Removing Redundancies of Fast Fourier Transform Computations

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Yingjie Lao

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Master of Science in Electrical Engineering

Advisor: Keshab K. Parhi

July, 2015

Acknowledgements

I would like to acknowledge the support of my adviser, Prof. Keshab K. Parhi in guiding and providing the necessary direction throughout this work.

I would like to acknowledge Manohar Ayinala and Megha Parhi for providing the help and contribution to this work, and also all those people who directly and indirectly have helped me in pursuing this work.

Abstract

The Fast Fourier Transform (FFT) is widely used in various digital signal processing (DSP) applications. In this thesis, we are exploring this problem from a fundamental arithmetic stance: maintain the degree of freedom at all stages to be the minimum by completely eliminating arithmetic redundancies. The proposed approach is canonic with respect to the number of signals (i.e., data-canonic) at the each stage.

First, we present novel decimation-in-time (DIT) and decimation-in-frequency (DIF) structures for computing RFFT that are canonic with respect to the number signal values computed at each FFT stage. In the proposed structure, in an N-point RFFT, exactly N signal values are computed at the output of each FFT stage and at the output. One of our contribution is in showing that the twiddle factors can be transformed to reduce the computational complexity.

Next, we present a novel algorithm to compute real-valued fast Fourier transform (RFFT) that is canonic with respect to the number of signal values. A signal value can correspond to a purely real or purely imaginary value, while a complex signal consists of 2 signal values. In order to reduce the redundant samples, a *sample removal* lemma, and two types of *twiddle factor transformations* are proposed: *pushing* and *modulation*. We consider 4 different cases for an $N = P \times Q$ point canonic RFFT: 1) P is odd, Q is odd; 2) P is odd, Q is even; 3) P is even, Q is odd; and 4) P is even, Q is even. No *twiddle factor transformation* is required when P is odd. It is shown that the number of twiddle factors can be reduced by performing *modulation transformation* when P is even and Q is odd, while 2 real twiddle factor operations are *pushed* to 1 complex twiddle factor operation when P and Q are both even. Canonic RFFT for any composite length can be computed by applying the proposed algorithm recursively. The major advantage of the canonic RFFTs is that they require the least butterfly operations and only involve real datapath when mapped to architectures.

Furthermore, we consider the FFT computation whose inputs are not only real but also even/odd symmetric. Novel algorithms for generating the flow-graphs of canonic RFFTs with even/odd symmetric signals are proposed. It is shown that by performing the proposed algorithms, the resulted canonic structure will have $\frac{N}{2}+1$ signal values at each stage for an N-point RFFT with even symmetric signals (REFFT), or $\frac{N}{2}-1$ signal values at each stage for an N-point RFFT with odd symmetric signals (ROFFT). In order to remove butterfly operations, twiddle factor operation may also be needed to pull the twiddle factors to previous stages. We also discuss the design of canonic REFFT for any composite length. Performances of the canonic REFFT/ROFFT are also discussed. It is shown that the flow-graph of canonic REFFT/ROFFT has less number of interconnections, less butterfly operations, and less twiddle factor operations, compared to prior works.

Finally, we present a novel *scalable* architecture for in-place FFT/IFFT computation for real-valued signals. The proposed computation is based on a modified radix-2 algorithm, which removes the redundant operations from the flow graph. A new processing element is proposed using two radix-2 butterflies which can process four inputs in parallel. A novel *conflict-free* memory addressing scheme is proposed to ensure the continuous operation of the FFT processor. Furthermore, the addressing scheme is extended to support multiple parallel processing elements (PEs). The proposed real-valued FFT processor simultaneously requires fewer computation cycles and lower hardware cost compared to prior work. For example, the proposed design with 2 PEs reduces the computation cycles by a factor of 2 for a 256-point RFFT compared to a prior work, while maintaining a lower hardware complexity. The number of computation cycles is reduced proportionately with increase in the number of PEs.

Contents

Acknowledgements **i**

Abstract **ii**

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**

1.1 Introduction 1

2 Canonic Real-Valued FFT Structures **4**

2.1 Introduction 4

2.2 RFFT 5

2.3 Canonic DIT RFFT 7

2.3.1 4-point RFFT 7

2.3.2 8-point RFFT 7

2.3.3 16-point RFFT 9

2.4 Algorithm for Canonic DIT RFFT Computation 10

2.4.1 Divide and Conquer 10

2.4.2 Generalization to $N = 2^n$-point DIT RFFT 11

2.5 Algorithm for Canonic DIF RFFT Computation 12

2.6 Performance Comparison 15
2.7 Conclusion 18

3 Data-Canonic Real FFT Flow-Graphs for Composite Lengths 19
3.1 Introduction 19
3.2 Canonic RFFT 20
3.3 Twiddle Factor Transformations 23
3.3.1 Sample Removal 23
3.3.2 Twiddle Factor Transformation Type I: Pushing 24
3.3.3 Twiddle Factor Transformation Type II: Modulation 25
3.4 Canonic Computations for Composite Size RFFTs 26
3.4.1 $N = P \times Q$, P is odd, Q is odd 27
3.4.2 $N = P \times Q$, P is odd, Q is even 29
3.4.3 $N = P \times Q$, P is even, Q is odd 31
3.4.4 $N = P \times Q$, P is even, Q is even 33
3.4.5 Proposed Algorithm to Generate Canonic RFFTs 36
3.5 Performance 36
3.6 Conclusion 40

4 Canonic Real-Valued FFT Flow-Graphs with Even/Odd Symmetric Signals 41
4.1 Introduction 41
4.2 Background 42
4.2.1 Canonic RFFT 42
4.2.2 REFFT/ROFFT 44
4.3 Canonic REFFT for Power-Of-Two Length 45
4.3.1 4-Point REFFT 45
4.3.2 8-Point REFFT 46
4.3.3 16-Point REFFT 48

4.3.4 Generalization to $N = 2^n$-Point DIF REFFT 51
4.4 Pre-Processing 54
4.4.1 Canonic Property 54
4.4.2 Pull the Twiddle Factors 56
4.5 Canonic ROFFT for Power-Of-Two Length 57
4.6 REFFT for Any Composite Length 59
4.6.1 Subcomponents 60
4.6.2 $N = P \times Q$, P is odd, Q is odd 64
4.6.3 $N = P \times Q$, P is odd, Q is even 69
4.6.4 $N = P \times Q$, P is even, Q is odd 69
4.6.5 $N = P \times Q$, P is even, Q is even 71
4.6.6 Summary 76
4.7 Performance 78
4.8 Conclusion 79

5 An In-Place FFT Architecture for Real-Valued Signals 80
5.1 Introduction 80
5.2 Proposed Architecture 81
5.2.1 Address Generation Scheme 83
5.2.2 Memory Bank Assignment 86
5.2.3 PE Configuration Unit 87
5.2.4 Generalization to Multiple Processing Elements 88
5.3 Hermitian-symmetric IFFT processor 90
5.4 Comparisons 92
5.5 Conclusion 93

6 Conclusion and Future Directions 94

References 96

List of Tables

2.1 Bottom Input Positions of the Removed Butterflies 11

2.2 Canonic DIT RFFT Output Signals 12

2.3 Comparison of Twiddle Factor Operation and Datapath for $N = 2^n$-point RFFT 17

3.1 Number of Twiddle Factor Operations 37

3.2 Normalized Cost of Different Types of Twiddle Factor Operations (16-bit) . . 38

4.1 Signal Values of the Bottom Half in Figure 4.10 49

4.2 Signal Values of the Bottom Half in Figure 4.15 54

4.3 Inputs of Each P-point RFFT at the First Stage 60

4.4 FFT types for canonic REFFT 77

4.5 Performance Comparison for $N = 2^n$-point FFT Algorithms 79

5.1 Address Patterns for N-point RFFT Computation ($N = 2^n$) 86

5.2 Memory Bank Assignment for 32-point RFFT with Single PE 87

5.3 Memory Bank Assignment for 2^n-point RFFT with L PEs 90

5.4 Memory Bank Assignment for 32-point RFFT with 2 PEs 90

5.5 Comparison of the RFFT processors 93

List of Figures

2.1 Flow-graph of a 16-point DIF FFT. Input signals are assumed to be complex. . 6

2.2 Flow-graph of a 16-point DIF real-valued FFT (RFFT). Input signals are purely real and redundant signals in the shaded regions are removed. 7

2.3 Flow-graph of a canonic 4-point DIT RFFT. 8

2.4 Flow-graph of a canonic 8-point DIT RFFT. 8

2.5 Flow-graph of a canonic 16-point DIT RFFT. 9

2.6 Remove butterfly and separate real part and imaginary part for DIT RFFT. . 11

2.7 Flow-graph of a canonic 32-point DIT RFFT. 13

2.8 Alternate flow-graph of a 16-point DIF FFT. 14

2.9 Remove butterfly and separate real part and imaginary part for DIF RFFT. . 15

2.10 Flow-graph of a canonic 16-point DIF RFFT. 16

3.1 Number of signals is increased by twiddle factor operations. 21

3.2 A canonic 16-point RFFT. 22

3.3 "Pushing" the twiddle factors. 22

3.4 M-point RFFT: (a) M is odd, (b) M is even. 24

3.5 Twiddle factor transformation type I: pushing. 25

3.6 Modulation property of a 3-point FFT. (a) The original 3-point FFT, (b) output shifted by $k_0 = 1$, (c) output shifted by $k_0 = 2$. 26

3.7 Twiddle factor transformation type II: modulation. (a) The original N-point FFT, (b) compute the shifted FFT, $k_0 = \frac{N}{2} + a$, (c) the final N-point RFFT. . 26

3.8 A 15-point canonic RFFT, where $P = 3, Q = 5$. 27
3.9 A 15-point canonic RFFT, where $P = 5, Q = 3$. 28
3.10 Flow-graph of a 6-point DIF FFT. 29
3.11 Flow-graph of a 3-point FFT. 30
3.12 Flow-graph of a 3-point RFFT. 30
3.13 Flow-graph of the canonic 6-point RFFT for the structure shown in Figure 3.10. 30
3.14 Operation to the Q-point FFT whose inputs are from the second real outputs of these P-point RFFTs after going through nontrivial twiddle factors. 31
3.15 Flow-graph of a 6-point DIF FFT. 32
3.16 Compute the shifted FFT for the bottom 3-point FFT at the second stage of Figure 3.15. The 3-point FFT can be reduced to RFFT (as shown in Figure 3.12) after modulation transformation. 33
3.17 Flow-graph of the canonic 6-point RFFT for the structure shown in Figure 3.15. 33
3.18 A Q-point FFT is considered as a $(2 \times \frac{Q}{2})$-point FFT. 34
3.19 The $(2 \times \frac{Q}{2})$-point RFFT after eliminating redundancy. 35
3.20 A 16-point canonic RFFT, where $P = 4, Q = 4$. 37
4.1 Obtain $y[n]$ by computing the IFFT of $X[k]H[k]$. 42
4.2 Twiddle factor transformation: push. 43
4.3 A canonic 16-point RFFT. 43
4.4 Flow-graph of a canonic 4-point RFFT. 45
4.5 Operation to the butterfly with two identical inputs. 46
4.6 Flow-graph of a canonic 4-point REFFT. 46
4.7 Flow-graph of a canonic 8-point RFFT. 47
4.8 Butterfly simplification. 47
4.9 Flow-graph of a canonic 8-point REFFT. 47
4.10 Flow-graph of a canonic 16-point RFFT. 48
4.11 Operation to the butterfly with two inputs have opposite values. 50
4.12 Flow-graph of a canonic 16-point REFFT. 50

4.13 Flow-graph of a canonic 32-point RFFT. 52
4.14 Flow-graph of a canonic 32-point REFFT. 53
4.15 Flow-graph of a canonic 16-point DIT RFFT. 55
4.16 Twiddle factor transformation: pull. 56
4.17 The two twiddle factors after the $n-1$th stage at the bottom half will still have the same pattern even if the twiddle factors have already been transformed. . . 57
4.18 Flow-graph of a canonic 4-point ROFFT. 58
4.19 Flow-graph of a canonic 16-point ROFFT. 59
4.20 M-point REFFT: (a) M is odd, (b) M is even. 60
4.21 (a) The original odd size P-point FFT whose inputs have the pattern of $x_P[k] = x_P[P-1-k]$, (b) compute the shifted FFT, $n_0 = \frac{P-1}{2}Q$, (c) the final P-point RFFT whose inputs are even symmetric. 63
4.22 (a) The original even size P-point FFT whose inputs have the pattern of $x_P[k] = x_P[P-1-k]$, (b) the P-point RFFT is considered as a $\frac{P}{2} \times 2$-point RFFT, (c) pull the twiddle factors, (d) the final structure which only has $\frac{P}{2}$ signals. . . . 64
4.23 A 15-point canonic RFFT, where $P = 3, Q = 5$. 66
4.24 A 15-point canonic RFFT, where $P = 5, Q = 3$. 67
4.25 A 15-point canonic REFFT, where $P = 3, Q = 5$. 68
4.26 A 15-point canonic REFFT, where $P = 5, Q = 3$. 68
4.27 Flow-graph of a canonic 6-point RFFT, where $P = 3, Q = 2$. 70
4.28 Flow-graph of a canonic 6-point REFFT, where $P = 3, Q = 2$. 70
4.29 Flow-graph of a canonic 6-point RFFT, where $P = 2, Q = 3$. 72
4.30 Flow-graph of a canonic 6-point REFFT, where $P = 2, Q = 3$. 72
4.31 A Q-point FFT is considered as a $(2 \times \frac{Q}{2})$-point FFT. 74
4.32 The $(2 \times \frac{Q}{2})$-point RFFT after eliminating redundancy. 74
4.33 Remove redundancy of the $(2 \times \frac{Q}{2})$-point RFFT, when $\frac{Q}{2}$ is odd. 75
4.34 Remove redundancy of the $(2 \times \frac{Q}{2})$-point RFFT, when $\frac{Q}{2}$ is even. 75
4.35 A 16-point canonic RFFT, where $P = 4, Q = 4$. 76

4.36 A 16-point canonic REFFT, where $P = 4, Q = 4$. 77
5.1 Data flow graph of 32-point FFT for real-valued signals. 82
5.2 Proposed memory-based RFFT architecture. 84
5.3 Processing element for real-valued signals. 84
5.4 2-parallel pipelined architecture for a 16-point FFT computation. 84
5.5 Illustration of proposed addressing scheme for one processing element. The arrows show how the data is read/write into the memory banks during the various stages of the RFFT computation. 85
5.6 PE configuration unit (generating select signals for each stage). 88
5.7 Illustration of proposed addressing scheme for two processing elements. . 89
5.8 Flow graph of the 16-point inverse FFT for Hermitian-symmetric input. 91
5.9 Addressing scheme for the proposed 16-point Hermitian-symmetric IRFFT processor. 92

Chapter 1

Introduction

1.1 Introduction

The computation of the Discrete Fourier Transform (DFT) is an important tool in many branches of science and engineering and has been studied extensively. Research during the past several decades has resulted in various algorithms for speeding up the transformation, notably the Fast Fourier Transform (FFT). The FFT algorithm has long been one of the most important topic in DSP and is widely used in applications such as telecommunications, biomedical signal processing, and spectral analysis. Particularly, FFT plays a critical role in noise reduction, global motion estimation, video broadcasting, and orthogonal frequency division multiplexing (OFDM) systems. Nowadays, there has been an increasing interest in the computation of FFT of real-valued signals, referred as RFFT. This is because most of the physical signals, such as biomedical signals, are real. The real-valued signals exhibit conjugate symmetry giving rise to redundancies. This property can be exploited to reduce both arithmetic and architectural complexities. In this thesis, we consider the implementation of FFT computation for specific applications.

An N-point FFT processes N complex signals to compute N output complex signals using decimation-in-time (DIT) or decimation-in-frequency (DIF) approach. The FFT makes use of $n = log_2 N$ stages of computations where each stage computes N complex signals; N is assumed to be power-of-two. In this thesis, we consider the implementation of FFT computation with a real signal of length N. Since the degrees of freedom of the input data is N, each stage of the FFT should not need to compute more than N signal values, where a signal value can corresponds to a purely real or purely imaginary value. Any more than N samples computed at any FFT stage is inherently redundant. Therefore, we could completely eliminate the arithmetic redundancy to improve the performance of real-valued FFT.

Furthermore, we consider another scenario where the input signals are not only real but also even/odd symmetric. As we know, if the input signals are real and even symmetric, the outputs will be purely real and even symmetric. If the input signals are real and odd symmetric, the outputs will be purely imaginary and odd symmetric. Therefore, in order to completely eliminate redundancy, we need to guarantee there are only $\frac{N}{2} + 1$ signal values at each stage for an N-point REFFT, or $\frac{N}{2} - 1$ signal values at each stage for an N-point ROFFT. As a result, the performance can be improved under this specific application.

Besides from the arithmetic perspective, we also explore to remove the redundancy of the RFFT from the architectural perspective. A novel in-place FFT/IFFT architecture for real-valued signals is presented in this thesis, which could achieve lower hardware complexity as well as computation cycles.

The thesis is organized in the following way.

- The backgrounds of FFT and RFFT are introduced in Chapter 2. Afterwards, we introduce the concept of canonic RFFT. The key idea to make sure that the arithmetic redundancy in the RFFT computation flow-graph is completely eliminated. Then we present novel DIT and DIF structures for computing RFFT with power-of-two size.

- Chapter 3 further presents a novel algorithm to design canonic RFFT computations for any composite length. A sample removal lemma, and two types of twiddle factor transformations are proposed.

- In Chapter 4, we consider the FFT computation whose inputs are not only real but also even/odd symmetric. We present novel algorithms for generating the flow-graphs of canonic RFFTs with even/odd symmetric signals.

- Chapter 5 introduces the a novel scalable in-place RFFT architecture which simultaneously requires fewer computation cycles and lower hardware cost.

- Finally, Chapter 6 concludes with a summary of total contributions of this thesis and future research directions.

Chapter 2

Canonic Real-Valued FFT Structures

2.1 Introduction

A number of RFFT computation algorithms and implementations have been proposed for both pipelined and in-place architectures in the literature [1, 2, 3, 4]. An approach to computing an N-point RFFT using an $\frac{N}{2}$-point complex FFT was presented in [1]. However, this approach requires significant amount of post-processing. Custom pipelined architectures for RFFT have been proposed in [5, 6]. In the work of [6], the computations of $\frac{N}{2} - 1$ conjugate-symmetric samples were eliminated to obtain more efficient RFFT structures, where N represents the size of the FFT. Here, we consider a complex signal as two signals: real part signal and imaginary part signal. Therefore, in these architectures, the number of signals computed at the output is the same as the input, i.e., N. However, although the outputs are canonic in the number of signals, these architectures still exhibit redundancies at the intermediate stages, as they are composed of hybrid datapaths consisting of both complex and real datapaths.

Recently, pipelined architectures consisting of only real datapaths for decimation-in-frequency (DIF) RFFT were proposed in [7]. Real-valued FFT architectures for radix 2^3 and radix 2^4 were presented in [8] based on hybrid datapath. The architecture in [8] does not maintain the canonic property in number of signal values computed at the output of each FFT stage.

The goal of this chapter is to design general RFFTs that are canonic with respect to the number of signals at the output of each stage, i.e., for an N-point RFFT, the total number of values computed at the output of each stage should be N. Furthermore, each stage should contain maximum $\frac{N}{2}$ real butterflies as opposed to $\frac{N}{2}$ complex butterflies. In this chapter, we demonstrate that it is possible to compute only N independent values at each stage for an N-point RFFT. This property has not been explicitly studied before. Although this property is satisfied by only one prior architecture proposed in [7], general approaches for designing canonic RFFT computations have been not presented. This chapter makes two contributions: first, we present an approach to design canonic RFFT computation based on decimation-in-time (DIT) approach; second, we present a formal method to design RFFT structures for decimation-in-frequency (DIF) approach as presented in [7].

2.2 RFFT

The N-point Discrete Fourier Transform (DFT) for a sequence $x[n]$ is defined as [9]:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{N-1} x[n]W_N^{nk},$$
$$k = 0, 1, ..., N-1, \tag{2.1}$$

where $W_N = e^{-j\frac{2\pi}{N}}$. FFT is a fast algorithm to compute the DFT [10]. In algorithmic terms, the DFT requires $O(N^2)$ arithmetic operations, whereas the FFT takes $O(NlogN)$ arithmetic operations. The original DFT equation can be rearranged using different radices, thus giving rise to various FFT algorithms [11, 12, 13, 14]. Each of

these algorithms and the architectures provide unique tradeoffs that can be exploited for an intended application.

For real-valued inputs $x[n]$, it can be shown that (assume N is even)

$$X[k] = X^*[N-k]. \tag{2.2}$$

In this case, there are $\frac{N}{2}-1$ conjugate output pairs, i.e., $X[k]$ and $X[N-k]$, for $k = 1, 2, ..., \frac{N}{2}-1$. Therefore, only $\frac{N}{2}+1$ outputs need to be computed in an N-point RFFT flow-graph, since we can compute either $X[k]$ or $X[N-k]$, along with two real output signals $X[0]$ and $X[N/2]$. The total number of purely real and purely imaginary signal values is N. For example, as shown in Figure 2.1, for a 16-point radix-2 DIF FFT, we can choose to only compute $X[0] \sim X[8]$, while $X[9] \sim X[15]$ can be obtained by conjugating $X[1] \sim X[7]$. Thus, only 9 samples consisting of 16 values need to be computed at the output. This property of RFFT can be utilized to simplify the computation. The 16-point radix-2 DIF RFFT is shown in Figure 2.2. The shaded regions in Figure 2.2 are removed and only 9 outputs of the FFT are needed, where the nodes marked by $\circ$ and $\bullet$ respectively represent purely real or purely imaginary signals and complex signals. Different RFFT computations can also be generated based on different radices FFT algorithms.

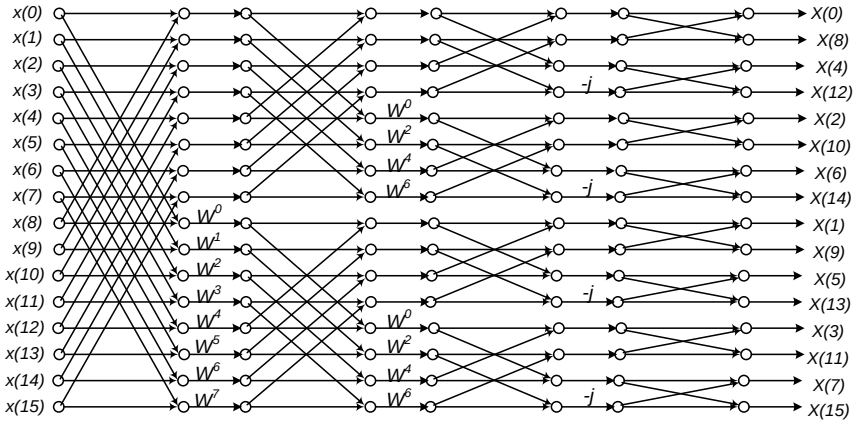

Figure 2.1: Flow-graph of a 16-point DIF FFT. Input signals are assumed to be complex.

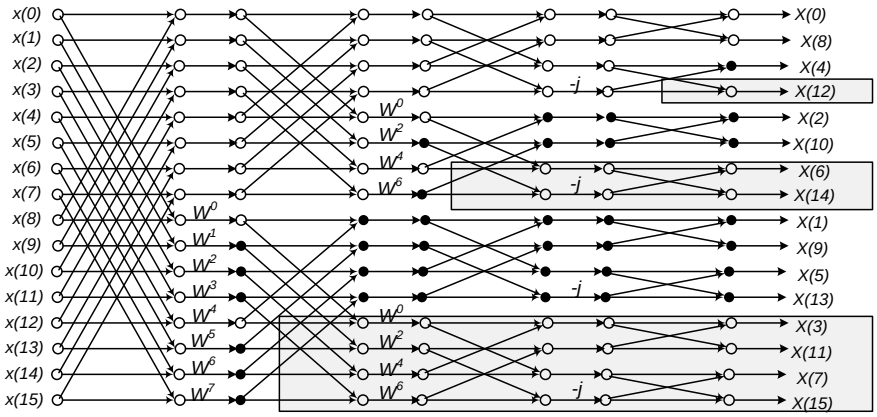

Figure 2.2: Flow-graph of a 16-point DIF real-valued FFT (RFFT). Input signals are purely real and redundant signals in the shaded regions are removed.

2.3 Canonic DIT RFFT

In this section, we present the flow-graphs for canonic DIT RFFT computations which are guaranteed to have N signals at each stage.

2.3.1 4-point RFFT

A 4-point canonic DIT RFFT flow-graph is shown in Figure 2.3. The nodes marked by $\circ$ and $\square$ respectively represent purely real and purely imaginary signals. Solid and dashed lines respectively represent purely real and purely imaginary datapaths. In the 4-point RFFT, since $X[1]$ and $X[3]$ are conjugates of each other, we can eliminate $X[3]$ by removing the bottom butterfly at the second stage. The computations of real and imaginary parts of $X[1]$ are separated as shown in Figure 2.3. Note that the number of inputs, the number of outputs and the number of signal values at the intermediate stage are all the same and equal 4, the size of FFT.

2.3.2 8-point RFFT

An 8-point canonic DIT RFFT can be obtained by merging two 4-point canonic RFFTs, as shown in Figure 2.4. In an 8-point FFT, $\frac{8}{2} - 1 = 3$ output computations can

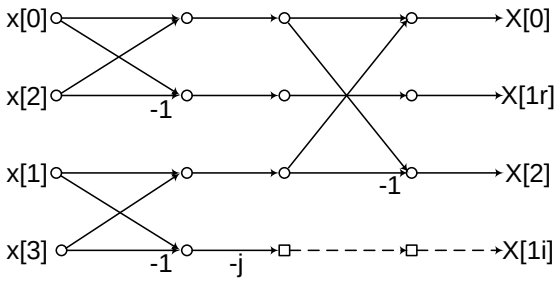

Figure 2.3: Flow-graph of a canonic 4-point DIT RFFT.

be eliminated. Since 2 samples have already been eliminated after the second stage by use of the 4-point canonic RFFTs, we need to eliminate $\frac{8}{2} - 1 - 2 \times 1 = 1$ more sample at the output. Thus, we can remove computation of $X[6]$ at the last stage, and the computation of the real part and imaginary part of $X[2]$ are separated. The real parts and imaginary parts of $X[1]$ and $X[5]$ are computed by two independent real butterflies, as these were separated in previous stages. The number of signal values computed at each FFT stage or the output is always 8; thus the structure is canonic.

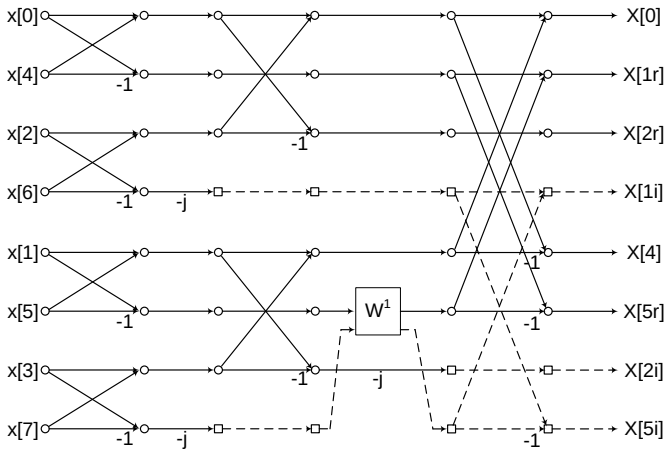

Figure 2.4: Flow-graph of a canonic 8-point DIT RFFT.

2.3.3 16-point RFFT

As shown in Figure 2.5, 16-point canonic RFFT flow-graph can also be obtained by merging two 8-point canonic RFFTs. At the last stage, only $\frac{16}{2} - 1 - 2 \times 3 = 1$ sample needs to be eliminated.

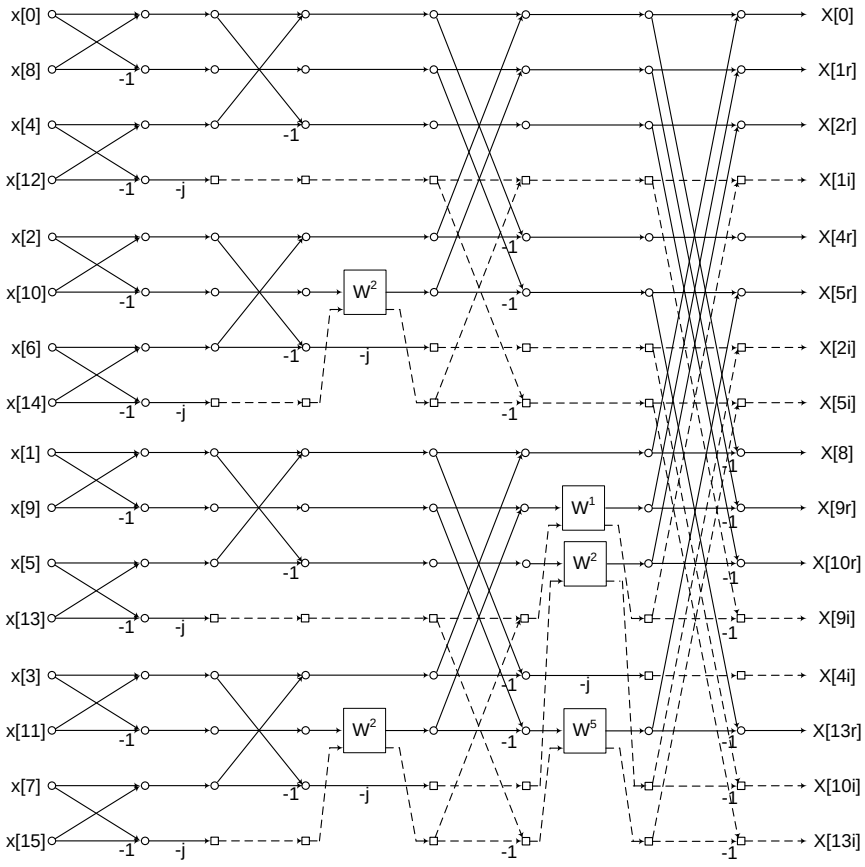

Figure 2.5: Flow-graph of a canonic 16-point DIT RFFT.

2.4 Algorithm for Canonic DIT RFFT Computation

2.4.1 Divide and Conquer

As illustrated in Section 2.3, any N-point RFFT, where N is power-of-two, can be obtained by merging two $\frac{N}{2}$-point RFFTs using the DIT property algorithm. This property can be expressed as

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{nk} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[2n]W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1]W_N^{(2n+1)k} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[2n]W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x[2n+1]W_N^{2nk} \\ & \qquad k = 0, 1, ..., N-1, \end{aligned} \tag{2.3}$$

where $\sum_{n=0}^{\frac{N}{2}-1} x[2n]W_N^{2nk}$ and $\sum_{n=0}^{\frac{N}{2}-1} x[2n+1]W_N^{2nk}$ respectively represent two $\frac{N}{2}$-point RFFTs for the even part and the odd part of the inputs. Twiddle factors W_N^k need to be multiplied to the odd part before the butterflies at the last stage, as shown in Figure 2.4 and Figure 2.5.

Therefore, due to the regularity of the canonic RFFT flow-graph, the canonic RFFT flow-graph can be extended for any $N = 2^n$-point DIT RFFT recursively. Note that for an $\frac{N}{2}$-point canonic RFFT, $\frac{N}{2}/2 - 1 = \frac{N}{4} - 1$ signals have been eliminated. Thus, when merging two $\frac{N}{2}$-point canonic RFFTs, only one more computation and its corresponding butterfly need to be eliminated at the last stage, as $\frac{N}{2} - 1 - 2 \times (\frac{N}{4} - 1) = 1$. We choose to eliminate the computation of $X[\frac{3N}{4}]$. Therefore, the butterfly which computes $X[\frac{N}{4}]$ and $X[\frac{3N}{4}]$ is removed, and the real and imaginary parts of $X[\frac{N}{4}]$ are separated. The twiddle factor before the bottom input of the removed butterfly (i.e., the $(\frac{3N}{4}+1)$th signal) can be calculated as $W_N^{\frac{3N}{4}+1-\frac{N}{2}-1} = W_N^{\frac{N}{4}} = -j$, as W_N^k is the twiddle factors

before the bottom inputs of the butterflies at the last stage for $k = 0, 1, ..., \frac{N}{2} - 1$. This operation is further illustrated as in Figure 2.6.

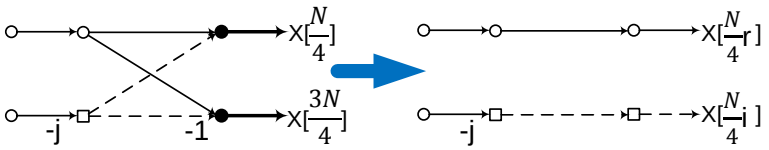

Figure 2.6: Remove butterfly and separate real part and imaginary part for DIT RFFT.

2.4.2 Generalization to $N = 2^n$-point DIT RFFT

We summarize the pattern for an $N = 2^n$-point canonic DIT RFFT in this section. The key idea of the proposed approach to implement a RFFT involves removing one butterfly for each 2^m-point RFFT at stage m for $2 \leq m \leq n$. Note that there are 2^{n-m} 2^m-point RFFTs at stage m. Therefore, we need to perform the operation as shown in Figure 2.6 to the butterflies whose bottom inputs are the $(2^m i + 3 \times 2^{m-2} + 1)$th signals, where $i = 0, 1, ..., 2^{n-m} - 1$. Similarly, it can be proved that the twiddle factors before the $(2^m i + 3 \times 2^{m-2} + 1)$th signal at stage m are

$$W_N^{2^{m-2}} \times W_N^{2^{n-m}} = W_N^{2^{n-2}} = W_N^{\frac{N}{4}} = -j, \tag{2.4}$$

as the twiddle factors for stage m are in group of 2^m (i.e., 2^{m-1} 0's and $W_N^{0:2^{n-m}:2^{n-1}-2^{n-m}}$).

This pattern is also summarized as in Table 2.1. The eliminations are performed from stage 1 to stage n to obtain the final N-point canonic DIT RFFT flow-graph.

Table 2.1: Bottom Input Positions of the Removed Butterflies

Stage	1	2	3	...	m	...	n
Positions	none	4,8,12,...	7,15,23,...	...	$2^m i + 3 \times 2^{m-2} + 1$, for $i = 0, 1, ..., 2^{n-m} - 1$	...	$3 \times 2^{n-2} + 1$

Moreover, it can be observed that the output signals for an N-point canonic RFFT will be $\{S_{\frac{N}{2}}, \frac{N}{2} + S_{\frac{N}{2}}\}$, where $S_{\frac{N}{2}}$ is the output set of an $\frac{N}{2}$-point canonic RFFT; and then replace $X[\frac{N}{4}]$ and $X[\frac{3N}{4}]$ with $X[\frac{N}{4}r]$ and $X[\frac{N}{4}i]$, respectively. This is also consistent

with the patterns described above, i.e., remove one butterfly whose bottom input is the $(\frac{3N}{4}+1)$th signal. This property is also summarized in Table 2.2.

Table 2.2: Canonic DIT RFFT Output Signals

RFFT Size N	4	8	16	32
Output S_N	0, 1r, 2, 1i	0, 1r, 2r, 1i	0, 1r, 2r, 1i, 4r, 5r, 2i, 5i	0, 1r, 2r, 1i, 4r, 5r, 2i, 5i, 8r, 9r, 10r, 9i, 4i, 13r, 10i, 13i
		4, 5r, 2i, 5i	8, 9r, 10r, 9i, 4i, 13r, 10i, 13i	16, 17r, 18r, 17i, 20r, 21r, 18i, 21i, 8i, 25r, 26r, 25i, 20i, 29r, 26i, 29i

Based on the patterns presented above, a 32-point DIT RFFT structure is designed as shown in Figure 2.7. In this structure, the number of signal values computed at the output of each FFT stage or the output is 32; thus, this structure is canonic.

2.5 Algorithm for Canonic DIF RFFT Computation

In this section, we discuss the canonic computation for DIF RFFT. An alternate flow-graph for a 16-point DIF FFT is shown in Figure 2.8. It can be seen that the DIT and DIF only differ in the twiddle factors at different stages. Similar to DIT, we eliminate the redundancies in the FFT computation to achieve the canonic RFFT structure. However, unlike the DIT, the DIF RFFT is *not* symmetric, i.e., the twiddle factors in the top $\frac{N}{2}$-point DIF RFFT are not the same as for the bottom $\frac{N}{2}$-point DIF RFFT. Therefore, we *cannot* directly use the divide and conquer idea to design a canonic DIF RFFT structures.

Moreover, we have to transform the twiddle factors for the DIF RFFT to ensure non-redundancy. For example, for a 16-point RFFT, there will be exactly 22 signal values at the beginning of the second stage, since no butterfly can be removed in the first stage and twiddle factors W^1, W^2, W^3, W^5, W^6, W^7 lead to complex signals. Therefore, we have to *push* these twiddle factors to a later stage to guarantee that only 16 signal values are input to the second FFT stage.

We could follow the same pattern as presented in Table 2.1 to design canonic DIF RFFT computation. The same number of butterflies will be removed at each stage. It can be observed that the twiddle factors before the top input and the bottom input

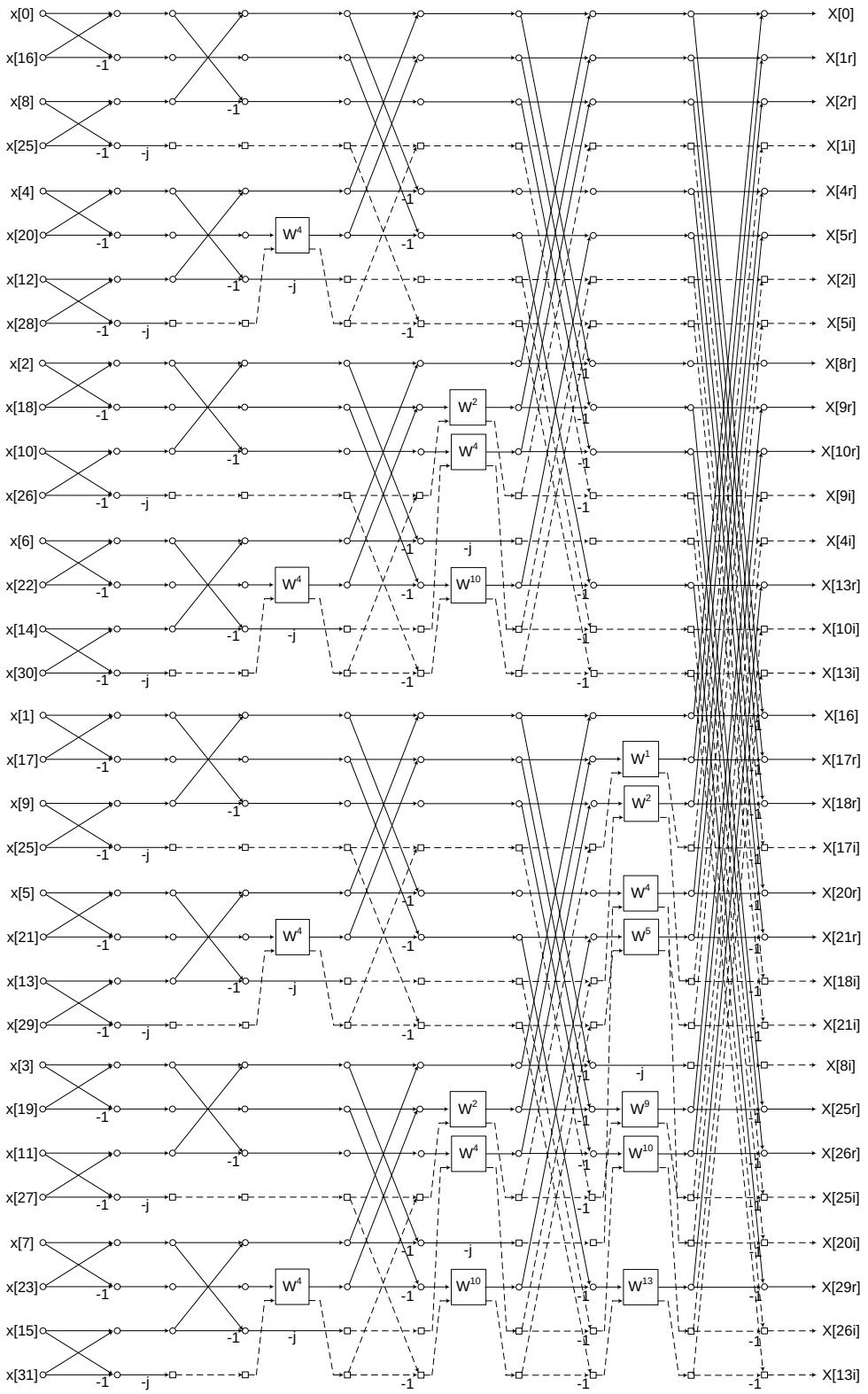

Figure 2.7: Flow-graph of a canonic 32-point DIT RFFT.

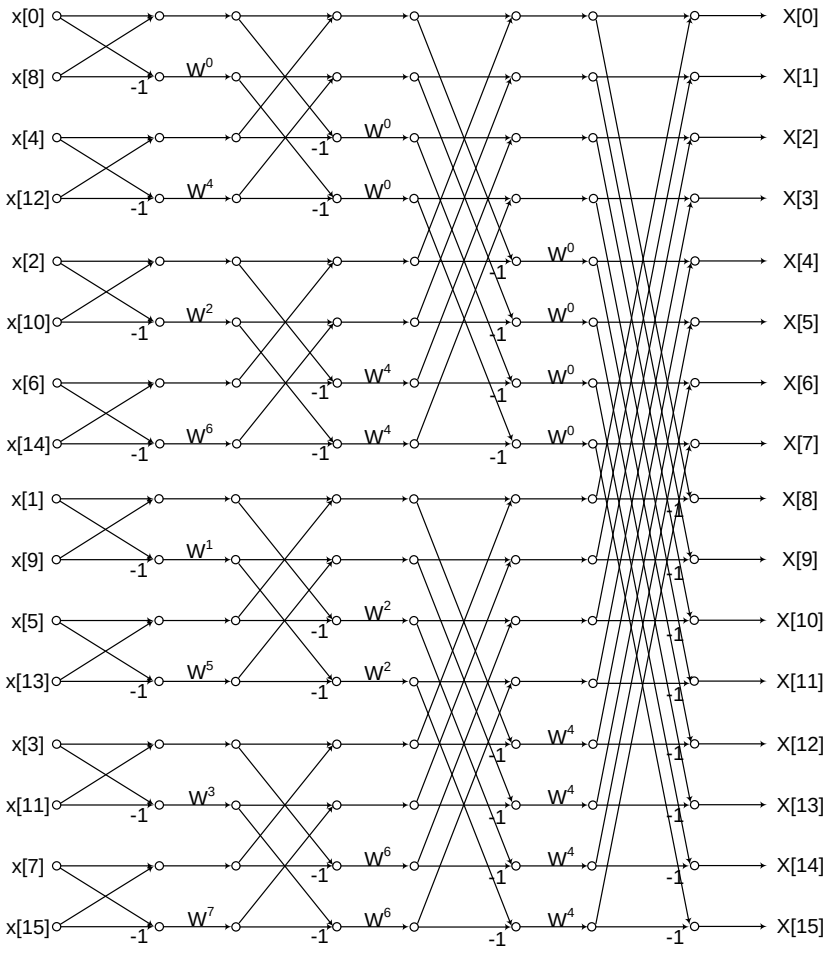

Figure 2.8: Alternate flow-graph of a 16-point DIF FFT.

of the removed butterflies can be expressed as W^k and $W^{k+N/4}$, respectively. Therefore, instead of performing the transformation shown in Figure 2.6 for a DIT RFFT, the transformation in Figure 2.9 is used to generate a canonic DIF RFFT structure. Figure 2.10 shows an example of a 16-point canonic DIF RFFT computation.

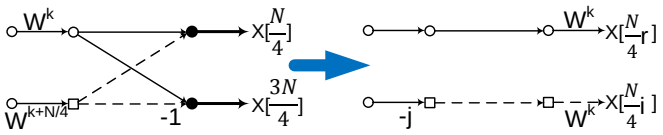

Figure 2.9: Remove butterfly and separate real part and imaginary part for DIF RFFT.

In conclusion, the canonic DIT and DIF RFFT computations can be designed by following the same pattern shown in Table 2.1. In order to remove the redundant butterflies and to separate the real and imaginary parts of complex values, we use the transformation in Figure 2.6 for a DIT RFFT, or the transformation in Figure 2.9 for a DIF RFFT. The patterns for the output signals for both the designs are the same and shown in Table 2.2.

2.6 Performance Comparison

The number of butterflies in the proposed canonic RFFT computation is the same as the previous works in [6, 7]. However, these structures require different numbers of twiddle factors. Furthermore, the number of twiddle factor operations even differs in the proposed canonic DIT RFFT and canonic DIF RFFT, as the transformations for removing the butterflies are different for the DIT RFFT and DIF RFFT, as shown in Figure 2.6 and Figure 2.9, respectively. Note that we do not consider $W^{\frac{N}{4}}$, $W^{\frac{N}{2}}$, and $W^{\frac{3N}{4}}$ as twiddle factor operations, since they only involve negation or switching.

It can be seen for an $N = 2^n$-point canonic DIT RFFT, the number of twiddle factors before stage k is

$$2^{n-k} \times (2^{k-2} - 1), \text{ for } 3 \leq k \leq n. \tag{2.5}$$

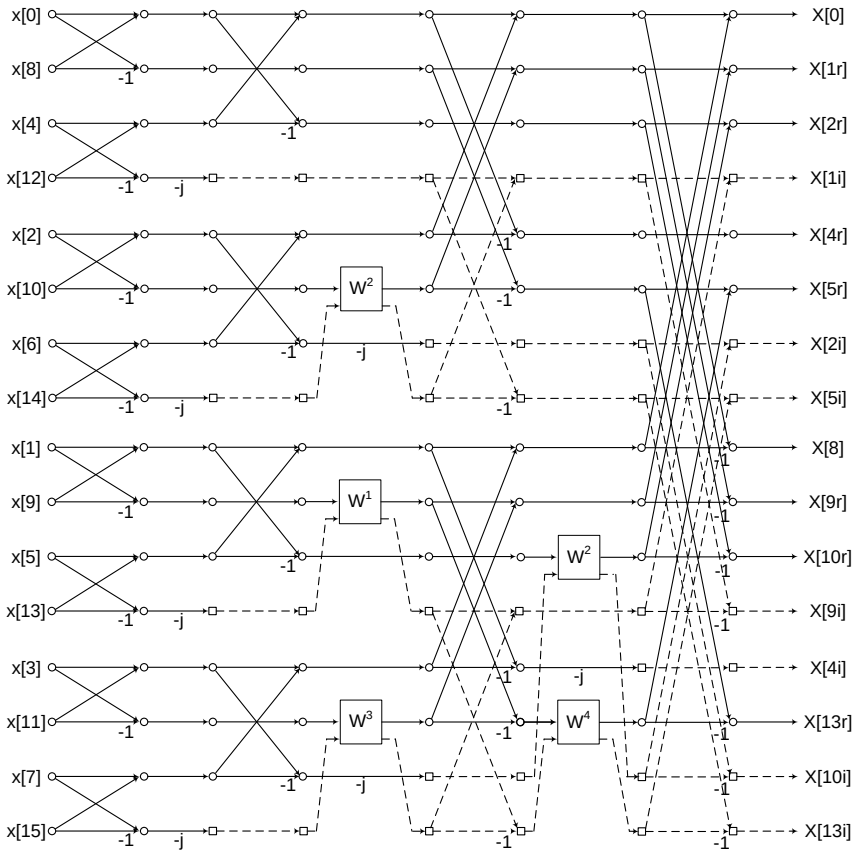

Figure 2.10: Flow-graph of a canonic 16-point DIF RFFT.

Therefore, the total number of twiddle factor operations can be calculated as

$$\begin{aligned}\#\text{twiddle factor} &= (n-1)\times 2^{n-2} - 2^{n-1} + 1 \\ &= (n-3)\times 2^{n-2} + 1. \end{aligned} \tag{2.6}$$

However, for an $N = 2^n$-point canonic DIF RFFT, the number of twiddle factors before stage k is

$$2^{k-3} \times (2^{n-k+1} - 2) - 1, \text{ for } 3 \leq k \leq n. \tag{2.7}$$

Therefore, the total number of twiddle factor operations can be calculated as

$$\begin{aligned}\#\text{twiddle factor} &= (n-2)\times 2^{n-2} - 2^{n-1} + 2 + (n-2) \\ &= (n-4)\times 2^{n-2} + n. \end{aligned} \tag{2.8}$$

As a result, it can be concluded that the canonic DIF RFFT computation has less twiddle factor operations than canonic DIT RFFT for $n \geq 4$, while canonic DIT and DIF RFFT computations have the same number of twiddle factors for $n = 2, 3$.

Table 2.3 compares the number of twiddle factor operations of the different approaches for the computation of the RFFT.

Table 2.3: Comparison of Twiddle Factor Operation and Datapath for $N = 2^n$-point RFFT

RFFT Algorithm	#Twiddle Factor Operations	Datapath
[6] DIF	$(n - \frac{7}{2}) \times 2^{n-2} + n - 1$	hybrid
[7] DIF	$(n-4) \times 2^{n-2} + n$	real
Proposed DIT	$(n-3) \times 2^{n-2} + 1$	real
Proposed DIF	$(n-4) \times 2^{n-2} + n$	real

As it is shown, the number of twiddle factor operations in all the approaches has the same order of magnitude. The proposed canonic DIF RFFT computation has less twiddle factor operations than the computation in [6], while it has the same performance as the work in [7].

2.7 Conclusion

This chapter has introduced the novel notion of canonic RFFT computations where the number of signal values computed at each intermediate FFT stage is same as the size of RFFT. The canonic computations for $N = 2^n$-point DIT and DIF RFFTs are proposed. It is shown that canonic DIF RFFT structures require less twiddle factor operations than the DIT counterparts. The proposed canonic structures are not necessarily canonic with respect to the twiddle factor operations and are non-unique.

Chapter 3

Data-Canonic Real FFT Flow-Graphs for Composite Lengths

3.1 Introduction

The designs of RFFTs for both decimation-in-time (DIT) and decimation-in-frequency (DIF) approaches that are canonic with respect to the number of signals at the output of each stage (i.e., *data-canonic*) have been proposed in [15]. Note that in this thesis, canonic is always referred to data-canonic unless otherwise specified. For a canonic N-point RFFT, the total number of values computed at the output of each stage is guaranteed to be N. Furthermore, each stage only contains maximum $\frac{N}{2}$ real butterflies as opposed to $\frac{N}{2}$ complex butterflies. However, the approaches presented in [15] are only general for radix-2 and power-of-two size RFFTs. However, no general algorithm to generate canonic RFFT for arbitrary length FFT has been presented yet. Furthermore, whether canonic RFFT designs require less complexity than hybrid datapath based architectures remains unknown.

In this chapter, we exploit the canonic property of RFFT designs. The main contribution is a *novel* general algorithm for designing canonic real-valued FFT computations for any composite length. First, a general *sample removal lemma* is introduced. Second, two types of *twiddle factor transformations*, i.e., *pushing* and *modulation*, are proposed to reduce the number of signal values. We consider 4 different cases for an $N = P \times Q$ point canonic RFFT: 1) P is odd, Q is odd; 2) P is odd, Q is even; 3) P is even, Q is odd; and 4) P is even, Q is even. We also compare the performances of different RFFT computations from both computation and architectural perspectives.

3.2 Canonic RFFT

RFFT algorithms can be further optimized, according to the specific application requirements. For example, three types RFFTs can be defined by considering the numbers of signal, multiplication, and addition, respectively:

Data-Canonic (Canonic): The RFFT algorithm has the least number of signals at each FFT stage (canonic is always referred to data-canonic in this thesis).

Multiplication-Canonic: The RFFT algorithm has the least number of multiplications.

Addition-Canonic: The RFFT algorithm has the least number of additions.

Note that data-canonic RFFTs are usually not multiplication-canonic or addition-canonic. There is no such RFFT algorithm that is optimized from all perspectives. Architectures for data-canonic DIT RFFT for radix-2 have been presented in [15]. The data-canonic DIF RFFT is also presented which requires transformations of twiddle factor operations to ensure non-redundancy. However, it is important to note that the dataflow does not correspond to a radix-2 flow graph anymore after transforming the twiddle factor operations.

In this chapter, we present a general solution to obtain data-canonic RFFT computation for any composite length FFT. We consider a 16-point radix-2 DIF RFFT for

example. The outputs are canonic with respect to the number of signals, 16 (i.e., 2 real values and 7 complex values). However, the intermediate stages of the dataflow as shown in Figure 2.2 are not canonic with respect to the number of signals. For instance, there are 10 real values and 6 complex values, i.e., 22 values in total before the butterfly operations at the second stage. In fact, it can be observed from Figure 2.2 that the canonic property is destroyed by the twiddle factor operations. A real signal becomes a complex signal after a W^k operation, where $k \neq 0, \frac{N}{4}, \frac{N}{2}, \frac{3N}{4}$, as shown in Figure 3.1. This type of operation increases the number of signals at the second stage and the third stage of Figure 2.2 by 6 and 2, respectively.

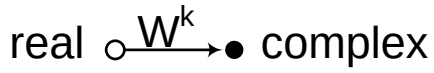

Figure 3.1: Number of signals is increased by twiddle factor operations.

In order to reduce the number of signal values to eliminate redundancy, one approach is to transform the twiddle factors. Figure 3.2 shows the the canonic structure of a 16-point RFFT obtained by pushing the twiddle factors from the original radix-2 16-point DIF RFFT dataflow. The "pushing" transformation of the twiddle factors can be described as shown in Figure 3.3. We can push a factor of W^k from before the butterfly operation to after the butterfly operation to reduce the number of signal values. For example, we can push a factor of W^2 to after the 4th butterfly operation at the third stage. After the twiddle factor transformation, the top input of the butterfly will be purely real and the bottom input will be purely imaginary. Therefore, the number of signals at this stage can be reduced to 16 from 18, which is canonic with respect to the number of signals. We also need to push the twiddle factors of the 6th, 7th, 8th butterflies at the second stage to obtain the canonic RFFT as shown in Figure 3.2. We can observe that in this 16-point DIF RFFT data-graph, all the twiddle factor transformations have the same pattern as shown in Figure 3.3, i.e., after pushing, the twiddle factor before the bottom input of the butterfly becomes $-j$. In this case, the

top output of the butterfly can be obtained by appending the two inputs, since the top input is purely real and the bottom input is purely imaginary; while the bottom output can be eliminated, as it will be conjugate symmetric to the top output.

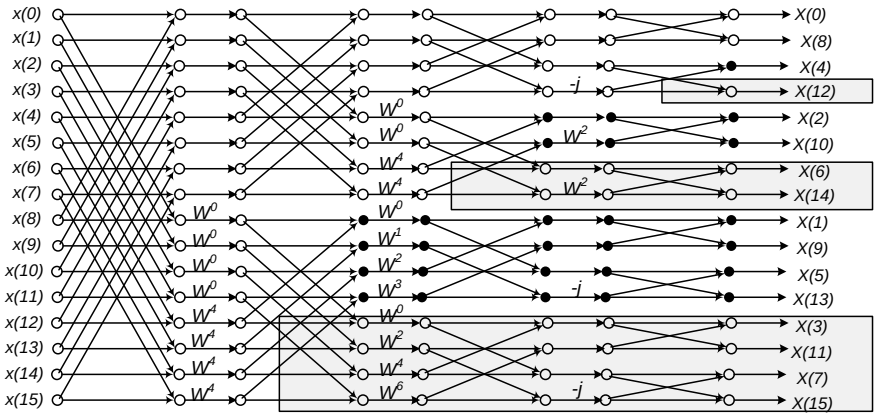

Figure 3.2: A canonic 16-point RFFT.

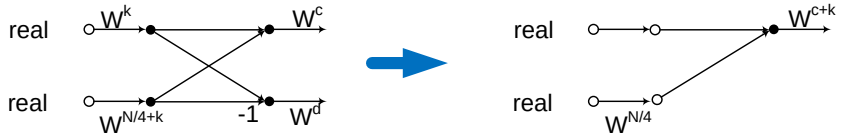

Figure 3.3: "Pushing" the twiddle factors.

The canonic structures for a certain size RFFT are not unique. This is because the twiddle factors can be moved from one stage to another if the signals before and after are complex without altering the number of signal values. For example, the twiddle factors after the second stage of the bottom part of Figure 3.2 can also be pushed to next stage. This operation does not affect the number of signal values for each stage.

Algorithm 1 describes a general algorithm to generate canonic computation for power-of-two size RFFTs. A canonic RFFT computation can be mapped into an architecture with only real datapaths, which is a major advantage of canonic computations. The real and imaginary components of complex signals are processed in separate time slots by the real butterflies to utilize the hardware efficiently.

Algorithm 1 Generate Canonic Computation for Power-of-Two Size RFFTs

1. For any given RFFT dataflow
2. Start from the first stage to the last stage
3. Find twiddle factors W^k which operate on real signals, for $k \neq 0, \frac{N}{4}, \frac{N}{2}, \frac{3N}{4}$
4. Push the twiddle factors to next stage

3.3 Twiddle Factor Transformations

3.3.1 Sample Removal

In previous sections, we have demonstrated it is possible to design canonic computations of both DIT and DIF structures for any $N = 2^n$-point RFFT. The canonic RFFT computation can also be extended to other non-power-of-two size RFFTs. The idea is to ensure N values stage by stage. The sample removal rule stated below forms the basis for design of canonic non-power-of-two size RFFTs.

Lemma (Sample Removal):

For an M-point RFFT, the number of samples that need to be removed is equal to $\lceil \frac{M}{2} \rceil - 1$ minus the number of samples removed at previous stages.

Proof: As shown in Figure 3.4(a), we should remove $\frac{M-1}{2}$ complex signals and keep the other $\frac{M-1}{2}$ complex signals and one real signal when M is odd. When M is even as shown in Figure 3.4(b), we need to remove $\frac{M-2}{2}$ complex signals and keep other $\frac{M-2}{2}$ complex signals and two real signals. Therefore, we need to make sure that $\lceil \frac{M}{2} \rceil - 1$ redundant samples are removed for an M-point RFFT. As a result, before the last stage of the M-point RFFT, the number of samples that need to be removed is equal to $\lceil \frac{M}{2} \rceil - 1$ minus the number of samples removed at previous stages. ■

Consider the 16-point FFT with 4 stages as shown in Figure 2.2. We illustrate sample removal lemma for each stage. For the first stage, as $\frac{2}{2} - 1 = 0$, we cannot remove any sample. At the second stage, we need to remove $\frac{4}{2} - 1 = 1$ sample for each 4-point RFFT block. Then at the third stage, $\frac{8}{2} - 1 - 2 \times 1 = 1$ sample is removed

for each 8-point RFFT block, since each 8-point RFFT is obtained by merging two 4-point RFFTs. Finally, $\frac{16}{2} - 1 - 2 \times 3 = 1$ more sample needs to be eliminated at the last stage. This resulting computation corresponds to the 16-point RFFT dataflow shown in Figure 3.2. For different radices, certain twiddle factor transformations may be required to eliminate the redundancy.

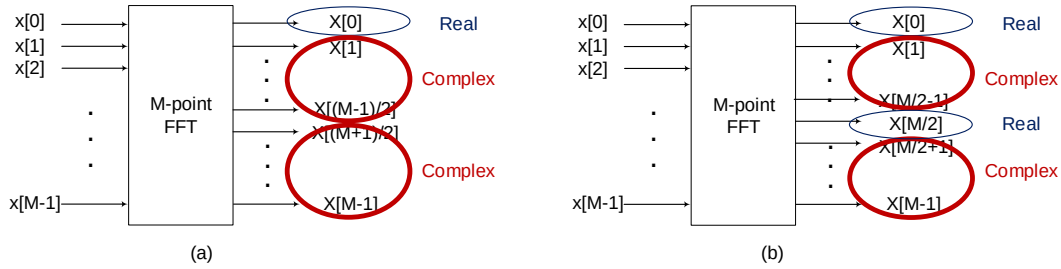

Figure 3.4: M-point RFFT: (a) M is odd, (b) M is even.

The key idea of generating canonic RFFT computation is to ensure canonic property from the first stage to the last stage. In order to delete redundant samples, we have to ensure the inputs of the corresponding FFT block are purely real or purely imaginary (i.e., RFFT). For example, as shown in Figure 3.3, when one input of a butterfly is purely real and the other input is purely imaginary, one output can be eliminated, as it is conjugate symmetric to the other output. If the number of deleted samples cannot meet the required number (i.e., $\lceil \frac{M}{2} \rceil - 1$ for an M-point RFFT), transformations of twiddle factor operations are required to force the inputs to be purely real or purely imaginary. In other words, we attempt to reduce nontrivial twiddle factors to $W^k = 0, -j, -1, j$, to maintain the data-canonic property.

3.3.2 Twiddle Factor Transformation Type I: Pushing

As described in Section 3.2, we can push the twiddle factors to reduce the number of signal values. For example, if there is a twiddle factor operation as shown in Figure 3.1, we can push the twiddle factors to the next stage. If $b = 0, \frac{N}{4}, \frac{N}{2}, \frac{3N}{4}$, the number of

signals can be reduced by 2 for each process; otherwise, the number of signals can be reduced by 1, as illustrated in Figure 3.5. Figure 3.3 essentially illustrates a special case when $b = \frac{N}{4}$.

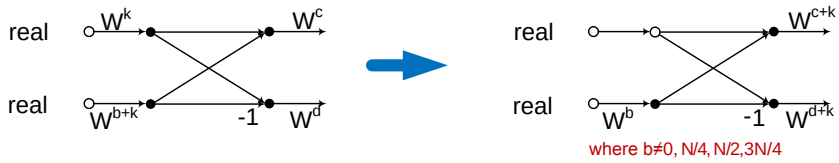

Figure 3.5: Twiddle factor transformation type I: pushing.

3.3.3 Twiddle Factor Transformation Type II: Modulation

Another important technique we can use to reduce the number of signal values is the modulation transformation. The modulation property (circular shift in frequency) can be expressed by

$$x[n] \leftrightarrow X[k], \tag{3.1}$$

$$W_N^{-k_0 n} x[n] \leftrightarrow X[(k - k_0)_N]. \tag{3.2}$$

It can be seen that the outputs of the FFT will be circularly shifted right by k_0 positions, if we multiply each input $x[n]$ by $W_N^{-k_0 n}$. For example, we consider a 3-point FFT as shown in Figure 3.6(a). If we multiply input signals b and c by W^{-1} and W^{-2}, respectively, the outputs will be circularly shifted by $k_0 = 1$, as shown in Figure 3.6(b). Similarly, if we want to circularly shift the output right by $k_0 = 2$, we need to multiply input signals b and c by W^{-2} and W^{-4}, respectively.

The modulation transformation is used to reduce the number of signal values if the inputs of an N-point FFT have the pattern as shown in Figure 3.7(a): the twiddle factors after the real inputs follow a geometric sequence, i.e., W^{an}. In this case, we can multiply each input to the FFT by $W^{-(\frac{N}{2}+a)n}$ (i.e., $k_0 = \frac{N}{2} + a$) to compute the circularly shifted FFT, as shown in Figure 3.7(b). As a result, the effective twiddle

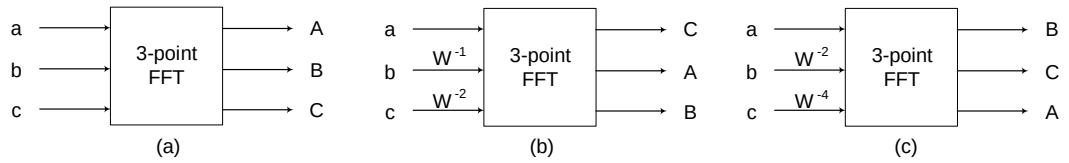

Figure 3.6: Modulation property of a 3-point FFT. (a) The original 3-point FFT, (b) output shifted by $k_0 = 1$, (c) output shifted by $k_0 = 2$.

factors are transformed to $W^{-\frac{N}{2}n}$, i.e., $1, -1, 1, -1, ..., 1$. Consequently, all the inputs of the N-point FFT become purely real, which leads to an N-point RFFT, as shown in Figure 3.7(c). The outputs will be shifted by k_0 according to the modulation property. It is important to note that $k_0 = \frac{N}{2} + a$ should be a multiple of the output signal interval, i.e., an integer in this case.

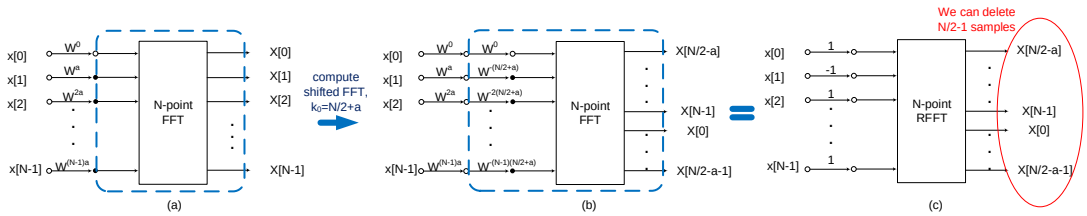

Figure 3.7: Twiddle factor transformation type II: modulation. (a) The original N-point FFT, (b) compute the shifted FFT, $k_0 = \frac{N}{2} + a$, (c) the final N-point RFFT.

3.4 Canonic Computations for Composite Size RFFTs

In this section, we present the algorithm for generating canonic RFFT computation for any composite size. We assume an N-point RFFT that constitutes Q P-point RFFTs at the first stage and P Q-point RFFTs at the second stage, i.e., $N = P \times Q$. We discuss the process for 4 different cases.

3.4.1 $N = P \times Q$, P is odd, Q is odd

If P is odd, we can remove $\frac{P+1}{2} - 1$ samples for each P-point RFFT. As a result, we can delete $Q(\frac{P+1}{2} - 1) = \frac{PQ-Q}{2}$ samples at the first stage, since there are Q P-point RFFTs. As we need to delete $\frac{PQ+1}{2} - 1$ samples in total, $\frac{PQ+1}{2} - 1 - \frac{PQ-Q}{2} = \frac{Q-1}{2}$ samples are required to be eliminated at the second stage. Since P is odd, there is only 1 real output sample for each P-point RFFT. All of these real samples will input to one Q-point RFFT at the second stage. Therefore, $\frac{Q+1}{2} - 2 = \frac{Q-1}{2}$ output samples can be removed from this Q-point RFFT; this satisfies the number of samples that are required to be deleted.

For example, we consider two 15-point RFFTs as shown in Figure 3.8 and Figure 3.9, respectively. The complex signals are marked as bold.

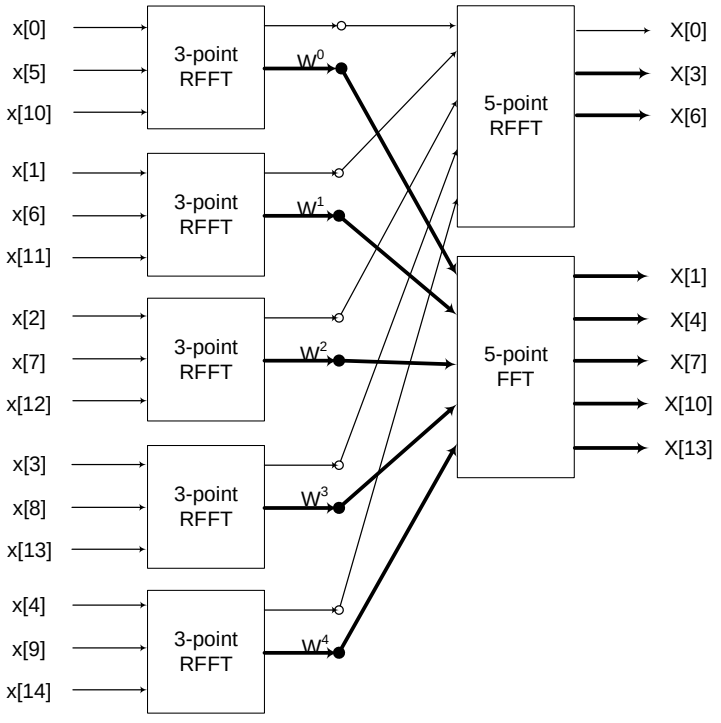

Figure 3.8: A 15-point canonic RFFT, where $P = 3, Q = 5$.

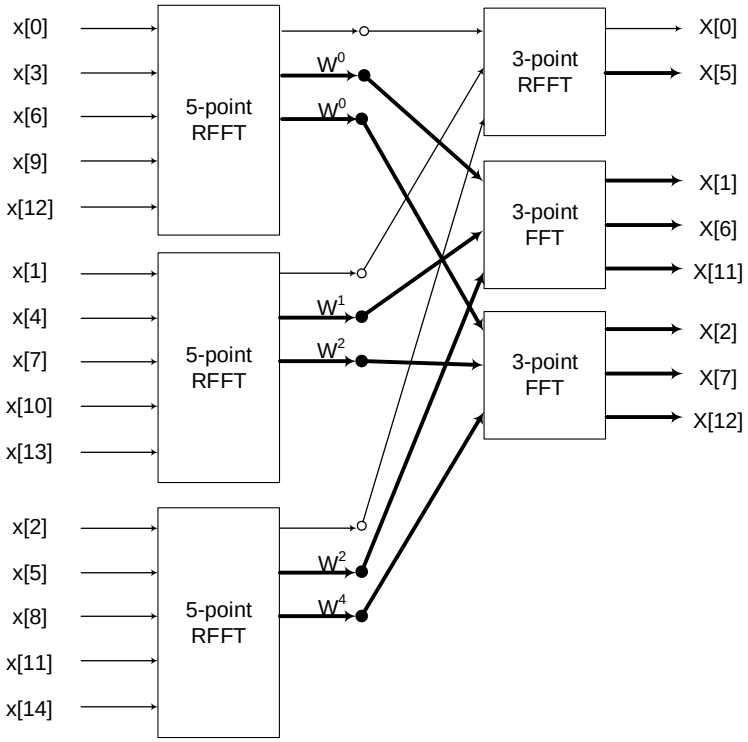

Figure 3.9: A 15-point canonic RFFT, where $P = 5, Q = 3$.

For the 3×5 structure as shown in Figure 3.8, we could delete $5(\lceil \frac{3}{2} \rceil - 1) = 5$ samples at the first stage. There are only one 5-point RFFT and one 5-point complex FFT at the second stage. We can delete two more samples from the outputs of the 5-point RFFT whose inputs are all purely real.

However, we could delete $3(\lceil \frac{5}{2} \rceil - 1) = 6$ samples at the first stage for the 5×3 structure as shown in Figure 3.9. In this case, there are one 3-point RFFT and two 3-point complex FFT at the second stage. We can delete one more sample from the outputs of the 3-point RFFT whose inputs are all purely real. Both of the two structures are canonic with respect to the number of signals.

3.4.2 $N = P \times Q$, P is odd, Q is even

If P is odd and Q is even, $\frac{PQ}{2} - 1 - Q(\frac{P+1}{2} - 1) = \frac{Q}{2} - 1$ samples need to be deleted at the second stage. Since there is one real output for each P-point RFFT and Q is even, $\frac{Q}{2} - 1$ samples can be deleted from the Q-point RFFT whose inputs are all purely real. As a result, a canonic structure can be obtained without any twiddle factor transformation.

For example, we consider a $3 \times 2 = 6$-point FFT shown in Figure 3.10, where the 3-point FFT is shown in Figure 3.11.

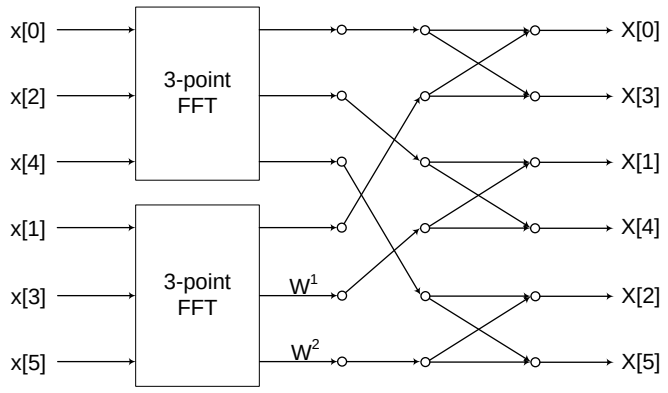

Figure 3.10: Flow-graph of a 6-point DIF FFT.

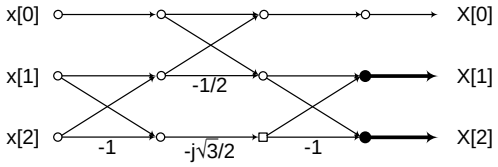

Figure 3.11: Flow-graph of a 3-point FFT.

For the 6-point RFFT computation in Figure 3.10, we can remove one sample (i.e., $\lceil \frac{3}{2} \rceil - 1 = 1$) in the 3-point FFT as shown in Figure 3.12, since the second and the third outputs are conjugate when the inputs are real. As a result, we do not need to remove any sample at the last stage, as $\frac{M}{2} - 1 = \frac{6}{2} - 1 = 2$, which is same as the number of samples removed at the first stage. The final canonic 6-point RFFT computation is shown in Figure 3.13. There is no need to transform the twiddle factors.

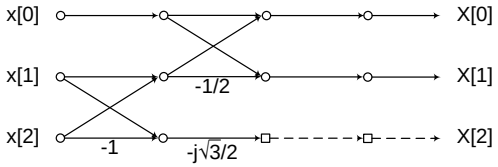

Figure 3.12: Flow-graph of a 3-point RFFT.

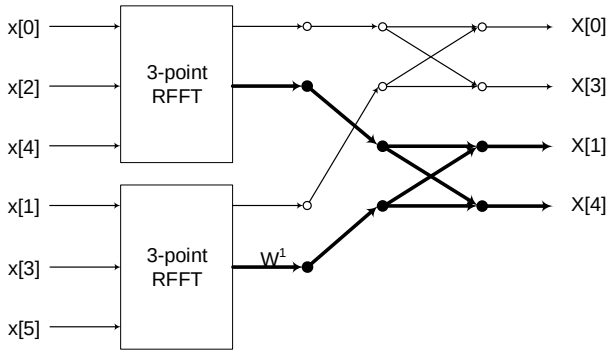

Figure 3.13: Flow-graph of the canonic 6-point RFFT for the structure shown in Figure 3.10.

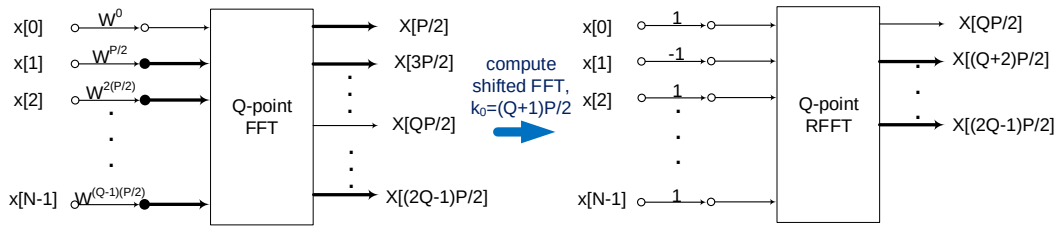

Figure 3.14: Operation to the Q-point FFT whose inputs are from the second real outputs of these P-point RFFTs after going through nontrivial twiddle factors.

3.4.3 $N = P \times Q$, P is even, Q is odd

If P is even, then $Q(\frac{P}{2}-1) = \frac{PQ}{2} - Q$ samples can be deleted. Therefore, $\frac{PQ}{2} - 1 - \frac{PQ}{2} + Q = Q - 1$ samples need to be removed at the second stage. Since P is even, 2 real samples are generated for each P-point RFFT. However, one of the two real outputs for each P-point RFFT will go through a nontrivial twiddle factor operation. These twiddle factors for these real samples are $W_N^0, W_N^{\frac{P}{2}}, W_N^{2\frac{P}{2}}, ..., W_N^{(Q-1)\frac{P}{2}}$, respectively, as shown in the left of Figure 3.14.

To make use of the modulation transformation as described in Section 3.3, we can circularly shift the outputs by $k_0 = (Q+1)\frac{P}{2}$ positions. This is equivalent to multiplying the inputs by $W_N^{-(\frac{N}{2}+\frac{P}{2})n} = W_N^{-(Q+1)\frac{P}{2}n}$, i.e., $W_N^0, W_N^{-(Q+1)\frac{P}{2}}, W_N^{-2(Q+1)\frac{P}{2}}, ..., W_N^{-(Q-1)(Q+1)\frac{P}{2}}$, respectively. As a result, these twiddle factors are transformed to $W_N^0, W_N^{-Q\frac{P}{2}}, W_N^{-QP}, ..., W_N^{-(Q-1)\frac{P}{2}}$, respectively, which are equivalent to $1, -1, 1, -1, ..., 1$. Thus, all the inputs of the corresponding Q-point FFT become purely real at the beginning of the second stage. In this case, the output signals of this Q-point FFT will be in the order of $X[Q\frac{P}{2}]$, $X[(Q+2)\frac{P}{2}]$, $X[(Q+4)\frac{P}{2}]$, ..., $X[(2Q-1)\frac{P}{2}]$, $X[\frac{P}{2}]$, $X[3\frac{P}{2}]$, ... $X[(Q-2)\frac{P}{2}]$, as $k_0 = (Q+1)\frac{P}{2}$, which is a multiple of the output interval P. The last $\frac{Q-1}{2}$ outputs $X[\frac{P}{2}]$, $X[3\frac{P}{2}]$, ... $X[(Q-2)\frac{P}{2}]$ can be removed due to redundancy. The operation is shown in Figure 3.14. The canonic property is achieved, as there are two Q-point RFFTs and $\frac{P}{2}-1$ Q-point FFTs (i.e., there are $2 \times Q + (\frac{P}{2}-1) \times 2Q = P \times Q$

signals). As a result, we will be able to remove $2(\frac{Q+1}{2} - 1) = Q - 1$ samples from the two Q-point RFFTs whose inputs are all purely real, as Q is odd.

We can consider another 6-point FFT as shown in Figure 3.15, where $P = 2$ and $Q = 3$. However, we have to transform the twiddle factors for this RFFT, since no butterfly can be removed in the first stage and twiddle factors W^1, W^2 lead to complex signals at the beginning of the second stage. According to the operation as shown in Figure 3.14, we can compute the shifted 3-point FFT by multiplying the second signal and the third signal of the bottom 3-point FFT at the second stage by W_N^{-4} and W_N^{-8}, respectively (i.e., $k_0 = 4$), as shown in Figure 3.16. As a result, there will be only 6 signals at the beginning of the second stage, since $W_N^1 \times W_N^{-4} = -1$ and $W_N^2 \times W_N^{-8} = 1$. In this case, the output signals of the bottom 3-point FFT will be in the order of $X[3]$, $X[5]$, and $X[1]$. The final canonic 6-point RFFT computation is shown in Figure 3.17.

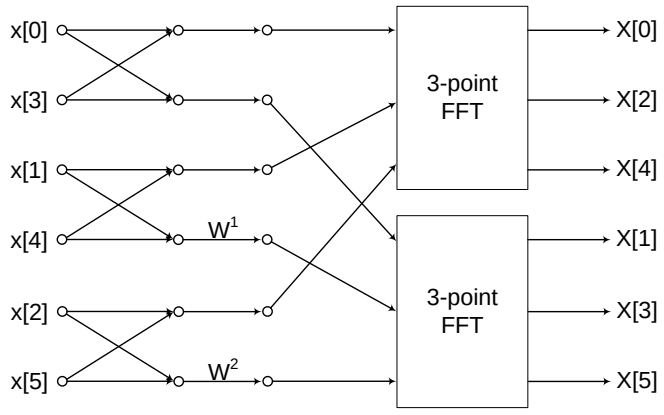

Figure 3.15: Flow-graph of a 6-point DIF FFT.

The two canonic 6-point RFFTs also have the same number of butterflies, i.e., 7 butterflies. However, their numbers of twiddle factors are different. It can be observed that the computation in Figure 3.13 has one complex twiddle factor operation, while the computation in Figure 3.17 does not require any twiddle factor operation. Therefore,

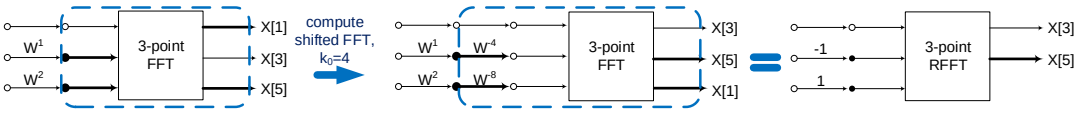

Figure 3.16: Compute the shifted FFT for the bottom 3-point FFT at the second stage of Figure 3.15. The 3-point FFT can be reduced to RFFT (as shown in Figure 3.12) after modulation transformation.

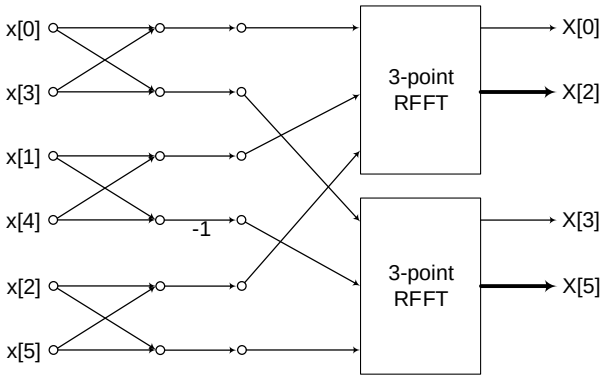

Figure 3.17: Flow-graph of the canonic 6-point RFFT for the structure shown in Figure 3.15.

we can conclude that the computation in Figure 3.17 requires less computations, which could save area and/or power.

In general, for an $(N = P \times Q)$-point RFFT, where P is even and Q is odd, $P \times Q$ RFFT structure will consume less area and/or power than $Q \times P$ RFFT structure, as $P \times Q$ structure could reduce the twiddle factor operations by transforming the twiddle factors.

3.4.4 $N = P \times Q$, P is even, Q is even

If P and Q are both even, we also need to remove $\frac{PQ}{2} - 1 - \frac{PQ}{2} + Q = Q - 1$ at the second stage. We can remove $\frac{Q}{2} - 1$ samples of the Q-point RFFT whose inputs are the first outputs of the P-point RFFTs. In this case, the nontrivial twiddle factors for the second real outputs of these P-point RFFTs are still $W_N^0, W_N^{\frac{P}{2}}, W_N^{2\frac{P}{2}}, ..., W_N^{(Q-1)\frac{P}{2}}$, respectively. However, we are not able to use the same method as for P is even and Q

is odd, as $(Q+1)\frac{P}{2}$ is not a multiple of P in this case; while k_0 must be a multiple of P in order to perform modulation transformation, as the output interval of the Q-point FFT is P.

Since Q is even, we can consider it as a $(2 \times \frac{Q}{2})$-point FFT, as shown in Figure 3.18. $x[k]$ and $x[k+\frac{Q}{2}]$ go through a butterfly operations first, for $0 \leq k \leq \frac{Q}{2}-1$. We can perform the operation as shown in Figure 3.3 to these butterflies, i.e., push W^k to after the butterflies. As a result, the top input and the bottom input of the butterfly operation become purely real and purely imaginary, respectively. The bottom output of each butterfly can be eliminated, as it is conjugate of the top output. Then, these outputs are processed by one $\frac{Q}{2}$ FFT, as shown in Figure 3.19. Note that two real twiddle factor operations at the inputs are transformed to one complex twiddle factor operation at the output for each butterfly of this Q-point FFT.

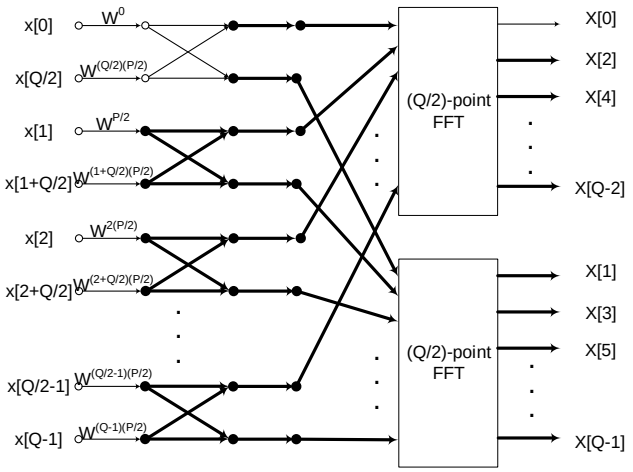

Figure 3.18: A Q-point FFT is considered as a $(2 \times \frac{Q}{2})$-point FFT.

Therefore, $X[\frac{Q}{2}]$ output samples of this Q-point FFT can be eliminated. As a result, a canonic structure is obtained. Figure 3.19 represents a canonic RFFT dataflow if we separate the real parts and imaginary parts of output signals. Note that when P is even, the operation as shown in Figure 3.14 or Figure 3.19 is only performed for the Q-point RFFT whose inputs are from $X[\frac{P}{2}]$ of the P-point RFFTs at the first stage,

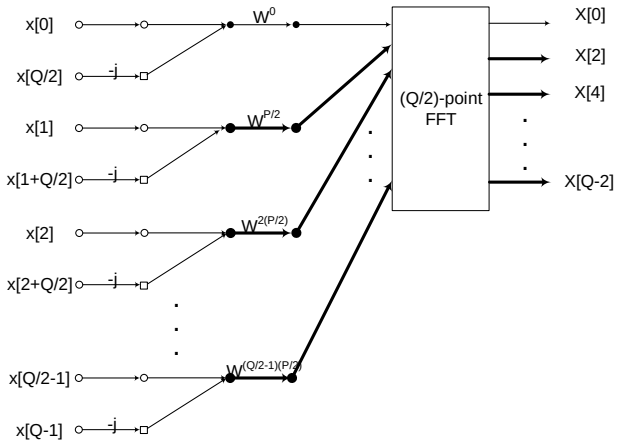

Figure 3.19: The $(2 \times \frac{Q}{2})$-point RFFT after eliminating redundancy.

i.e., the Q-point RFFT whose inputs are from the second real outputs of these P-point RFFTs at the first stage after going through nontrivial twiddle factors. No operation is needed for the first Q-point RFFT and the remaining $(\frac{P}{2} - 1)$ Q-point FFTs. Note that in this case, we need to ensure that $Q \geq 4$ in order to decompose Q as $(2 \times \frac{Q}{2})$. Since P is also even, if $Q = 2$, we will be able to re-decompose $P \times Q$ as $\frac{P}{2} \times 2Q$.

All radix-2^m RFFT structures fall into this category. For example, a radix-4 16-point RFFT is shown in Figure 3.20. At the first stage, we could delete $4(\lceil \frac{4}{2} \rceil - 1) = 4$. At the second stage, we could remove one more sample from the 4-point RFFT whose inputs are from the first real outputs of 4-point RFFTs at the first stage. For the 4-point FFT whose inputs are from the second real outputs of 4-point RFFTs at the first stage after going through nontrivial twiddle factors, we consider it as a 2×2 FFT. Then we push the twiddle factors to after the butterfly operations as shown in Figure 3.19. Two samples can be deleted from this 4-point FFT. As a result, there are only 9 samples at the output. A canonic RFFT structure is obtained.

3.4.5 Proposed Algorithm to Generate Canonic RFFTs

Based on the above discussion, we propose a novel algorithm for designing canonic RFFT computations from any given FFT dataflows, as described in Algorithm 2.

Algorithm 2 Generate Canonic RFFTs

1. For any given N-point FFT dataflow, where $N = M_1 \times M_2 \times M_3 ... \times M_L$, where L is the number of stages.

2. Begin from the first stage, delete $\lceil \frac{M_1}{2} \rceil - 1$ output samples for each M_1-point FFT.

3. Go to the second stage $x = 2$, where x is the stage number. Let $P = M_1$ and $Q = M_2$.

4. Do the operation for the 4 cases according to the values of P and Q.
   a) If P is odd and Q is odd, $\frac{Q+1}{2} - 1$ samples can be deleted from the Q-point FFT whose inputs are all purely real;
   b) else if P is odd and Q is even, $\frac{Q}{2} - 1$ samples can be deleted from the Q-point FFT whose inputs are all purely real;
   c) else if P is even and Q is odd, perform the operation as shown in Figure 3.14 to the Q-point FFT whose inputs are from the second real outputs of these P-point RFFTs after going through nontrivial twiddle factors.
   d) else when P is even and Q is even, by considering the Q-point FFT whose inputs are from the second real outputs of these P-point RFFTs after going through nontrivial twiddle factors as a $2 \times \frac{Q}{2}$-point FFT, push the twiddle factors to after the butterflies and remove redundant samples as shown in Figure 3.19.

5. If $x < L$, let $P = P \times M_x$, $Q = M_{x+1}$ and $x = x + 1$, go back to Step 4.

3.5 Performance

In this section, we discuss the performances of canonic RFFT computation with other RFFT algorithms.

Compared to other existing RFFT algorithms, the canonic structures have the least number of signals at each stage, which equals the degree of freedom of the computation. Therefore, the number of butterfly operations in the proposed canonic RFFT computation would also be the least among different RFFT algorithms, which is the same as the previous work in [7] when the size is power-of-two. The RFFT structures in [6] require more butterfly operations than the canonic approach.

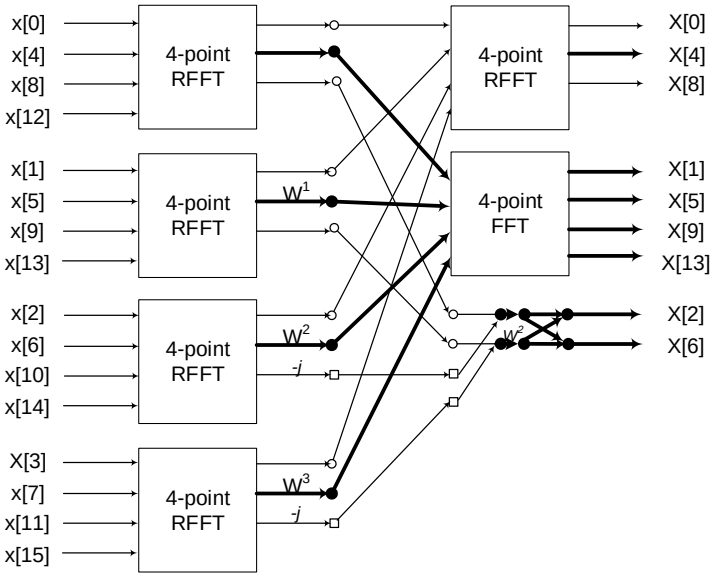

Figure 3.20: A 16-point canonic RFFT, where $P = 4, Q = 4$.

Moreover, canonic RFFT structures also have less number of twiddle factor operations, as two twiddle factor operations will be pushed to one twiddle factor operation at the next stage. We have already presented in [15] that the canonic power-of-two size DIF RFFT computation has less twiddle factor operations than the computation in [6], while it has the same performance as the work in [7].

We extend the comparison to general size cases: consider the two-stage structure with size of $P \times Q$. Table 3.1 presents the number of twiddle factor operations for FFT, conventional RFFT, and canonic RFFT, where conventional RFFT represents the RFFT structure obtained by simply removing redundant signals (i.e., without twiddle factor transformation) from FFT.

Table 3.1: Number of Twiddle Factor Operations

P	Q	FFT	Conventional RFFT		Canonic RFFT	
			Complex	Real	Complex	Real
odd	odd	$(P-1)(Q-1)$	$(P-1)(Q-1)/2$	0	$(P-1)(Q-1)/2$	0
odd	even	$(P-1)(Q-1)$	$(P-1)(Q-1)/2$	0	$(P-1)(Q-1)/2$	0
even	odd	$(P-1)(Q-1)$	$(P-2)(Q-1)/2$	$(Q-1)$	$(P-2)(Q-1)/2$	0
even	even	$(P-1)(Q-1)$	$(P-2)(Q-1)/2$	$(Q-2)$	$(P-2)(Q-1)/2 + (Q-2)/2$ #	0

$(Q-2)/2$ should be removed if we do not consider the twiddle factor operations generated inside the $(2 \times \frac{Q}{2})$ RFFT.

It can be seen that there is no real twiddle factor operation in canonic RFFTs, since these are removed by twiddle factor transformation when P is even. It can also be seen that the numbers of twiddle factors for conventional RFFT and canonic RFFT are the same when P is odd. However, when P is even, the canonic RFFT requires less computations than conventional RFFT with the use of twiddle factor transformation, which could save area and/or power. Furthermore, it can be concluded that for an $(N = P \times Q)$-point RFFT, where P is even and Q is odd, $P \times Q$ RFFT structure will consume less area and/or power than $Q \times P$ RFFT structure, as $P \times Q$ structure could reduce the twiddle factor operations by transforming the twiddle factors.

The costs of different types of twiddle factor operations are not the same. Note that in practical applications, we do not consider $W^{\frac{N}{4}}$, $W^{\frac{N}{2}}$, and $W^{\frac{3N}{4}}$, which appear quite frequently in power-of-two RFFTs, as twiddle factor operations, since these only involve negations or swap operations. In order to measure the total performance, we compare the costs of different types of twiddle factor operations through experimental results. All circuits were synthesized using Synopsys Design Compiler with optimization parameters set for minimum area and mapped to a 65 nm standard cell library. The area/power consumptions for different types of twiddle factor operations of word-length 16 are presented in Table 3.2, which are normalized to the cost of a 16×16 real multiplier.

Table 3.2: Normalized Cost of Different Types of Twiddle Factor Operations (16-bit)

	Area	Power
Real Twiddle	1	1
Complex Twiddle	2.09	2.04

Based on the experimental results, we can conclude that the canonic RFFT could reduce $(Q-1)$ real twiddle factor operations when P is even and Q is odd. Therefore, the performance of the RFFT computation would be improved using the canonic approach.

When P and Q are both even, we can calculate the total power consumption of the twiddle factor operations for conventional RFFT and canonic RFFT as

$$1.02(P-2)(Q-1)+(Q-2), \tag{3.3}$$

and

$$1.02(P-2)(Q-1)+1.02(Q-2), \tag{3.4}$$

respectively. It can be seen that the cost is increased by generating canonic RFFT computation (i.e., pushing the twiddle factors) in this case. This is because two real twiddle factor operations are transformed to one complex twiddle factor operation by pushing the twiddle factors, since the cost of a complex twiddle factor operation is more than two times that of the cost of a real twiddle factor operation. As a result, the twiddle factor operations of canonic RFFT computations consume more power than the corresponding non-canonic RFFT computations. Therefore, canonic RFFT structure might not be desirable with respect to the power consumption, when P and Q are both even.

Canonic RFFTs are also advantageous from the architecture perspective. Many parallel pipelined architectures to compute FFT with real inputs have been proposed in the literature [6, 8, 5, 14, 7, 4, 16]. The canonic RFFT structures can also be mapped to either feed-back or feed-forward architectures. In feed-back architectures, registers can be saved by reusing the twiddle factor operation elements, since a portion of the complex samples are computed at each stage. However, real parts and imaginary parts of the samples are separated to fully utilize the hardware in feed-forward architectures. One advantage of canonic RFFT computations is that the corresponding architectures only involve real datapaths. The number of registers can be saved, since they only need to store N signal values for an N-point RFFT during processing cycles. For other non-canonic RFFT architectures, more registers are required for the complex datapaths. For example, the architectures presented in [6, 8] require double the registers where the datapaths are complex as the architectures in [7], which only involve real datapaths.

3.6 Conclusion

This chapter has proposed a novel algorithm to design canonic RFFT computations for any composite length. Twiddle factor transformations can be used to eliminate the redundant samples. We have considered 4 different cases for an $N = P \times Q$ point canonic RFFT: 1) P is odd, Q is odd; 2) P is odd, Q is even; 3) P is even, Q is odd; and 4) P is even, Q is even. It is shown that twiddle factor transformations are required when P is even. The twiddle factors can be reduced by transforming the twiddle factors.

Chapter 4

Canonic Real-Valued FFT Flow-Graphs with Even/Odd Symmetric Signals

4.1 Introduction

This chapter explore the design of canonic FFT flow-graphs for when inputs are real-valued and also even/odd symmetric, referred as REFFT and ROFFT, respectively. The motivation of this work is that linear FIR filter impulse responses are even/odd symmetric. For example, the type I FIR filter has odd number of taps where the values of the taps are even symmetric. A 7-tap type I FIR filter can be given by the impulse response $h[n] = [a, b, c, d, d, c, b]$. As a result, we can improve the computation of $H[k]$ from $h[n]$ by eliminating the redundancies. Therefore, instead of computing $y[n]$ by $x[n] * h[n]$, we can choose to compute the IFFT of $X[k]H[k]$ to obtain the output $y[n]$, as shown in Figure 4.1. The complexity of $H[k]$ can be reduced by the proposed REFFT instead of RFFT.

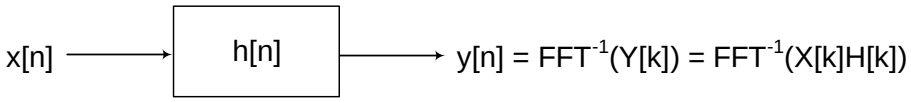

Figure 4.1: Obtain $y[n]$ by computing the IFFT of $X[k]H[k]$.

To the best of our knowledge, simplify the RFFT flow-graph for even/odd symmetric inputs has not been investigated before. The main contribution of this chapter is design of novel algorithms for canonic REFFT/ROFFT. We also propose twiddle factor transformations, which are required to transform the structures to be canonic and to reduce arithmetic complexity. We also discuss the design of canonic REFFT for any composite length.

4.2 Background

4.2.1 Canonic RFFT

Algorithms for generating canonic RFFT have been presented in [15], where the number of signals is guaranteed to be N at each stage for an N-point RFFT. For the 16-point RFFT as shown in Figure 2.2, the outputs are canonic with respect to the number of signals, 16 (i.e., 2 real values and 7 complex values). However, the intermediate stages of the flow-graph are not canonic with respect to the number of signals. For instance, there are 10 real values and 6 complex values, i.e., 22 values in total before the butterfly operations at the second stage. Therefore, Figure 2.2 is not canonic with respect to the number of signal values.

In order to reduce the number of signal values to eliminate redundancy, twiddle factor transformation is required. The "push" transformation of the twiddle factors can be described as shown in Figure 4.2. We can push a factor of W^k from before the butterfly operation to after the butterfly operation to reduce the number of signal values. For example, we can push a factor of W^2 to after the 4th butterfly operation at the third stage. After the twiddle factor transformation, the top input of the butterfly will

be purely real and the bottom input will be purely imaginary. Therefore, the number of signals at this stage can be reduced to 16 from 18, which is canonic with respect to the number of signals. We also need to push the twiddle factors of the 6th, 7th, 8th butterflies at the second stage to obtain the canonic RFFT, as shown in Figure 4.3. After pushing the twiddle factors, the top output of the butterfly can be obtained by appending the two inputs, since the top input is purely real and the bottom input is purely imaginary; while the bottom output can be eliminated, as it will be conjugate symmetric to the top output.

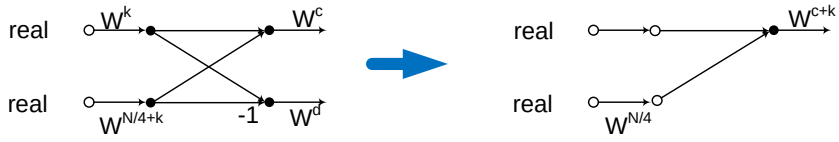

Figure 4.2: Twiddle factor transformation: push.

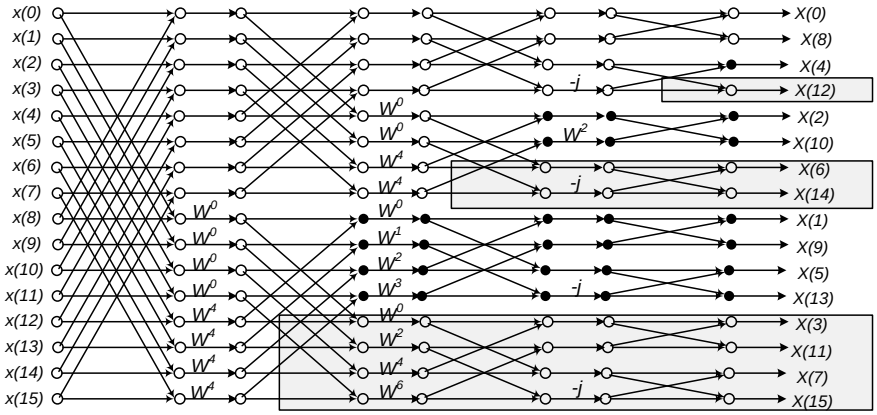

Figure 4.3: A canonic 16-point RFFT.

Note that the canonic structures for a certain size RFFT are not unique. This is because the twiddle factors can be moved from one stage to another if the signals before and after are complex without altering the number of signal values. For example, the twiddle factors after the second stage of the bottom part of Figure 4.3 can also be

pushed to next stage. This operation does not affect the number of signal values for each stage.

4.2.2 REFFT/ROFFT

If the inputs are real and even symmetric, it can be shown that

$$X[k] = X[N-k]. \tag{4.1}$$

Thus, as $X[k] = X^*[N-k]$ (inputs are real), we can get

$$X[k] = X^*[k]. \tag{4.2}$$

In order to satisfy the above equation, $X[k]$ must be purely real, i.e., $X[k]$ is also real and even, and can be computed by the proposed REFFT algorithm.

Similarly, if the inputs are real and odd symmetric, it can be shown that

$$X[k] = -X[N-k]. \tag{4.3}$$

Therefore, we will have

$$X[k] = -X^*[k]. \tag{4.4}$$

In this case, $X[k]$ must be odd symmetric and purely imaginary, can be computed by the proposed ROFFT algorithm.

In conclusion, when the inputs of a RFFT are even symmetric or odd symmetric, the outputs of the RFFT will be purely real or purely imaginary, respectively. This property can also be exploited to reduce the arithmetic complexity of the RFFT, as the $\frac{N}{2}-1$ inputs are redundant. In this chapter, we present algorithms to generate canonic REFFT/ROFFT.

4.3 Canonic REFFT for Power-Of-Two Length

In this section, we present the flow-graphs for REFFT which eliminate the redundancies in general RFFT. The number of signals is also guaranteed to be canonic at each stage, i.e., $\frac{N}{2} + 1$ signals.

4.3.1 4-Point REFFT

A 4-point canonic RFFT flow-graph is shown in Figure 4.4. The nodes marked by ∘ and □ respectively represent purely real and purely imaginary signals. Solid and dashed lines respectively represent purely real and purely imaginary datapaths. In the 4-point RFFT, due to redundancy, the bottom butterfly at the second stage is removed and the computations of real and imaginary parts of $X[1]$ are separated as shown in Figure 4.4.

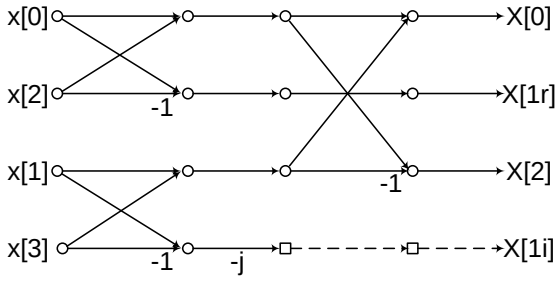

Figure 4.4: Flow-graph of a canonic 4-point RFFT.

If the inputs are even symmetric, i.e., $x[1] = x[3]$, the outputs will be purely real. Therefore, $X[1i]$ in Figure 4.4 will be 0. As a result, we can eliminate the computation of $X[1i]$ so that there will be only 3 signals at the output. Furthermore, we can also remove input $x[3]$ to achieve the canonic property from the beginning. However, we need to multiply $x[1]$ by 2, since $x[1] = x[3]$. The operation can be described by Figure 4.5, where the butterfly operation of two inputs with the same value is replaced by a multiplication of one input by 2. Finally, the flow-graph of a canonic 4-point REFFT can be derived as shown in Figure 4.6.

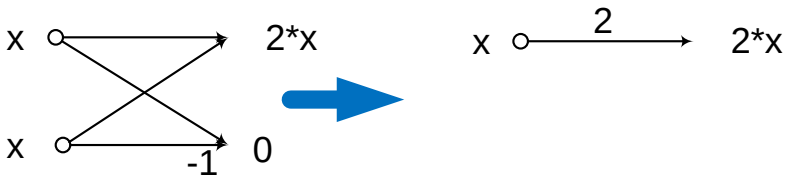

Figure 4.5: Operation to the butterfly with two identical inputs.

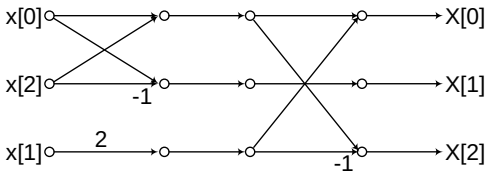

Figure 4.6: Flow-graph of a canonic 4-point REFFT.

4.3.2 8-Point REFFT

It can be observed that flow-graph in the red box of Figure 4.7 is the same as the 4-point RFFT. Therefore, we can eliminate the redundancy of this 4-point RFFT by replacing it with the flow-graph as shown in Figure 4.6. For the bottom half of the first two stages, since $x[1] = x[7]$ and $x[3] = x[5]$, we can remove the bottom butterfly of the first stage. Consequently, the bottom two datapaths at the following stages also need to be removed. It can be calculated that the bottom 4 signal values after the first stage of Figure 4.7 are $x[1]+x[3]$, $x[1]-x[3]$, $x[1]+x[3]$, $x[3]-x[1]$, respectively. The butterfly simplification shown in Figure 4.5 can be applied to the second butterfly operation of the second stage to eliminate redundancy, since the two inputs of the butterfly have the same value. For the twiddle factor operation W^1 after the second stage, if we assume $x[1] - x[3] = a$, then the result of the twiddle factor operation will be

$$(a - a(-j)) * W^1 = \sqrt{2}ae^{j\frac{\pi}{4}}e^{-j\frac{\pi}{4}} = \sqrt{2}a. \tag{4.5}$$

Therefore, the W^1 in Figure 4.7 should be replaced by $\sqrt{2}$, while the imaginary path is removed. The butterfly simplification can be described by Figure 4.8. As a result, the final flow-graph is shown in Figure 4.9.

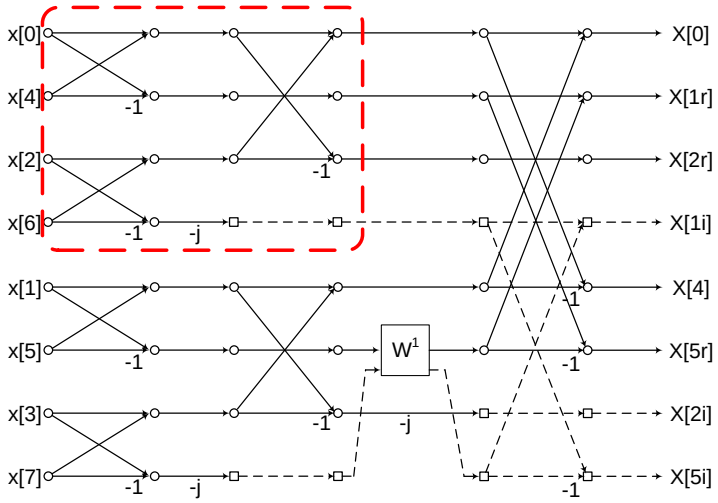

Figure 4.7: Flow-graph of a canonic 8-point RFFT.

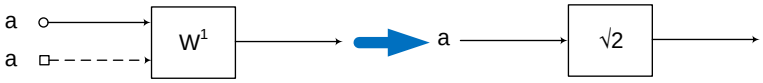

Figure 4.8: Butterfly simplification.

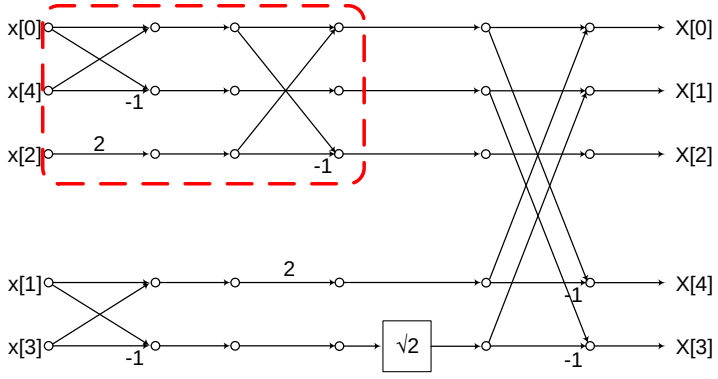

Figure 4.9: Flow-graph of a canonic 8-point REFFT.

4.3.3 16-Point REFFT

A canonic 16-point RFFT is shown in Figure 4.10. In fact, this flow-graph is the same as that of Figure 4.3 if we separate the real and imaginary signals.

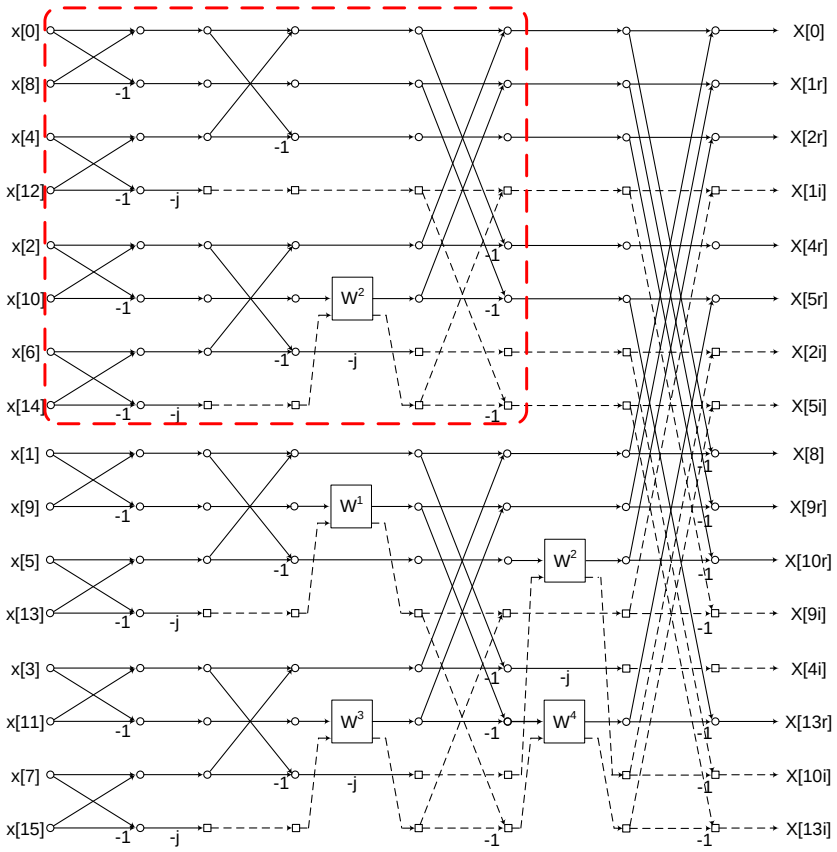

Figure 4.10: Flow-graph of a canonic 16-point RFFT.

Similarly, the top half of the first 3 stages can be reduced to the 8-point REFFT as shown in Figure 4.9. Furthermore, the last $\frac{N}{4}$ inputs can be removed, as these 4 signals are redundant, i.e., $x[1] = x[15]$, $x[3] = x[13]$, $x[5] = x[11]$, and $x[7] = x[9]$. In order to study the required operations to eliminate redundancy, we calculate the signal values of the bottom half in Figure 4.10, as presented in Table 4.1.

Table 4.1: Signal Values of the Bottom Half in Figure 4.10

Position	Input	After 1st Stage	After 2nd Stage	Before 3rd Stage
9th	x[1]	x[1]+x[7]	x[1]+x[7]+x[5]+x[3]	x[1]+x[7]+x[5]+x[3]
10th	x[7]	x[1]-x[7]	x[1]-x[7]	Re[(x[1]-x[7]-(x[5]-x[3])j)*W^1]
11th	x[5]	x[5]+x[3]	x[1]+x[7]-x[5]-x[3]	x[1]+x[7]-x[5]-x[3]
12th	x[3]	x[5]-x[3]	(x[5]-x[3])(-j)	Im[(x[1]-x[7]-(x[5]-x[3])j)*W^1]
13th	x[3]	x[3]+x[5]	x[1]+x[7]+x[5]+x[3]	x[1]+x[7]+x[5]+x[3]
14th	x[5]	x[3]-x[5]	x[3]-x[5]	Re[(x[3]-x[5]-(x[7]-x[1])j)*W^3]
15th	x[7]	x[7]+x[1]	x[3]+x[5]-x[1]-x[7]	(x[3]+x[5]-x[1]-x[7])(-j)
16th	x[1]	x[7]-x[1]	(x[7]-x[1])(-j)	Im[(x[3]-x[5]-(x[7]-x[1])j)*W^3]

Since the 9th signal and the 13th signal before the 3rd stage of Figure 4.10 have the same value, we can remove the butterfly by using the butterfly simplification as shown in Figure 4.5. For the 11th and 15th signals, as the real input and imaginary input of the twiddle factor operation W^2 have the same value, according to Equation (4.5), we can also replace the twiddle factor operation W^2 by $\sqrt{2}$, while the imaginary path is removed.

Now, let's consider the remaining signals, i.e., the 10th, 12th, 14th, and 16th signals. We assume $x[1] - x[7] = b$ and $x[3] - x[5] = c$. For simplicity, we consider $W^1 = p - qj$. Then $W^3 = q - pj$. After calculation, we can get that $Re[(b + cj) * W^1] = Re[(c + bj) * W^3] = bp + cq$ and $Im[(b+cj) * W^1] = -Im[(c+bj) * W^3] = cp - bq$, respectively. It can be seen that the two inputs of second butterfly operation of the bottom half at the third stage have the same value (i.e., the 10th signal and the 14th signal). Therefore, the butterfly simplification shown in Figure 4.5 can be applied to the butterfly operation to eliminate redundancy. For the butterfly operation whose inputs are the 12th signal and the 16th signal, the operation described in Figure 4.11 can be used to reduce the butterfly operation with two opposite value inputs to a single datapath. Note that the twiddle factor operation $W^4 = -j$ after the third stage also needs to be moved to the path of the 12th signal. Consequently, the final flow-graph is obtained, as shown in Figure 4.12.

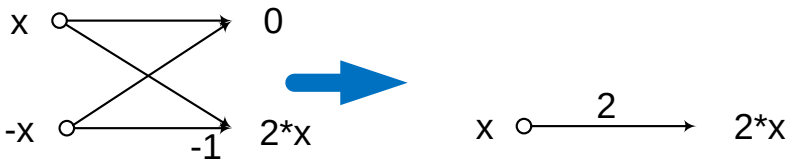

Figure 4.11: Operation to the butterfly with two inputs have opposite values.

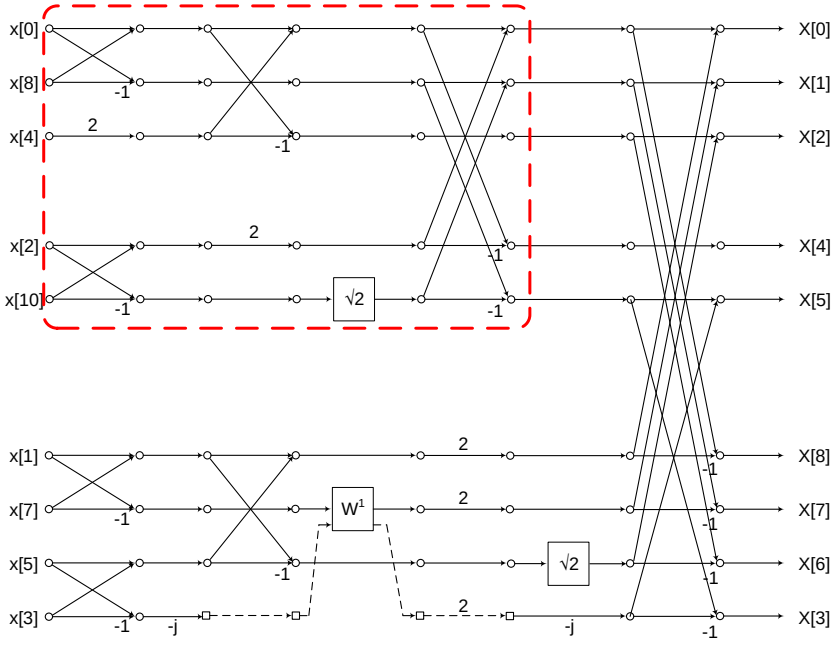

Figure 4.12: Flow-graph of a canonic 16-point REFFT.

4.3.4 Generalization to $N = 2^n$-Point DIF REFFT

In the above sections, we have illustrated that a canonic N-point REFFT can be derived from a canonic $\frac{N}{2}$-point REFFT. From these examples, the proposed algorithm can be summarized from previous sections that (assume we already have the flow-graph of a canonic $\frac{N}{2}$-point REFFT):

Algorithm 3 Generation of a Canonic $N = 2^n$-Point REFFT

1. For an N-point RFFT, the top half is same as an $\frac{N}{2}$-point RFFT. Therefore, this part flow-graph can be simply reduced to the canonic $\frac{N}{2}$-point REFFT.
2. Since we need to maintain $\frac{N}{2}+1$ signal values and $\frac{N}{4}+1$ signals have already been eliminated in the canonic $\frac{N}{2}$-point REFFT, we need to remove $\frac{N}{4}$ more signals.
3. Since the last quarter of the inputs are redundant, we can remove these $\frac{N}{4}$ signals.
4. Since the last quarter of the inputs are removed, the butterfly operations at the $(n-1)$st stage of the bottom half are all removed. If the two inputs of the butterfly have the same value, the operation described in Figure 4.5 needs to be performed. If the two inputs of a butterfly operation have the opposite values, the operation described in Figure 4.11 needs to be performed. The result is that all the datapaths of the third quarter at the $(n-1)$st stage except the $(\frac{5N}{8}+1)$st signal become multiplications of 2.
5. The third quarter flow-graph before the $(n-1)$st stage is unmodified.
6. When $N \geq 8$, the twiddle factor operation $W_N^{\frac{N}{8}}$ after the $(\frac{5N}{8}+1)$st signal at the $(n-1)$st stage needs to be replaced by $\sqrt{2}$.
7. For the last stage, there are butterfly operations before output pairs $X[k]$ and $X[\frac{N}{2}-k]$, where $0 \leq k \leq \frac{N}{4}-1$.

Therefore, due to the regularity of the canonic RFFT flow-graph, the canonic RFFT flow-graph can be extended for any $N = 2^n$-point REFFT recursively.

Based on the patterns presented above, given a canonic 32-point RFFT as shown in Figure 4.13, a 32-point REFFT is shown in Figure 4.14. In this structure, the number of signal values computed at each stage or the output is 17; thus, this structure is canonic.

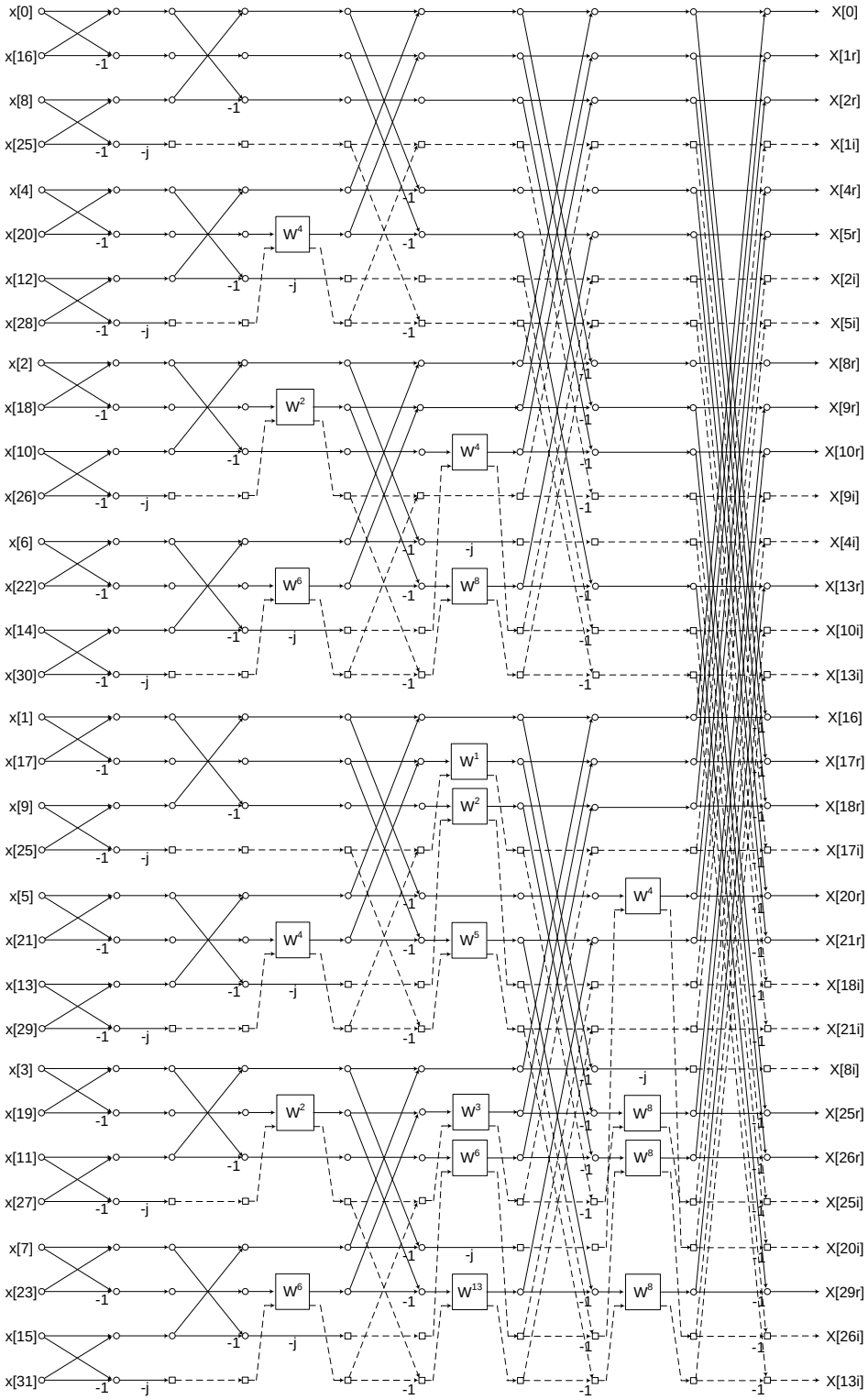

Figure 4.13: Flow-graph of a canonic 32-point RFFT.

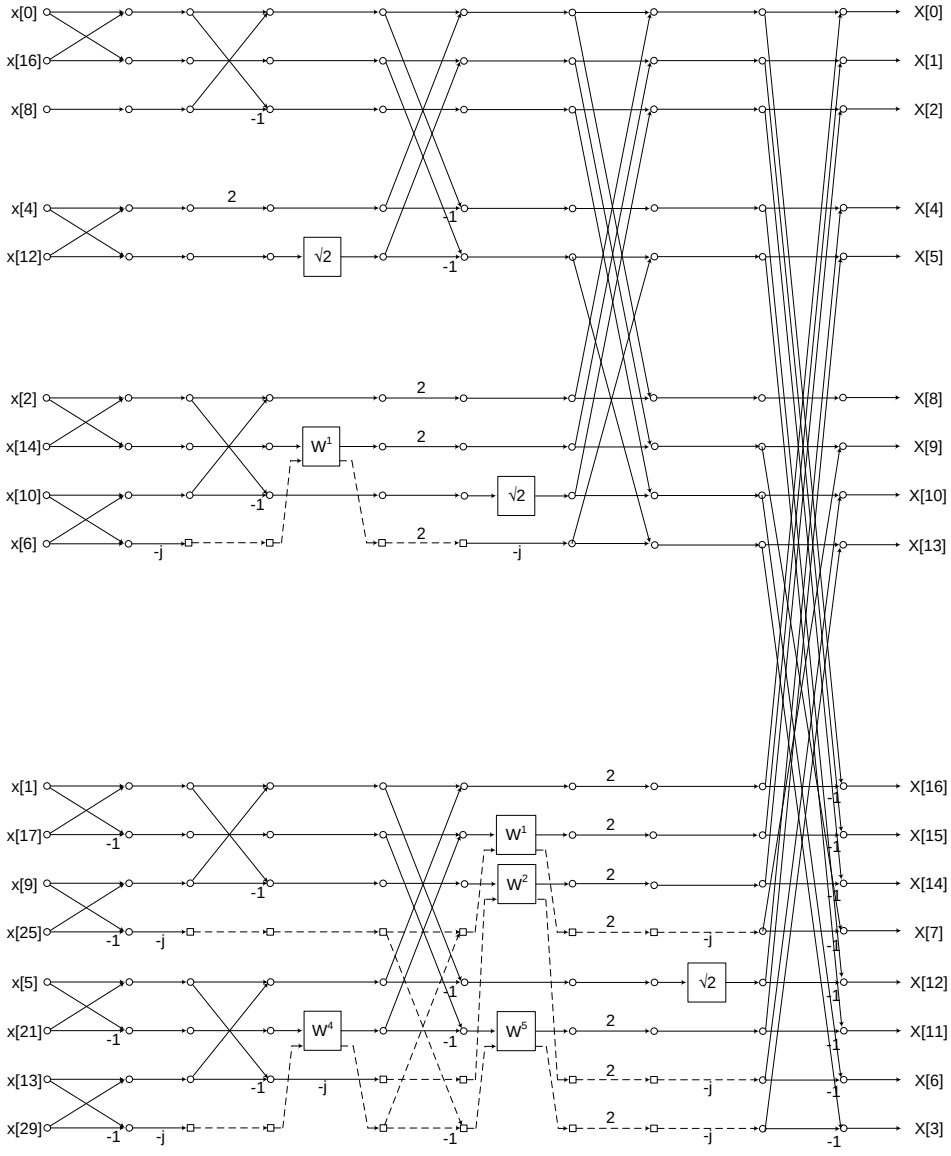

Figure 4.14: Flow-graph of a canonic 32-point REFFT.

4.4 Pre-Processing

4.4.1 Canonic Property

In fact, the canonic RFFTs presented above are all obtained from DIF FFTs by twiddle factor transformations as described in [15]. For the canonic RFFTs generated from DIT FFTs, we cannot derive a canonic REFFT directly. For example, we consider the canonic 16-point DIT RFFT as shown in Figure 4.15. The first 3 stages of the top half flow-graph can also be reduced to the canonic 8-point REFFT as shown in Figure 4.9. We calculate the bottom half signal values in Figure 4.15 as shown in Table 4.2. Note that $W^2 = \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j$. However, in this case, the 10th signal and the 14th signal are neither the same nor opposite. Therefore, we cannot remove this butterfly whose inputs are the 10th signal and the 14th signal by replacing the butterfly operation with a multiplication of 2. Furthermore, as the two input values are $x[1]-x[7]$ and $\frac{\sqrt{2}}{2}(x[3]-x[5]-x[7]+x[1])$, respectively, the butterfly operation cannot be reduced to a multiplication with another value. Similarly, the butterfly operation whose inputs are the 12th signal and the 16th signal also cannot be removed. Therefore, the canonic property cannot be achieved, as the number of signals before the 3rd stage will be greater than $\frac{16}{2} + 1 = 9$.

Table 4.2: Signal Values of the Bottom Half in Figure 4.15

Position	Input	After 1st Stage	After 2nd Stage	Before 3rd Stage
9th	x[1]	x[1]+x[7]	x[1]+x[7]+x[5]+x[3]	x[1]+x[7]+x[5]+x[3]
10th	x[7]	x[1]-x[7]	x[1]-x[7]	x[1]-x[7]
11th	x[5]	x[5]+x[3]	x[1]+x[7]-x[5]-x[3]	x[1]+x[7]-x[5]-x[3]
12th	x[3]	x[5]-x[3]	(x[5]-x[3])(-j)	(x[5]-x[3])(-j)
13th	x[3]	x[3]+x[5]	x[1]+x[7]+x[5]+x[3]	x[1]+x[7]+x[5]+x[3]
14th	x[5]	x[3]-x[5]	x[3]-x[5]	$\frac{\sqrt{2}}{2}$(x[3]-x[5]-x[7]+x[1])
15th	x[7]	x[7]+x[1]	x[3]+x[5]-x[1]-x[7]	(x[3]+x[5]-x[1]-x[7])(-j)
16th	x[1]	x[7]-x[1]	(x[7]-x[1])(-j)	$\frac{\sqrt{2}}{2}$(-x[3]+x[5]-x[7]+x[1])j

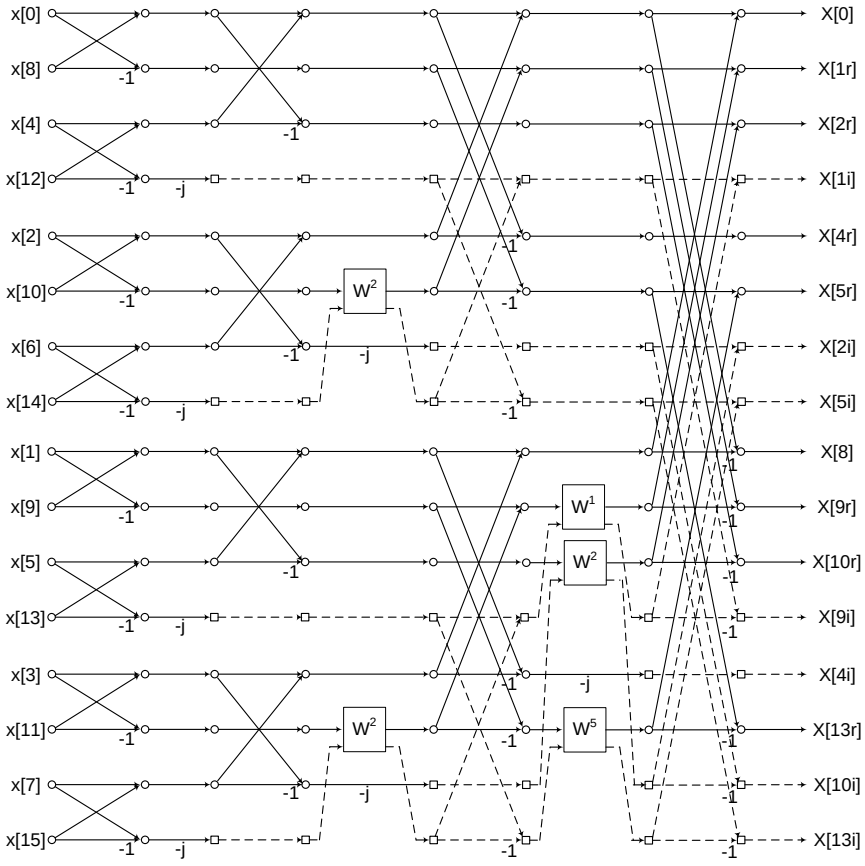

Figure 4.15: Flow-graph of a canonic 16-point DIT RFFT.

4.4.2 Pull the Twiddle Factors

Similar to the twiddle factor transformation as described in Figure 4.2, we can perform twiddle factor transformation to turn the 16-point RFFT flow-graph in Figure 4.15 into the flow-graph in Figure 4.10. However, the operation will be pulling the twiddle factors to previous stages instead of pushing the twiddle factors to later stages, as shown in Figure 4.16. For example, as shown in Figure 4.15, we can pull W^1 from after the third stage to before the third stage, which leads to the flow-graph as shown in Figure 4.10.

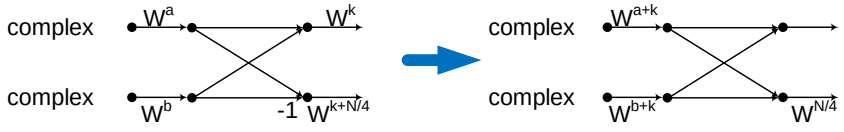

Figure 4.16: Twiddle factor transformation: pull.

According to the work in [15], as the signal values before and after the butterfly operation are both complex, the twiddle factors are free to move. Furthermore, it can be shown that since

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{nk} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{(2n+1)k} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{2nk} \\ & \qquad k = 0, 1, ..., N-1, \end{aligned} \tag{4.6}$$

the twiddle factors after the $(n-1)$st stage at the bottom half will be W_N^k at the path where the output is $X[\frac{N}{2}+k]$. The two output paths of the butterfly operation at the $(n-1)$st stage at the bottom half can be expressed as $X[\frac{N}{2}+k]$ and $X[\frac{N}{2}+\frac{N}{4}+k]$, respectively. Thus, the twiddle factors after the $(n-1)$st stage always follow the pattern as shown in the left butterfly in Figure 4.16 (i.e., W_N^k and $W_N^{k+\frac{N}{4}}$), if the complex

butterfly operation has not been removed in the canonic N-point RFFT. Note that the two twiddle factors will still have the same pattern even if the twiddle factors have already been transformed: as shown in Figure 4.17, after transforming W^m, the two twiddle factors after the butterfly can still be $W_N^{k'}$ and $W_N^{k'+\frac{N}{4}}$, if we consider $k' = k-m$.

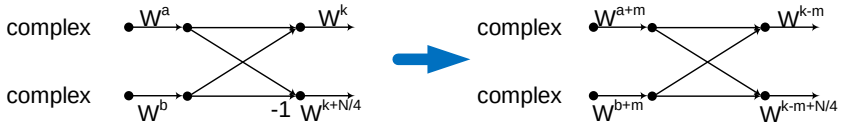

Figure 4.17: The two twiddle factors after the $n-1$th stage at the bottom half will still have the same pattern even if the twiddle factors have already been transformed.

In conclusion, the goal of the twiddle factor transformation is to make sure the twiddle factor operations before stage n are only $W_N^{\frac{N}{4}}$ or $W_N^{\frac{N}{8}}$ (only the twiddle factor after the $(\frac{7N}{8}+1)$st signal at the $(n-1)$st stage is $W_N^{\frac{N}{8}}$, which can be replaced by $\sqrt{2}$), when we extend a canonic $\frac{N}{2}$-point REFFT to a canonic N-point REFFT. If the twiddle factor is $W_N^{\frac{N}{4}} = -j$, the twiddle factor essentially transforms a purely imaginary signal to a purely real signal or transforms a purely real signal to a purely imaginary signal. We know that imaginary signals will equal to 0, as the inputs are even symmetric. Therefore, if the twiddle factor after the butterfly is removed or transformed to $W_N^{\frac{N}{4}}$, then one of the two outputs of the butterfly operation will be 0. In this case, we can eliminate the butterfly operation according either Figure 4.5 or Figure 4.11.

It can be concluded that twiddle factor transformation is helpful in eliminating butterfly operation, which needs to be applied to the RFFT flow-graph before performing the algorithm to generate canonic REFFT.

4.5 Canonic ROFFT for Power-Of-Two Length

In the previous sections, we have presented the approach to generate canonic REFFT. In this section, we present the algorithm to generate canonic ROFFT. As discussed in Section 4.2, the outputs of the RFFT will be purely imaginary if the inputs

are odd symmetric, i.e., $x[k] = -x[N-k]$, where $1 \leq k \leq \frac{N}{2} - 1$. Note that in order to ensure purely imaginary outputs, $x[0]$ and $x[\frac{N}{2}]$ should be equal to 0. Therefore, these two signals can also be removed. As a result, for an N-point ROFFT, a canonic flow-graph should only have $\frac{N}{2} - 1$ signal values at each stage. For example, for a canonic 4-point RFFT as shown in Figure 4.4, the flow-graph for the RFFT when the inputs are odd symmetric only has 1 signal, as shown in Figure 4.18.

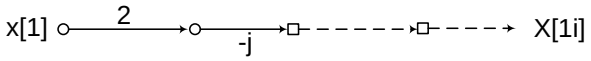

Figure 4.18: Flow-graph of a canonic 4-point ROFFT.

When eliminating the redundancies, the difference is that we need to keep imaginary paths, while removing real paths. Therefore, when we extend from $\frac{N}{2}$-point to N-point, we can choose to remove the third quarter of the inputs instead of the last quarter. The algorithm for generating canonic N-point ROFFT from a canonic $\frac{N}{2}$-point ROFFT is presented in Algorithm 4. Any $N = 2^n$-point ROFFT can be derived by using the Algorithm 4.

Algorithm 4 Generation of a Canonic $N = 2^n$-Point ROFFT with

1. For an N-point RFFT, the top half is same as an $\frac{N}{2}$-point RFFT. Therefore, this part flow-graph can be simply reduced to the canonic $\frac{N}{2}$-point ROFFT.
2. Since we need to maintain $\frac{N}{2} - 1$ signal values and $\frac{N}{4} - 1$ signals have already been eliminated in the canonic $\frac{N}{2}$-point ROFFT, we need to remove $\frac{N}{4}$ more signals.
3. Since the third quarter of the inputs are redundant, we can remove these $\frac{N}{4}$ signals.
4. Since the third quarter of the inputs are removed, the butterfly operations at the $(n-1)$st stage of the bottom half are all removed. If the two inputs of the butterfly have the same value, the operation described in Figure 4.5 needs to be performed. If the two inputs of a butterfly operation have the opposite values, the operation described in Figure 4.11 needs to be performed. The result is that all the datapaths of the last quarter at the $(n-1)$st stage except the $(\frac{7N}{8} + 1)$st signal become multiplications of 2.
5. The last quarter flow-graph before the $(n-1)$st stage is unmodified.
6. When $N \geq 8$, the twiddle factor operation $W_N^{\frac{N}{8}}$ after the $(\frac{7N}{8} + 1)$st signal at the $(n-1)$st stage needs to be replaced by $\sqrt{2}$.
7. For the last stage, there are butterfly operations before output pairs $X[k]$ and $X[\frac{N}{2} - k]$, where $0 \leq k \leq \frac{N}{4} - 1$.

Given a canonic 16-point RFFT as shown in Figure 4.10, according to the Algorithm 4, the flow-graph of a canonic 16-point ROFFT is shown in Figure 4.19. Note that as discussed above, before performing the Algorithm 4, we need to pull the twiddle factors from after the $(n-1)$st stage to before the $(n-1)$st stage if needed.

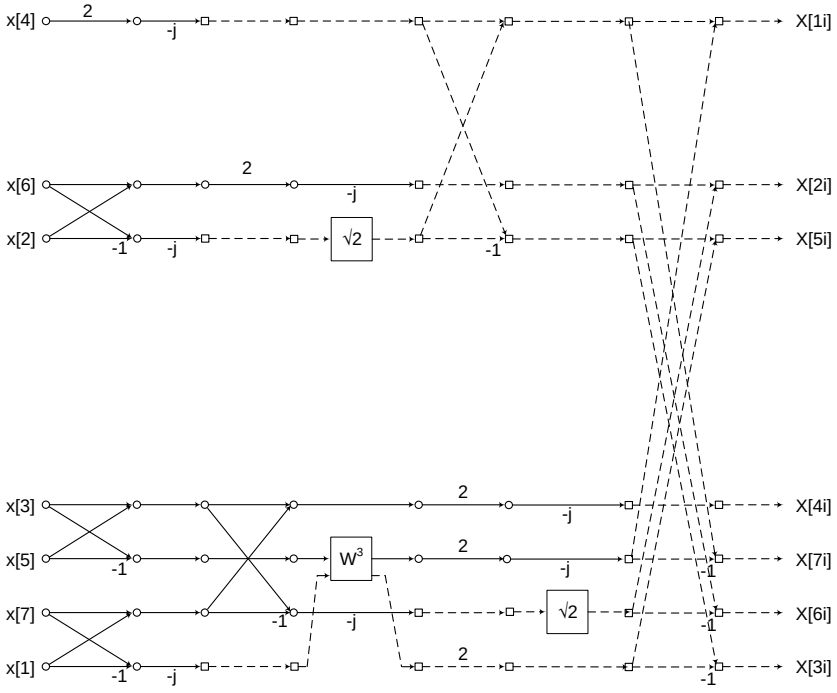

Figure 4.19: Flow-graph of a canonic 16-point ROFFT.

4.6 REFFT for Any Composite Length

The algorithm for generating canonic RFFT computation for any composite length has been presented in Chapter 3. In this section, we consider the design of canonic REFFT computation for any composite length. For an M-point REFFT, we need to ensure the number of real samples at each stage is equal to $\lfloor \frac{M}{2} \rfloor + 1$. As shown in Figure 4.20(a), we should remove $\frac{M-1}{2}$ real signals and keep the other $\frac{M-1}{2}$ real signals and $X[0]$ when M is odd. When M is even as shown in Figure 4.20(b), we need

to remove $\frac{M-2}{2}$ real signals and keep the other $\frac{M-2}{2}$ real signals and $X[0]$ and $X[\frac{N}{2}]$. Consider an N-point REFFT where $N = P \times Q$. To derive the N-point REFFT, we begin to consider the N-point RFFT that constitutes Q P-point RFFTs at the first stage and P Q-point RFFTs at the second stage. We discuss the process for 4 different cases, i.e., 1) P is odd, Q is odd; 2) P is odd, Q is even; 3) P is even, Q is odd; and 4) P is even, Q is even.

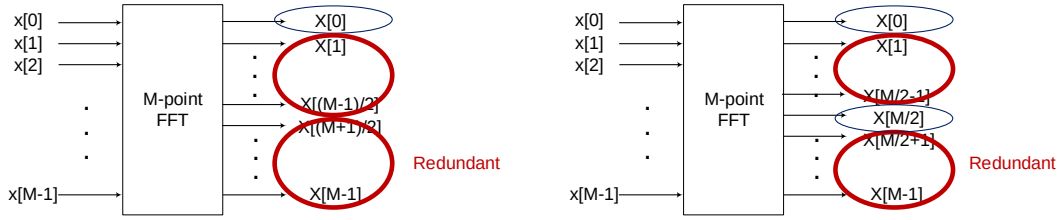

Figure 4.20: M-point REFFT: (a) M is odd, (b) M is even.

4.6.1 Subcomponents

If we consider a $P \times Q$ RFFT structure with even symmetric inputs, the inputs of each P-point RFFT at the first stage can be summarized in Table 4.3.

Table 4.3: Inputs of Each P-point RFFT at the First Stage

	Inputs
1st P-point RFFT	x[kQ], $0 \leq k \leq P-1$
2nd P-point RFFT	x[kQ+1], $0 \leq k \leq P-1$
...	...
(m)th P-point RFFT	x[kQ+m-1], $0 \leq k \leq P-1$
...	...
(Q)th P-point RFFT	x[kQ+Q-1], $0 \leq k \leq P-1$

It can be seen that only the inputs of the first P-point RFFT are even symmetric, as $x_P[k] = x_P[N-k]$. Note that the $x_P[k]$ represents the input order in each P-point RFFT. However, for other P-point RFFTs, the inputs are not even symmetric. When Q

is even, the inputs of the $(\frac{Q}{2}+1)$st P-point RFFT are $x[kQ+\frac{Q}{2}]$, where $0 \leq k \leq P-1$, which follow the pattern of $x_P[k] = x_P[P-1-k]$.

Moreover, it can also be seen that inputs of the (m)th P-point RFFT and the inputs of $(Q+2-m)$th P-point RFFT are reverse-ordered versions of each other, where $2 \leq m \leq Q$. The relation of the inputs of the two P-point RFFTs can be expressed as

$$x_2[k] = x_1[(-(k+1))_N]. \tag{4.7}$$

Note that the actual interval of the inputs of the P-point RFFT is Q. Therefore, according to the DFT time reversal and time shift properties, we can obtain

$$X_2[k] = X_1[(-k)_N] \times W^{-kQ}, \tag{4.8}$$

which leads to the relation that $X_2[0] = X_1[0]$ and $X_2[k] = X_1[N-k] \times W^{-kQ}$, where $1 \leq k \leq N-1$. The twiddle factors after the first stage for the (m)th P-point RFFT are $W^{(m-1)k}$, where $1 \leq k \leq P-1$. As a result, the values after the twiddle factor operations of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT can be expressed by $XW_1[k] = X_1[k]W^{(m-1)k}$ and $XW_2[k] = X_2[k]W^{(Q-m+1)k} = X_1[(-k)_N] \times W^{-kQ}W^{(Q-m+1)k} = X_1[(-k)_N] \times W^{(-m+1)k}$, respectively.

Since the we know that the outputs for RFFT are conjugate symmetric:

$$X[k] = X^*[N-k]. \tag{4.9}$$

Then, for $1 \leq k \leq N-1$, we have

$$\begin{aligned} XW_2[k] &= X_1[(-k)_N] \times W^{(-m+1)k} = X_1[N-k] \times W^{(-m+1)k} \\ &= X_1^*[k] \times W^{(-m+1)k}. \end{aligned} \tag{4.10}$$

Therefore, we can conclude that the values of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT after the twiddle factor operations are conjugates of each other:

$$XW_1[k] = XW_2^*[k]. \tag{4.11}$$

Moreover, as $W^{(m-1)k}$ and $W^{(m-1)(N-k)}$ are also conjugate of each other

$$XW_1[k] = XW_1^*[N-k]. \tag{4.12}$$

As a result, one of these two P-point RFFT is redundant which can be eliminated, while the outputs of the eliminated P-point RFFT can be obtained by simply conjugating the outputs of the retained P-point RFFT.

Before considering the 4 cases, we need to consider the designs of the following 3 FFT data-flows. Note that we only briefly discuss the approaches to remove redundancies of the FFTs with these 3 input patterns in this chapter. Future work will be directed towards addressing the complete algorithms for generating canonic FFTs with these input patterns.

FFT with Hermitian Symmetric Inputs (HFFT)

If the inputs of an FFT are Hermitian symmetric, the output will be purely real. We can use the designs of IFFT of Hermitian symmetric signals (RIFFT) such as the work presented in [7] to compute the HFFT. Note that the outputs of the RIFFT need to be reordered to obtain the outputs of the corresponding HFFT. We do not discuss the detailed designs in this chapter.

Odd Size RFFT with Inputs $x_P[k] = x_P[P-1-k]$

As we have discussed above, when Q is even, the inputs of the $(\frac{Q}{2}+1)$st P-point RFFT are $x[kQ+\frac{Q}{2}]$, where $0 \leq k \leq P-1$, which follow the pattern of $x_P[k] = x_P[P-1-k]$ (e.g., $[a,b,c,d,c,b,a]$, when $P=7$). Furthermore, the outputs of the P-point RFFT connect to twiddle factors $W_N^{\frac{Q}{2}k}$, respectively. We can circularly shift the inputs of an odd size P-point RFFT whose inputs have the pattern of $x_P[k] = x_P[P-1-k]$ by $\frac{P-1}{2}Q$ to an odd size REFFT. The circular time shift property can be expressed by

$$x[n] \leftrightarrow X[k], \tag{4.13}$$

$$x[(n-n_0)_N] \leftrightarrow X[k]W_N^{n_0 k}. \tag{4.14}$$

Therefore, if we shift the inputs of an odd size P-point RFFT whose inputs have the pattern of $x_P[k] = x_P[P-1-k]$ by $\frac{P-1}{2}Q$, the outputs will be $X_P[k]W_N^{\frac{P-1}{2}Qk}$, as the interval of the inputs is Q. If the outputs of the RFFT connect to twiddle factors $W_N^{\frac{Q}{2}k}$, the values after the twiddle factor operations can be expressed by $X_P[k]W_N^{\frac{P-1}{2}Qk}W_N^{\frac{Q}{2}k} = X_P[k]W_N^{\frac{PQ}{2}k} = X_P[k](-1)^k$, where $X_P[k]$ here are the outputs of a P-point REFFT. In this case, the values after the twiddle factor operations will be all purely real. The complete operation is shown in Figure 4.21. Therefore, we only need to keep $\frac{P+1}{2}$ signals for this P-point RFFT; the deleted $\frac{P-1}{2}$ values after the twiddle factor operation can be obtained by simply alternately negating $X_P[k](-1)^k$, where $1 \le k \le \frac{P-1}{2}$.

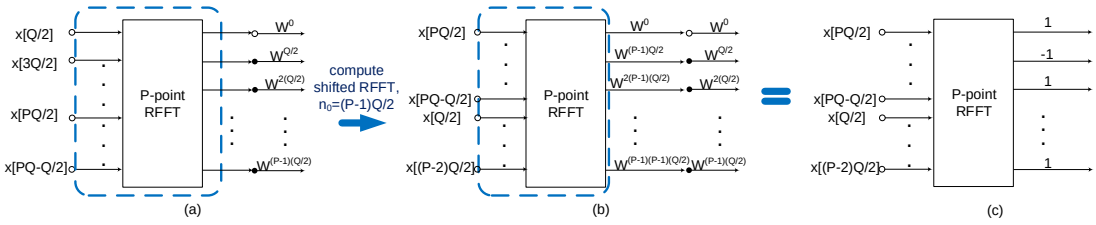

Figure 4.21: (a) The original odd size P-point FFT whose inputs have the pattern of $x_P[k] = x_P[P-1-k]$, (b) compute the shifted FFT, $n_0 = \frac{P-1}{2}Q$, (c) the final P-point RFFT whose inputs are even symmetric.

Even Size RFFT with Inputs $x_P[k] = x_P[N-1-k]$

When P and Q are both even, the inputs of the $(\frac{Q}{2}+1)$st P-point RFFT also follow the pattern of $x_P[k] = x_P[P-1-k]$ (e.g., $[a,b,c,d,d,c,b,a]$, when $P=8$), while the outputs also connect to twiddle factors $W_N^{\frac{Q}{2}k}$, respectively. In this case, we can consider a $\frac{P}{2} \times 2$ structure as shown in Figure 4.22(b). Then we can pull the twiddle factors before the butterfly operations, as shown in Figure 4.22(c).

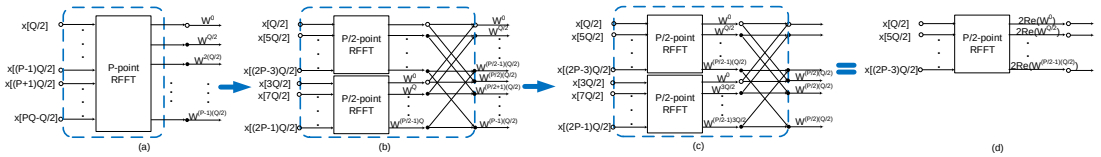

Figure 4.22: (a) The original even size P-point FFT whose inputs have the pattern of $x_P[k] = x_P[P-1-k]$, (b) the P-point RFFT is considered as a $\frac{P}{2} \times 2$-point RFFT, (c) pull the twiddle factors, (d) the final structure which only has $\frac{P}{2}$ signals.

It can be seen from Figure 4.22(c) that the inputs of the two $\frac{P}{2}$-point RFFTs are reverse-ordered. According to Equation (4.8), we can get the relation of the outputs of the two $\frac{P}{2}$-point RFFTs as below (note the interval of the inputs is $2Q$ in this case):

$$X_2[k] = X_1[(-k)_N] \times W^{-k2Q}, \tag{4.15}$$

Therefore, the values of the bottom $\frac{P}{2}$-point RFFT after twiddle factor operation as shown in Figure 4.22(c) are equal to $X_2[k]W^{k\frac{3}{2}Q} = X_1[(-k)_N] \times W^{-k2Q}W^{k\frac{3}{2}Q} = X_1[(-k)_N]W^{-k\frac{Q}{2}}$. Furthermore, according to Equation (4.9), we can obtain

$$X_2[k]W^{k\frac{3}{2}Q} = X_1[(-k)_N]W^{-k\frac{Q}{2}} = X_1^*[k]W^{-k\frac{Q}{2}}, \tag{4.16}$$

which is conjugate of the values of the top $\frac{P}{2}$-point RFFT after twiddle factor operation as shown in Figure 4.22(c), i.e., $X_1[k]W^{k\frac{Q}{2}}$. Therefore, the inputs of the each butterfly operation as shown Figure 4.22(c) are conjugates of each other. Consequently, we can remove the bottom half of the P-point RFFT to eliminate redundancy as shown in Figure 4.22(d). The twiddle factor operations $W^{\frac{Q}{2}k}$ needs to be replaced by $2Re(W^{\frac{Q}{2}k})$, where $1 \leq k \leq \frac{P}{2} - 1$.

4.6.2 $N = P \times Q$, P is odd, Q is odd

When P and Q are both odd, we need to make sure there are only $\left\lfloor \frac{PQ}{2} \right\rfloor + 1 = \frac{PQ+1}{2}$ signals at each stage. At the first stage, the inputs of the first P-point RFFT are even

symmetric. Therefore, we only need to keep $\frac{P+1}{2}$ outputs due to redundancy as described above. For the other $Q-1$ P-point RFFTs, we only need to keep half of them, since the values of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT after the twiddle factor operations are conjugate of each other, where $2 \leq m \leq Q$. In our design, we choose to keep the first half and delete the second half $\frac{Q-1}{2}$ P-point RFFTs. For each P-point RFFT, we use the corresponding canonic RFFT structure, i.e., the number of output signals is equal to P. Therefore, there are $\frac{P+1}{2} + \frac{Q-1}{2} \times P = \frac{PQ+1}{2}$ signals after the first stage, which achieves the canonic property.

At the second stage, there is one Q-point RFFT whose inputs are $X_P[0]$'s from the P-point RFFTs at the first stage. Since the outputs $X_P[0]$ from the (m)th P-point RFFT and the inputs of $(Q+2-m)$th P-point RFFT have the same value, the inputs of the first Q-point RFFT at the second stage are also even symmetric. Besides, there are $\frac{P-1}{2}$ Q-point FFTs whose inputs are $XW_P[m]$'s from the P-point RFFTs, where $1 \leq m \leq \frac{P-1}{2}$. Since the values of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT after the twiddle factor operations are conjugates of each other, where $2 \leq m \leq Q$, the inputs of each Q-point FFT are Hermitian symmetric. According to Section 4.6.1, the outputs of these FFTs whose inputs are Hermitian symmetric are purely real. Therefore, we can reduce these Q-point FFTs to HFFTs. As a result, there will be Q signals after each Q-point HFFT. Consequently, there are also only $Q + \frac{P-1}{2} \times Q = \frac{PQ+1}{2}$ signals after the second stage so that the canonic property is achieved.

For example, we consider the two 15-point canonic RFFTs as shown in Figure 4.23 and Figure 4.24, respectively. The complex signals are marked bold.

For the 3×5 structure as shown in Figure 4.23, we could remove 1 sample of the first 3-point RFFT at the first stage, since the inputs are even symmetric. Furthermore, we can remove the last 2 3-point RFFTs, as they are redundant. As a result, there are $2 + 2 \times 3 = 8$ signals at the first stage. At the second stage, we can remove 2 samples of the first 5-point RFFT, as its inputs are also even symmetric. The second 5-point FFT

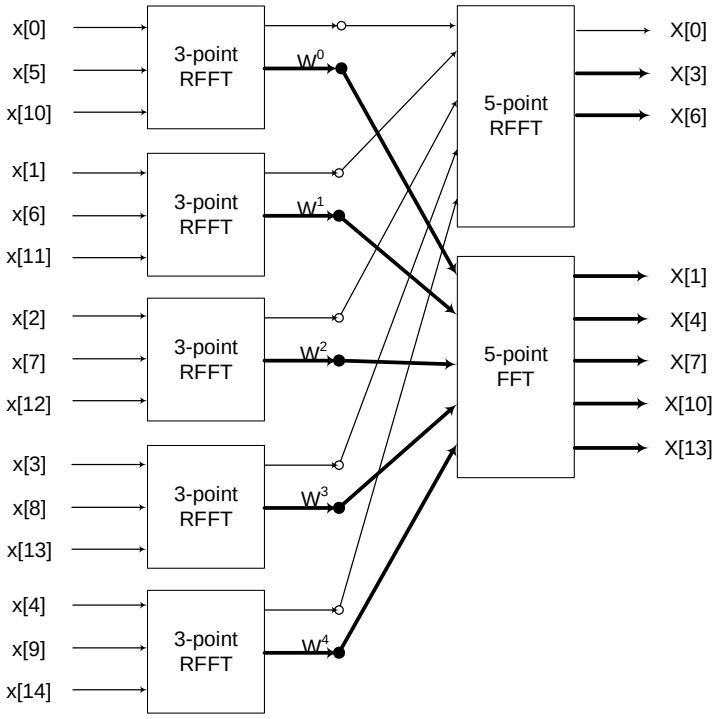

Figure 4.23: A 15-point canonic RFFT, where $P = 3, Q = 5$.

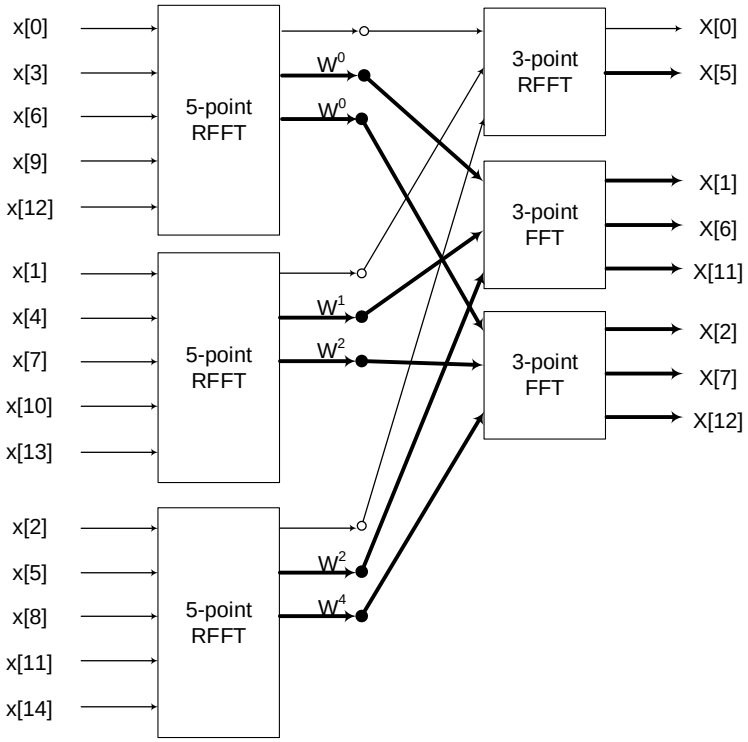

Figure 4.24: A 15-point canonic RFFT, where $P = 5, Q = 3$.

can be reduced to HFFT, since the inputs are Hermitian symmetric. Thus, there are $3+5=8$ signals after the second stage, which is canonic with respect to the number of signals. The final flow-graph is shown in Figure 4.25.

Similarly, for the 5×3 structure as shown in Figure 4.24, we can also design a canonic REFFT as shown in Figure 4.26.

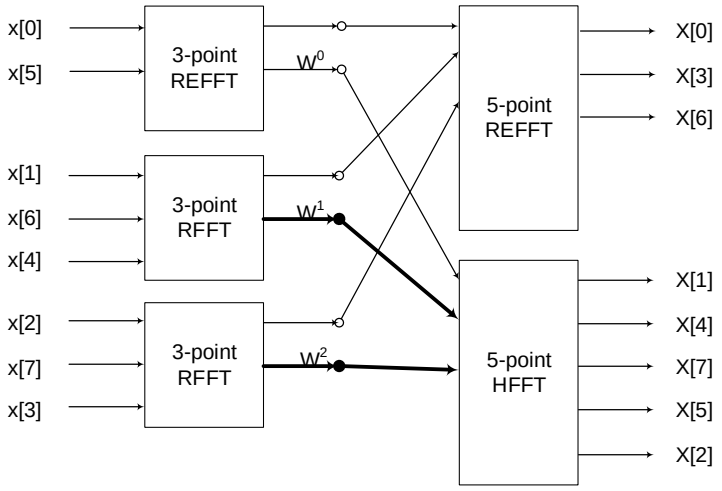

Figure 4.25: A 15-point canonic REFFT, where $P=3, Q=5$.

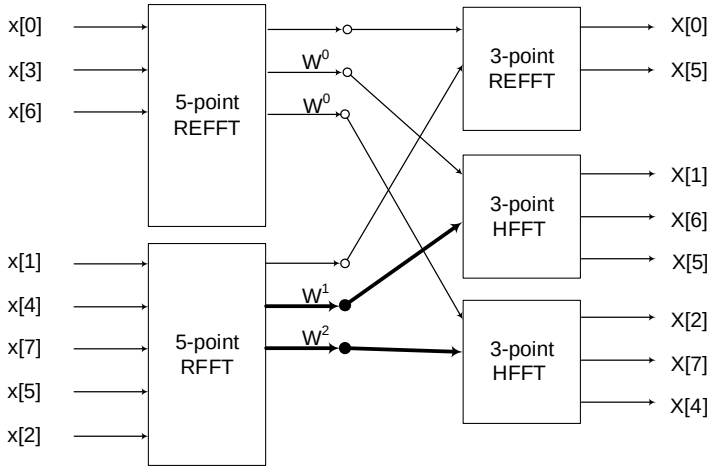

Figure 4.26: A 15-point canonic REFFT, where $P=5, Q=3$.

4.6.3 $N = P \times Q$, P is odd, Q is even

If P is odd and Q is even, we need to make sure there are $\left\lfloor \frac{PQ}{2} \right\rfloor + 1 = \frac{PQ}{2} + 1$ signals at each stage. At the first stage, the inputs of the first P-point RFFT are even symmetric, while the inputs of the $(\frac{Q}{2}+1)$st P-point RFFT have the pattern of $x_P[k] = x_P[P-1-k]$, which can also be transformed to a P-point REFFT as described in Section 4.6.1. For the other $Q-2$ P-point RFFTs, we only need to keep half of them, since the values of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT after the twiddle factor operations are conjugates of each other, where $2 \leq m \leq Q$ and $m \neq \frac{Q}{2}+1$. As a result, there are $\frac{P+1}{2} + \frac{P+1}{2} + \frac{Q-2}{2} \times P = \frac{PQ}{2} + 1$.

At the second stage, the inputs of the first Q-point RFFT are even symmetric. For the remaining $\frac{P-1}{2}$ Q-point RFFT, the inputs are Hermitian symmetric. Therefore, there are $\frac{Q}{2} + 1 + \frac{P-1}{2} \times Q = \frac{PQ}{2} + 1$ signals at the output, which achieves the canonic property.

For example, we consider the $3 \times 2 = 6$-point canonic RFFT as shown in Figure 4.27. The corresponding canonic REFFT is shown in Figure 4.28. According to Section 4.6.1, the inputs of the second 3-point RFFT at the first stage can be shifted by 2. As a result, this RFFT can be reduced to the 3-point REFFT, as $x[1] = x[5]$. Note that $(-1)^k$ needs to be added after each output of this 3-point RFFT. It can be seen that there are 4 signals at each stage, which is canonic with respect to the number of signals.

4.6.4 $N = P \times Q$, P is even, Q is odd

If P is even and Q is odd, we also need to make sure there are $\left\lfloor \frac{PQ}{2} \right\rfloor + 1 = \frac{PQ}{2} + 1$ signals at each stage. At the first stage, the inputs of the first P-point RFFT are even symmetric. Besides, we only need to keep half of the other $Q-1$ P-point RFFTs due to redundancy. Therefore, the number of the signals after the first stage is $\frac{P}{2}+1+\frac{Q-1}{2}\times P = \frac{PQ}{2} + 1$, which is canonic.

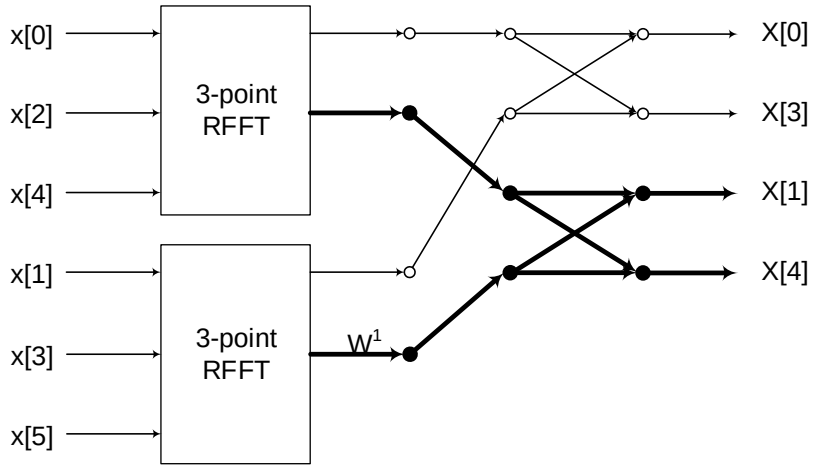

Figure 4.27: Flow-graph of a canonic 6-point RFFT, where $P = 3, Q = 2$.

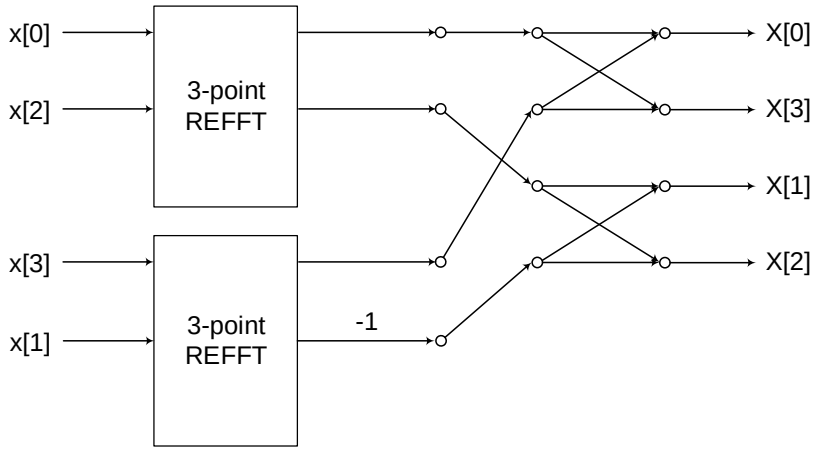

Figure 4.28: Flow-graph of a canonic 6-point REFFT, where $P = 3, Q = 2$.

The inputs of $(\frac{P}{2}+1)$st Q-point RFFT at the second stage are from $X_P[\frac{P}{2}]$'s of each P-point RFFT at the first stage, which are purely real. Note that we can circularly shift this Q-point RFFT in frequency to eliminated redundancy according to the following property:

$$x[n] \leftrightarrow X[k], \tag{4.17}$$

$$W_N^{-k_0 n} x[n] \leftrightarrow X[(k-k_0)_N], \tag{4.18}$$

which is referred as modulation transformation in Chapter 3. Additionally, we have shown that values of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT after the twiddle factor operations are conjugates of each other. Consequently, we can obtain that the $XW[\frac{P}{2}]$ values of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT after the twiddle factor operations are the same. Therefore, the inputs of the $(\frac{P}{2}+1)$st Q-point RFFT at the second stage are also even symmetric. In conclusion, there are two Q-point REFFTs and $\frac{P-2}{2}$ Q-point HFFTs at the second stage. Thus, the number of signals at the output is equal to $\frac{Q+1}{2} \times 2 + \frac{P-2}{2} \times Q = \frac{PQ}{2} + 1$, which is also canonic with respect to the number of signals.

We can consider another 6-point canonic RFFT as shown in Figure 4.29. Note that the second 3-point RFFT at the second stage has been circularly shifted in frequency to eliminate redundancy. The canonic REFFT is shown in Figure 4.30. There are also only 4 signals at each stage.

4.6.5 $N = P \times Q$, P is even, Q is even

If P and Q are both even, the number of signals for the canonic REFFT is also equal to $\frac{PQ}{2}+1$. At the first stage, the inputs of the first P-point RFFT are even symmetric, while the inputs of the $(\frac{Q}{2}+1)$st P-point RFFT have the pattern of $x_P[k] = x_P[P-1-k]$, which can also be reduced to $\frac{P}{2}$ signals as described in Section 4.6.1. For the other $Q-2$

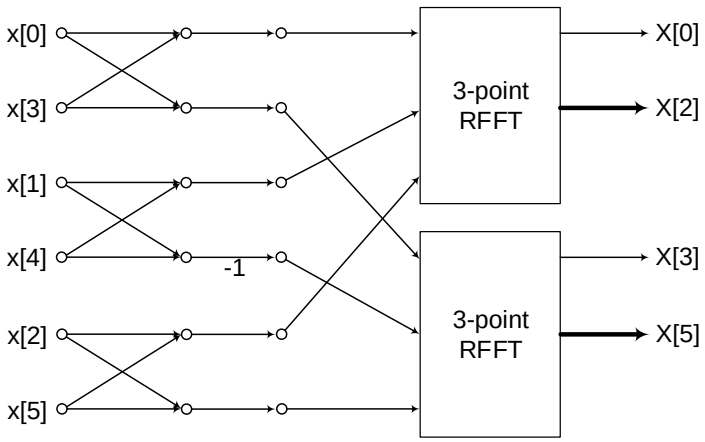

Figure 4.29: Flow-graph of a canonic 6-point RFFT, where $P = 2, Q = 3$.

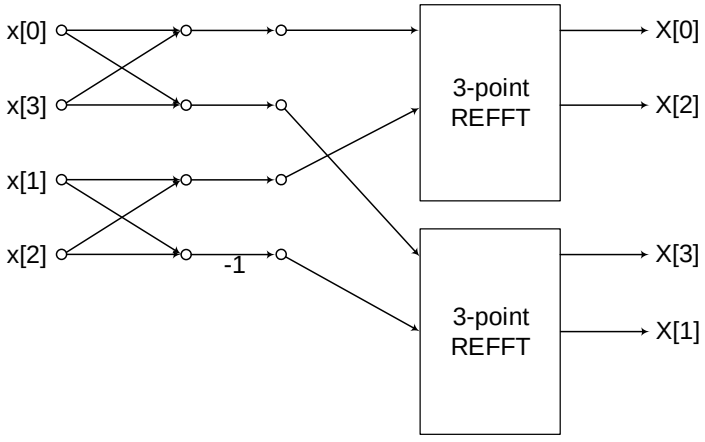

Figure 4.30: Flow-graph of a canonic 6-point REFFT, where $P = 2, Q = 3$.

P-point RFFTs, we only need to keep half of them, since the values of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT after the twiddle factor operations are conjugates of each other, where $2 \leq m \leq Q$ and $m \neq \frac{Q}{2}+1$. As a result, there are $\frac{P}{2}+1+\frac{P}{2}+\frac{Q-2}{2} \times P = \frac{PQ}{2}+1$ signals after the first stage.

At the second stage, for the $(\frac{P}{2}+1)$st Q-point FFT, we can consider it as a $(2 \times \frac{Q}{2})$-point FFT, as shown in Figure 4.31. $x[k]$ and $x[k+\frac{Q}{2}]$ go through a butterfly operations first, for $0 \leq k \leq \frac{Q}{2}-1$. We can perform the operation as shown in Figure 4.2 to these butterflies, i.e., push W^k to after the butterflies. As a result, the top input and the bottom input of the butterfly operation become purely real and purely imaginary, respectively. The bottom output of each butterfly can be eliminated, as it is conjugate of the top output. Then, these outputs are processed by one $\frac{Q}{2}$ FFT, as shown in Figure 4.32. Note that two real twiddle factor operations at the inputs are transformed to one complex twiddle factor operation at the output for each butterfly of this Q-point FFT. Therefore, we only need to keep $\frac{Q}{2}$ signals for this Q-point FFT in a $P \times Q$-point RFFT.

In the scenario that the inputs are even symmetric, since we have shown in Figure 4.32 that the bottom $\frac{Q}{2}$-point FFT can be deleted, we only need to make sure that the top $\frac{Q}{2}$-point FFT only involves real data-paths. We consider the flow-graph before pushing the twiddle factors, as shown in Figure 4.31. As the outputs $X[\frac{P}{2}]$ from the first P-point RFFT and the $(\frac{Q}{2}+1)$st P-point RFFT at the first stage are purely real and 0, respectively. As a result, the first butterfly in the $(2 \times \frac{Q}{2})$ structure can be reduced to a single data-path, as the bottom input is 0. For the remaining butterflies, the two inputs of each butterfly operation can be expressed by $aW^{\frac{P}{2}k}$ and $bW^{\frac{PQ}{4}+\frac{P}{2}k}$, as the outputs $X[\frac{P}{2}]$ from the P-point RFFTs at the first stage are all purely real. Furthermore, we have already proved that the values of the (m)th P-point RFFT and the $(Q+2-m)$th P-point RFFT after the twiddle factor operations are conjugates of each other. Then, the (m)th input $x_{\frac{Q}{2}}[m-1]$ and the $(Q+2-m)$th input $x_{\frac{Q}{2}}[Q-m+1]$ of the $\frac{Q}{2}$-point FFT are conjugates of each other. Therefore, the inputs of the $\frac{Q}{2}$-point are Hermitian

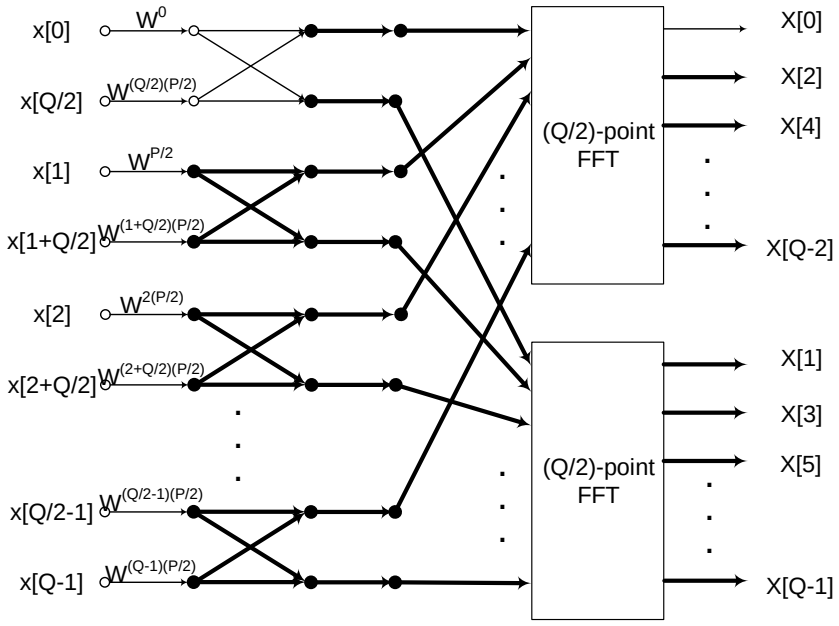

Figure 4.31: A Q-point FFT is considered as a $(2 \times \frac{Q}{2})$-point FFT.

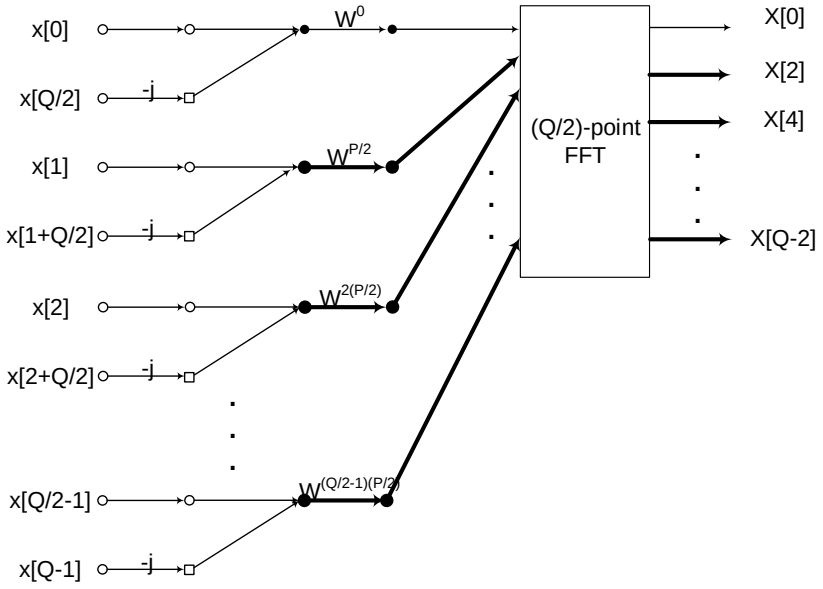

Figure 4.32: The $(2 \times \frac{Q}{2})$-point RFFT after eliminating redundancy.

symmetric. Thus, each of the rest butterflies in the $2 \times \frac{Q}{2}$ structure can also be reduced to one single datapath. If $\frac{Q}{2}$ is odd, the $2 \times \frac{Q}{2}$-point FFT can be reduced to the structure as shown in Figure 4.33; while if $\frac{Q}{2}$ is even, the canonic structure is shown in Figure 4.34. In Figure 4.34, similar to Equation (4.5), the twiddle factor $W^{\frac{PQ}{8}}$ is replaced by $\sqrt{2}$ after the input $x[\frac{Q}{4}]$, as the two inputs of the $(\frac{Q}{4}+1)$st butterfly can be expressed as $aW_N^{\frac{N}{8}} = a(\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j)$ and $-aW_N^{\frac{5N}{8}} = aj(\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j)$, respectively. Finally, the canonic REFFT is obtained.

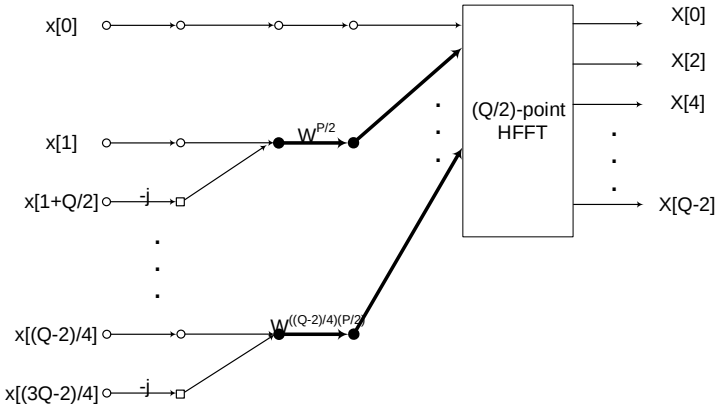

Figure 4.33: Remove redundancy of the $(2 \times \frac{Q}{2})$-point RFFT, when $\frac{Q}{2}$ is odd.

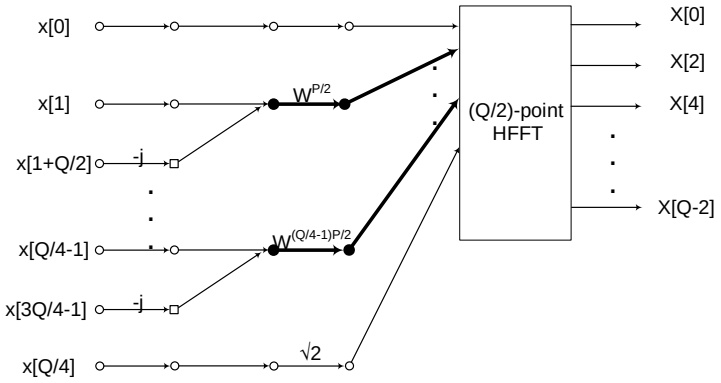

Figure 4.34: Remove redundancy of the $(2 \times \frac{Q}{2})$-point RFFT, when $\frac{Q}{2}$ is even.

All radix-2^m RFFT structures fall into this category. For example, a radix-4 16-point canonic RFFT is shown in Figure 4.35. At the first stage, there is one 4-point REFFT and one 4-point RFFT. For the third 4-point RFFT, the structure can be considered as a 2×2 structure. According to Section 4.6.1, we only need to keep 2 signals. Therefore, there are 9 signals at the first stage in total. At the second stage, the inputs of the first 4-point RFFT are even symmetric, while the inputs of the second 4-point RFFT are Hermitian symmetric. For the third 4-point FFT, we could also reduce it to the structure only with 2 signals, based on Figure 4.34. The total number of signals at the second stage is also 9. The canonic 16-point REFFT is shown in Figure 4.36.

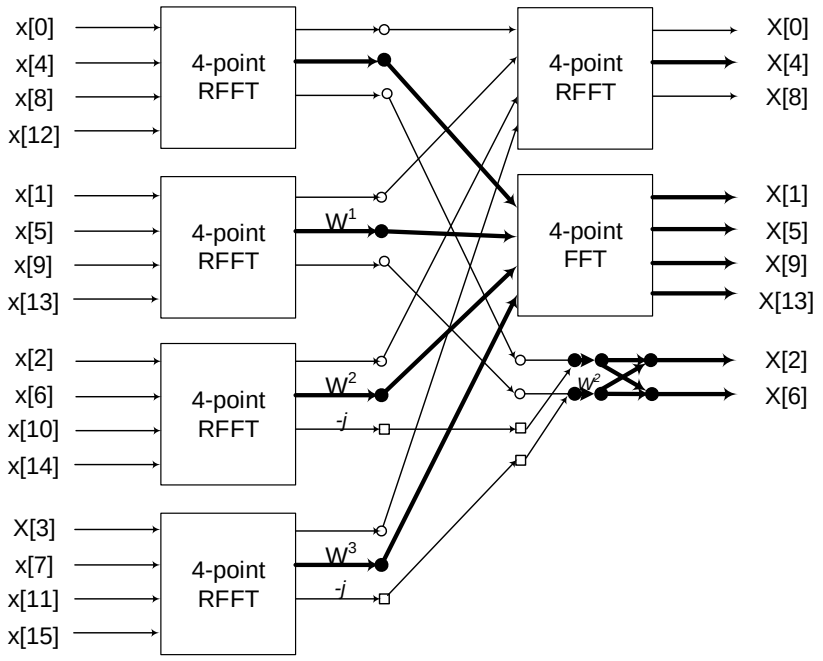

Figure 4.35: A 16-point canonic RFFT, where $P = 4, Q = 4$.

4.6.6 Summary

Based on the discussion above, we summarize the types of FFTs for the four different cases in Table 4.4.

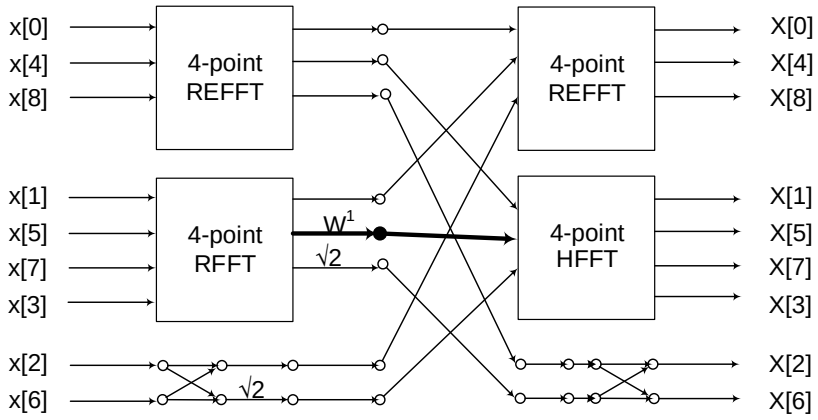

Figure 4.36: A 16-point canonic REFFT, where $P = 4, Q = 4$.

Table 4.4: FFT types for canonic REFFT

	P is odd, Q is odd		P is odd, Q is even		P is even, Q is odd		P is even, Q is even	
	P-point	Q-point	P-point	Q-point	P-point	Q-point	P-point	Q-point
#REFFT	1	1	2	1	1	2	1	1
#RFFT	$\frac{Q-1}{2}$	0	$\frac{Q-2}{2}$	0	$\frac{Q-1}{2}$	0	$\frac{Q-2}{2}$	0
#HFFT	0	$\frac{P-1}{2}$	0	$\frac{P-1}{2}$	0	$\frac{P-2}{2}$	0	$\frac{P-2}{2}$
$\#(\frac{P}{2} \times 2)$	0	0	0	0	0	0	1	0
$\#(2 \times \frac{Q}{2})$	0	0	0	0	0	0	0	1

The canonic ROFFT for any composite size can be obtained similarly by following these steps described in this section. We do not discuss these designs in detail in this thesis. The only difference is that we need to make sure there are only $\lceil \frac{M}{2} \rceil - 1$ signals for an M-point ROFFT instead of $\lfloor \frac{M}{2} \rfloor + 1$.

4.7 Performance

In this section, we discuss the performances of the canonic REFFT/ROFFT.

The number of signals in the canonic REFFT/ROFFT is less than FFT, RFFT or canonic RFFT, as we remove the redundant inputs from the beginning. Furthermore, the number of butterfly operations in the REFFT/ROFFT flow-graph is also reduced, as we remove the butterfly operation if the two inputs of the butterfly operation have the same value or opposite values, as described in Figure 4.5 or Figure 4.11, respectively. Consequently, the number of twiddle factor operation is also reduced for a power-of-two size RFFT, as one quarter of the datapaths are eliminated when we extend a canonic N-point REFFT/ROFFT from a canonic $\frac{N}{2}$-point REFFT/ROFFT. Moreover, from the third stage to the last stage, there is one twiddle factor $W_N^{\frac{N}{8}}$ before the stage is replaced by a multiplication by $\sqrt{2}$. Thus, for an $N = 2^n$-point RFFT, when $n \geq 2$, there will be $n-2$ multiplications of $\sqrt{2}$ in the flow-graph. Note that we do not consider the multiplications of 2 in the flow-graph which are generated by the operations of Figure 4.5 and Figure 4.11 as multipliers, since these only involve 1-bit left shift.

Table 4.5 compares the performance of the proposed canonic REFFT/ROFFT with FFT, RFFT, and canonic RFFT. Note that we consider a complex butterfly operation as two real butterfly operations.

It can be seen that the proposed canonic REFFTs/ROFFTs have less number of signals, less butterfly operations, and less twiddle factor operations. Due to the fact that the canonic ROFFT has less signal values at each stage compared to canonic REFFT, the canonic ROFFT also requires less butterfly operations.

Table 4.5: Performance Comparison for $N = 2^n$-point FFT Algorithms

FFT Algorithm	#Signal Values at Each Stage	#Real Butterfly Operation	#Twiddle Factor Operations
Complex FFT	$2N$	$Nlog_2N$	$n \times 2^{n-1} - 2^n + 2$
[6] DIF RFFT	$\geq N$	$(\frac{N}{2} - 1)log_2N + 1$	$(n - \frac{7}{2}) \times 2^{n-2} + n - 1$
[15] Canonic DIT RFFT	N	$\frac{N}{2}log_2N - \frac{N}{2} + 1$	$(n-3) \times 2^{n-2} + 1$
[15] Canonic DIF RFFT	N	$\frac{N}{2}log_2N - \frac{N}{2} + 1$	$(n-4) \times 2^{n-2} + n$
Canonic REFFT	$\frac{N}{2} + 1$	$(\frac{N}{4} + 1)log_2N - \frac{N}{2}$	$2^{n-2} - n + 1\#$
Canonic ROFFT	$\frac{N}{2} - 1$	$\frac{N}{4}log_2N - \frac{N}{2}$	$2^{n-2} - n + 1\#$

$\#$ *when* $n \geq 2$, *there will also be* $n-2$ *multiplications of* $\sqrt{2}$ *in the flow-graph.*

4.8 Conclusion

This chapter has proposed novel algorithms to design canonic FFT flow-graphs when the inputs are real and even/odd symmetric. A canonic N-point REFFT/ROFFT can be extended from a canonic $\frac{N}{2}$-point REFFT/ROFFT. Twiddle factor transformations are needed if there are twiddle factors other than $W_N^{\frac{N}{4}}$ and $W_N^{\frac{N}{8}}$ before the last stage. The designs of canonic REFFT for any composite length have also been presented. Furthermore, it can be concluded that the proposed designs of canonic REFFT/ROFFT have less number of signals, less butterfly operations, and less twiddle factor operations, compared to prior works.

Chapter 5

An In-Place FFT Architecture for Real-Valued Signals

5.1 Introduction

Most FFT architectures can be divided into two categories: pipelined and in-place. Pipelined architectures contain either feed-forward or feedback datapaths [17]. The feedback architectures have been referred to as single-path delay feedback (SDF) and the feed-forward architectures have been referred to as multi-path delay commutator (MDC). Much research has been carried out on the design of pipelined architectures for computing the FFT of complex and real-valued signals for high throughput applications [18, 19, 6]. These pipelined architectures are suitable for high-throughput applications. For example, a 2-parallel architecture requires $N/2$ cycles for an N-point FFT. The focus of this chapter is on in-place architectures where few processing elements (PEs) are used in a memory-based architecture. These architectures, also referred to as continuous-flow architectures, are well suited for low to moderate speed applications. Several memory-based architectures have been proposed to achieve smaller area [20, 21, 22, 23, 24, 25]. Higher radix butterfly units and/or parallel processing can be

utilized to increase the throughput [25]. Very few in-place architectures have been proposed for real-valued signals [2, 26] based on the packing algorithm, which computes a complex FFT and requires additional operations for post-processing.

In this chapter, we propose a novel memory-based FFT architecture which computes the RFFT based on the modified radix-2 algorithm in [5]. The algorithm computes only half of the output samples and removes the redundant operations from the flow graph. The modified flow graph contains only real data paths as opposed to complex data paths in a regular flow graph. Therefore, the word-length in the memory units of the proposed FFT processor is W, where W is the word-length chosen to represent either real/imaginary component. A new processing element is proposed to efficiently implement the operations in the modified flow graph. The processing element consists of two radix-2 butterflies and can process four samples in parallel. Novel addressing and bank assignment schemes are proposed for conflict-free memory accesses. Further, it is shown that the proposed scheme can be extended to support multiple parallel processing elements. The key contribution is the design of an in-place RFFT/RIFFT architecture based on an FFT flow-graph optimized for real signals that achieves the least area-time product for RFFT/RIFFT among all known designs, where area corresponds to the data-path area, and time corresponds to the number of cycles.

5.2 Proposed Architecture

A low-complexity algorithm to compute FFT of real signals has been proposed in [5] to compute the RFFT. The radix-2 flow graph is modified after removing the redundant operations to obtain a regular geometry. The modified data-flow graph of a 32-point RFFT is shown in Figure 5.1. The flow graph computes only 17 output samples instead of all 32 samples. Further, the entire datapath is real. We can observe that at any stage of the computation, we need to store only N real samples instead of complex samples.

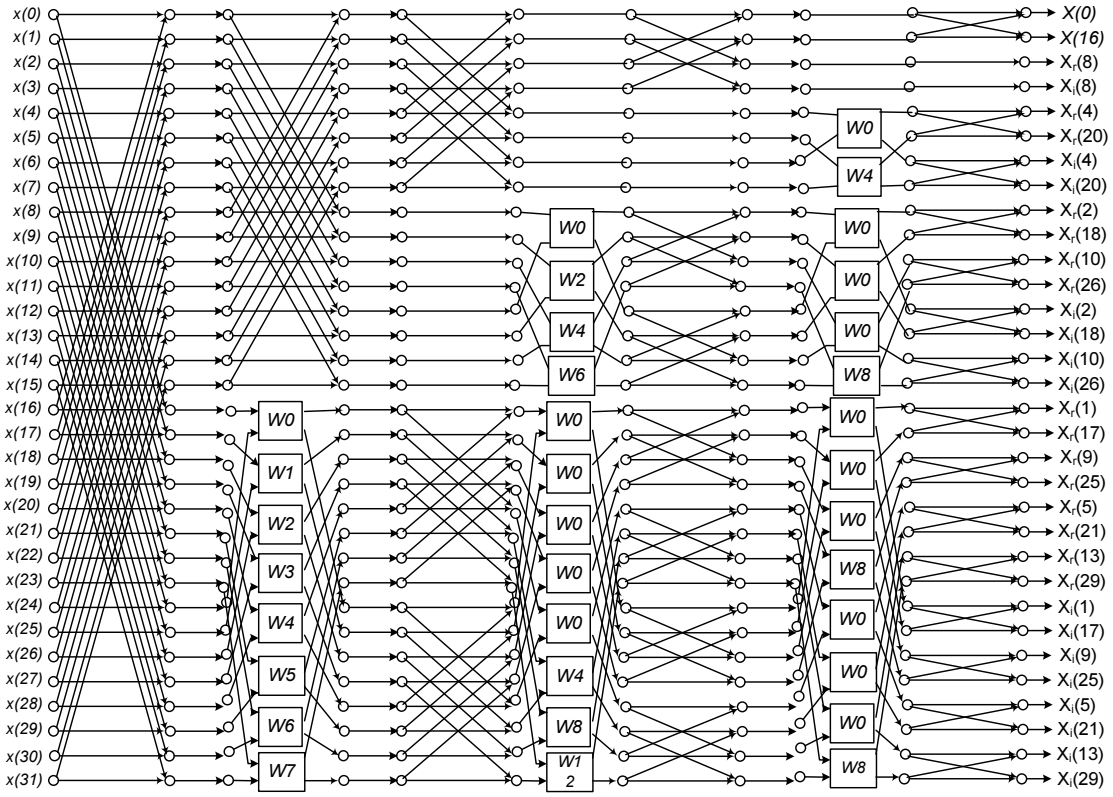

Figure 5.1: Data flow graph of 32-point FFT for real-valued signals.

Further, all the butterflies in the flow graph process only real samples which require two real adders instead of two complex adders.

Figure 5.2 shows the high-level architecture of the proposed FFT processor with L processing elements (PEs). Each memory module can store N words of length W. The number of banks within a memory module is $4L$. Each PE is associated with 4 memory banks in each memory module. The data streams are multiplexed between the I/O interface and the processing element by $\frac{N}{4}log_2N$ computational cycles. When I/O interface communicates with Memory Module 1, the PE reads and writes to Memory Module 2. When Memory Module 1 is full of new data, the PE can begin reading and working on the data stored there. At this time, the I/O reads from and writes to

Memory Module 2. The PE thus alternates between the two memory modules, and as long as new data is sent to the memory module not being used by PE, memory conflicts are avoided.

A new processing element is proposed based on the modified FFT flow graph [5]. Figure 5.3 shows the proposed processing element (PE) for the RFFT processor. The PE consists of two butterflies and one complex multiplier. The PE processes 4 samples in parallel to increase the throughput. For an architecture with one processing element, the number of required computational cycles is $\frac{N}{4}log_2N$ for an N-point RFFT. The bottom butterfly unit, the switch, and the multiplier are configured by the control unit shown in Figure 5.2 to different modes based on the data flow of the FFT stages. Dual-port SRAMs are necessary to read input data and write output data simultaneously. This choice of PE leads to less complexity and high throughput. A PE containing a single butterfly and a single multiplier would require $\frac{N}{2}log_2N$ cycles. If the number of these PEs is increased, the number of multipliers would increase proportionately, and would be twice that of the proposed architecture. On the other hand, a radix-2^2 datapath would have a longer latency and would negate the advantages of reducing the number of stages by a factor of 2.

5.2.1 Address Generation Scheme

We present a novel address generation scheme for the RFFT computation in this section. We first illustrate the conflict-free memory addressing scheme for one processing element. Then, we extend the approach to an architecture with multiple processing elements.

The proposed addressing strategy works similar to a pipelined FFT architecture. In a pipelined architecture, the intermediate values between the stages are stored in the registers which act as FIFOs. The number of registers depends on the stage of the computation. As an example, the feed-forward pipelined architecture of a radix-2 16-point FFT computation is shown in Figure 5.4 [6]. We can observe that after the 1st

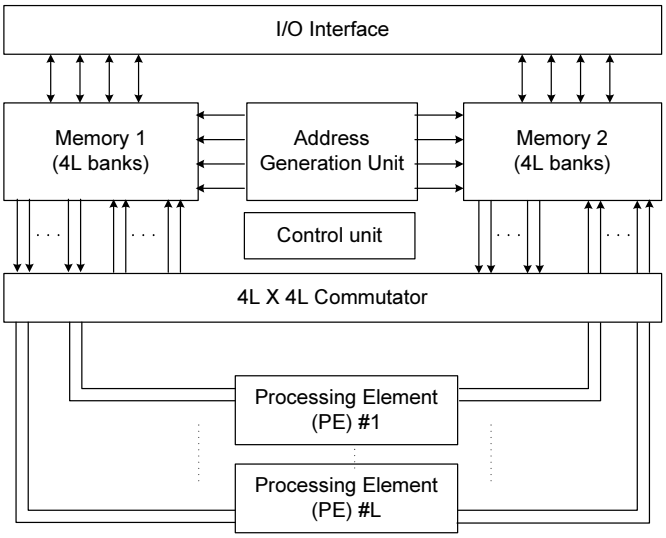

Figure 5.2: Proposed memory-based RFFT architecture.

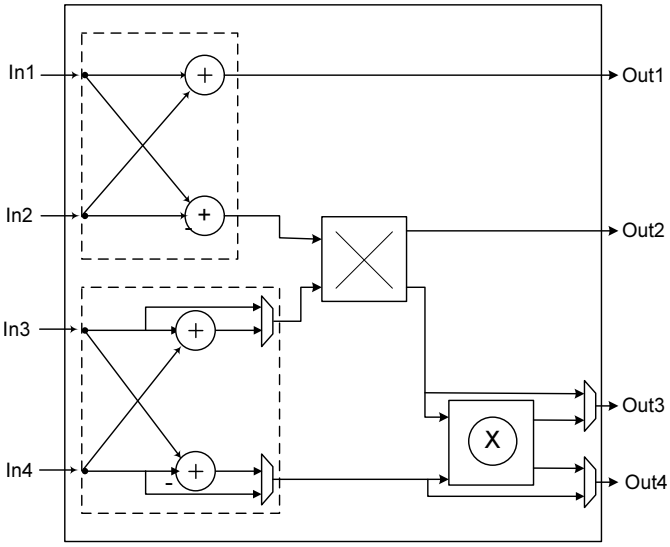

Figure 5.3: Processing element for real-valued signals.

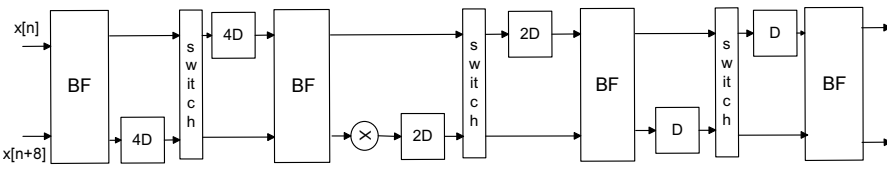

Figure 5.4: 2-parallel pipelined architecture for a 16-point FFT computation.

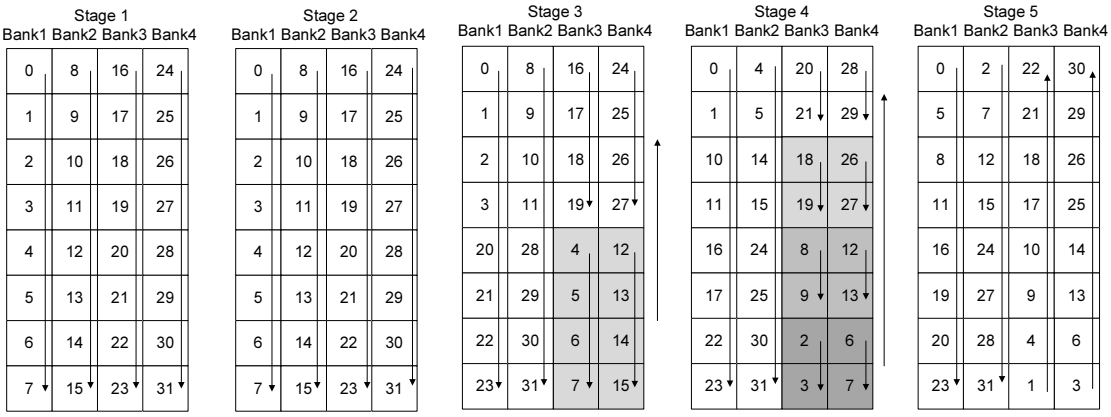

Figure 5.5: Illustration of proposed addressing scheme for one processing element. The arrows show how the data is read/write into the memory banks during the various stages of the RFFT computation.

stage the intermediate computations are stored in groups of 2^{n-k} in the stage k. We read/write the intermediate computations in groups in the proposed RFFT processor. The size of the group depends on the FFT computation stage.

If we use an $(n-2)$-bit counter as an address for memory access, then the address for an N-point RFFT can be generated using the pattern shown in Table 5.1. The proposed addressing scheme enables these concurrent reads and writes without bank conflicts. An example of the addressing scheme for a 32-point RFFT is shown in Figure 5.5. Each column corresponds to one memory bank and shows how the data is stored in the memory bank at that stage of the computation. The numbers correspond to the indices of the input/output at each stage. For example, the data with indices (0, 8, 16, 24), and (1, 9, 17, 25) are read in order to compute the butterflies in the first stage. After the computation, the output data are written to the same locations. The order in which they are written back is altered to avoid bank conflicts during the computation of the butterflies in the next stage. Based on Table 5.1, the read/write patterns of memory banks 0 and 1 are the same for all stages. For these banks, the memory address can be represented as $b_2b_1b_0$. The data is read/written in a serial order from address location 0

to 7 at every stage for these banks. Further, the addressing patterns of memory banks 2 and 3 are also the same but the pattern varies across the stages. The data are stored in groups after stage 2, similar to a pipelined architecture. The size of the group goes down by a factor of 2 with every subsequent stage as shown in Figure 5.5. The addressing pattern for stages 3, 4, and 5 are given by $\overline{b_2}b_1b_0$, $\overline{b_2}\overline{b_1}b_0$ and $\overline{b_2}\overline{b_1}\overline{b_0}$, respectively. These addresses can be generated from only one counter by simply selecting the value or its inverse of each bit.

Table 5.1: Address Patterns for N-point RFFT Computation ($N = 2^n$)

Counter	Bank 0,1 Address	Bank 2,3 Address					
		Stage 1	Stage 2	Stage 3	Stage 4	...	Stage n
$b_{n-3}b_{n-4}...b_0$	$b_{n-3}b_{n-4}...b_0$	$b_{n-3}b_{n-4}...b_0$	$b_{n-3}b_{n-4}...b_0$	$\overline{b_{n-3}}b_{n-4}...b_0$	$\overline{b_{n-3}}\overline{b_{n-4}}...b_0$	...	$\overline{b_{n-3}}\overline{b_{n-4}}...\overline{b_0}$

5.2.2 Memory Bank Assignment

This subsection considers specific memory bank assignments for different stages. We denote the read pattern as (r_1, r_2, r_3, r_4) and the write pattern as (w_1, w_2, w_3, w_4). For example, $(r_1, r_2, r_3, r_4) = (2, 3, 1, 0)$ means "in1" in Figure 5.3 reads from bank 2, "in2" reads from bank 3, "in3" reads from bank 1, and "in4" reads from bank 0. In the proposed RFFT architecture, I/O processes the memory in the natural order for the first and the second stage. Therefore, if we place the input samples in the memory banks in the order $(i, i + N/4, i + N/2, i + 3N/4)$ as shown in the first stage of Figure 5.5, the read pattern and the write pattern for the stage 1 are both $(0, 2, 1, 3)$. For the second stage, the read pattern has to be $(0, 1, 2, 3)$ in order to correctly compute the outputs. However, notice that the addressing pattern for the banks 2 and 3 of the third stage changes to $\overline{b_{n-3}}b_{n-4}...b_0$. Therefore, we can write the latter half of values $(w_1, w_2, w_3, w_4) = (2, 3, 0, 1)$ in stage 2 for further computation. In general, we always write the first half of the values as $(0, 1, 2, 3)$ and the latter half as $(2, 3, 0, 1)$ in a group of size 2^{n-k} for the stage k. As a result, the read pattern for the next stage $k+1$ will be $(0, 2, 1, 3)$ and $(2, 0, 3, 1)$ for the first half and the second half, respectively, in a group

with size 2^{n-k}. In conclusion, in stage k (after the first stage) of a proposed 2^n-point RFFT architecture with single PE, the read pattern alternates between $(0, 2, 1, 3)$ and $(2, 0, 3, 1)$ in groups of 2^{n-k}, and the write pattern alternates between $(0, 1, 2, 3)$ and $(2, 3, 0, 1)$ in groups of 2^{n-k-1} (except the last stage). The write pattern $(0, 1, 2, 3)$ of the last stage can be used to support normal-order output. While several feasible memory bank assignment methods exist, we only provide one simple and efficient scheme in this chapter that requires simple control logic. An example of memory bank assignment for a 32-point RFFT is illustrated in Table 5.2.

Table 5.2: Memory Bank Assignment for 32-point RFFT with Single PE

Counter	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
	Read Pattern				
000	(0,2,1,3)	(0,1,2,3)	(0,2,1,3)	(0,2,1,3)	(0,2,1,3)
001	(0,2,1,3)	(0,1,2,3)	(0,2,1,3)	(0,2,1,3)	(2,0,3,1)
010	(0,2,1,3)	(0,1,2,3)	(0,2,1,3)	(2,0,3,1)	(0,2,1,3)
011	(0,2,1,3)	(0,1,2,3)	(0,2,1,3)	(2,0,3,1)	(2,0,3,1)
100	(0,2,1,3)	(0,1,2,3)	(2,0,3,1)	(0,2,1,3)	(0,2,1,3)
101	(0,2,1,3)	(0,1,2,3)	(2,0,3,1)	(0,2,1,3)	(2,0,3,1)
110	(0,2,1,3)	(0,1,2,3)	(2,0,3,1)	(2,0,3,1)	(0,2,1,3)
111	(0,2,1,3)	(0,1,2,3)	(2,0,3,1)	(2,0,3,1)	(2,0,3,1)
	Write Pattern				
000	(0,2,1,3)	(0,1,2,3)	(0,1,2,3)	(0,1,2,3)	(0,1,2,3)
001	(0,2,1,3)	(0,1,2,3)	(0,1,2,3)	(2,3,0,1)	(0,1,2,3)
010	(0,2,1,3)	(0,1,2,3)	(2,3,0,1)	(0,1,2,3)	(0,1,2,3)
011	(0,2,1,3)	(0,1,2,3)	(2,3,0,1)	(2,3,0,1)	(0,1,2,3)
100	(0,2,1,3)	(2,3,0,1)	(0,1,2,3)	(0,1,2,3)	(0,1,2,3)
101	(0,2,1,3)	(2,3,0,1)	(0,1,2,3)	(2,3,0,1)	(0,1,2,3)
110	(0,2,1,3)	(2,3,0,1)	(2,3,0,1)	(0,1,2,3)	(0,1,2,3)
111	(0,2,1,3)	(2,3,0,1)	(2,3,0,1)	(2,3,0,1)	(0,1,2,3)

5.2.3 PE Configuration Unit

Besides the addressing scheme and memory bank assignment in the proposed RFFT architecture, a control unit also needs to be used to configure the PE under different

stages. As the switch propagates the signals straight and the bottom butterfly is bypassed for the first 2^{n-k} computational cycles in the stage k (except the first stage) of a 2^n-point RFFT architecture, a PE configuration unit can be designed to generate the select signal for the switch and bottom butterfly as shown in Figure 5.6. While the select signal is 0, the switch propagates the signals straight and the bottom butterfly is bypassed; otherwise, when the select signal is 1, the switch exchanges the two signals and the bottom butterfly is utilized. Moreover, for the first stage, the select signal for the switch should be 0, while the select signal for the bottom butterfly should be 1. Note that the multiplier unit in Figure 5.3 also needs to be configured which will be bypassed for the first and the last stage.

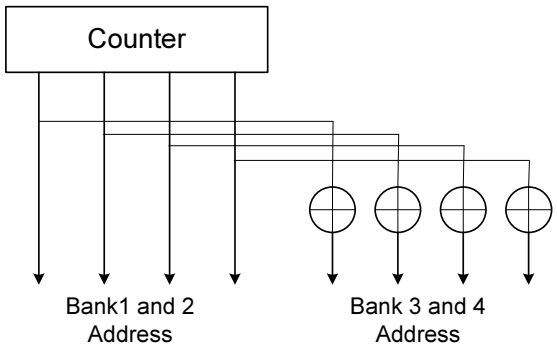

Figure 5.6: PE configuration unit (generating select signals for each stage).

5.2.4 Generalization to Multiple Processing Elements

Further, the proposed addressing scheme can be extended to an architecture with multiple processing elements. Figure 5.7 shows the proposed addressing scheme for two processing elements as an example. Eight data samples need to be read/written in each clock cycle. The memory module consists of 8 memory banks to enable parallel processing of two PEs. We can observe that the read/write address patterns for the first three stages are the same. Similar to the scheme with one PE, the data will be divided into groups from the 4-th stage of the FFT. As the addressing order for memory

banks 0-3 and memory banks 4-7 are the same, only two different addresses need to be generated.

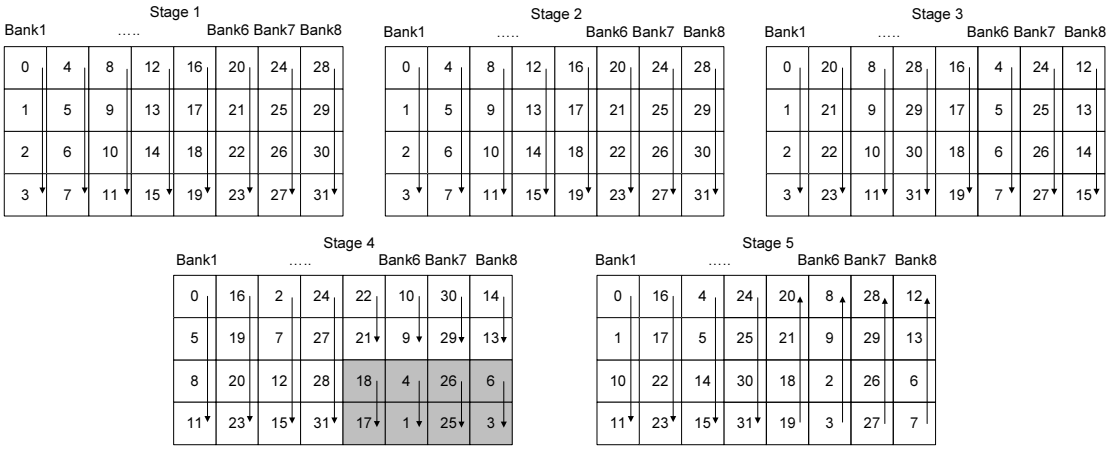

Figure 5.7: Illustration of proposed addressing scheme for two processing elements.

A more general method is described below. We consider the number of PEs in the proposed RFFT architecture to be a power of two, i.e., $L = 2^l$. There are $4L$ memory banks each of which can store $\frac{N}{4L}$ values. Thus, the address of each memory bank can be controlled by an $(n - l - 2)$-bit counter. Similar to Table 5.1, in stage k, the addressing pattern of memory bank $0 : 2L - 1$ can be represented as $b_{n-l-3}b_{n-l-4}...b_0$, and the addressing pattern of memory bank $2L : 4L - 1$ can be represented as $\overline{b_{n-l-3}}...\overline{b_{n-k}}b_{n-k-1}...b_0$, for $l + 3 \leq k \leq n$ (when $k = n$, the addressing pattern is simply $\overline{b_{n-l-3}}\overline{b_{n-l-4}}...\overline{b_0}$). For the stages $1 : l + 2$, the addressing patterns for all the memory banks are in the natural order.

Bank assignment is also similar to the architecture with one PE. In stage k (after the first stage) of a 2^n-point RFFT architecture with L processing elements, the read pattern still changes alternatively in groups of 2^{n-k}, and the write pattern changes alternatively in groups of 2^{n-k-1}. The specific patterns are dependent on L and the stage k, and are summarized in Table 5.3. Note that if $l = 0$, i.e., $L = 1$, the pattern of the fourth row in Table 5.3 is skipped. Moreover, when $l > n - 3$, the architecture

reduces to a fully-parallel architecture. We use a write pattern $(0 : 4L-1)$ to obtain the normal-order output in the last stage. The values for a, b and c need to be substituted in Table 5.3 to obtain the bank assignments for all $4L$ banks.

Table 5.3: Memory Bank Assignment for 2^n-point RFFT with L PEs

k	Read Pattern for Stage k	Write Pattern for Stage k
Stage 1	$(0, 2L, L, 3L, ..., a, a+2L, a+L, a+3L, ..., L-1, 3L-1, 2L-1, 4L-1)$	
Stage 2 to $l+1$	$(0, \frac{L}{2^{k-2}}, \frac{2L}{2^{k-2}}, \frac{3L}{2^{k-2}}, ..., b, b+\frac{L}{2^{k-2}}, b+\frac{2L}{2^{k-2}}, b+\frac{3L}{2^{k-2}}, ..., 4L-\frac{3L}{2^{k-2}}-1, 4L-\frac{2L}{2^{k-2}}-1, 4L-\frac{L}{2^{k-2}}-1, 4L-1)$	
Stage $l+2$	$(0 : 4L-1)$	$(0 : 2L-1, 2L : 4L-1)$
		$(2L : 4L-1, 0 : 2L)$
Stage $l+3$ to $n-1$ (1)	$(0, 2L, 1, 2L+1, ..., c, c+2L, c+1, c+2L+1, ..., L-1, 3L-1, 2L-1, 4L-1)$	$(0 : 2L-1, 2L : 4L-1)$
	$(2L, 0, 2L+1, 1, ..., c+2L, c, c+2L+1, c+1, ..., 3L-1, L-1, 4L-1, 2L-1)$	$(2L : 4L-1, 0 : 2L)$
Stage n (2)	$(0, 2L, 1, 2L+1, ..., c, c+2L, c+1, c+2L+1, ..., L-1, 3L-1, 2L-1, 4L-1)$	$(0 : 4L-1)$
	$(2L, 0, 2L+1, 1, ..., c+2L, c, c+2L+1, c+1, ..., 3L-1, L-1, 4L-1, 2L-1)$	

(1) In stage the read pattern alternates between the two rows of the middle column in groups of 2, and write pattern alternates between the two rows of the right columns in groups of 2.

(2) In the last stage, the read pattern alternates between the two middle columns every cycle.

For example, similar to Table 5.2, we can generate the memory bank assignment scheme for the 32-point RFFT with 2 PEs based on Table 5.3, as shown in Table 5.4.

Table 5.4: Memory Bank Assignment for 32-point RFFT with 2 PEs

Counter	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
	Read Pattern				
000	(0,4,2,6,1,5,3,7)	(0,2,4,6,1,3,5,7)	(0,1,2,3,4,5,6,7)	(0,4,1,5,2,6,3,7)	(0,4,1,5,2,6,3,7)
001	(0,4,2,6,1,5,3,7)	(0,2,4,6,1,3,5,7)	(0,1,2,3,4,5,6,7)	(0,4,1,5,2,6,3,7)	(4,0,5,1,6,2,7,3)
010	(0,4,2,6,1,5,3,7)	(0,2,4,6,1,3,5,7)	(0,1,2,3,4,5,6,7)	(4,0,5,1,6,2,7,3)	(0,4,1,5,2,6,3,7)
011	(0,4,2,6,1,5,3,7)	(0,2,4,6,1,3,5,7)	(0,1,2,3,4,5,6,7)	(4,0,5,1,6,2,7,3)	(4,0,5,1,6,2,7,3)
	Write Pattern				
000	(0,4,2,6,1,5,3,7)	(0,2,4,6,1,3,5,7)	(0,1,2,3,4,5,6,7)	(0,1,2,3,4,5,6,7)	(0,1,2,3,4,5,6,7)
001	(0,4,2,6,1,5,3,7)	(0,2,4,6,1,3,5,7)	(0,1,2,3,4,5,6,7)	(4,5,6,7,0,1,2,3)	(0,1,2,3,4,5,6,7)
010	(0,4,2,6,1,5,3,7)	(0,2,4,6,1,3,5,7)	(4,5,6,7,0,1,2,3)	(0,1,2,3,4,5,6,7)	(0,1,2,3,4,5,6,7)
011	(0,4,2,6,1,5,3,7)	(0,2,4,6,1,3,5,7)	(4,5,6,7,0,1,2,3)	(4,5,6,7,0,1,2,3)	(0,1,2,3,4,5,6,7)

5.3 Hermitian-symmetric IFFT processor

The IFFT of a Hermitian-symmetric signal can be computed using only half the samples. The flow graph can be modified in a way similar to the RFFT flow graph. We can derive the IFFT flow graph by transposing the RFFT flow graph. Figure 5.8 shows the flow graph of 16-point IFFT of a Hermitian-symmetric sequence. It can be seen that

only 9 input samples are utilized to compute the 16-point real sequence. Further, all the data paths carry real signals. Due to this, the word length of the required memory can be W instead of $2W$, where W is the word length of either real/imaginary component.

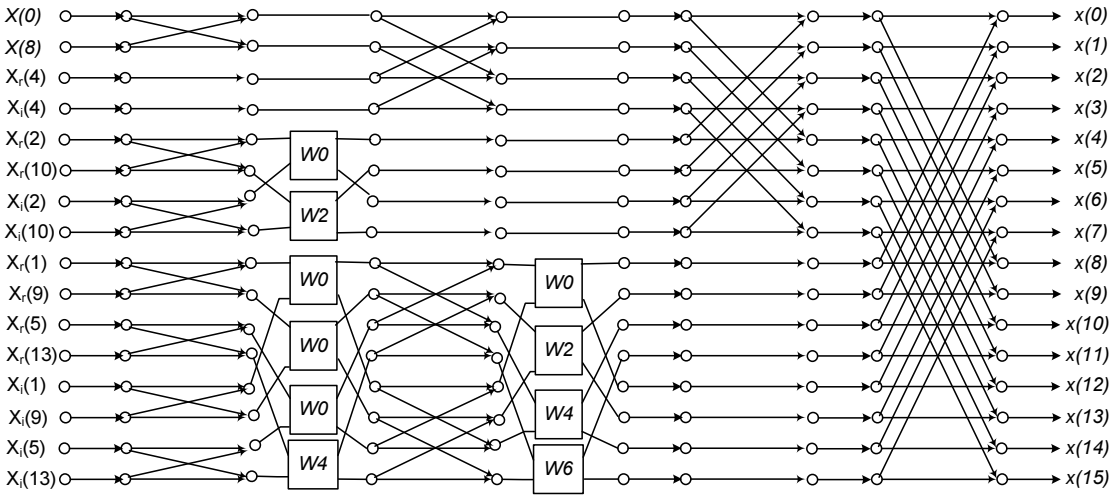

Figure 5.8: Flow graph of the 16-point inverse FFT for Hermitian-symmetric input.

The RIFFT can be computed using the proposed processing element shown in Figure 5.3. The architecture of the RIFFT processor will be same as the RFFT except the addressing strategy. Similar to the RFFT case, the processing element operates in different modes. The butterfly is bypassed and the switch propagates the signal straight for the first 2^{k-1} cycles during the first $(n-1)$ stages. Moreover, the multiplier unit is bypassed in the $(n-1)$ and n-th stages.

Figure 5.9 shows the addressing scheme for a 16-point RIFFT computation. The second column shows the initial order in which the data need to be stored. The corresponding data indices are shown in the first column. As an example, $R4$ and $I4$ correspond to the real and imaginary components of $X(4)$ sample. Figure 5.9 also shows the how the data are accessed with the arrows. The address patterns for the RIFFT computation are similar to the RFFT as shown in the Table 5.1. The only difference is in when selecting the value or its inverse of each counter bit to generate

the address for banks 2 and 3. The addressing pattern for stage k is $b_{n-3}...b_{k-2}\overline{b_{k-3}}...\overline{b_0}$ when $k \leq n$ (when $k = n$, then the addressing pattern is simply $\overline{b_{n-3}}...\overline{b_0}$). Therefore, by configuring the selecting signals, we can either compute RFFT or RIFFT by using the same processor.

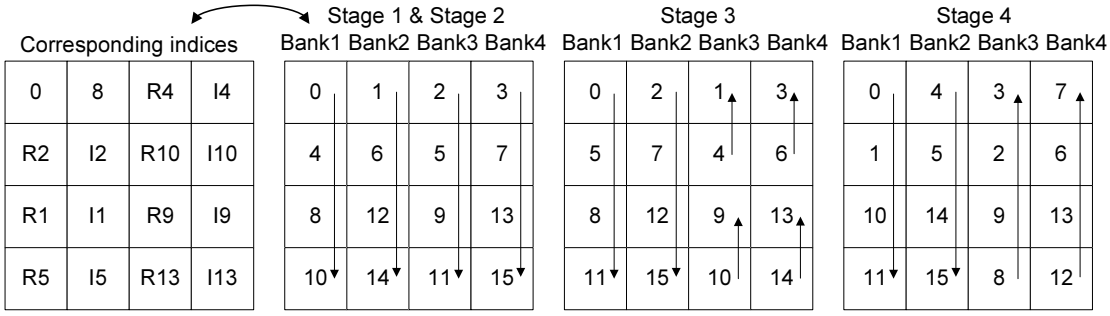

Figure 5.9: Addressing scheme for the proposed 16-point Hermitian-symmetric IRFFT processor.

5.4 Comparisons

In Table 5.5, we compare the hardware complexity and computation cycles of a prior memory-based RFFT processor with the proposed design. The radix and the number of memory banks required are also provided. The required computation cycles of our proposed design are $\frac{N}{4L}log_2N$, where L is the number of PEs. It can be seen that the proposed design with a single PE requires the same number of computational cycles as the one in [2]. This is achieved with a PE of two real butterflies and one complex multiplier, while the prior design requires a complex radix-4 butterfly consisting of 12 complex adders and three complex multipliers. Further, the proposed design with 2 PEs outperforms the prior approach in terms of both computational cycles and hardware complexity. We calculate the area-time (AT) product by taking *computation cycles*$\times$*complex adder area* as the unit (it is fair to assume the area of a complex multiplier is 10 times of the area of a complex adder). It can be seen that the proposed

design also achieves the best performance in terms of AT product among all known designs. The proposed design is scalable with respect to the number of PEs. Based on 65 nm technology and $1V$ supply, the critical path, i.e., the total delay of the PE, a read commutator and a write commutator, increases by 3% if the number of PEs is between 2 and 16, and by 6.8% if the number of PEs is between 32 and 4096. Thus, the total computation time decreases approximately linearly with increase in the number of PEs.

Table 5.5: Comparison of the RFFT processors

	Proposed		Proposed (2-PEs)		[2]		[20]		[21]	
Radix	radix-2		radix-2		radix-4		radix-2		radix-2/4	
Complex Adders	2		4		12		12		8	
Complex Multipliers	1		2		3		3		3	
Memory	N/4 x W		N/8 x W		N/8 x 2W		N/4 x 2W		N/4 x 2W	
# Memory banks	4		8		4		4		4	
N	# Cycles	AT	# Cycles	AT	# Cycles	AT	# Cycles	AT	# Cycles	AT
256	512	6144	256	6144	512	21504	512	21504	256	9728
512	1152	13824	576	13824	768	32256	1152	48384	640	24320
1024	2560	30720	1280	30720	2560	107520	2560	107520	1280	48640
2048	5632	67584	2816	67584	3854	161868	5632	236544	3072	116736
4096	12288	147456	6144	147456	12288	516096	12288	516096	6144	233472

Moreover, when compared to the in-place complex FFT [20, 21], one obvious advantage of our proposed RFFT architecture is that the length of the required memory can be reduced by a factor of 2.

5.5 Conclusion

This chapter has proposed a novel continuous-flow FFT processor for real-valued signals. The proposed computation scheme is based on a modified algorithm which removes the redundant operations from the flow graph. A new processing element is proposed to reduce the hardware complexity and can process 4 samples in parallel. A conflict-free addressing scheme is developed which can be extended to support parallel processing elements. The proposed RFFT processor reduces the number of computational cycles with low hardware complexity. Furthermore, the proposed processor consumes less energy compared to prior work based on packing algorithm.

Chapter 6

Conclusion and Future Directions

This thesis has considered the efficient implementations of FFT computation by removing redundancies under specific applications. Furthermore, the design of in-place real-valued FFT architecture have also been discussed.

We are the first to introduce the concept of canonic RFFT structure. In a canonic FFT structure, the number of signals is guaranteed to be the minimum at each stage. Therefore, the arithmetic redundancy is completely removed in the FFT flow-graph. We have shown that the canonic RFFT structures require the least butterfly operations and only involve real datapath when mapped to architectures. We have presented the algorithms for generating canonic RFFT structures for both power-of-two size and any composite lengths.

From the architectural perspective, we have proposed a novel scalable in-place RFFT/RIFFT architecture. The proposed real-valued FFT processor could simultaneously reduce the computation cycles and the hardware cost.

Although the design challenges shift from traditional metrics to reliability and security in these years, area and power consumptions are still important concerns in modern digital circuit design. A broad goal is to search over all possible implementations of a given DSP circuit and to find one that results the optimal area and/or power. One

promising research direction is to further explore the arithmetic redundancies that still exist in the state-of-the-art DSP computations. The solution can be twofold. On the one hand, area and/or power can be reduced by eliminating the redundancies (canonic RFFT is such an example). On the other hand, redundancies can also be exploited to correct errors in voltage over-scaling and precision over-scaling applications.

Furthermore, another exciting direction will be developing efficient and also sufficient secure VLSI DSP systems. One promising direction for future research is to employ reconfigurable architectures with low-cost control-flow. In many situations, we have the flexibility to choose a set of control signals to embed reconfigurability. Since securing many different control signals might be prohibitively costly, we could develop an approach that could minimize the overhead by selectively reconfiguring a small subset of components, while achieving a desired level of security.

References

[1] Henrik V. Sorensen, Douglas L. Jones, Ml Heideman, and C. Sidney Burrus. Real-valued fast Fourier transform algorithms. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(6):849–863, 1987.

[2] Hsiang-Feng Chi and Zhao-Hong Lai. A cost-effective memory-based real-valued FFT and hermitian symmetric IFFT processor for DMT-based wire-line transmission systems. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 6006–6009, 2005.

[3] Manohar Ayinala, Yingjie Lao, and Keshab K. Parhi. An in-place FFT architecture for real-valued signals. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(10):652–656, 2013.

[4] Yevgen Voronenko and Markus Puschel. Algebraic signal processing theory: Cooley–Tukey type algorithms for real DFTs. *IEEE Transactions on Signal Processing*, 57(1):205–222, 2009.

[5] Mario Garrido, Keshab K. Parhi, and Jesús Grajal. A pipelined FFT architecture for real-valued signals. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(12):2634–2643, 2009.

[6] Manohar Ayinala, Michael Brown, and Keshab K. Parhi. Pipelined parallel FFT architectures via folding transformation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(6):1068–1081, 2012.

[7] Sayed Ahmad Salehi, Rasoul Amirfattahi, and Keshab K. Parhi. Pipelined architectures for real-valued FFT and hermitian-symmetric IFFT with real datapaths. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(8):507–511, 2013.

[8] Manohar Ayinala and Keshab K. Parhi. FFT architectures for real-valued signals based on radix-2^3 and radix-2^4 algorithms. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(9):2422–2430, 2013.

[9] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time signal processing*. Prentice-hall Englewood Cliffs, 1989.

[10] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[11] Richard C. Singleton. An algorithm for computing the mixed radix fast Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 17(2):93–103, 1969.

[12] Glenn D. Bergland. A fast Fourier transform algorithm using base 8 iterations. *Mathematics of Computation*, 22(102):275–279, 1968.

[13] Pierre Duhamel and Henk Hollmann. Split radix FFT algorithm. *Electronics Letters*, 20(1):14–16, 1984.

[14] Shousheng He and Mats Torkelson. A new approach to pipeline FFT processor. In *Proceedings of the 10th International Parallel Processing Symposium (IPPS'96)*, pages 766–770, 1996.

[15] Megha Parhi, Yingjie Lao, and Keshab K. Parhi. Canonic real-valued FFT structures. In *48th Asilomar Conference on Signals, Systems and Computers*, pages 1261–1265, 2014.

[16] Bevan M. Baas. A low-power, high-performance, 1024-point FFT processor. *IEEE Journal of Solid-State Circuits*, 34(3):380–387, 1999.

[17] Shousheng He and Mats Torkelson. Design and implementation of a 1024-point pipeline FFT processor. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 131–134, 1998.

[18] Lee Jeesung and Lee Hanho. A high-speed two-parallel radix-2^4 FFT/IFFT processor for MB-OFDM UWB systems. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 91(4):1206–1211, 2008.

[19] Minhyeok Shin and Hanho Lee. A high-speed four-parallel radix-2^4 FFT/IFFT processor for UWB applications. In *IEEE International Symposium on Circuits and Systems*, pages 960–963, 2008.

[20] Ridha Radhouane, Peter Liu, and Cory Modlin. Minimizing the memory requirement for continuous flow FFT implementation: continuous flow mixed mode FFT (CFMM-FFT). In *IEEE International Symposium on Circuits and Systems*, volume 1, pages 116–119, 2000.

[21] Byung G. Jo and Myung H. Sunwoo. New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(5):911–919, 2005.

[22] Anthony T. Jacobson, Dean N. Truong, and Bevan M. Baas. The design of a reconfigurable continuous-flow mixed-radix FFT processor. In *IEEE International Symposium on Circuits and Systems*, pages 1133–1136, 2009.

[23] Chen-Fong Hsiao, Yuan Chen, and Chen-Yi Lee. A generalized mixed-radix algorithm for memory-based FFT processors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(1):26–30, 2010.

[24] Dionysios Reisis and Nikolaos Vlassopoulos. Conflict-free parallel memory accessing techniques for FFT architectures. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(11):3438–3447, 2008.

[25] Pei-Yun Tsai and Chung-Yi Lin. A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(12):2290–2302, 2011.

[26] Alice Wang and Anantha P. Chandrakasan. Energy-aware architectures for a real-valued FFT implementation. In *Proceedings of the 2003 international symposium on low power electronics and design*, pages 360–365, 2003.