

FPGA Implementation of 8K Points FFT/IFFT for a 5G Channel Emulator

By

Habib-ur-Rehman 01-133152-039

Hamna Tariq 01-133152-043

Supervised by

Dr. Atif Raza Jafri



2015-2019

A Report is submitted to the Department of Electrical Engineering,
Bahria University, Islamabad.

In partial fulfillment of requirement for the degree of BS(EE).

Dedication

This work of ours is dedicated to our families and our supervisor. With their continual support and backing we accomplished the task with maximum possible competence.

Acknowledgements

We are deeply grateful to the Almighty ALLAH for His countless blessings upon us, and for His bestowed strength and abilities, and for the opportunity to commence this project. We are forever grateful to our supervisor Dr. Atif Jafri, without his aspiring guidance and polite supervision this project would not be an achievement. In addition, we are thankful to our instructor Engineer Muhammad Noman for his assistance and dedicated involvement. Thanking Bahira University Islamabad for providing us the Equipment and labs whenever needed to complete this project.

Abstract

Channel emulator is an instrument for emulating the channel environment faced by transmitted waveforms. In a test environment, channel emulators replicate the channel between a transmitter and a receiver and provide a transmitted waveform affected through channel effects to the receiver input. FFT and IFFT blocks are major components of a channel emulator.

Fast Fourier Transform (FFT) is a proficient calculation used to process Discrete Fourier Transform (DFT). When deciding among alternate implementation methods, the algorithm chosen should be considering the execution speed, utilization rate of the placed hardware and hardware complexity of the system. For real time systems, execution speed is the major concern.

Objective of this project is to design a hardware for computation of 8K (8192) points FFT/IFFT with a throughput of 34 Mega samples per second. Since Cooley Tuckey algorithm breaks the DFTs, so it can be combined with any other DFT algorithm hence implying stages of multiple techniques in a row to achieve the results. We have designed Mix-Radix Butterfly architecture which uses stages of Radix 2 and Radix 4 simultaneously to achieve high speed.

Table of Contents

Dedication	i
Acknowledgements	ii
Abstract	iii
Table of Contents	iv
List of Figures	vi
List of Tables	vii
1. Introduction	1
1.1 Background	2
1.2 Problem Description	4
1.3 Project objective and scope	4
2. Literature Review	5
2.1 Fast Fourier Transform	6
2.2 Twiddle Factor	6
2.3 Radix 2	8
2.4 Radix 4	9
2.5 DIT, DIF	11
2.6 Mix Radix	12
2.7 Previous Work	13
3. Requirement Specifications	16
3.1 Existing Systems	17
3.2 Proposed System	18
3.3 Requirement Specifications	19
4. Architecture Design	21
4.1 System Architecture	22
4.2 Design Constraint	22
4.3 Design Methodology	22
5. System Implementation	25
5.1 Single Path Delay Feedback	26
5.2 SDF Modeling	26
5.3 Twiddle Factor Address Generation	28

5.4	Tools and Technology	29
6.	Results and Evaluation	30
6.1	Verification	31
6.2	Results	31
7.	Conclusion	33
	References	35
	Appendix	40

List of Figures

Figure 1.1	Fourier Transform of a signal	3
Figure 2.1	Breaking a DFT sequence	6
Figure 2.2	Twiddle Factors of 8-point FFT	7
Figure 2.3	Twiddle Factors of 12-point FFT	7
Figure 2.4	Basic Radix 2 BF	8
Figure 2.5	8-point Radix 2 BF	9
Figure 2.6	Basic radix 4 BF	10
Figure 2.7	16-point Radix 4 BF	11
Figure 2.8	DIT and DIF butterflies	12
Figure 2.9	32-point Mix Radix architecture	13
Figure 3.1	Proposed Mix Radix design	19
Figure 4.1	Mix Radix architecture for 8192 points	22
Figure 4.2	ASM Chart for the proposed architecture	24
Figure 5.1	Radix r N point SDF FFT architecture	26
Figure 5.2	Radix 2 SDF BF	27
Figure 5.3	Radix 4 SDF BF	27
Figure 5.4	SDF modeling of Radix 4 stages	28
Figure 6.1	FFT Results from Visual Studio	31
Figure 6.2	FFT Results from Xilinx	32
Figure 6.3	FFT Results from Xilinx	32

List of Tables

Table 3.1	Complexity analysis of Radix 2 and Radix BFs	17
Table 3.2	BF Calculations for 8K points	18
Table 5.1	Twiddle Factor address generation	28