


Figure (C)

Figure 4-4: Comparison of 8 bit input for m-Point FFT on basis of Figure (A): Slice LUT's ; Figure (B) LUT FF Pairs; Figure (C) Clock Rate

In the same way by increasing the input bits different results are achieved. For instance, using 16 bit input with variable input point FFT and Processing Elements, the architecture specifications are used in the following manner as depicted below:

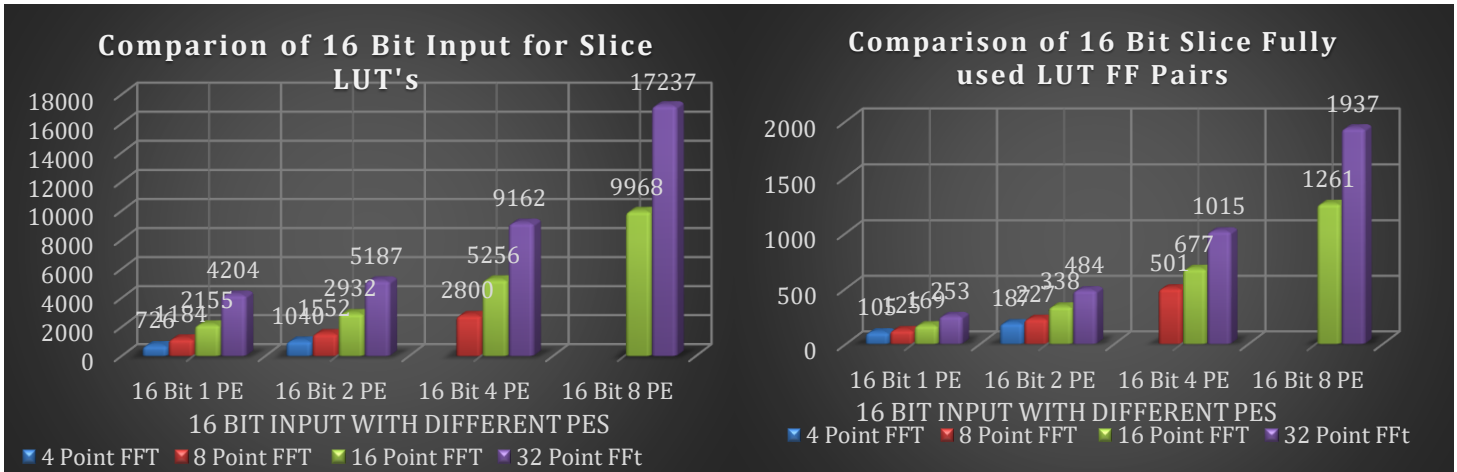


Figure (A)

Figure (B)

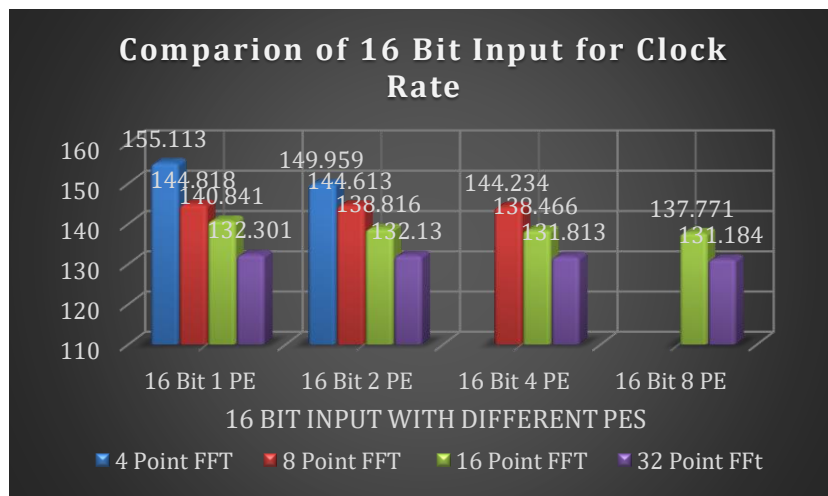


Figure (C)

Figure 4-5: Comparison of 16 bit input for m-Point FFT on basis of Figure (A): Slice LUT's ; Figure (B) LUT FF Pairs; Figure (C) Clock Rate

In addition to this different results are obtained when we increase the number of input bits. This is also noted that the hierarchy of the results are similar to that of the previously found results with different input bit size. The observations for different hardware parameters for 32 bit input are shown below.

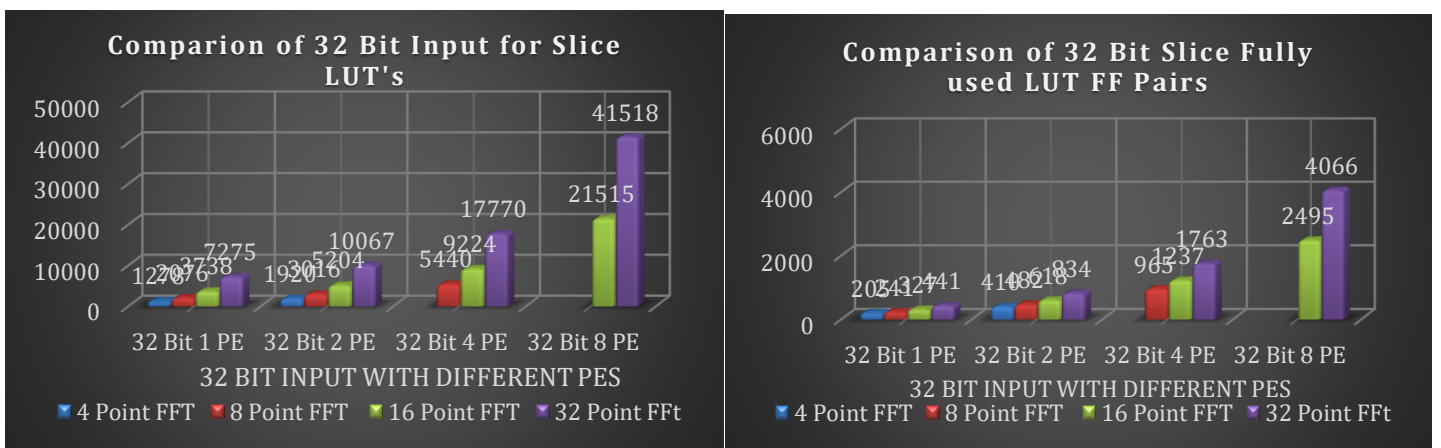


Figure (A)

Figure (B)

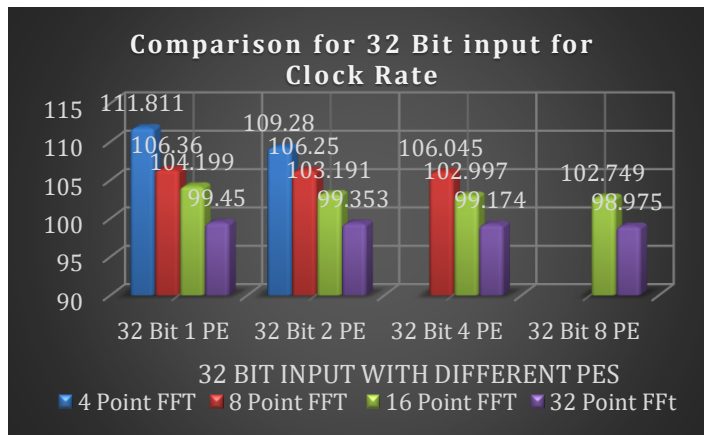


Figure (C)

Figure 4-6: Comparison of 32 bit input for m-Point FFT on basis of Figure (A): Slice LUT's ; Figure (B) LUT FF Pairs; Figure (C) Clock Rate

Graphical analysis for Slice LUT's of the hardware implemented in Figure 4-4 (A), Figure 4-5 (A) and Figure 4-6(A) shows that it increases as the number of point for FFT increases. In addition to this same trend was followed by the LUT FF Pairs Figure 4-4 (B) Figure 4-5(B) and Figure 4-6(B). Whereas unlike other two, clock rate decreases as the number of points for FFT increased Figure 4-4(C) Figure 4-5(C) and Figure 4-6(C). Another observation that was made in the clock rate was that, by increasing the processing elements the clock rate does not show a valuable change, infact it shows rigidity in its clock rate Figure 4-4(C) Figure 4-5(C) and Figure 4-6(C).

For a generalized empirical model for the architecture similar to Figure 3-3, the data acquired from several experiments for distinct specifications i.e. different input bits points and processing elements, were passed through a conventional machine learning algorithm i.e. regression algorithm of order 1, (linear regression), order 2 (quadratic regression) and order 3(cubic regression). The execution of this algorithm for 8 bit input using 1 Processing Element for the stated parameters are depicted in Figure 4-7.

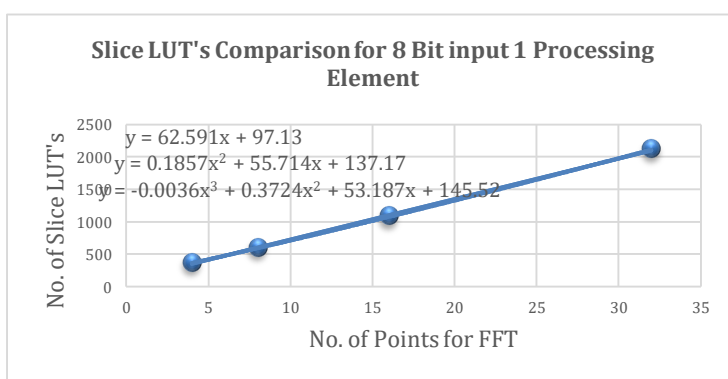


Figure (A)

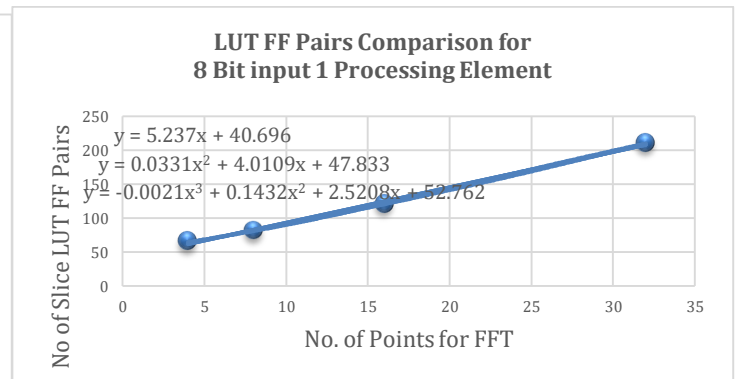


Figure (B)

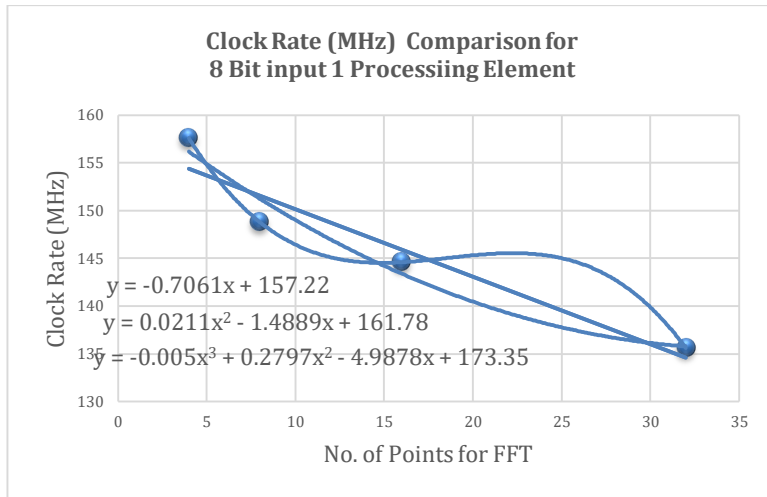


Figure (C)

Figure 4-7: Regression Technique for 8 Bit Input and 1 Processing Element for m-Point FFT on Figure (A): Slice LUT's ; Figure (B) LUT FF Pairs; Figure (C) Clock Rate

From Figure 4-7 it is observed that after applying linear, quadratic and cubic regression, there exists a very minor difference in their output for slice LUT's and LUT FF Pairs compared to original ones, whereas the data highly under fits using linear regression, whereas the data highly over fits by using cubic regression while considering the clock rate. So for representing the clock rate, quadratic regression is the most nominal and gives better results compared with the other two. The equation of linear, quadratic and cubic regression can be seen on the generated graphs (Figure: 4-7).

In view of the above we can say that, the best model that depict the hierarchy of the acquired data for making the empirical model in case of slice LUT's and LUT FF pairs are the first order polynomial (i.e. Linear Regression) whereas in case of clock rate its of second order equation (i.e. Quadratic Regression).

Graph plot and equations for the above mentioned hardware characteristics are for 8 bit input constitutes the same hierarchy for n bit input. In the same way, by applying the regression technique on n-bits $\in \{8,16 \text{ and } 32\}$, m-Points $\in \{4,8,16 \text{ and } 32\}$ for processing elements $\in \{1, 2, 4, \text{ and } 8\}$ following equations/results are obtained as in Table 6:

		ith-Order Model								
		Number of Slice LUTs			Number of Fully used LUT FF-Pairs			Clock Rate MHz		
		Linear	Quadratic	Cubic	Linear	Quadratic	Cubic	Linear	Quadratic	Cubic
8 Bit s	1 Processing Element	$y = 62.591x + 97.13$	$y = 0.1857x^2 + 55.714x + 137.17$	$y = -0.0036x^3 + 0.3724x^2 + 53.187x + 145.52$	$y = 5.237x + 40.696$	$y = 0.0331x^2 + 4.0109x + 47.833$	$y = -0.0021x^3 + 0.1432x^2 + 2.5208x + 52.762$	$y = -0.7061x + 157.22$	$y = 0.0211x^2 - 1.4889x + 161.78$	$y = -0.005x^3 + 0.2797x^2 - 4.9878x + 173.35$
	2 Processing Element	$y = 76.391x + 191.13$	$y = 0.3827x^2 + 62.214x + 273.67$	$y = -0.0036x^3 + 0.3724x^2 + 53.187x + 145.52$	$y = 7.237x + 70.696$	$y = 0.0331x^2 + 6.0109x + 77.833$	$y = -0.0021x^3 + 0.1432x^2 + 4.5208x + 82.762$	$y = -0.7607x + 157.95$	$y = 0.0369x^2 - 2.129x + 165.92$	$y = -0.0055x^3 + 0.3216x^2 - 5.9804x + 178.66$
	4 Processing Element	$y = 139.74x + 288.5$	$y = 0.6849x^2 + 111.56x + 507.67$	$y = 0.6849x^2 + 111.56x + 507.67$	$y = 14.848x + 155.5$	$y = 0.0495x^2 + 12.812x + 171.33$	$y = 0.0495x^2 + 12.812x + 171.33$	$y = -0.5312x + 151.76$	$y = 0.0119x^2 - 1.0209x + 155.57$	$y = 0.0119x^2 - 1.0209x + 155.57$
	8 Processing Element	$y = 274.75x + 332$	$y = 274.75x + 332$	$y = 274.75x + 332$	$y = 30.25x + 249$	$y = 30.25x + 249$	$y = 30.25x + 249$	$y = -0.4451x + 148.67$	$y = -0.4451x + 148.67$	$y = -0.4451x + 148.67$
16 Bit s	1 Processing Element	$y = 124.67x + 197.17$	$y = 0.3261x^2 + 112.59x + 267.5$	$y = -0.0105x^3 + 0.8672x^2 + 105.27x + 291.71$	$y = 5.3043x + 83.435$	$y = -0.002x^2 + 5.379x + 83$	$y = -0.0019x^3 + 0.0938x^2 + 4.0833x + 87.286$	$y = -0.7174x + 154.03$	$y = 0.0266x^2 - 1.7039x + 159.77$	$y = -0.0062x^3 + 0.3476x^2 - 6.047x + 174.14$
	2 Processing Element	$y = 149.58x + 434.04$	$y = -0.5049x^2 + 168.28x + 325.17$	$y = -0.1794x^3 + 8.7318x^2 + 43.312x + 738.52$	$y = 10.696x + 148.57$	$y = -0.1139x^2 + 14.915x + 124$	$y = -0.0186x^3 + 0.8438x^2 + 1.9583x + 166.86$	$y = -0.6018x + 150.41$	$y = 0.0189x^2 - 1.3035x + 154.49$	$y = -0.0014x^3 + 0.0892x^2 - 2.254x + 157.64$
	4 Processing Element	$y = 262.09x + 847$	$y = -2.6198x^2 + 369.87x + 8.6667$	$y = -2.6198x^2 + 369.87x + 8.6667$	$y = 21.375x + 332$	$y = -0.0365x^2 + 22.875x + 320.33$	$y = -0.0365x^2 + 22.875x + 320.33$	$y = -0.503x + 147.56$	$y = 0.0127x^2 - 1.0262x + 151.63$	$y = 0.0127x^2 - 1.0262x + 151.63$
	8 Processing Element	$y = 454.31x + 2699$	$y = 454.31x + 2699$	$y = 454.31x + 2699$	$y = 42.25x + 585$	$y = 42.25x + 585$	$y = 42.25x + 585$	$y = -0.4117x + 144.36$	$y = -0.4117x + 144.36$	$y = -0.4117x + 144.36$
32 Bit s	1 Processing Element	$y = 214.83x + 369.26$	$y = 0.5761x^2 + 193.49x + 493.5$	$y = -0.0047x^3 + 0.8203x^2 + 190.19x + 504.43$	$y = 8.4391x + 176.91$	$y = -0.1032x^2 + 12.26x + 154.67$	$y = -0.0106x^3 + 0.4427x^2 + 4.875x + 179.1$	$y = -0.3904x + 111.31$	$y = 0.0138x^2 - 0.8999x + 114.28$	$y = -0.0033x^3 + 0.1832x^2 - 3.1927x + 121.86$
	2 Processing Element	$y = 291.55x + 678.57$	$y = 1.057x^2 + 252.39x + 906.5$	$y = 0.0468x^3 - 1.3516x^2 + 284.98x + 798.71$	$y = 15.026x + 360.61$	$y = -0.1358x^2 + 20.055x + 331.33$	$y = -0.0022x^3 - 0.0208x^2 + 18.5x + 336.48$	$y = -0.334x + 109.53$	$y = 0.01x^2 - 0.7052x + 111.69$	$y = -0.0009x^3 + 0.0566x^2 - 1.3352x + 113.77$
	4 Processing Element	$y = 516.66x + 1167$	$y = 2.5469x^2 + 411.87x + 1982$	$y = 2.5469x^2 + 411.87x + 1982$	$y = 33.196x + 702$	$y = -0.0469x^2 + 35.125x + 687$	$y = -0.0469x^2 + 35.125x + 687$	$y = -0.2795x + 107.96$	$y = 0.0059x^2 - 0.5231x + 109.85$	$y = 0.0059x^2 - 0.5231x + 109.85$
	8 Processing Element	$y = 1252.3x + 1578$	$y = 1252.3x + 1578$	$y = 1252.3x + 1578$	$y = 98.063x + 1054$	$y = 98.063x + 1054$	$y = 98.063x + 1054$	$y = -0.1647x + 104.08$	$y = -0.1647x + 104.08$	$y = -0.1647x + 104.08$

Table 6: Empirical Models for n-bit {8, 16 and 32}, m-Point ∈ {4, 8, 16 and 32} for processing elements ∈ {1, 2, 4, and 8}

Now for the number of clock cycles that are required for the execution for m points input using p-processing elements irrespective of number of input bits required can be depicted by the following graph

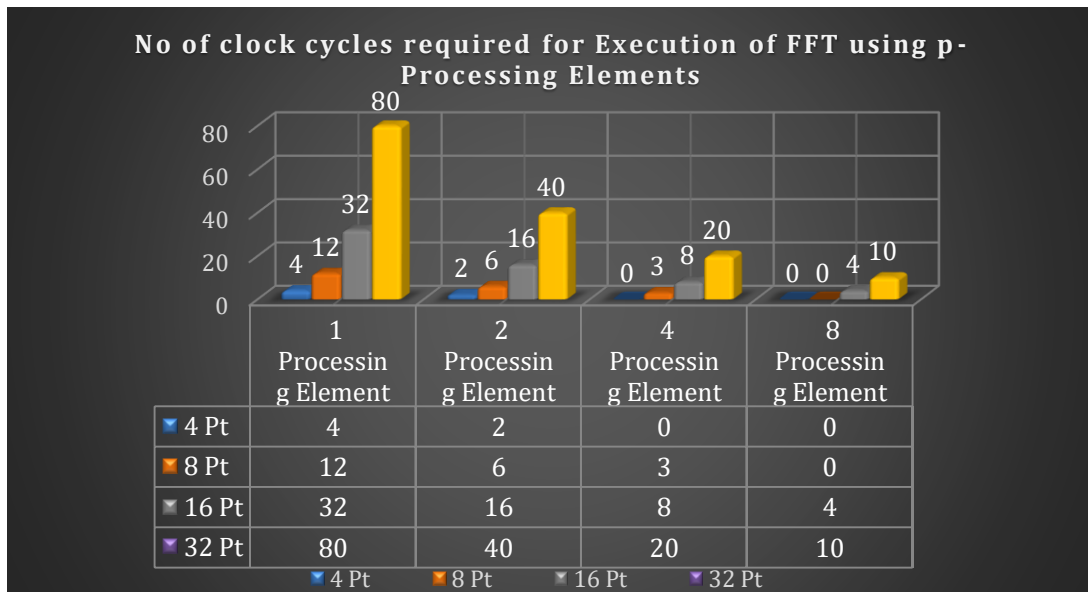


Figure 4-8: Comparison of 32 bit input for Clock Rate:

From the Figure 4-8, it is observed that the number of clock cycles required for the execution of n-points FFT using p- processing elements are decreased as the number of processing elements are increased. Furthermore it should also be noted that the number of PE's must not increase by $\frac{m}{2}$ and the minimum of stages / cycles in which m-point FFT can be executed is $\log_2 m$, as after that wastage of hardware resources will start.

4.2 Execution, Working and Implementation of Empirical Model

To show the practical implementation of the generated empirical models, of first order for slice LUT's and LUT FF Pairs; and second order for clock rate, let us suppose a scenario for generation of FFT hardware having different required specs from the user. In this scenario the user has following hardware requirements stated in Table 7.

	Requirements	
Desired	Input Bits	8
	Input Points	32
	Slice LUT	2500
	LUT FF Pairs	350
	Clock Rate	250

Table 7: Requirements for Generation of Hardware Parameters:

The hardware requirements for all the stated parameters is generated one by one in accordance with the specifications for making the hardware to achieve a tradeoff for all the parameters.

Initially the number of input bits are taken into consideration that depicts the family of empirical model i.e. either it would be 8, 16, or 32 bit input. After finalizing the family, the number of input points are put in the model and their error with required specs are calculated. The accumulative error that is taken into consideration is the squared error, because the error acquired after subtraction can be both positive and negative, depending upon one's requirement. That generated model will be considered which has least accumulative error ratio, or in other words for an optimal hardware implementation, that model is considered to which the squared error is least or bearable compared to others. The implementation of empirical model for Table 7 is shown as follows:

	8 Bit 1 PE			8 Bit 2 PE		
	Slice LUT	LUT FF Pairs	Clock Rate	Slice LUT	LUT FF Pairs	Clock Rate
Answer	2100.042	208.28	231.0312	2635.642	302.28	271.8336
Error	-0.1904523	-0.6804301	-0.08210492	0.0514645	-0.15786687	0.0803197
Squared Error	0.03627211	0.462985244	0.006741219	0.00264859	0.02492195	0.0064513
Total Squared Error	0.505998572			0.034021804		
	8 Bit 4 PE			8 Bit 8 PE		
	Slice LUT	LUT FF Pairs	Clock Rate	Slice LUT	LUT FF Pairs	Clock Rate
Answer	4760.18	630.636	200.4244	9124	1217	134.4268
Error	0.47480978	0.445004725	-0.24735312	0.72599737	0.71240756	-0.859748
Squared Error	0.22544432	0.198029206	0.06118356	0.52707218	0.50752453	0.739167
Total Squared Error	0.484657093			1.773763688		

Table 8: Arithmetic Calculations based on Empirical Model for Hardware Parameters Stated in Table 7

The squared errors for all PE $\in \{1, 2, 4, 8\}$ are as follows

No. of PE.	Squared Error
1	0.505998572
2	0.034021804
4	0.484657093
8	1.773763688

Table 9: Squared Error for different Processing Elements as per Requirement in Table 7

As stated earlier, that hardware is selected in which it has the least squared error compared to all the remaining ones, or the one with the bearable error. In this case the hardware specs of the third case having 4 PEs is selected. The reason to choose 4 PE is that it

can perform the FFT execution comparatively faster as compared to 1 and 2 PEs and the error of 0.48 is also bearable to an extent.

After finding the number of processing elements, i.e. 4 in the given case, the next step is to find the number of clock cycles required by the hardware for execution. Considering equation (5) for finding the requires number of cycles. The input and the results that are obtained after using the stated equation is shown in the table below

x	4
Processing Elements	4
Number of required cycles	20

Table 10: Required Cycles for Generated Hardware for 4 Processing Elements

Now in the same way if we increase the number of bits of input in Table: 7 from 8 bit input to 16 bit, keeping all the other required specifications same, the hardware parameters acquired from the empirical model will be as follows:

	16 Bit 1 PE			16 Bit 2 PE		
	Slice LUT	LUT FF Pairs	Clock Rate	Slice LUT	LUT FF Pairs	Clock Rate
Answer	4186.61	253.1726	241.5332	5220.6	490.842	215.5556
Error	0.40285816	-0.38245607	-0.03505439	0.52112784	0.28693958	-0.1597936
Squared Error	0.1622947	0.146272653	0.001228811	0.27157423	0.08233432	0.025534
Total Squared Error	0.30979616			0.379442538		
	16 Bit 4 PE			16 Bit 8 PE		
	Slice LUT	LUT FF Pairs	Clock Rate	Slice LUT	LUT FF Pairs	Clock Rate
Answer	9233.88	1016	197.4732	17236.92	1937	157.5344
Error	0.72925791	0.655511811	-0.26599458	0.85496249	0.81930820	-0.586955
Squared Error	0.53181709	0.429695734	0.07075311	0.73096086	0.67126594	0.3445161
Total Squared Error	1.032265943			1.74674294		

Table 11: Arithmetic Calculations based on Empirical Model for Hardware Parameters Stated in Table 7 for 16 Bit input

In such a case as stated in Table: 11 we can consider both two and four processing elements, this all depends upon the requirement of the application that either it needs an architecture that consumes less resources of it wants a faster FFT execution. In this case if the application requires a processor with least resources then 1 processing element can be the best possible architectural model. On the other hand if the application requires an FFT that

can be completed in less time then 4 PE are preferable. Whereas in case of an application that needs both faster execution as well as lesser resources then the model with 2 PEs is the best model among all. If we consider 4 processing elements, the number of clock cycles would be same as shown in in Table: 10, whereas if we consider the design for 2 processing elements, then the required number of clock cycles would be as follows:

x	8
Processing Elements	2
Number of required cycles	40

Table 12: Required Cycles for Generated Hardware for 2 Processing Elements

Using a model of 2 PE may have a tradeoff in all the requirements from the user.

Now at last considering the empirical model for a 32 bit input while considering the same specifications as stated earlier in Table: 7. After implementation and execution of the empirical model for 32 bit input, the results are as follows:

	32 Bit 1 PE			32 Bit 2 PE		
	Slice LUT	LUT FF Pairs	Clock Rate	Slice LUT	LUT FF Pairs	Clock Rate
Answer	7243.82	446.9612	157.208	10008.17	841.442	144.4964
Error	0.65487823	0.216934266	-0.59024986	0.75020408	0.58404738	-0.7301469
Squared Error	0.42886549	0.047060476	0.348394897	0.56280617	0.34111134	0.5331145
Total Squared Error	0.824320866			1.437032025		
	32 Bit 4 PE			32 Bit 8 PE		
	Slice LUT	LUT FF Pairs	Clock Rate	Slice LUT	LUT FF Pairs	Clock Rate
Answer	17700.12	1764.272	132.6308	41651.6	4192.016	109.3504
Error	0.85875802	0.801617891	-0.88493171	0.9399783	0.91650795	-1.286228
Squared Error	0.73746534	0.642591243	0.78310412	0.8835592	0.83998682	1.6543837
Total Squared Error	2.163160703			3.377929755		

Table 13: Arithmetic Calculations based on Empirical Model for Hardware Parameters Stated in Table 7 for 32 Bit input

In the above table, the difference in between the squared errors is much higher as compared to the previous calculations done for 8 and 16 bit input. So for an input of 32 bit with the given required specifications, it would be more appropriate to consider the hardware architecture with one processing element. Considering equation (5), the total number of clock cycles required for an architecture having one PE will be as follows:

x	16
Processing Element	1
Number of required cycles	80

Table 14: Required Cycles for Generated Hardware for 1 Processing Element

So for one processing element a total of 80 clock cycles are required to execute the FFT algorithm. Here it is also concluded that as the processing elements are increased, the required number of clock cycles are lessened accordingly

The summarized squared inputs for 8, 16 and 32 bit inputs for the requirements stated in Table 7 is as follows.

No. of PE.	Squared Error for 8 Bit Input	Squared Error for 16 Bit Input	Squared Error for 32 Bit Input
1	0.505998572	0.30979616	0.824320866
2	0.034021804	0.379442538	1.09592884
4	0.423489529	1.032265943	2.163160703
8	1.773763688	1.075485162	3.377929755

Table 15: Summarized Squared Errors for 8, 16 and 32 Bits input

Considering how much error is bearable and with how many bits of input we can achieve our expected results, we can have our best possible architectural model. For instance, let us suppose that a squared error of around 1.05 is bearable, then we can use 4 or 8 processing elements using 16 bit input whereas in case of 32 bit input we can use either 1 or 2 processing elements that may vary from application to application. So with the summarized squared error table we can also find out with how much of our input bits signal, which model will be best suitable among all the models, considering how much squared error is can be tolerable.

So, in this section of the report we have summarized, how our empirical model will work, by taking a generic specifications with different input bit size. The results of the implementation of the generated model by increasing the input bit size and its effect on the squared error are also shown in this section.

It should be noted here that all the calculations shown above as the implementation and execution of the empirical model includes all the stated parameters of the hardware that we are considering i.e. number of slice LUT's, LUT FF pairs and clock rate. On the other hand, besides taking into account all the specifications, if any one specification is taken into consideration from the user, then that hardware is considered which have least or bearable squared error specific to the particular hardware specification.

Chapter 5 : CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this report we have presented a Fast Fourier Transform architecture for m-Points $\in \{4,8,16 \text{ and } 32\}$ using p-Processing Elements $\in \{1, 2, 4, \text{ and } 8\}$. The basic building block for FFT algorithm is the butterfly architecture that is modeled as the processing element in the proposed model. These processing element are required to perform the desired m-points Fast Fourier Transform. After the formation of architectural design of all the possible hardware architectures, their analysis w.r.t. several hardware parameters i.e. number of Slice LUTs, LUT FF pairs, number of clock cycles required to perform m-point FFT and clock rate is done. An empirical model based upon n-Bits, m-Points and p-Processing Elements for FFT using a renowned machine learning algorithm i.e. Regression Technique, is proposed after the analysis of the architecture on the stated parameters. Linear regression, quadratic regression and cubic regression is implemented on the acquired data to generate the most generalized empirical model.

The desired parameters that are required to perform m-Points FFT by the user are considered as bench mark for the future architectural design and are passed through the particular empirical model. If we consider all the stated parameters then, the squared difference in value of specifications from the user and the actual architectural specs is taken. After the summation of the squared difference of all the parameters, either that model is considered for the implementation of design which has least squared error or that model is taken into consideration that has bearable squared error to increase the efficiency of the future hardware. In this way this model will provide the closest related real optimum specifications of actual parameterized hardware characteristics including the required number of processing elements, which needs to be incorporated to design the hardware. However if we consider a single parameter e.g. only number of Slice LUTs or some other, then the square error of only that parameter is considered. Rest of the procedure for choosing the optimal architecture is same in both the conditions.

An observation that is made in this context is that the number of clock cycles required by the architecture to perform m-points FFT is inversely proportional to the number of processing elements in that hardware design. Moreover, the number of bits of input directly

affect other parameters of the hardware architecture e.g. number of Slice LUTs, LUT-FF Pairs etc.

This thesis report can help the researchers for prediction of their FFT hardware specifications based upon their need/requirement including their parameter of interest e.g. area efficient, time efficient architecture etc. hence we are able to estimate the hardware feasibility before designing the actual hardware.

In addition to this, the technique proposed in the report to analyze the hardware specification, and predicting the futuristic parameters after making their empirical model and then finding its feasibility as per one's requirement, doesn't only refers to the hardware stated. Infact this technique can be applied to any system and thus allowing the researchers to predict their desired system before its actual implementation.

5.2 Contribution

Following are the primary contributions of this research work:

- We propose a reconfigurable architecture for n-bits, m-points Fast Fourier Transform (FFT) algorithm using p-processing elements.
- Empirical model for n-bits and m-points input for FFT is introduced, that can calculate the feasibility of the futuristic architecture.
- Pre-calculating the specifications and parameters of an optimized FFT architecture that is based upon one's need/requirement.
- A relation for number of clock cycles required to perform m-Point using n-Radix FFT is introduced.
- A procedure for finding the unique Permutation Matrix out of n! Permutation Matrices is introduced (Annex).

5.3 Future Work

Following tasks can be performed as future work in contribution to this report:

- Empirical model for floating point input can be designed.
- Devise an empirical model for fully parallel pipelined FFT architecture.
- Finding the effects of the same architecture on different hardware platforms e.g. GPUs, CPUs, FPGA etc.
- Applying several techniques to increase the efficiency of any parameter e.g. using multiplier less pipelined processor [15], ROM less FFT processor [16] etc.

- Empirical model for Radix-x FFT can be modelled.
- Effect of using different Radix FFT on empirical model and its effect on empirical model can be can be studied.
- Empirical model for different variants of FFT [11] can be modelled, to predict that which variant of FFT is the most feasible, as per one's need.

Chapter 6 : REFERENCES

- [1] Gnanishivaram, K. and Neeraja, S., 2014. FFT/IFFT Processor Design for 5G MIMO OFDM Systems. *International Journal*, 3(3).
- [2] <https://cadcammodelling.wordpress.com/2011/04/14/fourier-transform-and-its-applications/> Accessed on July 23rd,2019 Time: 1130 hrs
- [3] <https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#1271a74860ba> Accessed on July 19th,2019 Time: 1400 hrs
- [4] <https://www.sciencedaily.com/releases/2013/05/130522085217.htm> Accessed on June 29th,2019 Time: 1700 hrs
- [5] <https://www.sgl-rotec.com/history-evolution-digital-media/> Accessed on July 02nd,2019 Time: 2000 hrs
- [6] Cooley, J.W. and Tukey, J.W., 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90), pp.297-301.
- [7] Lenssen, N. and Needell, D., 2014. An introduction to fourier analysis with applications to music. *Journal of Humanistic Mathematics*, 4(1), pp.72-91.
- [8] Tjahyanto, A., Suprpto, Y.K., Purnomo, M.H. and Wulandari, D.P., 2012, May. Fft-based features selection for javanese music note and instrument identification using support vector machines. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)* (Vol. 1, pp. 439-443). IEEE.
- [9] Shen, F., Song, Z., Wu, C., Geng, J. and Wang, Q., 2015. Research on the fast Fourier transform of image based on GPU. *arXiv preprint arXiv:1505.08019*.
- [10] Aboleaze, M.A. and Elnaggar, A., 2006, June. Reducing memory references for FFT calculation. In *Proc. of the International Conference on Computer Design* (pp. 26-28).
- [11] Milder, P.A., Franchetti, F., Hoe, J.C. and Püschel, M., 2010, March. Hardware implementation of the discrete fourier transform with non-power-of-two problem size. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 1546-1549). IEEE.
- [12] Raju, K.S., Sengar, V., Gangal, M., Tanwar, P. and Prasad, P.B., Hardware Implementation of Discrete Fourier Transform and its Inverse Using Floating Point Numbers.
- [13] Ganesan, K., Govardhanarajan, T.S., Dhurkadas, A. and Veerabhadraiah, S., 1982. Hardware realization of a general purpose FFT processor in a distributed processing configuration. *Defence Science Journal*, 32(1), pp.41-46.
- [14] Wang, G., Yin, B., Cho, I., Cavallaro, J.R., Bhattacharyya, S. and Takala, J., 2014, May. Efficient architecture mapping of FFT/IFFT for cognitive radio networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3933-3937). IEEE.
- [15] Kannan, M. and Srivatsa, S., 2009. Hardware Implementation Low Power High Speed FFT Core. *International Arab Journal of Information Technology (IAJIT)*, 6(1).
- [16] Indirapriyadarsini, K., Kamalakumari, S. and Prasannakumar, G., VLSI Implementation of Pipelined Fast Fourier Transform. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(4).
- [17] Teymourzadeh, R., 2017. High Resolution Single-Chip Radix II FFT Processor for High-Tech Application. *Fourier Transforms: High-tech Application and Current Trends*, p.67.
- [18] Fande, A. and Sahu, A., Efficient Implementation & Comparison of Signed Complex Multiplier on FPGA using FFT Algorithm. *International Journal of Scientific Research Engineering & Technology (JSRET)*, 3(2), pp.188-191.
- [19] Algnabi, Y.S., Aldaamee, F.A., Teymourzadeh, R., Othman, M. and Islam, M.S., 2012, September. Novel architecture of pipeline Radix 2 2 SDF FFT Based on digit-slicing technique. In *2012 10th IEEE International Conference on Semiconductor Electronics (ICSE)* (pp. 470-474). IEEE.
- [20] Bo, Y., Dou, R., Han, J. and Zeng, X., 2013. A hardware-efficient variable-length FFT processor for low-power applications. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference* (pp. 1-4). IEEE.

- [21] Mukherjee, A., Sinha, A. and Choudhury, D., 2014. A novel architecture of area efficient FFT algorithm for FPGA implementation. *ACM SIGARCH Computer Architecture News*, 42(5), pp.1-6.
- [22] Ouerhani, Y., Jridi, M. and Alfalou, A., 2012. AREA-DELAY EFFICIENT FFT ARCHITECTURE USING PARALLEL PROCESSING AND NEW MEMORY SHARING TECHNIQUE. *Journal of Circuits, Systems, and Computers*, 21(06), p.1240018.
- [23] Puchała, D., Stokfiszewski, K., Yatsymirskyy, M. and Szczepaniak, B., 2015, September. Effectiveness of Fast Fourier Transform implementations on GPU and CPU. In *2015 16th International Conference on Computational Problems of Electrical Engineering (CPEE)* (pp. 162-164). IEEE.
- [24] Jhariya, P.K. and Dodkey, N., 2016. Implementation of Fast Fourier Transform using Resource Reuse Technique on FPGA. *Int. Journal of Scientific Research in Science, Engineering and Technology*, 2(1), pp.2395-1990.
- [25] Kumar, A., Gautam, G. and Ram, V.K., 2016. FPGA Implementation of 2x2 Crossbar Switch. *International Journal of Engineering Science*, 7435.
- [26] Khan, M.A. and Ansari, A.Q., 2011. Design of 8-bit programmable crossbar switch for network-on-chip router. In *Trends in Network and Communications* (pp. 526-535). Springer, Berlin, Heidelberg.
- [27] Ganee, S.A., Ganee, S.A. and Dar, J.R., FPGA Design of 8 bit 4x 4 Crossbar Switch for Multi Processor System on Chip Using Round Robin Arbitration Algorithm.
- [28] Freitas, H.C., Carvalho, M.B., Amaral, A.M., Diniz, A.R., Martins, C.A. and Ramos, L.E., 2006, May. Reconfigurable crossbar switch architecture for network processors. In *2006 IEEE International Symposium on Circuits and Systems* (pp. 4-pp). IEEE.
- [29] <https://towardsdatascience.com/polynomial-regression-bbe8b9d97491> as on September 10, 2019 at 1530 Hrs.

Annexure : FINDING THE UNIQUE PERMUTATION MATRIX FOR REVERSE ORDER KRONECKER PRODUCT INTUITIVELY FOR FAST FOURIER TRANSFORM

1 Introduction

All the experimentations, observations and results related to the subject are incorporated in the stated chapter being it to be a minor research project that leads to our actual research.

This section of the report presents a unique method for finding the unique Permutation Matrix out of $n!$ matrices, to perform Fast Fourier Transform using Kronecker Product. This uniquely identified permutation matrix is used to attain the reverse order Kronecker product without using the same technique, as used for obtaining the original Kronecker product.

Kronecker product, plays an imperative role in major disciplines of science as in mathematics, linear algebra, big data analysis and signal processing etc. that acts as a nucleus in the formation of this modern era especially from application perspective [1]. This product also finds its applications in matrix calculus [2], system theory [3], differential equations [4] that are the basis for circuit analysis and much more. A well-established transform to study a time domain signal from frequency perspective, filtering and analysis is Fourier transform [5]. Kronecker product is an arithmetic tool that finds its significance in many applications in field of research.

The main problem that arises during the implementation of the Kronecker product is its computation cost, especially to those algorithms in which it is used repeatedly. This problem becomes a bottleneck while computing two or more products simultaneously, especially when, in the same arithmetic calculation original and reverse order Kronecker product are to be calculated, e.g. in formulation of DFT as stated in equation 6, that is taken as a building block for the formation of this technique.

$$F'_{n^2} = (I_n \otimes F_n)D_{n^2}(F_n \otimes I_n) \quad (1)$$

Keeping in view the calculation of Kronecker product, an intuitive approach is proposed to bypass the rigorous calculations required to compute the permutation matrix. Here mechanism that calculates a unique permutation matrix is presented, so that

computation time for finding the reverse order product decreases, and reverse order Kronecker product from its original may be generated with minimum computations.

In such cases where we have to compute the original and reverse order KP at same time, being it to be computationally expensive algorithm, we can use Permutation Matrix for this purpose to reuse the previously calculated product for further use as below:

$$F'_{n^2} = (I_n \otimes F_n)D_{n^2}P_{n^2}(F_n \otimes I_n)P_{n^2} \quad (2)$$

And hence by using the permutation matrix, we can find the DFT with less computation using the previously calculated Kronecker Product.

2 Permutation Matrix

Permutation matrix is a binary matrix having two entries (i.e. 0 & 1), and is obtained by permuting the columns /rows of an $n \times n$ identity matrix. Note that every permutation in the identity matrix provides us with a unique permutation matrix that leads us towards different solutions after its multiplication with original matrix. For instance, let us suppose a 4x4 matrix, multiplied with different permutation matrices of same order as below:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{13} & a_{14} & a_{12} \\ a_{21} & a_{23} & a_{24} & a_{22} \\ a_{31} & a_{33} & a_{34} & a_{32} \\ a_{41} & a_{43} & a_{44} & a_{42} \end{bmatrix} \quad (3)$$

Using a different permutation matrix of same order as:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} a_{13} & a_{14} & a_{12} & a_{11} \\ a_{23} & a_{24} & a_{22} & a_{21} \\ a_{33} & a_{34} & a_{32} & a_{31} \\ a_{43} & a_{44} & a_{42} & a_{41} \end{bmatrix} \quad (4)$$

Hence from (3) & (4), using different permutation matrices multiplied with same matrix, different results are obtained.

2.1 Rows and Column Permuted Matrix

From (3) & (4), it is observed that in matrix multiplication, using PM after the original matrix swaps the columns of the original matrix. However, to achieve a row permuted matrix, the permutation matrix is placed behind the original matrix as below:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} a_{41} & a_{42} & a_{43} & a_{44} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \quad (5)$$

So from (4) and (5) it can be observed that for same $n \times n$ permutation matrix, the placement of the PM will decide the resultant matrix either to be a row permuted matrix or a column permuted matrix.

3 Kronecker Product and Permutation Matrix

Let us suppose we have A and B matrices of $m \times n$ and $p \times q$ order respectively, then the Kronecker product of A & B would be as

$$A \otimes B = \begin{bmatrix} a_{11} * \mathbf{B} & \dots & a_{1n} * \mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1} * \mathbf{B} & \dots & a_{mn} * \mathbf{B} \end{bmatrix} \quad (6)$$

that can be expressed more explicitly as

$$\begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \dots & a_{11}b_{1q} & \dots & a_{1n}b_{11} & \dots & \dots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \dots & a_{11}b_{2q} & \dots & a_{1n}b_{21} & \dots & \dots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & \dots & \vdots & \dots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \dots & a_{11}b_{pq} & \dots & a_{1n}b_{p1} & \dots & \dots & a_{1n}b_{pq} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \dots & a_{m1}b_{1q} & \dots & a_{mn}b_{11} & \dots & \dots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \dots & a_{m1}b_{2q} & \dots & a_{mn}b_{21} & \dots & \dots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \dots & a_{m1}b_{pq} & \dots & a_{mn}b_{p1} & \dots & \dots & a_{mn}b_{pq} \end{bmatrix}$$

And the reverse product i.e. $B \otimes A$ can be written as

$$B \otimes A = \begin{bmatrix} b_{11} * \mathbf{A} & \dots & b_{1p} * \mathbf{A} \\ \vdots & \ddots & \vdots \\ b_{p1} * \mathbf{A} & \dots & b_{pq} * \mathbf{A} \end{bmatrix} \quad (7)$$

From (6) and (7) it can be observed that $A \otimes B$ and $B \otimes A$ are entirely different matrices. But from $A \otimes B$, $B \otimes A$ can be achieved just by swapping some of the rows and

columns with the specific permutations. In other words we can say that $A \otimes B$ and $B \otimes A$ are permutation equivalent matrices, and there exists a unique permutation matrix such that

$$B \otimes A = P(A \otimes B)P^T \quad (8)$$

Furthermore, another characteristic of this unique permutation matrix is that the transpose and inverse of the unique PM is same as the PM itself, as

$$P = P^{-1} = P^T \quad (9)$$

All remaining $n \times n$ PM's, doesn't qualify the above property due to which they are not classified as the unique PM for Kronecker Product

4 Proposed Methodology

Let us suppose two, 2×2 matrices A and B for understanding. The two Kronecker products $A \otimes B$, as in (6), and $B \otimes A$, as in (7) will be as follows,

$$A \otimes B = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix} \quad (10)$$

Similarly

$$B \otimes A = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\ a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\ a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\ a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22} \end{bmatrix} \quad (11)$$

Total number of permutation matrices that can be produced from an n^{th} order identity matrix are calculated as

$$\text{no. of PMs for } I_{n \times n} = n! \quad (12)$$

As A & B are both 2×2 order matrices and the order of their Kronecker Product will be 4×4 , thus in this case a 4×4 identity matrix will be used as PM that generates $4!$ i.e. 24 different permutation matrices as stated in (12) which are shown in Figure Annex-1.

Out of these 24 matrices as shown in Figure Annex 1, a single unique permutation matrix say P_4 (eq.13) satisfies the requirements for reverse order Kronecker product as stated in (9). From the above two matrices (10) and (11) it can be observed that from $A \otimes B$ we can easily produce $B \otimes A$ just by swapping second and third rows and then

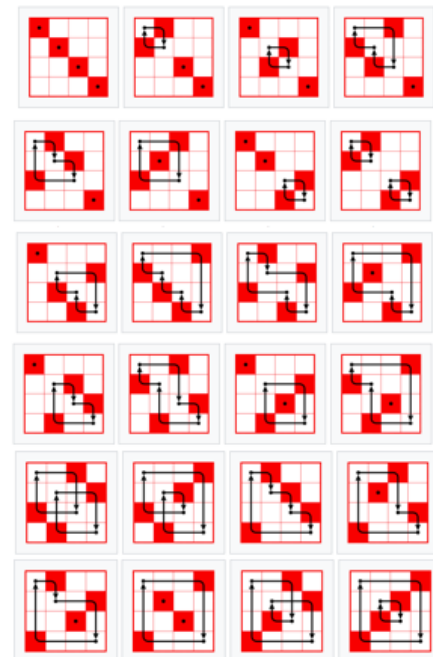


Figure Annex-0-1: Generation of 4! Permutations of 4x4 Identity

swapping same columns. To achieve this, an identity matrix will be used, that is also swapped in the same manner, and is then multiplied as in (4) & (5).

$$P_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

That is same number of rows are swapped from the identity matrix to make unique permutation matrix, that satisfies the unique PM property and hence by substituting, it may become

$$PM * B \otimes A * PM$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} \\ a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} \\ a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} \\ a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= A \otimes B$$

These operations will eventually give us the results, that corresponds to our reverse order Kronecker Product i.e. $A \otimes B$

4.1 Formation of $n \times n$ Permutation Matrix

In order to find the uniquely identified permutation matrix that satisfies requirements for reverse order Kronecker calculation stated in (6) and (7), we take two matrices A and B of order 2×2 having Kronecker product of order 4×4 as discussed earlier

$$A \otimes B \Rightarrow \begin{array}{cccc} (1^{st}) & (2^{nd}) & (3^{rd}) & (4^{th}) \\ a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} & \text{(I)} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} & \text{(II)} \\ a_{21}b_{11} & a_{21}b_{21} & a_{22}b_{11} & a_{22}b_{12} & \text{(III)} \\ a_{21}b_{12} & a_{22}b_{22} & a_{22}b_{21} & a_{22}b_{22} & \text{(IV)} \end{array}$$

Whereas it's reverse order Kronecker product would be as

$$B \otimes A \Rightarrow \begin{array}{cccc} (1^{st}) & (2^{nd}) & (3^{rd}) & (4^{th}) \\ a_{11}b_{11} & a_{12}b_{11} & a_{11}b_{12} & a_{12}b_{12} & \text{(I)} \\ a_{21}b_{11} & a_{22}b_{11} & a_{21}b_{12} & a_{22}b_{12} & \text{(II)} \\ a_{11}b_{21} & a_{12}b_{21} & a_{11}b_{22} & a_{12}b_{22} & \text{(III)} \\ a_{21}b_{21} & a_{22}b_{21} & a_{21}b_{22} & a_{22}b_{22} & \text{(IV)} \end{array}$$

The swapping hierarchy of entries from $A \otimes B$ towards $B \otimes A$ is shown below. Furthermore, these are the swapped rows similar to the swapping of rows/columns of an identity matrix to form a uniquely identified PM,

$$\begin{array}{ccc} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} & \longleftrightarrow & \begin{bmatrix} 1 & 3 & 2 & 4 \\ 9 & 11 & 10 & 12 \\ 5 & 7 & 6 & 8 \\ 13 & 15 & 14 & 16 \end{bmatrix} \\ A \otimes B & \Leftrightarrow & B \otimes A \end{array}$$

It can be observed in the above case that reverse order KP i.e. $B \otimes A$ can be attained from $A \otimes B$ by swapping the 2nd and 3rd rows and then same columns or vice versa. Taking $A \otimes B$ and reshaping to make it $B \otimes A$ using the assigned labels would lead towards the formation of permutation matrix as below

$$\begin{array}{l} A \otimes B \rightarrow 1 \ 2 \ 3 \ 4 \\ B \otimes A \rightarrow 1 \ 3 \ 2 \ 4 \end{array}$$

It is observed that the 1st and 4th rows of both are at the same place where as other 2nd and 3rd rows/columns are swapped with each other. Considering the stated hierarchy, from $A \otimes B$ the permutations of rows of $B \otimes A$ can be achieved as follows:

$$A \otimes B \rightarrow 1 \ 2 \ 3 \ 4 \rightarrow \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \begin{array}{c} 1 \\ 3 \\ 2 \\ 4 \end{array} \rightarrow [1 \ 3 \ 2 \ 4] \rightarrow B \otimes A$$

In order to transform $B \otimes A$ from $A \otimes B$, the matrix entries are first row permuted then column permuted. As stated earlier, the swapping hierarchy of the Kronecker product, to achieve its reverse order, is same as that of identity matrix. For the given example, the uniquely identified PM can be achieved using the same transitions of rows/columns as stated above. Here entries depict the placement of '1' in each row/column of identity matrix. This swapping technique would lead us towards the unique permutation matrix that is used as uniquely identified Kronecker permutation matrix

4.2 Pseudo code

Let us suppose two matrices A and B of $n \times n$ dimensions. The generation of $B \otimes A$ from $A \otimes B$ is to formulated by using an n^2 permutation matrix. This permutation matrix can be formulated as follows:

1. Generate a natural number $n \times n$ dimensional matrix having n^2 entities.
2. Reshape the matrix column wise in such a way that each $(n+1)$ st column lies beneath the n th column.

3. Make this column vector into a row vector.
4. The entities of this row vector point out towards the position of '1' at a specific row/column. Write '1' at every place pointing by the row vector and all other entries besides these locations are '0's.
5. This generated matrix is the required uniquely identified permutation matrix.

The implementation of the pseudo code for 4x4 PM is as follows

$$\xrightarrow{1^{st}} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \xrightarrow{2^{nd}} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix} \xrightarrow{3^{rd}} [1 \ 3 \ 2 \ 4] \xrightarrow{4^{th}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{5^{th}} (Unique PM)$$

Similarly for 16x16 PM, it can be obtained by the swapping of natural number matrix rows/columns and following rows will come one after the other in 16x16 identity matrix as

$$[1^{st}, 5^{th}, 9^{th}, 13^{th}, 2^{nd}, 6^{th}, 10^{th}, 14^{th}, 3^{rd}, 7^{th}, 11^{th}, 15^{th}, 8^{th}, 12^{th}, 16^{th}]$$

5 Observation and Results

In order to compare our proposed methodology, we generated random matrices of order 4x4 till 20x20 for Kronecker Product, requiring permutation matrix of order 16 x16 up to 400x400 to calculate its reverse order. The Kronecker product for these matrices were calculated and afterwards derived from their reverse order product, by generating permutation matrix. It is observed the original KP and the one calculated using its reverse order using PM is always same and their difference gives the null matrix that verifies the applicability of this technique. For instance let us suppose two matrices of order 3x3.

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \& B = \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix}$$

$$A \otimes B = \begin{bmatrix} aj & ak & al & bj & bk & bl & cj & ck & cl \\ am & an & ao & bm & bn & bo & cm & cn & co \\ ap & aq & ar & bp & bq & br & cp & cq & cr \\ dj & dk & dl & ej & ek & el & fj & fk & fl \\ dm & dn & do & em & en & eo & fm & fn & fo \\ dp & dq & dr & ep & eq & er & fp & fq & fr \\ gj & gk & gl & hj & hk & hl & ij & ik & il \\ gm & gn & go & hm & hn & ho & im & in & io \\ gp & gq & gr & hp & hq & hr & ip & iq & ir \end{bmatrix} \& B \otimes A = \begin{bmatrix} aj & bj & cj & ak & bk & ck & al & bl & cl \\ dj & ej & fj & dk & ek & fk & dl & el & fl \\ gj & hj & ij & gk & hk & ik & gl & hl & il \\ am & bm & cm & an & bn & cn & ao & bo & co \\ dm & em & fm & dn & en & fn & do & eo & fo \\ gm & hm & im & gn & hn & in & go & ho & io \\ ap & bp & cp & aq & bq & cq & ar & br & cr \\ dp & ep & fp & dq & eq & fq & dr & er & fr \\ gp & hp & ip & gq & hq & iq & gr & hr & ir \end{bmatrix}$$

The Permutation Matrix of order 9x9 using the stated methodology will be:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 4 \\ 7 \\ 2 \\ 5 \\ 8 \\ 3 \\ 6 \\ 9 \end{bmatrix} \rightarrow [1 \ 4 \ 7 \ 2 \ 5 \ 8 \ 3 \ 6 \ 9] \rightarrow I_9 = PM_9$$

$$PM_9 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

As stated in (8) & (9),

$$B \otimes A = P(A \otimes B)P^T \text{ where, } P = P^{-1} = P^T$$

Substituting (9) in (8), it would become

$$B \otimes A = P(A \otimes B)P$$

So using the above equation,

$$P[(A \otimes B)P^T] = \begin{bmatrix} aj & ak & al & bj & bk & bl & cj & ck & cl \\ am & an & ao & bm & bn & bo & cm & cn & co \\ ap & aq & ar & bp & bq & br & cp & cq & cr \\ dj & dk & dl & ej & ek & el & fj & fk & fl \\ dm & dn & do & em & en & eo & fm & fn & fo \\ dp & dq & dr & ep & eq & er & fp & fq & fr \\ gj & gk & gl & hj & hk & hl & ij & ik & il \\ gm & gn & go & hm & hn & ho & im & in & io \\ gp & gq & gr & hp & hq & hr & ip & iq & ir \end{bmatrix} PM_9$$

$$\begin{aligned}
&= PM_9 \begin{bmatrix} aj & bj & cj & ak & bk & ck & al & bl & cl \\ am & bm & cm & an & bn & cn & ao & bo & co \\ ap & bp & cp & aq & bq & cq & ar & br & cr \\ dj & ej & fj & dk & ek & fk & dl & el & fl \\ dm & em & fm & dn & en & fn & do & eo & fo \\ dp & ep & fp & dq & eq & fq & dr & er & fr \\ gj & hj & ij & gk & hk & ik & gl & hl & il \\ gm & hm & im & gn & hn & in & go & ho & io \\ gp & hp & ip & gq & hq & iq & gr & hr & ir \end{bmatrix} \\
&= \begin{bmatrix} aj & bj & cj & ak & bk & ck & al & bl & cl \\ dj & ej & fj & dk & ek & fk & dl & el & fl \\ gj & hj & ij & gk & hk & ik & gl & hl & il \\ am & bm & cm & an & bn & cn & ao & bo & co \\ dm & em & fm & dn & en & fn & do & eo & fo \\ gm & hm & im & gn & hn & in & go & ho & io \\ ap & bp & cp & aq & bq & cq & ar & br & cr \\ dp & ep & fp & dq & eq & fq & dr & er & fr \\ gp & hp & ip & gq & hq & iq & gr & hr & ir \end{bmatrix} = B \otimes A
\end{aligned}$$

The results obtained are by calculating the reverse product directly, and by evaluating it through permutation matrix technique gives the same result, as the subtraction of the two, gives a null matrix, that intern proves the reliability of this methodology.

6 Conclusion

This area of report gives a brief overview of Kronecker product and its implementation in finding the reverse order multiplication, as in formulation DFT in signal processing using a unique permutation matrix technique by finding it intuitively. This would lessen its computation cost by not calculating the reverse product being it to be computationally expensive algorithm and intern evaluating it by using the ground principles of matrix theory through a unique permutation matrix, from the same previously found Kronecker product. The computation cost to calculate the PM for reverse order KP will be minimum as the entities for an $n \times n$ unique PM can be depicted directly, without finding it from $n!$ PM's.