A RECONFIGURABLE SYSTOLIC ARRAY ARCHITECTURE FOR MULTICARRIER WIRELESS APPLICATIONS

by

Huong Ho

A Thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Ph.D. in Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering

Department of Electronics

Carleton University

June, 2009

Your file Votre référence
ISBN: 978-0-494-60115-0
Our file Notre référence
ISBN: 978-0-494-60115-0

Abstract

Multicarrier (MC) wireless communication technologies depend heavily on efficient realization of digital signal processing functions. For high data rate communication, these DSP functions must be realized by means of parallel signal processing techniques. Fortunately, the principal DSP building blocks for MC communications such as DFTs and FIR filter banks lend themselves readily to parallel processing architectures. However, the realization of MC communication and MC circuitry within a Cognitive Radio (CR) platform or Software Defined Radio (SDR) may impose further requirements on the DSP portion namely reconfigurability. In terms of these DSP building blocks, the reconfigurable nature of the circuits could for example involve changes to: dimensionality, functionality, as well as, filter coefficient and "twiddle factor" values.

This dissertation outlines the design, simulation and circuit implementation of a new dynamically reconfigurable systolic array (RSA) architecture that is capable of supporting a wide range of simple and complex DSP functions for MC wireless applications. The RSA architecture presented here supports the computations of Polyphase filters, IDFT/DFTs, and IDFT-Polyphase/Polyphase-DFT for input signals consisting of N channels. The IDFT-Polyphase and Polyphase-DFT functions are usually employed to perform up and down conversion of composite FDM signals. A circuit that supports dynamic reconfiguration of IDFT-Polyphase/Polyphase-DFT functions could be used for the up/down conversion of FDM signals consisting of different number of

subcarriers or channel bandwidths according to requirements.

The RSA is a homogeneous, coarse-grained and regular architecture where each reconfigurable Processing Element (PE) is connected to its nearest neighbours. The RSA's homogeneous and regular characteristics facilitate circuit expansion to support applications that require higher data throughput performance. These characteristics also enable the circuit to support scalable computations for large data path applications within the constraint of available hardware resources. The RSA's regular characteristics also facilitate reconfiguration of Polyphase filter circuits for different number of channels and different number of taps.

The DFT circuit implementations are based on a novel algorithmic technique introduced to reduce the overall number of vector-matrix products to be mapped on the RSA arrays. The proposed technique enables the RSA-based DFT/IDFT circuits to support computations for N-point input sequence where N is not restricted to a power of two.

Novel multiplier design techniques have been proposed and applied to the implementation of multipliers in the PE cell. Comparison results for multiplier designs based on the proposed technique against circuits designed using existing techniques show that the proposed designs offer higher throughput performance and require less hardware than the latter.

Acknowledgements

I would like to extend my sincere thanks and appreciation to my thesis advisor, Dr. Tadeusz Kwasniewski for his invaluable advice, guidance and for giving me such a wonderful opportunity to work on this research.

Equally, I would like to thank my thesis co-advisor Dr. Valek Szwarc for accepting the difficult task of overseeing this work from beginning to completion and especially for being patience during the course of my study. Dr. Szwarc is also my supervisor at Communications Research Centre in Ottawa and I have been extraordinarily fortunate in having worked in a pleasant environment under his supervision. I could never have embarked and started all of this without his advice, encouragement and guidance.

I am also grateful to Dr. Miodrag Bolic, Dr. Calvin Plett, Dr. Ian Marsland and Dr. Dhamin Al-Khalili, who are members of my thesis committee, for taking time out of their busy schedules to consider this work. I am extremely grateful for their constructive comments and suggestions on this thesis.

This thesis arose in part out of years of research that has been done at CRC. I wish to extend my warmest thanks to Dr. Chun Loo for providing me with materials related to Group Demultiplexer technologies that help generate data for simulations of Polyphase filter circuits.

My parents deserve special mention for their encouragement and persistence confidence in me. I am also grateful to my brothers and sisters for being a constant source of joy and motivation in all my life.

Last but not least, thanks to my husband Nam Ninh for his patience and support during this long and difficult process.

Table of Contents

CHAPTER 3: DSP FUNCTIONS FOR MULTICARRIER APPLICATIONS

CHAPTER 4: RECONFIGURABLE SYSTOLIC ARRAY ARCHITECTURE

CHAPTER 5: LOW COMPLEXITY RSA CIRCUIT DESIGN

CHAPTER 6: RSA CIRCUIT CONFIGURATIONS

List of Figures

List of Tables

List of Acronyms

ASIC	Application Specific Integrated Circuit
AU	Arithmetic Unit
BER	Bit Error Rate
CB	Complex Butterfly
CR	Cognitive Radio
CRC	Communications Research Centre
CSA	Carry Save Array
CSD	Canonic Signed Digit
CSE	Common Sub-Expression Elimination
CSE-BitSlice	Common Sub-Expression Elimination-Bit Slice
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSoC	Configurable System on Chip
CU	Control Unit
DAB/DVB	Digital Audio Broadcasting/Digital Video Broadcasting
DFT	Discrete Fourier Transform
DFS	Dynamic Frequency Selection
DIT	Decimation in Time
DSP	Digital Signal Processor
DS-CDMA	Direct Sequence- Code Division Multiple Access
FDM/CDMA	Time Division Multiplexing/Code Division Multiple Access
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Arrays
GD	Group Demultiplexer
GM	Group Multiplexer
GPP	General Purpose Processor
GC-IN	Global Column Input
GC-OUT	Global Column Output
GR-IN	Global Row Input
GR-OUT	Global Row Output
IDFT	Inverse Discrete Fourier Transform
LC-IN	Local Column Input

LR-IN	Local Row Input
LR-OUT	Local Row Output
LP-WPAN	Low power-Wireless Personal Area Network
MA	Multiplier Accumulator
MC	Multicarrier
MC-CDMA	Multi Carrier- Code Division Multiple Access
MC-DS-CDMA	Multi Carrier-Direct Sequence- Code Division Multiple Access
MOPS	Mega Operations Per Second
MSPS	Mega Samples Per Second
MU	Memory Unit
NC-MCM	Non Contiguous Multi Carrier Modulation
PE	Processing Element
QDMA	Quadrature Division Multiple Access
QoS	Quality of Service
RSA	Reconfigurable Systolic Array
R4B	Radix-4 Booth
RECBAR	Reconfigurable Chain-structured Butterfly Architecture
SDR	Software Defined Radio
SoC	System on Chip
SW	Switch
TDM/CDMA	Time Division Multiplexing/Code Division Multiple Access
WLAN	Wireless Local Area Network
WiMAX	Worldwide Interoperability for Microwave Access

Chapter 1

Introduction

1.1 Motivation

Multicarrier (MC) techniques have been widely used for data transmission in the last decades and could be found in many wireless applications [Wang'00]. The main characteristic of MC data transmission schemes is that they use multiple subcarriers to transmit data in parallel fashion across the channel. One of the most commonly used MC transmission techniques is Orthogonal Frequency Division Multiplexing (OFDM). OFDM technique has been used for MC signal modulation or used for channel estimation in audio/video broadcasting (DAB/DVB), WLAN, WiMAX, Terrestrial Digital Multimedia/Television Broadcasting (DMB–T), personal Area Networks (PAN) and mobile communication systems [Geie'01, Garc'06, Yang'01, ETSI'04, ETSI'06], and [802.11a, 802.15.4, 802.16a]. For these applications, the number of signal subcarriers used for carrier modulation is determined by the channel bandwidth or the transmission frequency. Take the DVB transmission mode for example; the number of OFDM subcarriers used for modulation of signals transmitting over the 8 MHz, 7MHz, 6MHz or 5MHz channel is either 2K, 4K or 8K. For DAB applications however, the number of subcarriers used for modulation of signals transmitted over the frequency range of 1452 MHz to 3 GHz is either 256, 512, 1K or 2K. FFT circuits for WLAN and DMB–T applications have been designed based on the specific number of subcarriers required by the standards [Yang'02, Khan'03]. For DAB/DVB applications however, a lot of research

has been put into the design of FFT circuits that support multi mode modulation [Wang'05, Yuh'06, Kuo'03, Hung'04, Cort'06].

Recently, the OFDM technique has also been proposed for use in the development of non-contiguous MC modulation (NC-MCM) systems for dynamic frequency selection (DFS) applications such as cognitive radio (CR) [Wygl'06, Marc'05, Zhan'07]. In a CR environment, the NC-MCM technique would allow unlicensed users to temporarily access a licensed spectrum band by disabling OFDM subcarriers, which would interfere with existing transmissions. A combination of OFDM and spread spectrum such as the multicarrier code division multiple access (MC-CDMA) technique has also been explored for its reconfigurability for CR applications [Sara'07].

A DFT/IDFT based circuit with dynamic reconfigurability of channel configurations and data rates could be part of the solution to address the issue of DFS in a CR environment. A dynamically reconfigurable DFT/IDFT circuit could also be used to support different OFDM modulation modes required by DAB/DVB applications.

Polyphase filtering is a common technique frequently used for interpolation or decimation of digital signals in speech and image processing applications [Vaid'90, Flie'94, Zahh'03]. Filtering of the signals obtained from an interpolated or decimated signal can be efficiently performed by a Polyphase filter consisting of a bank of FIR filters, each operating with a different phase shift [Re'02,Gere'00]. A Polyphase filter that supports reconfiguration for the number of channels and filter coefficients would provide a valuable capability for systems operating in a multirate communications environment [Mehr'05, Rama'00].

Software defined radio (SDR) technology has been used in CR systems to enable users to change the mode of operation on the fly via software based on a programmable hardware platform. For SDR applications such as emergency response, a reconfigurable multicarrier baseband processor design based on the Polyphase-FFT/IFFT-Polyphase algorithm could be used as part of a flexible system to enable a group of firefighters, police and medical personnel to communicate with one another on different frequency bands. In the US, ten different RF frequency bands from 25 MHz to 869 MHz have been assigned and provide spectrum bandwidths from 2 MHz to 26 MHz to local, state and federal public safety agencies for critical voice communications. Hence, a city's firefighter department may be assigned a spectrum bandwidth in one RF frequency while the city's emergency medical services assigned a different bandwidth in another RF frequency and the result is that they cannot talk to each other.

Supposing that the firefighters were initially assigned a spectrum of 5 MHz for signal reception consisting of 200 channels of 25 kHz per channel on the frequency bands of 450 MHz-470 MHz. The police on the other hand, transmit signals on a 4 MHz bandwidth in the frequency bands of 138 MHz-144 MHz consisting of 160 channels and the medics use a signal bandwidth of 1 MHz in the frequency bands of 25 MHz-50 MHz consisting of 40 channels for signal transmission. Thus in order to communicate to both the police and the medics, the firefighter's communication devices have to be able to support reception over the RF frequency bands of 138 MHz-144 MHz and 25 MHz-50 MHz in bands of 4 and 1 MHz consisting of 160 and 40 channels respectively.

In this case, a hardware platform consisting of up/down converters together with reconfigurable multicarrier baseband processors could be used to allow the firefighter's

devices to dynamically switch to the desired mode of operation. While the RF part of the hardware platform supports multi-band frequencies, the Polyphase-FFT technique enables the reconfigurable baseband processor to perform up/down conversion of the multicarrier signals consisting of different bandwidths and channel configurations.

In a mesh network [Arcu'03, Good'03, Poor'03], each user or node can be set up as a receiver and/or a router to relay messages to its neighbors. A user's data is forwarded through the network by multi-hop routing to an access point connected to the service provider network. The network boundaries in this case do not need to be defined and the hardware at the access point does not need to go through major reconfiguration process to support new nodes. As a result, mesh network architecture offers system scalability where the number of nodes can be dynamically changed. Recently, wireless mesh network technology for sensor applications has gathered a lot of attention [Rica'05, Akyi'02]. Sensor networks for military applications can be found in battlefield surveillance and enemy tracking. Civil applications can be found in habitat monitoring, environment observation, health and other commercial applications. Most sensor networks have limited storage capacity, are power constrained, and have limited communication bandwidth. A reconfigurable processor whose channels of equal or different data rates can be dynamically selected using the Polyphase-FFT technique offers a potential solution to the problem of power and bandwidth constraints in sensor networks.

Most SDR vendors offer limited reconfigurability and low computational capacity products whose baseband processing tasks are performed by general purpose processors (GPP) or Digital Signal Processors (DSP) [MRC6011, SDR-3000, Adaptacell]. The SoC LSI processor, introduced by Fujitsu, offers high throughput performance with common processing functions performed by accelerator modules whose configuration modes are controlled by a CPU [Nish'06]. The time required to download configuration data onto

the LSI processor for a wireless application is approximately 20 milliseconds. FPGA-based reconfigurable circuits offer large logic resources for parallel processing which is important in applications that require high computational capacity and high throughput. However, reconfiguration of an FPGA is time consuming and if the baseband system involves multiple FPGAs, the time required to reconfigure all of them would be prohibitive.

For sensor networks, a microcontroller is the preferred choice of hardware for the implementation of sensor nodes due to its low power consumption and its ability to be reprogrammed. As services and performance demanded by new applications increases, traditional hardware used for the implementation of digital processing functions in sensor and SDR systems will fall short in terms of flexibility and performance. All of these constraints require novel architectures and circuit design techniques that maximize hardware capabilities while keeping the system flexible for reconfiguration and inexpensive to deploy and maintain.

The Polyphase-FFT technique based on a circuit consisting of a Polyphase filter and an FFT has been effectively exploited in the implementation of transmultiplexers for terrestrial applications as well as demodulators for on board satellite processors [Wein'71, Taka'97, Bela'74, Sche'81, Gock'92, Saye'92, Ho'99, Ho'02]. A combination of a Polyphase filter and an FFT circuit provides a computationally efficient method of down converting a group of N FDM signals in baseband. Similarly, a combination of IFFT and a Polyphase filter circuit executes the complementary operation, namely the frequency division multiplexing of a group of subcarriers at baseband. One of the requirements of an FFT-Polyphase algorithm is that the number of input signals has to be a power of two otherwise the FFT technique is not applicable.

Fast Fourier Transform (FFT) algorithms have been widely used for DFT circuit implementations to achieve efficient hardware usage [Smit'95]. The FFT algorithm is normally used to compute an N point DFT where N is a power of two. Split-Radix or

mixed-radix FFT algorithms could be used for DFT computations more effectively if N is a composite number. These FFT algorithms have a common characteristic in that they are based on an iteration process where the successive term is calculated using a recursion. Circuit implementations based on FFT algorithms are usually inflexible and do not support reconfiguration due to the long distance connections between the circuit building blocks as a result of signal recursion. On the other hand, DFT circuit implementations based on direct DFT/IDFT algorithm with modular hardware structures are potentially suitable for reconfiguration. However, in exchange for reconfiguration flexibility, DFT-based circuit implementations normally require more logic resources. As Polyphase-DFT circuits offer significant flexibility, DFT/IDFT design techniques need to be investigated to ensure low hardware complexity for circuit implementation.

For wireless communication systems, DSP circuits designed with low hardware complexity are always desirable to achieve high computation throughput and low power consumption. The multiplier is the main building block in many DSP circuits and it is the key element that determines the system performance. Multiplier modules also consume more power than any other building block in the DSP circuit thus they are the logical target for circuit optimization. Most optimization techniques aimed at reducing hardware complexity for common DSP circuits rely on the fact that one or more components in the multiplier circuit are constant [Hary'96, Potk'96, Pask'99, Park'02, Nguy'00, Hosa'05, Wu'04]. However, generic multiplier circuits are often employed in systems where reconfigurability is required. Many optimization techniques proposed for generic multiplier circuit implementation such as Modified Booth and Canonic Signed Digit (CSD) have been widely published in the literature [Wu'98, Lin'04, Hewl'00, Full'98]. On the other hand, Common Expression Elimination (CSE) techniques have also shown that a significant reduction of hardware and throughput improvement could be achieved for multipliers with constant multiplicands. In fact, simulation results presented in [Wu'04, Hary'96, Vino'04] have shown that considerable logic resources could be saved

using the CSE technique for FIR filter and DFT/DCT circuit implementations. The common digit elimination technique used in constant multiplier designs presents an interesting option that needs to be explored to address the optimization problem for generic multiplier circuit implementations.

1.2 Thesis Objectives

The goal of this thesis is to implement a reconfigurable systolic array (RSA) with high throughput performance and low hardware usage that is suitable for multicarrier applications. The proposed circuit must support a class of DSP functions including FIR filters, Polyphase filter banks, phase shifters, DFT transforms of arbitrary dimensionality, and Polyphase-DFT. The RSA must also support circuit expansion for parallel processing. The proposed circuit must also support high throughput data based on time sharing computation. For reconfiguration flexibility, the RSA circuit must be implemented on the basis of a modular architecture consisting of reconfigurable arithmetic processors and reconfigurable signal routing modules.

For efficient hardware usage and high throughput performance, a multiplier design approach that combines CSE and Bit-Slice (CSE-Bitslice) techniques will be explored for the implementation of Processing Element (PE) circuit.

For N-point DFT circuit configurations where N is not restricted to a power of two, algorithmic optimization techniques that help to reduce the overall number of vector-matrix products are going to be investigated for the mappings of the DFT circuit on to the RSA.

1.3 Thesis Outline

In the following paragraphs, the contents of individual chapters are outlined.

Chapter 2. This chapter describes reconfigurable architectures and systems that have been proposed in the literature for data path circuits and wireless applications. Existing design techniques for generic multiplier circuits are also described in this section. In addition, existing design approaches and algorithms that have been employed for low complexity and flexible DFT/IDFT circuit implementations are also examined here.

Chapter 3. The first section of this chapter describes the FDM-based MC system and a set of DSP functions that are relevant to multicarrier applications. The important features and constraints of each function and of the Polyphase-DFT/IDFT-Polyphase system are analyzed. Subsequently, simulation parameters and representative configurations based on the analysis and of relevance to work presented in later chapters are described.

Chapter 4. The first section of this chapter describes the proposed RSA architecture. The architectures of the processing elements (PE) and the interconnection switch (SW) are described in the second section. Interconnection schemes between PE and SW cells and internal signal routing schemes within the PE and SW cells are explained in the following section. The last section of this chapter describes the design and realization process for the RSA circuit including circuit modeling, functional testing and Field Programmable Gate Arrays (FPGA)/Application Specific Integrated Circuit (ASIC) implementations.

Chapter 5. This chapter describes the design of the RSA circuit and its building blocks. Design techniques employed to reduce logic resources for the RSA circuit and its building blocks based on CSE-BitSlice technique are described. An analysis of the total number of additions required for multiplier circuit designs based on the CSE-BitSlice

technique and a comparison with designs based on the Carry Save Array (CSA) and Radix-4 Booth (R4B) algorithms is presented.

Chapter 6. In this chapter, RSA circuit configurations for DSP functions depicted in Chapter 3 are illustrated. The design technique proposed for DFT/IDFT circuit implementations to achieve hardware reduction is described. RSA circuit configurations for FIR filter, Polyphase filter, Polyphase-DFT/IDFT-Polyphase are also presented in this chapter. Several RSA circuit configurations for N-point DFT computations where N consists of even and odd length are also presented. Estimations of computation complexity in term of complex multiplication operations and throughput of RSA-based DFT/IDFT circuits are presented and compared to DFT mappings on existing reconfigurable structures.

Estimation results in term of size of the arrays and the expected performance of the RSA circuits configured for sequential and parallel mappings of Polyphase-DFT/IDFT-Polyphase function are presented. The estimation results can be used to determine whether parallel or sequential mapping should be used for the configuration of a Polyphase-DFT/IDFT-Polyphase circuit based on specific system requirements and hardware constraints.

Chapter 7. In this chapter FPGA implementations of the RSA circuit, its building blocks and several relevant circuits are illustrated. Multiplier circuit designs implemented on FPGAs using the CSE-BitSlice techniques described in chapter 5 are presented and the overall logic resources and throughput performance of these implementations are compared with circuits designed using the CSA and Radix-4 Booth algorithms. Simulation results of the RSA circuits implemented on FPGAs are also presented. It should be noted that the RSA and the CSE-BitSlice implementations are in no way limited to FPGAs. The FPGA technology is used to simply validate the circuit designs. Hardware complexity and throughput performance of the PE cell implementations based on different algorithms and techniques are described and discussed. RSA circuit

mappings of several DFTs configured for parallel and sequential modes of operation are described and simulation results are presented. Simulations results for Polyphase filters, DFT/IDFT and Polyphase-DFT/IDFT-Polyphase configured on the RSA circuit are also presented. Comparisons of hardware complexity and performance of FIR/Polyphase filter and DFT functions mapped on RSA structures and the same functions mapped on representative reconfigurable circuits are described in the last section of this chapter.

Chapter 8. This chapter reviews the most important contributions of this thesis and provides a short discussion of possible future improvements for the RSA design.

Chapter 2

Background and Literature Survey

2.1 Introduction

In this chapter existing reconfigurable architectures and systems are considered in Section 2.2. Multiplier circuit design techniques that have been published are illustrated in Section 2.3. In Section 2.4, existing design techniques for DFT/IDFT circuits with low hardware complexity and scalablability are considered.

2.2 Reconfigurable Architectures and Systems

Most existing reconfigurable hardware platforms have been implemented based on Digital Signal Processor (DSP) and Field Programmable Gate Array (FPGA) technologies or a combination of these devices. On the other hand, existing reconfigurable architectures that support system reconfiguration without the need of hardware re-design have often aimed at Application Specific Integrated Circuit (ASIC) implementations. The ability of several existing reconfigurable architectures and systems to support reconfiguration of complex DSP functions is discussed in the following paragraphs.

2.2.1 Reconfiguration Systems Using DSPs

DSPs have been widely used in many communication circuits and systems [Jone'97, Kama'05]. A DSP can be reprogrammed to compute different signal processing

functions but usually offers low throughput performance due to the sequential execution nature of its architectures. Furthermore, implementation of reconfigurable new functions does take time since the new algorithm has to be designed and verified before being programmed onto the device.

2.2.2 FPGA-based Reconfigurable Systems

FPGAs offer potentially more computing capacity as well as better circuit throughput performance than DSP processors. However, these devices require that circuit design and implementation procedures be repeated for every reconfiguration. Moreover, the reprogramming process for FPGA devices is also time consuming especially if the reconfigurable systems consists of multiple FPGA devices.

Reconfigurable computing based on systems consisting of multiple FPGAs or a combination of processors and FPGAs has been proposed to support complex DSP functions or data path based applications where time-consuming computations need to be sped up. Among the reconfigurable systems that employ a combination of FPGAs and microprocessors are the GARP [Haus'97], the PLEIADES [Zhan'00], the HERA [Wang'04] and the MPSoC [Zhan'07]. These systems use FPGAs to implement the reconfigurable module and microprocessors are used to load and execute configuration data.

Two well-known FPGA-based reconfigurable computing platforms designed to be used as data path accelerators are the DECPeRLE and the SPLASH-II systems. The Splash II platform [Arno'92] built at the Supercomputer Research Center has been implemented based on systolic arrays of FPGAs where redundant boards are added or removed to make the system expandable. The SPLASH-II system has been used to achieve high performance computation for image processing applications. The

DECPeRLE [Vuil'96] system developed at the DEC Paris Research Laboratory contains arrays of hardwired FPGAs that are used as a numeric accelerator for applications such as neural networks, cryptography, and image classification. The main computing engine of this coarse-grained array system is a PE that contains some combination of multipliers, registers, and memories. An FPGA may be used to map a PE or a large block of PE cells that provides the arithmetic resources required to carry out a specific function [Vill'98].

Although these systems offer more computational flexibility as compared to DSP-based systems, due to the time-consuming process required to program FPGAs, run-time reconfigurations are not supported. Another limitation of these systems relates to the complex process of mapping applications onto the hardware platform. The mapping process of an application onto multi-FPGA based hardware platforms usually has to be carried out either by software or manually. This process involves the partition of the targeted application into clusters, where each cluster is then configured on an FPGA in the system.

To improve the delay caused by the configuration process, partially reconfigurable FPGAs have been introduced [Atmel, Xilinx]. The reconfigurable portion of the design in this case could be reprogrammed onto a subsection of the device while the remaining logic resources are used for the static part of the design. The DISC platform proposed by researchers at Brigham Young University [Wirt'95] has been developed using Atmel's FPGAs and the reconfigurable portion of the devices has been used to support the demand-driven modification of its instruction set. Each instruction in the set is treated as a removable module and is mapped onto the DISC platform as demanded by the executing program. These instructions occupy FPGA resources when needed and could be removed to free the resources for other instructions. The architectures proposed in [Xzha'07, Merm'05, Choi'07] have also relied on the partial reconfigurability of these FPGAs for circuit reconfiguration. For these circuits, the tasks that require frequent reconfiguration are implemented onto a section of the FPGA that

has been reserved for reconfiguration while the remaining tasks of the targeted function are mapped onto the static part of the device.

2.2.3 Reconfigurable Architectures

Several well-known coarse-grained reconfigurable architectures such as KressArray [Harn'95, Harn'99], MATRIX [Mirs'96], RAW [Wain'97], MORPHOSYS [LuSi'99], and CHESS [Mars'99] have been proposed for datapath applications. The target application of these architectures is a datapath accelerator where the reconfigurable arrays often consist of ALUs with low or medium logic resources. The DReAM [Beck'02] is also a coarse-grained architecture proposed for mobile communication applications. This is a SoC architecture consisting of a combination of microcontroller, memory and logic blocks synthesized on a single ASIC. Multiplications are mapped onto memory blocks while other arithmetic functions are configured on the DReAM arrays.

Several reconfigurable architectures implemented on ASICs for DSP computations have been proposed in [Also'02, Gray'00, Sapi'90, Wosn'96, Zhan'05, Wall'05]. These reconfigurable architectures have been designed for application specific purposes where the circuit consists of arithmetic building blocks that contain low or medium logic resources.

The DRAW system proposed in [Also'02] supports DSP functions for baseband processing of WCDMA applications such as FIR filtering, Gold code generation, and Turbo coding. For illustration purposes, a 4-tap FIR filter with 8-bit input data and 16-bit coefficients that requires eight PE cells has been mapped on a DRAW structure. Simulations of the processing element design based on Fujitsu 0.25 μm standard cell ASIC technology shows that it supports processing power of 20 MOPS.

The ARCA architecture proposed in [Zhan'05] is an asynchronous array structure consisting of fine-grained logic cells designed to support low power applications. Each logic cell in the array shifts data in and out via four input data and four output data ports each of which is 1-bit wide. The structure only supports simple arithmetic functions such as addition, subtraction and data shifting. An 8-bit ripple carry adder realization needs a total of 8 logic cells. This fine-grained architecture is not suitable for data path applications. The ARCA architecture also does not support computations of DSP functions that require complex arithmetic operations.

The architecture proposed in [Sapi'90] is a homogeneous array of computing cells that offers limited reconfigurability and self-testing capability. The PE cells in this array structure have been designed for FFT circuit implementations exclusively.

The modular architecture proposed in [Wosn'96] has been designed for real-time object detection in image processing applications.

The RADComm architecture [Gray'00] is a systolic array of homogenous PE cells targeting high-speed data communications. This architecture supports a class of DSP functions and the main arithmetic engine in each PE cell consists of one real multiplier and one adder. DSP functions such as FIR or IIR filters can be mapped directly onto an array of PE cells. For more complex functions such as Goertzel's FFT and digital frequency synthesis, extra hardware needs to be incorporated onto the array to store, control, and shuffle data and coefficients required for the computation of these functions.

The CSoC architecture proposed in [Wall'05] consists of heterogeneous processing resources including a task processor and two clusters of reconfigurable cells. A function to be executed is first divided into several small tasks, which are mapped onto reconfigurable hardware sequentially. Performance results of several DSP functions mapped on a representative CSoC circuit where each cluster consists of 16 cells shows that a large number of execution cycles is required to compute each function.

While the RADComm offers better reconfiguration time than the CSoC architecture, reconfiguration for both architectures relies on a complex circuit mapping process. The fine-grained characteristic of the above architectures make them unsuitable for computation of functions that require complex arithmetic operations or large data path configurations.

2.3 Multiplier Design Techniques

Multiplication is one of the main arithmetic operations in DSP systems that perform computations involving complex functions such as FIR filtering, correlation or Fourier transform. For modern signal processing systems, low power multiplier circuits that support high throughput are required. Many design techniques have been proposed in the last decades for high speed, low complexity and low power multiplier circuit implementations.

The most common technique used for multiplier circuit implementations is the carry save array (CSA) algorithm [Kim'98, Khoo'99]. The CSA multiplier is frequently used in complex VLSI circuits due to its homogeneous and regular structure where neighboring cells are connected to one another by short wires. Many CSA-based techniques have been proposed for multiplier circuit implementations with low power consumption and high throughput performance. In [Econ'06, Wen'05, Fons'05], a multiplier with low power consumption is achieved by disabling certain logic blocks when a zero in the operand is detected or fast carry save adders are used to reduce switching activity of the circuit. The techniques proposed in [Erce'90, Cimi'96] aim at improving circuit throughput by using redundant or signed digit representations of partial products to reduce size of the carry propagate adder in the last stage of the array

multiplier. In [Asat'89], a pipelining technique is also proposed to achieve higher throughput for CSA multiplier circuits.

For low area requirements, multiplication techniques such as the Booth or Radix-4 Booth algorithms have been proposed [Boot'51, McSo'61]. These design techniques focus on reducing the numbers of partial products generated by the multiplier to lower the overall number of additions required. There are many techniques based on Modified Booth or Radix-4 Booth algorithms that have been proposed to either push for further reduction of circuit area [Shin'01] or to improve circuit throughput performance [Maki'96, Besl'02a, Besl'02b]. The technique proposed in [Shin'01] for multiplier circuit hardware reduction is based on employing a small number of adders to compute the addition of partial products iteratively. Redundant binary arithmetic has also been used in radix-4 Booth multiplier designs to improve circuit throughput performance [Makino'96].

For high throughput, residue number arithmetic (RNA) has also been proposed for multiplier circuit implementations [Tayl'84, Bayo'87, Jenk'85]. In residue number system (RNS), a large integer can be represented by a set of smaller integers based on a set of moduli. Arithmetic computation between two integers represented in the RNS is then performed on the respective residues independently. Multiplication between two RNA based numbers is fast since there is no carry chain between residue digits. However, outputs generated from an RNA based multiplier circuit need to be converted back to binary form which involves a complex computation process. Large logic resources required by the implementation of an RNA decoder would increase area as well as affect the multiplier circuit throughput performance [Ho'95].

Other techniques such as the Canonical Signed Digit (CSD) have also been used for multiplier circuit implementation [Chen'03, Kore'93]. Using CSD representation, the number of non-zero digits of a binary number is reduced, which in turn helps to reduce the number of additions in multiplier circuits [Hwang'79]. This technique is used mostly in multiplier circuits where at least one of the operands is a constant.

The idea of using common subexpression elimination (CSE) technique for constant multiplication optimization has been considered and is widely reported in literature [Potk'96, Mehe'95, Hary'96]. Hardware reduction of multiplier circuit designs based on CSE techniques rely on the fact that common digits belonging to the constant operand could be factored out which would help to reduce the overall number of partial products. Many multiplier designs where the CSE technique is used for the computation of vector and matrix multiplications have been published in the last decade. The vector multiplication technique proposed in [Macp'06, Pask'99, Demp'95] aims at hardware reduction for FIR filter designs where the filter coefficients are constant. The matrix multiplication and optimization techniques described in [Pasko'99, Macl'04] have been proposed for DFT or DCT circuit implementation.

A technique using digit coding has also been proposed and used for FIR filter design where both operands of the multiplier are variables [Youn'96]. This technique uses a lookup table to store the entire set of the multiplier partial products and hardware reduction is achieved when multiple variable multiplications such as scalar-vector multiplication is involved. The limitation of this technique is that the size of the look up table increases significantly if the number of variables is large.

2.4 Design Techniques and Architectures for DFT/IDFT

A large body of research on design techniques and architectures has been published on circuits employed in Fourier transform applications. To achieve efficient hardware usage and attain high throughput performance, FFT techniques have been extensively used for DFT circuit implementations [Chan'00, Yeh'03, Maha'04, He'98]. On the other hand, modular architectures consisting of arrays of processing elements have also been proposed for designs of FFT circuits incorporating scalability. The structure

based on an array of general-purpose multiprocessors proposed in [Bhuy'83] for DFT computation where the input sequence of N samples is a power of two is one such design. This proposed structure consists of P processors, each of which is used to perform butterfly computations. The total number of butterfly computations required to compute an N-point FFT is divided between the P available processors. Based on this mapping model, a P processor system could be used to compute an N point FFT for which $N \square P$. Other array architectures for FFT circuit implementation have also been proposed in [Tala'00, Bern'92]. These architectures offer scalability and hardware expansion to support computations of large FFTs. However, the input sequences in this case have to be shifted in sequentially due to the complex process of data permutation. The structure proposed in [Sapi'90] consists of 1-D array of processing elements where the N point input sequences are also shifted in sequentially. Each processing element in the array is designed to support one complex butterfly computation. The array architecture is homogeneous in nature so circuit expansion to support input sequences of larger size can be carried out simply by cascading extra PE cells to the array. This structure offers some reconfiguration flexibility but the trade-off is that it provides only low circuit throughput since input and output data can only be shifted in and out of the circuit serially.

DFT architectures that support Fourier transform computations of N sample input sequences where N consists of a composite number have also been considered. Two such architectures are proposed in [Seye'91] where the circuit could be used to compute an N-point DFT in parallel or sequentially. For parallel computation of an N-point DFT, a mesh like architecture consisting of N^2 multiply-add cells is used. For sequential computation, an array of m-by-m arithmetic cells is employed instead. Each arithmetic cell proposed for the sequential architecture consists of two complex multipliers and one complex adder. For sequential computation based on an m-by-m array $(N = m*m)$ structure, a total of $4m - 1$ clock cycles are required to collect the N point output sequences. Even though the sequential architecture proposed in this paper

offers low hardware complexity, its time*area complexity, however, is still too high when compared with the DFT architectures proposed in [Kung'80, Desp'79, Zhan'84].

Systolic architectures that support computations of N-point DFTs where N is a composite number that consists of two or more cofactors [Peng'97, Lim'99, Nash'05] have also been proposed. The architectures proposed in [Peng'97, Lim'99] offer an efficient way to calculate N-point DFTs based on an N_1- *by*- N_2 array where N is a product of two numbers $(N = N_1 * N_2)$. The base-4 algorithm for DFT computation based on systolic array architecture proposed by [Nash'05] offers a regular and low hardware complexity array architecture. The base-4 algorithm could be used to compute DFTs with N point input sequences where N is a multiple of 256.

Chapter 3

DSP Functions for Multicarrier Applications

3.1 Introduction

In this chapter, baseband processing functions suitable for wireless applications including DFT/IDFT, Polyphase filtering, phase shifting and Polyphase-DFT/IDFT-Polyphase are described. The first section of this chapter deals with DFT/IDFT, Polyphase filtering and phase shifting functions. In the second section, the Polyphase-DFT/IDFT-Polyphase based technique for multicarrier baseband processing is described. The important goals and constraints of the DFT/IDFT, Polyphase filtering, phase shifting and the Polyphase-DFT/IDFT-Polyphase function are analyzed. Subsequently, parameters of several representative systems for use in later chapters are presented.

3.2 DSP Functions for Wireless Applications

In this section, Polyphase filter, phase shifter and DFT/IDFT functions are described. In the following paragraphs, each function and its relevant parameters are considered for use in circuit design and simulation to be addressed in subsequent chapters.

3.2.1 Polyphase Filter

A system with the transfer function H(z) and the finite impulse response h(n) can be represented as follows [Flie'94]:

$$H(z)=\sum_{\lambda=0}^{M-1} z^{-\lambda} H_{\lambda}(z^{M}) \tag{3.1}$$

Here, the $z^{-\lambda}H_{\lambda}(z^{M})$ terms are the polyphase components of the transfer function H(z) and $H_0(z^M)...H_{M-1}(z^M)$ are the FIR filters. The Polyphase decimator and interpolator structures where the subfilters $H_0(z)...H_{M-1}(z)$ compute at 1/M the system clock rate are shown in Fig. 3.1a and Fig. 3.1b respectively.

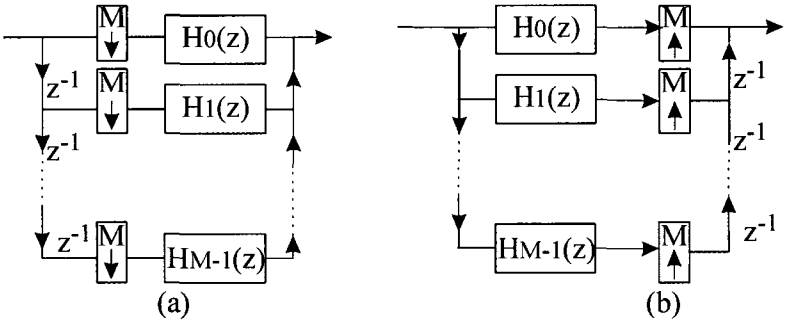


Figure 3.1: Polyphase structures (a) Decimator (b) Interpolator.

The FIR filters $H_0(z)...H_{M-1}(z)$ in the polyphase systems in Fig. 3.1a and Fig. 3.1b operate independently from one another and have the same frequency response but different phase shift. The response y(n) of an L tap FIR filter to an input x(n) is given by the following relation:

$$y(n)=\sum_{i=0}^{L-1} c(i)x(n-i) \tag{3.2}$$

In (3.2), c(i) are the filter coefficients. From (3.2), it is clear that each FIR filter consists of a sum of partial products generated by multiplication of the filter input and its coefficients. In a reconfigurable circuit, the filter coefficients and filter taps in each filter of the polyphase filter bank are variable and need to be updated whenever there are changes in channel conditions which require a system reconfiguration. For circuit design, a transpose FIR filter architecture offers a regular structure which facilitates circuit reconfigurations involving the number of taps. A transpose FIR filter structure is described in the figure below:

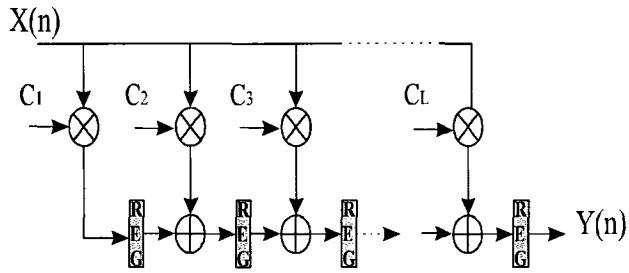


Figure 3.2: Block diagram of a transpose FIR filter.

3.2.2 Phase Shifter

In a Multicarrier (MC) system, phase shifting is usually performed by a circuit consisting of a number of individual phase shifters. Each of the phase shifters consists of a complex multiplier that multiplies the input data by a coefficient whose value corresponds to the phase shift associated with a particular channel. The phase shifting process performed by a phase shifter is given by the following equation:

$$y(n) = x(n) * e^{\frac{j\pi n}{N}} \tag{3.3}$$

In (3.3), x(n) is the input to the phase shifter; y(n) is the phase shifter output and $e^{\frac{j\pi n}{N}}$ is the phase shifter coefficient. For reconfigurable applications, coefficients used for phase shifting computation need to be updated whenever there are changes in the system which requires a set of new phase shift values. Thus, a phase shifter circuit has to be able to support complex multiplication computations when both operands are variable.

3.2.3 DFT/IDFT

For up and down conversion of FDM signals in baseband, FFT/IFFT has often been employed to compute the discrete Fourier transform required by the Polyphase-FFT algorithm [Bela'74, Taka'97]. Even though FFT/IFFT is an effective way to compute the Fourier transform for the sampled FDM signal, the algorithm can only be used on signals for which the number of samples in the input sequence is a power of two or consists of a composite number. Furthermore, the FFT/IFFT structure often lacks regularity and the need to shuffle data between butterfly computation stages makes the architecture relatively less flexible for reconfigurable applications than the techniques proposed in this work.

The direct DFT algorithm requires more arithmetic operations than the FFT, but the resulting hardware structure has a regular characteristic which facilitates circuit reconfiguration. DFT circuits also support computations for signals with N point input sequences where N is not restricted to be a power of two. A DFT is expressed by the following equation:

$$Y(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi nk}{N}} \qquad k = 0,...,N-1 \tag{3.4}$$

As seen from (3.4), each DFT output consists of a sum of partial products generated by the multiplication of the data inputs and a set of coefficients. For reconfigurable applications, the DFT/IDFT coefficients are variables which must be updated whenever there is a circuit reconfiguration which calls for a new set of DFT/IDFT coefficients.

Thus, design techniques aimed at DFT circuit implementation should be explored for low complexity and high throughput performance. To facilitate DFT coefficient reconfiguration, arithmetic modules in DFT circuits have to be designed to support complex multiplication computations where both operands of the multiplier are variable.

3.3 IDFT-Polyphase/Polyphase-DFT for Multicarrier Baseband Processing

IDFT-Polyphase and Polyphase-DFT algorithms enable the multiplexing and demultiplexing of FDM signals at the transmitter and receiver respectively. The N-channel IDFT-Polyphase and an N-channel Polyphase-DFT circuit are also known as a Group Multiplexer (GM) and a Group Demultiplexer (GD), respectively.

3.3.1 IDFT-Polyphase

An IDFT-Polyphase or GM circuit consists of three main building blocks: an IDFT, phase shifters and a Polyphase filter bank. For a system with N channels, the output FDM signal $Y(z)$ of the GM circuit is defined by the following equation [Taka'97]:

$$Y(z)=\sum_{i=0}^{N-1} z^{-i} H_i(-z^N) e^{(j\pi\frac{i}{N})} \sum_{k=0}^{N-1} e^{(j2\pi\frac{ki}{N})} X_k(-z^N) \qquad (3.5)$$

In (3.5), $X_k(-z^N)$ corresponds to the sampled QPSK modulated data being shifted into the GM circuit. The expression $e^{(j\pi\frac{i}{N})}$ represents the phase offset and $H_i(-z^N)$ represents the ith Polyphase sub-filter. The block diagram of the IDFT-Polyphase circuit is shown in Fig. 3.3.

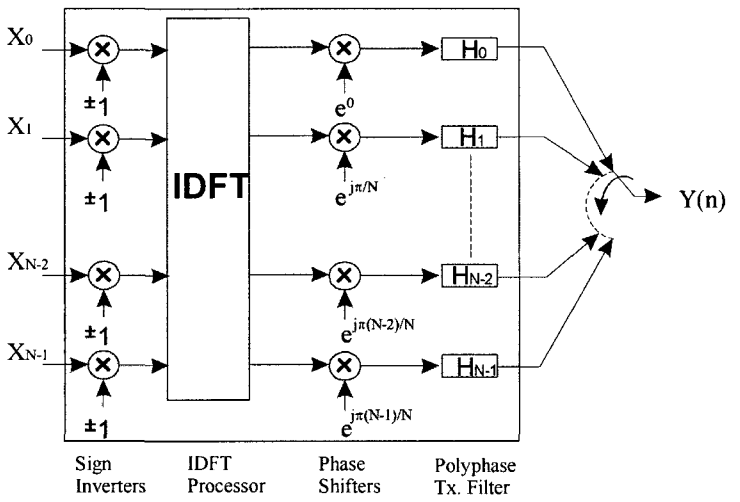


Figure 3.3: IDFT-Polyphase circuit block diagram.

3.3.2 Polyphase-DFT

The Polyphase-DFT or GD circuit demultiplexes the composite FDM carriers sampled at baseband and is denoted by the following equation:

$$X_k(-z^N) = \sum_{i=0}^{N-1} e^{(-j2\pi\frac{ki}{N})} e^{(-j\pi\frac{i}{N})} F_i(-z^N) Y_{N-i-1}(z^N) \tag{3.6}$$

In (3.6), $X_k(-z^N)$ is the k^{th} demultiplexed signal and $Y_{N-i-1}(z^N)$ corresponds to the sampled FDM input signal $Y(z)$. The symbol $F_i(-z^N)$ represents the i^{th} polyphase sub filter, derived from the decomposition of the receiver filter $F(z)$, and $e^{-\frac{j\pi i}{N}}$ represents the phase offsets. The filter $F(z)$ is typically a fourth order root Raised Cosine FIR filter. The subsequent processing by the DFT of the filtered and phase offset signals results in N down-converted channel signals. The block diagram of the Polyphase-DFT circuit is shown in Fig. 3.4.

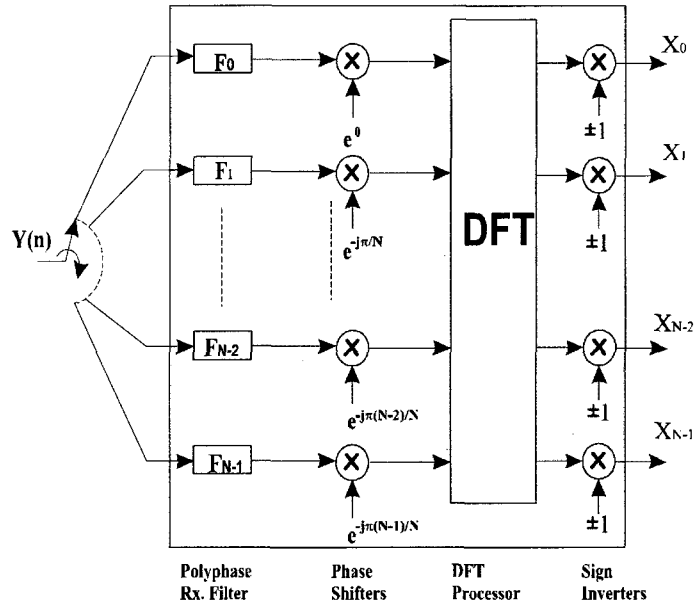


Figure 3.4: Polyphase-DFT circuit block diagram.

3.3.3 Polyphase-DFT and IDFT-Polyphase Circuit Design Issues

For reconfigurable applications, the parameters of the IDFT-Polyphase and Polyphase-DFT circuits need to be updated whenever there is a change in system reconfigurations. Changes in the system bandwidth, the number of channels or the channel bandwidth would require reconfiguration of the coefficients in the Polyphase

filter, phase shifters and DFT/IDFT building blocks. Changes in the number of channels would also require a reconfiguration of DFT/IDFT circuit to support the new system specifications. If the channel condition calls for better filtering performance, coefficient reconfiguration for the Polyphase filter would also be required. Similarly, new channel conditions might also require reconfiguration of the Polyphase filter and different filter taps.

All of the requirements illustrated in the previous paragraphs have to be taken into consideration for the design of the RSA circuit that is suitable for MC applications.

3.4 Parameters of Representative DSP Circuits

In this section, parameters of several representative DSP circuits for use in later chapters are presented. The sets of parameters for the polyphase filter and the polyphase-DFT system are extracted from the results of the analysis on intersymbol and interchannel interference effects in an 8-channel digital group demodulator done at CRC [Seco'95, Seco'96]. The effects of quantization of filter coefficients have also been taken into account in this analysis.

3.4.1 Parameters of a Representative Polyphase Filter Circuit

A set of parameters for a representative polyphase filter circuit is identified in Table 3.1 for subsequent circuit implementation.

Table 3.1 Parameters of a representative polyphase filter circuit.

Polyphase Filter Circuit Parameters	Number of FIR filters: 4 (complex)
	FIR filter characteristics [Seco'95, Seco'96]: Kaiser Fourth Root Nyquist roll off = 0.4 β=2
	FIR number of taps: 8 taps
	FIR coefficient quantization: 18 bits (complex)
	Input: 24 bits (complex) Output: 34 bits (complex)

3.4.2 Parameters of a Representative Phase Shifter Circuit

A set of parameters associated with a representative phase shifter circuit is identified in Table 3.2 for subsequent circuit implementation.

Table 3.2 Parameters of a representative phase shifter circuit.

Phase Shifter Circuit Parameters	Phase shifter coefficient quantization: 18 bits (complex)
	Input: 24 bits (complex) Output: 34 bits (complex)

3.4.3 Parameters of Representative DFT Circuits

Table 3.3 describes a set of parameters associated with several representative DFT circuits for subsequent circuit implementation.

Table 3.3 Parameters of several representative DFT circuits.

DFT Circuits Parameters	Number of points (N): 5, 9, 10,12, 16
	DFT coefficient quantization: 18 bits (complex)
	Input: 24 bits (complex) Output: 34 bits (complex)

3.4.4 Parameters of a Representative Polyphase-DFT System

In this section, a set of parameters associated with a representative polyphase-DFT system is identified for subsequent circuit implementation. The set of parameters for the 8-channel Polyphase-DFT system is selected based on the result of BER performance simulations [Seco'95, Seco'96] and is depicted in Table 3.4. The BER performance of the selected Polyphase-DFT system is shown in Fig. 3.5

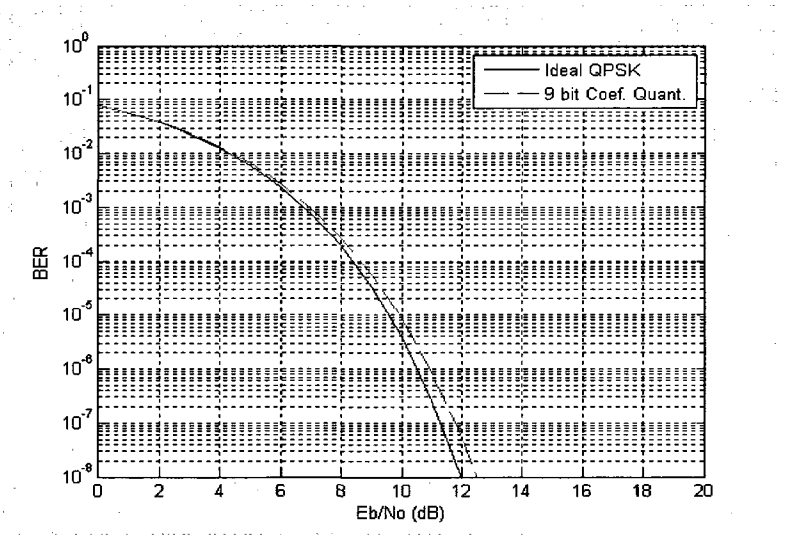


Figure 3.5: Performance of the 8-channel Polyphase-DFT based on the selected set of parameters depicted in Table 3.4.

Table 3.4 Parameters of a representative 8-channel Polyphase-DFT system.

System Simulation parameters	Input modulation: QPSK Number of inputs: 8 Channel separation: 1.544 MHz System bandwidth: 12.352 MHz Filtering: Full Raised Cosine Nyquist
System Input/Output	Input: 24 bits (complex) Output: 34 bits (complex)
Polyphase filter	Number of FIR filters: 8 (complex)
	FIR filter characteristics: Kaiser Fourth Root Nyquist roll off = 0.4; β=2
	FIR number of taps: 5 taps
	FIR coefficient quantization: 18 bits (complex)
DFT	Number of points: 8 Coefficient quantization: 18 bits (complex)

Chapter 4

Reconfigurable Systolic Array Architecture

4.1 Introduction

This chapter describes the RSA architecture and its building blocks. The first section contains the overall description of the RSA architecture, which consists of arrays of processing elements and routing switch cells. The architecture of the individual processing elements and the switch is also described in this section. Functional configurations for the processing element for different arithmetic operations are also illustrated. Configuration options for the switch in support of different signal routings are explained in detail. The ability of the RSA architecture to support hardware expansion for applications that require high throughput performance and to support scalable computation of signals with large data paths is discussed in sections 4.3 and 4.4. Finally, a description of the design and implementation process for the RSA circuit is provided in the last section.

4.2 The Reconfigurable Systolic Array Architecture

The RSA is a 2D systolic array architecture consisting of identical processing elements that could be configured to perform different DSP functions. The processing elements in the array are interconnected with one another by a network of switches. There are two main building blocks in the RSA architecture: the processing element (PE) and the switch (SW). Each PE cell processes data coming from two input sources: the first source of data corresponds to the RSA system input signals and the second source of

data comes from partial results forwarded from the outputs of neighboring PE cell. The PE cell modes of operation are controlled by configuration data; these modes of operation are reconfigurable in real time. The SW connects signals between PE cells based on routing configuration data. The SW modes of signal interconnections are also reconfigurable in real time. The block diagram of the RSA architecture is depicted in Fig. 4.1.

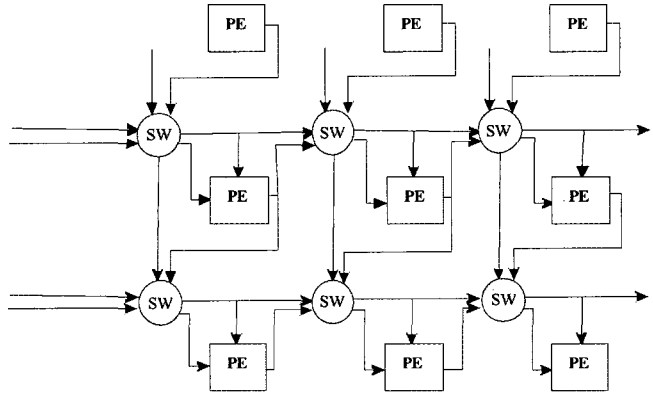


Figure 4.1: Block diagram of the RSA architecture.

4.2.1 The Processing Element

The PE architecture consists of three main building blocks: The arithmetic unit (AU), the control unit (CU) and the memory unit (MU). Each PE cell interfaces to two kinds of input data, the input signals and the configuration data. The signals shifted into the PE cell via eight data buses IN_1 to IN_8 are input data. The signals shifted into the PE cell via four 24 bit wide data buses $IN_1 ... IN_4$ are multiplied by coefficients to generate partial products. These partial products are combined with input signals shifted into the PE cell via four 34 bit wide data buses $IN_5 ... IN_8$ and the result is shifted out of the PE cell. Configuration data is loaded via the CONF bus reserved for loading configuration

data into the PE cell. Configuration data consists of control information that dictates the PE mode of operation and the coefficients to be used by the AU for arithmetic computation. The control information is saved in the CU while the coefficients are stored in the MU. The input data is represented in complex format while the configurations data is represented in real format. Computation results in complex format are shifted out of the PE cell via four 34 bit wide data buses.

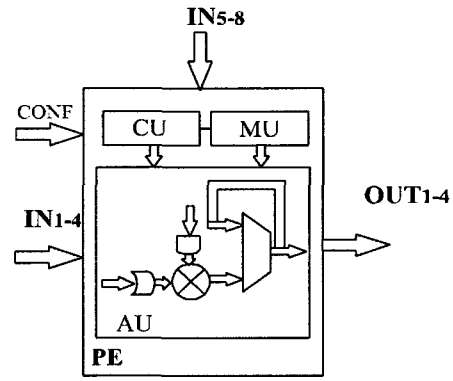


Figure 4.2: Block diagram of the PE architecture.

The MU consists of a block of RAM where coefficients are stored. The coefficients are reconfigurable in real time via the configuration data bus interface. The CU control data flow among the sub-blocks within the AU depends upon the mode of operation selected. A total of nine modes of operation are controlled by the CU. The selected mode of operation is loaded into the CU unit via the configurations data bus where real time configuration is supported.

The AU contains logic resources for arithmetic computation of DSP functions whose mode of operation (MO) is controlled by the CU. The interconnections between the MU, AU and CU are depicted in Fig. 4.3.

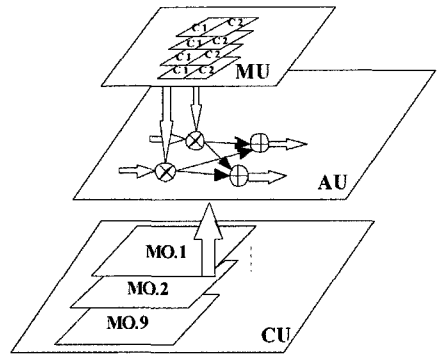


Figure 4.3: Dataflow between the MU, AU and CU building blocks.

4.2.1.1 The Memory Unit

The MU is a storage unit where a total of four sets of coefficients, each of which consists of four B-bit word lengths, are located. The four sets of coefficients allow the AU to perform sequential calculations on up to four different sets of input data. For parallel computation, only one set of coefficients is forwarded to the AU. The coefficients are loaded into the MU via the configuration data bus, which is separate from PE's input data bus. Coefficient reconfiguration could be carried out in real time.

4.2.1.2 The Arithmetic Unit

The AU consists of multiplier and adder units that could be configured to implement arithmetic functions that target MC applications. Figure 4.4 shows a block diagram of the AU architecture:

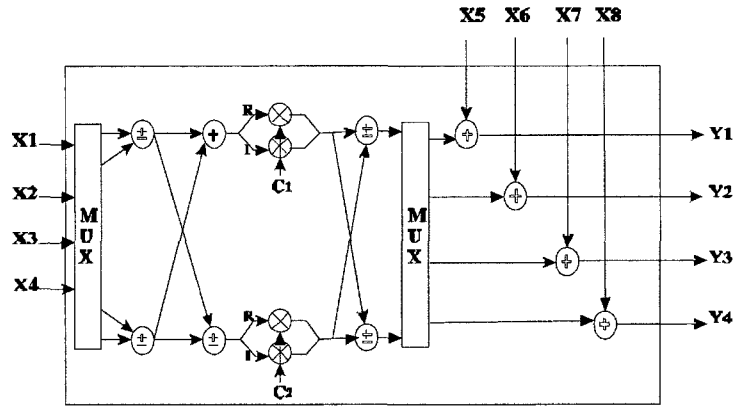


Figure 4.4: Block diagram of the AU architecture.

The AU can be configured to support several modes of operations, including complex multiplication, real multiplication, addition or a combination of these arithmetic operations. The complex multiplication-accumulation configuration is required for DFT/IDFT mapping while complex multiplication and real multiplication-accumulation configurations are needed for phase shifting and FIR filtering computations respectively. The complex multiplication-accumulation configuration is described in Fig. 4.5 while Fig. 4.6 depicts the AU configuration for complex multiplication. Figure 4.7 describes the AU configuration for real multiplication-accumulation.

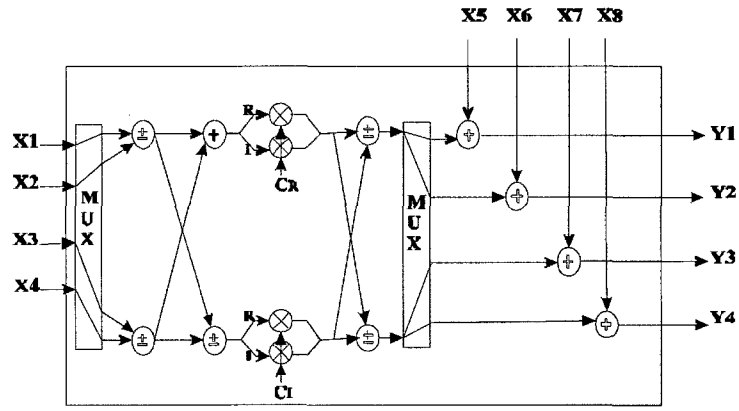


Figure 4.5 : AU configuration for complex multiplication-accumulation.

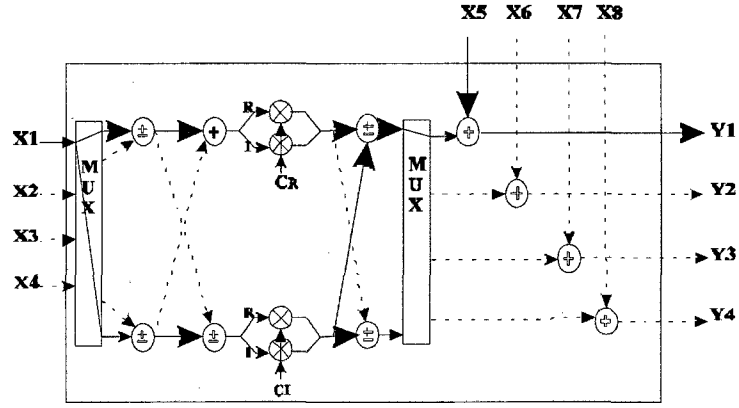


Figure 4.6 : AU configuration for complex multiplication.

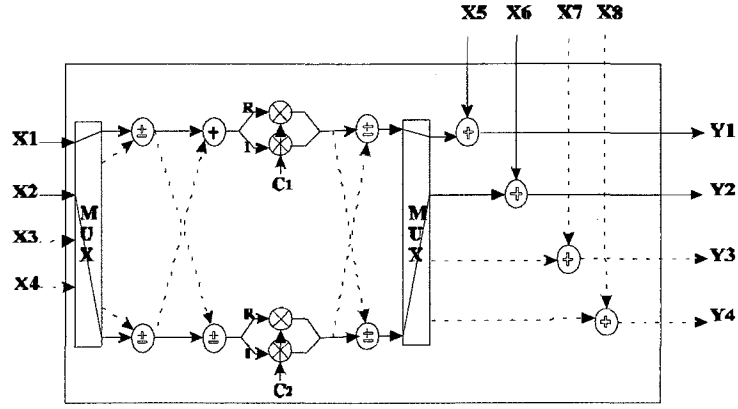


Figure 4.7: AU configuration for real multiplication-accumulation.

4.2.1.3 The Control Unit

The CU consists of a control block that is responsible for the PE's mode of operation. The control block consists of a multiplexer and a RAM block that contains nine modes of operation. The multiplexer selects one among the nine modes of operations based on a 4-bit control bus. The modes of operation dictate the flow of signals in the AU with respect to circuit configurations. The modes of operation are loaded in via the configuration data bus and the 4-bit control data could be reconfigured in real time. Figure 4.8 shows the block diagram of the CU control block.

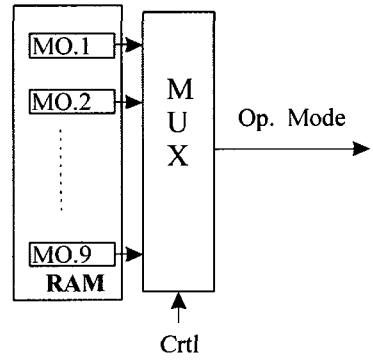


Figure 4.8: Block diagram of the CU control block.

The nine modes of operation supported by the CU are described in Table 4.1.

Table 4.1: Modes of operation supported by the CU.

Control Data	Mode of Operations
0000	MO.1 (Complex Mult-and-Add for DFT with $N = 8n$ $(n: Even)$)
0001	MO.2 (Complex Mult-and-Add for DFT with $N = 8n$ $(n: Odd)$)
0010	MO.3 (Complex Mult-and-Add for DFT with $N = 4n$ $(n: Odd)$)
0011	MO.4 (Complex Mult-and-Add for DFT with $N = 4n+2$)
0100	MO.5 (Complex Mult-and-Add for DFT with $N = 2n+1$)
0101	MO.6 (Complex Mult. for Phase Shifter Mode with two Outputs)
0110	MO.7 (Complex Mult. for Phase Shifter Mode with four Outputs)
0111	MO.8 (Real Mult-and-Add for FIR filter Mode with two Outputs)
1000	MO.9 (Real Mult-and-Add for FIR filter Mode with four Outputs)

4.2.2 The Switch

For the purposes of this section, primary signals are defined as signals that shift into and out of the RSA circuit. Signals that originate at the output of a PE cell and are routed to two or more PE cells are also defined as primary signals. Secondary signals are defined as signals that originate at the output of a PE cell and connect to the adjacent PE cell.

The SW is a unit that routes signals from the input of the RSA circuit to the PE cells. It also routes the output signals of the PE cell to other PE cells in the array or it shifts the computation results generated by the PEs to the output of the RSA circuit. Each SW interfaces with four neighboring SWs and three nearby PE cells. The signals from two neighboring switches on the north and the west sides of an SW are primary signals namely: the input signals to the RSA or the output signals from the adjacent PE cell on the west side of the SW. Signals originating from the two neighboring PE cells on the west and on the north side of the SW are secondary signals. The SW shifts primary signals out to its neighboring SWs on its east side and south sides. It also routes the primary and secondary signals to the PE cell on its east side. The primary signals shifted to neighboring PE cells contain data to be further operated on by the PE's CU. The secondary signals shifted to these PE cells are partial results from the previous stage to be combined with results generated by the multiplication process. Figure 4.9 depicts signal interconnections between an SW and its neighboring SW and PE cells.

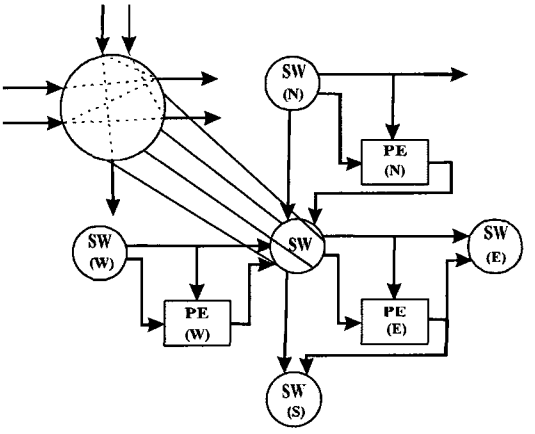


Figure 4.9: Connections between SW and PE cells.

The SW has two types of input interfaces. The first, which consists of sixteen data buses in complex format, supports data that is shifted in from system inputs or data that is

shifted out of the neighboring PE cell. The sixteen data buses are grouped into two sets of input data representing the primary and secondary signals. Each set of primary input data consists of eight, 24 bits data signals of which four are shifted in to the SW from the north side $(PN-IN_{1-4})$ and the other four are shifted in from the west side $(PW-IN_{1-4})$ as depicted in Fig. 4.10.

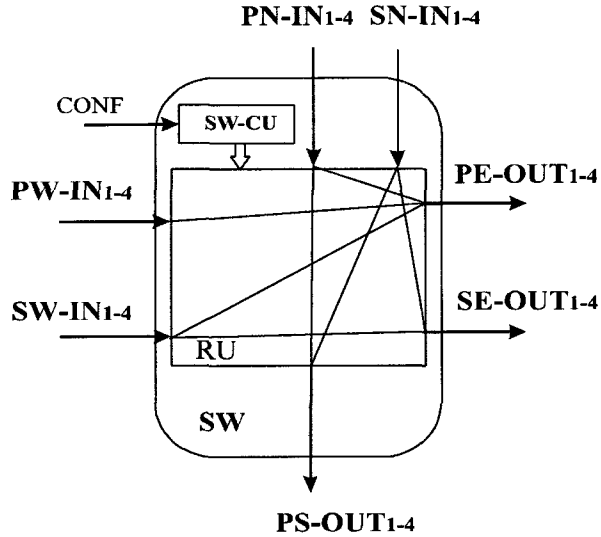


Figure 4.10: Block diagram of the SW.

The secondary input data consists of eight, 34 bit data signals of which four are shifted in to the SW from the west side $(SW-IN_{1-4})$ and the other four are coming from the north side $(SN-IN_{1-4})$ of the SW. The second input interface is dedicated to reconfiguration data that determines the modes of operation for the SW. Reconfiguration data can be loaded in to the switch in real time via the CONF bus. Signal outputs shift out of the SW via twelve data buses represented by two sets of primary and one set of secondary data buses, all in complex format. Each set of the primary buses consists of four data signals that are shifted out to PE cells to the east $(PE-OUT_{1-4})$ and to the south side $(PS-OUT_{1-4})$ of the SW respectively. The set of secondary buses consists of

four data signals that are shifted out to PE cell on the east side $(SE-OUT_{1-4})$ of the SW. Table 4.2 summarizes the characteristics of the input and output data buses of the SW.

Table 4.2: Description of the input and output buses of the SW.

Name		Descriptions
CONF		Configuration data bus (3 bit)
INPUT	$PN-IN_{1-4}$	Primary signals shifted in from SW on the north side (4 buses of 24 bit each in complex format)
	$SN-IN_{1-4}$	Secondary signals shifted in from PE on the north side (4 buses of 34 bit each in complex format)
	$PW-IN_{1-4}$	Primary signals shifted in from SW on the west side (4 buses of 24 bit each in complex format)
	$SW-IN_{1-4}$	Secondary signals shifted in from PE on the west side (4 buses of 24 bit each in complex format)
OUTPUT	$PE-OUT_{1-4}$	Primary signals shifted out to SW on the east side (4 buses of 24 bit each in complex format)
	$SE-OUT_{1-4}$	Secondary signals shifted out to PE on the east side (4 buses of 34 bit each in complex format)
	$PS-OUT_{1-4}$	Primary signals shifted out to SW on the south side (4 buses of 24 bit each in complex format)

The SW consists of two building blocks: the routing unit (RU) and the control unit (SW-CU). The SW-CU contains control data that is used to direct the flow of signals from the inputs to the outputs of the switch. The SW-CU consists of a multiplexer and a RAM block. The RAM block contains eight sets of routing data, each of which represents a specific routing configuration. The multiplexer selects one of the eight routing modes

based on a 3-bit control bus. The control mode is loaded via the configuration data bus in real time if necessary. Figure 4.11 shows the block diagram of the SW-CU building block.

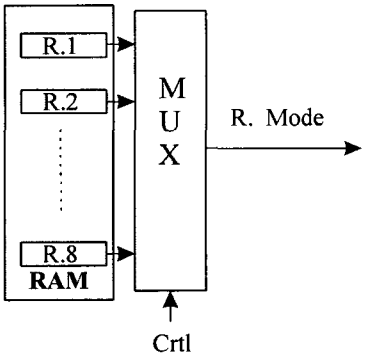


Figure 4.11: Block diagram of the SW-CU.

The RU routes the input signals to the SW's outputs based on the selected mode as shown in Fig. 4.12.

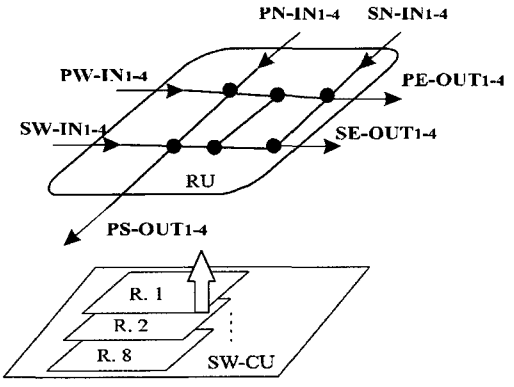


Figure 4.12: Dataflow between the SW-CU and RU building blocks inside the SW.

The SW's eight signal routing modes under RU control are depicted in Fig. 4.13:

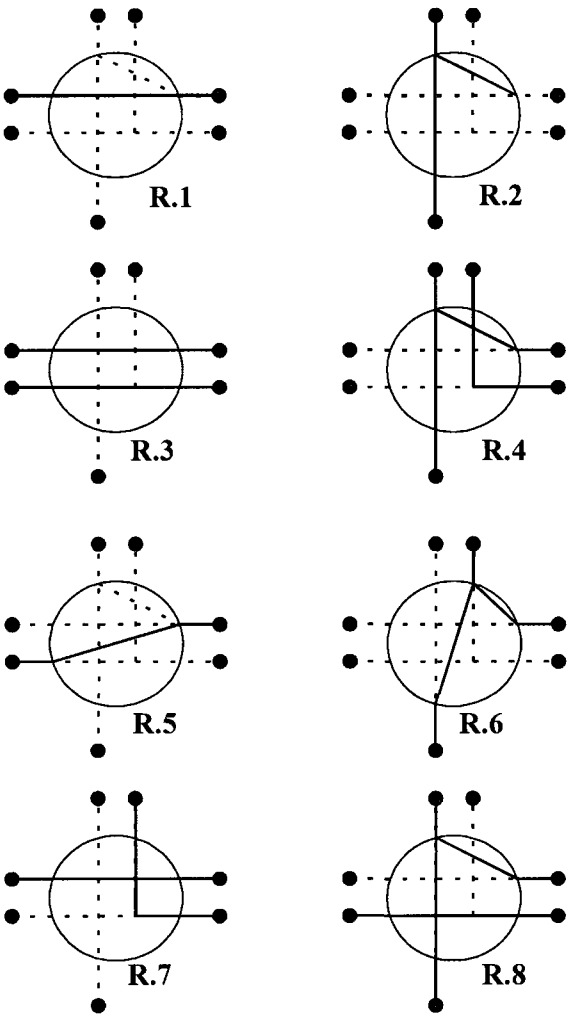


Figure 4.13: Block diagrams of the SW signal routing modes.

The eight modes of operation supported by the SW are summarized in Table 4.3.

Table 4.3: Modes of operation supported by the SW.

Control Data	Mode of Operations	Primary Outputs		Secondary Outputs (SE-OUT)
		PE-OUT	PS-OUT	
000	R.1	PW-IN	-	-
001	R.2	PN-IN	PN-IN	-
010	R.3	PW-IN	-	SW-IN
011	R.4	PN-IN	PN-IN	SN-IN
100	R.5	SW-IN	-	-
101	R.6	SN-IN	SN-IN	-
110	R.7	PW-IN	-	SW-IN
111	R.8	PN-IN	PN-IN	SW-IN

4.3 Expandability of the RSA Architecture

The RSA is a homogeneous and modular architecture that allows circuit expansion to support applications with high computational requirements. Circuit expansion could be achieved by expanding the number of PE cells in the RSA circuit or by cascading RSA circuits. By using M RSA circuits, each of which supports N complex multiplication and addition operations, the expanded system is capable of supporting MN complex multiplication and addition operations as shown in Fig. 4.14.

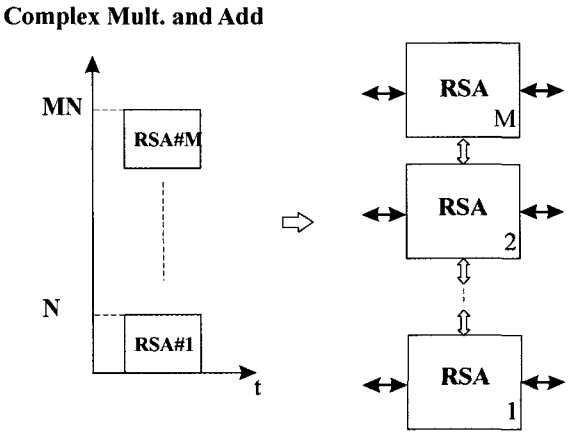


Figure 4.14: Complex multiplication and addition operations using M RSA circuits.

Circuit expansion techniques can be used to extend an RSA circuit to implement a FIR filter with a large number of taps. To realize a T-tap FIR filter configuration which requires NM multiplication and addition operations, a total of M RSA structures each of which supports N multiplication and addition operations can be cascaded together. Figure 4.15 shows the block diagram of a circuit consisting of M RSA structures corresponding to a T-tap FIR filter configuration.

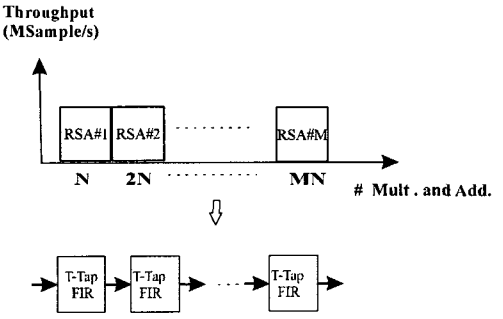


Figure 4.15: FIR filter example of RSA circuit expansion.

4.4 Scalability of the RSA Architecture

The homogeneous and regular characteristics of the RSA architecture also make it suitable for scalable computation. Scalable computation is needed when there is a requirement for greater computation capacity but system hardware expansion is either not practical or desirable.

For low data rate applications that require a large number of computations, relatively small RSA circuits could be used without the need of hardware expansion. Instead of replacing the RSA circuit for one with higher computation capacity, current RSA circuit could be used if the input data can be processed sequentially. If the input data consists of N samples, it could be decomposed into N/K groups of samples where each group is shifted into a K-input RSA circuit for processing. The RSA circuit in this case processes the N samples in a time-shared fashion and all N output data points are available after N/K computations.

The scalability of the RSA architecture could also be employed for applications that require intensive arithmetic operations based on a fixed size circuit. The reconfiguration of a RSA circuit that has been designed as a K-tap FIR filter to support N-tap FIR filtering where $N > K$ is an example of circuit scalability. The RSA circuit in this case designed as a K-tap FIR filter, consists of K real multipliers and K-1 adders. To implement an N-tap FIR filter operation; a total of N real multiplications and N-1 addition operations need to be calculated. The N multiplications and additions can be partitioned into N/K groups of multiplications and additions and each group would be executed in the RSA circuit at a time. The partial result shifted out from each iteration is then looped back to be combined with partial results generated on the next clock cycle. The final output of the N-tap FIR filter is available after $\frac{N}{K}$ clock cycles. For this sequential computation, filter circuit configuration data and the related filter coefficients

have to be updated for every iteration. Figure 4.16 describes the computation process for an N-tap FIR filter using a K-tap FIR filter circuit based on the time-sharing technique.

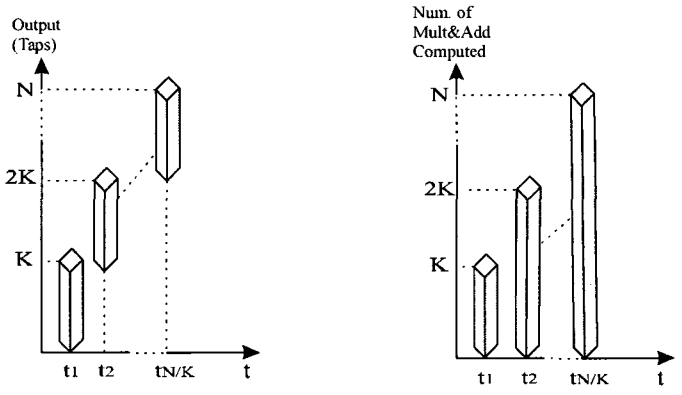


Figure 4.16: Time-sharing computation of N-tap FIR filter based on an RSA circuit designed as a K-tap FIR filter.

4.5 Design Process Flow

In this section, the research and design process associated with the conceptualization, modeling, verification and refinement of the RSA circuit for multicarrier wireless and multirate applications are briefly outlined. The first step of this process consisted of generating Matlab and SystemView models for Polyphase/FIR filters, DFT/IDFT, Phase Shifters, IDFT-Polyphase and DFT-Polyphase circuits. DFT/IDFT circuit models designed and simulated using Matlab were based on design approaches proposed for low computation complexity and included designs for even and odd length input sequences (Appendix A). These models were developed to generate the required input and output vectors for functional testing and verification of the RSA design and circuit configurations for the computation of the above mentioned functions.

Multiplier circuit designs based on a combination of CSE and bit slice techniques, CSA technique and Radix-4 Booth were designed using VHDL. Results extracted from simulations of these multiplier circuits were compared to one another to determine the parameters to be used in the design of the AU in PE cell.

The RSA VHDL circuit design was based on the proposed systolic array architecture. The multipliers in the PE cells were designed based on the CSE-BitSlice arithmetic following the previously established models. Mapping of the different circuits was carried out on the RSA structure. For the DFT/IDFT functions, a mapping approach to achieve low computational complexity was adopted (Appendix C). Functional testing of the targeted DSP functions mapped onto the RSA circuits was performed using test vectors generated by the Matlab and SystemView models. The RSA architecture, as well as the PE and SW building blocks were modified and refined at each stage of this iterative process.

The final step of the design and implementation process was FPGA implementation of the RSA and related circuits. Hardware requirements and throughput performance simulations of the RSA circuits for different configurations were carried out. Simulation results obtained from the mapping of DFT and FIR filter functions were used to compare with the performance of existing reconfigurable architectures. Two representative multipliers have also been designed on ASIC based on CSE-BitSlice, Radix-4 Booth and CSA techniques using Mentor Graphic's SCL05μ standard cell library where circuit gate counts and clock frequency are used to compare to simulation results of FPGA implementations of similar circuits (Appendix B). Figure 4.17 in the form of a flow chart depicts the design process for the RSA circuit and identifies the relevant chapters and research activities.

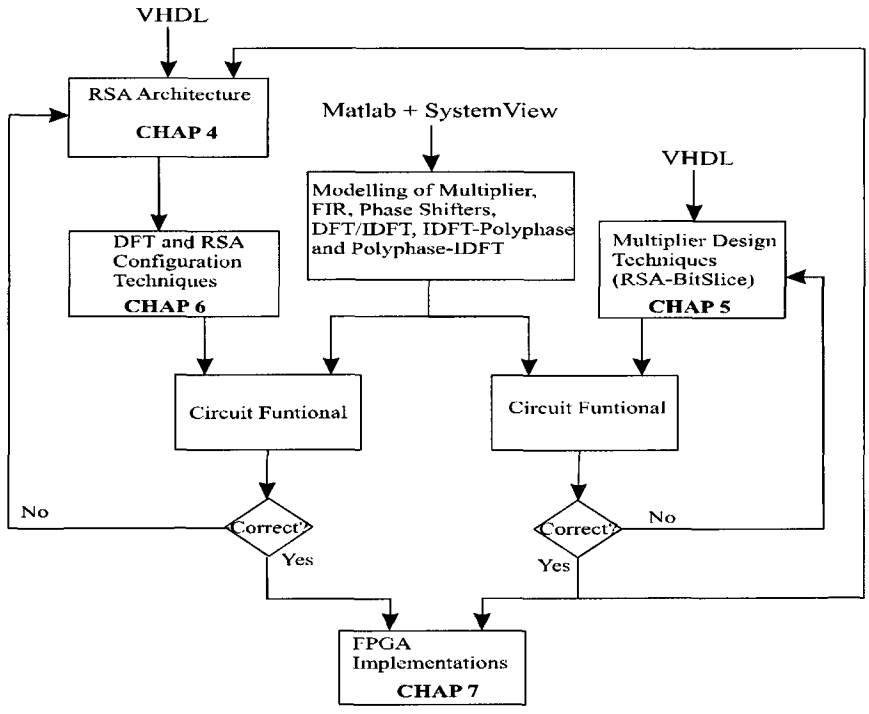


Figure 4.17: Flow chart of the research and design process of the RSA circuit and thesis organization.

Chapter 5

Low Complexity RSA Circuit Design

5.1 Introduction

This chapter discusses the design techniques employed to implement an RSA circuit and its building blocks. In the first section, a design technique for multiplier circuits using a combination of CSE and Bit-Slice arithmetic approach is introduced. Designs of real, vector and matrix multiplier circuits based on the CSE-BitSlice method are described and the number of addition operations are determined and compared with similar circuit designs based on the CSA and Radix-4 Booth algorithms.

The second section describes the design of the PE circuit and its building blocks using the CSE-BitSlice technique. The total number of adders required by the PE cell design using the CSE-BitSlice approach is then considered and compared with a PE cell circuit design based on the CSA and Radix-4 Booth techniques.

5.2 CSE and Bit-Slice Arithmetic for Multiplier Circuit Designs

In this section, multiplier circuit designs based on the CSE-BitSlice technique are presented. Hardware complexity estimates based on the total number of additions required for the design of real, vector, and matrix multipliers are then presented. The results are compared with multiplier circuit designs based on the CSA and Radix-4 Booth algorithms.

5.2.1. Bit-Slice Arithmetic

In bit-slice arithmetic, a binary represented value can be decomposed into a sum of multiple terms as depicted in the example that follows:

$$F = 101110001011110 = 101 * 2^{12} + 110 * 2^{9} + 001 * 2^{6} + 011 * 2^{3} + 110 \quad (5.1)$$

The equation above is just one way of partitioning F which consists of a large number of digits into several bit-slice components with shorter word lengths. Using this decomposition technique, a value with a long word length can be decomposed into several bit-slice components with smaller word lengths and arithmetic operations are performed on these bit-slice components separately [Smit'95]. Partial results generated from operations on the decomposed bit-slice components are then combined to produce the final output. The advantage of bit-slice decomposition is that arithmetic operations involving long word length variables do not need to be performed in a single shot, which would require a lot of logic resources for circuit implementation. Instead, the bit-slice components could be processed by a small circuit in a time shared fashion where each bit-slice is processed sequentially. This technique could be used to process data signals with large dynamic range where the circuit has limited logic resources or has a limited number of I/Os. A time sharing approach based on a 3-by-3 bit multiplier circuit for the computation of the product of two variables where the multiplicand and the multiplier operands consist of 15 bit and 3 bit word lengths respectively is illustrated by the example below:

$F = x_1 * x_2$
$x_1 = 101110001011110$
$x_2 = 100$

$$F = (101 * x_2) * 2^{12} + (110 * x_2) * 2^9 + (001 * x_2) * 2^6 + (011 * x_2) * 2^3 + (110 * x_2)$$
$$F = (e * x_2) * 2^{12} + (d * x_2) * 2^9 + (c * x_2) * 2^6 + (b * x_2) * 2^3 + (a * x_2)$$

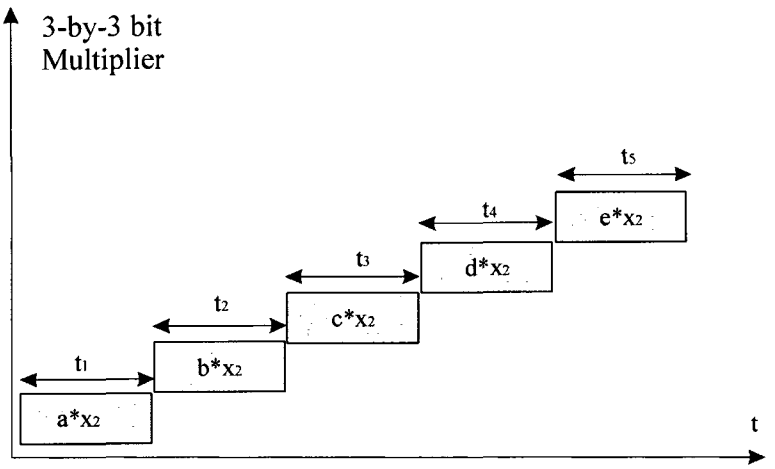


Figure 5.1: Time-sharing technique for multiplication of a 15-bit multiplicand by a 3-bit multiplier operand using a 3-by-3 bit multiplier.

As shown in Fig. 5.1, a bit-slice technique is used on signal x_1 whose 15-bit word length is decomposed into five smaller signals; each of which consists of three digits. The decomposed signals are then shifted into a 3-by-3 bit multiplier to be sequentially multiplied by x_2. The five partial results shifted out of the multiplier are then recombined to produce the final result.

5.2.2 CSE for Multiplier Design

Consider the sum F of the product of variables x_1 and x_2 and two constants as follows:

$$F = x_1 * 011100_2 + x_2 * 011101_2 \tag{5.2}$$

If this multiplication-addition expression is represented by shift and add operations, the end result is a structure that requires six shifts and six additions as shown in (5.3). Note that the "$<<n$" sign represents an n-step left shift.

$$F=(x_1<<4)+(x_1<<3)+(x_1<<2)+(x_2<<4)+(x_2<<3)+(x_2<<2)+x_2 \tag{5.3}$$

Now if the common sub-expression $d = 011100_2$ in (5.2) is factored out, then the multiplication and addition operations require only three shifts and four additions. The extraction of common bit patterns in the sum of two products expression in (5.2) is depicted in (5.4).

$$F = d + x_2 \tag{5.4}$$

where

$$d=(x_1+x_2)<<4+(x_1+x_2)<<3+(x_1+x_2)<<2$$

Comparing the approaches presented in (5.3) and (5.4) we note that by factoring out the common digits from the two multiplier operands, the number of shifts and additions for the multiplication shown in (5.2) has been reduced significantly. Fig. 5.2a and Fig. 5.2b illustrate the number of adders and shifters required by the multiplication expressions shown in (5.3) and (5.4) respectively.

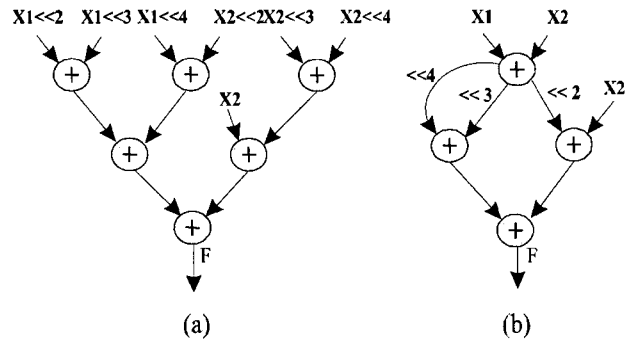


Figure 5.2: Multiplier circuit implementation based on the CSE technique.

Thus by identifying and factoring out common digits present in the constant operands, the common sub-expressions needs to be computed only once, which helps to reduce the logic resource requirements for multiplier circuit implementation.

5.2.3 CSE-BitSlice for Multiplier Circuit Implementations

For the purpose of this section, calculation of the number of additions needed to compute real, vector, and matrix multiplications is based on the assumption that the multiplier operand is a positive integer represented in two's complement format. For circuit implementations, both operands of the multiplier are negated before multiplying if the multiplier operand is negative. Moreover, for real number multiplier design based on the CSE-BitSlice technique discussed in this section, it is assumed that the word length B of the multiplier operand is a positive integer that is divisible by N. The variable N represents the number of bit slices with equal length obtained from the partition of the multiplier operand. If B is not divisible by N, an extension of the most significant bit (MSB) is needed to bring total digits of the multiplier operand to the nearest integer with N as one of its factor.

Multiplication of two variables x_1 and x_2 where the operand x_2 with a word length B is decomposed into upper and lower bits can be described as follows:

$$F = x_1 * x_2 = x_1 * \left(x_{2,u} * 2^{B/2} + x_{2,l} \right)$$
$$F = x_1 * x_{2,u} * 2^{B/2} + x_1 * x_{2,l} \tag{5.5}$$

where

$$x_i = \left[x_{i,B-1}, x_{i,B-2}, \ldots, x_{i,0} \right] = x_{i,B-1} 2^{B-1} + \sum_{n=0}^{B-2} x_{i,n} 2^n$$

and

$$x_{i,B-1} \in \{-1, 0\} \ ; \ x_{i,n} \in \{0, 1\} \ ; \ i \in \{1, 2\}$$

In (5.5), $x_{2,u}$ and $x_{2,l}$ are the upper and lower $B/2$ bits of x_2 respectively. If the two variables $x_{2,u}$ and $x_{2,l}$ have several common bits such that:

$$x_{2,ul}^{and} = x_{2,u} \bullet x_{2,l}$$

and $x_{2,u}^{xor}$ and $x_{2,l}^{xor}$ are two variables which represent disjointed digits between $x_{2,u}$ and $x_{2,l}$.

$$x_{2,u}^{xor} = x_{2,u} \bullet \overline{x_{2,l}}$$
$$x_{2,l}^{xor} = \overline{x_{2,u}} \bullet x_{2,l}$$

In the above equations, the symbol $\bullet$ represents the logical AND operator. The common digit elimination process applied to $x_{2,u}$ and $x_{2,l}$ transforms the equation in (5.5) as follows:

$$F = x_1 \ll 2^{B/2} * x_{2,u}^{xor} + \left(x_1 \ll 2^{B/2} + x_1 \right) * x_{2,ul}^{and} + x_1 * x_{2,l}^{xor} \tag{5.6}$$

From the decomposition expression shown in(5.6), the maximum number of additions needed to compute the output F is $B/2$. Without any optimization, direct computation of the multiplication of x_1 and x_2 would need a total of B-1 additions.

Now, if x_2 is decomposed into three slices, each of which consists of $B/3$ bits, the output F becomes:

$$\begin{aligned} F &= x_1 * x_2 = x_1 * \left(x_{2,3} * 2^{2B/3} + x_{2,2} * 2^{B/3} + x_{2,1} \right) \\ F &= \left(x_1 2^{2B/3} * x_{2,3} \right) + \left(x_1 2^{B/3} * x_{2,2} \right) + \left(x_1 * x_{2,1} \right) \end{aligned} \tag{5.7}$$

In the equation above, $x_{2,3}, x_{2,2}$, and $x_{2,1}$ are the upper $B/3$, the middle $B/3$ bits, and the lower $B/3$ bits of x_2 respectively. If the three variables $x_{2,3}, x_{2,2}$, and $x_{2,1}$ have several common bits such that:

$$x_{2,123}^{and} = x_{2,3} \bullet x_{2,2} \bullet x_{2,1}$$

and the pair wise jointed and disjointed sets of digits for $x_{2,3}, x_{2,2}$, and $x_{2,1}$ are:

$$x_{2,12}^{and} = x_{2,1} \bullet x_{2,2} \bullet \overline{x_{2,3}}$$

$$x_{2,13}^{and} = x_{2,1} \bullet x_{2,3} \bullet \overline{x_{2,2}}$$

$$x_{2,23}^{and} = x_{2,2} \bullet x_{2,3} \bullet \overline{x_{2,1}}$$

$$x_{2,3}^{xor} = x_{2,3} \bullet \overline{x_{2,2}} \bullet \overline{x_{2,1}}$$

$$x_{2,2}^{xor} = x_{2,2} \bullet \overline{x_{2,3}} \bullet \overline{x_{2,1}}$$

$$x_{2,1}^{xor} = x_{2,1} \bullet \overline{x_{2,2}} \bullet \overline{x_{2,3}}$$

The common digit elimination process applied to $x_{2,3}, x_{2,2}$, and $x_{2,1}$ transforms the equation in (5.5) as follows:

$$F = \left(x_1 \ll 2^{2B/3} + x_1 \ll 2^{B/3} + x_1\right) * x_{2,123}^{and} + \left(x_1 \ll 2^{B/3} + x_1\right) * x_{2,12}^{and} + \left(x_1 \ll 2^{2B/3} + x_1\right) * x_{2,13}^{and} + \left(x_1 \ll 2^{2B/3} + x_1 \ll 2^{B/3}\right) * x_{2,23}^{and} + \left(x_1 * x_{2,1}^{xor}\right) + \left(x_1 \ll 2^{B/3} * x_{2,2}^{xor}\right) + \left(x_1 \ll 2^{2B/3} * x_{2,3}^{xor}\right)$$

From the decomposition expression shown in the above equation, the maximum number of additions needed to compute the output F is $\left(\frac{B}{3}+3\right)$. Now, if x_2 is decomposed into four slices, each of which consists of $B/4$ bits, using a similar common digits elimination process the total number of additions required for the multiplication of x_1 and x_2 is $\left(\frac{B}{4}+10\right)$. Thus, the total number of additions $T_r(n,B)$ calculated based on the decomposition and common digit elimination technique for multiplication of two variables can be expressed in general terms as follows:

$$T_r(n,B) = \frac{B}{n} + 2^n - n - 2 \qquad \forall n \in \mathbb{N} \tag{5.8}$$

In (5.8), the variables n and B represent the number of terms decomposed and the word length of the multiplier operand x_2 respectively

Proof: (By Induction.)

True for n = 1:

$$T_r(1,B) = \frac{B}{1} + 2^1 - 1 - 2 = B - 1 \tag{5.9}$$

For n=2, the total number of additions needed to combine all partial results for the sum of products expression generated from the decomposition and common digit elimination of the multiplier operand x_2 is $\frac{B}{2}-1$. An extra addition is needed to obtain the sum of the multiplicand operand x_1 and $x_1 \ll 2^{B/2}$ as required by the decomposition and common digit elimination process. Thus, the total number of additions required for n=2 is:

$$T_r(2,B) = \frac{B}{2} + 2^2 - 2 - 2 = \left(\frac{B}{2} - 1\right) + 1 \tag{5.10}$$

Similarly, for n=3 and n=4, the total number of additions needed to combine all partial results for the sum of products expression corresponding to n=3 and n=4 is $\frac{B}{3}-1$ and $\frac{B}{4}-1$ respectively. The decomposition and common digit elimination process requires an extra four and eleven additions to generate four partial sums from the three terms $x_1, x_1 \ll 2^{B/3}$, and $x_1 \ll 2^{2B/3}$ and similarly eleven partial sums from $x_1, x_1 \ll 2^{B/4}, x_1 \ll 2^{2B/4}$ and $x_1 \ll 2^{3B/4}$ for n=3 and n=4 respectively. The total number of additions required for n=3 and n=4 are:

$$\begin{aligned} T_r(3,B) &= \frac{B}{3}+2^3-3-2=\left(\frac{B}{3}-1\right)+1+3=\left(\frac{B}{3}-1\right)+4 \\ T_r(4,B) &= \frac{B}{4}+2^4-4-2=\left(\frac{B}{4}-1\right)+4+7=\left(\frac{B}{3}-1\right)+11 \end{aligned} \tag{5.11}$$

From (5.9), (5.10) and (5.11) it can be seen that $T_r(n+1,B)$ can be defined as a recursive equation as follows:

$$T_r(n+1,B)=\left(\frac{B}{n+1}-1\right)+\underbrace{\left(2^n-n-1\right)}_{T_r(n,B)-\left(\frac{B}{n}-1\right)}+\in$$

where $\in=\left(2^n-1\right)$. Thus,

$$T_r(n+1,B)=\left(\frac{B}{n+1}-1\right)+T_r(n,B)-\left(\frac{B}{n}-1\right)+\left(2^n-1\right)$$

Then upon simplifying, we obtain:

$$T_r(n+1,B)=T_r(n,B)-\frac{B}{n(n+1)}+\left(2^n-1\right)$$

Assume that (5.8) is true for n=k: $T_r(k,B)=\frac{B}{k}+2^k-k-2=\left(\frac{B}{k}-1\right)+\left(2^k-k-1\right)$

Given this assumption, we will show that the statement is true for n=k + 1, since:

$$T_r(k+1,B)=T_r(k,B)-\frac{B}{k(k+1)}+\left(2^k-1\right)$$

$$T_r(k+1,B)=\underbrace{\left(\left(\frac{B}{k}-1\right)+\left(2^k-k-1\right)\right)}_{T_r(k,B)}-\frac{B}{k(k+1)}+\left(2^k-1\right)$$
$$=\left(\frac{B}{k+1}-1\right)+2^{k+1}-(k+1)-1 \quad QED$$

Therefore, by induction, the statement is true for all $n\geq 1$.

It can be seen that for larger word length B, decomposition of x_2 into multiple terms would lead to further reduction of the number of addition operations.

To determine conditions for which decomposition into 2, 3, or 4 segments would be preferable we need to consider the following two inequalities:

$$T_r(3,B)\leq T_r(2,B)$$

$$T_r(4,B)\leq T_r(3,B)$$

From the first equation it follows that for $B>18$, a 3 terms decomposition results in fewer additions than a 2 term decomposition and from the second equation we see that for $B>84$, a 4 term decomposition requires fewer additions that a 3 term decomposition. Similarly from the inequality:

$$T_r(N,B)\leq T_r(N-1,B) \tag{5.12}$$

It follows that a multiplication based on an N segment decomposition will require fewer additions than one based on N-1 decompositions provided that:

$$B>\left(2^N-2^{N-1}-1\right)\left(N^2-N\right) \tag{5.13}$$

Table 5.1 depicts the number of decomposition terms needed with respect to the word length B in order to achieve low complexity circuit design for the multiplication of two variables x_1 and x_2. The second column shows the range of the word length B that

would give lowest number of additions based on the given number of decomposition terms.

Table 5.1: Number of decomposed terms vs. word length of multiplier operand.

No. of Terms (n)	**Word Length (B)**	**No. of Additions (T_r)**
2	$B \leq 18$	$\frac{B}{2}$
3	$18 < B \leq 84$	$\frac{B}{3}+3$
N	$B > \left(2^N - 2^{N-1} - 1\right)\left(N^2 - N\right)$	$\frac{B}{N}+\left(2^N - N - 2\right)$

5.2.3.1 Real multiplier design

The CSE-BitSlice technique shown in the previous paragraphs can be applied to the design of the real multiplier circuit. Depending upon the word length of one of the operands; the multiplication is decomposed into N sum of products where the common digits among the decomposed expressions are then factored out. For two term decomposition, the real multiplication of two variables x_1 and x_2 are expressed as follows:

$$F = x_1 * x_2 = x_1 * \left(x_{2,u} * 2^{B/2} + x_{2,l} \right)$$

$$F = x_1 * x_{2,u} * 2^{B/2} + x_1 * x_{2,l}$$

$$F = x_{1,s} * x_{2,u} + x_1 * x_{2,l}$$

$$F = x_{1,s} * \left(x_{2,u} \bullet \overline{x_{2,l}} \right) + \left(x_{1,s} + x_1 \right) * \left(x_{2,u} \bullet x_{2,l} \right) + x_1 * \left(\overline{x_{2,u}} \bullet x_{2,l} \right)$$

Where $x_{1,s} = x_1 \ll 2^{B/2}$

Figure 5.3 shows the block diagram of the real multiplier design based on the two terms decomposition described above:

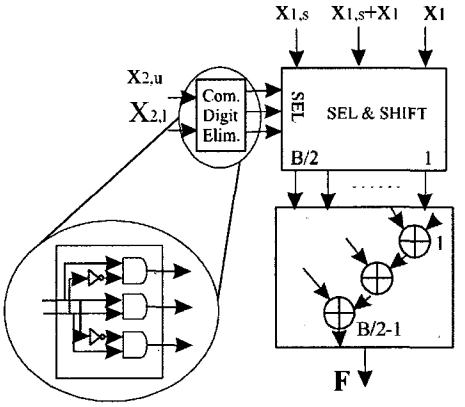


Figure 5.3: Block diagram of real multiplier circuit design based on two term decomposition.

It can be seen from Table 5.2 that there is almost a 50 per cent saving in the number of additions for the two term decomposition real multiplier design as compared to the CSA-based design. Compared to the Radix-4 Booth design, the reduction in the number of additions achieved is insignificant if B has a small number of digits. For B with larger word lengths, the difference in the total number of additions required by the Radix-4

Booth and the CSE-BitSlice designs increases. For example, if the word length B is 84, the number of additions in a three term CSE-BitSlice design is 31 as compared to 41 additions for the Radix-4 Booth design.

Table 5.2: Comparison of CSE-BitSlice based real multiplier and other design techniques.

Optimization Technique $F = x_1 * x_2$		**Number of Additions**
CSA		$B-1$
Radix-4 Booth		$(B/2)-1$
CSE-BitSlice	2-Term	$(B/2)$
	3-Term	$(B/3)+3$

Figure 5.4 illustrates the number of additions versus the word length B for the multiplication of two scalars based on 2-Term and 3-Term CSE-BitSlice and the CSA and Radix-4 Booth techniques.

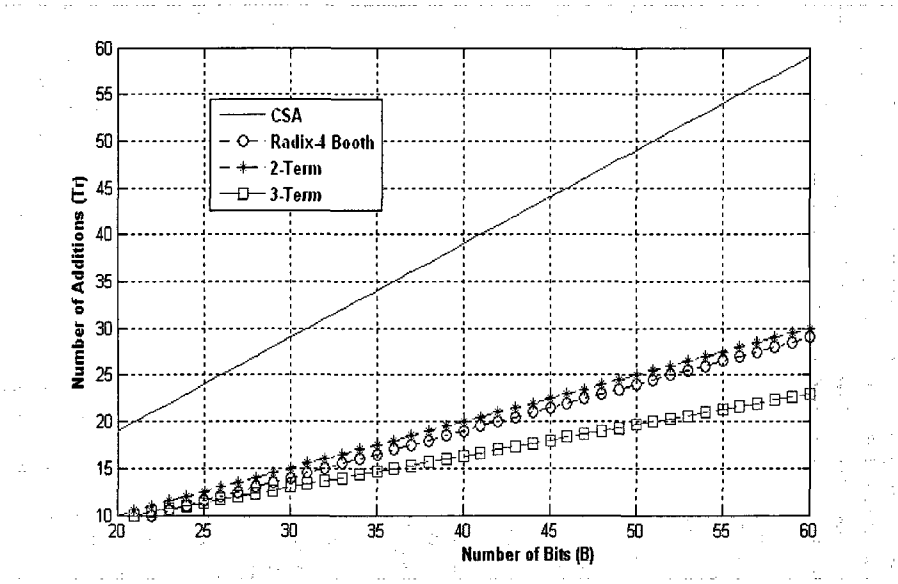


Figure 5.4: Number of additions versus word length B for a real multiplier.

5.2.3.2 Vector multiplier design

The multiplication of two vectors, each of which consists of two variables can be expressed as follows:

$$S = \begin{bmatrix} a_1 & a_2 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = a_1 x_1 + a_2 x_2 \tag{5.14}$$

It can be seen that the 2-Term optimization technique presented in the previous sections could be applied to this vector multiplication. For this multiplication, the vector containing the variables x_1 and x_2 represents the multiplier operands and the vector containing the variables a_1 and a_2 represents the multiplicand operands.

If the two variables x_1 and x_2 have several common bits such that:

$$x_{12}^{and} = x_1 \bullet x_2$$

and x_1^{xor} and x_2^{xor} are two variables representing the disjointed digits of x_1 and x_2 :

$$x_1^{xor} = x_1 \bullet \overline{x_2}$$

$$x_2^{xor} = x_2 \bullet \overline{x_1}$$

then (5.14) could be rewritten as follows:

$$S = a_1 x_1^{xor} + (a_1 + a_2) x_{12}^{and} + a_2 x_2^{xor} \tag{5.15}$$

If the multiplier operands x_1 and x_2 have the same word length B, it can easily be seen that the total number of additions required to produce the sum of two products $a_1 x_1$ and $a_2 x_2$ is equal to B. Without optimization, the total number of shifts and additions required to compute the sum of two real multiplications where the two multiplier operands have the same word length of B is $2B-1$. Figure 5.5 shows the block diagram of the dot product for the 2 variable vector multiplication design based on the 2-Term decomposition approach.

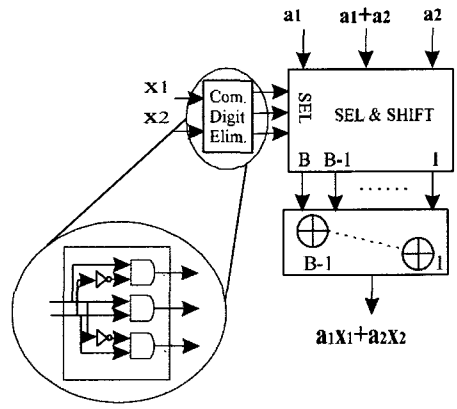


Figure 5.5: Block diagram of the dot product for the 2 variable vector multiplier design based on the 2-Term decomposition approach.

Now if the number of variables in the multiplier and multiplicand vectors is increased to three such that:

$$S=\begin{bmatrix}a_1 & a_2 & a_3\end{bmatrix}*\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=a_1x_1+a_2x_2+a_3x_3 \tag{5.16}$$

and if x_1, x_2, and x_3 have the same word length of B then the three term decomposition approach could be applied to the dot product of the two vectors shown in (5.16). It can be easily seen that the total number of additions required to produce the sum of three products a_1x_1, a_2x_2, and a_3x_3 is equal to $B+3$. Without optimization, the total number of additions required by this three variables vector multiplication is $3B-1$. Figure 5.6 shows the block diagram of the dot product for the 3 variable vector multiplication based on the 3-Term decomposition approach.

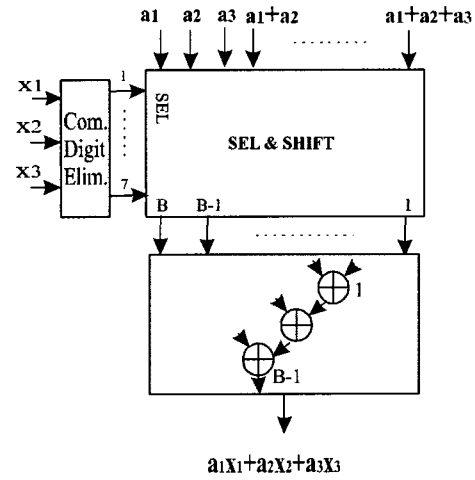


Figure 5.6: Block diagram of the dot product for the 3 variables vector multiplier design using the 3-Term decomposition approach.

For multiplication of two vectors consisting of m variables where m is larger than three, hardware optimization could be carried out based on a multiple term decomposition technique discussed in the previous paragraphs. The total number of

additions for m variable vector multiplication $T_v(m,B)$ can be determined based on (5.8) and is expressed in the equation below:

$$T_v(m,B) = B + (2^m - m - 2) \qquad m \geq 2 \qquad (5.17)$$

Proof: (By Induction.)

For m=2, the total number of additions needed to combine all partial results for the sum of products expression generated from the common digit elimination of the multiplier operand x_1 and x_2 is $B-1$. An extra addition is needed to obtain the sum of the multiplicand operand a_1 and a_2 as required by the decomposition and common digit elimination process. Thus, the total number of additions required for m=2 is:

$$T_v(2,B) = B + 2^2 - 2 - 2 = (B-1) + 1$$

For m=3, the total number of additions needed to combine all partial results for the sum of products expression generated from the common digit elimination of the multiplier operands x_1, x_2 and x_3 is also equal to $B-1$. The decomposition and common digit elimination process requires an extra four additions to generate four partial sums from the three terms a_1, a_2, and a_3. Thus, the total number of additions required for m=3 is:

$$T_v(3,B) = B + 2^3 - 3 - 2 = (B-1) + 4 \qquad (5.18)$$

Similarly, for m=4, the total number of additions needed to combine all partial results for the sum of products expression generated from the common digit elimination of the multiplier operands x_1, x_2, x_3, and x_4 is also equal to $B-1$. The decomposition and common digit elimination process requires an extra eleven additions to generate eleven partial sums from a_1, a_2, a_3, and a_4 for m=4. The total number of additions required for m=4 is:

$$T_v(4,B) = B + 2^4 - 4 - 2 = (B-1) + 4 + 7 = (B-1) + 11 \qquad (5.19)$$

From (5.17), (5.18), and (5.19) it can be seen that $T_v(m+1,B)$ can be defined as a recursive equation as follows:

$$T_v(m+1,B) = (B-1) + \underbrace{\left(2^m - m - 1\right)}_{T_v(m,B)-(B-1)} + \in$$

where $\in = \left(2^m - 1\right)$. Thus,

$$T_v(m+1,B) = (B-1) + T_v(m,B) - (B-1) + \left(2^m - 1\right)$$

Then upon simplifying, we obtain:

$$T_v(m+1,B) = T_v(m,B) + \left(2^m - 1\right)$$

Assume (5.17) is true for m=k: $T_v(k,B) = B + 2^k - k - 2$

Given this assumption, we will show that the statement is true for m=k + 1, since:

$$\begin{aligned} T_v(k+1,B) &= T_v(k,B) + \in \\ &= T_v(k,B) + \left(2^k - 1\right) \end{aligned}$$

$$\begin{aligned} T_v(k+1,B) &= \underbrace{\left(B + 2^k - k - 2\right)}_{T_v(k,B)} + \left(2^k - 1\right) \\ &= B + 2^{k+1} - (k+1) - 2 \quad QED \end{aligned}$$

Therefore, by induction, the statement is true for all $m \geq 2$.

Thus, optimization for the M-variable vector multiplication, where each element in the multiplier vector consists of an B-bit word length variable, could be carried out based on the optimization of multiplication of two variables of which the multiplier operand consists of an MB-bit word that has been decomposed into the sum of M products.

Table 5.3 provides calculation results for the number of additions required by M variable vector multiplication optimized using m-Term optimization techniques where $m = 2$ and $m = 3$. The 2-Term and 3-Term expressions shown in Table 5.3 reflect decomposition results where the number of variables M is a multiple of 2 and 3 respectively.

Table 5.3: Comparison of the number of additions for M variable vector multiplication.

No. of Variables	**No. of Additions (2-Term, M=2)**	**No. of Additions (3-Term, M=3)**
M	B	$B+3$

On the other hand, the sum of M products could also be partitioned into groups with smaller sum values where the CSE-BitSlice technique would be applied independently to each group. After the optimization process is finished, the final output is generated by summing the partial results together. For example, for the case of two vectors where each vector consists of four variables, the sum of the four products could be partitioned into two groups, each consisting of the sum of two products. The optimization applied to each group would result in a circuit of B adders. Thus, the total number of additions required to compute the product of two 4 element vectors is $2B+1$. If the vector multiplication design is optimized and the number of additions corresponds to (5.17) where M is equal to 4, a total of $B+10$ addition operations are required. If the word length of each variable in the multiplier vector is smaller than 9 bits then using 2-Term optimization would result in a more compact design.

Figure 5.7 shows the block diagram of the multiplication of two vectors, each of which consists of M variables. The sum of M products in this case has been partitioned into the sum of $M/2$ expressions:

$$S=[a_1 \ \ldots \ a_M]*\begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix}=S_1+S_2+\ldots+S_{\frac{M}{2}} \tag{5.20}$$

Where

$$S_1 = a_1x_1 + a_2x_2$$
$$S_2 = a_3x_3 + a_4x_4$$
$$\vdots$$
$$S_{\frac{M}{2}} = a_{\frac{M}{2}-1}x_{\frac{M}{2}-1} + a_{\frac{M}{2}}x_{\frac{M}{2}}$$

Each expression S_i is a partial result generated from the computation of the sum of two products. The output of this vector multiplication is determined by summing up all $M/2$ partial results.

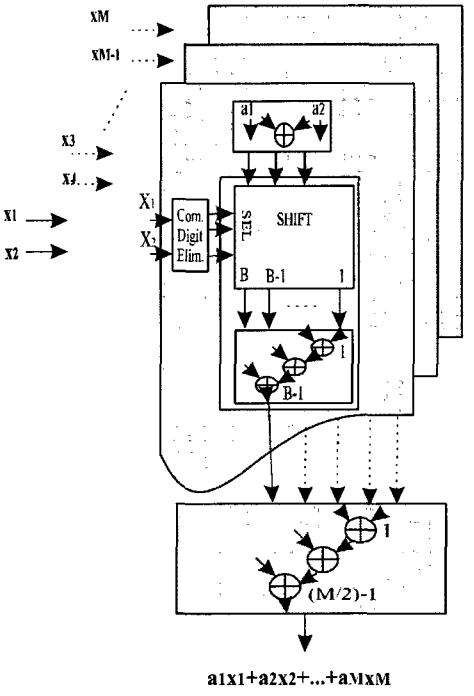


Figure 5.7: Block diagram of the dot product for the M variable vector multiplier design based on the 2-Term decomposition approach.

The total number of additions required to compute a vector multiplication involving M variables based on the 2-Term decomposition approach is $(M/2)(B+1)-1$. As

a result, this optimization technique offers an overall hardware saving of up to 50 per cent as compared to an implementation based on the CSA algorithm which requires a total of $MB-1$ additions as shown in Table 5.4. Also shown in Table 5.4, if the optimization involved three or more terms, the number of additions required using CSE-BitSlice technique would be a lot lower compared to the Radix-4 Booth technique.

Table 5.4: Comparison of CSE-BitSlice based M variable vector multiplier against other design techniques.

Optimization Technique $S=\begin{bmatrix}a_1 & \cdots & a_M\end{bmatrix}^*\begin{bmatrix}x_1\\ \vdots \\ x_M\end{bmatrix}$		**Number of Additions**
CSA		$MB-1$
Radix-4 Booth		$\frac{MB}{2}-1$
CSE-BitSlice	2-Term	$\frac{M}{2}(B+1)-1$
	3-Term	$\frac{M}{3}(B+4)-1$

Figure 5.8 illustrates the number of additions versus word length B for M variable vector multiplication required by the 2-Term and 3-Term CSE-BitSlice, as well as, the CSA and Radix-4 Booth techniques. The three graphs shown in Fig. 5.8 have been generated for vector multiplications involving 6, 9, and 12 variable vectors respectively.

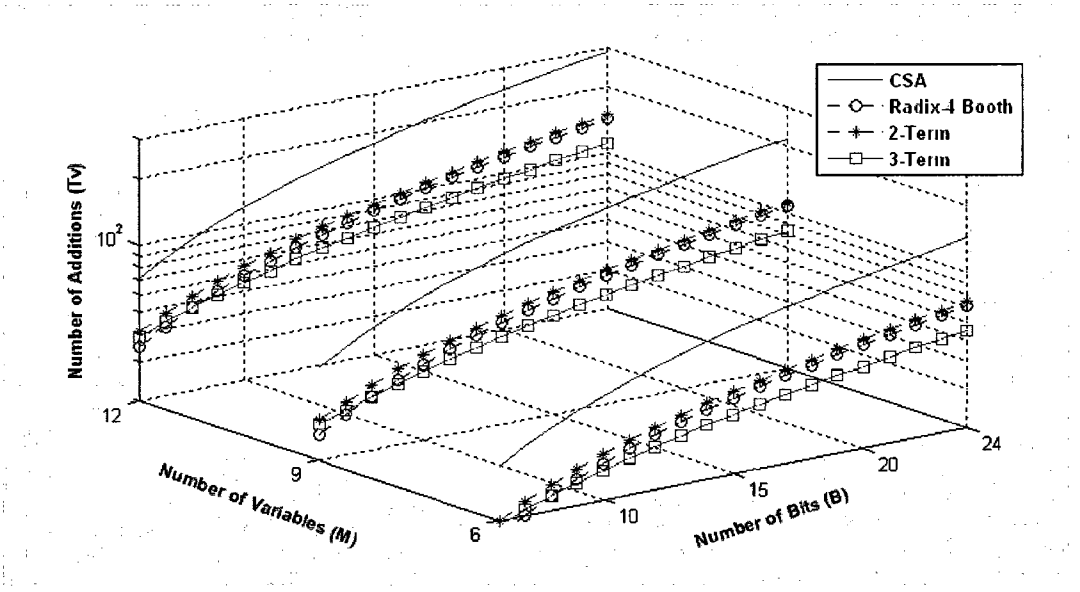


Figure 5.8: Number of additions versus word length B for M variables vector multiplication computations.

5.2.3.3 Matrix Multiplier

VECTOR AND MATRIX MULTIPLICATION

The multiplication of a 2 element vector and a 2-by-M matrix where the multiplier operands a_1 and a_2 have the same word length B can be expressed as follows:

$$[S_1 \;\; S_2 \;\; \ldots \;\; S_M] = [a_1 \;\; a_2] * \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,M} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,M} \end{bmatrix} \tag{5.21}$$

The individual outputs of this vector matrix multiplication are described as follows:

$$\begin{aligned} S_1 &= a_1 x_{1,1} + a_2 x_{2,1} \\ S_2 &= a_{1,} x_{1,2} + a_2 x_{2,2} \\ &\cdot \\ &\cdot \\ S_M &= a_1 x_{1,M} + a_2 x_{2,M} \end{aligned} \tag{5.22}$$

The individual sums in equation (5.22) could be optimized using the CSE-BitSlice method as follows:

$$S_i = \left(a_1 * \left(x_{1,i} \bullet \overline{x_{2,i}}\right)\right) + \left((a_1 + a_2) * \left(x_{1,i} \bullet x_{2,i}\right)\right) + \left(a_2 * \left(\overline{x_{1,i}} \bullet x_{2,i}\right)\right) \qquad i \in [1, M] \tag{5.23}$$

From (5.23), it can easily be seen that the total number of additions required to compute each output S_i is B based on the 2-Term decomposition method. For M outputs, a total of MB additions are required for this vector-matrix multiplication. However, since the sum of two multiplier operands $\left(a_{1,1} + a_{1,2}\right)$ is a common expression, it can be factored out. The total number of additions needed to produce all outputs from S_1 to S_M is now equal to $M(B-1)+1$. As a result, compared to the non-optimized implementation where $M(2B-1)$ additions are needed, there is an overall saving of $MB-1$ addition operations.

Let us consider the vector matrix multiplication shown in the following expression:

$$[S_1 \ S_2 \ \ldots \ S_K] = [a_1 \ \ldots \ a_M] * \begin{bmatrix} x_{1,1} & \cdots & x_{1,K} \\ \vdots & \ldots & \vdots \\ x_{M,1} & \cdots & x_{M,K} \end{bmatrix} \tag{5.24}$$

For the multiplication of a 1-by-M vector and an M-by-K matrix, each of the K outputs is the result of an M term vector multiplication. If the multiplier operand's word length B is large; each vector multiplication could be optimized on the basis of the M-Term decomposition technique. On the other hand, the sum of M terms could also be partitioned into groups of smaller sum value where the optimization technique can be applied instead. Using 2-Term decomposition, the total number of additions required to calculate the first output S_1 is $\frac{M}{2}(B+1)-1$. If three term decomposition is used instead, the number of additions required is $\frac{M}{3}(B+4)-1$. The outputs of $S_2, S_3, \ldots, S_K$ are determined on the basis of the same technique as used to produce the output S_1. Thus, the overall number of additions required by the vector and matrix multiplication based on 2-

Term decomposition is $\frac{MK}{2}(B+1)-K$. For 3-Term decomposition, the number of addition operations required for this vector matrix multiplication is $\frac{MK}{3}(B+4)-K$.

A comparison of the number of additions required to compute a 1-by-M and M-by-K vector-matrix product based on CSA, Radix-4 Booth, and CSE-BitSlice techniques is shown in Table 5.5. For the CSE-BitSlice case, optimization results based on 2-Term and 3-Term decomposition are presented.

Table 5.5: Comparison of CSE-BitSlice based vector-matrix multiplier and other design techniques.

Optimization Technique		**Number of Additions**
$[S_1\ S_2\ \ldots\ S_K]=[a_1\ \ldots\ a_M]*\begin{bmatrix} x_{1,1} & \ldots & x_{1,K} \\ \vdots & \ldots & \vdots \\ x_{M,1} & \ldots & x_{M,K} \end{bmatrix}$		
CSA		$K(MB-1)$
Radix-4 Booth		$K\left(\frac{MB}{2}-1\right)$
CSE-BitSlice	2-Term	$\frac{MK}{2}(B+1)-K$
	3-Term	$\frac{MK}{3}(B+4)-K$

MATRIX MULTIPLICATION

The multiplication of two M-by-M matrices can be expressed as follows:

$$\begin{bmatrix} S_{1,1} & \cdots & S_{1,M} \\ \vdots & \cdots & \vdots \\ S_{M,1} & \cdots & S_{M,M} \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,M} \\ \vdots & \cdots & \vdots \\ a_{M,1} & \cdots & a_{M,M} \end{bmatrix} * \begin{bmatrix} x_{1,1} & \cdots & x_{1,M} \\ \vdots & \cdots & \vdots \\ x_{M,1} & \cdots & x_{M,M} \end{bmatrix} \tag{5.25}$$

Using the decomposition technique for vector-matrix multiplication presented in the previous paragraphs, the $S_{i,j}$ where $i, j \in M$ could be each calculated as the product of 1-by-M vector and an M-by-M matrix. From the discussion of vector-matrix multiplication in the previous paragraphs, it is possible to determine the total number of additions for this matrix-matrix multiplication based on 2-Term and 3-Term decomposition. Thus, the total number of additions required for this M-by-M matrix multiplication is $M^2\left(\left(\frac{M}{2}(B+1)\right)-1\right)$ and $M^2\left(\left(\frac{M}{3}(B+4)\right)-1\right)$ based on 2-Term and 3-Term decompositions respectively. The total number of additions required to compute an M-by-M matrix multiplication optimized using N-Term decomposition where $M \geq 2$ can be expressed as follows:

$$T_m(n, M, B) = M^2\left(\frac{M}{n}\left(B + 2^n - n - 1\right) - 1\right) \qquad n \geq 2 \tag{5.26}$$

Proof: (By Induction.)

For n=2, the total number of additions needed to combine all partial results for the sum of products expression $S_{i,j}$ generated from the common digit elimination of $\frac{M}{2}$ pair of multiplier operands $x_{i,j}$ and $x_{i+1,j}$ is $\frac{M}{2}(B+1)-1$. Thus, the total number of additions required for the M-by-M matrices multiplication for n=2 is:

$$T_m(2, M, B) = M^2\left(\frac{M}{2}(B+1) - 1\right) \tag{5.27}$$

For n=3, the total number of additions needed to combine all partial results for the sum of products expression $S_{i,j}$ generated from the common digit elimination of $\frac{M}{3}$ sets of multiplier operands $x_{i,j}, x_{i+1,j}$ and $x_{i+2,j}$ is $\frac{M}{3}(B+4)-1$. Thus, the total number of additions required for the M-by-M matrices multiplication for n=3 is:

$$T_m(3,M,B) = M^2\left(\frac{M}{3}(B+4)-1\right) \tag{5.28}$$

For n=4, the total number of additions needed to combine all partial results for the sum of products expression $S_{i,j}$ generated from the common digit elimination of $\frac{M}{4}$ sets of multiplier operand $x_{i,j}, x_{i+1,j}, x_{i+2,j}$ and $x_{i+3,j}$ is $\frac{M}{4}(B+11)-1$. Thus, the total number of additions required for the M-by-M matrices multiplication for n=4 is:

$$T_m(4,M,B) = M^2\left(\frac{M}{4}(B+11)-1\right) \tag{5.29}$$

From (5.27), (5.28), and (5.29) it can be seen that $T_m(m+1,M,B)$ can be defined as a recursive equation as follows:

$$T_m(n+1,M,B) = M^2\left(\frac{M}{n+1}\left(B+\underbrace{\left(2^n-n-1\right)}_{\frac{n}{M}\left(\frac{T_m(n,M,B)}{M^2}+1\right)-B}+\in\right)-1\right)$$

where $\in=\left(2^n-1\right)$. Thus,

$$T_m(n+1,M,B) = M^2\left(\frac{M}{n+1}\left(\frac{n}{M}\left(\frac{T_m(n,M,B)}{M^2}+1\right)+\left(2^n-1\right)\right)-1\right)$$

Then upon simplifying, we obtain:

$$T_m(n+1,M,B) = M^2\left(\frac{n}{n+1}\left(\frac{T_m(n,M,B)}{M^2}+1\right)+\frac{M}{n+1}\left(2^n-1\right)-1\right)$$

Assume (5.26) is true for n=k: $T_m(k,M,B) = M^2\left(\frac{M}{k}\left(B+2^k-k-1\right)-1\right)$

Given this assumption, we will show that the statement is true for n=k + 1, since:

$$T_m(k+1,M,B) = M^2\left(\frac{k}{k+1}\left(\underbrace{\frac{M}{k}\left(B+2^k-k-1\right)-1}_{\frac{T_m(k,M,B)}{M^2}}+1\right)+\frac{M}{k+1}\left(2^k-1\right)-1\right)$$

$$T_m(k+1,M,B) = M^2\left(\frac{M}{k+1}\left(B+2^k-k-1\right)+\frac{M}{k+1}\left(2^k-1\right)-1\right)$$

Then upon simplifying, we obtain:

$$T_m(k+1,M,B) = M^2\left(\frac{M}{k+1}\left(B+2^{k+1}-(k+1)-1\right)-1\right) \quad QED$$

Therefore, by induction, the statement is true for all $n \geq 2$.

If the matrix multiplication circuit had been implemented using the CSA algorithm, the total number of additions required would be $M^3(B+1)-M^2$. As a result, almost a 50 percent saving of the number of additions could be achieved using the 2-Term CSE-BitSlice technique for matrix multiplication computation as compared to the CSA method. The number of additions required to compute the product of two matrices of dimension M-by-M based on the CSA, Radix-4 Booth, and CSE-BitSlice techniques is shown in Table 5.6.

Table 5.6: Comparison of the number of additions for M-by-M matrix multiplication based on different design techniques.

Optimization Technique $\begin{bmatrix} S_{1,1} & \cdots & S_{1,M} \\ \vdots & \cdots & \vdots \\ S_{M,1} & \cdots & S_{M,M} \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,M} \\ \vdots & \cdots & \vdots \\ a_{M,1} & \cdots & a_{M,M} \end{bmatrix} * \begin{bmatrix} x_{1,1} & \cdots & x_{1,M} \\ \vdots & \cdots & \vdots \\ x_{M,1} & \cdots & x_{M,M} \end{bmatrix}$		**Number of Additions**
CSA		$M^3(B+1)-M^2$
Radix-4 Booth		$M^2\left(\frac{MB}{2}-1\right)$
CSE-BitSlice	2-Term	$M^2\left(\frac{M}{2}(B+1)\right)-1$
	N-Term	$M^2\left(\frac{M}{N}\left(B+2^N-N-1\right)-1\right)$

Thus for M=N=3, using the CSE-BitSlice technique, a 3-by-3 matrix multiplication would require a total of $9B+27$ addition operations as compared to a total of 27B+18 and $\frac{27B}{2}-9$ additions required by CSA and Radix-4 Booth techniques respectively. The number of addition operations saved with the CSE-BitSlice technique, for B equal to 16 bits, is 62 and 20 per cent as compared to the CSA and Radix-4 Booth respectively. Figure 5.9 shows the block diagram of an 2-by-2 matrix multiplier circuit design based on the 2-Term decomposition approach.

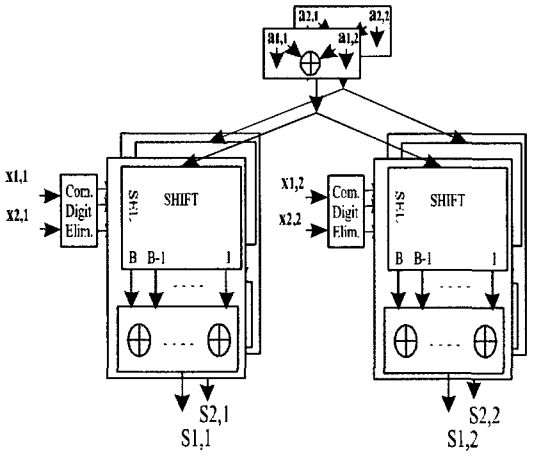


Figure 5.9: Block diagram of the 2-by-2 matrix multiplier circuit design based on the 2-Term decomposition approach.

Figure 5.10 illustrates the number of additions versus word length B for the M-by-M matrix multiplication required by the 2-Term and 3-Term CSE-BitSlice, as well as, the CSA and Radix-4 Booth techniques. The three graphs in Fig. 5.10 have been generated based on the matrix sizes of 6-by-6, 9-by-9, and 12-by-12.

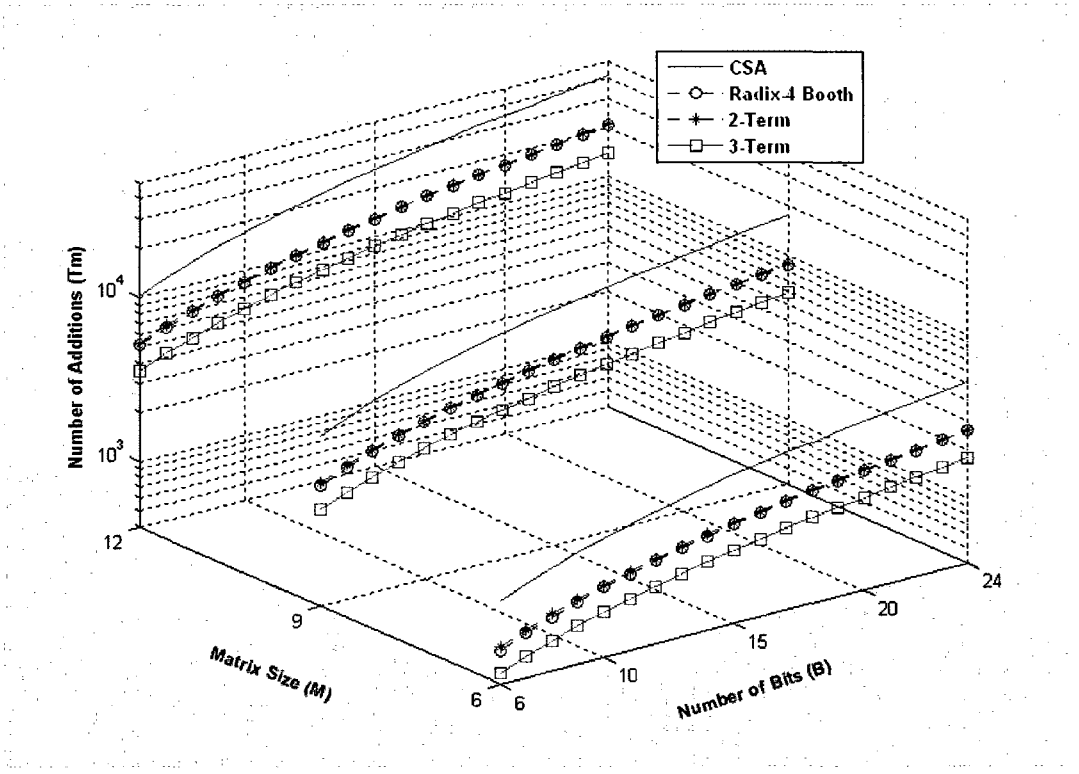


Figure 5.10: Number of additions versus word length B for an M-by-M matrix multiplier.

5.3 PE Circuit Design

The AU is the main building block of the PE cell where arithmetic operations are performed thus it dictates the throughput performances for the PE cell. As a result, it was determined that design techniques aiming at performance enhancement and complexity reduction for the RSA circuit need to be applied to the design of the AU building block.

In this section, the design of the PE, whose multiplier circuits in the AU building block are implemented on the basis of the CSE-BitSlice technique is described. Estimates of the total number of additions required by the multipliers in the PE cell design based on CSE-BitSlice technique are presented and compared to those of circuit designs based on

the CSA and Radix-4 Booth algorithms. Estimation results will be used to determine the number of terms to be employed in the design of the CSE-BitSlice based multiplier circuits in the AU building block.

The AU consists of four real multipliers and twenty real adders that can be configured to perform a wide range of arithmetic computations involving the input signals and a set of coefficients. The multipliers and adders in the AU are configurable and can perform complex multiplication, complex addition, real multiplication, and real addition as directed by the CU.

Complex multipliers are utilized for DFT/IDFT and phase shifter computations while Polyphase filters need an arithmetic circuit that supports real multiplication and addition. For complex multiplier circuit design, the CSE-BitSlice technique for vector-matrix multiplication presented in section 5.2 could be used. For real multiplier circuit design, CSE-BitSlice technique for multiplication between real variables discussed in the previous section could be applied. It is obvious that a common approach is needed for the design of the multiplier circuit to make it support complex multiplication, real multiplication, and addition. For that reason, the AU circuit is designed based on a combination of four independent real multiplier circuits. The CSE-BitSlice technique for multiplication of two variables is used to design each multiplier independently. Using the CSE-BitSlice technique on the design of each multiplier in the AU, four partial products are generated independently by the multiplier block. These partial products could be combined together or individually added to the set of local inputs shifted in from the north side of the PE cell to generate outputs involving complex multiplication, real multiplication, and/or addition according to the configuration mode. Figure 5.11 depicts the design of multipliers in the AU using the 2-Term decomposition technique.

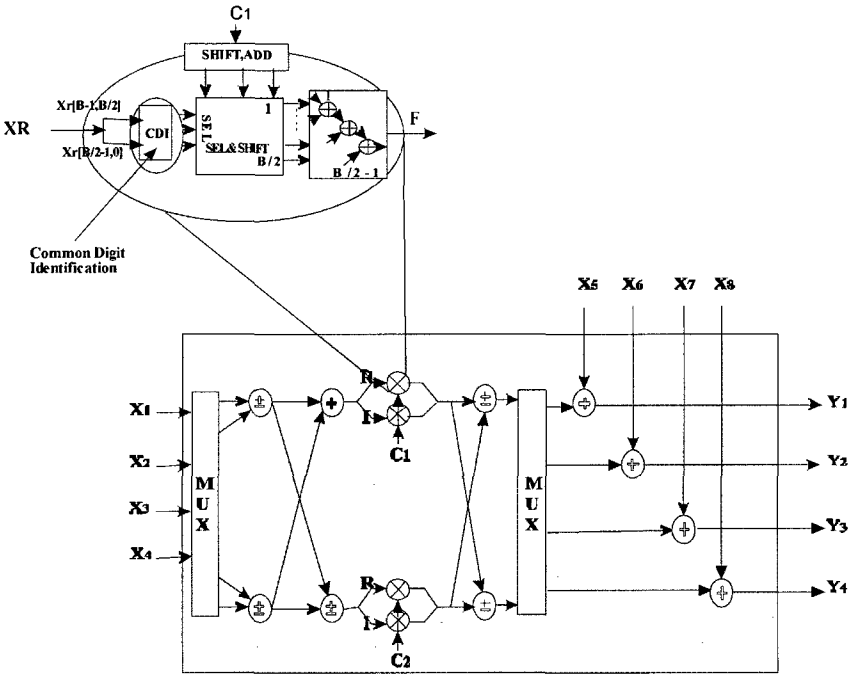


Figure 5.11: Design of the multiplier block in the AU using 2-Term decomposition approach.

As can be seen in Fig. 5.11, each of the four real multipliers performs multiplication involving the real or quadrature component of the complex inputs and the real coefficients and the partial results are shifted out to be added to other partial products or to be combined with inputs shifted in from the neighboring PE. For complex multiplication configuration, the AU generates one complex output, the result of four real multiplications of the complex inputs and a set of two real coefficients. For the real multiplication and accumulation configuration, the outputs generated by the four real multipliers in the AU are shifted out to be accumulated with data shifted in from the north side of the PE. The multiplexer at the output of the multiplier block switches the output signals according to the configured mode of operation.

If the 2-Term decomposition technique is used, each input X_k consisting of B bits is decomposed into two data slices, each of which consists of $\frac{B}{2}$ digits. The 2-Term based multiplication of X_k and C_k can be expressed as:

$$S_k = \begin{bmatrix} C_{k,1} & C_{k,2} \end{bmatrix} * \begin{bmatrix} X_{k,1} \\ X_{k,2} \end{bmatrix}$$

Where

$$C_{k,1} = C_k$$

$$C_{k,2} = C_k \ll 2^{\frac{B}{2}}$$

and $X_{k,1}$ $X_{k,2}$ are the first and the last $\frac{B}{2}$ digits of the input X_k respectively. By factoring out common digits of $X_{k,1}$ and $X_{k,2}$, a total of $\frac{B}{2}$ additions would be required to compute S_k. Thus the total number of additions required to compute all four real multiplications using 2-Term CSE-BitSlice technique is $2B$.

Now if the 3-Term decomposition technique is used, the multiplication of X_k and C_k can be expressed as:

$$S_k = \begin{bmatrix} C_{k,1} & C_{k,2} & C_{k,3} \end{bmatrix} * \begin{bmatrix} X_{k,1} \\ X_{k,2} \\ X_{k,3} \end{bmatrix}$$

where

$$C_{k,1} = C_k;\ C_{k,2} = C_k \ll 2^{\frac{B}{3}};\ C_{k,3} = C_k \ll 2^{\frac{2B}{3}}$$

and $X_{k,1}$, $X_{k,2}$, and $X_{k,3}$ are the first, second and third segment of $\frac{B}{3}$ digits of the X_k respectively.

Factoring out the common digits from $X_{k,1}, X_{k,2}$ and $X_{k,3}$, a total of $\frac{B}{3}+3$ additions is required to compute the S_k. Thus the total number of additions required for all four real multiplications using the 3-Term CSE-BitSlice technique is $4\left(\frac{B}{3}+3\right)$. Compare to the 2-Term decomposition where a total of $2B$ additions is needed to compute S_1, S_2, S_3, and S_4, the 3-Term decomposition would result in a circuit with lower complexity if:

$$2B > 4\left(\frac{B}{3}+3\right)$$

$$\Rightarrow B > 18$$

As a result, if the word length B of X_k were larger than 18 bits, using 3-Term optimization technique would be more advantagous in terms of the number of additions required. However, if B were smaller or equal to 18 bits, using the 2-Term method would produce a more compact circuit. Hardware saving could be larger if X_k is decomposed into more terms. However, the number of terms to be selected for this multiplier design has to be determined in conjunction with the word length B of X_k. Table 5.7 presents the number of additions needed to compute all four real multipliers in the AU circuit based on CSA, Radix-4 Booth, and CSE-BitSlice techniques.

The results shown in Table 5.7 indicate that if B has an 8 bit word length and the 2-Term CSE-BitSlice optimization technique is used, there is a saving of 50 percent for the multiplier circuit design as compared to the CSA-based design. If B has a 24 bit word length and the 3-Term technique is used, the hardware saving for the multiplier designs now corresponds to 52 and 8 percent as compared to the CSA and the Radix-4 Booth designs respectively.

Table 5.7: Comparison of the number of additions required by four AU multipliers designed using different techniques.

Optimization Technique $S_k = C_k * X_k$ $k = 1,2,3,4$	**Number of Adders**	
CSA	$4(B-1)$	
Radix-4 Booth	$2B-4$	
CSE-BitSlice	2-Term	$2B$ $(B \leq 18)$
	3-Term	$4\left(\frac{B}{3}+3\right)$ $(84 \geq B > 18)$

Chapter 6

RSA Circuit Configurations

6.1 Introduction

In this chapter, RSA circuit configurations for the computation of DSP functions that are suitable for MC applications are described. Mappings of GM and GD building blocks on the RSA circuit, which include Polyphase filter, phase shifters and DFT/IDFT are illustrated. RSA circuit configurations for the Polyphase filter and phase shifters are presented in Section 6.2 and Section 6.3 respectively. The technique proposed for configurations of DFT/IDFT on the RSA circuit to achieve low arithmetic complexity is described in Section 6.4. Several representative DFT/IDFT circuit configurations and implementations are discussed in this section. Comparisons of throughput performance and hardware complexity of the RSA-based DFT/IDFT circuit with reconfigurable DFT architectures are also presented in this section. RSA configurations for Polyphase-DFT and IDFT-Polyphase circuits are described in Section 6.5. Hardware complexity and throughput performance of the RSA-based Polyphase-DFT/IDFT-Polyphase circuit mappings based on parallel and sequential schemes are also determined and discussed in this section.

6.2 Polyphase Filter Mappings

As described in Chapter 3, a Polyphase filter is a filter bank consisting of a number of FIR filters that operate independently. Thus, an RSA configuration for a Polyphase filter is the same as an RSA configuration for a bank of individual FIR filters

where each one of them has the same number of taps but has a different set of coefficients.

For the Polyphase/FIR filter configurations, each PE cell in the RSA array is configured to perform four real multiplications involving two complex input data sequences and two filter coefficients and the partial products are then added to data shifted in from the previous stage to produce the partial result as shown in Fig. 6.1

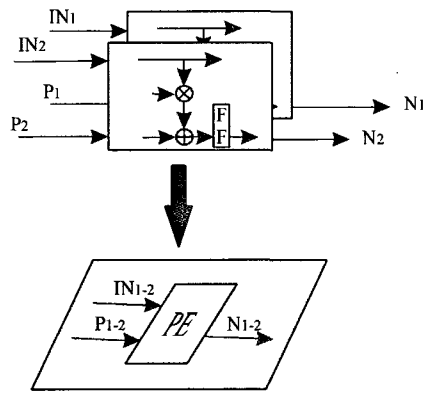


Figure. 6.1: Configuration of a PE cell for Polyphase/FIR filtering computation.

In Fig. 6.1, IN_1, IN_2 represent the two input sequences, P_1, P_2 represent two partial results from previous stages and N_1, N_2 are partial results of one tap calculations for two Polyphase/FIR filters. Thus, for a 2-channel and 4-tap Polyphase filter circuit mapping, the RSA structure consists of a 1-by-4 array. Figure 6.2 shows the mapping of the 2-channel Polyphase filter on an RSA circuit consisting of a 1-by-4 array of PE cells. Each FIR filter mapped onto this RSA structure has four taps.

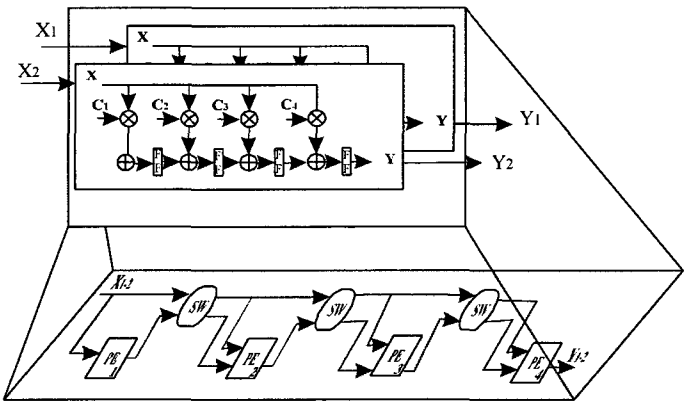


Figure 6.2: Configuration of the 2-channel, 4-tap Polyphase filter on an RSA structure consisting of a 1-by-4 array of PE cells.

For the Polyphase/FIR filter mapping, the SWs route primary signals from the inputs of the RSA circuit to each PE cell. Secondary signals shifted out from the previous PE cell in the array are routed to the current PE cell to be combined with partial results generated by multiplication process. Figure 6.3 depicts the mapping of the SW in a Polyphase/FIR filter circuit configuration.

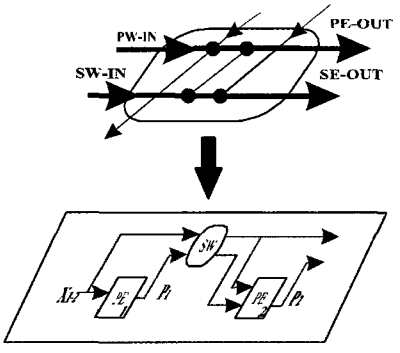


Figure 6.3: SW configuration for Polyphase/FIR filter mapping.

6.3 Phase Shifter Mapping

Each PE cell contains four real multipliers, which is suitable for a circuit configuration consisting of one phase shifter. A PE cell configured as a one channel phase shifter is shown in Fig. 6.4.

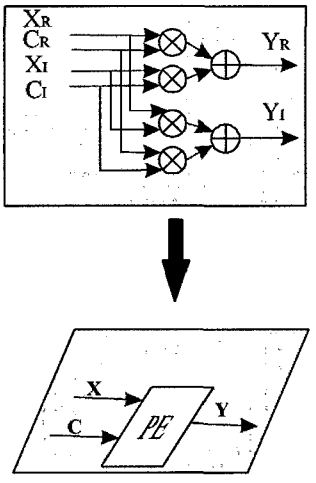


Figure 6.4: Mapping of phase shifter circuit onto one PE cell.

In Fig. 6.4, X_R, X_I and C_R, C_I are the real and imaginary components of the complex input X and the complex coefficient C respectively. The complex output Y is represented by real and imaginary components Y_R and Y_I respectively. Figure 6.5 depicts the mapping of 4-channel phase shifters on an RSA structure consisting of a 4-by-1 array of PE cells.

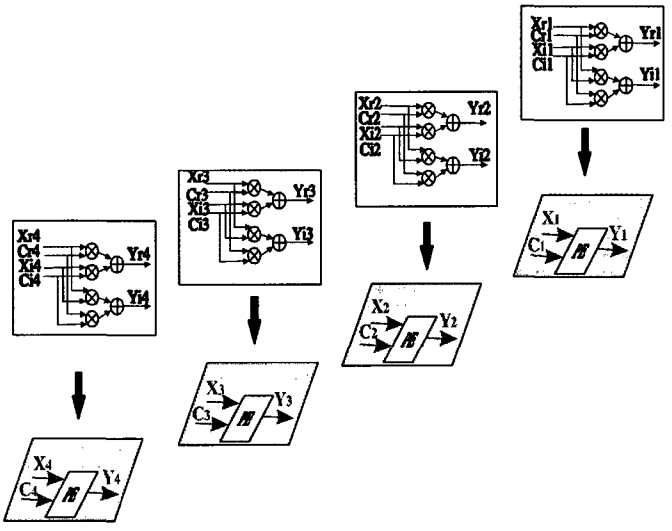


Figure 6.5: RSA mapping for a 4-channel phase shifter circuit.

6.4 DFT/IDFT Mappings

6.4.1 Low Complexity Algorithm for DFT/IDFT Computation

The modularity and regularity of the RSA structure enables circuit mappings for DFT computation for a wide range of input sequences N. The regularity of the RSA architecture requires that the DFT computations be carried out based on the direct multiplication of the input sequence elements and the DFT coefficients such that the partial products are accumulated independently. For the Fourier transform computation using the DFT/IDFT algorithm the arithmetic calculations are performed in a non-recursive fashion and all the outputs are calculated independently from one another. However, direct mapping of DFT/IDFT on the RSA would require considerable arithmetic resources especially when N is large. As a result, a design technique that

streamlines the arithmetic operations for DFT/IDFT to alleviate logic resource requirements of the RSA circuit is necessary.

The DFT and IDFT functions are expressed by the following equations:

$$Y(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi nk}{N}} \qquad k = 0,...,N-1 \qquad \text{(DFT)} \tag{6.1}$$

$$y(n) = \sum_{k=0}^{N-1} X(k) e^{\frac{j2\pi nk}{N}} \qquad n = 0,...,N-1 \qquad \text{(IDFT)} \tag{6.2}$$

As can be seen from (6.1) and (6.2), the outcome of each DFT/IDFT output depends on partial products generated from the multiplication of the input sequences and the DFT/IDFT coefficients that have been calculated independently. Direct DFT/IDFT computation of N input data however, requires a total of N^2 complex multiplications and $(N-1)N$ complex additions. For large N, the number of multiplications and additions required for direct DFT computation become substantial. However, the number of arithmetic operations required for each DFT/IDFT computation can be reduced by a factorization process. The factorization process is applied to the multiplication of the input vector and each set of coefficient vectors to enable the multiplications to be carried out with fewer arithmetic operations. The factorization process identifies inputs that multiply a common coefficient. These inputs are added together and the result is multiplied by the common coefficient. This process is repeated for the remaining coefficients and partial products are then accumulated to produce the output Y(k). The output Y(N-k) could also be generated simultaneously without the need of extra multiplications based on the conjugate values of partial products generated for the computation of Y(k). The same factorization process is repeated to generate the remaining outputs of the DFT/IDFT. Thus, the factorization process reduces multiple

multiplication operations required to generate several partial products into a single multiplication. This simplification technique helps to reduce the overall number of complex multiplications and additions for DFT/IDFT computation. The factorization process is examined in detail in Appendix C for the case that N the length of the input sequence is even or odd.

From the results depicted in Appendix C, the total number of complex multiplications required to compute all N outputs simultaneously is $\left(\frac{N}{4}\right)^2$ for an N-point DFT where N is a multiple of four. Similarly, a total of $\left(\frac{N}{2}\right)\left(\frac{N+2}{4}\right)$ complex multiplications is required to compute an N-point DFT where N is even but not a multiple of four. To compute the outputs of the N-point DFT where N is odd, the total number of complex multiplications required is $\left(\frac{N-1}{2}\right)^2$. The total number of complex operations required to compute an N point DFT/IDFT is summarized in Table 6.1. From Table 6.1 we see that for the three cases presented the smallest number of complex multiplications occurs when N is a multiple of four.

Table 6.1: Number of complex operations for DFT/IDFT computation based on proposed factorization technique.

N / Arithmetic Operations	Even		Odd
	N=4n	**N=4n+2**	**N=2n+1**
Complex Multiplication	$\left(\frac{N}{4}\right)^2$	$\left(\frac{N+2}{4}\right)\left(\frac{N}{2}\right)$	$\left(\frac{N-1}{2}\right)^2$
Complex Addition	$\left(\frac{3N}{2}+6\right)\left(\frac{N}{4}\right)$	$\left(\frac{3N}{2}-1\right)\left(\frac{N}{2}\right)$	$2N\left(\frac{N-1}{2}\right)$

6.4.2 Circuit Configurations

For DFT/IDFT circuit configurations, each PE cell in the RSA array is configured to perform one complex multiplication involving the combined input data sequence and the factored coefficient. The partial products are then added to or subtracted from the results shifted in from the previous stage to produce the partial outputs. Depending upon the nature of N, the combined input data sequence could be the result of the sum of two or four input sequences. Similarly, the total number of partial outputs generated by the multiplication process is equal to four if N is a multiple of four or equal to two otherwise. The mapping of arithmetic operations on a PE cell for DFT/IDFT functions where N=4n, N=4n+2, and N=2n+1 are shown in Fig. 6.6.

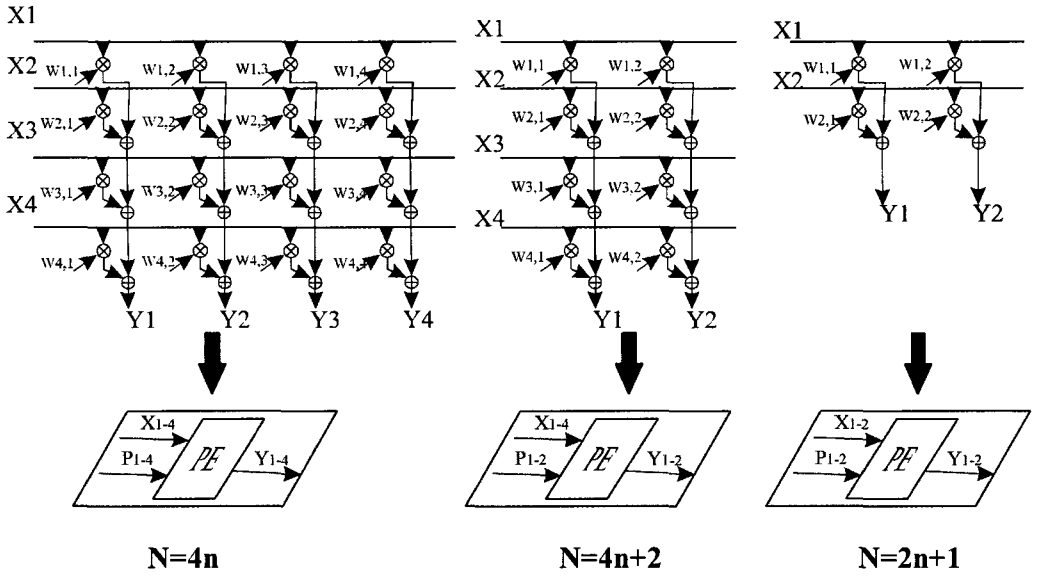


Figure 6.6: Configuration of a PE cell for DFT/IDFT computations.

Mapping of RSA for N-point DFT/IDFT computation can be carried out for parallel or sequential computation modes of operation.

6.4.2.1 DFT/IDFT circuit mappings for parallel computations

For parallel computation of 16-point DFT, an RSA structure consisting of a 4-by-4 array of PE cells is needed. The mapping of the 16-point DFT/IDFT circuit onto the RSA structure is shown in Fig. 6.7.

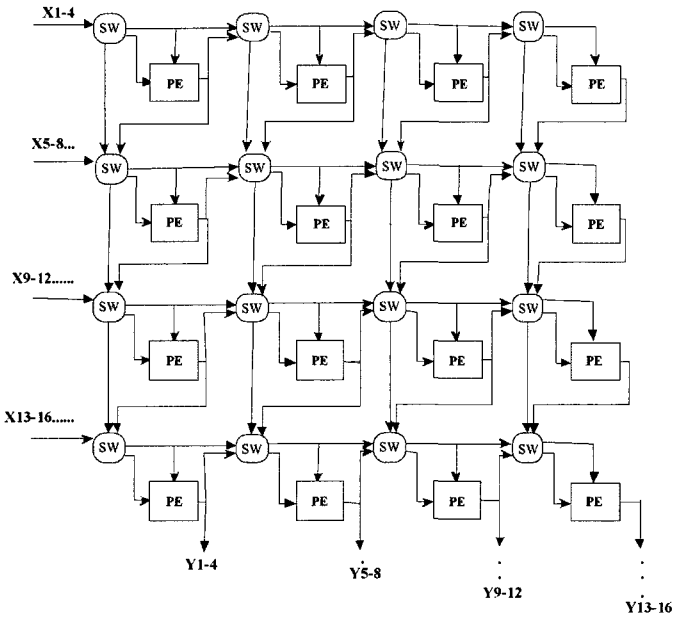


Figure 6.7: RSA mapping for parallel computation of 16-point DFT/IDFT.

As shown in Fig. 6.7, the PE cells in the RSA structure have been configured to execute complex multiplication and accumulation operations. Each PE cell is configured to perform exactly one complex multiplication. Thus, the overall number of complex multiplication operations that can be executed by this configuration is 16. For this 16-point DFT/IDFT mapping, the SWs route primary signals shifted in to the RSA circuit on

the west side to all PE cells in the array. Secondary signals shifted out from the neighboring PE cell on the north side are routed to the PE cell to be combined with the partial results generated by the multiplication process. Figure 6.8 depicts the configuration for an SW for this 16-point DFT/IDFT circuit mapping.

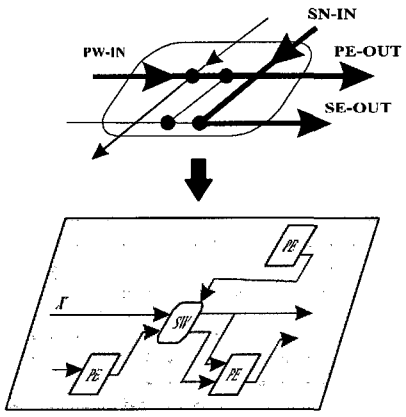


Figure 6.8: SW configuration for parallel computation of 16-point DFT/IDFT.

For parallel computation of a 14-point DFT, an RSA structure consisting of a 4-by-7 array of PE cells is needed. The mapping of the 14-point DFT/IDFT circuit onto the RSA structure is shown in Fig. 6.9. The overall number of complex multiplications required for this configuration is 28.

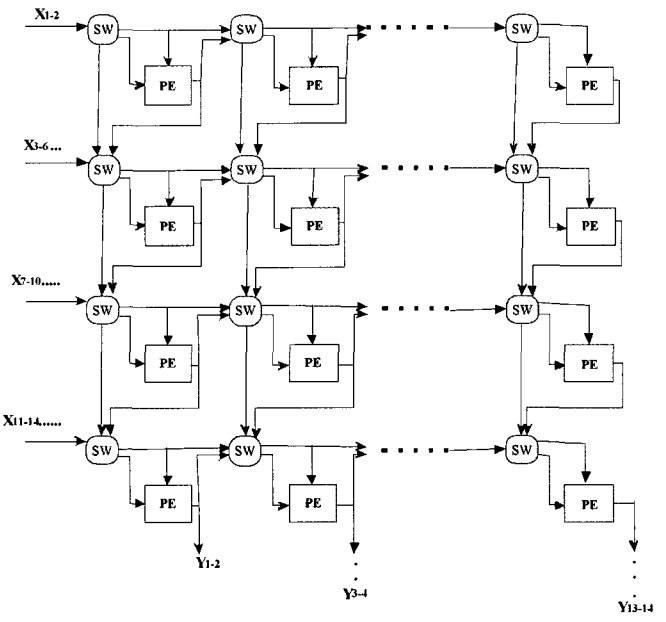


Figure 6.9: RSA mapping for parallel computation of 14-point DFT/IDFT.

For parallel computation of an 11-point DFT, an RSA structure consisting of a 5-by-5 array of PE cells is needed. The mapping of the 11-point DFT/IDFT circuit onto the RSA structure is shown in Fig. 6.10. For the 11-point DFT/IDFT circuit configuration, the overall number of complex multiplications required is 25.

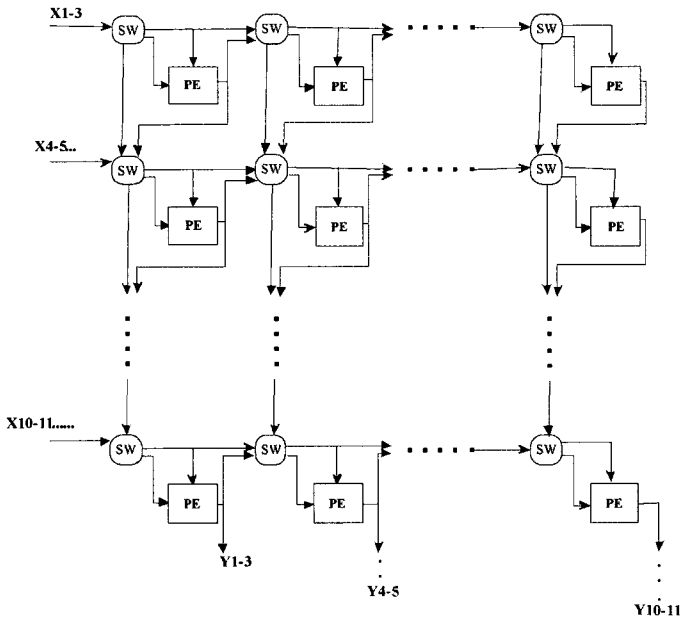


Figure 6.10: RSA mapping for parallel computation of 11-point DFT/IDFT.

The SW cells in the 14-point and 11-point DFT/IDFT mappings have the same configuration as the SWs in the 16-point DFT/IDFT circuit configuration shown in Fig. 6.8.

6.4.2.2 DFT/IDFT circuit mappings for sequential computations

As can be seen in Fig. 6.6, a set of m outputs ($m=4$ or $m=2$) is calculated and shifted out independently by a block of PE cells in the same column of the RSA. An RSA structure with N/m columns of PE cells all of whom operate simultaneously produce the N outputs in parallel. Since each one of the N/m columns consists of the same number of PE cells with identical inter signal connections, only a small number of these columns are needed for time-shared mode of computation.

Three mapping schemes for sequential computation of DFT/IDFT are shown in Fig. 6.11a, Fig. 6.11b and Fig. 6.11c. The block diagram shown in Fig. 6.11a is a parallel

inputs- serial outputs mapping scheme. As shown in Fig. 6.11a, all input sequences in this case are shifted in simultaneously and output sequences are available after $N/(c*m)$ clock cycles based on an RSA with c columns of PE cells ($c \in \left[1, \frac{N}{m}\right]$). The second mode of sequential mapping is shown in Fig. 6.11b where input sequences are shifted in serially and output sequences are shifted out in parallel. This serial inputs-parallel outputs mapping scheme shifts m input sequences into the circuit on every clock cycle. The products generated by the multiplication process are then combined with previously obtained partial results to produce a set of partial results on every clock cycle. All N outputs are available after $N/(r*m)$ clock cycles where r is the number of rows of PE cells ($r \in \left[1, \frac{N}{m}\right]$) in the RSA structure. The block diagram shown in Fig. 6.11c is a serial inputs- serial outputs mapping scheme. For this mapping scheme, all input and output sequences are shifted in and out of the circuit serially. For the serial inputs-serial outputs mapping, all N outputs are available after $\left(N/(r*m)\right)*\left(N/(c*m)\right)$ clock cycles where r and c are the number of rows and columns of PE cells in the RSA structure respectively.

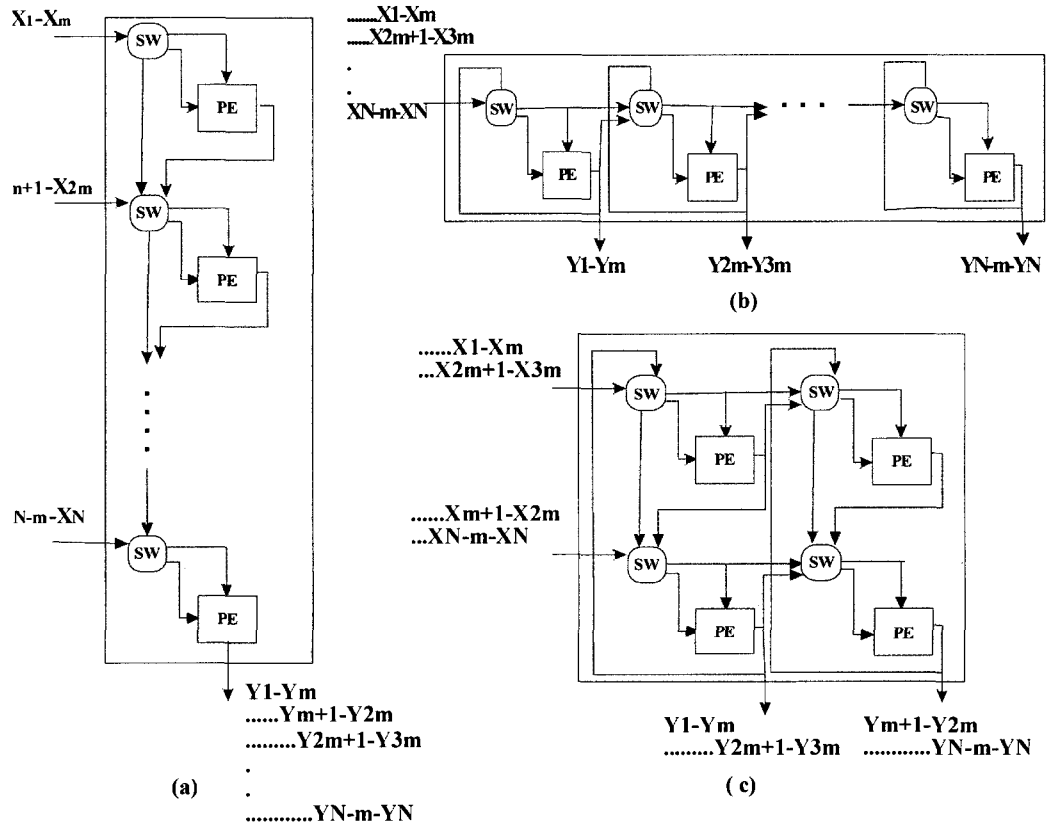


Figure 6.11: RSA mappings for sequential computation of DFT/IDFT: a) Parallel inputs-serial outputs b) Serial inputs-parallel outputs c) serial inputs-serial outputs.

The mapping of a serial inputs-serial outputs circuit for 1024-point DFT/IDFT computation is shown in Fig. 6.12 for illustration purposes. Figure 6.12 shows an RSA structure consisting of a 1-by-64 array of PE cells where a set of four input sequences is shifted in. The complete 1024 DFT/IDFT output sequence is available at the end of 1024 clock cycles based on this serial inputs-serial outputs configuration.

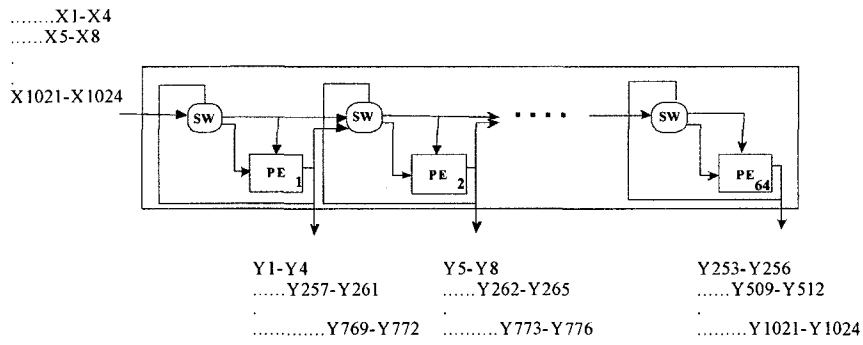


Figure 6.12: RSA circuit mapping for serial inputs-serial outputs implementation of 1024 point DFT.

For this serial inputs-serial outputs 1024-point DFT/IDFT circuit mapping, the SWs route primary signals from the inputs of the RSA structure on the west side to all PE cells in the array. Secondary signals from the previous computation are looped back and routed to the north side of the SWs to be combined with the partial results generated by the multiplication process. Figure 6.13 depicts the configuration for the SW in this serial inputs-serial outputs DFT/IDFT circuit mapping.

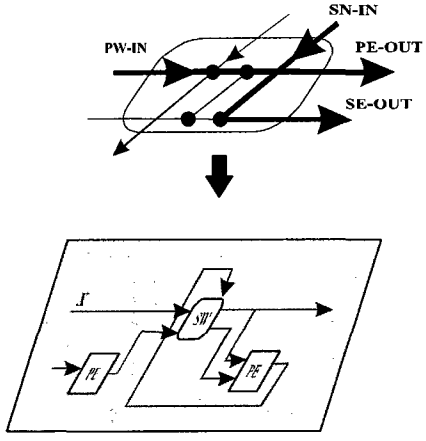


Figure 6.13: SW configuration for serial inputs-serial outputs implementation of 1024-point DFT.

6.4.3 Comparison of hardware complexity of N-point DFT configured on different reconfigurable architectures.

In this section, the hardware complexity of several existing systolic structures are evaluated and compared with the RSA-based DFT structures. Performances are also determined on the basis of the total number of clock cycles required to produce a complete N output sequence for each structure. The constraints in terms of the size of N supported by each of the architectures are also presented. The RSA circuit mappings used in this comparison are based on the serial inputs-parallel output configuration. Table 6.2 presents comparison results for throughput and hardware requirements of the RSA-based structure and circuit designs based on 2D-Systolic [Lim'99], base-4 [Nash'05], the RADComm [Gray'00], the CSoC [Wall'05], and the Modular architectures [Sapi'90]. For the comparisons between the RSA and the 2D-Systolic, base-4 and Modular based circuits, the DFT configured for multiple of four ($N=4n$) input sequences is used. For the comparison between the RSA circuit and the RADComm circuit, all three DFT configurations ($N=4n, N=4n+2$, and $N=2n+1$) are used in the analysis.

Table 6.2: Comparison for throughput and hardware requirements of the RSA-based DFT circuit and several existing DFT structures.

Architecture	**# Complex Multipliers**	**Throughput (Clock Cycles/N DFT Outputs)**	**Limit on N**
2-D Systolic [Lim'99]	N	N_3+N_4+1	$N=N_3*N_4$ $\left(N_3*N_4=2^n\right)$
base-4 [Nash'05]	$\frac{N_3}{4}$	$\frac{N}{4}+\frac{N_4^2}{4}+4N_4+28$	$N=N_3*N_4$ $\left(N_3*N_4=256n\right)$
Modular [Sapi'90]	$\frac{N}{2}$	N	$N=2^n$
CSoC [Wall'05]	4	$2\frac{N}{8}\log 2^n$	$N=2^n$
RADComm [Gray'00]	N	N	$N=2^n$
RSA	$\frac{N}{4}$	$\frac{N}{4}$	$N=4n$
	$\frac{N}{2}$	$\frac{N+2}{4}$	$N=4n+2$
	$\frac{(N-1)}{2}$	$\frac{N-1}{2}$	$N=2n+1$

Figure 6.14 and Fig. 6.15 depict the number of clock cycles per complete set of N DFT outputs and the number of complex multipliers required for the pairs RSA/base-4 and RSA/2D-systolic structures respectively. The RSA structure requires more complex multipliers as compared to the base-4 structure as can be seen in the lower part of Fig. 6.14. In exchange for greater hardware requirements however, the RSA structure offers

higher throughput. Results shown in the upper part of Fig. 6.14 demonstrate that the RSA structure always requires fewer clock cycles to produce a complete set of N DFT outputs as compared to the base-4 structure.

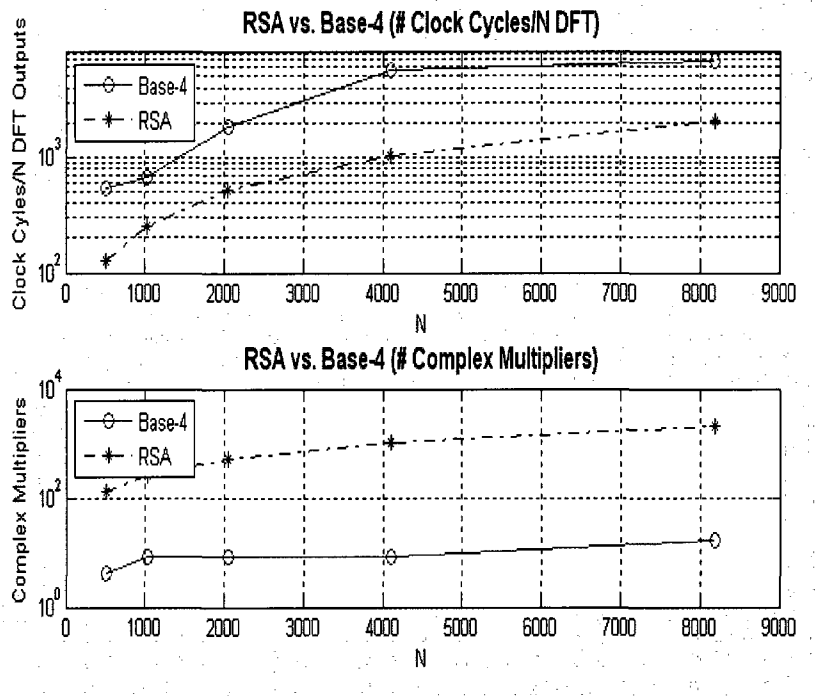
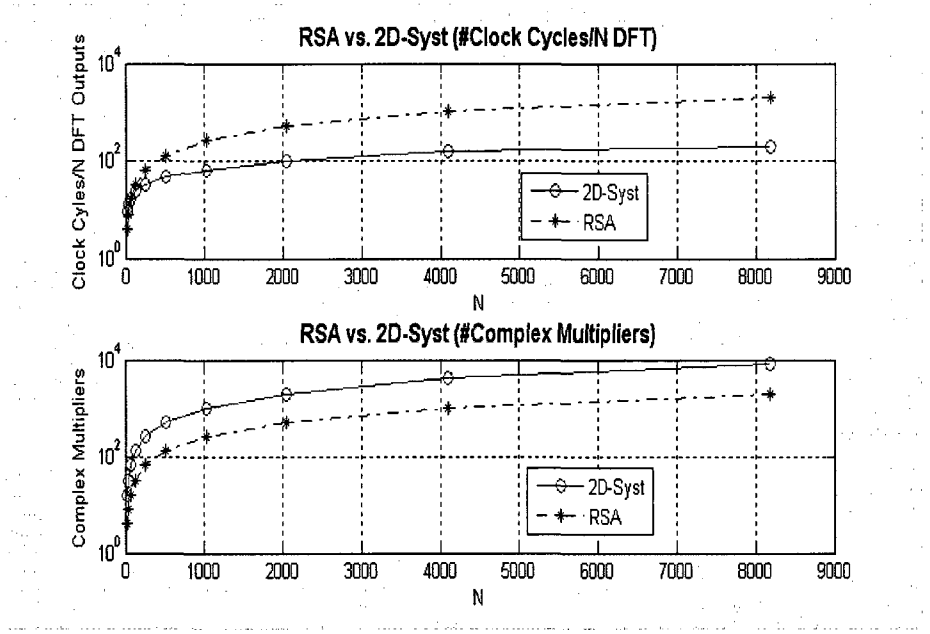
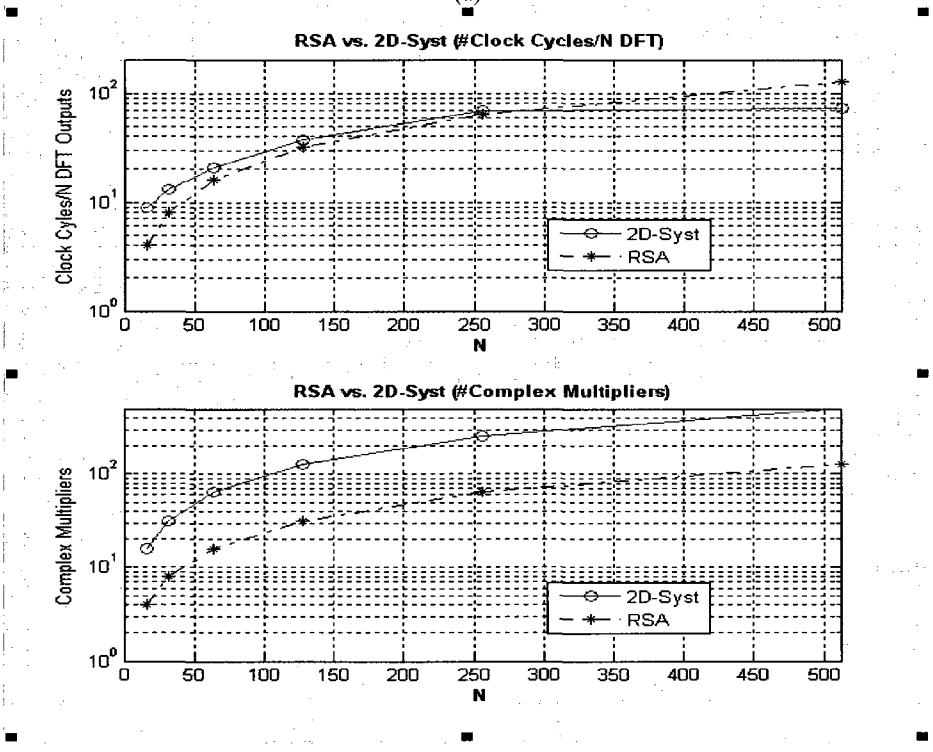


Figure 6.14: Throughput and complex multiplier comparison of the RSA-based DFT circuit and the base-4 structure [Nash'05].



(a)



(b)

Figure 6.15: Throughput and complex multiplier comparison of the RSA-based DFT circuit and the 2D-systolic structure [Lim'99] a) N=4 to N=8192 b) N=4 to N=512.

As shown in Fig. 6.15a and Fig. 6.15b, the number of complex multipliers required by the RSA-based structure is always equal to $\frac{1}{4}$ of the number of multipliers required by the 2D-systolic structure regardless of the size of the input sequence N. The 2D-systolic architecture requires fewer clock cycles to compute the N DFT outputs for N greater than 256 as shown in Fig. 6.15b. Thus, the RSA structure is more compact and offers higher throughput as compared to the 2D-systolic array structure if N smaller than 256. The modular architecture on the other hand requires more complex multipliers as well as more clock cycles to produce N DFT outputs as compared to the RSA-based circuit as depicted in Fig. 6.16.

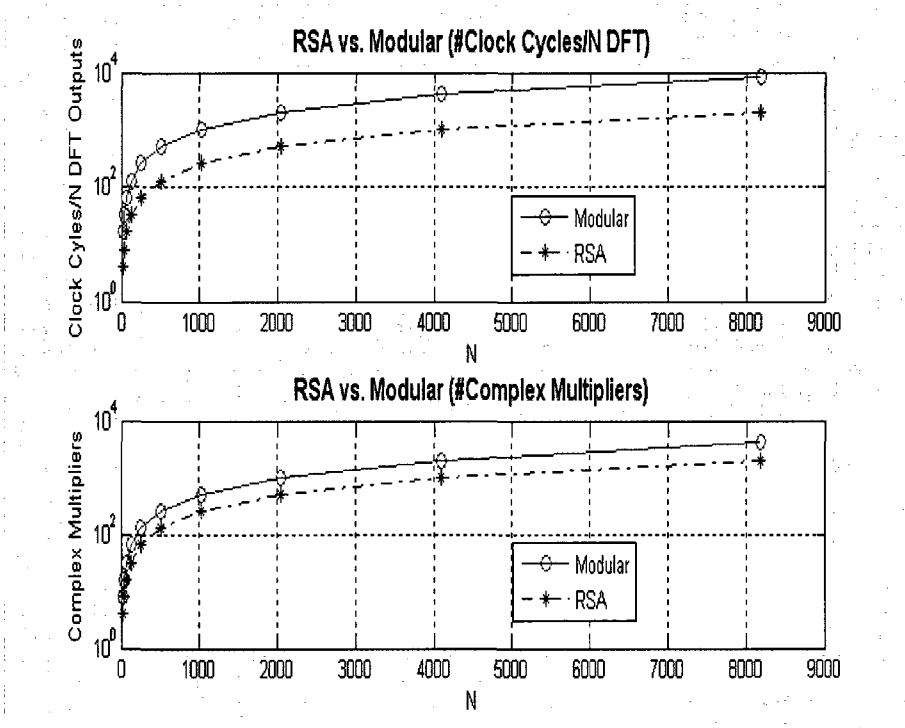


Figure 6.16: Throughput and complex multiplier comparison of the RSA-based DFT circuit and the Modular structure [Sapi'90].

The base-4, the 2D-systolic and the modular structures offer limited flexibility where the DFT dimensionality is concerned. The 2D-systolic and the modular

architectures are only applicable for DFTs for which N is a power of two and the base-4 architecture is applicable only for DFTs for which N= 256n. The RSA architecture presented in this section works for any N as long as it is a multiple of four.

The RADComm circuit supports DFTs for even or odd values of N. Compared to the RSA-based DFT circuit, the total number of complex multipliers used in the RADComm circuit is four times larger if $N = 4n$ and two times larger otherwise. For N even, the RADComm circuit requires four times the number of clock cycles as compared with the RSA based circuit to compute an N point DFT. For N odd, the RADComm circuit requires twice the number of clock cycles as the RSA circuit. Figure 6.17 shows comparison results for throughput performance and the number of complex multipliers of the RSA-based DFT circuit versus the RADComm circuit.

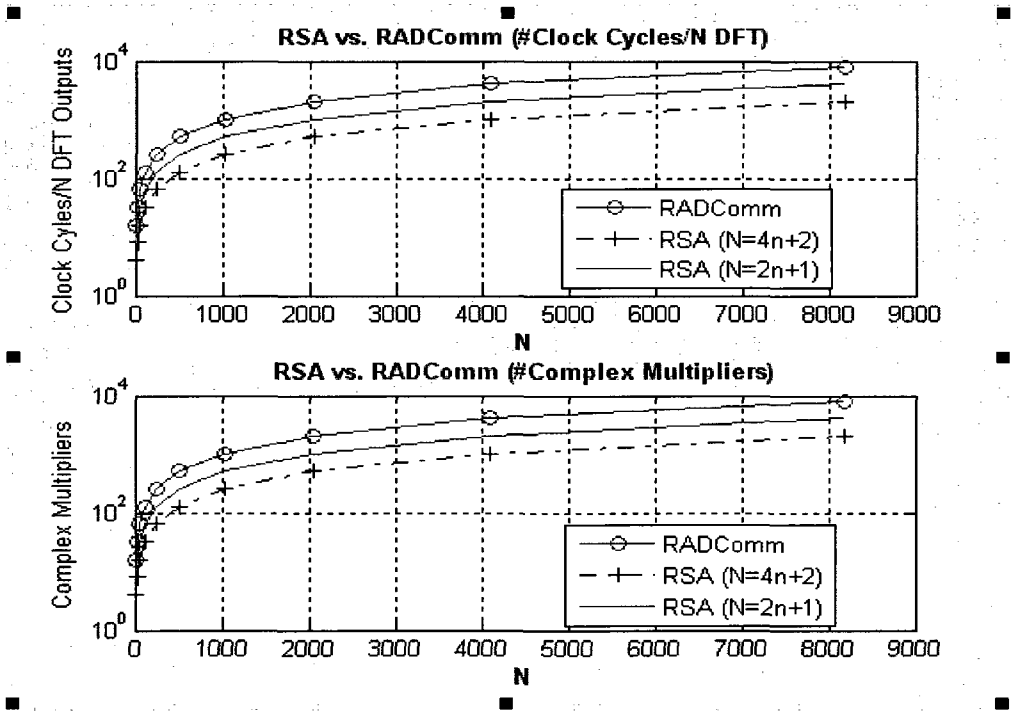


Figure 6.17: Throughput and complex multiplier comparison of the RSA-based DFT circuit and the RADComm structure [Gray'00].

The CSoC circuit incorporates 16 PE cells per cluster together with fixed arithmetic blocks in the circuit, which can be configured to perform 4 complex butterfly computations. For a 64-point FFT, approximately 100 clock cycles are needed to perform the complex butterfly computations and this does not include the number of clock cycles required for circuit configuration. In addition, the FFT function has to be partitioned into individual tasks for configuration onto the circuit and this process has to be performed manually. For FFTs for which N is large, this architecture would require too many clock cycles to be practical.

6.5 Polyphase-DFT and IDFT-Polyphase Circuit Configurations

6.5.1 Parallel configurations

As presented in Chapter 3, the polyphase-DFT/IDFT-Polyphase circuit consists of three main building blocks: a Polyphase filter, phase shifters and a DFT or IDFT. RSA circuit configurations for parallel realization of Polyphase-DFTs consist of three cascaded blocks namely Polyphase filter, phase shifters and DFT circuits as shown in Fig. 6.18. For this configuration, the Polyphase filter and the phase shifters operate at twice and four times the system clock respectively. The DFT building block on the other hand, operates off the system clock.

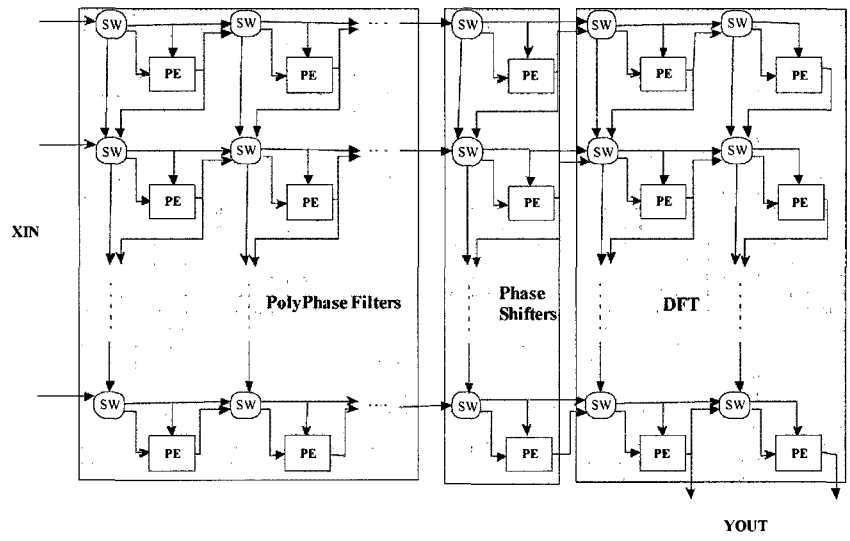


Figure 6.18: RSA mapping for parallel computation of N-channel Polyphase-DFT.

The mapping for parallel computation of the 8-channel Polyphase-DFT on an RSA structure and the mapping of the SWs associated with each building block in the circuit are shown in Fig. 6.19. For this configuration, the Polyphase filter consists of 8 sub-filters each of which is a 5-tap FIR filter. The Polyphase filter occupies the first block of PE cells in the RSA structure as shown in Fig. 6.19. Each row of the RSA structure accepts 4 input sequences and shifts out four filtered outputs, all in complex format. Thus, the total number of PE cells needed for this Polyphase configuration is 10 and the mapping onto the RSA structure is similar to the mapping shown in Fig. 6.2. The phase shifters mapping requires 2 PE cells configured as complex multipliers and the mapping is similar to that of the circuit shown in Fig. 6.5. The mapping for the 8-point DFT circuit requires a 2-by-2 array of PE cells as shown in Fig. 6.19. Thus, for this 8-channel Polyphase-DFT circuit configuration, a 2-by-8 RSA structure consisting of a total of 16 PE cells is required.

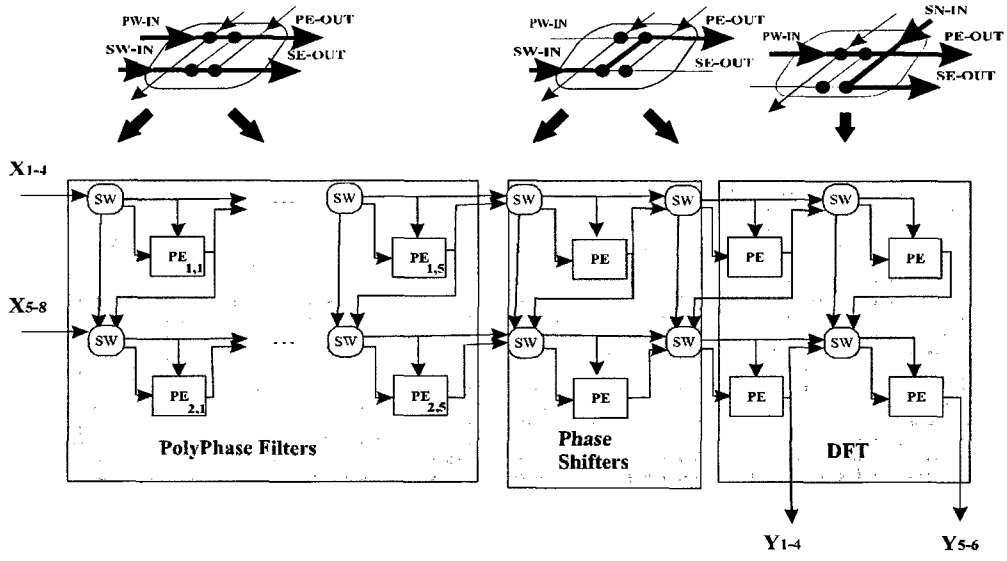


Figure 6.19: RSA and SWs mappings for parallel computation of an 8-channel Polyphase-DFT.

The mapping of an 8-channel IDFT-Polyphase on an RSA structure and the mapping of the SWs associated with each building block in the circuit are shown in Fig. 6.20. This mapping requires an RSA structure with the same number of PE cells as the one used for the 8-channel Polyphase-DFT circuit mapping. The same building blocks as those used in the mapping of the 8-channel Polyphase-DFT have been used for this circuit mapping except for the change in the relative positions of the IDFT and the Polyphase filters. Also, the configuration mode for the SW cells in the IDFT block is different from the one employed in the DFT block.

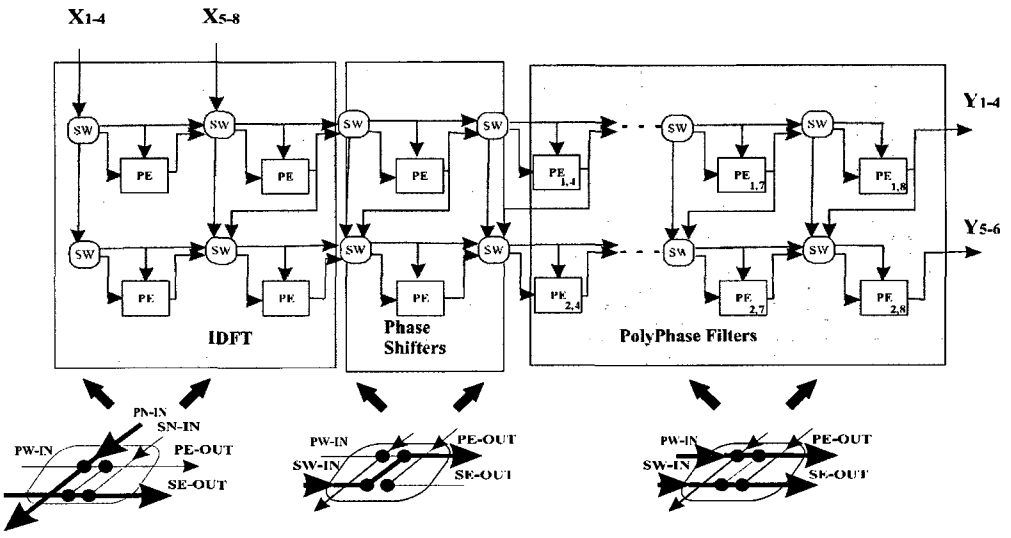


Figure 6.20: RSA and SWs mappings for parallel computation of an 8-channel IDFT-Polyphase.

6.5.2 Sequential configurations

The mapping of the 8-channel Polyphase-DFT circuit for sequential computation could be carried out on the basis of the serial-inputs-parallel outputs model illustrated in previous paragraphs. For the 8-channel Polyphase-DFT circuit mapping for sequential computation, each set of four input sequences is serially shifted into the RSA structure while the outputs are shifted out in a parallel fashion. Thus, two system clock cycles are required to compute all 8 outputs for this circuit configuration. The serial inputs-parallel outputs mapping for the 8-channel Polyphase-DFT circuit together with the mappings of SW cells are shown in Fig. 6.21.

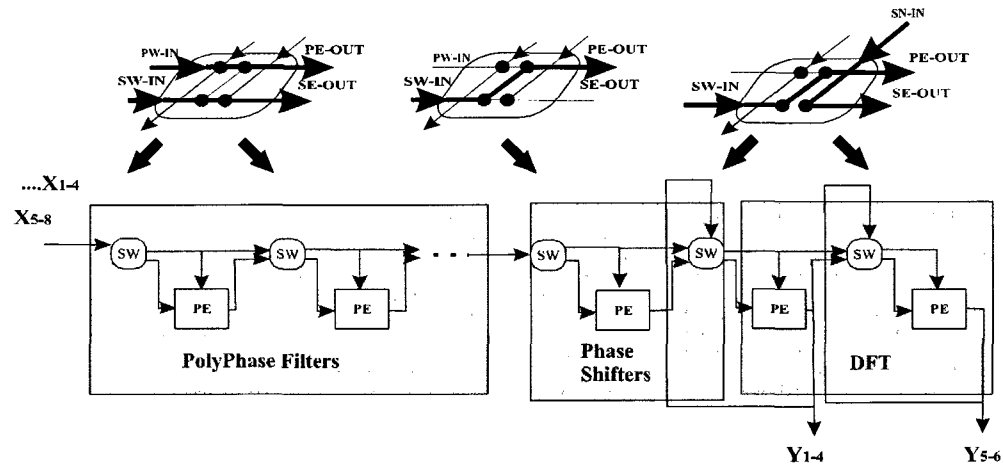


Figure 6.21: RSA mapping for serial inputs-parallel outputs implementation of 8-channel Polyphase-DFT.

The mapping of the 8-channel IDFT-Polyphase circuit for sequential computation could be carried out on the basis of the parallel-inputs-serial outputs mode of operation. For this circuit configuration, all 8 input sequences are shifted into the IDFT building block in parallel. The set of four outputs shifted out from the IDFT building block are processed by the phase shifters and the Polyphase filter building blocks to produce a set of four output sequences of the IDFT-Polyphase circuit. The same set of input sequences is shifted into the RSA circuit a second time to compute the remaining four output sequences of the 8-channel IDFT-Polyphase circuit. Overall, this circuit requires a total of two system clock cycles to compute all 8 outputs for this 8-channel IDFT-Polyphase configuration. The parallel inputs-serial outputs mapping for the 8-channel IDFT-Polyphase circuit together with the mappings of SW cells are shown in Fig. 6.22.

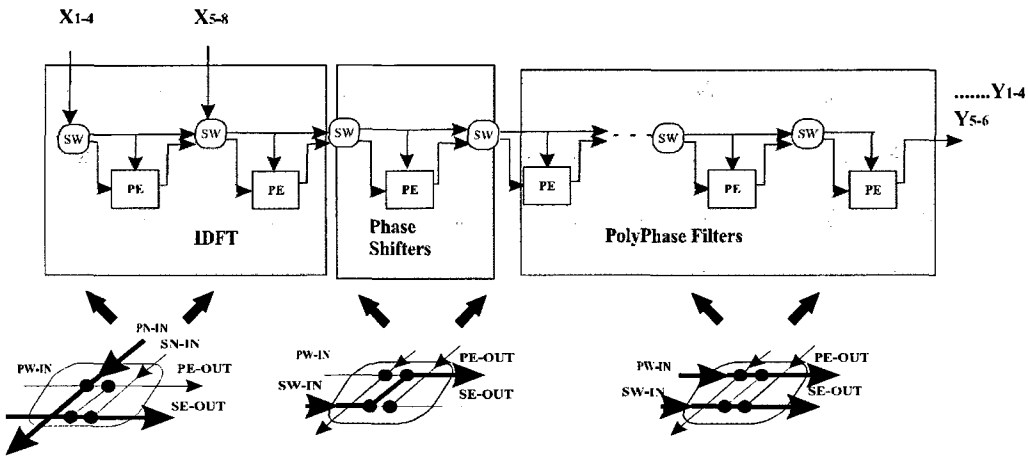


Figure 6.22: RSA mapping for parallel inputs-serial outputs implementation of 8-channel IDFT-Polyphase.

6.5.3 RSA-based Polyphase-DFT/IDFT-Polyphase Circuit Complexity

For the parallel mode configurations of the Polyphase-DFT/IDFT-Polyphase circuits, for which all N outputs are shifted out simultaneously, the logic resources requirements are higher than for the sequential structures. The sequential structure, however, can only shift out data sequences in a time-shared fashion. Thus, several system clock cycles would be required to produce a complete output sequence for a sequential structure. Table 6.3 summarizes the number of PE cells and throughput performance of RSA circuit mappings for N-channel Polyphase-DFT/IDFT-Polyphase circuits for parallel and sequential configurations respectively.

Table 6.3: Logic resources and throughput performance for parallel and sequential mappings of Polyphase-DFT/IDFT-Polyphase circuits.

Mapping Schemes N : # Input Sequences T : # Tap for FIR filters T_{PE} : Processing time of PE cell r: Number of columns in RSA PI-SO: Parallel Inputs-Serial Outputs		**Number of PE Cells**	**Throughput (Samples/s)**
Parallel	$N = 4n$	$\left(\frac{N}{4}\right)\left(\frac{N}{4}+T+1\right)$	$\frac{N}{T_{PE}}$
	$N = 4n+2$	$\left(\frac{N+2}{4}\right)\left(\frac{N}{2}+T+1\right)$	
	$N = 2n+1$	$\left(\frac{N-1}{2}\right)\left(\frac{N-1}{2}+T+1\right)$	
Sequential (PI-SO)	$N = 4n$ $(r = 1,\ldots,\frac{N}{4})$	$r\left(\frac{N}{4}+T+1\right)$	$\frac{4r}{T_{PE}}$
	$N = 4n+2$ $(r = 1,\ldots,\frac{N}{2})$	$r\left(\frac{N+2}{4}+T+1\right)$	$\frac{2r}{T_{PE}}$
	$N = 2n+1$ $(r = 1,\ldots,\frac{N-1}{2})$	$r\left(\frac{N-1}{2}+T+1\right)$	$\frac{2rN}{T_{PE}(N-1)}$

The results shown in Table 6.3 can be used to determine whether parallel or sequential mapping should be used for the configuration of a Polyphase-DFT/IDFT-Polyphase circuit based on system requirements. If circuit throughput performance is critical then a parallel mapping scheme should be employed. Otherwise, RSA circuit mappings based on the sequential approach could be used where lower usage of hardware resources would help to reduce power consumption. For sequential mappings of Polyphase-

DFT/IDFT-Polyphase circuits, the number of rows in the RSA arrays could be determined using Table 6.3. to achieve minimum hardware usage based on system throughput requirements.

Chapter 7

Circuit Implementations on FPGA

7.1 Introduction

In this chapter, FPGA implementations and simulation results of several representative CSE-BitSlice circuits as well as implementations of the RSA circuit and its building blocks are described. The implementations of CSE-BitSlice based circuits including a real multiplier, complex multiplier, FIR filter and DFT are illustrated in Section 7.2. Logic resources and throughput performance extracted from simulation results of the CSE-BitSlice based circuits are compared with results obtained from circuits with the same characteristics implemented using CSA and Radix-4 Booth algorithms.

FPGA implementation and simulation results for the PE cell design based on the CSE-BitSlice techniques are presented in Section 7.3. Overall logic resource and throughput performance of the PE circuit are extracted and compared to a PE cell designed using the CSA and Booth algorithms. Circuit implementation and simulation results for the SW design are also presented in this section. The implementations of the RSA circuit for DFT mappings are presented in the last paragraphs of Section 7.3. These DFT circuits have been configured for parallel or time shared modes of operation. Implementation results of an RSA circuit configured as a 32-point DFT, 4-channel Polyphase filter, 8-channel DFT/IDFT or Polyphase-DFT/IDFT-Polyphase functions are also presented in this section.

Finally, the RSA-based FIR filter and FFT/DFT circuits are considered and compared with representative reconfigurable circuits in Section 7.4.

Two's complement number representation has been used in all designs depicted in this chapter where arithmetic computations are based on fixed-point representations. Hardware implementations have been carried out using Xilinx's 5VLX330FF1760-2 FPGA device [Xilinx].

7.2 CSE-BitSlice Based Circuit Implementations

In this section, circuit implementations for real multiplier, complex multiplier, FIR filter and DFT designs based on the CSE-BitSlice technique are presented. Simulation results, logic resources, and throughput performance of the CSE-BitSlice based circuits are presented and compared with CSA and Radix-4 Booth based implementations.

7.2.1 Real Multiplier

Two real multiplier circuits have been implemented, the first circuit consists of 12 bit inputs and the second circuit consists of 42 bit inputs. The first and the second circuit shift computation results out via 24 and 84 bit data buses respectively. The CSE-BitSlice circuits have been implemented based on 2-Term and 3-Term decomposition approaches. Simulation results for the CSE-BitSlice based 12X12 bit real multiplier designs are shown in Table 7.1. Implementation results for the CSE-BitSlice based 42X42 bit real multiplier designs are shown in Table 7.2

Table 7.1: Simulation results for the CSE-BitSlice based 12X12 bit real multiplier circuit implementations.

Logic Resources/ Throughput	**Number of Terms**	
	N=2	**N=3**
FF	74	73
4-Input LUT	400	460
Throughput (MSPS)	112.3	104

Table 7.2: Simulation results for the CSE-BitSlice based 42X42 bits real multiplier circuit implementations.

Logic Resources/ Throughput	**Number of Terms**	
	N=2	**N=3**
FF	275	261
4-Input LUT	4697	4689
Throughput (MSPS)	48	49.5

As can be seen from Table 7.1, for the 12-bit multiplier circuit, the 2-Term design requires less logic resources and offers better throughput performance than the 3-Term design. Simulation results for the 42-bit multiplier depicted in Table 7.2, however, show that the 3-Term design requires less logic resources and provides better throughput performance than the 2-Term design. Based on equation (5.8) in Chapter 5, the 12-bit multiplier circuit designed using 2-Term and the 3-Term decomposition requires a total of 6 and 7 adders respectively. The lower number of adders explains the higher throughput and lower logic resource requirement for the circuit implementation based on

the 2-Term design. The 2-Term based 42-bit multiplier circuit requires 21 adders while a total of 17 adders are needed by the 3-Term design. Despite the fact that it requires fewer adders than the 2-Term design, the 3-Term design as shown in Table 7.2 does not show significant hardware saving. This is the result of greater logic resource requirements needed to implement the sub expression elimination blocks for the 3-Term design. In fact, the 2-Term design requires a total of 21 2-to-4 decoders as compared to 14 3-to-8 decoders required by the 3-Term design for the common bit elimination process. Simulation results shown in Table 7.1 and Table 7.2 are consistent with hardware usage estimates for real multiplier circuit design using CSE-BitSlice decomposition techniques established in Chapter 5.

Simulation results for the 12-bit and 42-bit real multiplier design implementations based on CSA, Radix-4 Booth, and CSE-BitSlice techniques, incorporating 3-Term decomposition, are shown in Table 7.3

Table 7.3: Comparison of simulation results for the 12X12 bit and 42X42 bit real multiplier circuit implementations based on CSE-BitSlice, CSA, and Radix-4 Booth algorithms.

Logic Resources/ Throughput		**Design Technique**		
		CSA	**Radix-4 Booth**	**CSE-BitSlice N=3**
FF	12X12	77	81	73
	42X42	268	272	261
4-Input LUT	12X12	539	508	460
	42X42	9076	5157	4689
Throughput (MSPS)	12X12	87	125	104
	42X42	27.8	47.6	50

The results in Table 7.3 show that the Radix-4 Booth based design requires more logic resources as compared to the CSE-BitSlice design for the 12X12 and 42X42 bit circuits. For the 12X12 bit circuit, however, the Radix-4 Booth design yields a 17% higher throughput. The higher throughput reflects the fact that the Radix-4 Booth design requires a total of five adders as compared to seven adders used in the 3-Term CSE-BitSlice design. However, the lower number of adders contained in the 12-bit Radix-4 Booth circuit does not translate into lower hardware usage as compared to the CSE-BitSlice design due to large logic resources requirements of the 3-to-8 decoders in the former circuit. For the 42-bit circuit, the greater logic resource and lower throughput indicated for the Radix-4 Booth design are consistent with the larger number of adders used as compared to the CSE-BitSlice design. The CSA based designs have the highest hardware requirements and offer the lowest throughput performances as compared to the

Radix-4 and CSE-BitSlice based designs for both multipliers. This outcome is the result of the large number of adders in the CSA-based implementations. Indeed, the number of partial products in the CSA design is nearly twice that of the other two designs. The outcomes of this simulation are consistent with the estimations established in Chapter 5.

7.2.2 Complex Multiplier

The multiplication of two complex numbers $X_r + jX_i$ and $Y_r + jY_i$ is expressed as:

$$Z_r + jZ_i = (X_rY_r - X_iY_i) + j(X_rY_i + X_iY_r) \tag{7.1}$$

where the r and i subscripts denote real and imaginary components of the complex numbers X and Y respectively. The complex multiplication described in (7.1) could be rewritten in vector-matrix multiplication format as follows:

$$[Z_r \quad Z_i] = [X_r \quad X_i] * \begin{bmatrix} Y_r & Y_i \\ -Y_i & Y_r \end{bmatrix} \tag{7.2}$$

Since the same vector X is multiplied by two different vectors in matrix Y, the CSE-BitSlice optimization technique for vector-matrix multiplication described in Chapter 5 can be used in this case.

The decomposed expression for the real and imaginary components of the output Z is as follows:

$$Z_r = \left(X_r \bullet \overline{X_i} * (Y_r)\right) + \left((X_r \bullet X_i) * (Y_r - Y_i)\right) + X_i \bullet \overline{X_r} * (-Y_i) \tag{7.3}$$

$$Z_i = \left(X_r \bullet \overline{X_i} * (Y_i)\right) + \left((X_r \bullet X_i) * (Y_r + Y_i)\right) + X_i \bullet \overline{X_r} * (Y_r) \tag{7.4}$$

As can be seen from (7.3) and (7.4), the total number of additions required for the computation of this complex multiplication is equal to 2B where B is the word length of the multiplier operands X_r and X_i.

For the complex multiplier circuit implementations, the word length of the real and quadrature components of the multiplier and multiplicand operands is 21 bits. The real and quadrature components of the multiplier output are each 43 bits. Simulation results for the 42X42 bit complex multiplier design implementations based on CSA, Radix-4 Booth, and CSE-BitSlice techniques are shown in Table 7.4 for comparison.

Table 7.4: Comparison of simulation results for the 42X42 bit complex multiplier implementations based on CSE-BitSlice, CSA, and Radix-4 Booth algorithms.

Logic Resources/ Throughput	**Design Technique**		
	CSA	**Radix-4 Booth**	**CSE-BitSlice**
FF	297	313	260
4-Input LUT	6933	5292	4247
Throughput (MSPS)	33.9	50	54

Simulation results depicted in Table 7.4 show that the CSE-Bitslice based circuit requires the least hardware resources and offers the highest throughput performance as compared to circuit implementations based on Radix-4 and CSA techniques. Even though the Radix-4 Booth based design requires the same number of adders, two of its adders are larger than the ones used by the CSE-BitSlice based design, which results in a more complex circuit. In fact, the CSE-Bitslice design uses 2 adders of 22 bits to sum up the multiplicand operands and 40 adders of 43 bits for the add-and-shift operations. On the other hand, all 42 adders used in the Radix-4 Booth are 43 bits.

The CSA based design is the most complex in term of hardware usage and offers the lowest throughput performance among the three designs due to the large number of adders required for circuit implementation. The outcomes of this simulation are consistent with the estimates established in Chapter 5.

7.2.3 6-Tap FIR filter

A T-tap FIR filter executes T real multiplications and T-1 accumulations as expressed in the equation bellow:

$$y(n) = \sum_{i=0}^{T-1} a_i x(n-i) = A^T X \tag{7.5}$$

where

$$A = [a_0, a_1, ..., a_{T-1}]$$

$$X = [x_n, x_{n-1}, ..., x_{n-T+1}]$$

The vectors X and A represent the filter input samples and the filter coefficients respectively.

The filter output is generated by the inner product of the two vectors X and A, each of which consist of T elements. Using the design technique for vector multiplication described in Chapter 5, the output of the T-tap FIR filter can be calculated based on a combination of 2 or more terms. As established in Chapter 5, a total of $\frac{B}{T}+2^T-T-2$ additions is required if a T-Term decomposition is used as compared to $\frac{T}{2}(B+1)-1$ additions based on a 2-Term decomposition computation. The block diagram of a T-tap FIR filter circuit design using the CSE-BitSlice technique based on 2-Term decomposition is shown in Fig. 7.1.

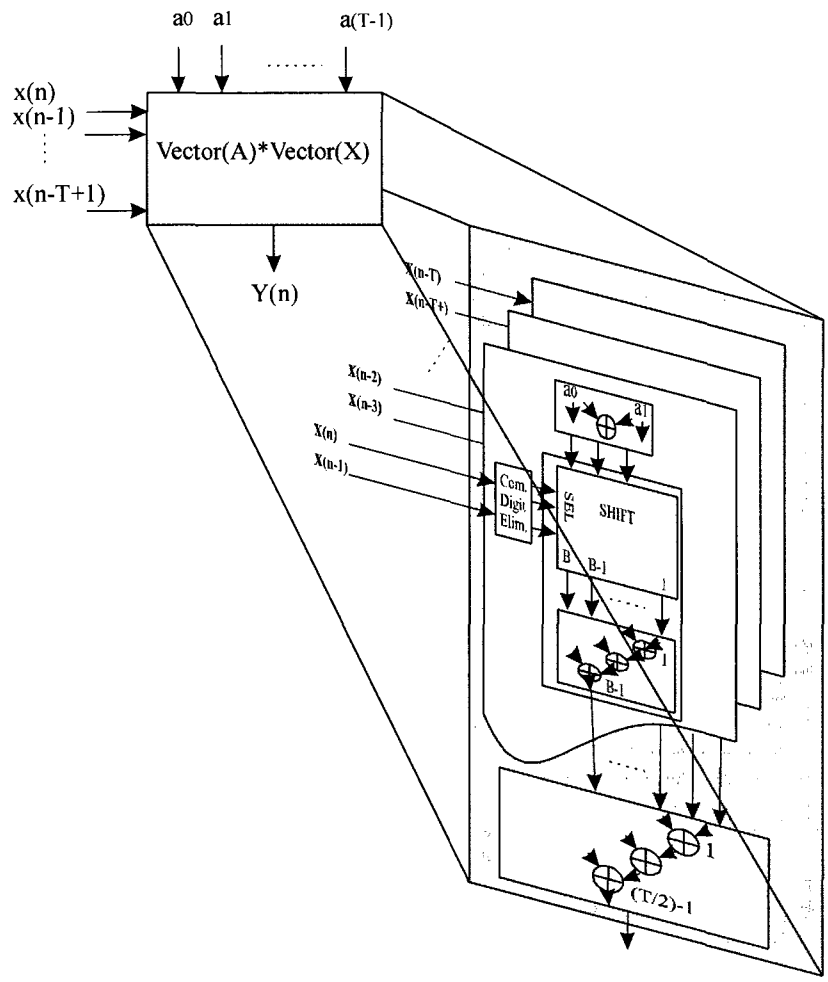


Figure 7.1: T-Tap FIR filter circuit implementation based on the CSE-BitSlice technique for vector multiplication.

A 6-tap direct architecture FIR filter based on the vector multiplication technique presented in Chapter 5 has been implemented. Two implementations of the 6-tap FIR filter have been carried out with coefficients and the filter inputs represented by 20-bit words in real format. The first FIR circuit has been designed using 2-Term and the second circuit has been designed using the 3-Term design technique for comparison. The 3-Term based circuit implementation has been carried out based on sums of two groups,

each of which consists of the sum of three products generated by the multiplication of the filter input and the filter coefficients. For the 2-Term based circuit implementation, the number of terms in each group is reduced to two thus the total number of groups is equal to three. The filter output is represented by 23-bit words in real format. The grouping of the products of the filter inputs and filter coefficients for the 2-Term and the 3-Term designs can be readily identified by rewriting the filter convolution equation as follows:

$$y(n) = \sum_{i=0}^{1} a_i x(n-i) + \sum_{i=2}^{3} a_i x(n-i) + \sum_{i=4}^{5} a_i x(n-i) \tag{7.6}$$

$$y(n) = \sum_{i=0}^{2} a_i x(n-i) + \sum_{i=3}^{5} a_i x(n-i) \tag{7.7}$$

Simulation results in terms of performance and hardware usage of the two CSE-BitSlice implementations for the 6-tap FIR filter are shown in Table 7.5.

Table 7.5: Implementation and simulation results for the 6-Tap FIR filter circuits designed using CSE-BitSlice technique for 2-Term and 3-Term decomposition.

Logic Resources/ Throughput	**Number of Terms**	
	N=2	**N=3**
FF	296	295
4-Input LUT	6592	5400
Throughput (MSPS)	28.5	34

Implementation results presented in Table 7.5 show that the overall logic resource requirement for the 6-tap FIR filter implementation using 3-Term decomposition is smaller than for the 2-Term circuit. This is consistent with the estimate established in

Chapter 5 since the 3-Term decomposition requires a lower number of adders than the 2-Term decomposition. The circuit based on 3-Term decomposition also shows superior throughput performance since fewer logic levels are involved in the addition process due to the combination of more variables (three) in the common expression process as opposed to two variables in the 2-Term based designs. Table 7.6 shows the throughput performance and hardware requirements for the 20-bit input 6-tap FIR filter design implementations based on CSA, Radix-4, and CSE-BitSlice techniques.

Table 7.6: Implementation and simulation results for the 6-Tap FIR filter designs based on CSA, Radix-4 Booth, and CSE-BitSlice techniques.

Logic Resources/ Throughput	**Design Technique**		
	CSA	**Radix-4 Booth**	**CSE-BitSlice N=3**
FF	296	293	295
4-Input LUT	13629	8216	5400
Throughput (MSPS)	17.4	26	34

Simulation results presented in Table 7.6 show that the CSE-BitSlice based design offers better performances and requires fewer logic resources for circuit implementation as compared to the CSA and Radix-4 Booth designs. The higher complexity of the CSA based design results from the fact that the main purpose of the CSA algorithm is to shorten the carry data path to improve circuit throughput performance and not to target hardware reduction. The Radix-4 Booth circuit is more complex than the CSE-BitSlice circuit due to an overall larger number of partial products that need to be combined in the multiplication process. In fact, for a FIR filter with N taps and B bit coefficients, the total

number of partial products generated by the Radix-4 Booth design is $\frac{NB}{2}$ whereas the 3-Term CSE-BitSlice based design generates only $\frac{NB}{3}$ partial products. As a result, for the 6 tap filter design with 20 bit coefficients, a total of 60 partial products are generated by the Radix-4 Booth circuit as compared to 40 partial products generated by the 3-Term CSE-BitSlice circuit.

7.2.4 5-Point DFT

The N-point discrete Fourier transform is expressed by the equation below:

$$Y(k) = \sum_{n=0}^{N-1} x(n) * a_{n,k} \quad k = 0,\ldots,N-1 \tag{7.8}$$

The variables $x(n)$ and $a_{n,k}$ are both represented in complex format. The DFT equation could be rewritten in vector-matrix format as follows:

$$\begin{aligned}[Y_0 \;\; Y_1 \;\; \ldots \;\; Y_{N-1}] &= X^T A \\ &= [x_0 \;\; x_1 \;\; \ldots \;\; x_{N-1}] * \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \cdot & \cdot & \cdots & \cdot \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{bmatrix}\end{aligned} \tag{7.9}$$

Since each complex input in the vector X is multiplied by a complex coefficient in matrix A, the CSE-BitSlice design technique for the multiplication of two complex variables can be used to design this N-point DFT circuit. From the implementation results for the complex multiplier circuit shown in section 7.2.2, each complex multiplication of a variable $x(n)$ and a coefficient $a(n,k)$ requires a total of $2B$ additions based on the 2-Term decomposition technique. The value B is the word length of the real and quadrature components of the complex variable $x(n)$. As a result, a total of $2NB$ additions are

required to compute each output of the vector-matrix multiplication in (7.9). The same design process is used to generate the other N-1 outputs of the DFT design. Overall, a total of $2BN^2$ real addition operations are needed to compute an N-point DFT. Since the real and quadrature components of the complex variable $x(n)$ are used as common operands for the 2-Term decomposition, the N partial products generated by the multiplication of the variables $x(n)$ and the N coefficients of a row of matrix $a_{n,k}$ could be computed at the same time. By doing so, the logic resources required for the common bit elimination design can be further reduced. Figure 7.2 represents the block diagram of the N-point DFT design implementation based on the 2-Term CSE-BitSlice decomposition technique.

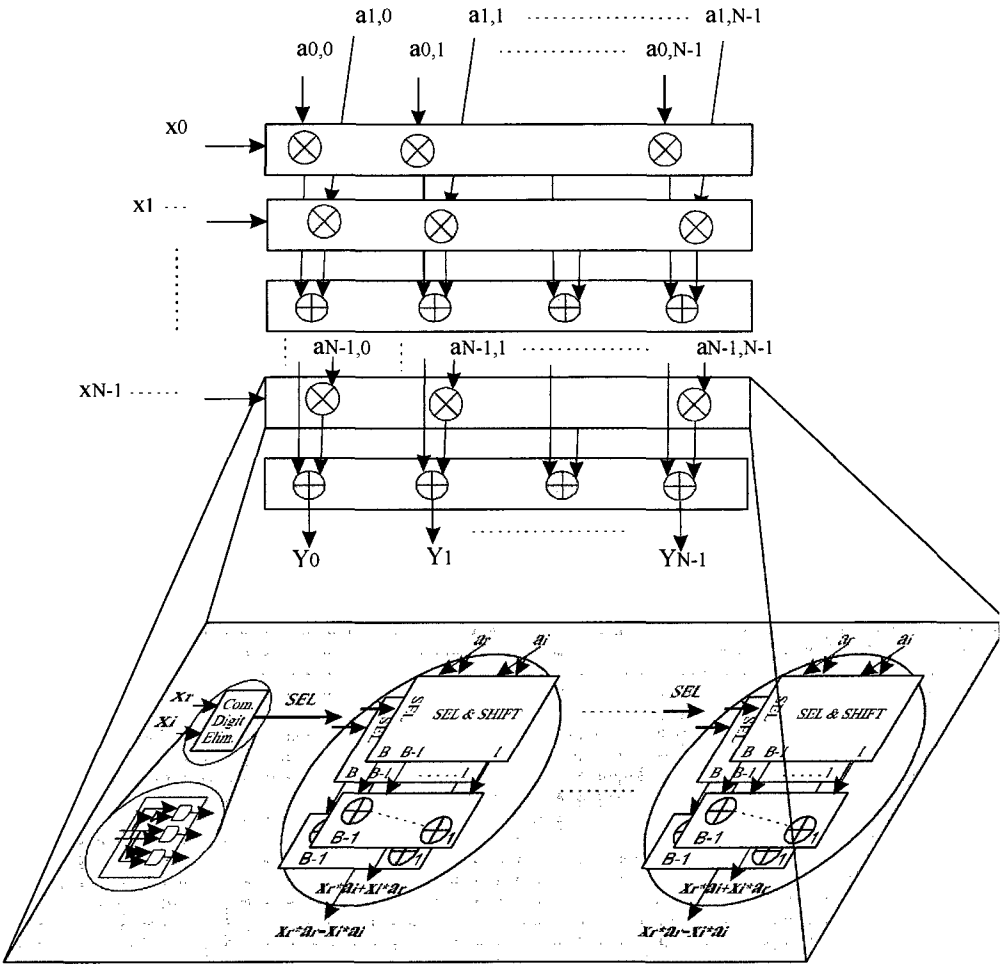


Figure 7.2: Block diagram of N-point DFT design based on 2-Term decomposition technique.

A 5-point DFT circuit design based on the CSE-BitSlice technique has been implemented with each real and quadrature component of the data inputs and coefficients are represented by 21 bit format. The circuit design is based on the serial inputs-parallel outputs scheme where the input sequence is shifted in and processed at every clock cycle and all 5 outputs are shifted out every 5 clock cycles. Each real and quadrature output component is represented by a 25 bit word. Figure 7.3 depicts the block diagram of the 5-

point serial inputs-parallel outputs DFT circuit design based on 2-Term CSE-Bitslice technique.

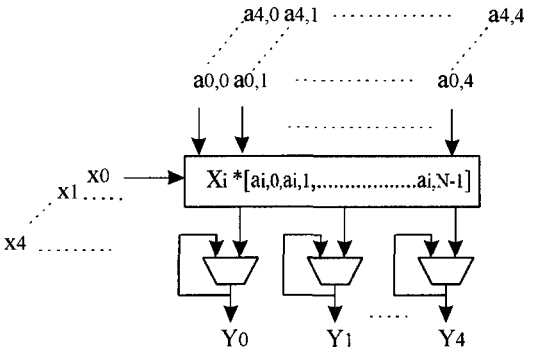


Figure 7.3: Block diagram of the serial inputs-parallel outputs 5-point DFT circuit design.

Simulation results for the 5-point DFT design based on 2-Term decomposition technique are shown in Table 7.7.

Table 7.7: Implementation and simulation results for the 5-point DFT design based on the CSE-BitSlice technique for 2-Term decomposition.

Logic Resources/ Throughput	**Number of Terms N=2**
FF	740
4-Input LUT	19644
Throughput (MSPS)	43.8

For comparison purpose, implementations of the serial inputs-parallel outputs 5-point DFT circuit described in Fig. 7.3 have also been carried out using CSA and Radix-4

Booth algorithms. Table 7.8 shows simulation results for the 5-point DFT circuit designs based on CSA, Radix-4 Booth, and CSE-BitSlice design techniques.

Table 7.8: Implementation and simulation results for the 5-point DFT designs based on different techniques.

Logic Resources/ Throughput	**Design Technique**		
	CSA	**Radix-4 Booth**	**CSE-BitSlice N=2**
FF	964	832	740
4-Input LUT	33700	25929	19644
Throughput (MSPS)	29.3	19	43.8

Implementation results for the 5-point DFT designs illustrated in Table 7.8 show that the CSE-BitSlice based circuit is the most compact among the three 5-point DFT designs. Even though the Radix-4 Booth design and the CSE-BitSlice design require the same number of adders to combine the partial products, some of the adders used in the latter design are smaller in size. As a consequence, the CSE-Bitslice circuit also offers the highest throughput since its critical path length is shorter as compared to the Radix-4 Booth design. The CSA based circuit is the most complex in terms of hardware usage due to the large number of partial products that need to be combined.

7.3 RSA Circuit Implementations

In this section, implementations of the RSA circuit and its building blocks based on CSE-BitSlice technique are described. Throughput performance and hardware requirements of the PE cell circuit implementation is extracted from simulation results and compared with circuits implementations based on the CSA and Radix-4 Booth algorithms.

7.3.1 The Processing Element Circuit Implementation

The PE circuit implementation is based on the architecture depicted in Fig. 4.2 in Chapter 4. Real coefficients are represented by 9-bit words while the complex coefficients required by the DFT circuit consist of 18 bit words with 9 bit real and 9 bit quadrature. Each input data bus of the PE cell is 24 bits wide and each output cell output is represented by 34 bits, all in complex format. The four real multipliers in the AU have been implemented on the basis of different techniques for comparison purpose. The first implementation of the PE circuit is based on the CSE-BitSlice techniques based on 2-Term and 3-Term decomposition. Simulation results for the CSE-based circuit designs are shown in Table 7.9. Simulation results presented in Table 7.9 show that the 2-Term based circuit offers better throughput performance as compared to the 3-Term based circuit. This outcome is the result of the greater logic resource requirements for the design of the 3-Term based circuit. In fact, simulation results in Table 7.9 indicate that the 3-Term based design requires a larger number of look-up tables for circuit implementation. The reason for the difference in hardware requirements between these two designs is the 3-to-8 decoders used by the 3-Term common expression elimination circuitry. The elimination process in the 2-Term design on the other hand uses 2-to-4 decoders, which results in a more compact circuit as compared to the 3-Term design.

Table 7.9: Implementation and simulation results for the PE circuit design based on CSE-BitSlice technique for 2-Term and 3-Term decomposition.

Logic Resources/ Throughput	**Number of Terms**	
	N=2	**N=3**
FF	413	417
4-Input LUT	3431	4177
Gate Count	35843	42682
Throughput (MSPS)	220	190

Circuit implementations for the PE cell using CSA and Radix-4 Booth techniques have also been carried out. Table 7.10 shows logic resources and throughput performance of the PE circuit implementations using CSA, Radix-4 Booth, and CSE-BitSlice techniques.

As shown in Table 7.10, compared to the Radix-4 Booth and the CSA based circuits, the 2-Term CSE-BitSlice circuit is the least complex based on the gate count obtained from simulation results. The Radix-4 Booth and the 2-Term CSE-BitSlice designs offer comparable throughput even though the number of gates required by the Radix-4 Booth circuit is larger than the gate count required by the CSE-BitSlice circuit. Although both designs use almost the same number of adders in the multiplication process, the greater logic resource requirement of the Radix-4 Booth circuit is the direct result of the 3-to-8 decoders employed in the decoding process. These decoders require more LUTs for circuit implementation as compared to the 2-to-4 decoders used by the 2-Term CSE-BitSlice circuit. The CSA based circuit has a greater gate count and offers the lowest throughput performance of the three designs due to the greater number of adders used in the multiplication process.

Table 7.10: Implementation and simulation results for the PE circuit design based on different design techniques.

Logic Resources/ Throughput	Design Technique		
	CSA	Radix-4 Booth	CSE-BitSlice N=2
FF	413	459	413
4-Input LUT	4351	3907	3431
Gate Count	42269	39550	35843
Throughput (MSPS)	174	210	220

7.3.2 The Switch Circuit Implementation

The input interfaces of the SW circuit include four input data busses consisting of two global and two local input busses. The SW shifts its output sequences via two global and one local output data bus. Each global and local data bus consists of four independent data signals. Each signal in the global input and output data busses is represented by 24 bits, whereas the dynamic range of signals in the local input and output data busses is represented by 34 bits, all in complex format. Logic resources and clock performance of the SW circuit implementation are shown in Table 7.11.

Table 7.11: Clock performance and hardware requirements for the SW circuit implementation.

Logic Resources		Clock Rate (MHz)
FF	385	715
4-Input LUT	727	

7.3.3 The Reconfigurable Systolic Array Circuit Implementations

In this section, simulation results for several RSA circuit implementations are presented. For RSA circuit implementations presented in this section, the input and output data are represented by 24 and 34 bit-words respectively, all in complex format. The coefficients are represented by 9 bit words in real format. The RSA circuits presented in this section have been implemented using PE cells designed with the 2-Term CSE-BitSlice technique.

In support of DFT circuit implementations configured for parallel computations, four RSA structures have been implemented for 9, 10, 12 and 16-point DFTs based on the mapping techniques described in Chapter 6. The RSA structure for the 10-point DFT mapping consists of an array of 3-by-5 PE and SW cells. The RSA structures for 9, 12 and 16-point DFTs consist of arrays of 4-by-4, 3-by-3 and 4-by-4 PEs and supporting SW cells respectively. The RSA structures for the 9, 10, 12 and 16-point DFT circuit mappings are depicted in the block diagrams of Fig. 7.4. For representation simplicity, the SW cells have been omitted in the block diagrams of the RSA circuits discussed in this section.

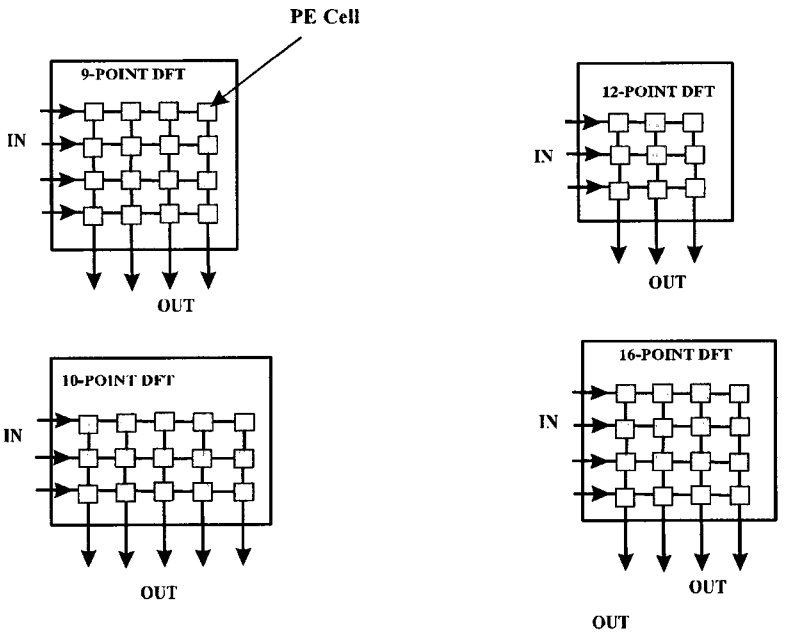


Figure 7.4: RSA structures for parallel mapping of 9, 10, 12 and 16-point DFTs.

Simulation results for four RSA circuits configured for parallel computation of 9, 10, 12 and 16-point DFTs are shown in Table 7.12.

Table 7.12: Throughput and hardware requirements of RSA circuits implemented for parallel computation of 9, 10, 12 and 16-point DFTs.

N	Logic Resources		Throughput (DFT/s)
	FF	4-Input LUT	
9	15706	61821	55M
10	14717	58031	
12	8612	28383	
16	15706	61821	

For sequential computations, the RSA structures described in Fig. 7. 4 can be used to compute a DFT with greater number of points. Investigation of sequential configuration by means of a 4-by-4 RSA circuit implemented for parallel computation of 16-point DFT confirmed that it could be used for sequential computations of 64 and 256-point DFTs. The sequential computation of 64-point DFT based on the 4-by-4 RSA structure supported a throughput of 3.4M DFTs per second while the computation of a 256-point DFT was demonstrated at the rate of 0.21M DFTs per second. This 4-by-4 RSA circuit could be used for example in the implementation of WLAN or fixed WiMAX transceivers for which data throughput and OFDM symbol processing time of $\sim 4\mu$ and $\sim 72\mu$ seconds corresponds to the IEEE 802.11 and 802.16 standards respectively [802.11b, 802.16a].

The 4-by-4 structure can also shift out the complete set of 512 and 1024 point DFTs after 19μ, and 75μ seconds respectively. The RSA circuit performance satisfies the processing time of 62μ, and 91.4μ seconds which corresponds to the OFDM transmission schemes involving 512 and 1024-point FFTs proposed for DAB [ETSI'06] and WiMAX [802.16a] applications respectively. For DAB and DVB applications where FFT in the

transceivers needs to be able to work in 2K, 4K or 8K modes, an RSA structure based on multiple instances of 4-by-4 arrays connected to one another could be employed to configure the required transmission mode. For example, an 8-by-8 RSA circuit that computes 2048 DFT every 74.5μ seconds is suitable for WiMAX, DAB and DVB [ETSI'04] applications where the proposed processing time for OFDM symbol of 2K point transmission mode is 91.4μ, 248μ, and 91.4μ seconds respectively.

To investigate system configuration, a representative RSA circuit consisting of an array of 2-by-8 PE and SW cells has been implemented for parallel or sequential computation of different DSP functions. This RSA circuit consists of two rows and eight columns of PE and SW cells with a total of 16 PE and 16 SW cells. This circuit supports configurations of DFT, Polyphase filters, Polyphase-DFT and IDFT-Polyphase respectively. For DFT configuration, a serial inputs-parallel outputs 32-point DFT design is mapped onto this RSA structure and throughput performance is simulated. For parallel configurations, a 4-chanel Polyphase filter, 8-channel Polyphase-DFT, and 8-channel IDFT-Polyphase designs have been mapped onto this 2-by-8 RSA structure. Simulation results for throughput performance of these parallel configured circuits have been determined. The mapping of the 2-by-8 RSA structure for 32-point DFT, 4-channel Polyphase filter and 8-channel Polyphase-DFT and 8-channel IDFT-Polyphase circuits are depicted in the block diagrams of Fig. 7.5.

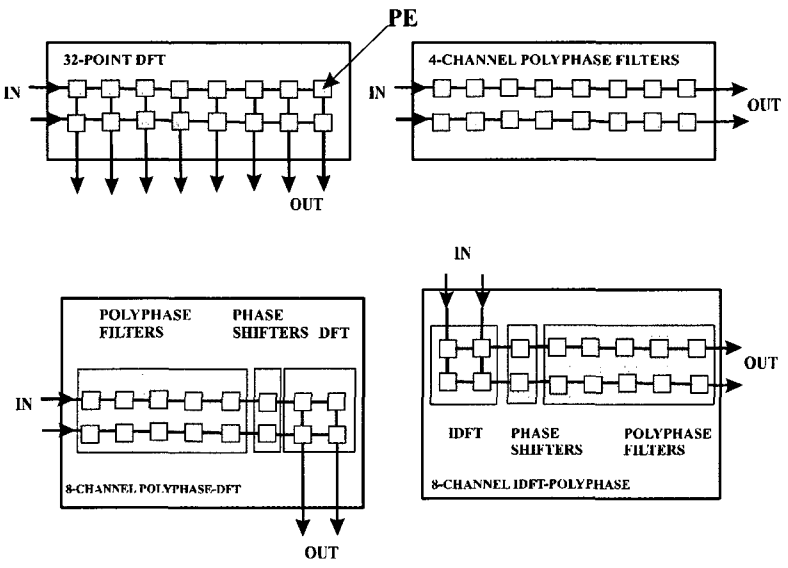


Figure 7.5: Circuit mappings for 32-point DFT, 4-channel Polyphase filter, 8-channel Polyphase-DFT and 8-channel IDFT-Polyphase designs on a 2-by-8 RSA structure.

Simulation results for the 2-by-8 RSA circuit show that it can operate at a frequency of 55 MHz. This result demonstrates that the RSA circuit throughput is limited by the performance of the PE cell together with the delay due to the interconnects between the cells in the RSA structure since the SW can operate at a clock rate of 715 MHz. Thus, for the 32-point DFT circuit configured in a serial input-parallel output mode, a complete set of outputs can be shifted out every 72 nano seconds. For the 4-channel Polyphase filter configuration, the circuit offers a throughput performance of 220 MSPS. For the 8-channel IDFT-Polyphase and 8-channel Polyphase-DFT configurations, simulation results show the RSA circuit supports maximum throughput of 110 MSPS in both cases. Table 7.13 depicts logic resource requirements for the 2-by-8 RSA circuit implementation. Throughput performance results for the RSA circuit configured for 32-point DFT, 4-channel Polyphase filter, 8-channel IDFT-Polyphase and 8-channel Polyphase-DFT are presented in Table 7.14.

Table 7.13: Clock rate and hardware requirements for the 2-by-8 RSA circuit implementation.

Logic Resources		Clock Rate (MHz)
FF	15722	55
4-Input LUT	61363	
Gate Count	689950	

Table 7.14: Throughput performance results for 32-point DFT, 4-channel Polyphase filter, 8-channel IDFT-Polyphase and 8-channel Polyphase-DFT circuit configurations.

RSA Circuit Mappings	Throughput
32-point DFT	13.8M DFT/s
4-channel Polyphase Filter	220 MSPS
8-channel IDFT-Polyphase	110 MSPS
8-channel Polyphase-DFT	110 MSPS

7.4 Flexibility and Performance Comparisons of Reconfigurable Architectures

In this section, comparisons of configuration flexibility and computation performance of the RSA architecture and existing configurable architectures are presented. The existing reconfigurable architectures used in this comparison are the DRAW, the RADComm and the CSoC architectures proposed in [Also'02], [Gray'00] and [Wall'05] respectively. The features of interest are the circuit's architecture, ability to

realize complex functions, support real time reconfigurability, scalability, and performance.

Upon consideration of the different architectures we note that all architectures support FIR filter implementations, however, not all support FFT or DFT circuits. Indeed the DRAW [Also'02] circuit does not support either DFTs or FFTs, and the CSoC [Wall'05] supports only FFTs while the RADComm [Gray'00] circuit supports DFT implementations based on the Goertzel algorithm. As previously indicated the FFT realization restricts unduly the dimensionality of the transform to powers of two. Furthermore, the CSoC [Wall'05] circuit with 32 reconfigurable PE cells requires a total of 100 clock cycles to compute a 64-point FFT. Thus in terms of the ability to support both FIR filters and DFT the RSA and RADComm [Gray'00] circuits meet some of the previously identified requirements. However, the RADComm [Gray'00] circuit was not designed for real time reconfiguration as its control data and coefficients must be shifted in serially during start up and reconfiguration can only be initiated by circuit reset.

To address the issue of reconfigurability in real time, the RSA was designed to support the routing of configuration and coefficient data in real time. The DRAW [Also'02] architecture also supports real-time configurations since signal connections and modes of operation of the circuit are controlled by a network of individually reconfigurable switches and processing elements. Reconfigurations on a CSoC [Wall'05] structure on the other hand would take many clock cycles and involve manual partitioning of the new function into small tasks and mapping of each task onto the circuit.

For DFT mappings, the RSA structure can operate in sequential or parallel modes depending upon throughput performance required by the system and the circuit logic resources available. If the target application requires high throughput performance, DFT design can be mapped onto an RSA structure configured for parallel computation. If the

system hardware resources are limited, sequential mappings could be employed instead. On the other hand, the CSoC [Wall'05] architecture does not support circuit expansion.

Chapter 8

Conclusions and Recommendations

In this chapter, the major contributions of this thesis are outlined in the first section. A list of published and submitted papers is presented in the second section where the contributions of each paper to the specific research field are identified. Possible future work on the RSA architecture is discussed in the last section of this chapter.

8.1 Conclusions

A reconfigurable system based on systolic array architecture, suitable for a class of runtime MC applications is presented in this thesis. Design issues of runtime reconfiguration of systems and characteristics necessary for runtime applications, for which the proposed architecture is targeted, have been discussed. Furthermore, a review of representative reconfigurable hardware and architectures as well as an assessment of their advantages and disadvantages for the target class of applications has been carried out. The RSA architecture proposed in this thesis consists of 2D arrays of identical processing elements connected together via switching elements. The RSA can be configured to implement such DSP functions as FIR/Polyphase filters, DFT/IDFT and Polyphase-DFT/IDFT-Polyphase. The PEs, which perform arithmetic operations on vector data, are dynamically reconfigurable. The mapping of the targeted DSP functions on RSA circuits has also been described. Design methodology, circuit implementation and circuit performance of the RSA have been presented.

The major contributions of this thesis can be summarized as follows:

1. A novel reconfigurable systolic array architecture that supports dynamic reconfigurations for muticarrier wireless and multirate applications has been proposed and verified by means of circuit implementations executed on an FPGA. The RSA circuit can be dynamically configured to support mappings of a set of DSP functions including FIR filters, DFT/IDFT, bank of polyphase filters, phase shifters, Polyphase-DFT and IDFT-Polyphase circuits. The RSA circuit is a flexible and low hardware complexity platform for MC applications where low cost and high performance are essential. Currently, there is no reconfigurable architecture available that supports runtime reconfigurations of FIR filters, DFT and Polyphase-DFT functions.

2. The RSA architecture's modular and regular characteristics allow circuit expansion to be carried out seamlessly. For large data path applications, circuit expansion can be achieved simply by replicating arrays of PE and SW cells without incurring extra design cost. The homogenous nature of the RSA architecture also makes scalable computation of data intensive applications possible based on use of shared hardware resources with view to minimizing the hardware and optimizing performance. Reconfigurable architectures for multicarrier applications where interconnections between the cells in the array are not local [Wall'05, Gray'00] or the array consists of two or more types of processing elements [Wall'05], to implement a circuit expansion or require typically major circuit modifications.

3. An RSA based circuit implementation for low complexity N-point DFTs where N is not restricted to be a power of two or a composite number has been proposed. An RSA circuit consisting of $N^2/16$ PE cells can perform N-point DFTs with a

throughput of $1/t_{PE}$ DFTs per second for N a multiple of four where t_{PE} is the input to output delay of the PE cell. If N is even but not a multiple of four or if N is odd, the RSA circuit must consist of $\frac{N}{2}\left(\frac{N+2}{4}\right)$ and $\left(\frac{N-1}{2}\right)^2$ PE cells respectively. Using this mapping technique the RSA circuit can be configured for N-point DFT computations where N is any integer. The regularity of the RSA architecture combined with the flexibility of the DFT mapping approach enables the RSA circuit to compute N-point DFTs in parallel or in recursive fashion. These characteristics allow an RSA circuit designed for N-point DFT computation to support M-point DFT computations where $M > N$. Currently, no reconfigurable architecture is available that supports runtime reconfigurations of DFT in parallel or in recursive modes of operation, where the total number of complex multiplications required is smaller than N^2.

4. A novel algorithm and design approach based on a combination of CSE and Bit-Slice techniques targeting the multiplication of variables has been developed to achieve circuit implementations with low hardware complexity. The proposed algorithm addresses the issues of logic resource reduction and has been demonstrated and verified for the following types of circuits:

   - Multiplication of two numbers
   - Multiplication of a number by a vector of variables
   - Multiplication of two vectors
   - Multiplication of two matrices

   Depending upon the word length of the multiplier operand, multiplier circuit implementations based on this algorithm have shown that significant hardware reduction can be achieved as compared to circuits using CSA and Radix-4 Booth algorithms. Simulations have also shown that the CSE-BitSlice based circuits

perform better than circuits designed using CSA and Radix-4 Booth techniques. Implementation results of several representative multiplier circuits on both an FPGA (Chapter 7) and ASIC (Appendix B) demonstrate the advantage of the CSE-BitSlice technique as compared to the CSA and Radix-4 Booth techniques in terms of hardware realization and high throughput performance.

8.2 Publications on Dissertation Research

1. H. Ho, V. Szwarc and T. Kwasniewski, "Design and FPGA Implementation of a MultiCarrier Baseband Processor," Proceedings of the 3rd IASTED Int'l Conf. on Circuits, Signals and Systems CSS'03, May 2003.

   Paper details a full duplex multicarrier baseband processor design based on Polyphase-FFT algorithm and implemented on a Xilinx Vertex II Pro FPGA device. This modem is designed to support modulation and demodulation for a group of up to 8 subcarriers with equal bandwidth channels of T1 data rates. Simulation results show that the 8-channel full duplex multicarrier circuit can support a data throughput of up to 71Mbps. As a result, the circuit can be reconfigured to support channel data rates of up to 5T1. The 8-channel multicarrier modem design was validated and its performance was verified against functional models developed using SystemView.

   First reported single chip realization of a T1 rate multicarrier baseband processor based on the Polyphase-FFT architecture.

2. H. Ho, V. Szwarc and T. Kwasniewski, "Design and Implementation of a Reconfigurable Polyphase-FFT System," Proceedings of the 4th IASTED Int'l Conf. on Circuits, Signals and Systems CSS'04, Oct. 2004.

In this paper, the design and implementation of a reconfigurable high throughout Polyphase-FFT macrocell on a Xilinx Virtex II xc2vp100ff1704-6 device is presented. The circuit can be configured to operate as a Group Multiplexer and/or Demultiplexer and its versatility allows it to support half and full-duplex modes of operation. Furthermore, the reconfigurable system also provides the flexibility of dynamic reconfiguration for the number of channels as well as bandwidth of channels. Logic resource requirements and performance for the reconfigurable Polyphase-FFT circuit have been determined and presented in the paper. These simulation results were used and compared against the results extracted from simulations of the fixed circuit designs. Comparison results indicate that the tradeoff for the reconfigurability is an increased gate count and lower throughput of the reconfigurable circuit as compared to the non-reconfigurable circuit designs.

First reported SoC implementation of a reconfigurable Group Multiplexer and Demultiplexer based on the Polyphase-FFT design.

3. H. Ho, V. Szwarc and T. Kwasniewski, "Hardware Optimization of a Configurable Polyphase-FFT Design Using Common Sub-Expression Elimination," Proceedings of the IEEE MWSCAS/NEWCAS '07 Conf., Montreal, Aug 5-8. 2007.

In this paper, a combination of CSE and bit-slice techniques have been applied to the multiplication of two variables and its effectiveness demonstrated in the design of a reconfigurable Polyphase-FFT system. The Polyphase-FFT circuit can be reconfigured to support 8, 16 or 32 channels and each FIR filter in the Polyphase filter can be configured to have up to 15 taps. The CSE-BitSlice

algorithm has been used to reduce the number of partial products for multiplication operations in FIR filter, phase shifter, and FFT circuits. Real and complex multiplications in these building blocks have been transformed into sums of two products (STP) based on the bit slice decomposition approach with view to detecting and eliminating digits shared by both terms.

Simulation results performed on the FIR filter, the Phase Shifter, and the Complex Butterfly circuits show significant logic resource reductions for the CSE-BitSlice based circuits as compared to conventional designs.

First reported realization of generic multipliers and complex DSP functions based on a combination of CSE and bit-slice techniques.

4. H. Ho, V. Szwarc and T. Kwasniewski, "Low Complexity Reconfigurable DSP Circuit Implementations Based on Common Sub-Expression Elimination," to be published in the Journal of VLSI Signal Processing Systems. Submitted: November 2008; Reviewed: June 2009; 12 pages.

A design technique based on a combination of Common Sub-Expression Elimination and Bit-Slice (CSE-BitSlice) arithmetic for hardware reduction and performance improvement of multiplier designs with variable operands is presented in this paper. The CSE-BitSlice technique can be extended to hardware optimization of multiplier circuits operating on vectors or matrices of variables. The CSE-BitSlice technique has been applied to the design and implementation of 12x12 and 42x42 bit real multipliers, a complex multiplier, a 6-tap FIR filter, and a 5-point DFT circuit. For comparison purposes, circuit implementations have been carried out using Radix-4 Booth and CSA algorithms. Simulation results based on implementations using the Xilinx FPGA 5VLX330FF1760-2 device show that the circuits based on the CSE-BitSlice

techniques require fewer logic resources and yield higher throughput as compared to the CSA and Radix-4 Booth based circuits.

5. H. Ho, V. Szwarc and T. Kwasniewski, " A Reconfigurable Systolic Array Architecture for Multicarrier Wireless and Multirate Applications," to be published in the International Journal of Reconfigurable Computing. Submitted: November 2008; Reviewed: May 2009; 13 pages.

A novel reconfigurable systolic array (RSA) architecture that supports the realization of DSP functions for multicarrier wireless and multirate applications is presented. The RSA consists of coarse-grained processing elements that can be configured as complex DSP functions and basic building blocks of Polyphase-FIR filters, phase shifters, DFTs, and Polyphase-DFT circuits. The homogeneous characteristic of the RSA architecture, where each reconfigurable processing element (PE) cell is connected to its nearest neighbors via configurable switch (SW) elements, enables array expansion for parallel processing and facilitates time sharing computation of high throughput data by individual PEs. For DFT circuit configurations, an algorithmic optimization technique has been employed to reduce the overall number of vector-matrix products to be mapped on the RSA. The hardware complexity and throughput of the RSA-based DFT structures have been evaluated and compared against several conventional modular FFT realizations. Designs and circuit implementations of the PE cell and several RSAs configured as DFT and Polyphase filter circuits are also presented. The flexibility of the RSA architecture is demonstrated by means of representative 2-by-8 arrays of PEs onto which different functions are mapped and simulated. The RSA architecture offers significant flexibility and computational capacity for applications that require real time reconfiguration and high density computing.

6. H. Ho, V. Szwarc and T. Kwasniewski, " A Reconfigurable Systolic Array SoC Design for Multicarrier Wireless Applications," Proceedings of the IEEE Int'l Midwest Symp. on Circuits and Systems MWSCAS'08 Conf., Knoxville, Aug. 2008.

   The reconfigurable systolic array (RSA) architecture presented in this paper includes coarse grained processing elements and interconnection switches. The RSA based circuit can be configured to compute Polyphase-FIR filter, DFT, Polyphase-DFT and IDFT-Polyphase functions. A representative RSA circuit has been designed and implemented on an FPGA for operation in the following modes: 32-point DFT; 4-channel Polyphase filter; 8-channel IDFT-Polyphase; and 8-channel Polyphase-DFT. The circuit is scalable and can be extended to accommodate larger configurations and architectures. The scalability and reconfigurability of the circuit's architecture provides a flexible solution for multicarrier wireless applications incorporating Polyphase-DFT circuits.

   First reported implementation of an RSA intended for multicarrier applications and the SoC realization of Polyphase-FFT circuit on it.

7. H. Ho, V. Szwarc and T. Kwasniewski, "Design and Implementation of a Multiplierless Reconfigurable DFT/DCT Processor," has been accepted for publication in the proceedings of the IEEE NEWCAS-TAISA '09 Conf., Toulouse, June 28-July 2. 2009.

   In this paper, a Multiplierless Reconfigurable DFT/DCT Processor (MRP) design suitable for multicarrier applications is presented. The MRP implementation is based on a Reconfigurable Systolic Array (RSA) architecture that supports N-point DFT or DCT computations. All multiplication blocks in the MRP circuit

have been implemented using the CSE-BitSlice technique to reduce hardware usage, which in turn, helps to lower power consumption. Simulation results show that the MRP DFT circuit implementations can be used in most OFDM modulation realizations required by broadband communication systems. Simulation results of the MRP circuit configured for 8-point DCT computations also show that the MRP supports the frame rates required by data compression schemes of major digital video standards. The reconfigurability of the MRP makes it suitable for Shape Adaptive DCT (SA-DCT) computations required by object based video coding systems.

First reported implementation of an DFT/DCT processor without the use of conventional multipliers intended for broadband and digital video applications where N-point DFT and DCT is supported and N does not need to be a power of two.

8.3 Possibilities for Future Research

The distributed characteristic and reconfigurability of the RSA architecture make it suitable for fault tolerant design. As a result, the possibility of incorporating fault tolerant capability in the RSA architecture could be explored. The investigation could focus on suitable techniques to incorporate fault detection and fault reconfiguration mechanism onto the RSA architecture. It may be also worthwhile to investigate ASIC implementation of the CSE-BitSlice based RSA design. ASIC technologies would support high logic density RSA circuit implementation. ASIC-based RSA circuit could also provide more efficient MC realizations in terms of throughput performance and power. Another area that merits consideration involves the investigation of DFT implementation techniques on the RSA circuit to reduce computation complexity. Novel mapping techniques that lead to further reduction of the number of partial products

required by the vector matrix multiplications could lead to further reduction in hardware. Finally, the implementation of a configuration tool that combines the tasks of generation of coefficients and generation of configuration data to simplify and automate the mapping process for the RSA circuit may provide new venues for future research. Potential contributions of future research to engineering knowledge are:

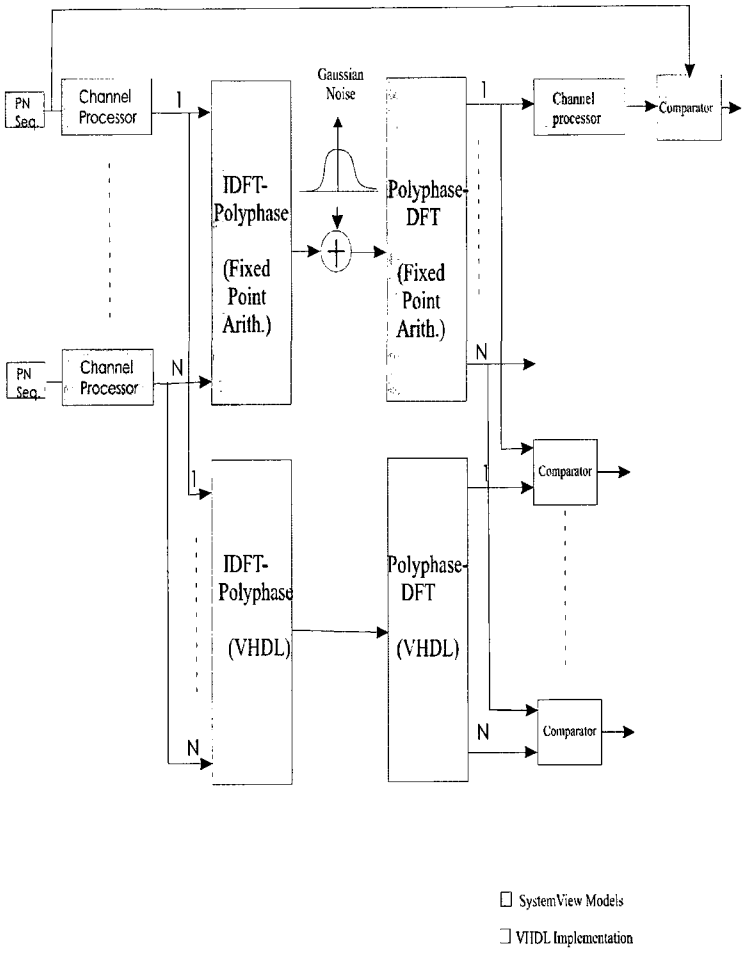
1. New design methodology for fault detection and reconfiguration of systolic array for fault tolerant applications.
2. Original technique for efficient realization of DSP systems without the use of conventional multipliers for low power and high performance applications.
3. Novel DFT mapping techniques on hardware that requires low arithmetic complexity.
4. New design automation technique for reconfigurable systolic arrays.

Appendix A

SystemView Model of IDFT-Polyphase and Polyphase-DFT Circuits

This model has been developed to generate input and output vectors for functional testing of the Polyphase-DFT and IDFT-Polyphase designs. The Polyphase filter, phase shifter and FFT building blocks in the GM and GD system have been modeled based on the set of parameters employed in the Polyphase-DFT and IDFT-Polyphase designs.

Inputs data shifted into the GM and GD system is generated by N PN generators as shown in the figure below. The signals shifted out of the transmit channel processors and the IDFT-Polyphase building blocks have been used as inputs vectors for functional testing of the IDFT-Polyphase and Polyphase-DFT designs respectively. The output sequences generated by the IDFT-Polyphase and Polyphase-DFT blocks are used to compare to the output sequences of the IDFT-Polyphase and Polyphase-DFT designs respectively.


PN Seq.
Channel Processor
1
IDFT-Polyphase
(Fixed Point Arith.)
Gaussian Noise
Polyphase-DFT
(Fixed Point Arith.)
1
Channel processor
Comparator
PN Seq.
Channel Processor
N
N
Comparator
1
IDFT-Polyphase
(VHDL)
Polyphase-DFT
(VHDL)
1
N
Comparator
SystemView Models
VHDL Implementation

Appendix B

ASIC Implementations of Representative Multipliers

In this appendix, implementation results of two representative real multiplier circuits synthesized based on Mentor Graphic's SCL05μ standard cell library are presented including the gate count and clock frequency of the multiplier circuits obtained using Leonardo Spectrum. The implemented circuits are the 12X12 bit and 48X48 bit multipliers designed using the CSE-BitSlice technique for 2-Term and 3-Term decomposition. The implementation of the same circuit designs based on the Radix-4 Booth and CSA techniques have also been carried out for comparisons. The 12X12 bit and 48X48 bit real multiplier circuits have been implemented to validate the conditions for multiplier implementations with low usage of logic resources established in Chapter 5. The gate counts and clock frequency obtained from the ASIC implementations of these multiplier circuits are compared with simulation results extracted from the FPGA implementations of similar circuits presented in Chapter 7.

The 12X12 multiplier circuit consists of 12 bit inputs. Computation results are shifted out via a 24 bit data bus. The 48X48 multiplier circuit consists of 48 bit inputs and shifts computation results out via a 96 bit data bus. Simulation results for the 12-bit and 48-bit real multiplier design implementations based on CSA, Radix-4 Booth, and CSE-BitSlice techniques, incorporating 2-Term and 3-Term decomposition, are shown in Table B1.

Table B1: Comparison of simulation results for the 12X12 bit and 48X48 bit real multiplier circuit implementations based on CSE-BitSlice, CSA, and Radix-4 Booth algorithms.

Gate Counts/ Clock Frequency		**Design Technique**			
		CSA	**Radix-4 Booth**	**CSE-BitSlice**	
				N=2	**N=3**
Gate Counts	12X12	10203	9129	6895	7485
	48X48	114334	99823	85208	78845
Clock Freq. (MHz)	12X12	48.4	65	56.4	59
	48X48	10	19	12	19

The results in Table B1 show that the 3-Term design requires more logic resources but performs at higher clock frequency as compared to the 2-Term based CSE-BitSlice design for the 12-bit circuit. For the 48-bit circuit, however, the 3-Term design requires fewer gates as well as yields a (17%) higher clock rate. The gate counts obtained in both cases are consistent with the greater and smaller number of additions required by the 3-Term design as compared to the 2-Term design for the circuits with multiplier word lengths of 12 bit and 48 bit respectively. The somewhat higher clock rate of the 3-Term design in the former case is the result of smaller word lengths of some of the adders used in the accumulation process even though both designs require a similar number of additions overall.

The Radix-4 Booth based design, however, requires a higher gate count and yields a higher clock rate for the 12-bit circuit as compared to the CSE-BitSlice designs. The higher clock frequency reflects the fact that the Radix-4 Booth design requires a smaller number of additions as compared to the 2-Term and 3-Term CSE-BitSlice designs. The

higher frequency performance of the Radix-4 Booth design is also the result of the concurrent generation of the circuit's partial products. While decoding of partial products in parallel helps boost the circuit frequency performance, it also requires a greater gate count for the Radix-4 Booth design as shown in Table B1. For the 48-bit circuit, the Radix-4 circuit requires more additions than the 3-Term CSE-BitSlice which results in a higher gate count. The throughput performance of the Radix-4 Booth and the 3-Term CSE-BiSlice designs are similar for the 48-bit multiplier circuit even though the former design requires more additions than the latter design. This is due to the fact that the CSE-BiSlice design generates its partial products based on a factorization process as compared to the parallel decoding of partial products employed by the Radix-4 Booth design. The CSA based designs have the highest gate counts and offer the lowest throughput performance as compared to the Radix-4 and CSE-BitSlice based designs for both multipliers. This outcome is the result of the large number of adders required in the CSA-based circuit implementations. The outcomes of the ASIC implementations shown in this appendix are consistent with the estimations established in chapter 5. The gate counts and clock performances of the ASIC implementations shown in Table B1 are also consistent with simulation results of similar multiplier implementations based on FPGAs presented in Table 7.1, Table 7.2 and Table 7.3 in Chapter 7.

Appendix C

DFT/IDFT Factorization for Low Complexity Computation

In this appendix, the decomposition of DFT/IDFT into multiple sums of products and then combining the expressions that multiply to a common coefficient to reduce the overall number of complex multiplication is described.

C1. Algorithm for N Even

The DFT and IDFT functions are expressed by the following equations:

$$Y(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi nk}{N}} \qquad k = 0,...,N-1 \qquad \text{(DFT)} \tag{C.1}$$

$$y(n) = \sum_{k=0}^{N-1} X(k) e^{\frac{j2\pi nk}{N}} \qquad n = 0,...,N-1 \qquad \text{(IDFT)} \tag{C.2}$$

In the following paragraphs, we will only consider the DFT algorithm since the IDFT is easily obtained from the DFT. If N is a multiple of four, equation (C.1) can be decomposed into a sum of products as follows:

$$\begin{aligned} Y(k) = & \left[x(0) + x(\frac{N}{2}) e^{-j\pi k} + \left(x(\frac{N}{4}) + x(\frac{3N}{4}) e^{-j\pi k} \right) e^{-\frac{j\pi k}{2}} \right] + \\ & \sum_{n=1}^{\frac{N}{4}-1} \left(x(n) + x(\frac{N}{2}+n) e^{-j\pi k} \right) e^{\frac{-j2\pi nk}{N}} + \\ & \sum_{n=1}^{\frac{N}{4}-1} \left(x(N-n) + x(\frac{N}{2}-n) e^{-j\pi k} \right) e^{\frac{j2\pi nk}{N}} \end{aligned} \tag{C.3}$$

The coefficients in equation (C.3) have the following relations:

$$e^{\frac{-j2\pi nk}{N}} = \left(\cos\frac{2\pi nk}{N} - j\sin\frac{2\pi nk}{N}\right)$$
$$e^{-j\pi k} = (-1)^k \qquad \text{(C.4)}$$
$$e^{\frac{j\pi k}{2}} = (-j)^k$$

Replacing the relations shown in (C.4) in equation (C.3) and simplifying we obtain:

$$Y(k) = S_1(n) + (-j)^k S_2(n) + \left(\sum_{n=1}^{\frac{N}{4}-1} \left(S_3(n) + S_4(n)\right) * a(n) + j \sum_{n=1}^{\frac{N}{4}-1} \left(S_3(n) - S_4(n)\right) * b(n) \right) \qquad \text{(C.5)}$$

where

$$S_1(n) = x(0) + (-1)^k x(\frac{N}{2})$$
$$S_2(n) = x(\frac{N}{4}) + (-1)^k x(\frac{N}{2} + \frac{N}{4}) \qquad \text{(C.6)}$$
$$S_3(n) = x(N-n) + (-1)^k x(\frac{N}{2} - n)$$
$$S_4(n) = x(n) + (-1)^k x(\frac{N}{2} + n)$$
$$a(n) = \cos\frac{2\pi nk}{N}$$
$$b(n) = \sin\frac{2\pi nk}{N}$$

From (C.5), the output sequences $Y(N-k)$, $Y(\frac{N}{2}+k)$, and $Y(\frac{N}{2}-k)$ for k = 1,..., $\left(\frac{N}{4}-1\right)$

can be inferred to be as follows:

$$Y(N-k) = S_1(n) + (j)^k S_2(n) + \left(\sum_{n=1}^{\frac{N}{4}-1} \left(S_3(n) + S_4(n)\right) * a(n) - j \sum_{n=1}^{\frac{N}{4}-1} \left(S_3(n) - S_4(n)\right) * b(n) \right) \qquad \text{(C.7)}$$

$$Y(\frac{N}{2}+k) = S_1(n) + (-j)^{k+\frac{N}{2}} S_2(n) + \left(\sum_{n=1}^{\frac{N}{4}-1} (-1)^n \left(\left(S_3(n) + S_4(n)\right) * a(n) + j\left(S_3(n) - S_4(n)\right) * b(n) \right) \right) \qquad \text{(C.8)}$$

$$Y(\frac{N}{2}-k)=S_1(n)+(j)^{k+\frac{N}{2}}S_2(n)+\left(\sum_{n=1}^{\frac{N}{4}-1}(-1)^n\left(\left(S_3(n)+S_4(n)\right)*a(n)-j\left(S_3(n)-S_4(n)\right)*b(n)\right)\right) \quad \text{(C.9)}$$

The output sequences $Y(0), Y(\frac{N}{4})$, $Y(\frac{N}{2})$ and $Y(\frac{3N}{4})$ can be calculated based on (C.5) as follows:

$$Y(k=0)=S_1(n)+S_2(n)+\sum_{n=1}^{\frac{N}{4}-1}\left(S_3(n)+S_4(n)\right) \quad \text{(C.10)}$$

$$Y(k=\frac{N}{4})=S_1(n)+(-j)^{\frac{N}{4}}S_2(n)+\sum_{n=1}^{\frac{N}{4}-1}(j)^n\left(S_3(n)+(-1)^n S_4(n)\right) \quad \text{(C.11)}$$

$$Y(k=\frac{N}{2})=S_1(n)+(-j)^{\frac{N}{2}}S_2(n)+\sum_{n=1}^{\frac{N}{4}-1}(-1)^n\left(S_3(n)+S_4(n)\right) \quad \text{(C.12)}$$

$$Y(k=\frac{3N}{4})=S_1(n)+(-j)^{\frac{3N}{4}}S_2(n)+\sum_{n=1}^{\frac{N}{4}-1}(j)^n\left((-1)^n S_3(n)+S_4(n)\right) \quad \text{(C.13)}$$

As can be seen from the above equations, by factoring out expressions that multiply a common coefficient, a significant number of complex multiplications have been eliminated. It can also be seen that each set of output sequences $Y(k), Y(N-k), Y(\frac{N}{2}+k), Y(\frac{N}{2}-k)$ can be calculated simultaneously with $\frac{N}{4}$ complex multiplication operations.

Thus, using this optimization technique, the total number of complex multiplications required to compute all N outputs simultaneously has been reduced to $\left(\frac{N}{4}\right)^2$ for an N-point DFT where N is a multiple of four.

If N is even but not a multiple of four, the output sequence Y(k) can be derived from the decomposition of (C.1) using a similar approach as follows:

$$Y(k)=\left[x(0)+x(\frac{N}{2})e^{-j\pi k}\right]+ \sum_{n=1}^{\frac{N-2}{4}}\left(x(n)+x(\frac{N}{2}+n)e^{-j\pi k}\right)e^{\frac{-j2\pi nk}{N}}+ \sum_{n=1}^{\frac{N-2}{4}}\left(x(N-n)+x(\frac{N}{2}-n)e^{-j\pi k}\right)e^{\frac{j2\pi nk}{N}} \tag{C.14}$$

From the relations defined in (C.4) and (C.6), equation (C.14) can be rewritten as:

$$Y(k)=S_1(n)+\left(\sum_{n=1}^{\frac{N-2}{4}}\left(\left(S_3(n)+S_4(n)\right)*a(n)\right)+j\sum_{n=1}^{\frac{N-2}{4}}\left(\left(S_3(n)-S_4(n)\right)*b(n)\right)\right) \tag{C.15}$$

the output sequence $y(N-k)$ for k = 1,..., $\left(\frac{N}{2}-1\right)$ is derived from (C.15) as:

$$Y(N-k)=S_1(n)+\left(\sum_{n=1}^{\frac{N-2}{4}}\left(\left(S_3(n)+S_4(n)\right)*a(n)\right)-j\sum_{n=1}^{\frac{N-2}{4}}\left(\left(S_3(n)-S_4(n)\right)*b(n)\right)\right) \tag{C.16}$$

The output sequences $Y(0)$ and $Y(\frac{N}{2})$ can be calculated based on (C.15) as follows:

$$Y(k=0)=S_1(n)+\sum_{n=1}^{\frac{N-2}{4}}\left(S_3(n)+S_4(n)\right) \tag{C.17}$$

$$Y(k=\tfrac{N}{2})=S_1(n)+\sum_{n=1}^{\frac{N-2}{4}}(-1)^n\left(S_3(n)+S_4(n)\right) \tag{C.18}$$

Thus, the sum or the difference of the factorized expressions S_1, S_3, S_4 generated for k even and odd need to be computed only once and factored out to be used as common expressions. These common expressions are then used as primary signals throughout the multiplication process to compute partial products of $\frac{N}{4}$ outputs if N is a multiple of four or $\frac{N}{2}$ outputs if N is even but not a multiple of four.

Based on (C.15) and (C.16), the total number of complex multiplications required to compute each one of the two output sequences $Y(k)$ and $Y(N-k)$ where $N=4n+2$ has been reduced to $\left(\frac{N+2}{4}\right)$. Thus, using this factorization technique, a total of $\left(\frac{N}{2}\right)\left(\frac{N+2}{4}\right)$ complex multiplications is required to compute an N-point DFT where N is even but not a multiple of four.

C2. Algorithm for N Odd

If N is odd, the output Y(k) is derived from decomposition of (C.1) as follows:

$$Y(k)=x(0)+\sum_{n=1}^{\frac{(N-1)}{2}}x(n)e^{-\frac{j2\pi nk}{N}}+\sum_{n=1}^{\frac{(N-1)}{2}}x(N-n)e^{\frac{j2\pi nk}{N}} \tag{C.19}$$

$$Y(k)=x(0)+\sum_{n=1}^{\frac{(N-1)}{2}}\left(\left(x(N-n)+x(n)\right)\cos\frac{2\pi nk}{N}\right)+ j\sum_{n=1}^{\frac{(N-1)}{2}}\left(\left(x(N-n)-x(n)\right)\sin\frac{2\pi nk}{N}\right) \tag{C.20}$$

$$Y(k)=S_5(n)+\sum_{n=1}^{\frac{(N-1)}{2}}\left(S_6(n)*a(n)\right)+j\sum_{n=1}^{\frac{(N-1)}{2}}\left(S_7(n)*b(n)\right) \tag{C.21}$$

where

$$S_5(n)=x(0)$$
$$S_6(n)=x(N-n)+x(n)$$
$$S_7(n)=x(N-n)-x(n)$$

The output $Y(N-k)$ for k = 1,..., $\left(\frac{N-1}{2}\right)$ is derived from (C.21) as follows:

$$Y(N-k)=S_5(n)+\sum_{n=1}^{\frac{(N-1)}{2}}\left(S_6(n)*a(n)\right)-j\sum_{n=1}^{\frac{(N-1)}{2}}\left(S_7(n)*b(n)\right) \tag{C.22}$$

The output sequences $Y(0)$ can be calculated based on (C.20) as follows:

$$Y(0)=S_5(n)+\sum_{n=1}^{\frac{(N-1)}{2}}S_6(n) \tag{C.23}$$

Thus, to compute the set of two outputs $Y(k)$ and $Y(N-k)$ for N odd, the total number of complex multiplications required is $\frac{N-1}{2}$.

Bibliography

[Adaptacell] Tecore Network Corporation Website http://www.airnetcom.com

[Atmel] Atmel Corporation Website http://www.atmel.com

[Akyi'02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," Computer Networks, Vol. 38, No. 4, March 2002.

[Also'02] A. Alsolaim, "Dynamically Reconfigurable Architecture for Third Generation Mobile Systems," Ph.D. Thesis, Ohio University, 2002.

[Arcu'03] Gerard Arcuri, "Wireless Mesh Networking," Faulkner Information Services, 2003.

[Arno'92] J. Arnold, D. Buell, and E. Davis, "SPLASH-II," Proc. of the 4th ACM Symp. of Parallel Algorithms and Architectures, pp. 316-322, 1992.

[Asat'90] C. Asato, C. Ditzen, and S. Dholakia, "A Data-Path Multiplier with Automatic Insertion of Pipeline Stages," IEEE Journal of Solid-State Circuits, Vol. 25, No. 2, pp. 383-387, Apr. 1990.

[Bayo'87] M. A. Bayoumi, G. A. Jullien, and W. C. Miller, "A Look-up Table VLSI Design Methodology for RNS Structures Used in DSP Applications," IEEE Trans. on Circuits and Systems, Vol. 34, No. 6, pp. 604-616, 1987.

[Beck'02] J. Becker, "Configurable systems-on-chip (CSoC)," Proc. of the 15th Symp. on Integrated Circuits and Systems Design, pp. 379-384, Sept. 2002.

[Bela'74] M. Bellanger and J. Daguet, "TDM-FDM Transmultiplexer: Digital Polyphase and FFT," IEEE Trans. on Comm., Vol. 22, No. 9, pp. 1199-1204, Sept. 1974.

[Bern'92] E. Bernard, J. G. Krammer, M. Sauer, and R. Schaweizer, "A Pipeline Architecture for Modified Higher Radix FFT," Proc. of the IEEE Int'l Conf. on Acoustic Speech and Sig. Proc., pp. 617-620, March 1992.

[Besl'02a] N. Besli and R. G Deshmukh, "A Novel Redundant Binary Signed-Digit Booth's Encoding," Proc. of the IEEE Southeast Conf., pp. 426-431, 2002.

[Besl'02b] N. Besli and R. G Deshmukh, "A 54X54 Bit Multiplier with a New Redundant Binary Booth's Encoding," Proc. of the Canadian Conf. on Elect. and Comp. Eng., Vol. 2, pp. 597-602, 2002.

[Bhuy'83] L. Bhuyan and D. P. Agrawal, "Performance Analysis of FFT Algorithms on Multiprocessor Systems," IEEE Trans. on Software Engineering, Vol. SE-9, No. 4, pp. 512-521, July 1983.

[Boot'51] A. D. Booth, "A Signed Binary Multiplication Technique," Quarterly J. of Mechanics and App. Math., Vol. 4, Part 2, pp. 236-240, 1951.

[Cimi'96] L. Ciminiera and P. Montuschi, "Carry-Save Multiplication Schemes Without Final Addition," IEEE Trans. on Computers Vol. 45, Iss. 9, pp. 1050-1055, Sept. 1996.

[Chan'00] C. H. Chang, C. L. Chang, and Y. L. Chang, "Efficient VLSI Architectures for Fast Computation of the Discrete Fourier Transform and its Inverse," IEEE Trans. Sig. Proc., Vol. 48, pp. 3206-3216, Nov 2000.

[Chen'03] K. H. Chen and T. D. Chiueh, "Design and Implementation of a Reconfigurable FIR Filter," Proc. of the Int'l. Symp. on Circuits and Systems Conf., ISCAS'03, pp. 205-208, 2003.

[Choi'07] C. Choi and H. Lee, "A Self-Reconfigurable Adaptive FIR Filter System on Partial Reconfiguration Platform," IEICE Trans. on Information and Systems, Vol.E90-D, No.12, pp. 1932-1938, Dec. 2007.

[Cort'06] A. Cortes, J.F. Sevillano, I. Velez, and A. Irizar, "An FFT Core for DVB-T / DVB-H Receivers," Proc. of the IEEE Int'l Conf. on Electronics, Circuits and Systems, ICECS'06, pp. 102-105, Dec. 2006.

[Demp'95] A. G. Dempster and M. D. Macleod, "Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters," IEEE Trans. on Circuits and Syst.-II: Analog and Dig. Sig. Proc., Vol. 42, No. 9, pp. 569-577, Sept. 1995.

[Desp'79] A. Despain, "Very Fast Fourier Transform Algorithms Hardware Implementation," IEEE Trans. on Comp., Vol. C-28, No. 5, pp. 33-341, May 1979.

[Econ'06] G. Economakos and K. Anagnostopoulos, "Bit Level Architectural Exploration Technique for the Design of Low Power Multipliers," Proc. of the IEE Int'l Conf. on Circuits and Systems, May 2006.

[Erce'90] M. Ercegovac and T. Lang, "Fast Multiplication without Carry-Propagate Addition," IEEE Trans. on Computers, Vol. 39, Issue 11, pp. 1385-1390, Nov. 1990.

[ETSI'04] ETSI 302 304, "Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)," EN 302 304 v1.1.1, Nov. 2004.

[ETSI'06] ETSI EN 300 401 "Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to Mobile, Portable and Fixed Receivers," ETSI EN 300 401 v1.4.1, 2006.

[Flie'94] N. J. Fliege, "Multirate Digital Signal Processing," John Wiley & Sons, New York, NY, USA, 1994.

[Fons'05] M. Fonseca et al., "Design of a Radix-2m Hybrid Array Multiplier Using Carry Save Adder," Proc. of the 18th Symp. on Integrated Circuits and Systems Design, pp. 172-177, Sept. 2005.

[Full'98] A. Fuller, B. Nowrouzian, and F. Ashrafzadeh, "Optimization of FIR Digital Filters over the Canonical Signed-Digit Coefficient Space Using Genetic Algorithms," Proc. of the 1998 Midwest Symposium on Circuits and Systems, MWSCAS'98, pp. 456-459, Aug. 1998.

[Garc'06] M. J. Fernández-Getino Garcia, O. Edfors, and J. M. Páez-Borrallo, "Peak Power Reduction for OFDM Systems with Orthogonal Pilot Sequences," IEEE Trans. on Wireless Communications, Vol. 5, No. 1, January 2006.

[Geie'01] J. Geier, "Enabling Fast Wireless Networks with OFDM," CommsDesign, Feb. 2001.

[Gere'00] O. Gerek and A. Cetin, "Adaptive Polyphase Subband Decomposition Structures for Image Compression," IEEE Trans. on Image Proc., Vol. 9, No. 10, pp. 1649-1660, Oct. 2000.

[Good'03] Rupert Goodwins, "Intel Makes a Mesh of Wireless Networks," ZDNet Magazines, Feb. 2003.

[Gock'92] H. Gockler and H. Eyssele, "Study of On-Board Digital FDM-Demultiplexing for Mobile SCPC Satellite Communications," European Trans. on Telecom, Vol. 3, No. 1, pp. 7-30, Feb. 2002.

[Gray'00] E. Grayver and B. Daneshrad, "A Reconfigurable 8 GOP ASIC Architecture for High-Speed Data Communications," IEEE J. on Selected Areas in Comm., Vol. 18, No. 11, pp. 2161-2171, Nov. 2000.

[Harn'95] R. Hartenstein, R. Kress, and H. Reinig, "A Scalable, Parallel, and Reconfigurable Datapath Architecture," Proc. of the Sixth International Symposium on IC Technology, Systems & Applications, ISIC'95, Singapore, Sept. 1995.

[Harn'99] R. Hartenstein, M. Herz, T. Hoffmann, and U. Nageldinger, "Mapping Applications onto Reconfigurable KressArrays," Proc. of the Int'l Workshop

on Field Programmable Logic and Applications, FPL'99, Glasgow, UK, Sept. 1999.

[Hary'96] R. I. Hartley, "Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers," IEEE Trans. Circuits Syst. II, Vol. 43, pp. 677–688, Oct. 1996.

[Haus'97] J. R. Hauser and J. Wawrzynek,"Garp: A MIPS Processor with a Reconfigurable Coprocessor," Proc. of the IEEE Symp. on FPGAs for Custom Computing Machines, CA, pp. 12-21, April 1997.

[He'94] S. He and M. Torkelson, "A Systolic Array Implementation of Common Factor Algorithm to Compute DFT," Proc. of the Int'l Symp. on Parallel Architectures, Algorithms and Networks, Kanazawa, Japan, pp. 374–381, 1994.

[Hewl'00] R. Hewlitt and E. Swartzlander, "Canonical Signed Digit Representation for Fir Digital Filters," Proc. of the IEEE Workshop on Signal Processing Systems, SIPS'2000, pp. 416-419, Oct. 2000.

[Ho'95] H. Ho, V. Szwarc, and L. Desormeaux, "A Comparison of FIR Filter Implementations Based on Two's Complement and Residue Number Arithmetic," Proc. of the 8th IEEE Int'l ASIC Conference and Exhibit, ASIC'95, pp. 35-38, Sept. 1995.

[Ho'99] H. Ho, V. Szwarc, C. Loo, and T. Kwasniewski, "Design and PLD Implementation of a Group Demultiplexer," Proc. of the 42nd Midwest Symposium on Circuits and Systems MSCAS'99, Aug. 1999.

[Ho'02] H. Ho, V. Szwarc, C. Loo, and T. Kwasniewski, "Design and Implementation of a Multi-Carrier Demodulator," Proc. of the 6nd WSEAS Int'l Conf on Circuits, July 2002.

[Ho'03] H. Ho, V. Szwarc, and T. Kwasniewski, "Design and FPGA Implementation of a MultiCarrier Baseband Processor," Proc. of the 3rd IASTED Int'l Conf. on Circuits, Signals and Systems CSS'03, May 2003.

[Ho'04] H. Ho, V. Szwarc, and T. Kwasniewski, "Design and Implementation of a Reconfigurable Polyphase-FFT System," Proc. of the 4th IASTED Int'l Conf on Circuits, Signals and Systems CSS'04, Oct. 2004.

[Ho'07] H. Ho, V. Szwarc, and T. Kwasniewski, "Hardware Optimization of a Configurable Polyphase-FFT Design Using Common Sub-Expression Elimination," Proc. of the IEEE Int'l MWSCAS/NEWCAS '07 Conf., Aug. 2007.

[Ho'08] H. Ho, V. Szwarc, and T. Kwasniewski, "A Reconfigurable Systolic Array SoC Design for Multicarrier Wireless Applications," Proc. of the IEEE Int'l Midwest Symp. on Circuits and Systems MWSCAS'08 Conf., Knoxville, Aug. 10-13 2008.

[Ho'09a] H. Ho, V. Szwarc, and T. Kwasniewski, " Design and Implementation of a Multiplierless Reconfigurable DFT/DCT Processor," has been accepted for publication in the Proc. of the IEEE NEWCAS-TAISA'09 Conf., Toulouse, June 28-July 1 2009.

[Ho'09b] H. Ho, V. Szwarc, and T. Kwasniewski, "A Reconfigurable Systolic Array Architecture for Multicarrier Wireless and Multirate Applications," has been accepted for publication in the International Journal of Reconfigurable Computing. Submitted: November 2008; Reviewed: May 2009; 13 pages.

[Ho'09c] H. Ho, V. Szwarc, and T. Kwasniewski, " Low Complexity Reconfigurable DSP Circuit Implementations Based on Common Sub-Expression Elimination," has been accepted for publication in the Journal of VLSI Signal Processing Systems. Submitted: November 2008; Reviewed: June 2009; 12 pages.

[Hosa'05] A. Hosangadi, F. Fallah, and R. Kastner, "Simultaneous Optimization of Delay and Number of Operations in Multiplierless Implementation of Linear Systems," Proc. of the 14th Intl. Workshop on Logic and Synthesis, IWLS'05, 2005.

[Hung'04] C. P. Hung, S. G. Chen, and K. L. Chen, "Design of an Efficient Variable-Length FFT Processor," Proc. of the IEEE Int'l Symp. on Circuits and Systems, ISCAS '04, Vol. 2, pp. 833-836, May 2004.

[Hwan'79] K. Hwang, "Computer Arithmetic: Principles, Architecture and Design," New York, John Wiley and Sons, 1979.

[Jenk'85] W. Jenkins and E. Davidson, "A Custom-Designed Integrated Circuit for the Realization of Residue Number Digital Filters," Proc. of the ICASSP, pp. 220-223, 1985.

[Jone'97] Jones et al., "The Use of Digital Signal Processors in Underwater Communication Systems," Proc. of the IEE Colloquium on DSP Chips in Real-Time Instrumentation and Display Systems, pp. 9/1-9/5, Sept. 1997.

[Kama'05] A. Kamalizad, et al., "A Programmable DSP Architecture for Wireless Communication Systems," Proc. of the IEEE Int'l Conf. on Application-Specific Syst., Arch. Proc., ASAP'05, pp. 231-238, July 2005.

[Kim'98] T. Kim, W. Jao, and S. Tjiang, "Arithmetic Optimization using Carry-Save-Adders," Proc. of the 35th Design Automation Conference, pp. 433–438, June 1998.

[Khoo'99] K. Khoo, Z. Yu, and A. Wilson, "Bit-Level Arithmetic Optimization for Carry-Save Additions," Proc. of the IEEE Int. Conf. Computer-Aided Design, Nov.1999.

[Khan'03] A. Khan, M. Ai-Akaidi, S. Khan, S. Khattak, and A. Mir, "Performance Analysis of a 64-point FFT/IFFT Block Designed for OFDM Technique Used in WLAN's," Proc. of the Multitopic Conf., INMIC , pp. 65-71, Dec. 2003.

[Koren'93] I. Koren, "Computer Arithmetic Algorithms," Prentice-Hall, 1993.

[Kung'80] H. T. Kung, "Special Purpose for Signal Processing Opportunity in Very Large Scale Integration (VLSI)," Real Time Signal Processing Magazine, No. 111, 1980.

[Kuo'03] J. Kuo, C. Wen, and A. Wu, "Implementation of a Programmable 64-2048-Point FFT/IFFT Processor for OFDM-Based Communication Systems," Proc. of the IEEE Int'l Symp. on Circuits and Systems, ISCAS'03, pp. II-121 - II-124, May 2003.

[Lim'99] H. Lim and E. E. Swartzlander, "Multidimensional Systolic Array for the Implementation of Discrete Fourier Transforms," IEEE. Trans. Sig. Proc., Vol. 47, No. 5, pp. 1359-1370, 1999.

[Lin'04] H. Lin, R. Chang, and M. Chan, "Design of a Novel Radix-4 Booth Multiplier," Proc. of the IEEE Asia-Pacific Conference on Circuits and Systems, pp. 837-840, Dec. 2004.

[LuSi'99] G. Lu, H. Singh, M. Lee, N. Bagherzadeh, and F. Kurdahi, "The MorphoSys Parallel Reconfigurable System," Proc. of the Euro-Par'99 Conf., Toulouse, France, Sept. 1999.

[Macl'04] M. D. Macleod and A. G. Dempster, "Common Subexpression Elimination Algorithm for Low-Cost Multiplierless Implementation of Matrix Multipliers," Elect. Lettters, Vol. 40, No. 11, May 2004.

[Macp'06] K. N. Macpherson and R. W. Stewart, "Area Efficient FIR Filters for High Speed FPGA Implementation," IEE Proceedings for Vision, Image and Sig. Proc., Vol. 153, Issue 6, pp. 711-720, Dec. 2006.

[Maha'04] K Maharatna, E. Grass, and U. Jagdhold, "A 64-Point Fourier Transform Chip for High Speed Wireless LAN Application Using OFDM," IEEE J. Solid State Circuits, Vol. 39, pp. 484-493, Mar. 2004.

[Maki'96] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara and K. Mashiko, "An 8.8-11s 54X54 Bit Multiplier With High Speed Redundant

Binary Architecture," IEEE J. Solid State Circuits, Vol. 31, No. 6, pp. 773-783, June 1996.

[Marc'05] M. J. Marcus, "Unlicensed Cognitive Sharing of TV Spectrum: The Controversy at the Federal Communications Commission," IEEE Comm. Mag., Vol. 43, No. 5, pp. 24–25, May 2005.

[Mars'99] A. Marshall, T. Stansfield, I. Kostarnov, J. Vuillemin, and B. Hutchings, "A Reconfigurable Arithmetic Array for Multimedia Applications," Proc. of the FPGA'99 Conf., Monterey, CA, pp. 135-143, Feb. 1999.

[McSo'61] O. L. McSorley, "High Speed Arithmetic in Binary Computers," Proc. of Inst. of Radio Eng (IRE), Vol. 49, No. 1, pp. 67-91, 1961.

[Mehe'95] M. Mehendale, S. D. Sherlekar, and G. Vekantesh, "Synthesis of Multiplierless FIR Filters With Minimum Number of Additions," Proc. of the 1995 IEEE/ACM International Conf. on Computer-Aided Design. Los Alamitos, CA, pp. 668–671, 1995.

[Mehr'05] A. Mehrnia and B. Daneshrad, "A Low-Complexity Multirate Channel Selector Transmit Filter Bank with Reconfigurable Bandwidth," Proc. of the IEEE Aerospace Conf., pp. 1739-1749, March 2005.

[Merm'05]G. Mermoud, A. Upegui, C. Pena, and E. Sanchez, "A Dynamically-Reconfigurable FPGA Platform for Evolving Fuzzy Systems," Proc. of the Int'l Conf. on Artificial Neural Networks, IWANN'05, pp.572-581, 2005.

[Mirs'96] E. Mirsky and A. DeHon, "MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources," Proc. of the IEEE Workshop on FPGAs for Custom Computing Machines, FCCM'96, CA, 1996.

[MRC6011] Freescale Semiconductor Corporation Website http://www.freescale.com

[Nash'05] J. Greg Nash, "Computationally Efficient Systolic Architecture for Computing the Discrete Fourier Transform," IEEE Trans. on Signal Processing, Vol. 53, No. 12, pp. 4640-4651, Dec. 2005.

[Nguy'00] H. T. Nguyen and A. Chatterjee, "Number-Splitting with Shift-and-Add Decomposition for Power and Hardware Optimization in Linear DSP Synthesis," IEEE Trans. on Very Large Scale Integration Systems, Vol. 8, No. 4, pp. 419-424, 2000.

[Nish'06] S. Nishijima, M. Saito, and I. Sugiyama, "Single-Chip Baseband Signal Processor for Software-Defined Radio," Fujitsu Scientific and Technical Journal, Vol. 42, No. 2, pp. 240-247, Apr. 2006.

[Park'02] I. C. Park and H. J. Kang, "Digital Filter Synthesis Based on an Algorithm to Generate all Minimal Signed Digit Representations," IEEE Trans. on Comp. Aided Design of Integrated Circuits and Systems, Vol. 21, No. 12, pp. 1525-1529, 2002.

[Pask'99] R. Pasko, P. Schaumont, V. Deruder, S. Vernalde, and D. Durackova, "A New Algorithm for Elimination of Common Subexpression," IEEE Trans. on Comp. Aided Design of Int. Circuits and Systems, Vol. 18, No. 1, pp. 58-68, Jan. 1999.

[Peng'97] S. Peng, I. Sedukhin, and S. Sedukhin, "Design of Array Processors for 2-D Discrete Fourier Transform," IEICE Trans. Inform., Syst., Vol. E80-D, No. 4, pp. 455-465, Apr. 1997.

[Poor'03] Robert Poor, "Wireless Mesh Networks," Sensors Online Magazine, Feb. 2003.

[Potk'96] M. Potkonjak, M. B. Shrivasta, and P. A. Chandrakasan, "Multiple Constant Multiplication: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination," IEEE Trans. Computer-Aided Design, Vol. 15, pp. 151–161, Feb. 1996.

[Rama'00] S. Ramanathan, S.K. Nandy, and V. Visvanathan, "Reconfigurable Filter Coprocessor Architecture for DSP Applications," Journal of VLSI Signal Processing, Vol. 26, No. 3, pp. 333–359, Nov. 2000.

[Re'02] M. Re, A. Del Re, and G. Cardarilli, "Efficient Implementation of a Demultiplexer Based on a Multirate Filter Bank for the Skyplex Satellites DVB System," Journal of VLSI Design, Vol. 15, No. 1, pp.427-440, 2002.

[Rica'05] A. Ricadela, "Sensors Everywhere," Information Week, Jan. 24 2005.

[Sara'07] Sarath, et al., "Exploring the Reconfigurability Options of Multi-Carrier CDMA in Cognitive Radio Systems," Proc. of the 8th IEEE Int'l Symp. on Personal, Indoor and Mobile Radio Communications, PIMRC '07, pp. 3-7, Sept. 2007.

[Sapi'90] K. Sapiecha and R. Jarocki, "Modular Architecture for High Performance Implementation of the FFT Algorithm," IEEE Trans. on Comp., Vol. 39, No. 12, pp. 1464-1468, Dec. 1990.

[Saye'92] S. I. Sayegh, J. M. Kappes, and S. J. Campanella, "On-Board Multi-Carrier Demultiplexer/Demodulator," Proc. of the Int'l Conf. on Digital Satellite Comm., pp. 433-438, 1992.

[Sche'81] H. Scheuermann and H. Gockler, "A Comprehensive Survey of Digital Transmultiplexing Methods," Proc. of the IEEE, Vol. 69, No. 11, pp. 1419-1450, Nov. 1981.

[SDR-3000] Spectrum Signal Processing Corporation Website http://www.spectrumsignal.com

[Seco'95] N. P. Secord, "Overview of Interference Analysis Methods for a Multi-Carrier Demultiplexer/Demodulator," Tech. Rep. VPCS #23/95, Communications Research Centre, July 28 1995.

[Seco'96] N. P. Secord, "Analysis of Interference Effects in Satellite On-Board Regenerative Repeaters," Tech. Rep. VPCS #02/96, Communications Research Centre, April 1996.

[Seye'91] T. Baradaran-Seyed and L. G. Johnson, "Systolic Architectures for Parallel Fourier Transform," Proc. of the 34th Midwest Symp. on Circuits and Systems, MWSCAS'91, pp. 283-286, May 1991.

[Shin'01] M. C. Shin, S. H. Kang, and I. C. Park, "An Area-Efficient Iterative Modified-Booth Multiplier Based on Self-Timed Clocking," Proc. of the IEEE Int'l Conf. on Computer Design, ICCD'01, pp. 511-514, 2001.

[Smit'95] W. W. Smith and J. M. Smith, "Handbook of Real-Time Fast Fourier Transforms: Algorithms to Product Testing," IEEE Press, New York, NY 10017-2394, 1995.

[Taka'97] F. Takahata, M. Yasunaga, Y. Hirata, T. Ohsawa, and J. Namiki, "A PSK Group Modem for Satellite Communications," IEEE J. on Selected Areas in Comm., Vol. 5, No. 4, pp. 648-661, May 1997.

[Tala'00] J. Takala, D. Akopian, J. Astola, and J. Saarinen, "Scalable Interconnection Networks for Partial Column Array Processor Architectures," Proc. of the IEEE Int'l Sym. on Circuits and Syst., pp. 513-516, May 2000.

[Tayl'84] F. J. Taylor, "Residue Arithmetic: A Tutorial with Examples," IEEE Computer Magazine, pp. 50-62, 1984.

[Vaid'90] P.P. Vaidyanathan, "Multirate Digital Filters, Filter Banks, Polyphase Networks and Applications: A Tutorial," IEEE Proc., Vol. 78, pp. 56-93, Jan. 1990.

[Vill'98] John Villasenor and Brad Hutchings, "The Flexibility of Configurable Computing," IEEE Signal Processing Magazine, Vol. 15, No. 5, pp. 67-84, Sept. 1998.

[Vino'04] A. Vinod and E. Lai, "Hardware Efficient DCT Implementation for Portable Multimedia Terminals Using Subexpression Sharing," Proc. of the IEEE Region 10 Conf., TENCON'04, Vol. 1, pp. 227-230. Nov. 2004.

[Vuil'96] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, and P. Bouchard, "Programmable Active Memories: Reconfigurable Systems Come of Age," IEEE Trans. on VLSI Systems, Vol. 4, No. 1, pp. 56-69, March 1996.

[Wain'97] E. Waingold, et al., "Baring It All to Software: Raw Machines," IEEE Computer, pp. 86-93, Sept. 1997.

[Wall'05] S. Wallner, "A Configurable System-on-Chip Architecture for Embedded and Real-Time Applications: Concepts, Design and Realization," Journal of Systems Architecture, pp. 350-367, 2005.

[Wang'00] Z. Wang and G. Giannakis, "Wireless Multicarrier Communications," IEEE Signal Processing Magazine, pp. 29-48, May 2000.

[Wang'04] X. Wang and S. G. Ziavras, "HERA: A Reconfigurable and Mixed-Mode Parallel Computing Engine on Platform FPGAs," Proc. of the 16th Int'l Conf. on Parallel and Distributed Computing and Systems, PDCS'04, pp. 374-379, Nov 2004.

[Wang'05] C.-C. Wang, J.-M. Huang, and H. C. Cheng, "A 2K/8K Mode Small-Area FFT Processor for OFDM Demodulation of DVB-T Receivers," IEEE Transactions on Consumer Electronics, Vol. 51, No. 1, pp. 28-32, 2005.

[Wein'71] S. Weinstein and P. Ebert, "Data Transmission by Frequency-Division Multiplexing Using the Discrete Fourier Transform," IEEE Trans. on Comm. Technology, Vol. 19, No. 5, pp. 628-634, Oct. 1971.

[Wen'05] M. Wen, S.Wang, and Y.Lin, "Low-Power Parallel Multiplier with Column Bypassing," Electronics Letters, Vol. 41, No. 10. pp. 581-583, May 2005.

[Wigl'06] A. Wiglinski, "Effects of Bit Allocation on Non-Contiguous Multicarrier-based Cognitive Radio Transceivers," Proc. of Vehicular Technology Conference VTC'06, pp.1-5, Sept. 2006.

[Wirt'95] M. J. Wirthlin and B. L. Hutchings, "A Dynamic Instruction Set Computer," Proc. of the IEEE Workshop on FPGAs for Computing Machines, pp. 99-107, Apr. 1995.

[Wosn'96] M. Wosnitza, M. Cavadini, M. Thaler, and G. Troster, "A Scalable VLSI Architecture for High Resolution Real Time Object Detection," Proc. of the IEEE Int'l Symp. on Circuits and Systems, ISCAS'96, Vol. 2, pp. 644-647, 1996.

[Wu'04] Q. Wu and Y. Sun, "A Novel Algorithm for Common Subexpression Elimination in VLSI Implementation of High Speed Multiplierless FIR Filters," Proc. of the 6th IASTED Intl. Conference on Signal and Image Processing, SIP'04, 2004.

[Wu'98] A. Wu, K. Tang, and C. Ng, "Pipelined Modified Booth Multiplication," Proc. IEEE Int'l Conf. on Elect. Circuits and Syst., Vol.3, pp. 51-54, Sept. 1998.

[Xilinx] Xilinx Corporation Website http://www.xilinx.com

[Xzha'07] X. Zhang, H. Rabah, and S. Weber, "Auto-Adaptive Reconfigurable Architecture for Scalable Multimedia Applications," Proc. of the NASA/ESA Conf. on Adaptive Hardware and Systems, AHS'07, pp. 139-145, 2007.

[Yang'01] Z. Yang, "Terrestrial Digital Multimedia/Television Broadcasting System," P.R.China Patent 00 123 597.4, Mar. 2001.

[Yang'02] Z. Yang, Y. Hu, C. Pan, and L. Yang, "Design of a 3780-Point IFFT Processor for TDS-OFDM," IEEE Trans. on Broadcasting, Vol. 48, No. 1, pp. 57-61, March 2002.

[Yeh'03] W. C. Yeh and C. W. Jen, "High Speed and Low Power Split Radix FFT," IEEE Trans. Sig. Proc., Vol. 51, pp. 864-874, Mar. 2003.

[Youn'96] C. Young and D. L. Jones, "Area-Efficient VLSI Implementation of Digital Filters via Multiple Product Intercoding," Proc. of the ASAP'96 Conf., pp. 19-21, Aug. 1996.

[Yu'06] C. Yu, S. Chen, and J. Chih, "Efficient Cordic Designs for Multi-Mode OFDM FFT," Proc. of the Int'l Conf. on Acoustics, Speech and Sig. Processing, ICASSP'06, pp. III1036-III1039, May 2006.

[Zahh'03] M. Abo-Zahhad, "Current State and Future Directions of Multirate Filter Banks and their Applications," Digital Signal processing, Vol. 13, No. 3, pp. 495-518, July 2003.

[Zhan'00] H. Zhang, et al., "A 1-V Heterogeneous Reconfigurable DSP IC for Wireless Baseband Digital Signal Processing," IEEE Journal of Solid-State Circ., Vol. 35, No. 11, pp. 1697-1704, Nov. 2000.

[Zhan'84] N. Zhang, "Multi-Dimensional Systolic Networks for Discrete Fourier Transform," Proc. of the 11th Int'l. Conf. on Comp. Architecture, pp. 215-222, 1984.

[Zhan'05] J. Zhang, X. Pan, and H. Shen," Asynchronous Reconfigurable Computing Array Design," Proc. of the 11th Int'l. Conf. on Embedded Software and Syst., ICESS'05, Dec. 2005.

[Zhan'07] Q. Zhang, A. Kokkeler, and G. Smit, "Cognitive Radio Design on an MPSoC Reconfigurable Platform," Proc. of the IEEE 2nd Int'l. Conf. on Cognitive Radio Oriented Wireless Networks and Comm., CROWNCOM'07, Aug. 2007.

[802.11a] IEEE STD 802.11a, "High-speed Physical Layer in 5 GHz Band," http://ieee802.org/, 1999.

[802.11b] IEEE STD 802.11b-1999, "Wireless LAN Medium Access Control (MAC) and Physical Layer Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band,"

[802.15.4] IEEE STD 802.15.4-2003," Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANS),"

[802.16a] IEEE STD 802.16a-2003, "Air Interface for Fixed Broadband Wireless Access Systems: Medium Access Control Modifications and Additional Physical Layer Specifications for 2-1 I GHz,"