STUDY OF ACCURACY AND HARDWARE PERFORMANCE IN DISCRETE TRANSFORMS AND THEIR FAST ALGORITHMS

By

Violeta Reyes Rodríguez

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS

September, 2015

Approved by:

Manuel Jiménez, Ph.D. Chairman, Graduate Committee	Date
Domingo Rodríguez, Ph.D. Member, Graduate Committee	Date
Nayda Santiago, Ph.D. Member, Graduate Committee	Date
Omar Molina, Ph.D. Graduate Studies Representative	Date
Raúl Torres, Ph.D. Department Chairperson	Date

Abstract of Thesis Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

STUDY OF ACCURACY AND HARDWARE PERFORMANCE IN DISCRETE TRANSFORMS AND THEIR FAST ALGORITHMS

By

Violeta Reyes Rodríguez

September 2015

Chair: Dr. Manuel Jiménez
Department: Electrical and Computer Engineering Department

Nowadays, during the design of digital arithmetic units, most research efforts are centered in finding algorithms that reduce resource consumption or latency. Efforts to find algorithms that provide higher accuracy are scarce. This thesis presents a study of accuracy and hardware performance of discrete transforms and their fast algorithms. The discrete transforms studied included the Fourier (DFT), the Hartley (DHT), and the cosine (DCT) direct algorithms. The fast DFT treatments were the Cooley-Tukey and Pease. The fast DHT treatments included the Bracewell and Hou. In the case of the DCT the fast treatments evaluated were the Nikara and Translation. This work used approximation and statistical methods for the accuracy analysis. These methods quantify the normwise relative error of the discrete transform treatments and determine significant differences in their accuracy. For the hardware performance analysis, a FPGA synthesis methodology was adopted to quantify resource consumption and latency of the treatments. The results of the study showed that the discrete transforms direct treatment provide higher accuracy, and the highest resource consumption and latency. We observed in the accuracy analysis that as the resolution of the discrete transform computation incremented, the range magnitude of the treatments experimental normwise relative error incremented. But the range magnitude of the fast algorithms treatments incremented at a higher scale.

Resumen de tesis presentado a la Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
requerimientos para el grado de Maestría en Ciencias

ESTUDIO DE EXACTITUD Y RENDIMIENTO HARDWARE EN TRANSFORMADAS DISCRETAS Y SUS ALGORITMOS RÁPIDOS

Por

Violeta Reyes Rodríguez

Septiembre 2015

Consejero: Dr. Manuel Jiménez
Departamento: Ingeniería Eléctrica y Computadoras

Actualmente, durante el diseño digital de unidades aritméticas, los esfuerzos se centran en buscar algoritmos que reduzcan el consumo de recursos o latencia. Esfuerzos para encontrar algoritmos que provean mayor exactitud son descuidados. Esta tesis presenta un estudio de exactitud y desempeño hardware de transformadas discretas y sus algoritmos rápidos. Las transformadas discretas estudiadas fueron las formulaciones directas de Fourier (DFT), Hartley (DHT) y coseno (DCT). Los algoritmos rápidos de la DFT fueron Cooley-Tukey y Pease. Para la DHT los algoritmos rápidos fueron Bracewell y Hou. Los algoritmos rápidos de la DCT fueron Nikara y Traslación. Este trabajo utiliza métodos de aproximación y métricas estadísticas para el análisis de exactitud. Los mismos cuantifican el error de los tratamientos y determinan diferencias significativas en su exactitud. Para el análisis de desempeño hardware, se opto por una metodología de síntesis de FPGA. Los resultados mostraron que las formulaciones directas de las transformadas discretas proporcionan mayor exactitud, consumo de recursos y latencia. En el estudio se observo que a medida que aumenta la resolución del computo de una transformada discreta, la magnitud del rango del error relativo normalizado experimental de los tratamientos aumenta. La diferencia significativa entre los tratamientos fue que la escala en que aumenta la magnitud del rango de los tratamientos de algoritmos rápidos es mayor.

I want to dedicate this work to my mother. For teaching me to be courageous and persistent; and for believing always in me. I also want to dedicate this work to my family for giving their support and help through this process.

Acknowledgements

To Professor Manuel Jimenez, for trusting and giving me the opportunity to do research and offer me his advisorship. Thanks for the encouragement, guidance, and support during this process.

To Professors Domingo Rodriguez and Nayda Santiago for being part of my graduate committee and giving me advice and support in this work.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AE	Absolute Error
RE	Relative Error
FEA	Forward Error Analysis
FDFT	Fast Discrete Fourier Transform
FDHT	Fast Discrete Hartley Transform
FDCT	Fast Discrete Cosine Transform
FIDCT	Fast Inverse Discrete Cosine Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DHT	Discrete Hartley Transform
DTT	Discrete Trigonometric Transform
DIT	Decimation In Time
DIF	Decimation In Frequency

Chapter 1
Introduction

One of the main reasons digital systems fail is due to the finite amount of bits to represent numbers. This finite representation of numbers causes errors in arithmetic computations. This problem has been the source of fatal accidents in the past. In 1991 an American Patriot missile failed to intercept an Iraqi missile, an incident caused by rounding error [1]. In 1996 the Ariane 5 rocket exploded after forty seconds in flight. The root event was caused by an overflow error [2]. During 1985 -1987 the THERAC-25 machine used for radiation therapy, caused the deaths of four patients by radiation poisoning. The reason behind the deathly overexposures was an overflow error [3]. These accidents could be avoided if proper accuracy analysis of their arithmetic units were performed.

In arithmetic algorithms, the error caused by rounding and its propagation can be analyzed and approximated. The problem during the design process is that most of the efforts are centered in finding arithmetic algorithms formulations that improve resource consumption or latency, and efforts to find formulations that improve accuracy are scarce.

This work developed methodology to study the accuracy and hardware performance of three discrete transforms and some of their fast algorithms. The discrete transforms studied were the Fourier (DFT), the Hartley (DHT), and cosine (DCT). The methodology combines approximation and statistical methods for the accuracy analysis. To assess their hardware performance, a FPGA synthesis methodology was adopted.

The rest of this document is organized as follows. Next chapter presents a theoretical background and previous work on the subject of error analysis. Chapter 3 presents the problem statement and hypothesis of the proposed work, among the work objectives. Then, in Chapter 4 the methodology followed is presented. Chapter 5 presents the results of accuracy and hardware performance of the formulations evaluated. In Chapter 6 the recommendations and contributions from this work are provided. Finally, Chapter 7 present the conclusions of this work.

Chapter 2
Previous Work

This chapter presents a theoretical background and previous work about error caused by floating point representation of numbers in arithmetic operations. Also, a literature review of works related to accuracy analysis performed on arithmetic formulations and their fast algorithms.

2.1 Error in Floating Point Representation

Error in the floating point representation of a number x, occurs when its value falls between two consecutive numbers, $Fl(x_1)$, $Fl(x_2)$, in the selected format. Namely, when a floating point arithmetic operation or some other mapping produces a result x, such that $Fl(x_1) < x < Fl(x_2)$, a rounding scheme is needed to select which of the two consecutive numbers will represent $Fl(x)$. The decision of using either $Fl(x_1)$ or $Fl(x_2)$ as $Fl(x)$ will depend on the selected rounding scheme. The absolute error resulting from the selection of the value to represent x is a metric that describes the magnitude of the difference between the approximation, $Fl(x)$ and the exact value, x.

$$|\epsilon_x| = |Fl(x) - x| \tag{2.1}$$

Another commonly used error metric is the absolute relative error, which quantifies the magnitude of the significance of the error to the exact value.

$$|\epsilon_x| = \left| \frac{Fl(x) - x}{x} \right| \tag{2.2}$$

The upper bound of the absolute relative error is commonly used to approximate the error in floating point arithmetic operations. This upper bound will depend on the rounding technique used in the arithmetic operations [4] [5]. In a single floating point operation, this is denoted as the machine absolute relative error upper bound, ϵ_{mach}. When the rounding scheme used is the truncation, the machine relative error upper bound is

$$|\epsilon_{mach}| = 2^{-m+1} \tag{2.3}$$

When using the rounding-to-nearest scheme, the machine absolute relative error upper bound of a single floating point operation is

$$|\epsilon_{mach}| = 2^{-m}, \tag{2.4}$$

where m is the number of bits used to represent the significand in a floating point representation. The previous upper bounds are for uniform distributions of the operands in a computation. Tsao studied the distribution of the absolute relative error for two rounding techniques in various floating point arithmetic operations [6]. The rounding schemes used were rounding-to-nearest and truncation. The study used the most significant digits of random integers to derive the relative error distributions of rounding schemes in floating point arithmetic operations. The results describe the relative error distribution of rounding schemes as a combination of uniform and reciprocal distributions. For truncation, the absolute relative error distribution is described as

$$P(|\epsilon|) = \left\{ \begin{array}{ll} \frac{\beta^{m-1}}{\log_e \beta}(\beta - 1) & 0 \leq \epsilon_0 \leq \beta^{-m} \\ \frac{1}{\log_e \beta}\left(\frac{1}{\epsilon_0} - \beta^{m-1}\right) & \beta^{-m} \leq \epsilon_0 < \beta^{-m+1} \end{array} \right\} \tag{2.5}$$

For rounding-to-nearest scheme, the absolute relative error distribution is described as

$$P(|\epsilon|) = \left\{ \begin{array}{ll} \frac{\beta^{m-1}}{\log_e \beta}(\beta - 1) & \frac{-\beta^{-m}}{2} \leq \epsilon_0 \leq \frac{\beta^{-m}}{2} \\ \frac{1}{log_e \beta}\left(\frac{1}{2|\epsilon_R|} - \beta^{m-1}\right) & \frac{\beta^{-m}}{2} < |\epsilon_0| < \frac{\beta^{-m+1}}{2} \end{array} \right\}, \tag{2.6}$$

where β and m are the base and the number of bits in the significand of the floating point representation. Kuck, et. al, proposed additional error metrics for other rounding schemes, namely the average relative error and the average bias error. [7]. The average relative error measures the average magnitude of the relative error, while the average bias error measures the average tendency of the rounding technique to favor error.

2.1.1 Error Analysis

An analysis method commonly used to estimate the error of floating point computations is the forward error analysis. The purpose of this analysis is to estimate the upper bound of the absolute relative error of a floating point computation. As an example, consider the sum of a and b. The relative error is defined as

$$|\epsilon_{a+b}| = \left| \frac{(a+b) - Fl(a+b)}{a+b} \right|, \tag{2.7}$$

therefore,

$$Fl(a+b) = (a+b)(1+\epsilon_{a+b}) \tag{2.8}$$

The forward error analysis represents $Fl(a+b)$ with the floating point representation of every operand and arithmetic operation. Therefore,

$$\begin{aligned} Fl(a+b) =& (Fl(a) + Fl(b))(1+\epsilon_{sum}) \\ =& (a(1+\epsilon_a) + b(1+\epsilon_b))(1+\epsilon_{sum}) \end{aligned} \tag{2.9}$$

Here, the terms $Fl(a)$ and $Fl(b)$ are defined using equation 2.2. The term ϵ_{sum} refers to the relative error of the sum operation caused by the rounding technique. Since

$$\left\{ \begin{array}{c} |\epsilon_a| \\ |\epsilon_b| \\ |\epsilon_{sum}| \end{array} \right\} \leq |\epsilon_{mach}|, \tag{2.10}$$

then

$$Fl(a+b) \leq (a+b)(1+|\epsilon_{mach}|)^2 \tag{2.11}$$

Therefore,

$$|\epsilon_{a+b}| \leq (1+|\epsilon_{mach}|)^2 - 1 \tag{2.12}$$

2.2 Literature Review

Stoutemyer performed pre-linearized forward error analysis on mathematical expressions using the computer algebra language *REDUCE* [8]. The analysis he performed was to determine the relative inherent and rounding errors of floating point operations. His analysis technique ignores the error caused by the product of relative inherent and rounding errors. He determined "error propagation rules" for basic arithmetic operations and for more transcendental functions as the power, logarithm, sine, cosine, tangent, and others. He performed the error analysis on the equivalent arithmetic expressions: $a^2 - b^2$ and $(a+b)(a-b)$, both obtaining the same relative inherent error. No resource consumption or latency analysis was provided in this study.

Yun and Lee performed fixed-point error analysis on three fast DCT (FCT) algorithms [9]. The FCT algorithms analyzed were those proposed by Lee, Hou and Vetterli. In their error analysis they used a statistical model to estimate the absolute error mean and variance of the FCTs algorithms. The results showed that the Vetterli's FCT provided best results in terms of variance error and signal-to-noise ratio (SNR). In the analysis, the Direct DCT computation was considered as base line; and obtained the best results in the variance error. In the study, no resource consumption or latency information of the algorithms was provided. Therefore no tradeoffs can be assumed for the algorithms.

Yun and Lee later performed the same fixed-point error analysis on the inverse FCT (IFCT) algorithms proposed by Lee, Hou and Vetterli [10]. The results showed

that Hou's algorithms had better accuracy performance in terms of error variance. Their work did not perform a resource consumption or latency analysis to the evaluated algorithms.

Harish and Prabhu also performed a fixed-point error analysis for the FCT proposed by Hou and Makhoul [11]. The error analysis used a statistical model to estimate the error variance of the algorithms. The results showed that the Makhoul's algorithm obtained better results in the error variance. They presented a vague comparison between resource consumption and latency of the algorithms. Their comparison established that Hou's algorithm was twice faster and it used half the memory used by Makhoul's algorithm. The latency comparison was based on the number of multiplication operations. They also compared the algorithm complexity in terms of design interpretation, in which the Hou's algorithm resulted in disadvantage.

Rao and Prabhu performed a fixed-point error analysis for various radix-4 fast DHT (FHT) [12]. The algorithms studied were the radix-4, the radix-2, and the split-radix decimation-in-time (DIT). Their error analysis used a statistical model to estimate the error variance of the algorithms. The results of the analysis showed that the radix-4 algorithms obtained better variance error and NRS performance than the radix-2 and the split radix. No resources consumption or latency analysis was performed to the algorithms under evaluation.

Glaros and Carayannis introduced an analytic methodology for fixed-point error analysis for various Toeplitz algorithms [13]. The analysis was performed on the Schur and the split Schur algorithms. The their exact and first-order error analysis used a statistical model to estimate error variance of the algorithms. The analysis showed the Schur algorithm with better accuracy performance than the split Schur algorithm. In their work, no resource consumption and latency analysis was performed.

Tao performed a floating point error analysis for matrix multiplication and matrix chain product computation [14]. The fast matrix multiplication algorithms analyzed

were those proposed by Winograd and Strassen. Both algorithms were compared to the standard algorithms for matrix multiplication computation. The error analysis used was the complexity error analysis that used as metric the absolute error. Tao's work showed that the standard algorithms provided a minimal error growth compared to Winograd's and Strassen's. From the three algorithms, Strassen's provided the worst error upper bound. This work did not perform resource consumption or latency analysis to the studied matrix multiplication algorithms.

Pitas and Strintzis studied the accuracy of floating point structures of three different algorithms used to compute two-dimensional fast DFT (FFT) [15]. The studied algorithms were the conventional row-column FFT (RCFFT), the vector radix FFT (VRFFT), and the polynomial transform FFT (PTFFT). A error analysis used a statistical model to estimate the error variance of the algorithms. The results showed that the VRFFT and PTFFT had similar error performances and both performed better than the CRFFT. Again, no latency analysis was provided in this study.

Table 2.1 shows a summary of the previous work found in the literature. As seen in the column of limitations, most of the previous works have no resource consumption nor latency analysis presented. When studying algorithms to be implemented on hardware, those attributes are as important as the accuracy. An essential part of this work is to analyze the accuracy and hardware performance of the discrete transforms and their fast algorithms.

Table 2.1 : Summary of previous work.

Author(s)	Error Analysis	Error Metric	Number Format	Algorithms	Limitation(s)
Stoutemyer [8]	Forward Error Analysis	Relative Error	Floating	$a^2 - b^2$, $(a+b)(a-b)$	No resource consumption or latency analysis.
Yun and Lee [9]	Statistical Model	Absolute Error Mean and Variance	Fixed	FCT	No resource consumption or latency analysis.
Yun and Lee [10]	Statistical Model	Absolute Error Variance	Fixed	IFCT	No resource consumption or latency analysis.
Harish and Prabhu [11]	Statistical Model	Absolute Error Variance	Fixed	FCT	Vague assumption of resource consumption and latency.
Rao and Prabhu [12]	Statistical Model	Absolute Error Variance	Fixed	FHT	No resource consumption or latency analysis.
Glaros and Carayannis [13]	Statistical Model	Absolute Error Variance	Fixed	Toeplitz	No resource consumption or latency analysis.
Tao [14]	Complexity Error Analysis	Absolute Error	Floating	Matrix Multiplication	No resource consumption or latency analysis.
Pitas and Strintzis [15]	Statistical Model	Absolute Error Variance	Floating	2-D FFT	No latency analysis.

Chapter 3
Problem Statement, Hypothesis, and Objectives

During the process of designing a digital arithmetic algorithm, most research efforts are centered in finding algorithms that reduce resource consumption or improve the latency of the operation. Usually, the accuracy is not a parameter for consideration during the research of the appropriate algorithm. Those analyzing accuracy rarely consider hardware performance parameters. The problem addressed in this work is that of determining how the arithmetic structures of discrete transforms and their fast algorithms impacts both accuracy and hardware performance.

3.1 Hypothesis

The work reported was developed under the hypothesis that the arithmetic structure of discrete transforms and their fast algorithms have different impact in the accuracy of the results.

3.2 Objectives

Below is a description of the general objective and the specific objectives of this work.

General Objective

Analyze the accuracy difference between the discrete transforms and their fast algorithms using accuracy quantification methods. These methods were based on statistical and error analysis. Also, analyze the hardware performance of the discrete transforms and their fast algorithms using a FPGA synthesis methodology.

Specific Objectives

1. Develop a methodology for accuracy quantification of discrete transforms and their fast algorithms
2. Synthesize hardware designs of discrete transforms and their fast algorithms structures
3. Analyze the accuracy of discrete transforms and their fast algorithms structures
4. Study the hardware performance advantages of discrete transforms and their fast algorithms
5. Describe the behavior of accuracy of discrete transforms and their fast algorithms based on their arithmetic structures
6. Identify structures where it is possible to improve the accuracy and how the improvements affect the hardware performance

Chapter 4
Methodology

The objectives of this work were based on the analysis of accuracy and hardware performance of the discrete transforms and their fast algorithms. For the data compilation of accuracy and hardware performance, quantitative methods were employed.

For the accuracy quantification the metric used was the normwise relative error; the methods used were of estimation and experimentation. The main tasks performed for the accuracy quantification were:

- **Software Designs** - Generate software designs of the discrete transforms and their fast algorithms
- **Experimental Data Compilation** - Run experiments on the software design of the algorithms for the compilation of experimental data
- **Error Analysis** - Forward error analysis of the discrete transforms and their fast algorithms, for the estimation of the error in the results
- **Error Analysis Validation** - Validation of the error analysis using the experimental data
- **Statistical Analysis** - Apply statistical test to the experimental data to compare the accuracy between the discrete transforms and their fast algorithms

For the performance quantification, FPGA design methods were employed. The main tasks performed for the performance quantification were:

- **Hardware Designs** - Generate hardware designs of the discrete transforms and their fast algorithms

- **Structures Synthesis** - FPGA synthesis of the hardware design of the discrete transforms and their fast algorithms
- **Resources Report** - Generate resources consumption reports from the synthesized hardware designs
- **Latency Report** - Generate latency reports from the synthesized hardware designs

The main software platforms used for some of these tasks were:

- MATLAB
- Xilinx Vivado

Figure 4.1 shows the relation among these tasks and the results analysis.

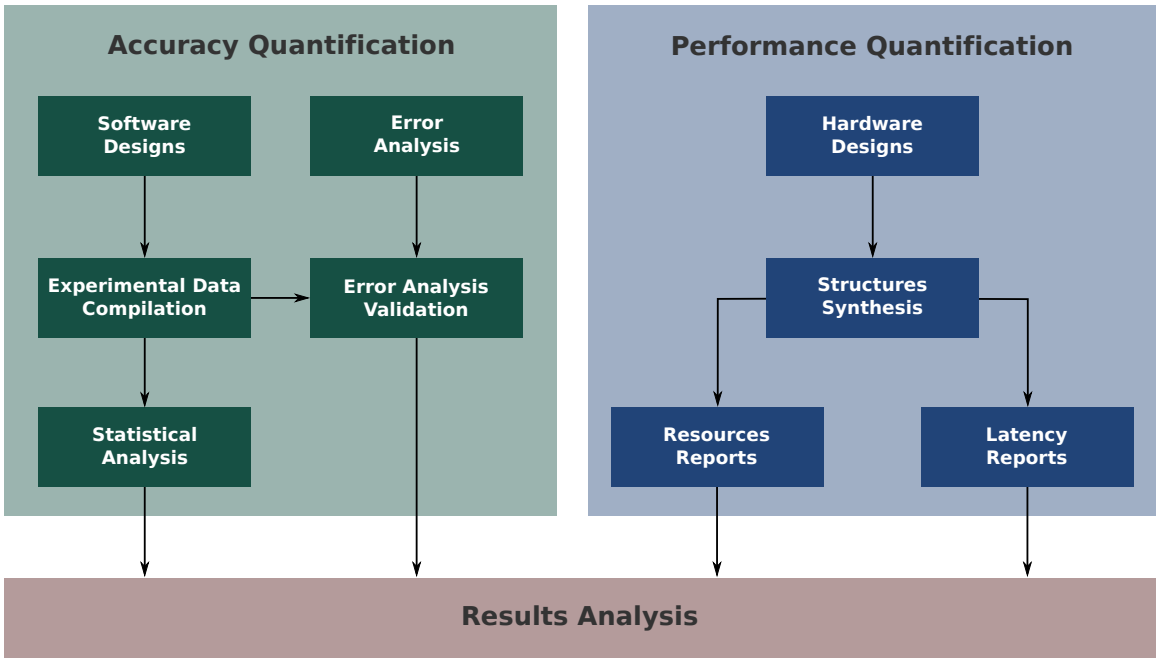


Figure 4.1 : Methodology

The next sections of the methodology are organized as follows. Section 4.1 provides a theoretical background of the selected discrete transforms and their fast algorithms. Also, provides a example of their arithmetic structure, which are used for the software and hardware designs. Section 4.2 provides additional details of the software and hardware designs. The Section 4.3 discussed the experimental design

used for the Experimental Data Compilation. Section 4.4 discussed the method used for the Error Analysis and the approach used for its validation. Finally, Section 4.5 discusses the approach taken for the Statistical Analysis.

4.1 Discrete Transforms and their Fast Algorithms

The discrete transforms selected for this study belong to the sinusoidal unitary transforms class [16]. The chosen discrete transforms are widely used in applications of digital signal processing. The applications include image compressing, spectral analysis and spread spectrum systems [16–18]. Before discussing the evaluated discrete transforms formulations, we present a review of discrete transform operations in their matrix representation.

The operator of a discrete transform of the signal $x[n]$, of length N, is represented as

$$X[k] = DT\{x[n]\}, \quad k = 0, 1, \cdots, N-1, \tag{4.1}$$

where DT denotes the discrete transform operation over the discrete signal $x[n]$, and $X[k]$ denotes the result signal of the operation.

The matrix representation of a discrete transform operation has the form of

$$X_N = DT_N x_N, \tag{4.2}$$

where X_N and x_n are column vectors, of size N, of the signals $X[k]$ and $x[n]$, defined as

$$X_N = \begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} \tag{4.3}$$

$$x_N = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}, \tag{4.4}$$

and, DT_N is denoted as the discrete transform matrix of size $N \times N$. The matrix coefficient depend on the discrete transform operation. The discrete transforms fast algorithms used Kronecker product ($\otimes$) and direct sum ($\oplus$) operators, along with identity and permutation matrices, to represent the DT_N matrix.

The Kronecker product of the matrices A, of size $m \times n$, and B, of size $p \times q$, is defined as

$$A \otimes B = \begin{bmatrix} a_{0,0}B & \cdots & a_{0,n-1}B \\ \vdots & \ddots & \vdots \\ a_{m-1,0}B & \cdots & a_{m-1,n-1}B \end{bmatrix} \tag{4.5}$$

The direct sum of coefficients and matrices are defined in Equations 4.6 and 4.7, respectively. The direct sum of coefficients results in a diagonal matrix, and the direct sum of matrices results in a block diagonal matrix.

$$\bigoplus_{i=0}^{N-1} a_i = a_0 \oplus a_1 \oplus \cdots \oplus a_{N-1} = \begin{bmatrix} a_0 & & & \\ & a_1 & & \\ & & \ddots & \\ & & & a_{n-1} \end{bmatrix} \tag{4.6}$$

$$\bigoplus_{i=0}^{N-1} A_i = A_0 \oplus A_1 \oplus \cdots \oplus A_{N-1} = \begin{bmatrix} A_0 & & & \\ & A_1 & & \\ & & \ddots & \\ & & & A_{n-1} \end{bmatrix} \tag{4.7}$$

The identity matrix is a diagonal matrix where the value of the diagonal elements is one. The identity matrix formal definition is

$$[I_N]_{i,j} = \left\{ \begin{array}{ll} 1, & i = j \\ 0, & otherwise \end{array} \right\} \tag{4.8}$$

Permutation matrices are characterized by having a single "1" in each column and row, and "0" elsewhere. The purpose of permutation matrices is to rearrange the rows or columns of a matrix or vector. A permutation matrix of size $N \times N$ is defined as

$$P_N = \left[\begin{array}{c} e_{P_N(0)} \\ e_{P_N(1)} \\ \vdots \\ e_{P_N(N-1)} \end{array} \right], \tag{4.9}$$

where $e_{P_N(i)}$ denotes the ith row vector of length N with "1" in the $P_N(i)$th position and "0" elsewhere. The term $P_N(i)$ denotes the permutation matrix function. The multiplication of a permutation matrix P_N with a column vector x_N, results as

$$P_N x_N = \left[\begin{array}{c} x[P_N(0)] \\ x[P_N(1)] \\ \vdots \\ x[P_N(N-1)] \end{array} \right] \tag{4.10}$$

A permutation matrix used in most of the matrix representation of the discrete transforms fast algorithms is the Stride-s permutation matrix. The Stride-st permutation matrix of size $N \times N$ is defined as

$$[L^N_{st}]_{i,j} = \left\{ \begin{array}{ll} 1, & (i \cdot st \bmod N) + \lfloor \frac{i \cdot st}{N} \rfloor = j \\ 0, & otherwise \end{array} \right\} \tag{4.11}$$

The evaluated discrete transforms include Fourier, Hartley, and cosine. The Fourier transform yields complex results, while the other evaluated transforms produce real results. The selected fast algorithms were based on their structure. In the case of Fourier and cosine transforms, we selected parallel and decimation fast algorithms to observe the accuracy behavior when using complex and real operators. In the case of Hartley transform, decimation-in-time and decimation-in-frequency fast algorithms were evaluated to observe the effect on accuracy caused by the kind of decimation. Next is the theoretical background of the selected discrete transforms and their fast algorithms, including their matrix representations and arithmetic structure interpretations. These arithmetic structures are used for their software and hardware designs.

4.1.1 Discrete Fourier Transform

The main purpose of the discrete Fourier transform (DFT) is to transform a time domain signal into the frequency domain. This transformation provides information about the frequencies present in the time domain signal. The N-point DFT of a signal $x[n]$, is defined as

$$X[k] = DFT\{x[n]\} = \sum_{n=0}^{N-1} x[n] w_N^{kn}, \quad k = 0, 1, ..., N-1, \tag{4.12}$$

where

$$w_N = e^{-2\pi \hat{\imath}/N} \quad \hat{\imath} = \sqrt{-1} \tag{4.13}$$

Equation 4.14 describes the matrix-vector representation of this transformation.

$$X_N = DFT_N x_N, \tag{4.14}$$

where DFT_N denotes the DFT matrix of size $N \times N$, where the matrix coefficients are defined as

$$[DFT_N]_{i,j} = w_N^{ij} \tag{4.15}$$

Figure 4.2 shows the structural interpretation of the 8-point Direct DFT.

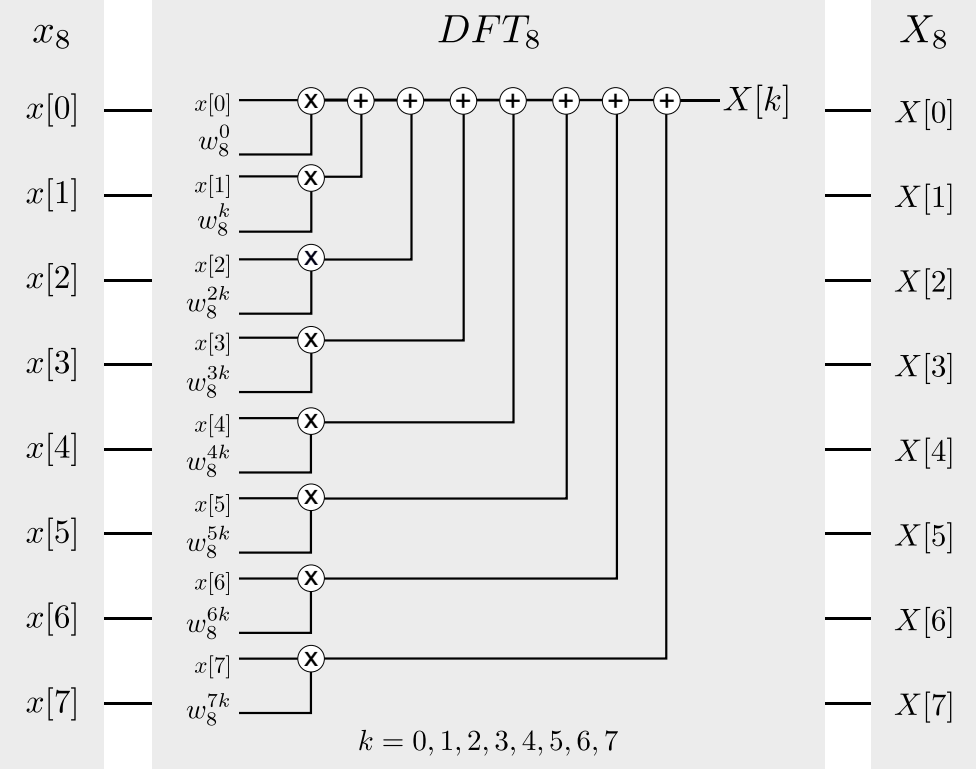


Figure 4.2 : structural interpretation of the 8-point Direct DFT

A N-point DFT requires an amount of N^2 complex multiplication. Exploiting the properties of the DFT, fast algorithms were achieved, which reduced the amount of complex multiplications.

Cooley-Tukey FFT

The factorization presented by Cooley and Tukey is the most commonly used algorithms for the fast discrete Fourier transform (FFT) [19]. They factorize a N-point DFT, when $N = r_1 r_2$, as follows,

$$X[k] = \sum_{n''=0}^{r_1-1} \sum_{n'=0}^{r_2-1} x[n'r_1 + n''] w_N^{n'r_1 k} w_N^{n''k}, \quad k = 0, 1, ..., N-1 \tag{4.16}$$

When $r_1 = 2$, the inner sum in Equation 4.16 represents a $N/2$-point DFT for the even ($n'' = 0$) and odd ($n'' = 1$) points in $x[n]$. Therefore, when $N = 2 \cdot r_2$, equation

4.16 can be rewritten as

$$X[k] = DFT_{r_2}\{x[2n']\}w_N^{0k} + DFT_{r_2}\{x[2n'+1]\}w_N^{1k} \tag{4.17}$$

Equation 4.17 can be described as

$$X[k] = \left\{ \begin{array}{ll} DFT_{r_2}\{x[2n']\}w_N^{0\overline{k}} + DFT_{r_2}\{x[2n'+1]\}w_N^{1\overline{k}}, & k = 0, 1, ..., r_2 - 1 \\ DFT_{r_2}\{x[2n']\}w_N^{0\overline{k}} - DFT_{r_2}\{x[2n'+1]\}w_N^{1\overline{k}}, & k = r_2, ..., N - 1 \end{array} \right\}, \tag{4.18}$$

where

$$\overline{k} = k \bmod N/2 \tag{4.19}$$

The matrix factorization of the Cooley-Tukey FFT is described as

$$X_N = (DFT_{r_1} \otimes I_{r_2})T_{r_2}^N(I_{r_1} \otimes DFT_{r_2})L_{r_1}^N x_N, \tag{4.20}$$

where $T_{r_2}^N$ is a diagonal matrix defined as

$$T_{r_2}^{r_1 r_2} = \bigoplus_{i=0}^{r_1-1} \left(\bigoplus_{j=0}^{r_2-1} w_{r_1 r_2}^{ij} \right) \tag{4.21}$$

Figure 4.3 shows the structural interpretation of the 8-point Cooley-Tukey FFT, when $r_1 = 2$ and $r_2 = 4$.

The main advantage of this FFT formulation is a reduction in the amount of complex multiplication from N^2 to approximately $N(r_1 + r_2)$. Its disadvantage is that limits the value of N-point to non-prime numbers.

Pease FFT

Pease presented this formulation with the purpose by generating a parallel FFT architecture [20]. He accomplished the architecture by manipulating the DFT matrix using bitreversal and stride-2 permutations matrices. Equation 4.22 presents the

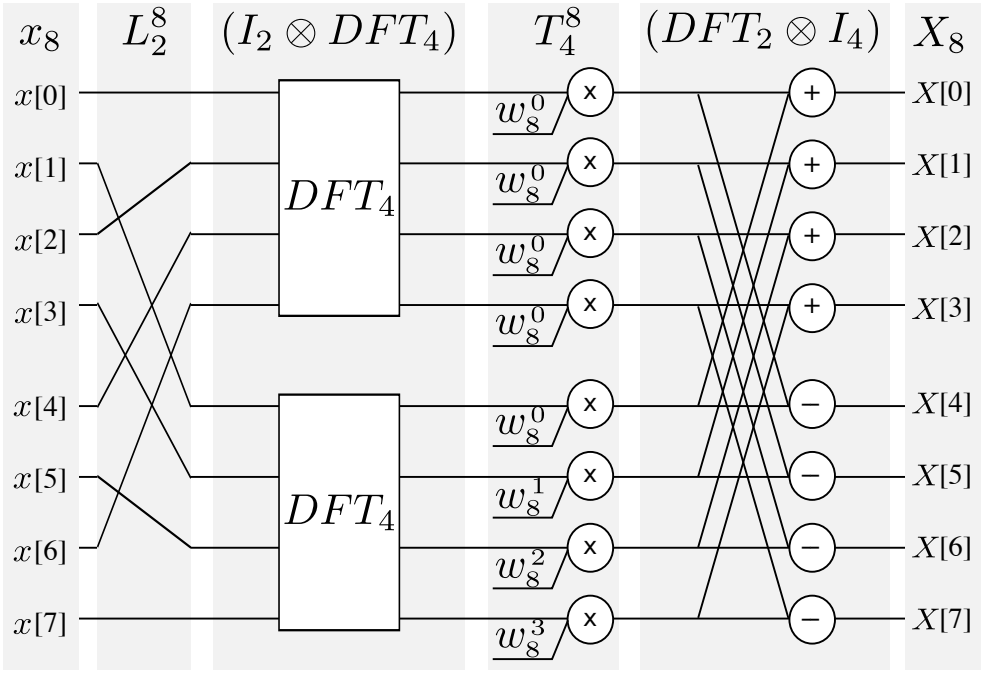


Figure 4.3 : Structural interpretation of the 8-point Cooley-Tukey FFT

matrix factorization formulated by Pease.

$$X_{2^p} = \left\{ \prod_{i=1}^{p} L_2^{2^p} (I_{2^{p-1}} \otimes DFT_2) T_i' \right\} R_{2^p} x_{2^p} \tag{4.22}$$

where R_{2^p} is the Bit-reversal permutation matrix, and T_i' is a diagonal matrix. The Bit-reversal permutation matrix is defined as

$$R_{2^p} = \prod_{i=1}^{p} (I_{2^{p-i}} \otimes L_2^{2^i}) \tag{4.23}$$

The diagonal matrix T_i' is defined as [21]

$$T_i' = \bigoplus_{j=0}^{2^{p-i}-1} (I_{2^{i-1}} \otimes W_2^{j2^{i-1}}), \tag{4.24}$$

where the matrix W_2 is defined as

$$W_2 = \begin{bmatrix} w_{2^p}^0 & 0 \\ 0 & w_{2^p}^1 \end{bmatrix} \tag{4.25}$$

Figure 4.4 presents the structural interpretation of the 8-point Pease FFT.

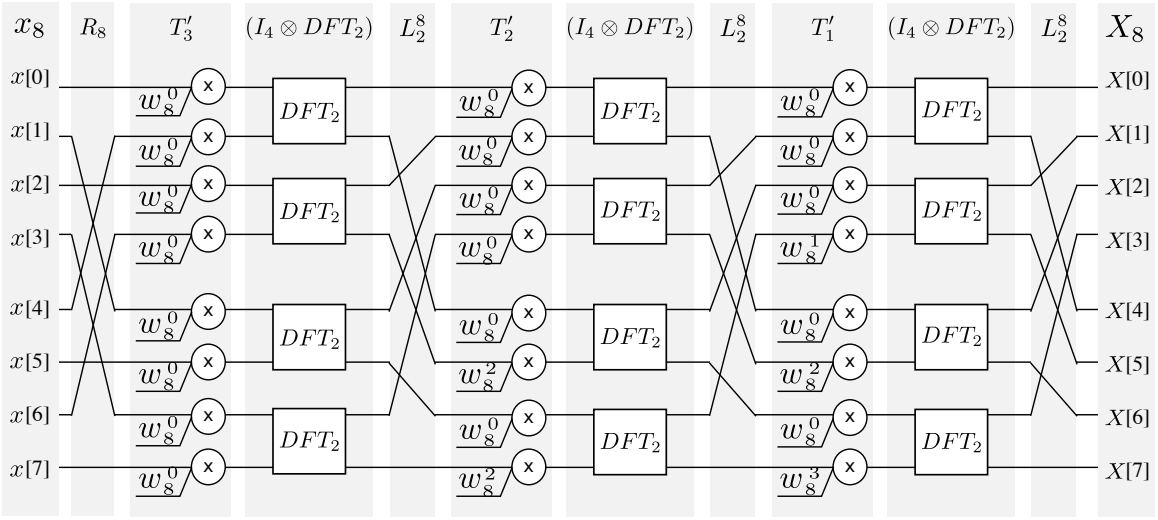


Figure 4.4 : structural interpretation of the 8-point Pease FFT

This FFT formulation has two main advantages: a reduction of complex multiplications from N^2 to $Nlog_2N$, and a regular structure. The drawback of this formulation is that limits the value of N to a power of two.

4.1.2 Discrete Hartley Transform

The discrete Hartley transform (DHT) was initially proposed as an analogous discrete transform of the DFT. The difference between these discrete transforms is that the DHT produces real results and the DFT yields complex results. The N-point DHT of a signal $x[n]$, is defined as

$$X[k] = DHT_N\{x[n]\} = \sum_{n=0}^{N-1} x[n] cas\left(\frac{2\pi kn}{N}\right), k = 0, 1, ..., N-1, \tag{4.26}$$

where $cas(\theta) = cos(\theta) + sin(\theta)$. Equation 4.27 describes the matrix-vector representation of this transformation.

$$X_N = DHT_N x_N, \tag{4.27}$$

where DHT_N denotes the DHT matrix of size $N \times N$ and its coefficients are defined as

$$[DHT_N]_{i,j} = cas\left(\frac{2\pi ij}{N}\right) \tag{4.28}$$

Figure 4.5 shows the structural interpretation of the 8-point Direct DHT.

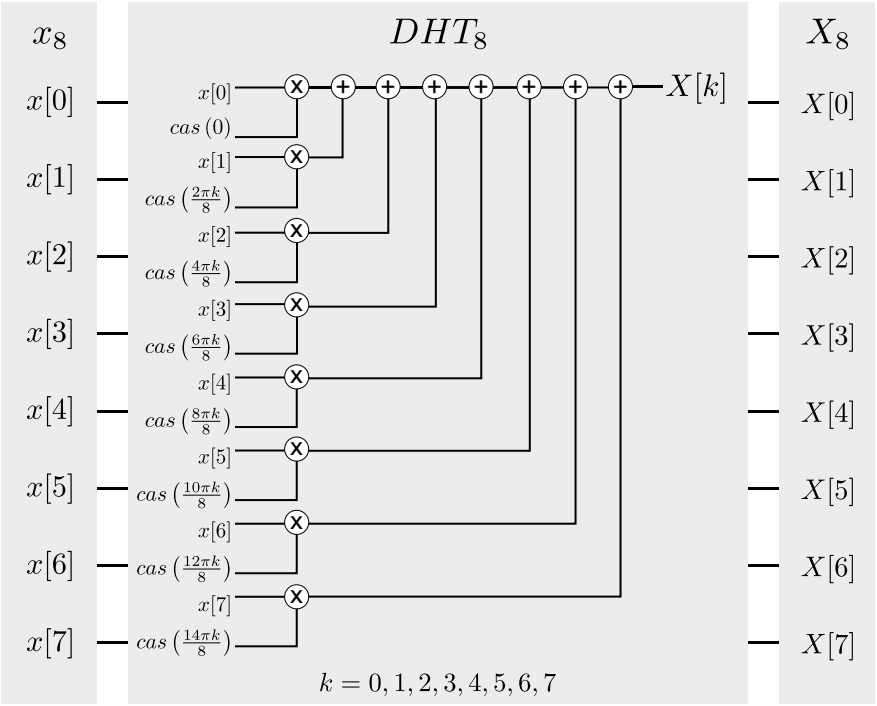


Figure 4.5 : Structural interpretation of the 8-point Direct DHT

An N-point DHT computation requires N^2 multiplication. Exploiting the properties of the DHT matrix, fast algorithms were achieved, which reduced the amount of multiplications [22, 23].

Bracewell FHT

Bracewell proposed a radix-2 decimation-in-time (DIT) fast discrete Hartley transform (FHT) [22]. He separated the sum of product in Equation 4.26 into the even and odd points of the signal $x[n]$ as shown below

$$X[k] = \sum_{n'=0}^{\frac{N}{2}-1} x[2n'] cas\left(\frac{2\pi kn'}{N/2}\right) + \sum_{n'=0}^{\frac{N}{2}-1} x[2n'+1] cas\left(\frac{2\pi k(n'+1/2)}{N/2}\right) \tag{4.29}$$

Equation 4.29 can be rewritten for the first and second half of the points in $X[k]$ as shown in Equations 4.30 and 4.30, for $k' = 0, 1, ..., N/2 - 1$.

$$X[k'] = \sum_{n'=0}^{\frac{N}{2}-1} x[2n']cas\left(\frac{2\pi k'n'}{N/2}\right) + \sum_{n'=0}^{\frac{N}{2}-1} x[2n'+1]cas\left(\frac{2\pi k'(n'+\frac{1}{2})}{N/2}\right) \tag{4.30}$$

$$X[k'+N/2] = \sum_{n'=0}^{\frac{N}{2}-1} x[2n']cas\left(\frac{2\pi k'n'}{N/2}\right) - \sum_{n'=0}^{\frac{N}{2}-1} x[2n'+1]cas\left(\frac{2\pi k'(n'+\frac{1}{2})}{N/2}\right) \tag{4.31}$$

The matrix representation of this formulation is

$$X_N = (DFT_2 \otimes I_{N/2})(DHT_{N/2} \oplus \widehat{DHT}_{N/2})L_2^N x_N, \tag{4.32}$$

where $\widehat{DHT}_{N/2}$ denotes a matrix of size $N/2 \times N/2$ and its coefficients are defined as

$$[\widehat{DHT}_N]_{i,j} = cas\left(\frac{2\pi i(j+1/2)}{N}\right) \tag{4.33}$$

Figure 4.6 shows the structural interpretation of the 8-point Bracewell FHT. The main advantage of this formulation is that it reduces the amount of multiplications from N^2 to $\frac{N^2}{2}$. Its disadvantage is that limits the value of N to multiples of two.

Hou FHT

Hou proposed a radix-2 decimation-in-frequency (DIF) FHT [23]. This formulation first separates the sum of products in Equation 4.26 into the first and second half of points of the signal $x[n]$ as follows

$$\begin{aligned} X[k] &= \sum_{n'=0}^{r-1} x[n']cas\left(\frac{2\pi kn'}{N}\right) + \sum_{n'=0}^{r-1} x[n'+N/2]cas\left(\frac{2\pi k(n'+N/2)}{N}\right) \\ &= \sum_{n'=0}^{r-1} (x[n'] + (-1)^k x[n'+N/2])cas\left(\frac{2\pi kn'}{N}\right) \end{aligned} \tag{4.34}$$

Then Hou describes Equation 4.34 for the even and odd points of $X[k]$ as in Equations 4.35 and 4.36, for $k' = 0, 1, ..., N/2 - 1$.

$$X[2k'] = \sum_{n'=0}^{r-1} (x[n'] + x[n'+N/2])cas\left(\frac{2\pi k'n'}{N/2}\right) \tag{4.35}$$

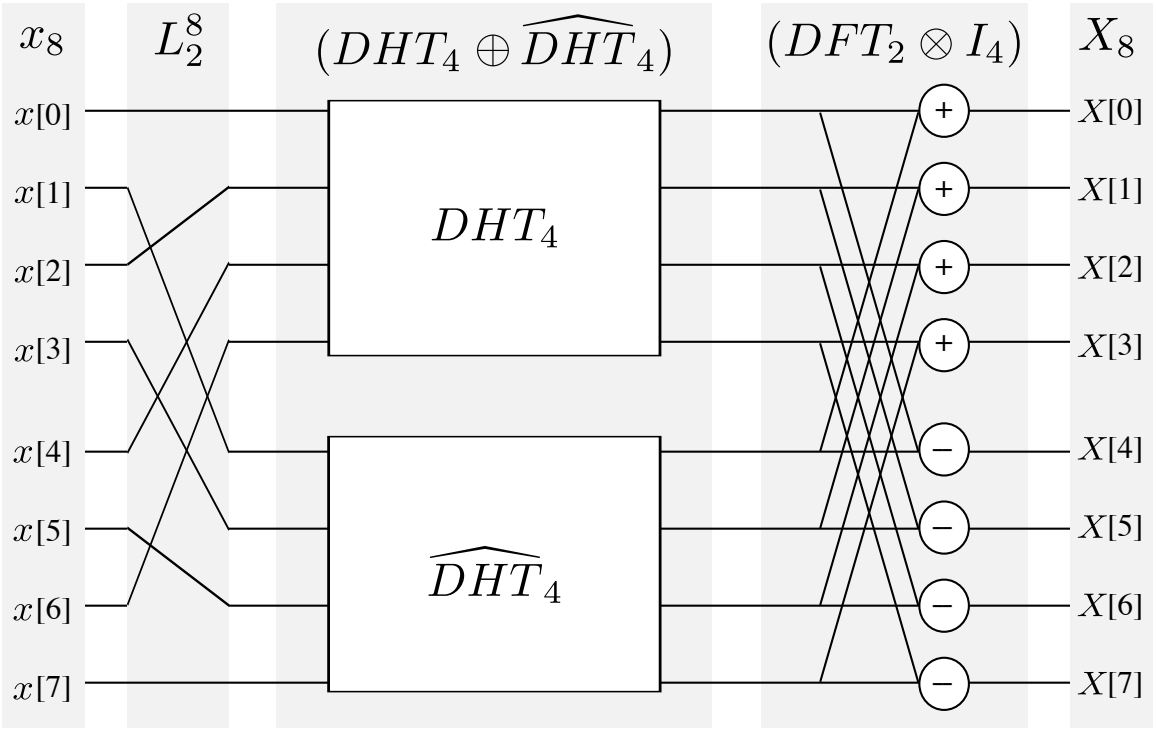


Figure 4.6 : Structural interpretation of the 8-point Bracewell FHT

$$X[2k'+1] = \sum_{n'=0}^{r-1} (x[n'] - x[n'+N/2]) cas\left(\frac{2\pi(k'+\frac{1}{2})n'}{N/2}\right) \tag{4.36}$$

The matrix representation of this formulation is

$$X_N = L_{N/2}^N (DHT_{N/2} \oplus \widehat{DHT}_{N/2}^T)(DFT_2 \otimes I_{N/2}) x_N, \tag{4.37}$$

where $\widehat{DHT}_{N/2}^T$ is the transpose of matrix $\widehat{DHT}_{N/2}$ of size $N/2 \times N/2$, as established by Equation 4.33. The coefficients of the matrix $\widehat{DHT}_{N/2}^T$ are defined as

$$[\widehat{DHT}_N^T]_{i,j} = cas\left(\frac{2\pi(i+1/2)j}{N}\right) \tag{4.38}$$

Figure 4.7 shows the structural interpretation of the 8-point Hou FHT. This FHT formulation has the same advantages and disadvantages of the Bracewell FHT formulation.

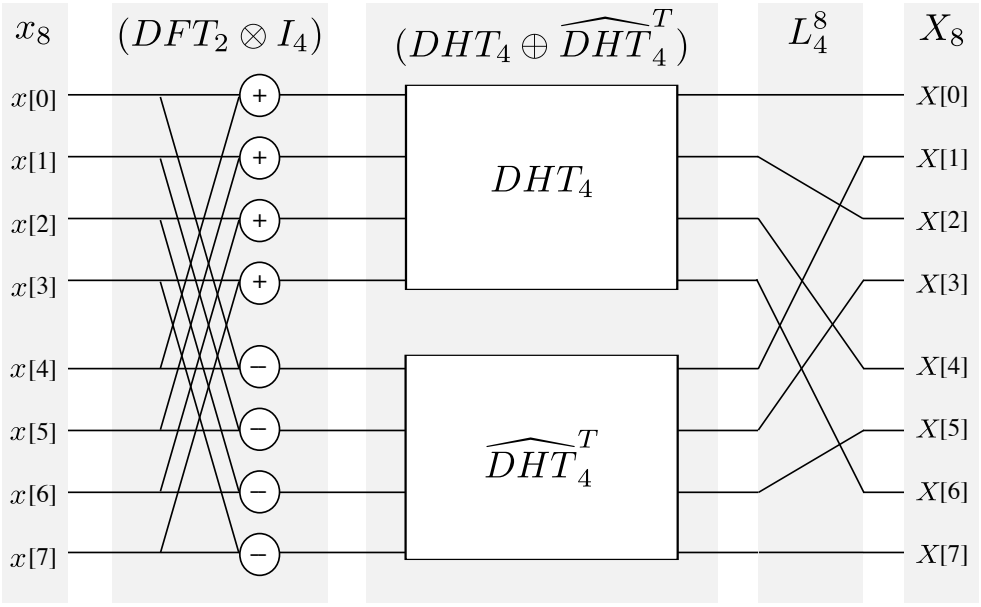


Figure 4.7 : Structural interpretation of the 8-point Hou FHT

4.1.3 Discrete Cosine Transform

The discrete cosine transform (DCT) is of great use in image compression algorithms [16, 24]. This work used the even, type II DCT, defined as follows

$$\begin{aligned} X[k] = DCT_N^{II}\{x[n]\} &= sf_N(k)\overline{DCT}_N^{II}\{x[n]\}, \qquad k = 0, 1, ..., N-1 \\ &= sf_N(k)\sum_{n=0}^{N-1} x[n] cos\left(\frac{\pi k\left(n+\frac{1}{2}\right)}{N}\right) \end{aligned} \tag{4.39}$$

where, $sf_N(k)$ is denoted as a *scaling factor*, and $\overline{DCT}_N^{II}\{x[n]\}$ as an *unscaled* DCT$^{\text{II}}$ operation. These are defined as

$$sf_N(i) = \left\{ \begin{array}{ll} \sqrt{\frac{1}{N}}, & k = 0 \\ \sqrt{\frac{2}{N}}, & k \neq 0 \end{array} \right\} \tag{4.40}$$

$$\overline{DCT}_N^{II}\{z[n]\} = \sum_{n=0}^{N-1} z[n] cos\left(\frac{\pi k\left(n+\frac{1}{2}\right)}{N}\right), \quad k = 0, 1, ..., N-1 \tag{4.41}$$

The matrix representation of the a N-point DCT computation is defined as

$$X_N = SF_N\overline{DCT}_N^{II}x_N, \tag{4.42}$$

where SF_N is a diagonal matrix of size $N \times N$, and $\overline{DCT}_N^{II}$ is denoted as an *unscaled* DCT$^{\text{II}}$ matrix of size $N \times N$. These are defined as

$$SF_N = \bigoplus_{i=0}^{N-1} sf_N(i) \tag{4.43}$$

$$[\overline{DCT}_N^{II}]_{i,j} = cos\left(\frac{\pi i\left(j+\frac{1}{2}\right)}{N}\right) \tag{4.44}$$

Figure 4.8 shows the structural interpretation of the 8-point DCT.

An N-point DCT computation of requires a total of $N(N+1)$ multiplication. Exploiting the properties of the $\overline{DCT}_N^{II}$ matrix, fast algorithms were achieved.

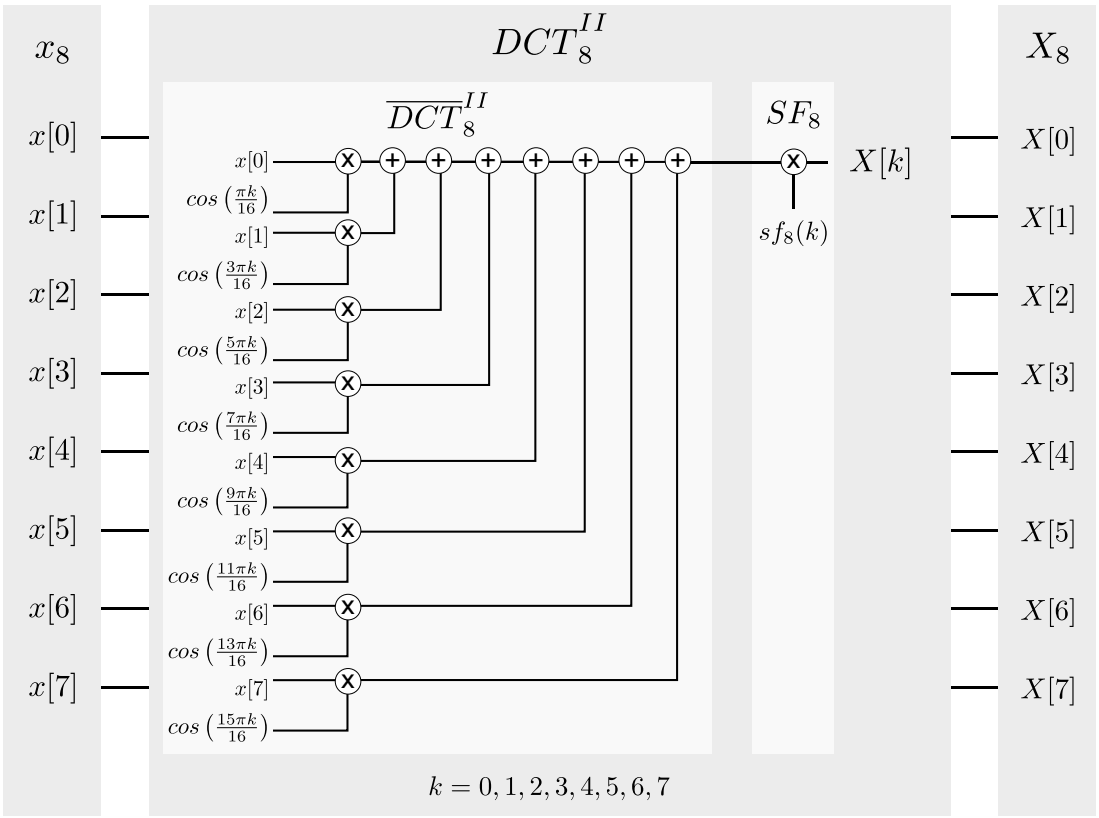


Figure 4.8 : Structural interpretation of the 8-point DCT

Nikara FCT

Nikara proposed a fast discrete cosine transform (FCT) that worked when the number of points of the DCT operation was $N = 2^p$. His formulation exploits the characteristics of the $\overline{DCT}_N^{II}$ matrix, to develop a parallel structure [25]. He used diverse permutation matrices and Kronecker product operators for his FCT formulation. His formulation is defined as follows

$$X_{2^p} = \sqrt{\frac{2}{2^p}} U_{2^p}^{(p-1)} \left\{ \prod_{s=p-1}^{1} A_{2^p}^{(s)} S_{2^p}^{(s)} \right\} A_{2^p}^{(0)} P_{2^p}^{H} x_{2^p}, \tag{4.45}$$

where

$$U_{2^p}^{(p-1)} = \prod_{i=0}^{p-2} (I_{2^{p-1}} \oplus R_{2^p - 2^{p-1}})(I_{2^i} \otimes L_{2^{p-i-1}}^{2^{p-i}}), \tag{4.46}$$

$$R_1 = I_1, \quad R_2 = I_2, \quad R_4 = L_2^4 P_4^H, \quad R_N = I_{N/4} \otimes R_4, \tag{4.47}$$

$$[P_{2^p}^H]_{k,l} = \left\{ \begin{array}{ll} 1, & l = h_{2^p}(k) \\ 0, & l \neq h_{2^p}(k) \end{array} \right\}, \tag{4.48}$$

$$h_1(0) = 0, h_{2N}(2i) = h_N(i), \quad h_{2N}(2i+1) = 2N - 1 - h_N(i), i = 0, 1, ..., N-1, \tag{4.49}$$

$$A_{2^p}^{(s)} = M_{2^p}^{(s)} D_{2^p}^{(s)} H_{2^p}^{(s)} (I_{2^{p-1}} \otimes DFT_2), \tag{4.50}$$

$$M_{2^p}^{(s)} = \bigoplus_{i=0}^{2^{p-1}-1} \begin{bmatrix} 1 & 0 \\ -\mu_s(i) & 1 \end{bmatrix}, \tag{4.51}$$

$$\mu_s(i) = \left\{ \begin{array}{ll} 0, & i \bmod 2^s = 0 \\ 1, & i \bmod 2^s \neq 0 \end{array} \right\}, \tag{4.52}$$

$$D_{2^p}^{(s)} = \bigoplus_{i=0}^{2^p-1} g_p(i, s), \tag{4.53}$$

$$g_p(i, s) = \left(2^{\mu_s(\lfloor i/2 \rfloor)} d(2^{p-s-1} + \lfloor i/2^{s+1} \rfloor)\right)^{f_p(i,s)}, \tag{4.54}$$

$$d(1) = \sqrt{0.5}; \quad d(2i) = \sqrt{0.5(1 + d(i))}; \quad d(2i+1) = \sqrt{0.5(1 - d(i))}, \tag{4.55}$$

$$f_p(i, s) = (i \bmod 2) + (1 - \tau_0(i))(1 - \tau_{p-1}(s)), \tag{4.56}$$

$$\tau_i(s) = \left\{ \begin{array}{ll} 0, & s = i \\ 1, & s \neq i \end{array} \right\}, \tag{4.57}$$

$$H_{2^p}^{(s)} = \bigoplus_{i=0}^{2^{p-2}-1} (L_2^4 R_4 L_2^4)^{\mu_{s-1}(i)}, \tag{4.58}$$

and

$$S_{2^p}^{(s)} = I_{2^{p-s-1}} \otimes L_{2^s}^{2^{s+1}} \tag{4.59}$$

Figure 4.9 shows the structural interpretation of the 8-point Nikara FCT. The main advantages of this FCT include the reduction of arithmetic computations and the regularity of the formulation. A disadvantage is the complexity of the matrices used for the matrix factorization.

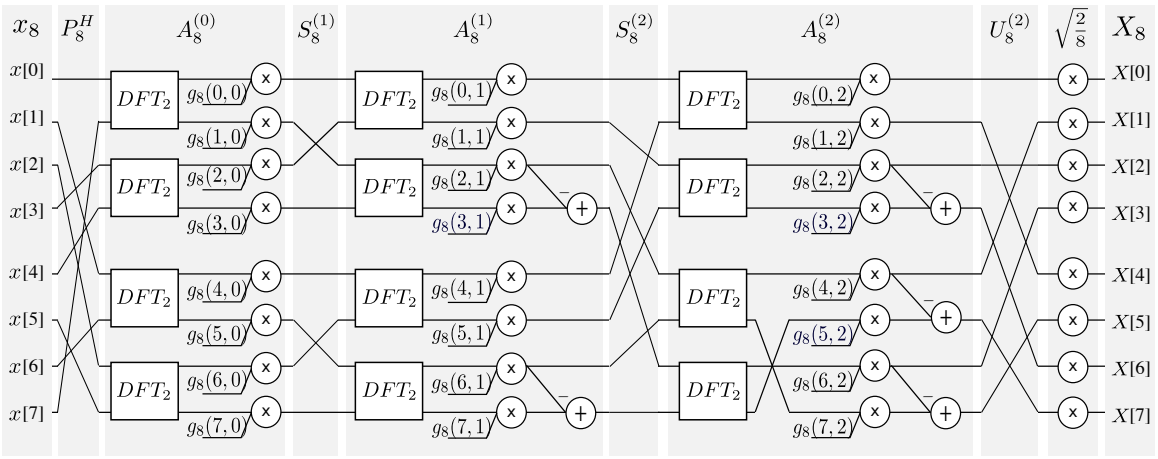


Figure 4.9 : Structural interpretation of the 8-point Nikara FCT

Translation FCT

Püschel and Moura discussed translation techniques used to formulate fast algorithms for discrete trigonometric transforms (DTT), as the DCT [26]. These techniques were employed to represent a N-point DCT using other types of DCTs when the number of points is a multiple of two. Next is shown how a $DCT_N^{II}\{x[n]\}$ can be formulated using a $\overline{DCT}_{N/2}^{II}\{z[n]\}$ and a $\overline{DCT}_{N/2}^{IV}\{y[n]\}$. The $\overline{DCT}_N^{IV}\{y[n]\}$ denotes an *unscaled* DCT$^{\text{IV}}$ operation, which is defined as

$$\overline{DCT}_N^{IV}\{y[n]\} = \sum_{n=0}^{N-1} y[n] cos\left(\frac{\pi\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)}{N}\right), \quad k = 0, 1, ..., N-1 \tag{4.60}$$

The translation technique first rewrites Equation 4.39 as follows

$$\begin{aligned} X[k] =& sf_N(k) \sum_{n'=0}^{\frac{N}{2}-1} x[n'] cos\left(\frac{\pi k\left(n'+\frac{1}{2}\right)}{N}\right) \\ &+ sf_N(k) \sum_{n'=0}^{\frac{N}{2}-1} x[N-1-n'] cos\left(\frac{\pi k\left(N-1-n'+\frac{1}{2}\right)}{N}\right) \\ =& sf_N(k) \sum_{n'=0}^{\frac{N}{2}-1} (x[n'] + (-1)^k x[N-1-n']) cos\left(\frac{\pi k\left(n'+\frac{1}{2}\right)}{N}\right) \end{aligned} \tag{4.61}$$

Then, it defines the even and odd points of the signal $X[k]$ as in Equations 4.62 and 4.63, for $k' = 0, 1, \cdots, N/2 - 1$.

$$\begin{aligned} X[2k'] &= sf_N(k) \sum_{n'=0}^{\frac{N}{2}-1} (x[n'] + x[N-1-n'])cos\left(\frac{\pi k' \left(n' + \frac{1}{2}\right)}{N/2}\right) \\ &= sf_N(k)\overline{DCT}^{II}_{N/2}\{x[n'] + x[N-1-n']\} \end{aligned} \tag{4.62}$$

$$\begin{aligned} X[2k'+1] &= sf_N(k) \sum_{n'=0}^{\frac{N}{2}-1} (x[n'] - x[N-1-n'])cos\left(\frac{\pi \left(k' + \frac{1}{2}\right)\left(n' + \frac{1}{2}\right)}{N/2}\right) \\ &= sf_N(k)\overline{DCT}^{IV}_{N/2}\{x[n'] - x[N-1-n']\} \end{aligned} \tag{4.63}$$

The matrix representation of this formulation is given by Equation 4.64

$$X_N = SF_N L^N_{N/2}(\overline{DCT}^{II}_{N/2} \oplus \overline{DCT}^{IV}_{N/2})(DFT_2 \otimes I_{N/2})(I_{N/2} \oplus \bar{I}_{N/2})x_N, \tag{4.64}$$

where $\bar{I}_{N/2}$ denotes a reflection matrix of size $N/2 \times N/2$, and $\overline{DCT}^{IV}_{N/2}$ an *unscaled* DCT$^{\text{IV}}$ matrix . These are defined as

$$[I_N]_{i,j} = \left\{ \begin{array}{ll} 1, & i = N-1-j \\ 0, & i \neq N-1-j \end{array} \right\}, \tag{4.65}$$

and

$$[\overline{DCT}^{IV}_N]_{i,j} = cos\left(\frac{\pi \left(i + \frac{1}{2}\right)\left(j + \frac{1}{2}\right)}{N}\right) \tag{4.66}$$

Figure 4.10 shows the structural interpretation of the 8-point Translation FCT. The main advantage of this formulation is that it reduces the number of multiplications to $\frac{N^2}{2}$. A disadvantage is that limits the value of N to multiples of two.

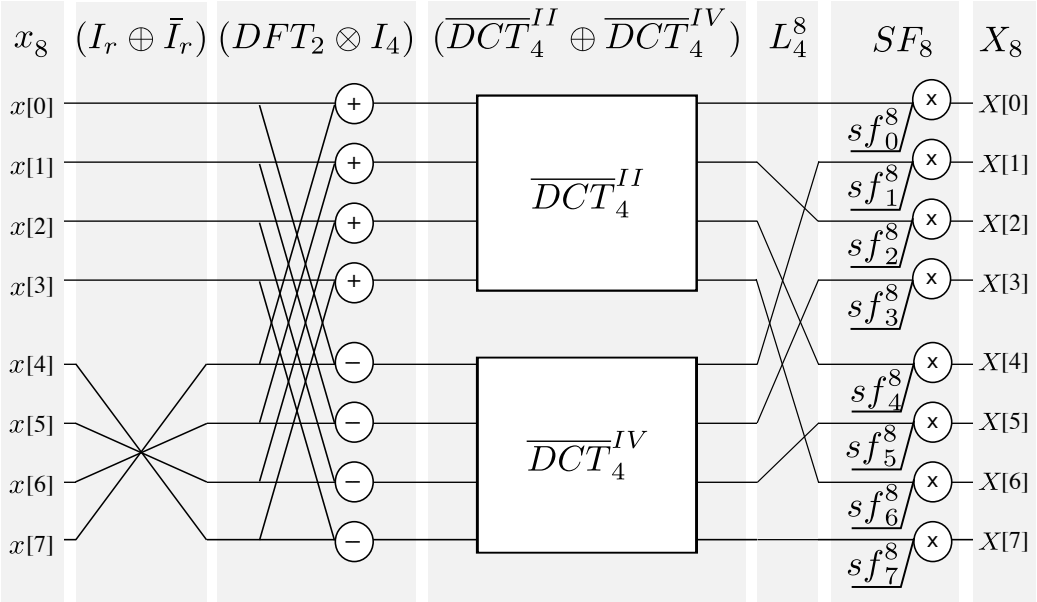


Figure 4.10 : Structural interpretation of the 8-point Translation FCT

4.2 Software and Hardware Designs

The structural interpretation of the treatments were used in the software and hardware designs. The software designs were used in the experimental design, which acquire experimental data from the discrete transforms treatments for their accuracy quantification. The hardware designs were used in the FPGA synthesis of the treatments, for the data compilation of hardware performance. The software and hardware designs of the treatments, used floating point arithmetic operators. The floating point standard used was the IEEE 754 standard for single precision representation.

The software designs of the discrete transforms treatments were develop in MATLAB. The approach was to define a MATLAB function for each discrete transform. The function accepted two parameters: x_N and *treatment*. The x_N parameter is the input signal to be transformed. With this parameter number of point of the discrete transform is defined. The *treatment* parameter define the treatment used for the discrete transform computation. The general MATLAB function flow used for a discrete transform computation was

```
function X=DiscreteTransform(x_N, treatment)
```

```
    N=length(x_N);
    if strcmp(treatment, "FDT1")
        X_N= FastDiscreteTransform1(x_N,N);
    elseif strcmp(treatment, "FDT2")
        X_N= FastDiscreteTransform2(x_N,N);
    else
        X_N= DirectDiscreteTransform(x_N,N);
    end if;
end function;
```

The hardware designs developed were FPGAs based. The design entry used VHDL as the hardware description language. The synthesis platform used for the hardware designs was Xilinx Vivado. An automated script was develop for the generation of custom VHDL files, based on the number of point of the discrete transform. The software platform used for the automated method was MATLAB. The general flow used for the VHDL file generation of the discrete transform treatments is illustrated in Figure 4.11 .

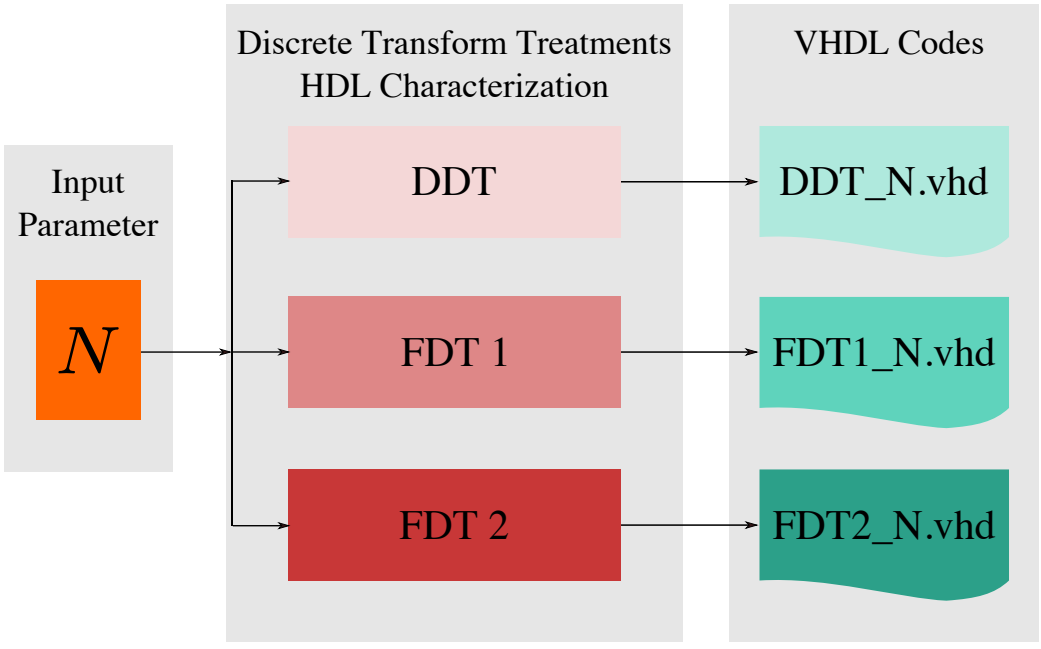


Figure 4.11 : HDL Codes Generation

Appendices A and B provide the code of the MATLAB functions developed for the software and hardware designs.

4.3 Experimental Design

The goal of the experimental design was quantifying the accuracy of the evaluated discrete transforms treatments. The metric used for accuracy quantification was the normwise relative error. In the discrete transform context, the normwise relative error is defined as

$$||\epsilon_{X_N}||_g = \frac{||Fl(X_N) - X_N||_g}{||X_N||_g}, \tag{4.67}$$

where g refers to the norm rule, $Fl(X_N)$ refers to the floating point computation of the discrete transform treatment, and X_N refers to the exact computation of the discrete transform.

This study used the two-norm, defined by Equations 4.68 and 4.69.

$$||x||_2 = \sqrt{\sum_{i=0}^{N-1} |x[i]|^2} \tag{4.68}$$

$$||A||_2 = \sqrt{max_{i=0}^{N-1} \lambda_i(A^T A)} \tag{4.69}$$

Here, $\lambda_i(A^T A)$ refers the i^{th} eigenvalue of $A^T A$.

In this work, $Fl(X_N)$ used single precision floating point operators. The exact computation of the discrete transform, X_N, was defined as the computation of the Direct discrete transform treatment with double precision floating point operators.

The experiments were defined by two controlled factors:

- **Size** - Number of points of the discrete transform, N. Six levels were used, N=2, 4, 8, 16, 32, 64.
- **Treatment** - Refers to the discrete transform treatment used for the computation of $Fl(X_N)$. Three levels were used: the direct discrete transform treatment (DDT) and two fast discrete transform treatments (FDT 1 and FDT 2).

A total of 18 experiments were evaluated for each discrete transform. Since the main objective of this study was to observe the effect of the treatment on the accuracy, the experiments were grouped by the size factor. Therefore, a total of six experiment groups were evaluated for each discrete transform.

An total of 1,000 replicas of the experiment groups were performed for each discrete transform. The replicas were defined by the values of the input vector elements, $x[n]$. These had the following characteristics:

- Pseudorandom values with a standard uniform distribution
- Values in the range [0,1]
- Values with single precision representation

Figure 4.12 shows the defined experimental flow for a experiment group.

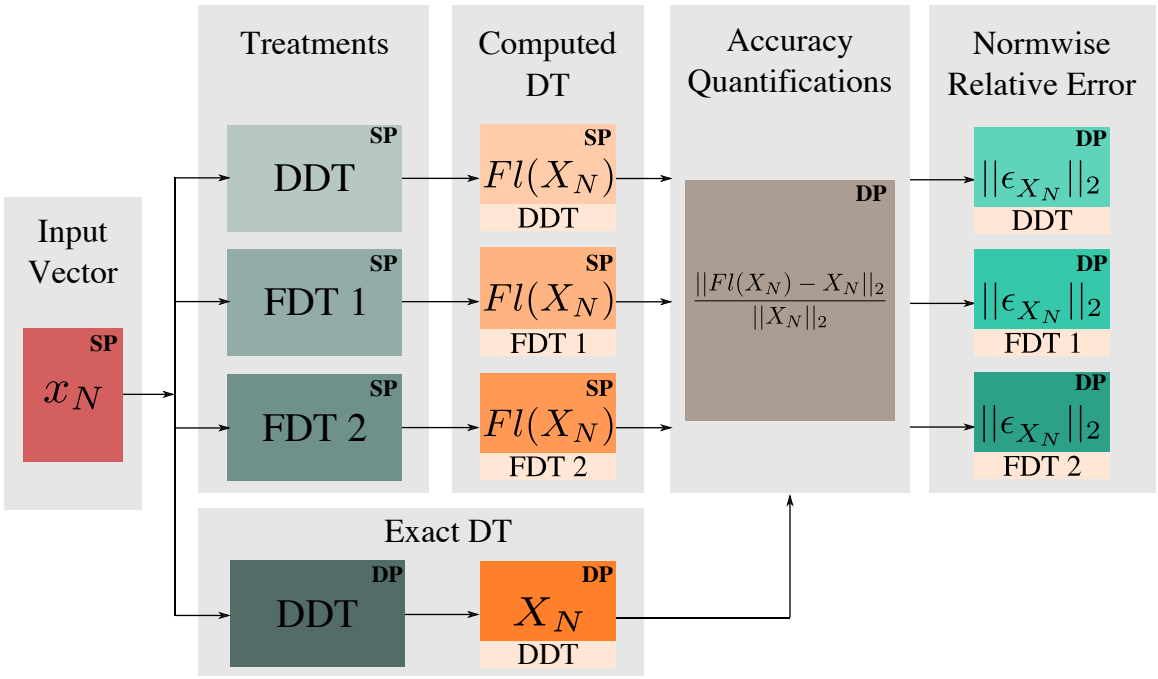


Figure 4.12 : Experimental Flow

4.4 Error Analysis

One of the main objectives of this study was to empirically estimate the accuracy of discrete transforms treatments. The forward error analysis, discussed in Chapter

2, was the methodology followed for the empirical error estimations. The floating point computation of a discrete transform is described as

$$Fl(X_N) = (I_N + RE_{X_N})DT_N x_N \tag{4.70}$$

where RE_{X_N} is the relative error diagonal matrix. The normwise relative error is defined as

$$||\epsilon_{X_N}||_2 = \frac{||Fl(X_N) - X_N||_2}{||X_N||_2} \leq ||RE_{X_N}||_2 \kappa(DT_N), \tag{4.71}$$

where

$$\kappa(DT_N) = ||DT_N||_2 ||DT_N^{-1}||_2 \tag{4.72}$$

The forward error analysis also provided insight about the architectural differences between the discrete transforms treatments. These differences will be taken in consideration for the recommendations.

For the validation of the error analysis, the experimental and estimations of the normwise relative error were compared.

4.5 Statistical Analysis Design

The purpose of the statistical analysis was to determine significant differences between the accuracy of the discrete transform treatments using statistical tools. This analysis consisted of two phases. The first phase was the distribution identification of the experimental data in each experiment. The second phase was to perform statistical tests to compare the accuracy between the treatments pairs in each level of the size factor. The statistical tests were dependent on the distribution of the experimental data.

Chapter 5
Results

This chapter presents accuracy and hardware performance results of the discrete transforms treatments. Section 5.1 presents the forward error analysis results. This analysis estimates the normwise relative error of the treatments. Section 5.2 provides the statistical analysis results. Section 5.3 presents the hardware performance of the treatments, in terms of resource consumption and latency. Section 5.4 discusses the accuracy and hardware performance of the treatments.

5.1 Error Analysis

For the forward error analysis of the discrete transforms treatments the next assumptions were made.

- The length of signal $x[n]$ is $N = 2^p$, $p = 1, 2, ..., 6$
- The points of signal $x[n]$ are error free
- The matrix coefficients are error free.
- The machine relative error, ϵ_{mach}, refers to the maximum relative error of a single precision arithmetic operation. This means $\epsilon_{mach} = 2^{-23}$

The main properties of the basic floating point computations used in the forward error analysis were:

- The floating point addition of operands a and b, with relative errors ϵ_a and ϵ_b, where $\{a, b \in \mathbb{R}\}$ or $\{a, b \in \mathbb{C}\}$, is

$$Fl(a+b) \leq (a+b)\left(1 + \epsilon_{mach} + \frac{(a\epsilon_a + b\epsilon_b)(1+\epsilon_{mach})}{a+b}\right) \tag{5.1}$$

- The floating point multiplication of operands a and b, with relative errors ϵ_a and ϵ_b, where $\{a, b \in \mathbb{R}\}\}$, is

$$Fl(a \cdot b) \leq (a \cdot b)(1 + \epsilon_a)(1 + \epsilon_b)(1 + \epsilon_{mach}) \tag{5.2}$$

- The floating point multiplication of operands a and b, with relative errors ϵ_a and ϵ_b, where $\{a, b \in \mathbb{C}\}\}$, is

$$Fl(a \cdot b) \leq (a \cdot b)(1 + \epsilon_a)(1 + \epsilon_b)(1 + \epsilon_{mach})^2 \tag{5.3}$$

The approach in the forward error analysis was to represent the computations of $Fl(X_N)$ as the computation of its matrix representation. As mentioned before, the matrix representation of X_N is in the form

$$X_N = DT_N x_N, \tag{5.4}$$

where DT_N is the discrete transform matrix. Therefore, the relative error estimations are proportional to the amount of matrices used to represent DT_N. The main properties of the floating point matrix computations used in the forward error analysis were:

- The floating point computation of the expression $y = Ax$, where $\{A, \in \mathbb{R}^{N \times N}\}$, $\{x \in \mathbb{R}^N\}$, is described as

$$Fl(y) = (I_N + RE_y)Ax, \tag{5.5}$$

where

$$[Fl(y)]_i \leq \left(\sum_{m=0}^{N-1} Fl([A]_{i,m}) Fl([x]_m) \right) (1 + \epsilon_{mach})^N \tag{5.6}$$

- The floating point computation of the expression $y = Ax$, where $\{A, \in \mathbb{C}^{N \times N}\}$, $\{x \in \mathbb{C}^N\}$, is described as

$$Fl(y) = (I_N + RE_y)Ax, \tag{5.7}$$

where

$$[Fl(y)]_i \leq \left(\sum_{m=0}^{N-1} Fl([A]_{i,m}) Fl([x]_m) \right) (1 + \epsilon_{mach})^{N+1} \tag{5.8}$$

- The floating point computation of the expression $y = A \cdot x$, where A is a permutation matrix does not generate relative error.

5.1.1 Discrete Fourier Transform (DFT)

The computation of an N-point Direct DFT, as described in Equation 4.14, consists of one matrix to describe the DFT_N matrix. The floating point computation of X_N is described as

$$Fl(X_N) = (I_N + RE_{X_N}) DFT_N x_N \tag{5.9}$$

Since the operands are complex, the upper bound of $||RE_{X_N}||_2$ in the direct computation of an N-point DFT is

$$||RE_{X_N}||_2 \leq (1 + \epsilon_{mach})^{N+1} - 1 \tag{5.10}$$

Next are the forward error analyses of the Cooley-Tukey and Pease Fast Fourier Transforms (FFTs) treatments.

Cooley-Tukey FFT

The computation of an N-point Cooley-Tukey FFT, as described Equations 4.20, consists of four matrices to describe DFT_N matrix. The floating point computation of X_N is described as

$$Fl(X_N) = (I_N + RE_{X_N})(DFT_2 \otimes I_{\frac{N}{2}}) T^N_{\frac{N}{2}} (I_2 \otimes DFT_{\frac{N}{2}}) L^N_2 x_N \tag{5.11}$$

where $I_{\frac{N}{2}}$ is an identity matrix, $T^N_{\frac{N}{2}}$ is a diagonal matrix, and L^N_2 is a permutation matrix; all previously defined in Chapter 4. Therefore, the upper bound of $||RE_{X_N}||_2$

in the computation of an N-point Cooley-Tukey FFT is

$$||RE_{X_N}||_2 \leq (1+\epsilon_{mach})^{3(N+1)} - 1 \tag{5.12}$$

Pease FFT

The computation of an N-point Pease FFT, as described in Equation 4.22, consists of $3p+1$ matrices to describe the DFT_N matrix. The floating point computation of X_N is described as

$$Fl(X_{2^p}) = ||RE_{X_N}||_2 \left\{ \prod_{i=1}^{p} L_2^{2^p} (I_{2^{p-1}} \otimes DFT_2) T_i' \right\} R_{2^p} x_{2^p}, \tag{5.13}$$

where $I_{2^{p-1}}$ is an identity matrix, T_i' is a diagonal matrix, and $L_2^{2^p}$ and R_{2^p} are permutation matrices; all previously defined in Chapter 4. Therefore, the upper bound of $||RE_{X_N}||_2$ on the computation of an N-point Pease FFT is

$$||RE_{X_N}||_2 \leq (1+\epsilon_{mach})^{2p(N+1)} - 1 \tag{5.14}$$

Normwised Relative Error Estimations

Table 5.1 provides the upper bounds of $||\epsilon_{X_N}||_2$, as defined in Equation 4.71, of the N-point DFT treatment. These values show that the Direct DFT had the lowest $||\epsilon_{X_N}||_2$ upper bound as N increases.

Table 5.1 : Upper bound of $||\epsilon_{X_N}||_2$ of the N-point DFT treatments.

N	$\kappa_2(DFT_N)$	$\|\|\epsilon_{X_N}\|\|_2 \leq \|\|RE_{X_N}\|\|_2 \kappa_2(DFT_N)$		
		Direct DFT	Cooley-Tukey FFT	Pease FFT
2	$\leq 1+2^{-51}$	$< 2^{-21}$	$< 2^{-19}$	$< 2^{-20}$
4	$\leq 1+2^{-52}$	$< 2^{-20}$	$< 2^{-19}$	$< 2^{-18}$
8	$< 1+2^{-25}$	$< 2^{-19}$	$< 2^{-18}$	$< 2^{-17}$
16	$< 1+2^{-25}$	$< 2^{-18}$	$< 2^{-17}$	$< 2^{-15}$
32	$< 1+2^{-24}$	$< 2^{-17}$	$< 2^{-16}$	$< 2^{-14}$
64	$< 1+2^{-24}$	$< 2^{-16}$	$< 2^{-15}$	$< 2^{-13}$

Error Analysis Validation

Table 5.2 shows the ranges of the experimental values of $||\epsilon_{X_N}||_2$ of the DFT treatments. As observed, the ranges are lower than the $||\epsilon_{X_N}||_2$ upper bounds estimations of these treatments (Table 5.1). The ranges of the Direct DFT are lower than those of the FFTs treatments as N increases. This behavior was also observed in the forward error analysis, proving a relationship between the estimations with the magnitude of the experimental data ranges. This validates the forward error analysis results with the experimental data of the DFT treatments.

Table 5.2 : Ranges of experimental $||\epsilon_{X_N}||_2$ of the DFT structures

N-point	Direct DFT	Cooley-Tukey FFT	Pease FFT
2	$2^{-52} < ||\epsilon_{X_N}||_2 < 2^{-24}$	$2^{-28} < ||\epsilon_{X_N}||_2 < 2^{-23}$	$2^{-28} < ||\epsilon_{X_N}||_2 < 2^{-23}$
4	$2^{-28} < ||\epsilon_{X_N}||_2 < 2^{-23}$	$2^{-26} < ||\epsilon_{X_N}||_2 < 2^{-23}$	$2^{-26} < ||\epsilon_{X_N}||_2 < 2^{-23}$
8	$2^{-26} < ||\epsilon_{X_N}||_2 < 2^{-22}$	$2^{-23} < ||\epsilon_{X_N}||_2 < 2^{-20}$	$2^{-23} < ||\epsilon_{X_N}||_2 < 2^{-20}$
16	$2^{-26} < ||\epsilon_{X_N}||_2 < 2^{-22}$	$2^{-22} < ||\epsilon_{X_N}||_2 < 2^{-19}$	$2^{-22} < ||\epsilon_{X_N}||_2 < 2^{-19}$
32	$2^{-25} < ||\epsilon_{X_N}||_2 < 2^{-22}$	$2^{-20} < ||\epsilon_{X_N}||_2 < 2^{-18}$	$2^{-20} < ||\epsilon_{X_N}||_2 < 2^{-18}$
64	$2^{-24} < ||\epsilon_{X_N}||_2 < 2^{-21}$	$2^{-19} < ||\epsilon_{X_N}||_2 < 2^{-17}$	$2^{-19} < ||\epsilon_{X_N}||_2 < 2^{-17}$

5.1.2 Discrete Hartley Transform (DHT)

The computation of an N-point Direct DHT, as described in Equation 4.27, consists of one matrix to describe theDHT_N matrix. The floating point computation of X_N is described as

$$Fl(X_N) = (I_N + RE_{X_N})DHT_N x_N \tag{5.15}$$

Since the operands are real, the upper bound of $||RE_{X_N}||_2$ on the direct computation of an N-point DHT is

$$||RE_{X_N}||_2 \leq (1 + \epsilon_{mach})^N - 1 \tag{5.16}$$

Next are the forward error analysis of the Bracewell and Hou Fast Hartley Transforms (FHTs) treatments.

Bracewell FHT

The computation of an N-point Bracewell FHT, as described in Equation 4.32, consists of three matrices to describe the DHT_N matrix. The floating point computation of X_N is described as

$$Fl(X_N) = (I_N + RE_{X_N})(DFT_2 \otimes I_{\frac{N}{2}})(DHT_{\frac{N}{2}} \oplus \widehat{DHT}_{\frac{N}{2}})L_2^N x_N, \tag{5.17}$$

where $I_{\frac{N}{2}}$ is an identity matrix, and L_2^N is a permutation matrix; all previously defined in Chapter 4. Therefore, the upper bound of $||RE_{X_N}||_2$ on the computation of an N-point Bracewell FHT is

$$||RE_{X_N}||_2 \leq (1 + \epsilon_{mach})^{2N} - 1 \tag{5.18}$$

Hou FHT

The computation of an N-point Hou FHT, as described in Equation 4.37, consists of three matrices to describe the DHT_N matrix. The floating point computation of X_N is described as

$$Fl(X_N) = (I_N + RE_{X_N})L_{N/2}^N(DHT_{N/2} \oplus \widehat{DHT}_{N/2}^T)(DFT_2 \otimes I_{N/2})x_N, \tag{5.19}$$

where $L_{N/2}^N$ is a permutation matrix. Therefore, the upper bound of $||RE_{X_N}||_2$ on the computation of an N-point Hou FHT is

$$||RE_{X_N}||_2 \leq (1 + \epsilon_{mach})^{2N} - 1 \tag{5.20}$$

Normwised Relative Error Estimations

Table 5.3 provides the upper bound of $||\epsilon_{X_N}||_2$, as defined in Equation 4.71, of the N-point DHT treatments. These values show that the Direct DHT had the lowest $||\epsilon_{X_N}||_2$ upper bound as N increases. Also, that the Bracewell and Hou FHTs had the same $||\epsilon_{X_N}||_2$ upper bounds.

Table 5.3 : Upper bound of $||\epsilon_{X_N}||_2$ of the N-point DHT treatments.

N	$\kappa_2(DHT_N)$	$\|\|\epsilon_{X_N}\|\|_2 \leq \|\|RE_{X_N}\|\|_2 \kappa_2(DHT_N)$		
		Direct DHT	Bracewell FHT	Hou FHT
2	$< 1 + 2^{-23}$	$\leq 2^{-22}$	$\leq 2^{-21}$	$\leq 2^{-21}$
4	$< 1 + 2^{-22}$	$\leq 2^{-21}$	$\leq 2^{-20}$	$\leq 2^{-20}$
8	$< 1 + 2^{-19}$	$\leq 2^{-20}$	$\leq 2^{-19}$	$\leq 2^{-19}$
16	$< 1 + 2^{-18}$	$\leq 2^{-19}$	$\leq 2^{-18}$	$\leq 2^{-18}$
32	$< 1 + 2^{-17}$	$\leq 2^{-18}$	$\leq 2^{-17}$	$\leq 2^{-17}$
64	$< 1 + 2^{-15}$	$\leq 2^{-17}$	$\leq 2^{-16}$	$\leq 2^{-16}$

Error Analysis Validation

Table 5.4 shows the ranges of the experimental values of $||\epsilon_{X_N}||_2$ of the DHT treatments. As observed, the ranges are lower than those of the $||\epsilon_{X_N}||_2$ upper bounds estimations of these treatments (Table 5.3). It is also observed that the ranges of the Direct DHT were lower than those of the FHTs treatments as N increases. This behavior was also observed in the forward error analysis, proving a relationship between the estimations with the magnitude of the experimental data ranges. Also, was observed that the ranges of the experimental $||\epsilon_{X_N}||_2$ of the Bracewell and Hou FHTs were similar. This validates the forward error analysis results with the experimental data of the DHT treatments.

Table 5.4 : Ranges of experimental $||\epsilon_{X_N}||_2$ of the DHT structures

N-point	Direct DHT	Bracewell FHT	Hou FHT
2	$2^{-34} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-23}$	$2^{-28} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-22}$	$2^{-28} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-22}$
4	$2^{-28} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-22}$	$2^{-27} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-23}$	$2^{-27} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-22}$
8	$2^{-26} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-22}$	$2^{-23} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-20}$	$2^{-23} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-20}$
16	$2^{-26} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-22}$	$2^{-22} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-18}$	$2^{-22} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-19}$
32	$2^{-25} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-21}$	$2^{-20} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-18}$	$2^{-20} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-18}$
64	$2^{-24} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-21}$	$2^{-19} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-17}$	$2^{-19} < \|\|\epsilon_{X_N}\|\|_2 < 2^{-17}$

5.1.3 Discrete Cosine Transform (DCT)

The computation of an N-point Direct DCT, as described in Equation 4.42, consists of two matrices to describe the DCT_N matrix. The floating point computation

of X_N is described as

$$Fl(X_N) = (I_N + RE_{X_N})SF_N\overline{DCT}^{II}_N x_N \tag{5.21}$$

Since the operands are real, the upper bound of $||RE_{X_N}||_2$ on the direct computation of an N-point DCT is

$$||RE_{X_N}||_2 \leq (1 + \epsilon_{mach})^{2N} - 1 \tag{5.22}$$

Next are the forward error analysis of the Nikara and Translation Fast Cosine Transforms (FCTs) treatments.

Nikara FCT

The computation of an N-point Nikara FCT, as described in Equations 4.45, consists of $5p+2$ matrices to describe the DCT_N matrix. The floating point computation of X_N is described as

$$Fl(X_N) = (I_N + RE_{X_N})\sqrt{\frac{2}{2^p}}U^{(p-1)}_{2^p}\left\{\prod_{s=p-1}^{1} A^{(s)}_{2^p} S^{(s)}_{2^p}\right\} A^{(0)}_{2^p} P^H_{2^p} x_{2^p}, \tag{5.23}$$

where matrix $A^{(s)}_{2^p}$ (as defined in Equation 4.50) consist of four matrices, including one permutation matrix. The terms $U^{(p-1)}_{2^p}$, $S^{(s)}_{2^p}$, and $P^H_{2^p}$ are permutation matrices. Therefore, the upper bound of $||RE_{X_N}||_2$ on the computation of an N-point Nikara FCT is

$$||RE_{X_N}||_2 \leq (1 + \epsilon_{mach})^{(3p+1)N} - 1 \tag{5.24}$$

Translation FCT

The computation of an N-point Translation FCT, as described in Equations 4.64, consists of five matrices to describe the DFT_N matrix. The floating point computation of X_N is described as

$$Fl(X_N) = (I_N + RE_{X_N})SF_N L^N_{N/2}(\overline{DCT}^{II}_{N/2} \oplus \overline{DCT}^{IV}_{N/2})(DFT_2 \otimes I_{N/2})(I_{N/2} \oplus \bar{I}_{N/2})x_N \tag{5.25}$$

where $L^N_{N/2}$ and $(I_{N/2} \oplus \bar{I}_{N/2})$ are a permutation matrices. Therefore, the upper bound of $||RE_{X_N}||_2$ on the computation of an N-point Translation FCT is

$$||RE_{X_N}||_2 \leq (1 + \epsilon_{mach})^{3N} - 1 \tag{5.26}$$

Normwised Relative Error Estimations

Table 5.5 provides the upper bound of $||\epsilon_{X_N}||_2$, as defined in Equation 4.71, of the N-point DCT treatments. These values show that the Direct DCT had the lowest $|\epsilon_{X_N}|$ upper bound as N increases.

Table 5.5 : Upper bound of $||\epsilon_{X_N}||_2$ of the N-point DCT treatments.

N	$\kappa_2(DCT_N)$	$\|\|\epsilon_{X_N}\|\|_2 = \|\|RE_{X_N}\|\|_2 \kappa_2(DCT_N)$		
		Direct DCT	Nikara FCT	Translation FCT
2	$\leq 2 + 2^{-23}$	$\leq 2^{-20}$	$\leq 2^{-19}$	$< 2^{-19}$
4	$< 2 - 2^{-22}$	$\leq 2^{-19}$	$< 2^{-17}$	$< 2^{-18}$
8	$< 2 - 2^{-26}$	$\leq 2^{-18}$	$< 2^{-15}$	$< 2^{-17}$
16	$< 2 - 2^{-24}$	$\leq 2^{-17}$	$< 2^{-14}$	$< 2^{-16}$
32	$< 2 + 2^{-26}$	$\leq 2^{-16}$	$\leq 2^{-13}$	$< 2^{-15}$
64	$< 2 - 2^{-24}$	$\leq 2^{-15}$	$< 2^{-11}$	$< 2^{-14}$

Error Analysis Validation

Table 5.6 shows the ranges of the experimental $||\epsilon_{X_N}||_2$ of the DCT treatments. As observed, the ranges are lower than the $||\epsilon_{X_N}||_2$ upper bounds estimation for these treatments (Table 5.5). The ranges of the Direct DCT were lower than those of the FCTs treatments as N increases. This behavior was also observed in the forward error analysis, proving a relationship between the estimations with the magnitude of the experimental data ranges. Also, was observed that the ranges of the experimental $||\epsilon_{X_N}||_2$ of the Nikara and Translation FCTs were similar. This validates the forward error analysis results with the experimental data of the DCT treatments.

Table 5.6 : Ranges of experimental $||\epsilon_{X_N}||_2$ of the DCT structures

N-point	Direct DCT	Nikara FCT	Translation FCT
2	$2^{-32} < ||\epsilon_{X_N}||_2 < 2^{-23}$	$2^{-30} < ||\epsilon_{X_N}||_2 < 2^{-23}$	$2^{-30} < ||\epsilon_{X_N}||_2 < 2^{-23}$
4	$2^{-28} < ||\epsilon_{X_N}||_2 < 2^{-23}$	$2^{-27} < ||\epsilon_{X_N}||_2 < 2^{-21}$	$2^{-26} < ||\epsilon_{X_N}||_2 < 2^{-21}$
8	$2^{-27} < ||\epsilon_{X_N}||_2 < 2^{-22}$	$2^{-25} < ||\epsilon_{X_N}||_2 < 2^{-21}$	$2^{-24} < ||\epsilon_{X_N}||_2 < 2^{-21}$
16	$2^{-25} < ||\epsilon_{X_N}||_2 < 2^{-22}$	$2^{-23} < ||\epsilon_{X_N}||_2 < 2^{-20}$	$2^{-23} < ||\epsilon_{X_N}||_2 < 2^{-20}$
32	$2^{-25} < ||\epsilon_{X_N}||_2 < 2^{-21}$	$2^{-22} < ||\epsilon_{X_N}||_2 < 2^{-19}$	$2^{-22} < ||\epsilon_{X_N}||_2 < 2^{-19}$
64	$2^{-24} < ||\epsilon_{X_N}||_2 < 2^{-21}$	$2^{-20} < ||\epsilon_{X_N}||_2 < 2^{-18}$	$2^{-20} < ||\epsilon_{X_N}||_2 < 2^{-18}$

5.2 Statistical Analysis

The purpose of the statistical analysis was to determine if the experimental normwise relative error (experimental data) differences between the discrete transform treatments were statistically significant.

First an Anderson-Darling (AD) test was performed to the treatments experimental data [27]. The AD test determines if the treatments experimental data followed a normal distribution at a significance level of 0.05. The AD test null and alternative hypotheses were:

H_0: The data from treatment y is from a normal distribution.

H_1: The data from treatment y is not from a normal distribution.

The AD test statistic was defined as

$$A_y^2 = -n_y - \sum_{i=1}^{n_y} \frac{2i-1}{n_y} \left[ln(F(y_i)) + ln(1 - F(y_{n_y+1-i})) \right] \tag{5.27}$$

where n_y refers to the sample size of the treatment y, y_i refers to the sample i of the ordered sample set of y, and $F(y_i)$ is defined as

$$F(y_i) = P(y_i) \tag{5.28}$$

The null hypothesis is rejected if the AD test statistic, A_y^2, is higher than the critical value of the theoretical normal distribution at a 0.05 significance level.

From the 54 all the treatments experiments, 16% of experimental data distributions rejected the null hypothesis of the AD test. This means that the continuous

distribution of the treatments experimental data did not follow a normal distribution. Since transforming the treatments experimental data to fit a normal distribution could cause misleading statistical interpretations, a non-parametric approach was used to compare the pairs of treatments experimental data.

The non-parametric test used was a two-sample Kolmogorov-Smirnov (KS) test [28]. This statistical test determines if the continuos distributions of two treatments experimental data are equals, at a significance level of 0.05. The two-sample KS test null and alternative hypotheses were:

H_0: The data from treatments v and w comes from the same continuous distribution.

H_1: The data from treatments v and w are from different continuous distributions.

The KS test statistic uses the cumulative distribution function (CDF) of the treatments experimental data to compare their continuos distributions. The CDF, $F_X(x)$, is defined as the probability that a random variable in the sample set X takes a value less than or equal to x. Therefore,

$$F_X(x) = P(X \leq x) \tag{5.29}$$

The KS test statistic is defined as

$$G^*_{v,w} = max_x(|F_v(x) - F_w(x)|) \tag{5.30}$$

The null hypothesis is rejected at a significance level of 0.05 if

$$G^*_{v,w} > 1.36\sqrt{\frac{n_v + n_w}{n_v \times n_w}} \tag{5.31}$$

where n_v and n_w refers to the sample size of the treatments v and w.

By plotting the CDF of the treatments, various statistical properties of the experimental data can be observed. These properties include range size, range magnitudes, minimum, maximum, and median. An examination of the treatments CDF plots was

performed for a better understanding of the KS test results, and to observe the behavior of the treatments experimental data as the number of point increases. Next are the statistical tests results and observations.

5.2.1 Discrete Fourier Transform

Table 5.7 shows the AD test results of the experimental normwise relative error of the N-point DFT treatments. Only four out of the 18 DFT experiments failed to reject the null hypothesis. Next are the detailed results of the two samples KS test of the N-point DFT treatments.

Table 5.7 : AD test results of the experimental data of the DFT structures

N-point	Direct DFT	Cooley-Tukey FFT	Pease FFT
2	H_0 rejected	H_0 rejected	H_0 rejected
4	H_0 rejected	H_0 rejected	H_0 rejected
8	H_0 rejected	H_0 rejected	H_0 rejected
16	H_0 rejected	H_0 not rejected	H_0 not rejected
32	H_0 rejected	H_0 rejected	H_0 rejected
64	H_0 rejected	H_0 not rejected	H_0 not rejected

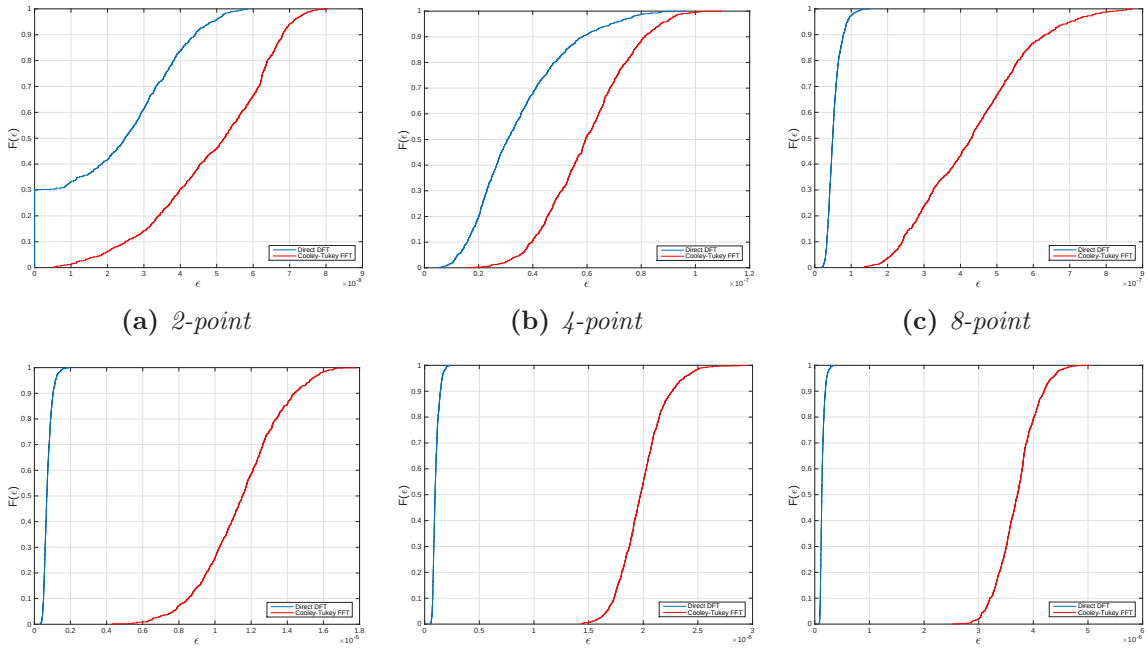
Direct DFT vs. Cooley-Tukey FFT

Table 5.8 shows the KS test results of the N-point Direct DFT and Cooley-Tukey FFT treatments experiments. These tests rejected the null hypothesis of the KS test, therefore the difference in accuracy between these treatments were statistically significant.

Table 5.8 : KS test results of the Direct DFT and the Cooley-Tukey FFT treatments

N-point	H_0	P-value	G^*
2	H_0 rejected	2.1759×10^{-130}	5.4400×10^{-1}
4	H_0 rejected	3.1267×10^{-151}	5.8600×10^{-1}
8	H_0 rejected	0	9.9900×10^{-1}
16	H_0 rejected	0	1.0000
32	H_0 rejected	0	1.0000
64	H_0 rejected	0	1.0000

Figure 5.1 shows the CDF plots of these treatments experimental data of the N-point DFT computation. In these plots the differences in range, and range magnitudes between the treatments experimental data are visible. These differences are proof that the Direct DFT and Cooley-Tukey FFT treatments provide different accuracy. In these CDF plots we observed that the range magnitude of the Direct DFT experimental data was lower than the Cooley-Tukey FFT. This means that the Direct DFT treatment provides higher accuracy than the Cooley-Tukey FFT.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*

(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.1 : CDF plots of the N-point Direct DFT and Cooley-Tukey FFT Experimental Data

Figure 5.2 shows the CDF plots of the experimental data of these treatments N-point DFT computation. It was observed that as the number of point of the DFT computation increases, the range magnitude of the treatments experimental data increases. But the difference observed between these treatments was that the range magnitude of the Cooley-Tukey FFT experimental data increased at a higher scale.

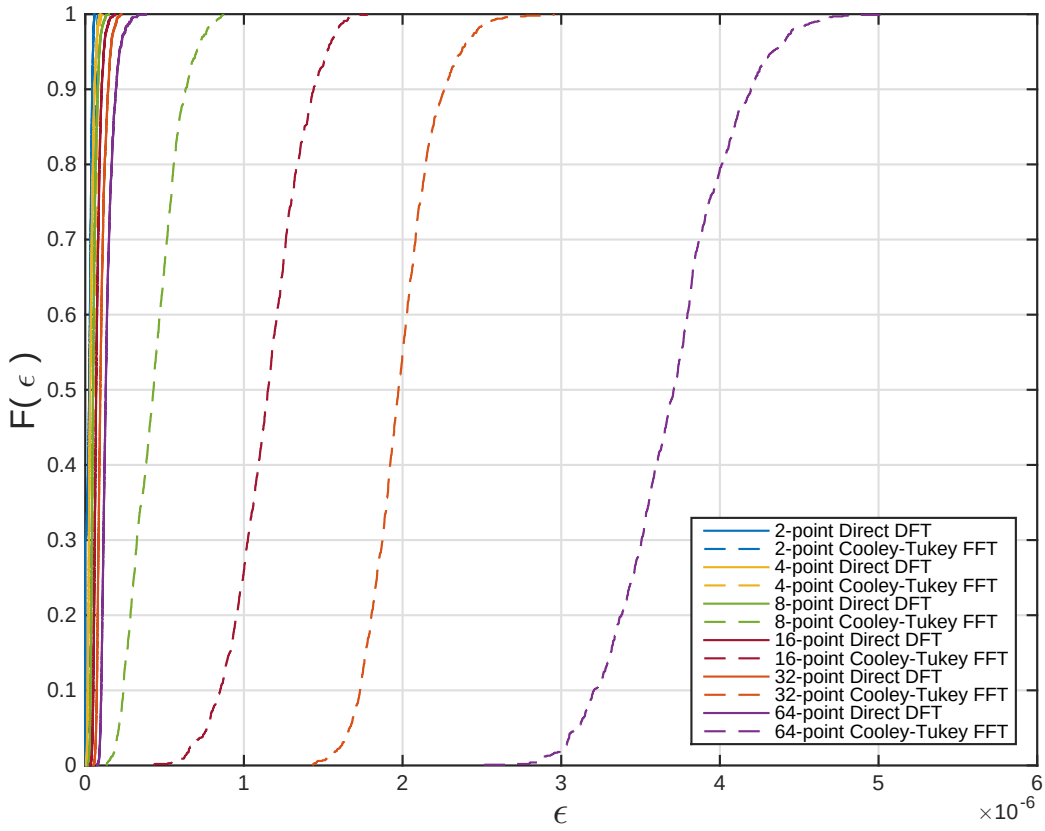


Figure 5.2 : CDF plots of the Direct DFT and Cooley-Tukey FFT Experimental Data

Direct DFT vs. Pease FFT

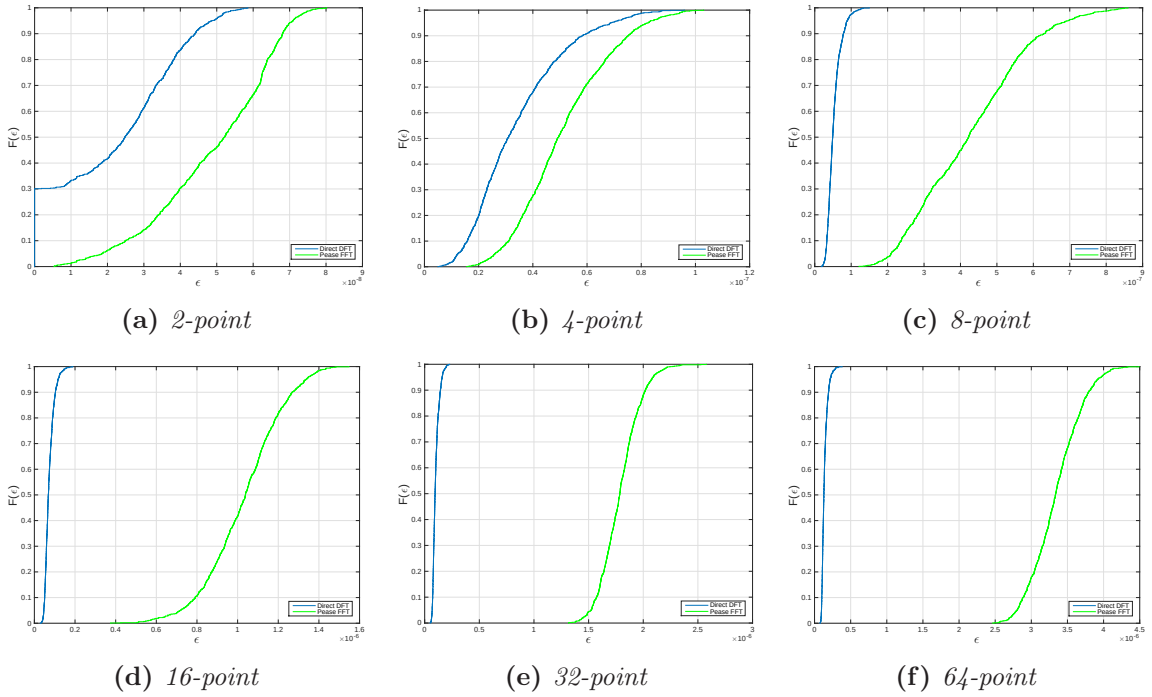
Table 5.9 shows the KS test results of the N-point Direct DFT and Pease FFT treatments experiments. These tests rejected the null hypothesis of the KS test, therefore the differences in accuracy between these treatments were statistically significant.

Figure 5.3 shows the CDF plots of these treatments experimental data of the N-point DFT computation. In these plots the differences in range, and range magnitudes between the treatments experimental data are visible. These differences are proof that the Direct DFT and Pease FFT treatments provide different accuracy. In the CDF plots we observed that the range magnitude of the Direct DFT experimental data

Table 5.9 : KS test results of the Direct DFT and the Pease FFT treatments

N-point	H_0	P-value	G^*
2	H_0 rejected	2.1759×10^{-130}	5.4400×10^{-1}
4	H_0 rejected	1.5907×10^{-77}	4.1900×10^{-1}
8	H_0 rejected	0	9.9700×10^{-1}
16	H_0 rejected	0	1.0000
32	H_0 rejected	0	1.0000
64	H_0 rejected	0	1.0000

was lower than the Pease FFT. This means that the Direct DFT treatment provides higher accuracy than the Pease FFT.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*

(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.3 : CDF plots of the N-point Direct DFT and Pease FFT Experimental Data

Figure 5.4 shows the CDF plots of the experimental data of these treatments N-point DFT computation. It was observed that as the number of point, N, of the DFT computation increases, the range magnitude of the treatments experimental data increases. But the difference observed between these treatments was that the range magnitude of the Pease FFT experimental data increased at a higher scale.

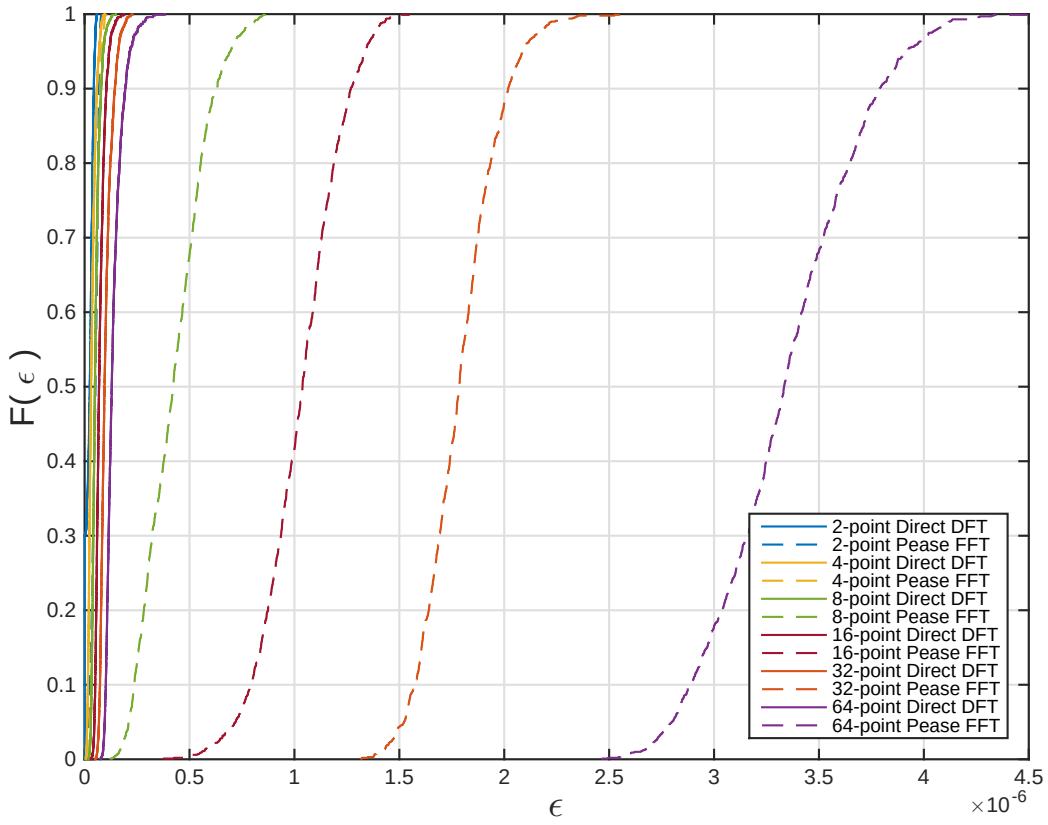


Figure 5.4 : CDF plots of the Direct DFT and Pease FFT Experimental Data

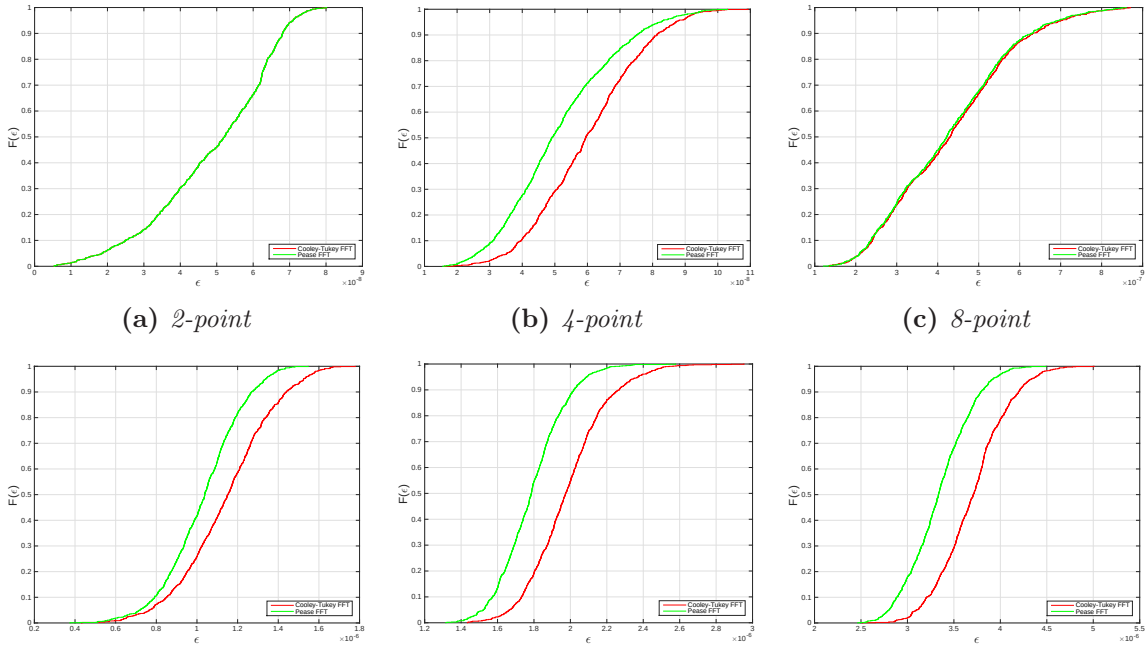
Cooley-Tukey FFT vs. Pease FFT

Table 5.10 shows the KS test results of the N-point Cooley-Tukey and Pease FFTs treatments experiments. Four out of the six tests rejected the null hypothesis of the KS test, therefore the difference in accuracy of between these treatments are statistically significant.

Table 5.10 : KS test results of the Cooley-Tukey FFT and the Pease FFT treatments

N-point	H_0	P-value	G^*
2	H_0 not rejected	1.0000	0
4	H_0 rejected	8.7181×10^{-27}	2.4500×10^{-1}
8	H_0 not rejected	8.2284×10^{-1}	2.8000×10^{-2}
16	H_0 rejected	5.3063×10^{-27}	2.4600×10^{-1}
32	H_0 rejected	1.5726×10^{-73}	4.0800×10^{-1}
64	H_0 rejected	6.0397×10^{-69}	3.9500×10^{-1}

Figure 5.5 shows the CDF plots of these treatments experimental data of the N-point DFT computation. The CDF plots of the treatments experimental data of the 2 and 8 -point DFT computations overlap; those were the experiments that failed to reject the null hypothesis of the KS test. The other experiments, which rejected the null hypothesis, we observed significant differences in the distributions of the treatments experimental data. In these plots the distributions of the Pease FFT treatment tented to lower values. That means that the Pease FFT treatment provide higher accuracy than the Cooley-Tukey FFT.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*

(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.5 : CDF plots of the N-point Cooley-Tukey FFT and Pease FFT Experimental Data

Figure 5.6 shows the CDF plots of the experimental data of these treatments N-point DFT computation. It was observed that as the number of point, N, of the DFT computation increases, the range magnitude of the treatments experimental data increases.

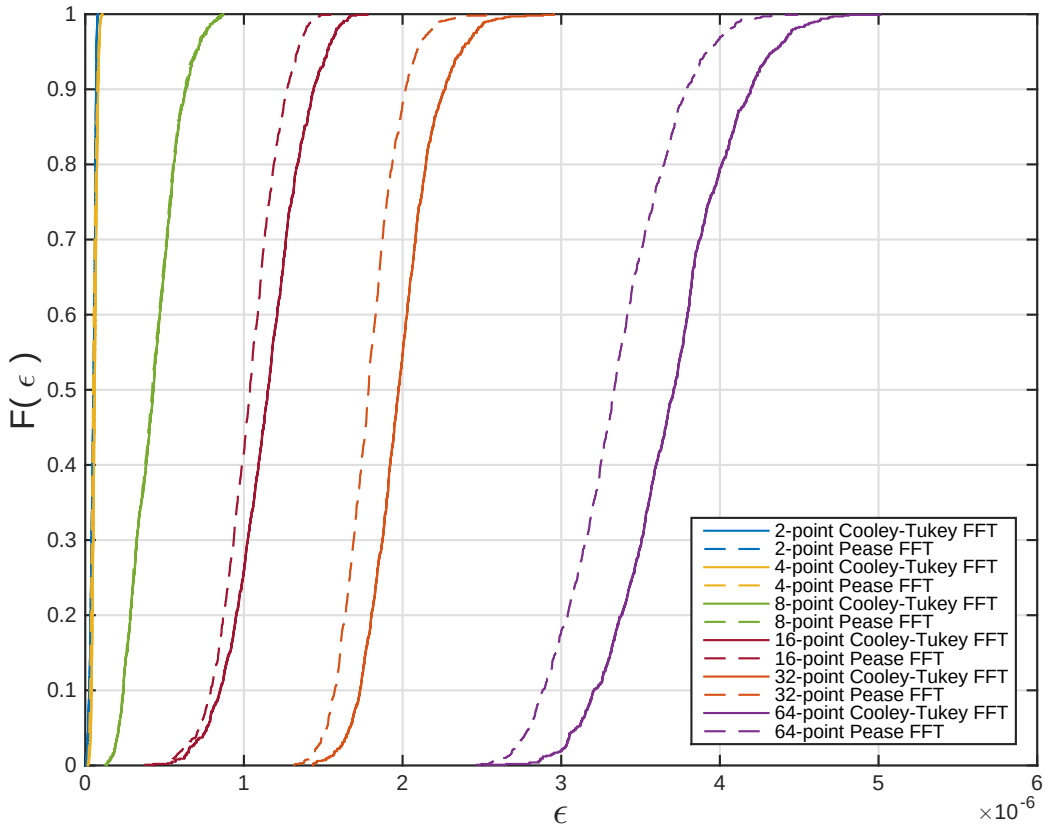


Figure 5.6 : CDF plots of the Cooley-Tukey and Pease FFTs Experimental Data

5.2.2 Discrete Hartley Transform

Table 5.11 shows the AD test results of the experimental normwise relative error of the N-point DHT treatments. Only three out of 18 DCT experiments failed to reject the null hypothesis. Next are the detailed results of the two samples KS test of the N-point DHT treatments.

Table 5.11 : AD test results of the experimental data of the DHT structures

N-point	Direct DHT	Bracewell FHT	Hou FHT
2	H_0 rejected	H_0 rejected	H_0 rejected
4	H_0 rejected	H_0 rejected	H_0 rejected
8	H_0 rejected	H_0 rejected	H_0 rejected
16	H_0 rejected	H_0 not rejected	H_0 rejected
32	H_0 rejected	H_0 rejected	H_0 not rejected
64	H_0 rejected	H_0 rejected	H_0 not rejected

Direct DHT vs. Bracewell FHT

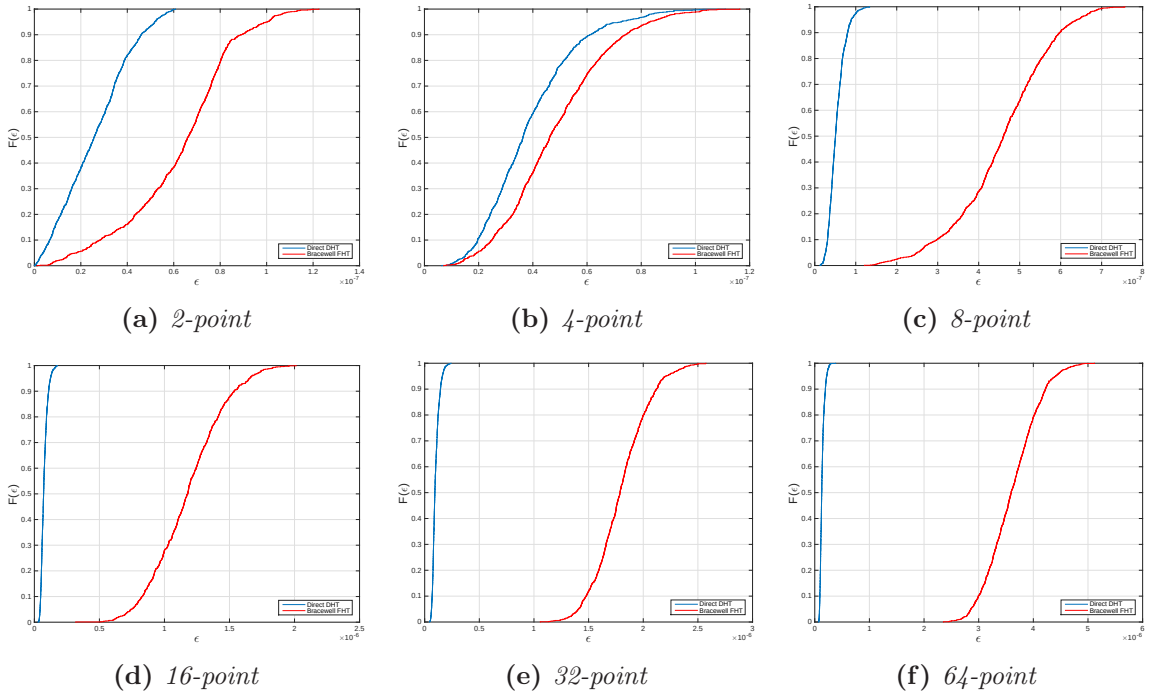
Table 5.12 shows the KS test results of the N-point Direct DHT and Bracewell FHT treatments experiments. These tests rejected the null hypothesis of the KS test, therefore the difference in accuracy between these treatments were statistically significant.

Table 5.12 : KS test results of the Direct DHT and the Bracewell FHT treatments

N-point	H_0	P-value	G^*
2	H_0 rejected	2.7384×10^{-205}	6.8300×10^{-1}
4	H_0 rejected	2.8804×10^{-24}	2.3300×10^{-1}
8	H_0 rejected	0	9.9900×10^{-1}
16	H_0 rejected	0	1.0000
32	H_0 rejected	0	1.0000
64	H_0 rejected	0	1.0000

Figure 5.7 shows the CDF plots of these treatments experimental data of the N-point DHT computation. In these plots we observed significant differences in distribution, range, and range magnitude between the treatments experimental data. This proof a difference in accuracy between these treatments, in which the Direct DHT provides the higher accuracy.

Figure 5.8 shows the CDF plots of the experimental data of these treatments N-point DHT computation. It was observed that as the number of point, N, of the DHT computation increases, the range magnitude of the treatments experimental data increases. But the difference observed between these treatments was that the range magnitude of the Bracewell FHT experimental data increased at a higher scale.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*

(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.7 : CDF plots of the N-point Direct DHT and Bracewell FHT Experimental Data

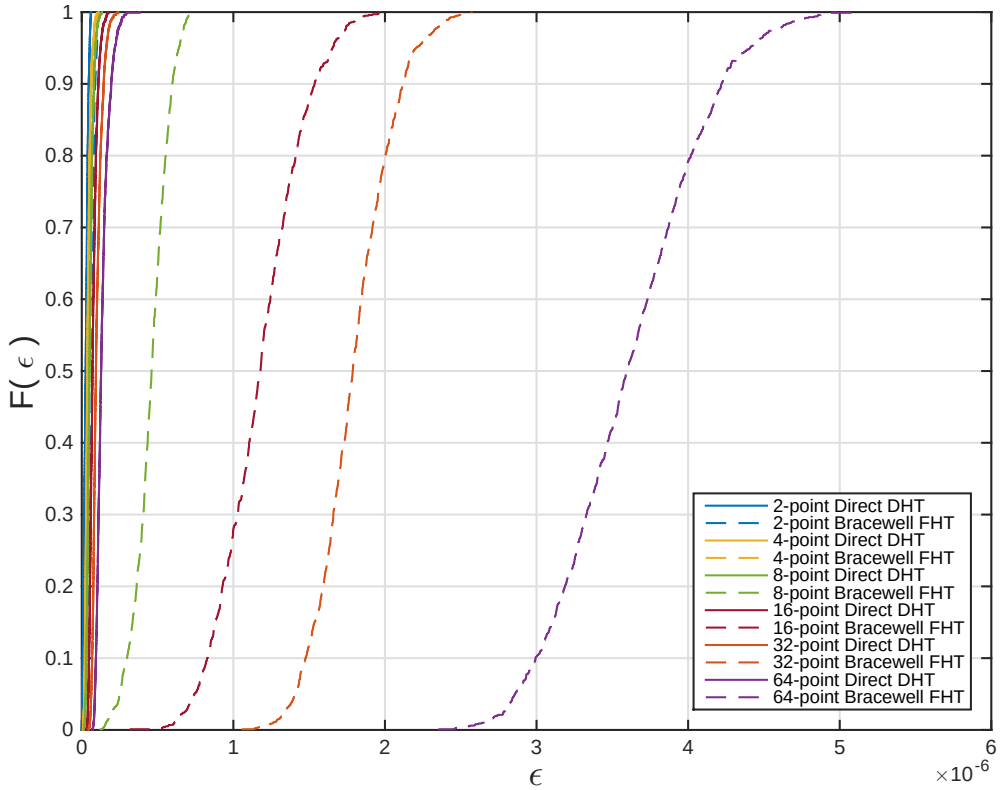


Figure 5.8 : CDF plots of the Direct DHT and Bracewell FHT Experimental Data

Direct DHT vs. Hou FHT

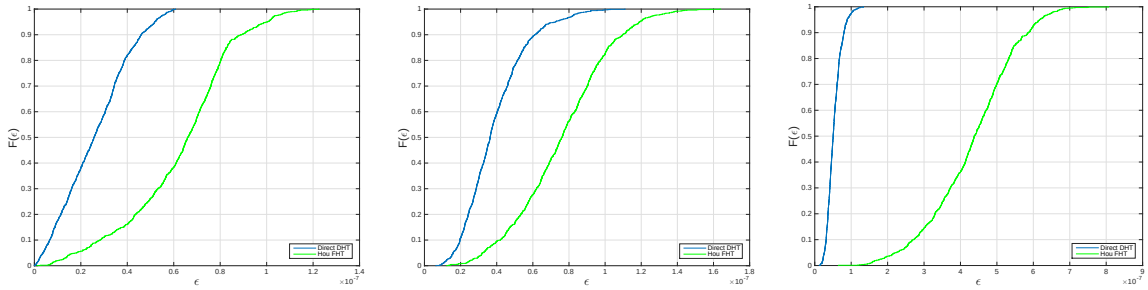
Table 5.13 shows the KS test results of the N-point Direct DHT and Hou FHT treatments experiments. These tests rejected the null hypothesis of the KS test, therefore the difference in accuracy between these treatments were statistically significant.

Table 5.13 : KS test results of the Direct DHT and the Hou FHT treatments

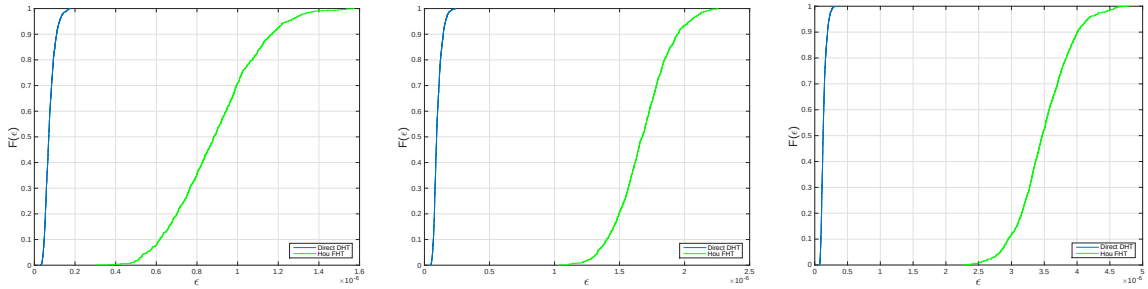
N-point	H_0	P-value	G^*
2	H_0 rejected	2.7384×10^{-205}	6.8300×10^{-1}
4	H_0 rejected	1.2686×10^{-178}	6.3700×10^{-1}
8	H_0 rejected	0	9.9600×10^{-1}
16	H_0 rejected	0	1.0000
32	H_0 rejected	0	1.0000
64	H_0 rejected	0	1.0000

Figure 5.9 shows the CDF plots of these treatments experimental data of the N-point DHT computation. We observed in these plots significant differences in distribution, range, and range magnitudes between the treatments experimental data. These are proof that the treatments provide different accuracy. In these plots the range magnitude of the Direct DHT experimental data were lower than the Bracewell FHT, therefore Direct DHT treatment provides higher accuracy.

Figure 5.10 shows the CDF plots of the experimental data of these treatments N-point DHT computation. It was observed that as the number of point, N, of the DHT computation increases, the range magnitude of the treatments experimental data increases. But the difference observed between these treatments was that the range magnitude of the Hou FHT experimental data increased at a higher scale.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*



(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.9 : CDF plots of the N-point Direct DHT and Hou FHT Experimental Data

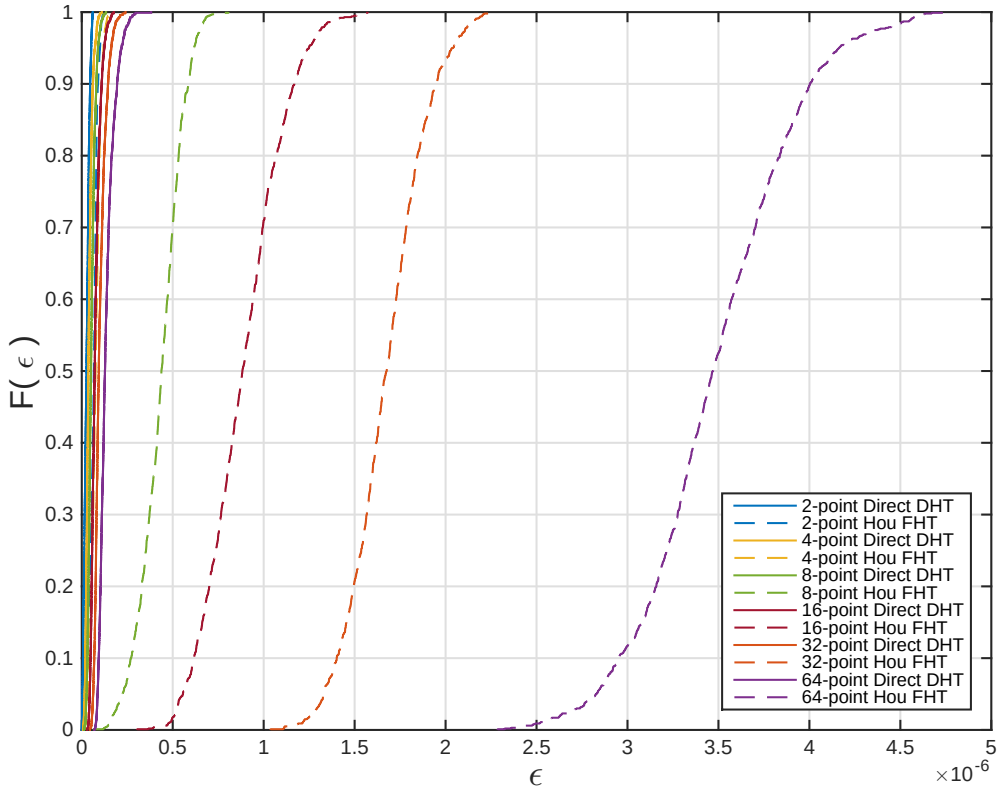


Figure 5.10 : CDF plots of the Direct DHT and Hou FHT Experimental Data

Bracewell FHT vs. Hou FHT

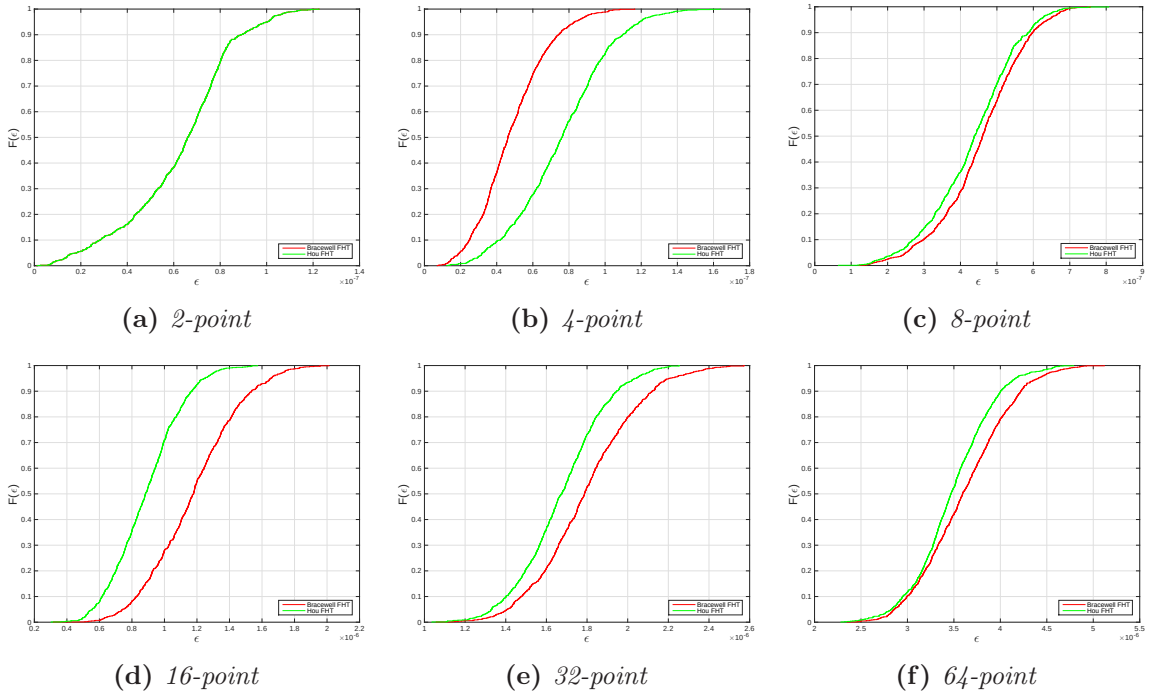
Table 5.14 shows the KS test results of the N-point Bracewell and Hou FHTs treatments experiments. Five out of the six tests rejected the null hypothesis of the KS test, therefore the difference in accuracy of between these treatments were statistically significant.

Table 5.14 : KS test results of the Bracewell FHT and the Hou FHT treatments

N-point	H_0	P-value	G^*
2	H_0 not rejected	1.0000	0
4	H_0 rejected	4.9708×10^{-102}	4.8100×10^{-1}
8	H_0 rejected	6.6442×10^{-4}	8.9000×10^{-2}
16	H_0 rejected	3.6717×10^{-94}	4.6200×10^{-1}
32	H_0 rejected	1.5461×10^{-20}	2.1400×10^{-1}
64	H_0 rejected	3.4097×10^{-8}	1.3300×10^{-1}

Figure 5.11 shows the CDF plots of these treatments experimental data of the N-point DHT computation. The CDF plots of the treatments experimental data of the 2-point DHT computation overlap, those were the experiments that failed to reject the null hypothesis of the KS test. In the other plots, where the null hypothesis was rejected, we observed differences in their distributions and ranges of the treatments experimental data. Also, that most of the distributions of the Hou FHT treatment tended to lower values than the Bracewell FHT. Therefore, the Hou FHT treatment provides the higher accuracy.

Figure 5.12 shows the CDF plots of the experimental data of these treatments N-point DHT computation. It was observed that as the number of point, N, of the DHT computation increases, the range magnitude of the treatments experimental data increases.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*

(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.11 : CDF plots of the N-point Bracewell and Hou FHTs Experimental Data

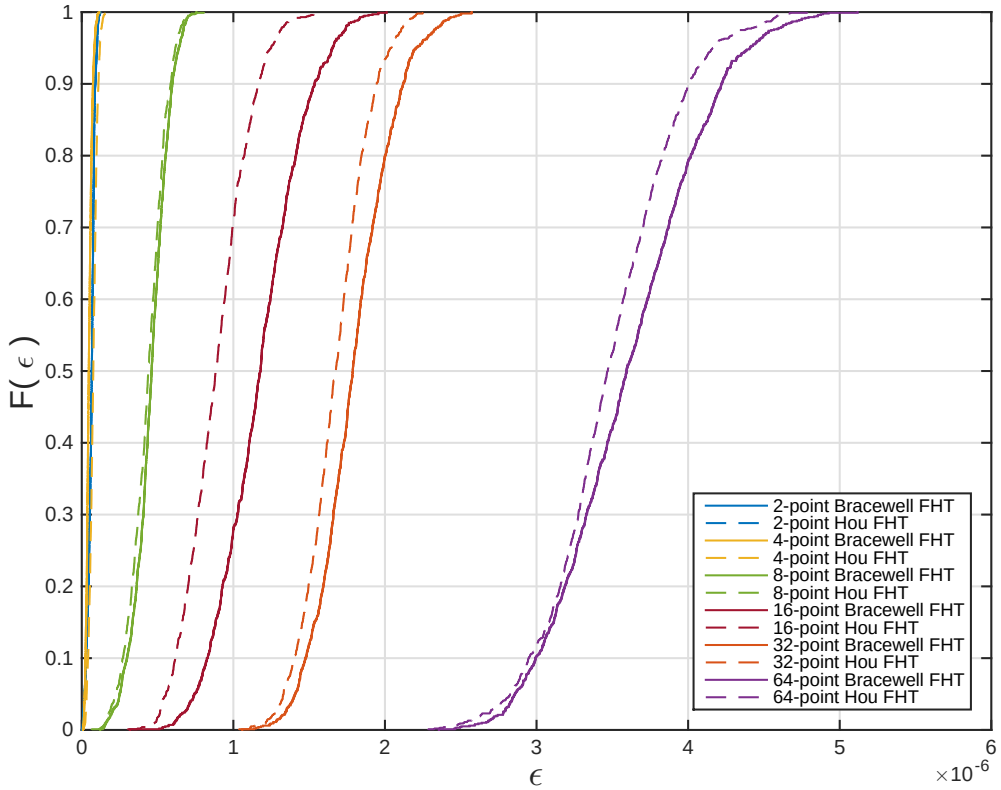


Figure 5.12 : CDF plots of the Bracewell and Hou FHTs Experimental Data

5.2.3 Discrete Cosine Transform

Table 5.15 shows the AD test results of the experimental normwise relative error of the N-point DCT treatments. Only two out of the 18 DCT experiments failed to reject the null hypothesis. Next are the detailed results of the two samples KS test of the N-point DCT treatments.

Table 5.15 : AD test results of the experimental data of the DCT structures

N-point	Direct DCT	Nikara FCT	Translation FCT
2	H_0 rejected	H_0 rejected	H_0 rejected
4	H_0 rejected	H_0 rejected	H_0 rejected
8	H_0 rejected	H_0 rejected	H_0 not rejected
16	H_0 rejected	H_0 rejected	H_0 rejected
32	H_0 rejected	H_0 rejected	H_0 rejected
64	H_0 rejected	H_0 not rejected	H_0 rejected

Direct DCT vs. Nikara FCT

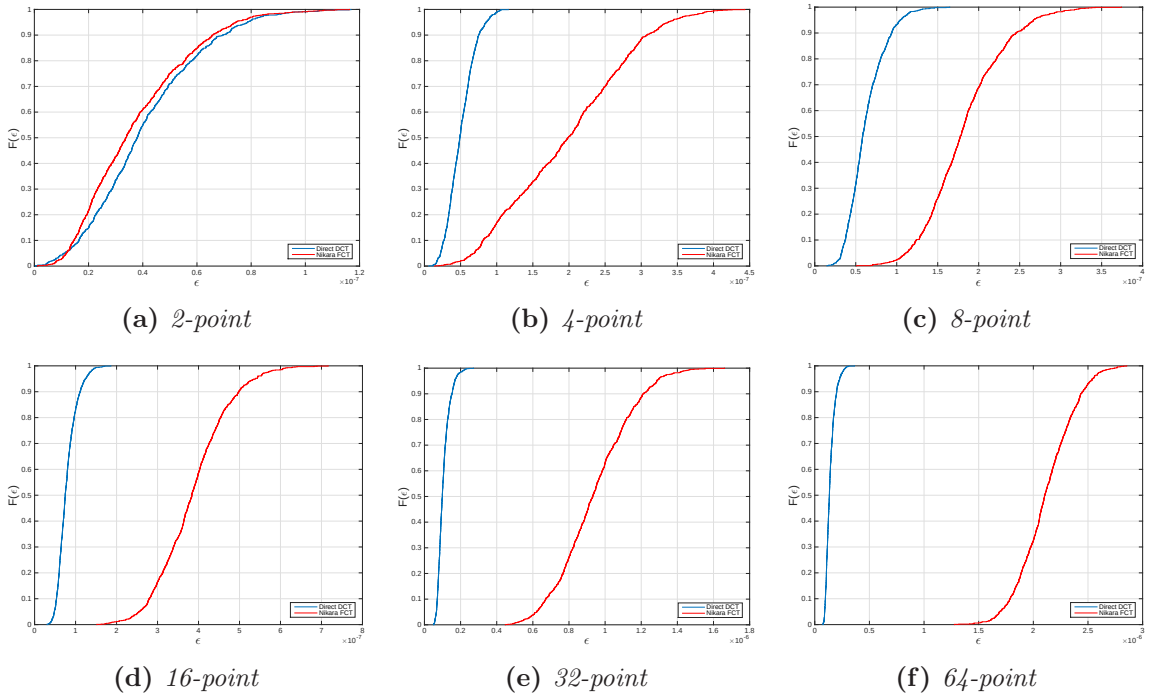
Table 5.16 shows the KS test results of the N-point Direct DCT and Nikara FCT treatments experiments. These tests rejected the null hypothesis of the KS test, therefore the difference in accuracy between these treatments were statistically significant.

Table 5.16 : KS test results of the Direct DCT and the Nikara FCT treatments

N-point	H_0	P-value	G^*
2	H_0 rejected	2.8793×10^{-5}	1.0500×10^{-1}
4	H_0 rejected	9.7020×10^{-315}	8.4600×10^{-1}
8	H_0 rejected	0	9.2200×10^{-1}
16	H_0 rejected	0	9.9700×10^{-1}
32	H_0 rejected	0	1.0000
64	H_0 rejected	0	1.0000

Figure 5.13 shows the CDF plots of these treatments experimental data of the N-point DCT computation. In these plots the significant differences in distributions, range, and range magnitude between these treatments proof a difference in their accuracy. In these CDF plots we observed that the range magnitude of the Direct

DCT experimental data was lower than the Nikara FCT. This means that the Direct DCT treatment provides the higher accuracy than the Nikara FCT.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*

(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.13 : CDF plots of the N-point Direct DCT and Nikara FCT Experimental Data

Figure 5.14 shows the CDF plots of the experimental data of these treatments N-point DCT computation. It was observed that as the number of point, N, of the DCT computation increases, the range magnitudes of the treatments experimental data increases. But the difference observed between these treatments was that the range magnitude of the Nikara FCT experimental data increased at a higher scale.

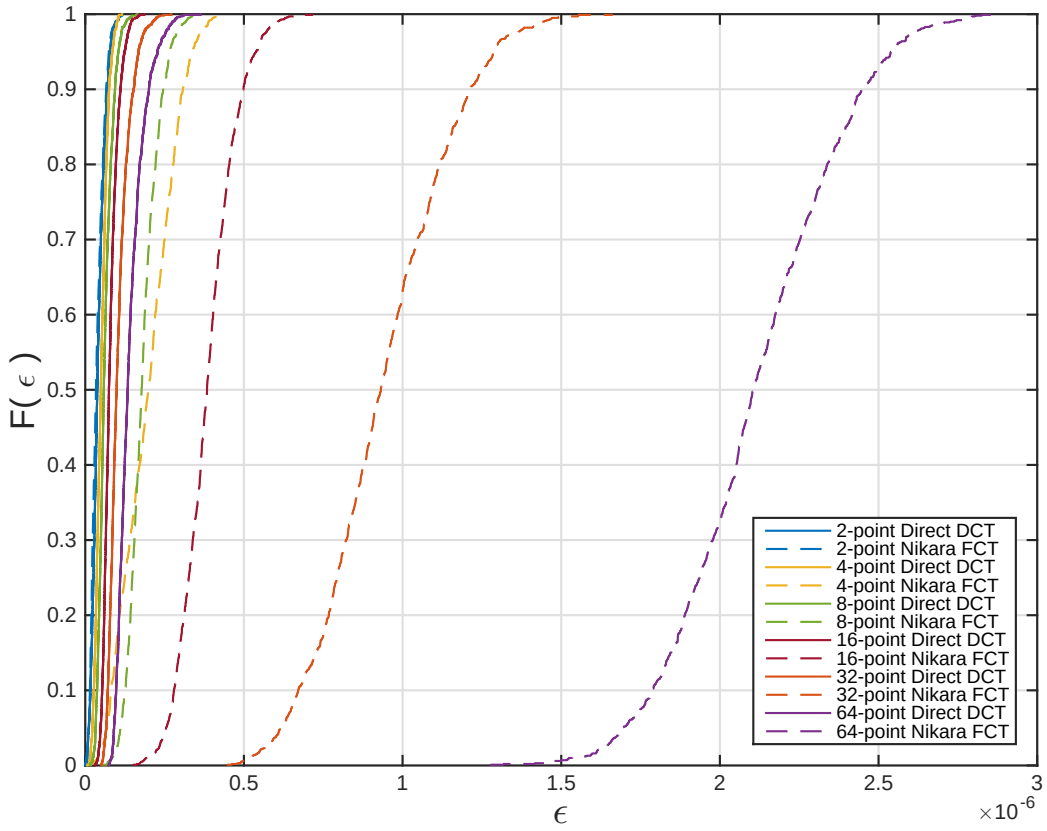


Figure 5.14 : CDF plots of the Direct DCT and Nikara FCT Experimental Data

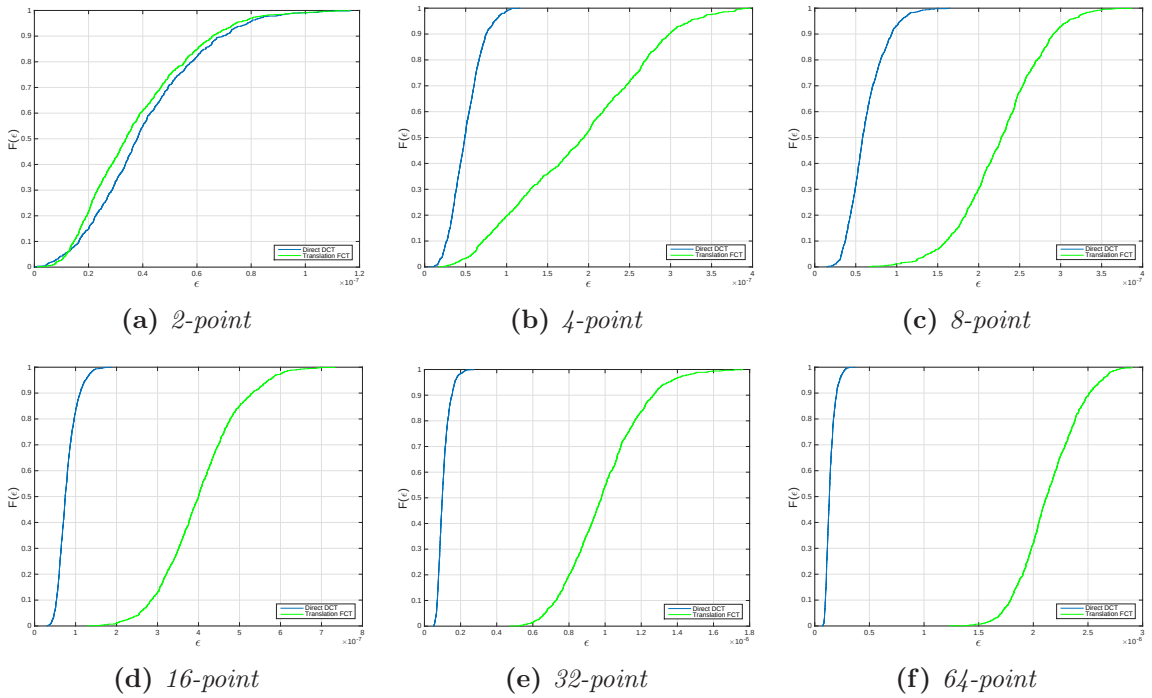
Direct DCT vs. Translation FCT

Table 5.17 shows the KS test results of the N-point Direct DCT and Translation FCT treatments experiments. These tests rejected the null hypothesis of the KS test, therefore the difference in accuracy between these treatments were statistically significant.

Table 5.17 : KS test results of the Direct DCT and the Translation FCT treatments

N-point	H_0	P-value	G^*
2	H_0 rejected	2.8793×10^{-5}	1.0500×10^{-1}
4	H_0 rejected	1.0153×10^{-285}	8.0600×10^{-1}
8	H_0 rejected	0	9.6400×10^{-1}
16	H_0 rejected	0	9.9500×10^{-1}
32	H_0 rejected	0	1.0000
64	H_0 rejected	0	1.0000

Figure 5.15 shows the CDF plots of these treatments experimental data of the N-point DCT computation. In these plots the significant differences in distributions, range, and range magnitude between these treatments proof a difference in their accuracy. Based on the range magnitude of the treatments experimental data, the Direct DCT treatment provides the higher accuracy.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*

(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.15 : CDF plots of the N-point Direct DCT and Translation FCT Experimental Data

Figure 5.16 shows the CDF plots of the experimental data of these treatments N-point DCT computation. It was observed that as the number of point, N, of the DCT computation increases, the range magnitudes of the treatments experimental data increases. But the difference observed between these treatments was that the range magnitude of the Translation FCT experimental data increased at a higher scale.

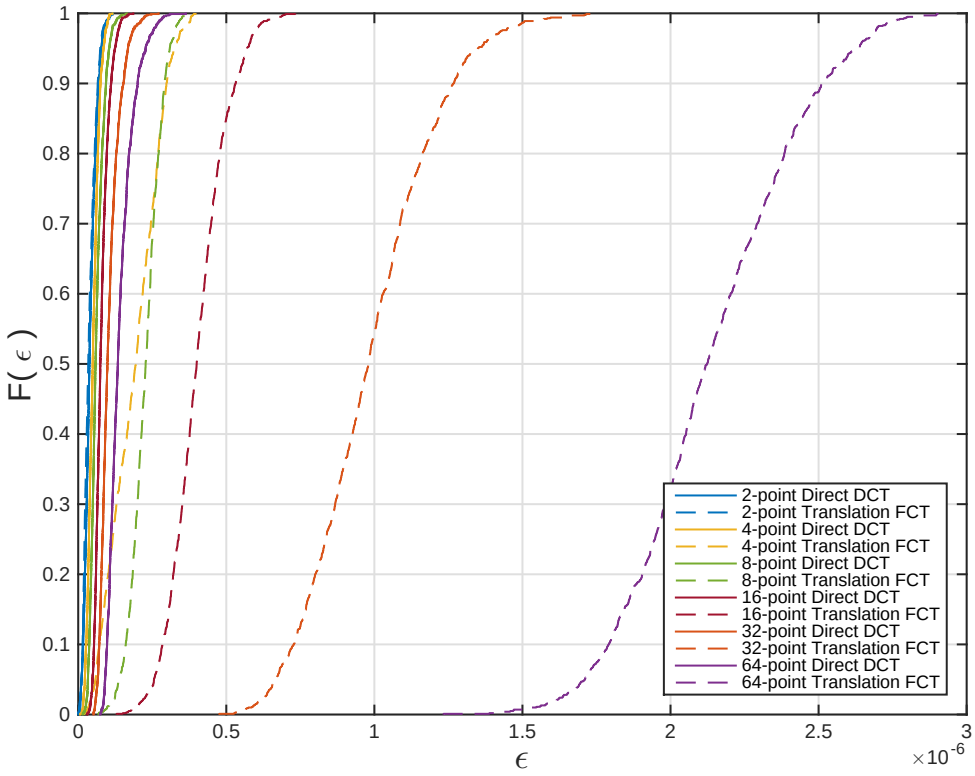


Figure 5.16 : CDF plots of the Direct DCT and Translation FCT Experimental Data

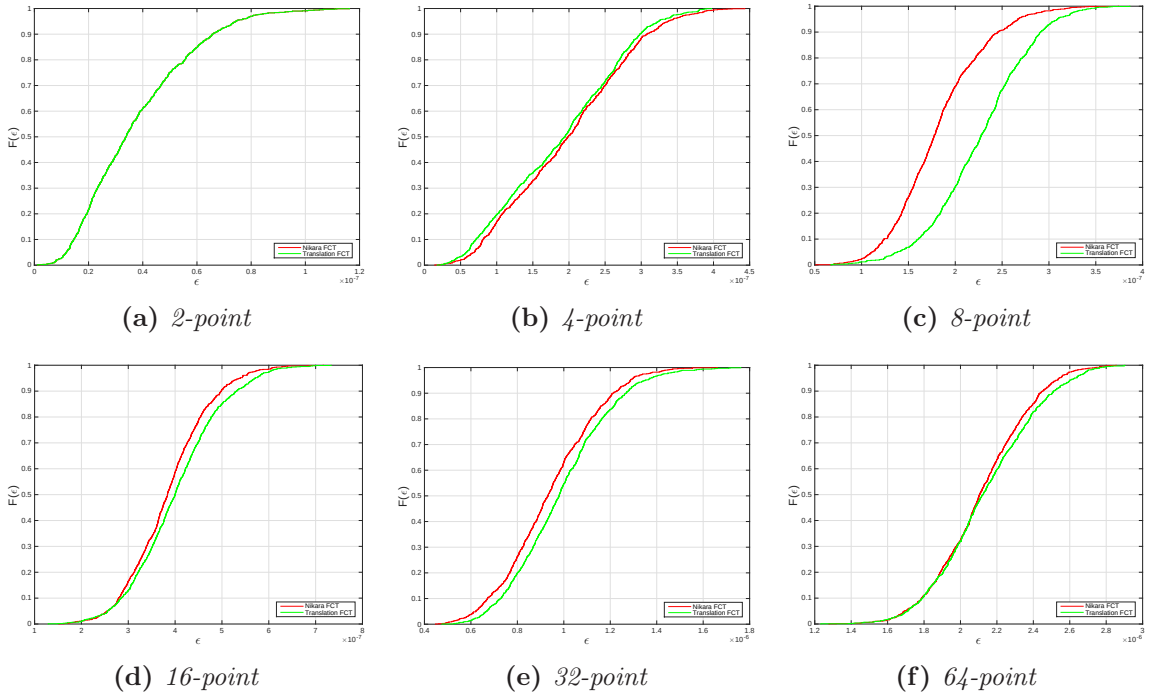
Nikara FCT vs. Translation FCT

Table 5.18 shows the KS test results of the N-point Nikara and Translation FCTs treatments experiments. Three out of the six tests rejected the null hypothesis of the KS test, therefore the difference in accuracy of between these treatments were statistically significant.

Table 5.18 : KS test results of the Nikara FCT and the Translation FCT treatments

N-point	H_0	P-value	G^*
2	H_0 not rejected	1.0000	0
4	H_0 not rejected	2.5751×10^{-1}	4.5000×10^{-2}
8	H_0 rejected	6.5747×10^{-68}	3.9200×10^{-1}
16	H_0 rejected	1.7936×10^{-4}	9.6000×10^{-2}
32	H_0 rejected	1.4756×10^{-4}	9.7000×10^{-2}
64	H_0 not rejected	1.0480×10^{-1}	5.4000×10^{-2}

Figure 5.17 shows the CDF plots of these treatments experimental data of the N-point DCT computation. The CDF plots of the treatments experimental data of the 2,4 and 64 -point DFT computations overlap; those were the experiments that failed to reject the null hypothesis of the KS test. While the other plots is notable the difference in location between the CDF plot of the treatments. The other experiments, which rejected the null hypothesis, we observed a significant differences in the distributions the treatments experimental data. In these plots the distributions of the Nikara FCT treatment tented to lower values. That means that the Nikara FCT treatment provide the higher accuracy than the Translation FCT.



(a) *2-point* **(b)** *4-point* **(c)** *8-point*

(d) *16-point* **(e)** *32-point* **(f)** *64-point*

Figure 5.17 : CDF plots of the N-point Nikara and Translation FCTs Experimental Data

Figure 5.18 shows the CDF plots of the experimental data of these treatments N-point DCT computation. It was observed that as the number of point, N, of the DCT computation increases, the range magnitude of the treatments experimental data increases.

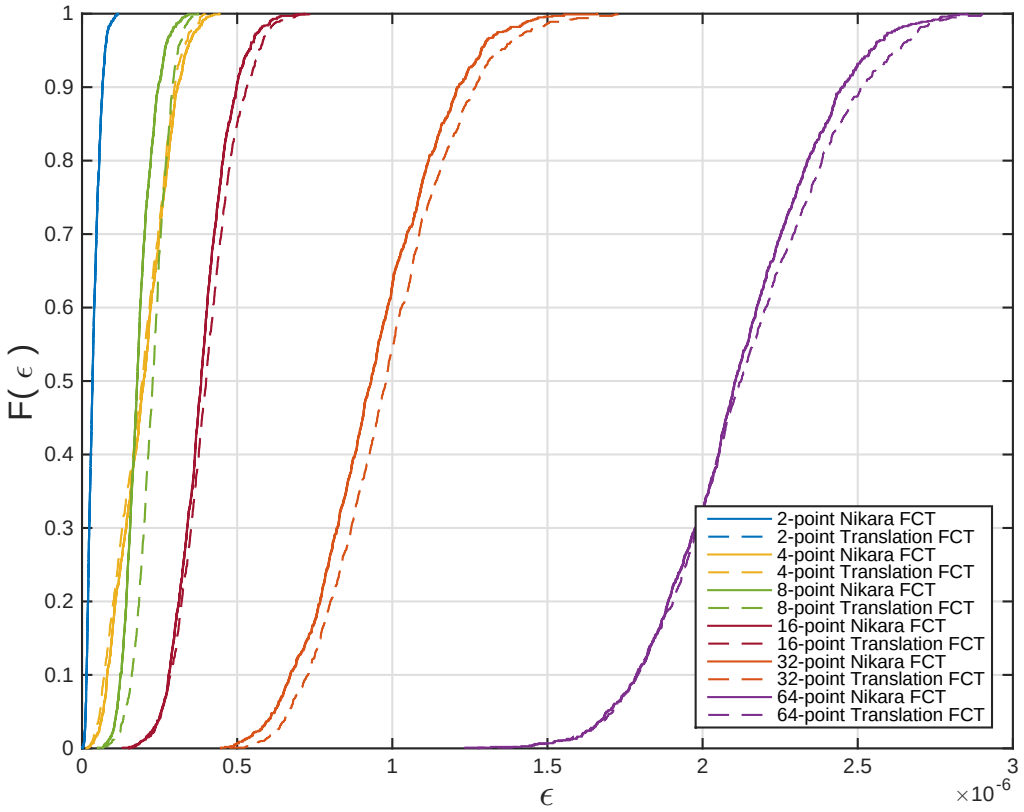


Figure 5.18 : CDF plots of the Nikara and Translation FCTs Experimental Data

5.3 Hardware Performance

This section presents the hardware performance results of the discrete transforms treatments evaluated. The hardware performance results includes the resources consumption and latencies of the treatments FPGA synthesis. The FPGA used was the Xilinx Virtex-7 XC7VX690T-2FFG1761C with system clock of 200 MHz.

The main FPGA hardware units used by these structures were the configurable logic block (CLB) and digital signal processing (DSP) slices. The main logic units that this FPGA CLB slices can generate are [29]

- Four 6-to-2 Look-up-Tables (LUTs)
- Eight Registers
- Two 8-to-1 Multiplexers (MUXs)
- One 16-to-1 Multiplexers (MUXs)

The equation used to approximate the CLB slices consumption based on the logic units consumptions in the *Resource Consumption Report* of the structures was

$$N_{CLBS} = \left\lceil \frac{N_{LUTs}}{4} \right\rceil + \left\lceil \frac{N_{REGs}}{8} \right\rceil + \left\lceil \frac{N_{MUX8-1}}{2} \right\rceil + N_{MUX16-1} \tag{5.32}$$

where N_{LUTs}, N_{REGs}, N_{MUX8-1}, and $N_{MUX16-1}$ refers to the number of LUTs, Registers, 8-to-1 MUXs, and 16-to-1 MUXs, respectively.

The DSP slices of this FPGA, the DSP48E1 slice, support many rapid arithmetic units as

- Multiplier
- Multiplier-Accumulator
- Multiplier-Adder

Next are the hardware performance results of the discrete transforms treatments evaluated in this work. Due to insufficient computer resources not all the N-point discrete transform treatments designs were synthesized and some of the synthesized designs were not able to produce the timing reports for the latency performance. The detailed hardware performance results are in Appendix C.

5.3.1 Discrete Fourier Transform

Figures 5.19 and 5.20 show the FPGA CLB and DSP slices consumption of the N-point DFT treatments designs. These results denote that the N-point Direct DFT treatment consumed more FPGA resources than the N-point FFTs treatments. The treatments that consumed less slices was the N-point Pease FFT treatment. It was also observed that as N increases, the amount of slices consumption of the DFT treatments designs increases.

Figure 5.21 shows the total latency of the N-point DFT treatments designs. In this plot was observed that the Direct DFT treatment had the highest total latency in most of the N-point DFT designs. Also, that the N-point Pease FFT treatment had the lowest total latency.

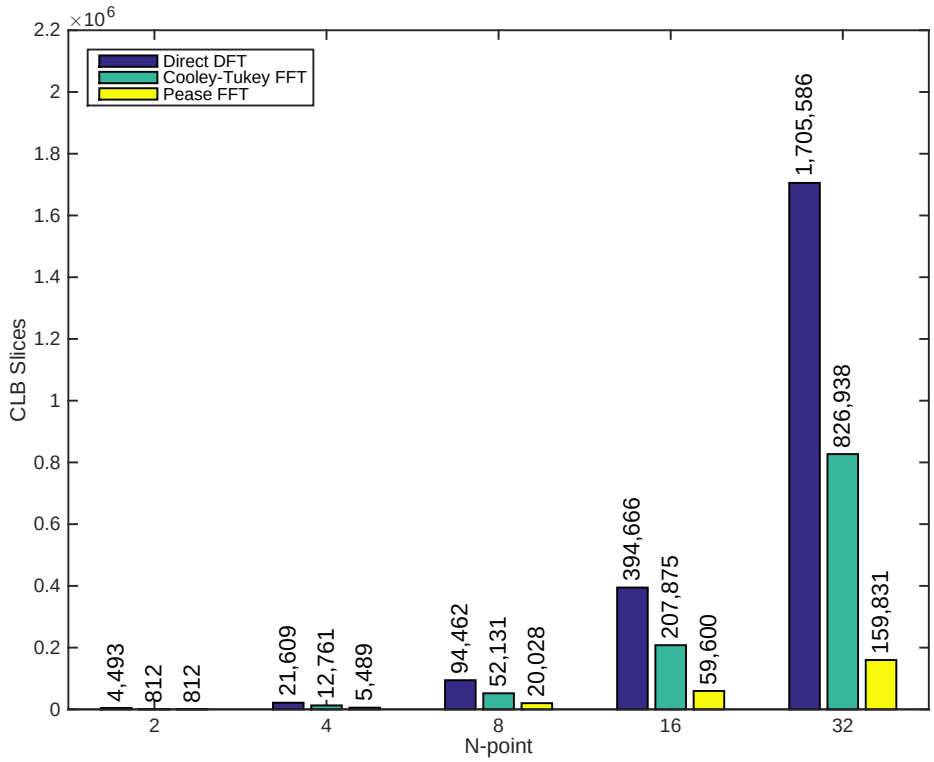


Figure 5.19 : FPGA CLB Slices consumption of the N-point DFT treatments

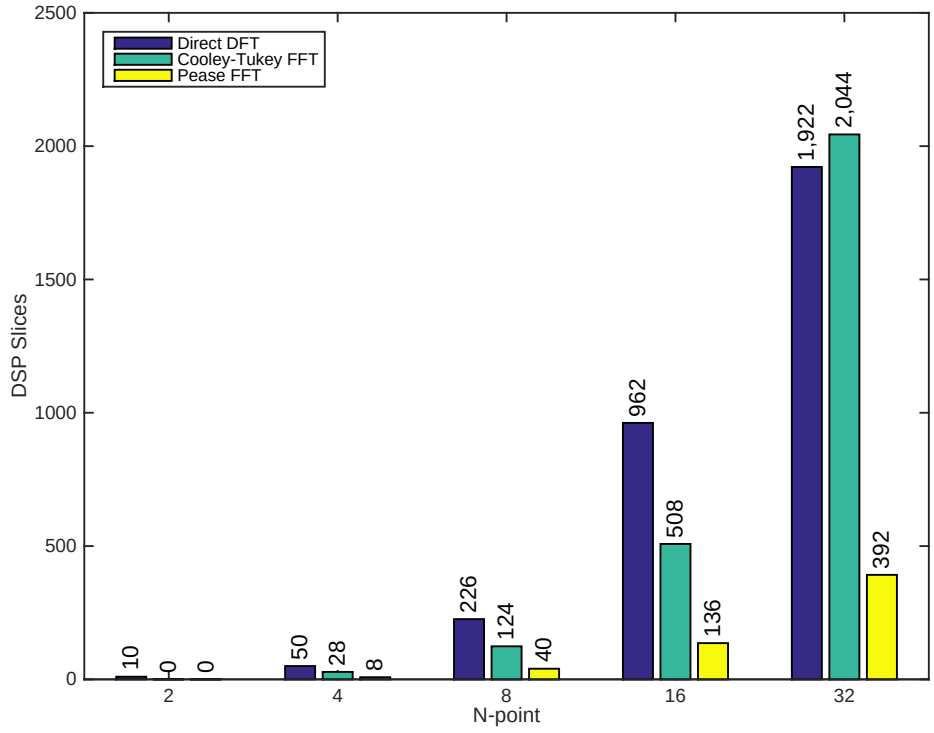


Figure 5.20 : FPGA DSP Slices consumption of the N-point DFT treatments

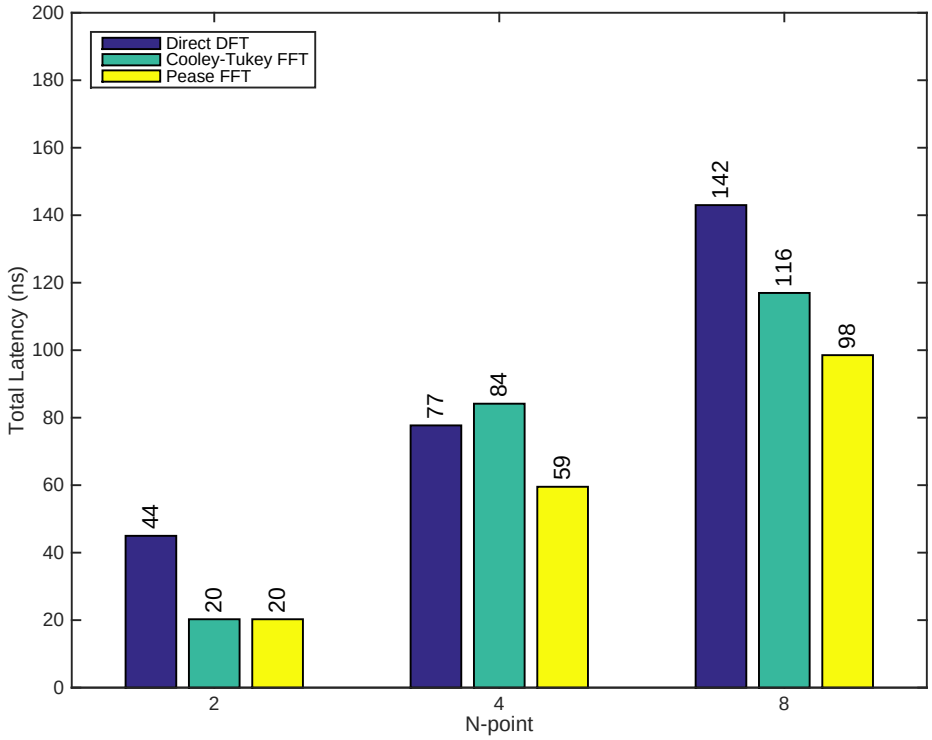


Figure 5.21 : Total latency of the N-point DFT treatments

5.3.2 Discrete Hartley Transform

Figure 5.22 shows the FPGA CLB slices consumption of the N-point DHT treatments designs. These results denote that the N-point Direct DHT treatment consumed more FPGA resources than the FHTs treatments. Its was observed in this plot that the amount of FPGA resources consumption of the Bracewell and Hou FHTs treatments was similar. In this plot we observed that as N increases, the amount of slices consumption of the DHT treatments designs increases.

Figure 5.23 shows the total latency of the N-point DHT treatments designs. It was observed in this plot that the N-point Direct DHT treatment designs had the highest total latency. Also, that the N-point Bracewell and Hou FHTs treatments designs had similar total latency. But the Hou FHT treatment had the lowest total latency.

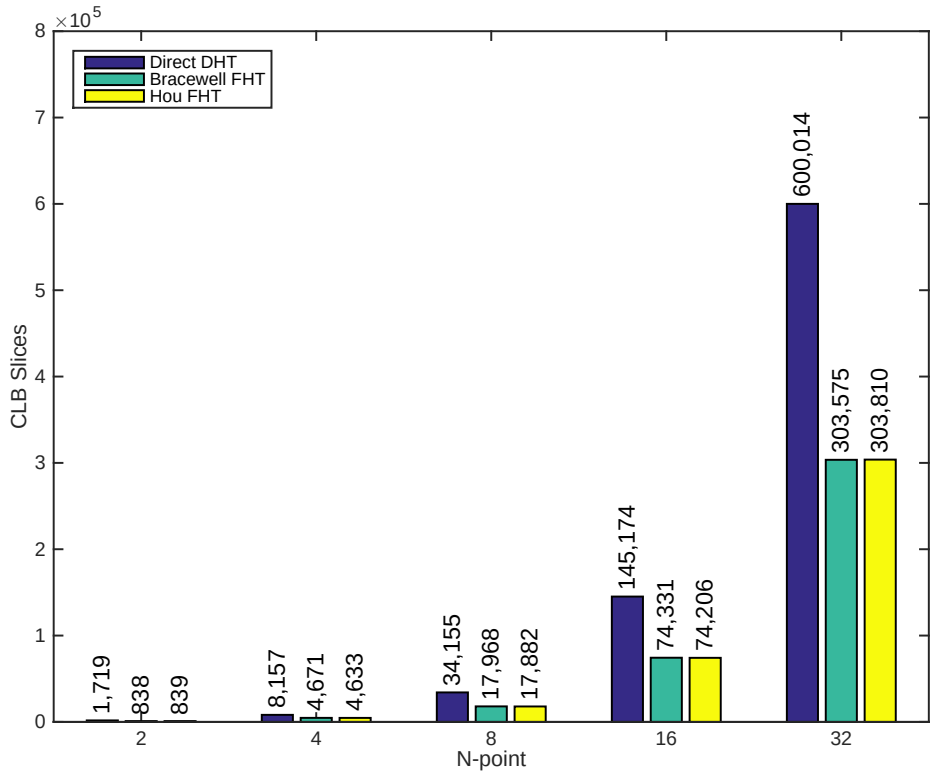


Figure 5.22 : FPGA CLB Slices consumption of the N-point DHT treatments

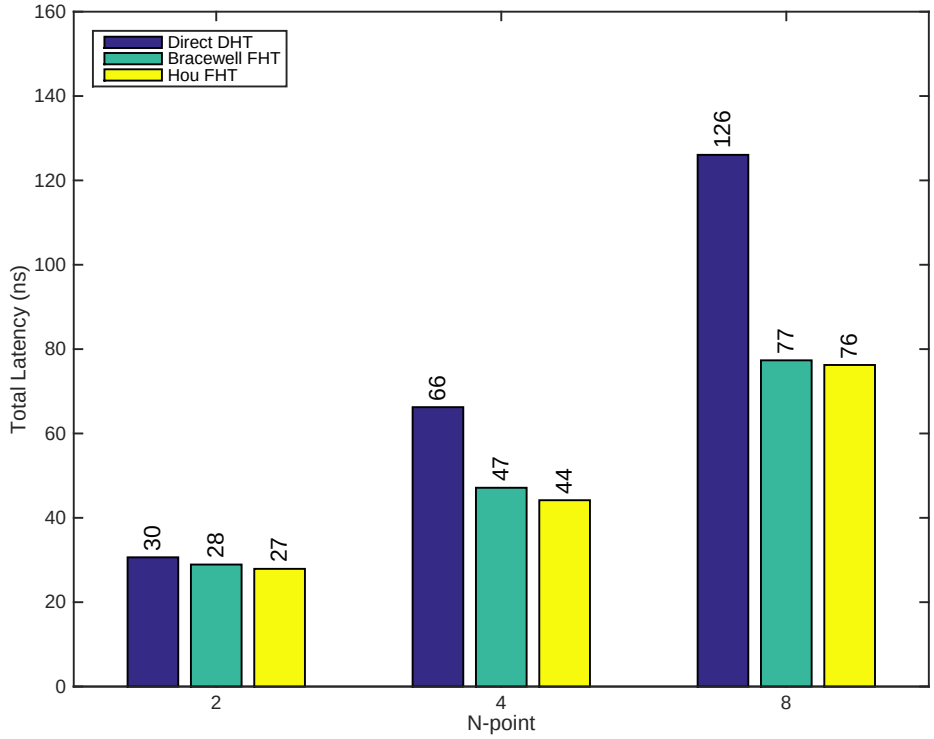


Figure 5.23 : Total latency of the N-point DHT treatments

5.3.3 Discrete Cosine Transform

Figures 5.24 and 5.25 show the FPGA CLB and DSP slices consumption of the N-point DCT treatments designs. These results show that the N-point Direct DCT treatment had the highest FPGA resource consumption. Also, showed that the N-point Nikara FCT treatment designs had the lowest FPGA resource consumption. Its was observed that as N increases, the amount of slices consumption of the DCT treatments designs increases.

Figure 5.26 shows the total latency of the N-point DCT treatments designs. This plot shows that the Direct DHT treatment had the highest total latency in most of the N-point DCT designs. Also, that the N-point Translation FCT treatment had the lowest total latency.

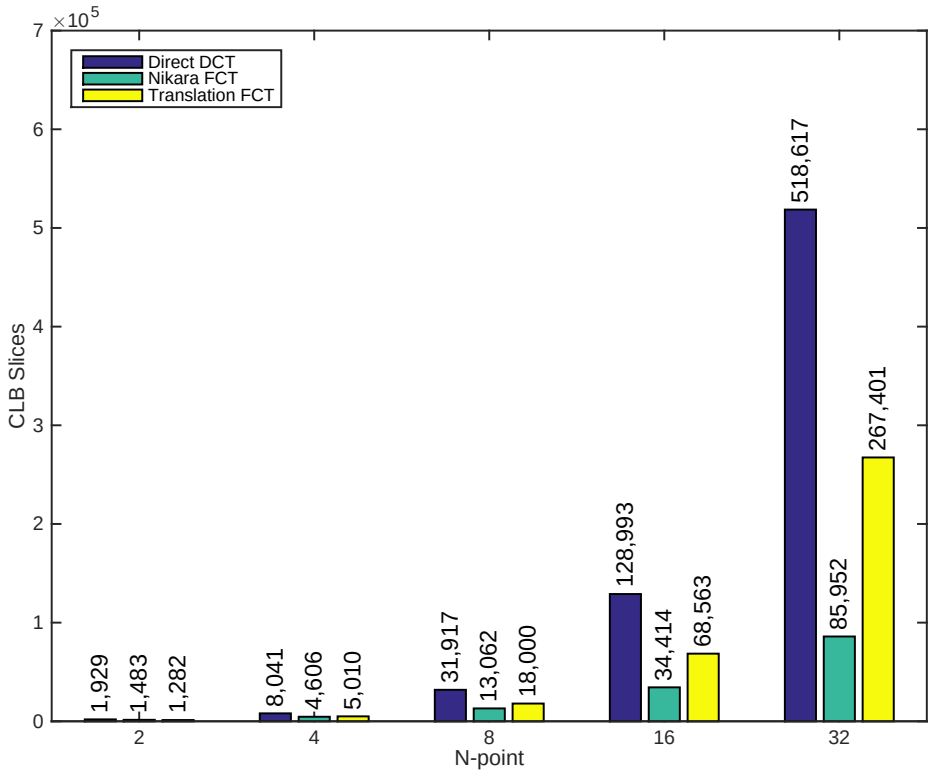


Figure 5.24 : FPGA CLB Slices consumption of the N-point DCT treatments

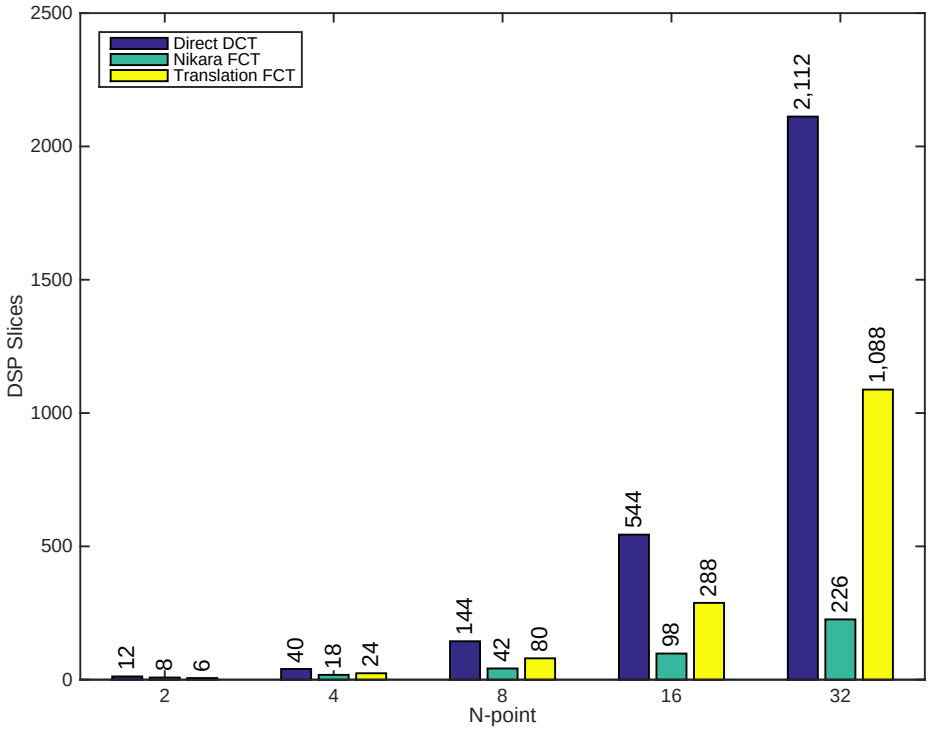


Figure 5.25 : FPGA DSP Slices consumption of the N-point DCT treatments

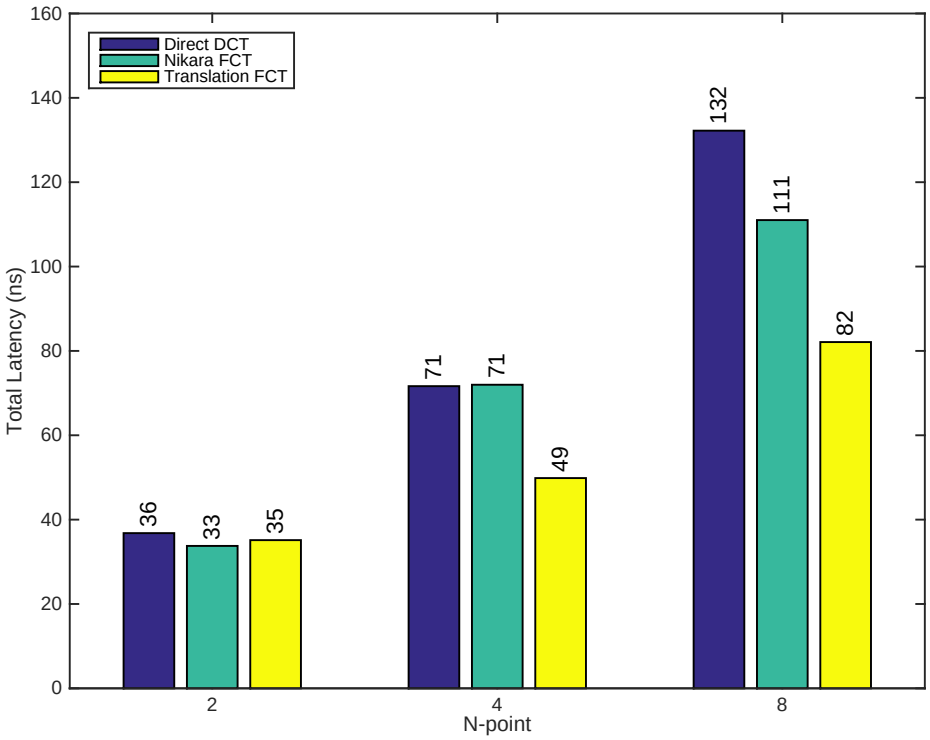


Figure 5.26 : Total latency of the N-point DCT treatments

5.4 Results Discussion

Next are the accuracy and performance results discussion of the evaluated discrete transforms treatments.

5.4.1 Discrete Fourier Transform

The error analysis of the DFT treatments shows that the N-point Direct DFT treatment had the lowest normwise relative error upper bound. In the validation of the error analysis it was observed that experimental normwise relative error range of the N-point Direct DFT treatment was lower than the FFTs treatments.

The statistical analysis of the DFT treatments determined a significant difference between the treatments experimental data distributions. This means that the treatment used for the DFT computation has its individual impact in accuracy, which is characterized by their factorization of the DFT matrix. In the statistical analysis it was observed that as the number of point of the DFT computation increases, the range magnitude of the treatments experimental data increases. The significant difference between the DFT treatments was that the range magnitude of the FFTs treatments increased at higher scales. Therefore, the Direct DFT treatment provides higher accuracy as the resolution of the DFT computation increases. Between the FFTs treatments, the Pease FFT provides higher accuracy than the Cooley-Tukey FFT.

The hardware performance results of the DFT treatments showed that the Direct DFT designs had the largest FPGA resources consumption; while the Pease FFT designs had the lowest resources consumption. In the latency results of the treatments designs we observed that most of the Direct DFT designs had the highest total latency. These hardware performance results agree with the theory of FFTs treatments, that these reduce the amount arithmetic operation. Therefore, the treatment used for DFT computation has a different impact in the hardware performance than other.

5.4.2 Discrete Hartley Transform

During the error analysis of the DHT treatments, two characteristics were observed. First we observed that the Direct DHT treatment had the lowest normwise relative error upper bound. The second was that the normwise relative error upper bound of the Bracewell and Hou FHTs treatments were the same. In the validation of the error analysis of these treatments the first characteristic was validated, the experimental normwise relative error range of the N-point Direct DHT treatment was lower than the FHTs treatments. The second characteristic was also validated, the experimental normwise relative error range of the N-point Bracewell and Hou FHTs treatments were similar and below the error analysis estimations.

In the statistical analysis of the DHT treatments was determined a significant difference between the treatments experimental data distributions. This result means that although the experimental normwise relative error range of the N-point Bracewell and Hou FHTs structures were similar, their distributions were not the same. Therefore, the treatment used for the DHT computation has its individual impact in accuracy, which is characterized by their factorization of the DHT matrix. In the statistical analysis it was observed that as the number of point of the DHT computation increases, the range magnitude of the treatments experimental data increases. The significant difference between the DHT treatments was that the range magnitude of the FHTs treatments increased at higher scales. Therefore, the Direct DHT treatment provides higher accuracy as the resolution of the DHT computation increases. Between the FHTs treatments, the Hou FHT provides higher accuracy.

The hardware performance results of the DHT treatments showed that the Direct DHT designs had the largest FPGA resources consumption and highest latencies than the FHTs treatments. We also observed that the hardware performance of the N-point Bracewell and Hou FHTs designs were similar; but the Hou FHT treatment had the lowest resource consumption and total latency. The main difference between

these FHTs treatments is that the Bracewell FHT is a decimation-in-time formulation, while the Hou FHT is a decimation-in-frequency formulation. These hardware performance results are proportional to the theory of these treatments, in terms of their reduction of arithmetic operations.

5.4.3 Discrete Cosine Transform

In the error analysis of the DCT treatments was observed that the N-point Direct DCT treatment had the lowest normwise relative error upper bound. During the validation of the error analysis we observed that the experimental normwise relative error range of the N-point Direct DCT treatment was lower than the FCTs treatments.

The statistical analysis of the DCT treatments determined a significant difference between the treatments experimental data distributions. This means that the treatment used for the DCT computation has its individual impact in accuracy, which is characterized by their factorization of the DCT matrix. In the statistical analysis it was observed that as the number of point of the DCT computation increases, the range magnitude of the treatments experimental data increases. The significant difference between the DCT treatments was that the range magnitude of the FCTs treatments increased at higher scales. Therefore, the Direct DFT treatment provides higher accuracy as the resolution of the DCT computation increases. Between the FCTs treatments, the Nikara FCT provides higher accuracy.

In the hardware performance results of the DCT treatment we observed that the Direct DCT designs had the highest FPGA resources consumption, and the highest latencies in most of the designs. We also observed that most of the Nikara FCT designs had the lowest resources consumptions. But, most of the Translation FCT designs had the lowest latencies. Therefore, the treatment used for the DCT computation has a different impact in the hardware performance than other treatments.

Chapter 6
Recommendations and Contributions

6.1 Recommendations

The results of this work showed a tradeoff between accuracy and hardware performance of the discrete transforms treatments. It was observed that the Direct discrete transforms treatments provided higher accuracy than the fast algorithms treatments, but its hardware performance was in disadvantage. Improvements to the hardware design of the direct discrete transform treatments can be performed to reduce resources consumption or latency. A sequential design approach is recommended to reduce hardware resource consumption. This approach reuses arithmetic cores (i.e. adders/subtracters, multiplier, sum of products, stages, etc) by introducing memory and data addressing blocks to the design [30]. The main drawback of the sequential design approach is that increments the total latency of the designs. To improve the total latency, its recommended the utilization of fast arithmetic operators units (i.e. adders/subtracters, multipliers) [5]. The disadvantage of these fast units is that consumes a higher amount of hardware resources.

The arithmetic structures of some of the fast algorithms treatments allow for structure modifications to improve the accuracy. The treatments are those that implement sum of products; i.e. Cooley-Tukey FFT, the Bracewell and Hou FHTs and the Translation FCT. A method to improve the accuracy of these treatments was discussed by Wilkinson [4]. The method sorts the elements of a sum of elements operator, $\sum_{i=0}^{N} a_i$, in ascending order of their value then proceed to the sum of the

sorted elements. This method tries to avoid the cancelation of elements. The disadvantage of this method is that the sorting unit increments the resource consumption and latency of the treatments.

To improve the accuracy of the Pease FFT and Nikara FCT treatments is recommended the utilization of floating point arithmetic units with higher precision. This method will impact the hardware performance of these treatments.

A future work could be the implementation of the previous recommendations to observe their effect in accuracy and hardware performance. Also, since the platform used for FPGA synthesis caused some limitations in this work, a future consideration would be using a different platform with more efficient memory usage in the computer host.

6.2 Contributions

The main contributions of this work include:

- A method to study the accuracy and hardware performance of the discrete transforms and their fast algorithms. The same method can be adjusted and applied to other arithmetic formulations.
- General estimations of the normwise relative error of the discrete transforms and their fast algorithms evaluated.
- Establishment that the treatment used for the computation of a discrete transform has its individual impact in the accuracy, which is characterized by their discrete transform factorization.
- Establishment of the existence of a tradeoff between the accuracy and the hardware performance of the discrete transforms and their fast algorithms.
- Recommendations to improve the hardware performance of the discrete transforms formulation with higher accuracy.
- Recommendations to improve the accuracy of discrete transform fast algorithms.

Chapter 7
Conclusions

A study of accuracy and performance of three discrete transforms and their fast algorithms was performed. The transforms evaluated were the Fourier (DFT), Hartley (DHT), and cosine (DCT). The DFT treatments evaluated were the Direct DFT, and the Cooley-Tukey and Pease Fast Fourier transforms (FFTs). The DHT treatments evaluated were the Direct DHT, and the Bracewell and Hou Fast Hartley transforms (FHTs). The DCT treatments evaluated were the Direct DCT, and the Nikara and Translation Fast Cosine transforms (FCTs).

The method established in this study consists of accuracy and hardware performance analyses. The accuracy analysis combined approximation and statistical methods to quantify error and determine the differences and similitudes between the discrete transforms and their fast algorithms. The hardware performance analysis compared the resource consumption and latency of the treatments based on their FPGA synthesis. These methods served to determine which treatment provided the highest accuracy and what were their hardware performance implications. This method can be adjusted and used in other arithmetic formulations and their fast algorithms. By using this methodology a better understanding of how the arithmetic structure of a formulation affects its accuracy and hardware performance is achieved.

The results of this study show a tradeoff between the accuracy and hardware performance properties of the discrete transforms treatments. The treatments with lower normwise relative error (higher accuracy), provided higher FPGA resources consumption and latencies; and vice versa. In the accuracy analysis the Direct discrete

transforms treatment provided the higher accuracy. Between the FFT treatments, Pease provided higher accuracy and better hardware performance than the Cooley-Tukey. The FHT treatments showed similar results in the forward error analysis, but the statistical analysis showed a significant difference in the experimental normwise relative error. Where the Hou FHT treatment provide higher accuracy than the Bracewell. The hardware performance of the FHT treatments were similar. The accuracy analysis of the FCT treatments showed a significant difference in the accuracy, in the Nikara FCT treatment provided a higher accuracy.

In the accuracy analysis we observed that as the resolution of the discrete transform computation increases, the range magnitude of the treatments experimental normwise relative error increases. The differences observed between the treatments was that the range magnitude of the fast algorithms incremented at a higher scale.

These results showed a significant difference in accuracy between the discrete transforms and their fast algorithms. Therefore the hypothesis of this work is accepted.

Bibliography

[1] D. N. Arnold. The Patriot Missile Failure. August 2000. http://www.ima.umn.edu/~arnold/disasters/patriot.html Last visited on October 16, 2014.

[2] D. N. Arnold. The Explosion of the Ariane 5. August 2000. http://www.ima.umn.edu/~arnold/disasters/ariane.html Last visited on October 16, 2014.

[3] A. Jones. 10 Seriously Epic Computer Software Bugs. December 2012. http://listverse.com/2012/12/24/10-seriously-epic-computer-software-bugs/ Last visited on October 16, 2014.

[4] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, New Jersey, 1963.

[5] I. Koren. *Computer Arithmetic Algorithms*. Prentice-Hall, New Jersey, 1993.

[6] N. Tsao. On the distribution of significant digits and roundoff errors. *IEEE Transactions on Computers*, pages 577–586, June 1974.

[7] D. J. Kuck, D. S. Parker Jr., and A. H. Sameh. Analysis of rounding methods in floating-point arithmetic. *IEEE Transactions on Computers*, C-26:643–650, July 1977.

[8] D. R. Stoutemyer. Automatic Error Analysis Using Computer Algebraic Manipulation. *ACM Transaction on Mathematic Software*, 3:26–43, March 1977.

[9] I. D. Yun and S. U. Lee. On the Fixed-Point Error Analysis of Several Fast DCT Algorithms. *IEE Transactions on Circuits and Systems for Video Technology*, 3, February 1993.

[10] I. D. Yun and S. U. Lee. On the Fixed-Point Error Analysis of Several Fast IDCT Algorithms. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, 42, November 1995.

[11] K. S. Harish and K. M. M. Prabhu. Fixed-point error analysis of two DCT algorithms. *IEE Proceedings Visual Image Signal Processing*, April 2000.

[12] M. M. Rao and K. M. M. Prabhu. Fixed-point error analysis of radix-4 FHT algorithm with imized scaling schemes. *IEE Proceedings Visual Image Signal Processing*, April 1995.

[13] N. Glaros and G. Carayannis. Exact and First-Order Error Analysis of the Schur and Split Schur Algorithms: Theory and Practice. *IEE Transactions on Signal Processing*, August 1994.

[14] N. Tsao. Error Complexity Analysis of Algorithms for Matrix Multiplication and Matrix Chain Product. *IEEE Transactions on Computers*, October 1981.

[15] I. Pitas and M. G. Strintzis. Floating Point Error Analysis of Two-Dimensional Fast Fourier Transform Algorithms. *IEEE Transactions on Circuits and Systems*, January 1988.

[16] V. Britanak. Discrete Cosine and Sine Transforms. In K. R. Rao and P. C. Yip, editors, *The Transform and Data Compression Handbook*. CRC Press, 2001.

[17] Sanjit K. Mitra. *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill, New York, NY, 2001.

[18] H. M. Oliveira and R. M. Campello de Souza. Orthogonal Multilevel Spreading Sequence Design. In *Coding, Communications and Broadcasting*, pages 291–303. Hertfordshire: Research Studies Press, 2000.

[19] J. W. Cooley and J. W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics Computation*, 19:297–301, 1965.

[20] M. C. Pease. An Adaptation of the Fast Fourier Transform for Parallel Processing. *Journal of the Association for Computing Machinery*, 15:252–264, April 1968.

[21] J. R. Johnson. Pease FFT Algorithm. *Technical Report DU-MCS-98-01, Department of Mathematics and Computer Science, Drexel University*, November 1998.

[22] R. A. Bracewell. The Fast Harley Transform. *Proceedings of the IEEE*, 72:1010–1018, 1984.

[23] H. S. Hou. The Fast Hartley Transform Algorithm. *IEEE Transactions on Computers*, C-36:147 – 156, 1987.

[24] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, C-23:90–93, January 1974.

[25] J. Nikara. *Application-Specific Parallel Structure for Discrete Cosine Transform and Variable Legth Decoding*. PhD thesis, Tampere University of Technology, June 2004.

[26] M. Püschel and J. M. F. Moura. The Algebraic Approach to the Discrete Cosine and Sine Transforms and Their Fast Algorithms. *SIAM Journal on Computing*, 32:1280–1316, 2003.

[27] NIST/SEMATECH. *e-Handbook of Statistical Methods*. http://www.itl.nist.gov/div898/handbook/, 2003.

[28] G. W. Corder and D. I. Foreman. *Nonparametric Statistics: A Step-by-Step Approach*. John Wiley & Sons Inc, New York, 2014.

[29] Xilinx. *7 Series FPGAs Configurable Logic Block User Guide*, 2014.

[30] Agenor Polo Zabaleta. Design Methodology for Resource Efficient Implementation of Fast Fourier Transform Cores on Field Programmable Gate Arrays. Master's thesis, University of Puerto Rico - Mayaguez Campus, 2012.

Appendices

Appendix A
MATLAB Code of the Software Designs

A.1 Discrete Fourier Transform

```
function results = FT_Simulation( AlgType,data,N,S,DP )
  N=double(N);
  if DP==0
    results(1:S,1:N)=single(0);
    if AlgType==1
      cDFTs(1:N,1:N)=single(0);
      for row=single(1:1:N)
        for col=single(1:1:N)
          angle=((pi*2*(row-1)*(col-1))/N);
          cDFTs(row,col)=single(exp(-1i*angle));
        end
      end
      for s=1:1:S
        for row=1:1:N
          %SOP
          partialResult=single(0);
          for col=1:1:N
            partialResult=CASim_SP(partialResult,...
              CMSim_SP(data(s,col),cDFTs(row,col)));
          end
          results(s,row)=partialResult;
        end
      end
    elseif AlgType==2
```

```
  cDFTs(1:N/2,1:N/2)=single(0);
  for row=single(1:1:N/2)
    for col=single(1:1:N/2)
      angle=single((pi*4*(row-1)*(col-1))/N);
      cDFTs(row,col)=single(exp(-1i*angle));
    end
  end
  for s=1:1:S
    dataP=PERM(data(s,1:N),'STn',2);
    dSE=dataP(1:N/2);
    dSO=dataP((N/2)+1:N);
    for row=single(1:1:N/2)
      PRE=single(0);
      PRO=single(0);
      % SOP
      for col=1:1:N/2
        PRE=CASim_SP(PRE,CMSim_SP(dSE(col),cDFTs(row,col)));
        PRO=CASim_SP(PRO,CMSim_SP(dSO(col),cDFTs(row,col)));
      end
      angle=single((2*pi*(row-1))/N);
      twf=single(exp(-1i*angle));
      PROM=CMSim_SP(PRO,twf);
      results(s,row)=CASim_SP(PRE,PROM);
      results(s,row+(N/2))=CASim_SP(PRE,-PROM);
    end
  end
else%if AlgType==3
  n=log2(N);
  twf(1:n,1:N/2)=single(0);
  for st=n:-1:1
    row=1;
    for m=1:1:2^(n-st)
      for bf=1:1:2^(st-1)
```

```
            j=single((m-1)*(2^(st-1)));
            angle=single((2*pi*j)/N);
            twf(st,row)=single(exp(-1i*angle));
            row=row+1;
          end
        end
      end
      StageResult(1:n+1,1:N)=single(0);
      for s=1:1:S
        StageResult(n+1,1:N)=PERM(data(s,1:N),'BR',0);
        for st=n:-1:1
          for k=1:1:N/2
            CM_RESULT=CMSim_SP(StageResult(st+1,k*2),twf(st,k));
            BFResult((k*2)-1)=CASim_SP(StageResult(st+1,(k*2)-1),...
              CM_RESULT);
            BFResult(k*2)=CASim_SP(StageResult(st+1,(k*2)-1),...
              -CM_RESULT);
          end
          StageResult(st,1:N)=PERM(BFResult,'STn',2);
        end
        results(s,1:N)=StageResult(1,1:N);
      end
    end
  else
    results(1:S,1:N)=0;
    cDFTs(1:N,1:N)=0;
    for row=single(1:1:N)
      for col=single(1:1:N)
        angle=single(((pi*2*(row-1)*(col-1))/N));
        cDFTs(row,col)=double(single(exp(-1i*angle)));
      end
    end
    for s=1:1:S %S:SET
```

```
      for row=1:1:N
        %SOP
        partialResult=double(0);
        for col=1:1:N
          partialResult=CASim_DP(partialResult,...
            CMSim_DP(data(s,col),cDFTs(row,col)));
        end
        results(s,row)=partialResult;
      end
    end
  end
end
```

A.2 Discrete Hartley Transform

```
function results = HT_Simulation( AlgType,data,N,S,DP )
  results(1:S,1:N)=0;
  if DP==0
    if AlgType==1
      dht(1:N,1:N)=single(0);
      for row=single(1:1:N)
        for col=single(1:1:N)
          angle=single((2*pi*(row-1)*(col-1))/N);
          dht(row,col)=single(single(cos(angle))+single(sin(angle)));
        end
      end
      for s=1:1:S
        for row=1:1:N
          %SOP
          partialResult=single(0);
          for col=1:1:N
            partialResult=FlPASim_SP(partialResult,...
              FlPMSim_SP(data(s,col),dht(row,col)));
          end
```

```
      results(s,row)=partialResult;
    end
  end
elseif AlgType==2
  dhtE(1:N/2,1:N/2)=single(0);
  dhtO(1:N/2,1:N/2)=single(0);
  for row=single(1:1:N/2)
    for col=single(1:1:N/2)
      angleE=single((2*pi*(row-1)*(col-1))/(N/2));
      dhtE(row,col)=single(single(cos(angleE))...
        +single(sin(angleE)));
      angleO=single((2*pi*(row-1)*(col-(1/2)))/(N/2));
      dhtO(row,col)=single(single(cos(angleO))...
        +single(sin(angleO)));
    end
  end
  for s=1:1:S
    %STRIDE-2 PERM
    dataP=PERM(data(s,1:N),'STn',2);
    dPE(1:N/2)=dataP(1:N/2);
    dPO(1:N/2)=dataP((N/2)+1:N);
    for row=1:1:N/2
      PRE=single(0);
      PRO=single(0);
      %SOP
      for col=1:1:N/2
        PRE=FlPASim_SP(PRE,FlPMSim_SP(dPE(col),dhtE(row,col)));
        PRO=FlPASim_SP(PRO,FlPMSim_SP(dPO(col),dhtO(row,col)));
      end
      results(s,row)=FlPASim_SP(PRE,PRO);
      results(s,row+(N/2))=FlPASim_SP(PRE,-PRO);
    end
  end
```

```
else%if AlgType==3
  dhtE(1:N/2,1:N/2)=single(0);
  dhtO(1:N/2,1:N/2)=single(0);
  for row=single(1:1:N/2)
    for col=single(1:1:N/2)
      angleE=single((2*pi*(row-1)*(col-1))/(N/2));
      dhtE(row,col)=single(single(cos(angleE))...
        +single(sin(angleE)));
      angleO=single((2*pi*(row-(1/2))*(col-1))/(N/2));
      dhtO(row,col)=single(single(cos(angleO))...
        +single(sin(angleO)));
    end
  end

  F2_E(1:N/2)=single(0);
  F2_O(1:N/2)=single(0);

  for s=1:1:S
    PREven(1:N/2)=data(s,1:N/2);
    PROdd(1:N/2)=data(s,(N/2)+1:N);
    for row=1:1:N/2
      F2_E(row)=FlPASim_SP(PREven(row),PROdd(row));
      F2_O(row)=FlPASim_SP(PREven(row),-PROdd(row));
    end
    PR(1:N)=single(0);
    for row=1:1:N/2
      %SOP
      PRE=single(0);
      PRO=single(0);
      for col=1:1:N/2
        PRE=FlPASim_SP(PRE,FlPMSim_SP(F2_E(col),dhtE(row,col)));
        PRO=FlPASim_SP(PRO,FlPMSim_SP(F2_O(col),dhtO(row,col)));
      end
```

```
          PR(row)=PRE;
          PR(row+(N/2))=PRO;
        end
        %STRIDE-2 PERM
        results(s,1:N)=PERM(PR,'STn',N/2);
      end
    end
  else
    dht(1:N,1:N)=0;
    for row=single(1:1:N)
      for col=single(1:1:N)
        angle=single((2*pi*(row-1)*(col-1))/N);
        dht(row,col)=double(single(cos(angle)+sin(angle)));
      end
    end
    for s=1:1:S
      for row=1:1:N
        %SOP
        partialResult=0;
        for col=1:1:N
          partialResult=FlPASim_DP(partialResult,...
            FlPMSim_DP(data(s,col),dht(row,col)));
        end
        results(s,row)=partialResult;
      end
    end
  end
end
```

A.3 Discrete Cosine Transform

```
function results = CT_Simulation( AlgType,data,N,S,DP)
  n=log2(N);
  results(1:S,1:N)=0;
```

```
if DP==0
  if AlgType==1
    cons(1:N)=single(0);
    cons(1)=(1/single(N))^(1/2);
    cons(2:N)=(2/single(N))^(1/2);
    c(1:N,1:N)=single(0);
    for row=single(1:1:N)
      for col=single(1:1:N)
        angle=single((pi*(row-1)*((col-1)+(1/2)))/N);
        c(row,col)=single(cos(angle));
      end
    end
    for s=1:1:S
      for row=single(1:1:N)
        partialResult=single(0);
        %SOP
        for col=single(1:1:N)
          partialResult=FlPASim_SP(partialResult,...
            FlPMSim_SP(single(data(s,col)),c(row,col)));
        end
        results(s,row)=FlPMSim_SP(partialResult,cons(row));
      end
    end
  elseif AlgType==2
    for s=1:1:S
      cons=(2/single(N))^(1/2);
      dataP(1:N)=single(data(s,1:N));
      % HADAMARD PERMUTATION
      dataP(1:N)=single(PERM(dataP(1:N),'Ha',0));
      % A(0)
      st=0;
      dataP(1:N)=single(ID_N_F_2(dataP(1:N),0));
      dataP(1:N)=single(PERM(dataP(1:N),'H',st));
```

```
    dataP(1:N)=single(DM(dataP(1:N),CT_A2_COEF(st,N),0));
    dataP(1:N)=single(CT_A2_MN(dataP(1:N),st));
    for st=1:1:n-1
      % [I_{N/2^(S+1)} (X) P^{T}_{2^(S+1),2}]
      dataP(1:N)=single(PERM(dataP(1:N),'S',st));
      dataP(1:N)=single(ID_N_F_2(dataP(1:N),0));
      dataP(1:N)=single(PERM(dataP(1:N),'H',st));
      dataP(1:N)=single(DM(dataP(1:N),CT_A2_COEF(st,N),0));
      dataP(1:N)=single(CT_A2_MN(dataP(1:N),st));
    end
    dataP(1:N)=single(PERM(dataP(1:N),'U',st));
    %FACTOR MULTIPLIER
    for i=1:1:N
      results(s,i)=single(FlPMSim_SP(dataP(i),cons));
    end
  end
elseif AlgType==3
  cons(1:N)=single(0);
  cons(1)=(1/single(N))^(1/2);
  cons(2:N)=(2/single(N))^(1/2);
  cIIb(1:N/2,1:N/2)=single(0);
  cIVb(1:N/2,1:N/2)=single(0);
  for row=single(1:1:N/2)
    for col=single(1:1:N/2)
      angleIIb=single((2*pi*(row-1)*(col-(1/2)))/N);
      cIIb(row,col)=single(cos(angleIIb));
      angleIVb=single((2*pi*(row-(1/2))*(col-(1/2)))/N);
      cIVb(row,col)=single(cos(angleIVb));
    end
  end
  for s=1:1:S % SAMPLE
    dataP(1:N)=PERM(data(s,1:N),'IIb',0);
    dPFH=dataP(1:N/2);
```

```
      dPSH=dataP((N/2)+1:N);
      %F_2_ID_N
      dE(1:N/2)=single(0);
      dO(1:N/2)=single(0);
      for i=1:1:N/2
        dE(i)=FlPASim_SP(dPFH(i),dPSH(i));
        dO(i)=FlPASim_SP(dPFH(i),-dPSH(i));
      end
      PRDCTb(1:N)=single(0);
      for row=1:1:N/2
        PRE=single(0);
        PRO=single(0);
        %BEGIN SOP
        for col=1:1:N/2
          PRE=FlPASim_SP(PRE,FlPMSim_SP(dE(col),cIIb(row,col)));
          PRO=FlPASim_SP(PRO,FlPMSim_SP(dO(col),cIVb(row,col)));
        end
        PRDCTb(row)=PRE;
        PRDCTb(row+(N/2))=PRO;
      end
      dataP(1:N)=PERM(PRDCTb,'STn',N/2);
      for row=1:1:N
        results(s,row)=FlPMSim_SP(dataP(row),cons(row));
      end
    end
  end
else
  cons(1:N)=0;
  cons(1)=double((1/single(N))^(1/2));
  cons(2:N)=double((2/single(N))^(1/2));
  c(1:N,1:N)=0;
  for row=single(1:1:N)
    for col=single(1:1:N)
```

```
        angle=single((pi*(row-1)*((col-1)+(1/2)))/N);
        c(row,col)=double(single(cos(angle)));
      end
    end
    for s=1:1:S
      for row=1:1:N
        %SOP
        partialResult=0;
        for col=1:1:N
          partialResult=FlPASim_DP(partialResult,...
            FlPMSim_DP(double(data(s,col)),c(row,col)));
        end
        results(s,row)=FlPMSim_DP(partialResult,cons(row));
      end
    end
  end
end
```

Appendix B
MATLAB Code to Generate the Hardware Designs

B.1 Discrete Fourier Transform

```
function FT_Code_Gen(Alg,N,folder)
  vcode=verifyParameters(Alg,N);
  if vcode==-1
    disp('Invalid parameters');
    return;
  end
  fnName=['FT'  sprintf('%02d',N) '_A' num2str(Alg)];
  fnFFT=[folder fnName '.vhd'];
  frFFT=fopen(fnFFT(2:length(fnFFT)),'wt');
  % VHDL LIBRARIES
  fprintf(frFFT,'library IEEE;\n');
  fprintf(frFFT,'use IEEE.std_logic_1164.all;\n\n');
  % STRUCTURE ENTITY
  FT_entity(frFFT,N,fnName);
  % STRUCTURE ARCHITECTURE
  fprintf(frFFT,['architecture arch' fnName ' of ' fnName ' is\n\n']);
  if Alg==1
    FT_A1(frFFT,folder,N);
  elseif Alg==2
    FT_A2(frFFT,folder,N);
  else
    FT_A3(frFFT,folder,N);
  end
```

```
  fprintf(frFFT,['\nend arch' fnName ';']);
  fclose(frFFT);
end
```

Direct DFT

```
function FT_A1( frFFT,folder,N )
  fprintf(frFFT,['signal xt_C:std_logic_vector(' ...
    num2str((64*N)-1) ' downto 0);\n']);
  FT_A1_COEF(frFFT,N);
  fprintf(frFFT,'\n');
  SOP_Code_Gen(folder(2:length(folder)),N,1);
  SOP_COMP(frFFT,N,1);
  fprintf(frFFT,'begin\n\n');

  fprintf(frFFT, '\tCOMPLEX_CONVERTION:\n');
  fprintf(frFFT,['\tfor x in 0 to (' num2str(N) '-1) generate\n']);
  fprintf(frFFT, '\t\txt_C((64*(x+1))-1 downto (64*x)) ');
  fprintf(frFFT, '<= xt((32*(x+1))-1 downto (32*x))');
  fprintf(frFFT, ' & \"00000000000000000000000000000000\";\n');
  fprintf(frFFT, '\tend generate COMPLEX_CONVERTION;\n\n');

  fprintf(frFFT,['\tSOPC_GENERATION:\n\tfor k in 0 to (' ...
    num2str(N) '-1) generate']);
  nameRef='\tSOPC_ROWx';
  coef='C(k)';
  Xf='Xf((64*(k+1))-1 downto (64*k))';
  SOP_PM(frFFT,N,nameRef, 'xt_C',coef, Xf,1);
  fprintf(frFFT,'\tend generate SOPC_GENERATION;\n');
end
```

Cooley-Tukey FFT

```
function FT_A2(frFFT,folder,N)
  if N>2
    Alg=['FT'  sprintf('%02d',N) '_A2'];
```

```
xt_Even='xtE';
Xf_Even1='XfE1';
xt_Odd='xtO';
Xf_Odd1='XfO1';
Xf_EO1='XfEO1';
Xf_EO2='XfEO2';
d=FT_A2_COEF(N);
fprintf(frFFT,['signal xt_Stride2:std_logic_vector('...
  num2str((32*N)-1) ' downto 0);\n']);
fprintf(frFFT,['signal ' xt_Even ', ' xt_Odd ':std_logic_vector('...
  num2str((32*(N/2))-1) ' downto 0);\n']);
fprintf(frFFT,['signal ' Xf_Even1  ', ' Xf_Odd1 ':std_logic_vector('...
  num2str((64*(N/2))-1) ' downto 0);\n']);
fprintf(frFFT,['signal ' Xf_EO1 ', ' Xf_EO2 ':std_logic_vector('...
  num2str((64*N)-1) ' downto 0);\n']);
DM_Code_Gen(folder,Alg,1,d,1);
DM_COMP(frFFT,Alg,N,1,1);
PERM_Code_Gen(folder,N,'STn',2,0);
PERM_COMP(frFFT,N,'STn',2,0)
FT_Code_Gen(1,N/2,folder)
FT_COMP(frFFT,N/2,1)
BF_Code_Gen(folder,1);
BF_COMP(frFFT,1);
fprintf(frFFT,'begin\n\n');
PERM_PM(frFFT,N,'STn',2,'Stride2_Permutation','xt','xt_Stride2',0);
% FFT(N/2) EVEN ELEMENTS
fprintf(frFFT,['\n\t' xt_Even '<= xt_Stride2(' ...
  num2str(((N/2)*32)-1) ' downto ' num2str(0) ');\n']);
FT_PM(frFFT,N/2,'FT_Even',xt_Even,Xf_Even1,1);
% FFT(N/2) ODD ELEMENTS
fprintf(frFFT,['\t' xt_Odd '<= xt_Stride2('...
  num2str((N*32)-1) ' downto ' num2str((N/2)*32) ');\n']);
FT_PM(frFFT,N/2,'FT_Odd',xt_Odd,Xf_Odd1,1);
```

```
    % DIAGONAL
    fprintf(frFFT,['\n\t' Xf_EO1 ' <= ' Xf_Odd1 ' & ' Xf_Even1 ';\n']);
    DM_PM(frFFT,'DIAG',Alg,1,Xf_EO1, Xf_EO2,1);
    % BUTTERFLIES
    F_2_ID_N_Code_Gen(frFFT,N/2,Xf_EO2,'Xf',1);
  else %DFT OF 2 POINTS
    fprintf(frFFT,['signal xtC:std_logic_vector(' ...
      num2str((64*N)-1) ' downto 0);\n']);
    fprintf(frFFT,'\n');
    BF_Code_Gen(folder,1);
    BF_COMP(frFFT,1);
    fprintf(frFFT,'begin\n\n');
    for i=1:1:N
      fprintf(frFFT,['\txtC(' num2str((i*64)-1) ' downto '...
        num2str(64*(i-1)) ')']);
      fprintf(frFFT,['\t<= xt(' num2str((i*32)-1) ' downto '...
        num2str(32*(i-1)) ')']);
      fprintf(frFFT,'\t& \"00000000000000000000000000000000\";\n');
    end
    xt0='xtC(63 downto 0)  ';
    xt1='xtC(127 downto 64)';
    Xf0='Xf(63 downto 0)  ';
    Xf1='Xf(127 downto 64)';
    BF_PM(frFFT,'BF_1',xt0,xt1,Xf0,Xf1,1);
  end
end
```

Pease FFT

```
function FT_A3(frFFT,folder,N)
  Alg=['FT'  sprintf('%02d',N) '_A3'];
  n=log2(N);
  fprintf(frFFT,['signal xtC:std_logic_vector(' num2str((64*N)-1) ...
    ' downto 0);\n']);
```

```
FT_A3_COEF(frFFT,n);
for st=1:1:(n);
  fprintf(frFFT,['signal S' num2str(st) '_DM:std_logic_vector((64*' ...
    num2str(N) ')-1 downto 0);\n']);
  fprintf(frFFT,['signal S' num2str(st) '_FN:std_logic_vector((64*' ...
    num2str(N) ')-1 downto 0);\n']);
  fprintf(frFFT,['signal S' num2str(st) '_ST:std_logic_vector((64*' ...
    num2str(N) ')-1 downto 0);\n']);
end

PERM_Code_Gen(folder,N,'BR',0,1);
PERM_COMP(frFFT,N,'BR',0,1);

PERM_Code_Gen(folder,N,'STn',2,1);
PERM_COMP(frFFT,N,'STn',2,1);

for i=1:1:n
  DM_COMP(frFFT,Alg,N,i,1)
  DM_Code_Gen(folder,Alg,i,FT_A3_COEF(i,n),1);
end

ID_N_F_2_COMP(frFFT,N/2,1);
ID_N_F_2_Code_Gen(folder,N/2,1);

fprintf(frFFT,'begin\n\n');
fprintf(frFFT, '\tCOMPLEX_CONVERTION:\n');
fprintf(frFFT,['\t\tfor x in 0 to (' num2str(N) '-1) generate\n']);
fprintf(frFFT, '\t\t\txtC((64*(x+1))-1 downto (64*x)) ');
fprintf(frFFT, '<= xt((32*(x+1))-1 downto (32*x))');
fprintf(frFFT, ' & \"00000000000000000000000000000000\";\n');
fprintf(frFFT, '\t\tend generate COMPLEX_CONVERTION;\n');
% BIT REVERSAL PERMUTATION
PERM_PM(frFFT,N,'BR',0,'BR_PERM','xtC',['S' num2str(n) '_DM'],1)
```

```
  for st=n:-1:1
    % DIAGONAL MATRIX
    DM_PM(frFFT,['DM_' num2str(st)],Alg,st,['S' num2str(st) '_DM'],...
      ['S' num2str(st) '_FN'],1);
    % (Id_(N/2)(X)F_2) MATRIX
    ID_N_F_2_PM(frFFT,['IxF' num2str(st)],N/2,['S' num2str(st) '_FN'],...
      ['S' num2str(st) '_ST'],1);
    % STRIDE-2 PERMUTATION
    if st==1
      vs='Xf';
    else
      vs=['S' num2str(st-1) '_DM'];
    end
    PERM_PM(frFFT,N,'STn',2,['ST_PERM_S' num2str(st)],...
      ['S' num2str(st) '_ST'],vs,1);
  end
end
```

B.2 Discrete Hartley Transform

```
% function HT_Code_Gen(Alg,N,dir)
function HT_Code_Gen(Alg,N,folder)
    vcode=verifyParameters(Alg,N);
    if vcode==-1
        disp('Invalid parameters');
        return;
    end
%     folder=[dir '/HT_RESULTS/'];
%     mkdir(folder(2:length(folder)));
    fnName=['HT'  sprintf('%02d',N) '_A' num2str(Alg)];
    fnFHT=[folder fnName '.vhd'];
    frFHT=fopen(fnFHT(2:length(fnFHT)),'wt');
%     LIB
    fprintf(frFHT,'library IEEE;\n');
```

```
    fprintf(frFHT,'use IEEE.std_logic_1164.all;\n\n');
%     ENTITY
     HT_entity(frFHT,N,fnName);
%     ARCHITECTURE
    fprintf(frFHT,['architecture arch' fnName ' of ' fnName ' is\n\n']);
%     SIGNALS & COMPONENTS
    if Alg==1
        HT_A1(frFHT,folder,N);
    elseif Alg==2
        HT_A2(frFHT,folder,N);
    else
        HT_A3(frFHT,folder,N);
    end
    fprintf(frFHT,['\nend arch' fnName ';']);
    fclose(frFHT);
end
```

Direct DHT

```
function HT_A1( frFHT,folder,N )
  HT_A1_COEF(frFHT,N);
  fprintf(frFHT,'\n');
  SOP_Code_Gen(folder(2:length(folder)),N,0);
  SOP_COMP(frFHT,N,0);
  fprintf(frFHT,'begin\n\n');
  for k=1:1:N
    nameRef=['SOP_ROW_' num2str(k)];
    coef=['C' sprintf('%02d',k)];
    Xf=['X_f(' num2str((k*32)-1) ' downto ' num2str((k-1)*32) ')'];
    SOP_PM(frFHT,N,nameRef, 'x_t',coef, Xf,0);
    SOP_Code_Gen(folder(2:length(folder)),N,0);
  end
end
```

Bracewell FHT

```
function HT_A2( frFHT,folder,N )
  fprintf(frFHT,['signal ST_P:std_logic_vector(' ...
    num2str((32*N)-1) ' downto 0);\n']);
  fprintf(frFHT,['signal xt_E, xt_O:std_logic_vector(' ...
        num2str((32*N/2)-1) ' downto 0);\n']);
  fprintf(frFHT,['signal DIT_E, DIT_O:std_logic_vector(' ...
        num2str((32*N/2)-1) ' downto 0);\n']);
  fprintf(frFHT,['signal DIT:std_logic_vector(' ...
    num2str((32*N)-1) ' downto 0);\n']);
  DEC='DIT';
  TorF=0;
  BF_Code_Gen(folder,0);
  BF_COMP(frFHT,0);
  PERM_COMP(frFHT,N,'STn',2,0);
  PERM_Code_Gen(folder,N,'STn',2,0);
  %EVEN
  DIT_E=['HT'  sprintf('%02d',N/2) '_' DEC '_E'];
  HT_COMP(frFHT,N/2,DIT_E);
  HT_DIT_DIF_Code_Gen(N/2,folder,0,TorF);
  %ODD
  DIT_O=['HT'  sprintf('%02d',N/2) '_' DEC '_O'];
  HT_COMP(frFHT,N/2,DIT_O);
  HT_DIT_DIF_Code_Gen(N/2,folder,1,TorF);
  fprintf(frFHT,'begin\n');
  PERM_PM(frFHT,N,'STn',2,'STRIDE_2','x_t','ST_P',0);
  fprintf(frFHT,['\n\txt_E<=ST_P((32*' ...
    num2str(N/2) ')-1 downto 0);\n']);
  fprintf(frFHT,['\n\txt_O<=ST_P((32*' ...
    num2str(N) ')-1 downto (32*' num2str(N/2) '));\n']);
  HT_PM(frFHT,DIT_E,'HT_DIT_EVEN','xt_E','DIT_E');
  HT_PM(frFHT,DIT_O,'HT_DIT_ODD','xt_O','DIT_O');
  fprintf(frFHT,'\n\tDIT<= DIT_O & DIT_E;\n\n');
  F_2_ID_N_Code_Gen(frFHT,N/2,'DIT','X_f',0);
```

```
end
```

Hou FHT

```
function HT_A3( frFHT,folder,N )
  fprintf(frFHT,['signal F2_ID_N:std_logic_vector(' ...
        num2str((32*N)-1) ' downto 0);\n']);
  fprintf(frFHT,['signal xt_E, xt_O:std_logic_vector(' ...
        num2str((32*N/2)-1) ' downto 0);\n']);
  fprintf(frFHT,['signal DIF_E, DIF_O:std_logic_vector(' ...
        num2str((32*N/2)-1) ' downto 0);\n']);
  fprintf(frFHT,['signal DIF:std_logic_vector(' ...
    num2str((32*N)-1) ' downto 0);\n']);
  BF_Code_Gen(folder,0);
  BF_COMP(frFHT,0);
  DEC='DIF';
  TorF=1;
  PERM_COMP(frFHT,N,'STn',N/2,0);
  PERM_Code_Gen(folder,N,'STn',N/2,0);
  %EVEN
  DIF_E=['HT'  sprintf('%02d',N/2) '_' DEC '_E'];
  HT_COMP(frFHT,N/2,DIF_E);
  HT_DIT_DIF_Code_Gen(N/2,folder,0,TorF);
  %ODD
  DIF_O=['HT'  sprintf('%02d',N/2) '_' DEC '_O'];
  HT_COMP(frFHT,N/2,DIF_O);
  HT_DIT_DIF_Code_Gen(N/2,folder,1,TorF);
  fprintf(frFHT,'begin\n\n');
  F_2_ID_N_Code_Gen(frFHT,N/2,'x_t','F2_ID_N',0);
  fprintf(frFHT,['\n\txt_E<=F2_ID_N((32*' ...
    num2str(N/2) ')-1 downto 0);\n']);
  fprintf(frFHT,['\n\txt_O<=F2_ID_N((32*' ...
    num2str(N) ')-1 downto (32*' num2str(N/2) '));\n']);
  HT_PM(frFHT,DIF_E,'HT_DIT_EVEN','xt_E','DIF_E');
```

```
  HT_PM(frFHT,DIF_O,'HT_DIT_ODD','xt_O','DIF_O');
  fprintf(frFHT,'\n\tDIF<= DIF_O & DIF_E;\n');
  PERM_PM(frFHT,N,'STn',N/2,['STRIDE_' num2str(N/2)],'DIF','X_f',0);
end
```

B.3 Discrete Cosine Transform

```
% function CT_Code_Gen(Alg,N,dir,bar)
function CT_Code_Gen(Alg,N,folder,bar)
    vcode=verifyParameters(Alg,N);
    if vcode==-1
        disp('Invalid parameters');
        return;
    end
%     folder=[dir '/CT_RESULTS/'];
%     mkdir(folder(2:length(folder)));
    if bar==1 || Alg==4
        fnName=['CT'  sprintf('%02d',N) '_A' num2str(Alg) '_bar'];
    else
        fnName=['CT'  sprintf('%02d',N) '_A' num2str(Alg)];
    end
    fnFCT=[folder fnName '.vhd'];
    frFCT=fopen(fnFCT(2:length(fnFCT)),'wt');
    %VHDL LIBRARIES
    fprintf(frFCT,'library IEEE;\n');
    fprintf(frFCT,'use IEEE.std_logic_1164.all;\n\n');
    % STRUCTURE ENTITY
    CT_entity(frFCT,N,fnName);
    % STRUCTURE ARCHITECTURE
    fprintf(frFCT,['architecture arch' fnName ' of ' fnName ' is\n\n']);
    if Alg==1
        if bar==0
            CT_A1(frFCT,folder,N);
        else
```

```
            CT_A1_bar(frFCT,folder,N);
        end
    elseif Alg==2
        CT_A2(frFCT,folder,N);
    elseif Alg==3
        CT_A3(frFCT,folder,N);
    elseif Alg==4
        CT_IV_bar(frFCT,folder,N);
    end
    fprintf(frFCT,['\nend arch' fnName ';']);
    fclose(frFCT);
end
```

Direct DCT

```
function CT_A1(frFCT,folder,N)
  CT_A1_COEF(frFCT,N);
  cons=single((2/N)^(1/2));
  cons_bin=single2bin(cons);
  CONS='CONS';
  fprintf(frFCT,['signal ' CONS ':std_logic_vector(' ...
    num2str(32) '-1 downto 0):= \"' cons_bin '\";\n']);
  fprintf(frFCT,['signal SOPs:std_logic_vector((32*' ...
    num2str(N) ')-1 downto 0);\n']);
  fprintf(frFCT,'\n');
  FlPM_COMP(frFCT);
  SOP_Code_Gen(folder(2:length(folder)),N,0);
  SOP_COMP(frFCT,N,0);
  fprintf(frFCT,'begin\n\n');
  fprintf(frFCT,['\tSOP_GENERATION:\n\tfor k in 0 to (' ...
    num2str(N) '-1) generate\n']);
  coef='C(k)';
  SOP='SOPs((32*(k+1))-1 downto (32*k))';
  Xf='X_f((32*(k+1))-1 downto (32*k))';
```

```
  CON='CONS                ';
  SOP_PM(frFCT,N,'\tSOP_ROWx', 'x_t ',coef, SOP,0);
  FlPM_PM(frFCT,'\tFlPMx',SOP,CON,Xf);
  fprintf(frFCT,'\tend generate SOP_GENERATION;\n');
end
```

Nikara FCT

```
function CT_A2(frFCT,folder,N)
  Alg=['CT'  sprintf('%02d',N) '_A2'];
  n=log2(N);
  % FACTOR
  cons=single((2/N)^(1/2));
  cons_bin=single2bin(cons);
  CONS='CONS';
  fprintf(frFCT,['signal ' CONS ':std_logic_vector(' ...
    num2str(32) '-1 downto 0):= \"' cons_bin '\";\n']);
  % SIGNALS
  fprintf(frFCT,['signal HaP:std_logic_vector((32*' ...
    num2str(N) ')-1 downto 0);\n']);
  % A(ST)
  % [I_{N/2} (X) F_{2}]
  for st=0:1:(n-1);
    fprintf(frFCT,['signal S' num2str(st) '_FN:std_logic_vector((32*'...
      num2str(N) ')-1 downto 0);\n']);
  end
  % [H_N^ST]
  for st=0:1:(n-1);
    fprintf(frFCT,['signal S' num2str(st) '_HP:std_logic_vector((32*'...
      num2str(N) ')-1 downto 0);\n']);
  end
  % [D_N^ST]
  for st=0:1:(n-1);
    fprintf(frFCT,['signal S' num2str(st) '_Di:std_logic_vector((32*'...
```

```
    num2str(N) ')-1 downto 0);\n']);
end
% [M_N^ST]
for st=0:1:(n-1);
  fprintf(frFCT,['signal S' num2str(st) '_MN:std_logic_vector((32*'...
    num2str(N) ')-1 downto 0);\n']);
end
% [I_{N/2^(S+1)} (X) P^{T}_{2^(S+1),2}]
for st=1:1:(n-1);
  fprintf(frFCT,['signal S' num2str(st) '_STP:std_logic_vector((32*' ...
    num2str(N) ')-1 downto 0);\n']);
end
% U
fprintf(frFCT,['signal UP:std_logic_vector((32*' ...
  num2str(N) ')-1 downto 0);\n']);
% COMPONENTS
FlPM_COMP(frFCT);
% HADAMARD PERM
PERM_Code_Gen(folder,N,'Ha',0,0);
PERM_COMP(frFCT,N,'Ha',0,0);
%A(s)
% [I_{N/2} (X) F_{2}]
ID_N_F_2_Code_Gen(folder,N/2,0);
ID_N_F_2_COMP(frFCT,N/2,0);
% [H_N^ST]
for st=0:1:(n-1);
  PERM_Code_Gen(folder,N,'H',st,0);
  PERM_COMP(frFCT,N,'H',st,0);
end
% [D_N^ST]
for st=0:1:(n-1);
  DM_Code_Gen(folder,Alg,st,CT_A2_COEF(st,N),0);
  DM_COMP(frFCT,Alg,N,st,0);
```

```
end
% [M_N^ST]
for st=0:1:(n-1);
  CT_A2_MN_Code_Gen(folder,N,st);
  CT_A2_MN_COMP(frFCT,N,st);
end
% [I_{N/2^(S+1)} (X) P^{T}_{2^(S+1),2}]
for st=1:1:(n-1);
  PERM_Code_Gen(folder,N,'S',st,0);
  PERM_COMP(frFCT,N,'S',st,0);
end
% U
PERM_Code_Gen(folder,N,'U',0,0);
PERM_COMP(frFCT,N,'U',0,0);
% BEGIN
fprintf(frFCT,'begin\n\n');
st=0;
STi_HaP='HaP';
STi_FN=['S' num2str(st) '_FN'];
STi_HP=['S' num2str(st) '_HP'];
STi_Di=['S' num2str(st) '_Di'];
STi_MN1=['S' num2str(st) '_MN'];
%HADAMARD PERM
PERM_PM(frFCT,N,'Ha',0,'HadamardPerm','x_t',STi_HaP,0);
% A(0)
ID_N_F_2_PM(frFCT,['IxF_S' num2str(st)],N/2,STi_HaP,STi_FN,0);
PERM_PM(frFCT,N,'H',st,['HP_S' num2str(st)],STi_FN,STi_HP,0);
DM_PM(frFCT,['Diag_S' num2str(st)],Alg,st,STi_HP,STi_Di,0);
CT_A2_MN_PM(frFCT,['MN_S' num2str(st)],N,st,STi_Di,STi_MN1);
for st=1:1:(n-1);
  STi_MN0=['S' num2str(st-1) '_MN'];
  STi_STP=['S' num2str(st) '_STP'];
  STi_FN=['S' num2str(st) '_FN'];
```

```
    STi_HP=['S' num2str(st) '_HP'];
    STi_Di=['S' num2str(st) '_Di'];
    STi_MN1=['S' num2str(st) '_MN'];
    % [I_{N/2^(S+1)} (X) P^{T}_{2^(S+1),2}]
    PERM_PM(frFCT,N,'S',st,['SP_S' num2str(st)],STi_MN0,STi_STP,0);
    % A(st)
    ID_N_F_2_PM(frFCT,['IxF_S' num2str(st)],N/2,STi_STP,STi_FN,0);
    PERM_PM(frFCT,N,'H',st,['HP_S' num2str(st)],STi_FN,STi_HP,0);
    DM_PM(frFCT,['Diag_S' num2str(st)],Alg,st,STi_HP,STi_Di,0);
    CT_A2_MN_PM(frFCT,['MN_S' num2str(st)],N,st,STi_Di,STi_MN1);
  end
  % U
  PERM_PM(frFCT,N,'U',0,'UPerm',STi_MN1,'UP',0);
  %FACTOR MULTIPLICATION
  fprintf(frFCT,['\tFACTOR_MULT:\n\tfor k in 0 to ' ...
    num2str(N) '-1 generate']);
  U='UP((32*(k+1))-1 downto (32*k))';
  CON='CONS              ';
  Xf='X_f((32*(k+1))-1 downto (32*k))';
  FlPM_PM(frFCT,'\tFlPMx',U,CON,Xf);
  fprintf(frFCT,'\tend generate FACTOR_MULT;\n');
end
```

Translation FCT

```
function CT_A3(frFCT,folder,N)
  Alg=['CT'  sprintf('%02d',N) '_A3'];
  xt='x_t';
  IIbP='IIbPerm';
  F2IN='F2IN';
  F2OUT='F2OUT';
  STnP='STnPerm';
  Xf='X_f';
  d=CT_A3_COEF(N);
```

```
  fprintf(frFCT,['signal ' IIbP ', ' F2IN ', ' F2OUT ', ' ...
    STnP ':std_logic_vector(' num2str(32*N) '-1 downto 0);\n']);
  PERM_Code_Gen(folder,N,'IIb',0,0);
  PERM_COMP(frFCT,N,'IIb',0,0);
  CT_Code_Gen(1,N/2,folder,1);
  CT_COMP(frFCT,1,N/2,1);
  CT_Code_Gen(4,N/2,folder,1);
  CT_COMP(frFCT,4,N/2,1);
  PERM_Code_Gen(folder,N,'STn',N/2,0);
  PERM_COMP(frFCT,N,'STn',N/2,0);
  DM_Code_Gen(folder,Alg,0,d,0);
  DM_COMP(frFCT,Alg,N,0,0);
  BF_Code_Gen(folder,0);
  BF_COMP(frFCT,0);
  %IIb PERM
  fprintf(frFCT,'begin\n\n');
  PERM_PM(frFCT,N,'IIb',0,'IIb_PERM',xt,IIbP,0);
  F_2_ID_N_Code_Gen(frFCT,N/2,IIbP,F2IN,0);
  CT_II_IN=[F2IN '((32*' num2str(N/2) ')-1 downto 0)'];
  CT_II_OUT=[F2OUT '((32*' num2str(N/2) ')-1 downto 0)'];
  CT_IV_IN=[F2IN '((32*' num2str(N) ')-1 downto 32*'...
    num2str(N/2) ')'];
  CT_IV_OUT=[F2OUT '((32*' num2str(N) ')-1 downto 32*'...
    num2str(N/2) ')'];
  CT_PM(frFCT,N/2,'CT_II_bar', CT_II_IN,CT_II_OUT,1,1);
  CT_PM(frFCT,N/2,'CT_IV_bar', CT_IV_IN,CT_IV_OUT,4,1);
  PERM_PM(frFCT,N,'STn',N/2,'STn_PERM',F2OUT,STnP,0);
  DM_PM(frFCT,'DIAG',Alg,0,STnP,Xf,0);
end
```

Appendix C
Hardware Performance Details

C.1 Discrete Fourier Transform

Table C.1 : DFT Structures FPGA Resource Consumption

N-point	FPGA Resources						
	Structure	LUTs Slices	Registers Slices	DSP Slices	8-to-1 Muxes	16-to-1 Muxes	GCB
2	Direct DFT	17,730	85	10	98	0	1
	Cooley-Tukey FFT	3,203	0	0	19	1	0
	Pease FFT	3,203	0	0	19	1	0
4	Direct DFT	85,223	425	50	498	0	1
	Cooley-Tukey FFT	50,343	238	28	290	0	1
	Pease FFT	21,621	68	8	144	2	1
8	Direct DFT	372,414	1,921	226	2,234	0	1
	Cooley-Tukey FFT	205,550	1,054	124	1,222	0	1
	Pease FFT	78,890	340	40	516	4	1
16	Direct DFT	1,555,672	8,177	962	9,450	0	1
	Cooley-Tukey FFT	819,422	4,318	508	4,958	0	1
	Pease FFT	234,737	1,156	136	1,524	8	1
32	Direct DFT	6,711,878	65,489	1,922	38,858	0	1
	Cooley-Tukey FFT	3,259,226	17,374	2,044	19,918	0	1
	Pease FFT	629,456	3,332	392	4,068	16	1

Table C.2 : DFT Structures FPGA Latency

N-point	Latency Type			
	Structure	Total	Logic	Route
2	Direct DFT	44.984 ns	13.061 ns	31.922 ns
	Cooley-Tukey FFT	20.263 ns	7.946 ns	12.317 ns
	Pease FFT	20.263 ns	7.946 ns	12.317 ns
4	Direct DFT	77.710 ns	22.116 ns	55.593 ns
	Cooley-Tukey FFT	84.144 ns	24.002 ns	60.141 ns
	Pease FFT	59.524 ns	18.502 ns	41.022 ns
8	Direct DFT	142.996 ns	40.663 ns	102.332 ns
	Cooley-Tukey FFT	116.966 ns	33.438 ns	83.527 ns
	Pease FFT	98.530 ns	29.245 ns	69.285 ns

C.2 Discrete Hartley Transform

Table C.3 : DFT Structures FPGA Resource Consumption

N-point	FPGA Resources						
	Structure	LUTs Slices	Registers Slices	DSP Slices	8-to-1 Muxes	16-to-1 Muxes	GCB
2	Direct DHT	6,667	192	0	51	2	0
	Bracewell FHT	3,256	50	0	30	2	0
	Hou FHT	3,275	50	0	24	1	0
4	Direct DHT	31,815	768	0	194	10	0
	Bracewell FHT	18,201	384	0	136	4	0
	Hou FHT	18,058	384	0	127	6	0
8	Direct DHT	133,597	2,976	0	766	0	0
	Bracewell FHT	70,339	1,440	0	406	0	0
	Hou FHT	69,994	1,440	0	406	0	0
16	Direct DHT	568,256	12,096	0	3,196	0	0
	Bracewell FHT	291,057	5,952	0	1,644	0	0
	Hou FHT	290,559	5,952	0	1,644	0	0
32	Direct DHT	2,349,527	48,864	0	13,047	0	0
	Bracewell FHT	1,188,904	24,288	0	6,626	0	0
	Hou FHT	1,189,841	24,288	0	6,626	0	0

Table C.4 : DFT Structures FPGA Latency

N-point	Latency Type			
	Structure	Total	Logic	Route
2	Direct DHT	30.637 ns	9.214 ns	21.422 ns
	Bracewell FHT	28.924 ns	9.233 ns	19.690 ns
	Hou FHT	27.903 ns	9.300 ns	18.603 ns
4	Direct DHT	66.237 ns	18.046 ns	48.190 ns
	Bracewell FHT	47.128 ns	13.607 ns	33.520 ns
	Hou FHT	44.175 ns	13.415 ns	30.760 ns
8	Direct DHT	126.058 ns	35.470 ns	90.587 ns
	Bracewell FHT	77.341 ns	22.387 ns	54.953 ns
	Hou FHT	76.238 ns	23.637 ns	52.601 ns

C.3 Discrete Cosine Transform

Table C.5 : DFT Structures FPGA Resource Consumption

N-point	FPGA Resources						
	Structure	LUTs Slices	Registers Slices	DSP Slices	8-to-1 Muxes	16-to-1 Muxes	GCB
2	Direct DCT	7,534	102	12	59	2	1
	Nikara FCT	5,808	68	8	44	0	1
	Translation FCT	5,006	51	6	44	1	1
4	Direct DCT	31,473	340	40	238	10	1
	Nikara FCT	18,078	153	18	132	0	1
	Translation FCT	19,616	204	24	147	6	1
8	Direct DCT	125,408	1,224	144	824	0	1
	Nikara FCT	51,407	357	42	329	0	1
	Translation FCT	70,732	680	80	464	0	1
16	Direct DCT	507,035	4,624	544	3,312	0	1
	Nikara FCT	135,511	833	98	861	0	1
	Translation FCT	269,508	2,448	288	1,760	0	1
32	Direct DCT	2,038,931	17,952	2,112	13,280	0	1
	Nikara FCT	338,557	1,921	226	2,141	0	1
	Translation FCT	1,051,283	9,248	1,088	6,848	0	1

Table C.6 : DFT Structures FPGA Latency

N-point	Latency Type			
	Structure	Total	Logic	Route
2	Direct DCT	36.800 ns	10.650 ns	26.150 ns
	Nikara FCT	33.767 ns	9.805 ns	23.962 ns
	Translation FCT	35.139 ns	10.234 ns	24.905 ns
4	Direct DCT	71.646 ns	20.112 ns	51.534 ns
	Nikara FCT	71.979 ns	21.806 ns	50.173 ns
	Translation FCT	49.851 ns	14.981 ns	34.870 ns
8	Direct DCT	132.218 ns	37.496 ns	94.721 ns
	Nikara FCT	111.016 ns	32.616 ns	78.400 ns
	Translation FCT	82.098 ns	24.843 ns	57.255 ns