VHDL Modeling and Efficient FPGA Implementation of Some Interleavers for Applications in OFDM based Wireless Communication Systems

Thesis Submitted by

Bijoy Kumar Upadhyaya

Doctor of Philosophy (Engineering)

Department of Electronics & Telecommunication Engineering
Faculty Council of Engineering & Technology
Jadavpur University
Kolkata, India
2015

Dedicated to my family

Jadavpur University
Kolkata, 700 032, India

INDEX NO. 113/11/E

1. Title of Thesis:

VHDL Modeling and Efficient FPGA Implementation of Some Interleavers for Applications in OFDM based Wireless Communication Systems

2. Name, Designation & Institution of the Supervisor:

Prof. Salil Kumar Sanyal
Professor
Department of Electronics and Telecommunication Engineering.
Jadavpur University, Kolkata-700 032
India
e-mail: s_sanyal@ieee.org and salil_sanyal@etce.jdvu.ac.in

3. List of Publication:

A) Journal Publication:

[1] **B. K. Upadhyaya** and S. K. Sanyal, "High throughput resource efficient reconfigurable interleaver for MIMO WLAN application", Computer & Digital Techniques, IET, UK, 2015 (Communicated).

[2] **B. K. Upadhyaya**, P. K. Goswami and S. K. Sanyal, "Memory Efficient LUT based Address Generator for OFDM-WiMAX De-interleaver", International Journal of Electronics and Electrical Engineering, Vol. 2, No. 1, USA, March, 2014, pp. 31-35.

[3] **B. K. Upadhyaya** and S. K. Sanyal, "Efficient FPGA Implementation of Address Generator for WiMAX De-interleaver", **IEEE Transactions** on Circuits and Systems – II, Vol. 60, Issue 8, USA, August, 2013, pp. 492-496.

[4] **B. K. Upadhyaya** and S. K. Sanyal, "Novel design of WiMAX Multimode Interleaver for Efficient FPGA Implementation using Finite State Machine based Address Generator", International Journal of Communications, Vol. 6, Issue 2, North Atlantic University Association (NAUN), USA, 2012, pp. 27-36.

[5] **B. K. Upadhyaya** and S. K. Sanyal, "An Improved LUT Based Reconfigurable Multimode Interleaver for WLAN Application", International Journal on Recent Trends in Engineering and Technology, Vol. 6, No. 2, ACEEE, USA, November, 2011, pp. 183-188.

[6] **B. K. Upadhyaya** and S. K. Sanyal, "VHDL Modeling of Convolutional Interleaver- Deinterleaver for Efficient FPGA Implementation", International Journal of Recent Trends in Engineering, Academy Publisher, Finland, Vol 2, No. 6, November, 2009, pp. 66-68.

[7] **B. K. Upadhyaya** and S. K. Sanyal, "FPGA based resource efficient QPP interleaver address generator for LTE/LTE-A application", Manuscript under preparation.

B) Conference Publication:

[8] **B. K. Upadhyaya** and S. K. Sanyal, "Design of A Novel FSM based Reconfigurable Multimode Interleaver for WLAN Application" International Conference on Devices and Communications (ICDeCom-11), Birla Institute of Technology, Mesra, India. 2011, pp. 1-5.

[9] **B. K. Upadhyaya**, I. S. Misra and S. K. Sanyal, "Novel Design of Address Generator for WiMAX Multimode Interleaver using FPGA Based Finite State Machine," 13th International Conference on Computer and Information Technology, 2010, (ICCIT-2010) Ahsanulla University of Science and Technology, Dhaka, Bangladesh, pp. 153-158

4. List of Patents:

Nil

5. List of Presentation in National / International

A) Conferences held in Abroad

[1] **B. K. Upadhyaya**, I. S. Misra and S. K. Sanyal, "Novel Design of Address Generator for WiMAX Multimode Interleaver using FPGA Based Finite State Machine," 13th International Conference on Computer and Information Technology, 2010, (ICCIT-2010) Ahsanulla University of Science and Technology, Dhaka, Bangladesh, pp. 153-158.

[2] **B. K. Upadhyaya**, P. K. Goswami and S. K. Sanyal, "Memory Efficient LUT based Address Generator for OFDM-WiMAX De-interleaver", International Journal of Electronics and Electrical Engineering, Vol. 2, No. 1, USA, March, 2014, pp. 31-35, presented in the International Conference on Advances in Electronics Engineering (ICAEE 2014), 19-20th February, 2014 at Singapore.

B) Conferences held in India

[1] **B. K. Upadhyaya** and S. K. Sanyal, "Design of A Novel FSM based Reconfigurable Multimode Interleaver for WLAN Application" International Conference on Devices and Communications (ICDeCom-11), Birla Institute of Technology, Mesra, India. 2011, pp. 1-5.

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the thesis entitled "**VHDL Modeling and Efficient FPGA Implementation of Some Interleavers for Applications in OFDM based Wireless Communication Systems**", submitted by **Sri Bijoy Kumar Upadhyaya**, who got his name registered on 06/01/2011 for the award of Ph. D. (Engineering) degree of Jadavpur University is absolutely based upon his own work under the supervision of **Prof. Salil Kumar Sanyal** and that neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

……………………………………

Prof. Salil Kumar Sanyal

Supervisor,

Professor,

Department of Electronics and Telecommunication Engineering.

Jadavpur University, Kolkata-700 032

Acknowledgements

At the outset, I would like to express my deep sense of appreciation and thanks to my supervisor Prof. Salil Kumar Sanyal, for his invaluable advice, excellent ideas, outstanding cooperation and constructive comments during my doctoral research endeavour. I would also like to thank him for encouraging me to grow as a researcher to take up future challenges in research and development. I feel honoured to get the opportunity to work under him as research fellow.

I would like to convey my sincere thanks to Prof. Iti Saha Misra, Head, Electronics and Telecommunication Engineering (ETCE) Department, Jadavpur University for her valuable advice and necessary support during the research work. My special thanks to Prof. Amit Konar, Prof. Mrinal Kanti Naskar and to all other faculty members of ETCE Department, Jadavpur University for their encouraging words and valuable advice. Special thanks are also being accorded to Sri Pratap Kumar Sarkar, Supdt. Tech. Asstt. (Retd.), DSP Laboratory for his necessary support and encouragement during the initial stage of the work. I feel myself privileged to have worked in the ETCE Department, Jadavpur University.

I also like to offer my sincere thanks to fellow researchers Sri Budhaditya Bhattacharyya, Sri Tamal Chakraborty and other scholars of OPNET Laboratory, ETCE Department, Jadavpur University for their help and support in various phases of the research work.

Taking the opportunity, I express my sincere gratitude to Dr. B. Palit, Director, Higher Education, Govt. of Tripura for permitting me to pursue Ph. D. work at Jadavpur University, Kolkata. I am grateful to Prof. Sekhar Datta, Principal, Tripura Institute of Technology, Narsingarh, Tripura for his valuable advice and support in pursuing the thesis work. Being a faculty of Department Electronics and Tele-communication Engineering of Tripura Institute of Technology, Narsingarh, I received necessary support and help from fellow colleagues within and outside the department at different point of time which I like to sincerely acknowledge.

A special thanks to my family. I am grateful to my mother, father, mother-in-law and father-in-law for all the sacrifice they made for me. Their prayers and blessings have helped me to route my career up to this point. Thanks to my sisters and brothers-in-laws for inspiring me. I acknowledge the best wishes of my dear nieces Kabyasree and Aditi. I am grateful and indebted to my beloved wife Mousumi who has displayed enormous patience and stood by me in difficult moments apart from extending help in preparation of the thesis. Her enthusiasm and constant encouragement have always been a source of inspiration to me. Thanks to my sweet daughter Sreeja for allowing me to work by sharing her part of time at home.

Finally, I extend my sincere acknowledgement to one and all who have helped me directly or indirectly in this research endeavour.

Date: 26.11.2015

Place: Jadavpur, Kolkata

Bijoy Kumar Upadhyaya
INDEX NO. 113/11/E

List of Abbreviations and Acronyms

Abbreviations/Acronyms	Description
SPC	Stored Program Control
DTMF	Dual Tone Multi Frequency
CRT	Cathode Ray Tube
VLSI	Very Large Scale Integration
EDA	Electronic Design Automation
AMPS	Advanced Mobile Phone Service
FCC	Federal Communications Commission
FDMA	Frequency Division Multiple Access
1G	First generation
2G	Second generation
TDMA	Time Division Multiple Access
CDMA	Code Division Multiple Access
GSM	Global System for Mobile
GPRS	General Packet Radio Service
EDGE	Enhanced Data for Global Evolution
3G	Third generation
W-CDMA	Wideband CDMA
UWC	Universal Wireless Communication
ISDN	Integrated Services Digital Network
DSL	Digital Subscriber Line
HFC	Hybrid Fibre Coax
ADSL	Asymmetric DSL
BWA	Broadband Wireless Access
LAN	Local Area Network
WLAN	Wireless Local Area Network
STA	Station
AP	Access Point

Abbreviations/Acronyms	Description
ISM	Industrial, Scientific and Medicine
WMAN	Wireless Metropolitan Area Network
WiMAX	Worldwide Interoperability for Microwave Access
OEM	Original Equipment Manufacturer
FBWA	Fixed BWA
MBWA	Mobile BWA
NLOS	Non Line of Sight
LTE	Long Term Evolution
LTE-A	Long Term Evolution - Advanced
E-UTRAN	Evolved Universal Terrestrial Access Network
UMTS	Universal Mobile Telecommunication System
HSPA	High Speed Packet Access
ITU	International Telecommunication Union
OFDM	Orthogonal Frequency Division Multiplexing
ICI	Inter Channel Interference
ISI	Inter Symbol Interference
CP	Cyclic Prefix
DAB	Digital Audio Broadcasting
DVB-T	Digital Video Broadcasting for Terrestrial television
DVB-H	Digital Video Broadcasting for Handheld terminals
ECC	Error Correction Codes
BER	Bit Error Rate
HDL	Hardware Description Language
FPGA	Field Programmable Gate Array
CLB	Configurable Logic Block
ASIC	Application Specific Integrated Circuit
OTP	One-Time Programmable
ATSC	Advanced Television System Committee

Abbreviations/Acronyms	**Description**
TAT	Turn Around Time
LUT	Look-up Table
IP	Intellectual Property
FSM	Finite State Machine
VHSIC	Very High Speed Integrated Circuit
IC	Integrated Circuit
VHDL	VHSIC Hardware Description Language
IEEE	Institute of Electrical and Electronic Engineers
PLD	Programmable Logic Device
SPLD	Simple Programmable Logic Device
CPLD	Complex Programmable Logic Device
I/O Block	Input Output Block
LB	Logic Block
LC	Logic Cell
STB	Set Top Box
DCM	Digital Clock Manager
IOB	Input / Output Block
DDR	Double Data-Rate
DCI	Digitally Controlled Impedance
CMT	Clock Management Tile
PLL	Phase Locked Loop
MCB	Memory Controller Block
AM	Amplitude Modulation
FM	Frequency Modulation
CD	Compact Disc
MPEG	Moving Pictures Experts Group
CA	Conditional Access
SIPO	Serial In Parallel Out
D-QPSK	Differential Quadrature Phase Shift Keying

Abbreviations/Acronyms	**Description**
MUX	Multiplexer
XST	Xilinx Synthesis Technology
FEC	Forward Error Correcting
ARQ	Automatic Repeat Request
SRAM	Static Random Access Memory
SRL16	16-bit shift register
AGB	Address Generator Block
IMB	Interleaver Memory Block
ISE	Integrated Software Environment
CC	Convolutional Coder
RS	Reed Solomon
IFFT	Inverse Fast Fourier Transform
DID	Different Interleaver Depths
BRAM	Block RAM
DRAM	Distributed RAM
DSP	Digital Signal Processing
CD	Cyclic Delay
CDD	Cyclic Delay Diversity
GI	Guard Interval
4G	Fourth Generation
MAP	Maximum a Posteriori
QPP	Quadrature Permutation Polynomial
DMB	Digital Multimedia Broadcasting
IMT	International Mobile Telecommunication
3GPP	3rd Generation Partnership Project

List of Figures

Chapter 7

List of Tables

Contents

Abstract

Error free digital wireless communication system is the ultimate goal to be achieved by communication engineers. In pursing such quest, tremendous efforts are being made by researchers to reduce the effect of channel noises. Presence of channel noises increase Bit Error Rate (BER) and degrade the performance of the communication systems considerably. Broadly, channel noises may be divided into two groups: random bit errors and burst errors. Random errors have no relation between one another whereas in burst errors a group of consecutive bits become erroneous. Researchers have developed various error correcting mechanisms to reduce the effect of such errors. Error Correcting Codes (ECC) designed for random errors are not effective for burst errors and vice versa. In most practical systems, both random and burst errors may exist together. Usually, techniques to overcome burst error are applied before ECC in order to ensure data fidelity from both types of errors. Interleaving technique is traditionally used to enhance the quality of digital transmission over a bursty channel. Interleaving is a process to rearrange code symbols so as to spread burst of errors into random like errors and can be handled by ECCs. Convolutional and block are the most popular types of interleavers being deployed in majority of the modern day communication systems to protect data against burst error.

Interleavers help to preserve data integrity during transmission over noisy channel against burst errors. The advantage is encompassed with drawbacks like additional memory requirement, system complexity and increased delay. Improved design of interleavers and efficient use of resources of the implementation platform make the interleaver a good choice to protect data from error bursts. **In case of convolution interleaver being used in DAB applications, memory wastage in the incremental shift registers is an issue to be addressed in design and implementation along with the operating speed of the circuit.** The permutation steps as prescribed in the standard documents for block interleavers of various OFDM based Broadband Wireless Access (BWA) applications like

WLAN, WiMAX, MIMO WLAN and LTE/LTE-A involves complex mathematical functions like floor, modulus and square. Implementation of these functions on hardware platform is very difficult due to the absence of direct digital hardware. Conventionally, Look-up Table (LUT) based approach is used which suffers from the drawbacks like slower speed of operation and large resource (especially memory) occupancy. **Therefore, resource efficient and low latency block interleaver design for the aforesaid applications is an important research area to work and contribute.**

In line with the formulation of research problem, efforts have been made to resolve the bottlenecks by proposing novel algorithms / efficient designs of the interleavers. MATLAB programmes are developed to verify the correctness of the novel algorithms. The proposed algorithms / designs are then transformed into digital hardware. VHDL models of these hardware have been prepared by judicious use of embedded resources available inside the reconfigurable target platform i.e. FPGA. Such efforts have clearly resulted in reduction of FPGA resources requirement with important achievement of improved speed performance. Consumption of lower power by the proposed designs is another important outcome to be reported. Timing simulations of the interleaver address generators / interleavers have been extensively carried out to verify functionality of the proposed hardware designs.

In the work to design efficient convolutional interleaver for DAB application, FPGA's embedded Shift registers (SRLC16) are used to model the incremental memory. This modelling lowers the hardware resource occupancy of FPGA in addition to reduction in memory wastage over existing implementations. In the issue of block interleaver design for IEEE 802.11 a/g based WLAN transceiver, two approaches namely improved LUT based and Finite State Machine (FSM) based have been proposed. The former technique demonstrates reduction in resource utilization like slices, flip flop and LUTs over conventional LUT based approach with improved operating speed of the Interleaver. Similar results are also obtained for FSM based implementation with further faster performance.

WiMAX is based on IEEE 802.16 d/e standard which employs special type of block interleaver. In this work, improved LUT based technique has been designed to generate de-interleaver addresses. The improvement in terms of memory saving and faster circuit operation over the conventional LUT based approach could be achieved. In addition, the author designed FSM based interleaver for the WiMAX application. Finally, a low-complexity and novel technique is proposed to efficiently implement the address generation circuitry of the 2-D de-interleaver used in the WiMAX transceiver. All these approaches result in resource efficient and high speed interleaver/de-interleaver implementations on FPGA platform. Transceiver used in MIMO WLAN employs multi stream block interleaver. In this work, hardware efficient model of MIMO WLAN interleaver eliminating the need for floor and modulus functions has been designed. To improve the performance of the address generator, embedded DSP blocks have been utilized. The work is also extended to model the interleaver memory using FPGA's embedded memory and thus provides complete hardware interleaver solution. The proposed work shows noticeable improvement in terms of maximum frequency and power consumption over the existing works. In the final phase, hardware efficient Quadratic Permutation Polynomial (QPP) interleaver address generator for LTE/LTE-A communication system is demonstrated. The address generator involves a quadratic equation and modulus function which do not have direct digital circuitry. A novel algorithm has been proposed to eliminate the need of squarer and modulus function. The algorithm is converted into efficient digital hardware and is implemented on FPGA platform with improved test results over conventional implementations.

Chapter 1
Introduction

Outline of this Chapter

The basic idea behind this Chapter is to provide description about a chronological evolution of communication systems starting from the age of Samuel Morse to modern technologies like LTE and LTE-A. It introduces and highlights the important aspects of several communication techniques including OFDM in regards to high spectral efficiency, low ICI, and ISI. Importance of interleavers used in various communication systems along with its working principle has also been elaborated. In addition, historical background, types and applications of interleavers have been incorporated. A brief discussion on VHDL and FPGA have been placed to provide basic idea to the reader. This discussion is carried forward in the next chapter more elaborately. Challenges faced during the design and implementation of interleavers for different OFDM based wireless standards like DAB, DVB, WLAN, WIMAX, MIMO WLAN and LTE / LTE-A are discussed. Major contributions of this doctoral research work in the field of communication systems have been highlighted. The chapter thereafter presents the report of extensive literature survey that has been carried out throughout this work. Finally, the chapter is concluded with discussion on methodology adopted in the research work along with organization of the thesis.

1.1 Background

1.1.1 Communication System

Communication systems have become an integral part of present day human life. It becomes almost impossible to think about survival of human being without using some or other type of communication systems [1]. Telegraph system [2] is considered to be one of the successful oldest communication systems. The idea of sending electrical signal through wire was conceptualized in France around 1798 [3], much earlier than the invention of Telegraph system. The Telegraph system developed by Samuel Morse [4] in the year 1832, became gradually popular due to the use of Morse coding technique [5] and was widely accepted by the international community.

Telephony [6] is another popular communication system which has wide spread impact in human life. Telephone was invented by Alexander Graham Bell in the year 1876 while trying to invent a talking Telegraph system [7]. “Mr. Watson, come here, I want you.” was the first experimental voice signal transmitted by Bell over the telephone to call his assistant Thomas Watson. Since then, Telephony has evolved through many generations to arrive at the present shape. In 1877-78, the first telephone exchange had been made operational. These types of exchanges were termed manual exchanges [8] wherein an operator manually connected calls with cord pairs in the telephone switchboard [9]. Automatic exchanges came into existence around in the early 1900s. They did not require manual intervention, rather followed step by step method for switching and were named after their inventor, A.B. Strowger [10]. Exchanges based on crossbar technology [11] were the next to follow in which the electromechanical telephone switchboards were arranged in matrix fashion. The first 100 line Crossbar exchange was demonstrated by the designers in 1913 at London [12]. Later in 1938, AT & T Laboratories in US introduced the crossbar-switching system commercially. Crossbar exchanges offered advantages like faster switching time and improved pulse rate. Due to the advancement in semiconductor research and invention of transistors, electronic switch based exchanges started to replace the crossbar exchanges gradually. The electronic switches are controlled by a computer through a Stored Program Control (SPC) [13]. The electronic exchanges started their operation since 1960s. Initial electronic exchanges were analog type. After the invention of microprocessors in 1971, switching control mostly relies on digital techniques leading to the introduction digital electronic exchanges [14]. Electronic exchanges offer advantages like lesser and easy maintenance, compact design, supporting additional features at reduced cost over its predecessor. Likewise, the telephone set, which is used to make call has also evolved through many changes right from analog rotary type dial to modern Dual Tone Multi Frequency (DTMF) type [15].

Radio and Television are some of the popular broadcasting communication systems. Radio broadcasting was experimented in 1905-06 but commercial broadcasting started from 1920-21 [16]. In 1927, first demonstration of Television transmission was done by J. L. Baird in UK and C. F. Jenkins in USA [17]. Rapid development in TV transmission took place due to the inventions of Cathode Ray Tube (CRT) and Picture Tube which were used both in the video camera and in the TV receiver set.

The fundamental blocks of a typical communication system [18] are shown in Fig. 1.1.

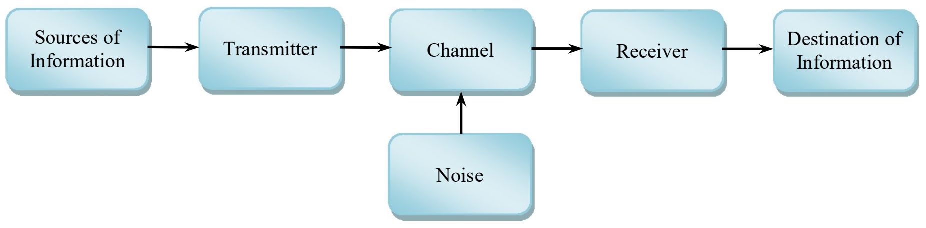


Fig. 1.1 Fundamental blocks of a Communication System

Sources of Information	There can be variety of information sources like audio signal, video signal, data etc. which the user wants to send to the destination. Suitable transducer is used to convert the signal to be transmitted before feeding to the transmitter.
Transmitter	A transmitter receives the information to be transmitted from a source, converts it into suitable format for transmission over the channel.
Channel	Media, wire or wireless, through which signal travels from transmitter to the receiver.
Noise	Unwanted disturbance may be superimposed on the signal while transmission occurs over the channel.

Receiver	The receiver receives the signal with noise from channel, removes noise to the extent possible, separates the signal from its carrier and forwards the information to the intended destination.
Destination of Information	Destination information may be a loud speaker, a CRT/Picture Tube, a Printer and so on which reproduces the original information.

1.1.2 Wireless Communication System

Wireless communication is the most exciting area of communication engineering today [19], [20]. Wireless communication system may be treated to be operational from the era when Marconi had demonstrated transmission of three-dot Morse code for the letter 'S' over a distance of three kilometres using electromagnetic waves signal [21]. Contribution of Sir J. C. Bose in early days of wireless communication for small wavelength (6 mm) millimetre wave signal generator is being recognized worldwide [22]. Early wireless communication systems were analog type. Today, most of the wireless communication systems transmit digital signal as a sequence of ones and zeros [23].

Wireless communication system is becoming more and more popular as it supports mobility of the user [24], [25]. Advancement in other associated fields like Very Large Scale Integration (VLSI) technology has provided small area, low power consuming hardware to act as catalyst for further popularity of the wireless communication system [26]. The VLSI designer is now empowered with multiple options in selecting supply voltages and transistor threshold due to technology scaling [27]. In addition, other circuit design techniques like use of dynamic power management meaning selective shut-off or slow-down of system components that are idle or underutilized though complicated, enables the designer to achieve the objectives of lower power and area efficient design of VLSI chips with throughput and latency constraints, targeted for wireless communication applications [28]. Use of Electronic Design Automation (EDA) tools assist the designer in modeling and characterizing the hardware architectures that are described using various levels of design abstraction and hence permit the designer to apply design optimizations and explore the behaviour of alternative hardware architectures [29], [30].

Mobile telephony probably is the most popular type of wireless communication system that mankind has been gifted with. Its journey started in 1947 from Bell laboratory [31]. In the immediate next year, the concept of cellular telephone service was designed to cater the increasing demand of mobile telephony. But due to lack of implementation technology, it could not become reality until 1983 when an AT&T subsidiary, Advanced Mobile Phone Service Inc. (AMPS) [32], was granted commercial license by Federal Communications Commission (FCC). AMPS was an early analog mobile phone system based on Frequency Division Multiple Access (FDMA) [33] with frequency reuse concept [34] and is known as first generation (1G) mobile telephony. Second generation (2G) was introduced in early 90s and was based on digital access technology such as Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA). It offered text messaging popularly known as Short Message Service (SMS). 2G communication is generally associated with Global System for Mobile (GSM) services for unified single standard and employs TDMA technology [35]. In order to respond to the increasing demand for internet access on mobile phone, General Packet Radio Service (GPRS), a packet oriented mobile data service has been incorporated in the 2G system and thereby introducing 2.5G mobile service. Enhanced Data for Global Evolution (EDGE) also known as Enhanced GPRS or EGPRS [36] is a data system used on top of GSM networks. It provides nearly three times faster speeds than the GPRS system. IS-95 is another 2G mobile telephony system which uses CDMA access technology and has become popular with the brand name of CDMA One.

Third generation (3G) wireless technology represents the convergence of various 2G wireless telecommunications systems into a single global system [35], [37]. It is comprised of three air interface modes: Wideband CDMA (W-CDMA), CDMA2000 and Universal Wireless Communication (UWC-136). W-CDMA is backward compatible with 2G GSM and CDMA2000 with IS-95 based 2G [38]. UWC-136 is TDMA based and is backward compatible with IS-136 TDMA digital cellular phone system defined by the ANSI-136 and IS-41 standards.

Recently, tremendous growth in wireless data networks has been witnessed [39]. Fundamentally there are two types of internet access technologies for accessing data: narrowband and broadband [40]. Narrowband refers to technologies that deliver data at up to 128 Kbps [41]. Dial up telephone connection, leased circuit using modem, Integrated Services Digital Network (ISDN) etc. are examples of narrowband internet access technologies.

On the contrary, broadband generally refers to technologies that offer high data rates, but the exact boundary between broadband and narrowband is blurry. Many suggest that broadband technologies deliver more than 1 Mbps but this is not always the case, and may mean any speed higher than dialup [41]. Examples of broadband wireless access technologies are: Digital Subscriber Line (DSL), Cable modem, Hybrid Fibre Coax (HFC) and Wireless access [41].

DSL access technology has been used to provide high speed data communication services to the subscribers over a telephone line through exchange [42]. Asymmetric DSL (ADSL) is the most widely used variant of DSL. Cable modem access technology [43] has already utilized the available wiring of cable television. HFC system uses a combination of optical fibers and coaxial cables. Fiber is used for the central facilities which demands highest bandwidth and coaxial cable is used for connections to individual subscribers requiring lesser bandwidth.

Presently popularity of Broadband Wireless Access (BWA) has been increased tremendously as it supports user mobility [44]. Wireless Local Area Network (WLAN) is the one of the oldest BWA which was originally intended to allow wireless connection to their base Local Area Network (LAN) [45]. It provides network connectivity in areas where wiring/cabling is neither cost effective nor feasible. They provide connectivity for slow mobility with high throughput for both indoor and outdoor environments. In 1997, IEEE has defined the 802.11 standard for WLAN [44]. The components of WLANs consist of a wireless network interface card, known as station (STA), and a wireless router/bridge, referred to as an Access Point (AP) [46]. The AP interfaces the wireless network with the wired network. In an outdoor environment, network coverage of 100m is typically available. The most widely used WLANs use the license free Industrial, Scientific and Medical (ISM) frequency band around 2.4 GHz [47].

Wireless Metropolitan Area Network (WMAN) is a wireless network deployed for network coverage in a wider area, targeted for covering both urban and remote areas [48]. IEEE 802.16, which defines the WMAN standards to provide cost-effective, spectrally efficient connectivity for neighbourhoods, villages, and cities. Worldwide Interoperability for Microwave Access (WiMAX) [49] [50], [51], is a WMAN technology developed by an industrial working group, with an aim to promote deployment of BWA networks by using global standards and also to provide the means for certifying interoperability of products and technologies from various vendors and Original Equipment Manufacturers (OEMs). WiMAX provides broadband connectivity over a much wider area than WLAN and may or may not require a line-of-sight path between the subscriber terminal and the APs. It claims to provide a theoretical data rate of up to 70Mbps with a range up to a maximum of 50 km. IEEE 802.16d, now known as, IEEE 802.16-2004 [52] defines fixed

BWA (FBWA) in the frequency band of 2 to 11GHz. Amended IEEE 802.16e [53] adds the mobility support to IEEE 802.16 and defines standard for mobile BWA (MBWA) in the frequency band 2 to 6 GHz. Typical data rate in IEEE 802.16e is 5 Mbps with bandwidth 1.25 to 20 MHz. Both IEEE 802.16d and IEEE 802.16e permit Non Line of Sight (NLOS) connectivity.

OFDM may be combined with multiple antennas at both the access point and mobile terminal to increase the diversity gain and/or enhance system capacity on a time-varying multipath fading channel, resulting in a Multiple Input Multiple Output (MIMO) OFDM wireless system [54]. The MIMO technology is introduced in the IEEE 802.11n protocol and brings the WLAN technology into a multi-antenna era. MIMO WLAN utilizes the MIMO-OFDM transmission techniques to enable high speed data communication with maximum throughput of 600 Mbps [55].

Long Term Evolution (LTE), or Evolved Universal Terrestrial Access Network (E-UTRAN), popularly marketed as 4G LTE, is a standard for wireless communication of high-speed data for mobile phones and data terminals [56], [57]. It is based on the GSM/EDGE and Universal Mobile Telecommunication System (UMTS)/High Speed Packet Access (HSPA) network technologies, increasing the capacity and speed using a different radio interface together with core network improvements. Long Term Evolution - Advanced (LTE-A) [58], [59], [60] aims at even higher data rate than LTE with peak data rate- downlink 3 Gbps and uplink 1.5 Gbps [61]. LTE-A is called true 4G as it actually meets the International Telecommunication Union's (ITU's) specifications for 4th generation wireless systems [62].

Satellite communication is another type popular wireless communication system having deployed in many applications like point to point communication, satellite television/radio, satellite phone, remote sensing, imaging etc. [63], [64]. It has spread its impact relatively in shorter period of time, starting with Sputnik in 1957 to sophisticated and modern present day satellite. A satellite is a self-contained system for communication which receives signal from the earth and retransmit the signals back to the earth with the help of a transponder.

1.1.3 Orthogonal Frequency Division Multiplexing Technique

Orthogonal Frequency Division Multiplexing (OFDM) [65] is becoming a popular solution for transmission of signal over a wireless channel. The basic principle of the OFDM system is to decompose the high rate data stream (bandwidth = W) into N lower rate data streams [66]. These data streams thereafter are transmitted simultaneously over a large number of subcarriers [67]. When the value of N is chosen to be sufficiently large, the individual bandwidth (W/N) of subcarriers becomes narrower than the coherence bandwidth (B_c) of the channel. In wireless channels, multiple copies of the same signal arrive in the receiver with certain time delay due to multipath propagation which is also known as delay spread. This delay spread in frequency domain shows a range of frequency having approximately flat magnitude response and is termed as coherence bandwidth (B_c) of the channel. If the bandwidth of a signal being transmitted through the wireless channel is less than the B_c, there will be no distortion in the output and the channel is known as flat fading channel. On the contrary, if signal bandwidth is larger than that of the B_c of the channel, the signal undergoes distortion which is termed as frequency selective fading. By definition, fading is the term used to refer variation in the received signal power whereas variation in the amount of fading with radio frequency is known as frequency selective fading [68].

Fundamentally, orthogonality between two signals $x_i(t)$ and $x_j(t)$ may be defined as:

$$\int_{-\infty}^{\infty} x_i(t)x_j(t) = \begin{Bmatrix} 1, & i = j \\ 0, & i \neq j \end{Bmatrix} \qquad (1.1)$$

The individual subcarriers are selected to be orthogonal to each other, which allow the overlapping between them. Due to the orthogonality feature, separation of subcarriers at the receiver end is ensured. OFDM results in a better spectral efficiency than that of FDMA systems, where no spectral overlap of carriers is permitted. Fig. 1.2 shows the spectral efficiency of OFDM pictorially. It explains the difference between the conventional non-overlapping multicarrier technique such as Frequency Division Multiple Access (FDMA) and the overlapping multicarrier modulation technique (e.g. OFDM). From the fig. it is clear that OFDM requires $BW = \left(\frac{N+1}{N}\right) R$ against $BW=2R$ of FDMA for N number of sub carriers.

The term orthogonal refers a precise mathematical relationship between frequencies of subcarriers in the OFDM-based system. In a normal FDM system, many carriers are spaced apart using guard band in such a way that the signals can be received using conventional filters in the receiver. Use of guard band between carriers (to avoid adjacent carrier interference), results in reduction of the spectrum efficiency. In an OFDM system, it is possible to arrange the carriers such that the sidebands of the individual subcarriers overlap and the signals are still received without adjacent carrier interference.

To make OFDM robust against Inter Channel Interference (ICI), and Inter Symbol Interference (ISI), Cyclic Prefix (CP) is used [70].

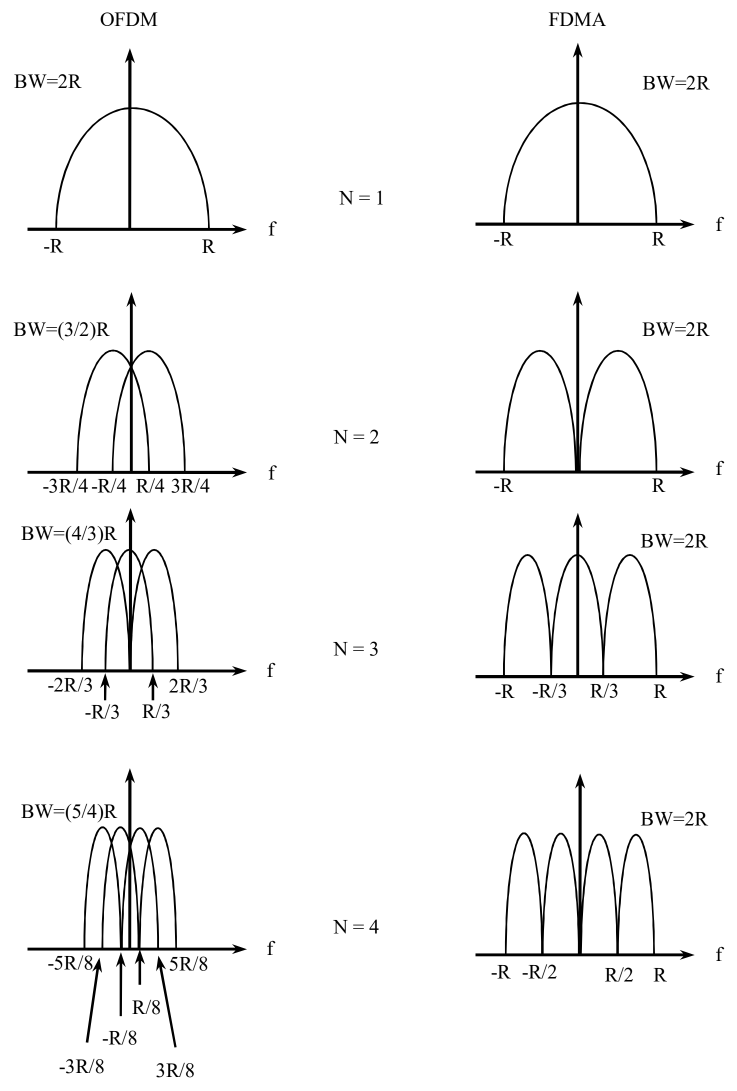


Fig. 1.2 OFDM and FDMA spectrum [69]

OFDM has been adopted by many standards, such as Digital Audio Broadcasting (DAB) [71], Digital Video Broadcasting for Terrestrial television (DVB-T) [72], Digital Video Broadcasting for Handheld terminals (DVB-H) [73], IEEE 802.11 based Wireless Local Area Networks (WLANs) [74] and IEEE 802.16 based fixed [52] and mobile [53] Broadband Wireless Access (BWA). Apart from these, OFDM is also used in latest standards like MIMO-WLAN [75], LTE [76] and LTE-A [77].

1.2 Interleaver and its significance in OFDM System

In an OFDM system, the data is divided into multiple parallel sub-streams at a reduced data rate, and each is modulated and transmitted on a separate orthogonal subcarrier. This increases symbol duration and improves system robustness. Most of the advanced high speed communication systems employ OFDM modulation along with interleaver to protect data from burst errors [78]. Error Correction Codes (ECCs) [79] play vital role in reducing the effect of random errors in communication channel. In doing so, redundancy is added that helps to identify the erroneous bit(s) in the receiver. Error bursts [80] may be defined as a group of consecutive error bits that may occur in the channel due to deep fading. ECCs do not prove to be effective during error burst. A powerful ECC however may correct the burst error but the overhead of using such ECC is very high and it may be a waste in case there is no such error. In most practical systems, both random and burst errors may exist. So, usually techniques to overcome burst error are applied before ECC in order to ensure data fidelity from both types of errors [81].

Interleaving technique is traditionally used to enhance the quality of digital transmission over a bursty channel. The principal idea behind interleaving is to mix up the code symbols from different code-words so that when the code-words are reconstructed (de-interleaved) at the receiving end error bursts encountered in the transmission are spread across multiple code words [82]. Consequently, the errors occurred within one code-word may be small enough to be corrected by using a simple random ECC. Thus interleaving is a process to rearrange code symbols to spread burst of errors into random like errors [81]. Interleaving is achieved when adjacent code symbols are separated by more than the average duration of an error burst. It improves the performance of digital transmission at the cost of increased memory requirement, system complexity, and delay. Fig. 1.3(a) and (b) explains the basic interleaving technique. In the example of Fig. 1.3(a), eight un-interleaved code words A-H, each with eight code symbols has been shown. Let, the ECC can correct one bit of error within a code word. Fig. 1.3(b) shows the interleaved code words in which one code symbol from each of the un-interleaved code word is present. These code words are allowed to travel through the channel wherein noise may superimpose. Let a burst error of eight code word length as shown in Fig. 1.3(b) occurs in the interleaved code words 1 and 2. After de-interleaving, as shown in Fig. 1.3(c), the original code words A-H with one code symbol erroneous each are obtained in the receiver. The one code symbol error can be corrected with the help of the ECC.

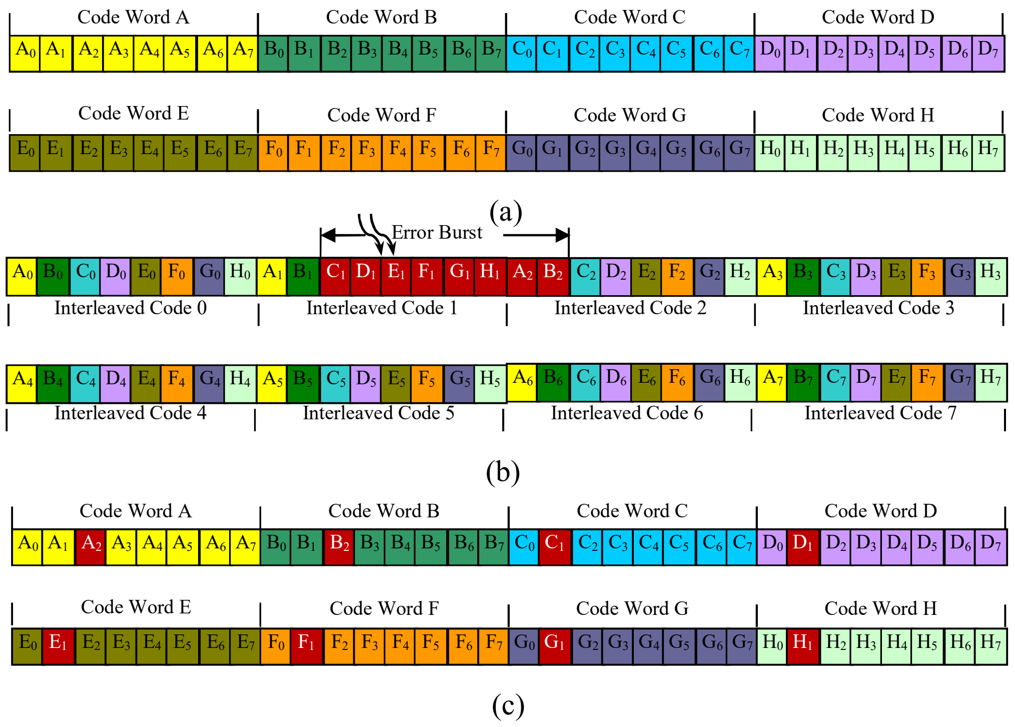


Fig. 1.3 Code words (a) Un-interleaved (b) interleaved (c) de-interleaved

This simple example demonstrates the effectiveness of interleaving technique in combating bursts of errors, i.e., how the interleaving spreads code symbols over multiple code words so as to convert a burst of errors occurred in the interleaved array into random-like errors in the de-interleaved array. In other words, the pair of interleaving and de-interleaving can equivalently convert a bursty channel into a random-like channel. Consequently, random error correction codes can be used efficiently to correct bursts of errors.

1.3 Historical Background of Interleaver

Interleavers are broadly classified into two categories [83], [84], [85]: periodic interleavers and pseudo-random interleaver [86]. In a periodic interleaver, symbols of a code word are scrambled as a periodic function of time. The period, *T* determines length of the error burst that can be effectively spread out into single bit error after de-interleaving. A pseudo-random interleaver scrambles the code word symbols in random fashion but at a distance greater than S, the separation threshold [87]. A pseudo-random sequence is generated for scrambling the code word which is to be transmitted to the receiver side for de-interleaving. Block and convolutional interleaving are the two main types of periodic interleaving techniques [88]. In block interleaver, data enters the memory in row wise manner and read out in column wise fashion. To achieve more spreading of erroneous symbols in block interleavers, certain permutation patterns are prescribed instead of simple row column combination for some applications. Similarly, based on the application, variation in the structure of convolutional interleaver is available in the literature. Table 1.1 shows important features and applications of periodic and pseudo-random interleavers.

Table 1.1: Some Features and Applications of Interleavers

Category of Interleavers	**Name of the Interleaver**	**Important features**	**Principal applications**
Periodic	Block	Memory requirement:m_r^* = 2MN Delay incurred t_{ee} = (2MN – 2M + 2)	WLAN, WiMAX, MIMO WLAN, LTE/LTE-A
	Convolutional	m_r= M(N-1) t_{ee} = M(N-1)	DAB, DVB
Pseudo-random	S-Random	$\lvert i-j\rvert \leq s\ \ after\ interleaving\ \lvert\pi(i)-\pi(j)\rvert > s^{\$}$	CDMA2000, WCDMA@

* M and N represents row and column numbers of interleaver memory respectively.

$ i and j are any two positions and s denotes the spread. Π(x) represents interleaved symbols.

@ Wide Band CDMA

Interleavers help to preserve data integrity during transmission over noisy channel against burst errors. The advantage is encompassed with drawbacks like additional memory requirement and increased delay. Improved design of interleaver and efficient use of resources on the implementation platform make the interleaver a good choice to protect data from error burst. In case of convolution interleaver, memory wastage in the incremental shift registers is an issue to be addressed in design and implementation along with the operating speed of the circuit. The permutation steps as prescribed in the standard documents for block interleavers of various applications like WLAN, WiMAX, MIMO WLAN, LTE/LTE-A [89] involves complex mathematical functions like modulus and floor. Implementation of these functions on hardware platform is the important issue to be addressed in the realization of block interleavers.

1.4 Applications of Interleaver

Interleavers are widely used in wireless communication system with principal objective to reduce the effect of fading in the channel. Concatenation of random ECC with interleavers is used in some of the communication systems to make the system more robust against both random and burst types of error. For example, DAB [90] transmitter use a convolutional interleaver between the inner (convolutional code) and outer (Reed Solomon code) encoder to achieve low Bit Error Rate (BER). In the receiver side, RS decoder is placed first followed by the convolutional de-interleaver and finally the Viterbi decoder. Convolutional code helps to reduce the random error whereas interleaver along with Reed Solomon code ensures least possible effect of burst error [91]. DVB systems also use convolutional interleaver with different parameters [72], [73]. On the other hand block interleavers are widely used in applications like WLAN [74], WiMAX [52], [53], MIMO-WLAN [75], LTE [76] and LTE-A [77]. These block interleavers generate the interleaved code words based on certain permutation patterns as prescribed in their standard document. The permutation patterns ensure separation between subsequent data symbols to achieve maximum performance out of the interleaver. Some of these interleavers are relatively simple to implement whereas some other have complex structure.

1.5 VHDL Modeling and FPGA

VHDL stands for Very High Speed Integrated Circuit - Hardware Description Language [92]. It is used to model digital electronic circuits / systems and is intended for circuit synthesis as well as circuit simulation. The VHDL modeling of the digital system can be done at different levels of abstraction: from algorithm to gate.

New or improved algorithms to be tested on hardware platform need to be converted into a Hardware Description Language (HDL) model. VHDL is usually chosen as one of the preferred alternative for such modelling. Once such model is prepared, it can be simulated to test its functionality and verify its working using test benches. On successful simulation, the model can be downloaded into reconfigurable hardware platforms like FPGA for its hardware testing. Any discrepancy noticed at any stage may be quickly rectified by making suitable changes in the VHDL model.

Field Programmable Gate Arrays (FPGAs) [93] are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. FPGAs are available in both versions: One-Time Programmable (OTP) and Static Random Access Memory (SRAM) based which can be reprogrammed as the design evolves. The latter is more popular and widely acceptable for product design. Following paragraphs present brief overview about three FPGAs being used in experimentation during the course of the research pursued by the author.

Spartan 3 [94] is one of the low cost Xilinx FPGA produced on the 90nm process technology whose design methodologies, tools, and architecture is aimed to address high-density consumer oriented applications. It contains abundance of logic gates (up to 5000K) inside it to house fairly large digital circuits. Apart from Look-up Tables (LUTs), Spartan 3 devices contain on chip dedicated 18x18 multipliers to enhance the performance of computing operation. It also includes on chip memory called Block RAM and Digital Clock Manager (DCM) to improve the performance of logic circuits implemented using them.

Spartan 3AN [95] is another FPGA developed by Xilinx with certain additional features like in-system flash memory for configuration and non-volatile data storage than its predecessor, i.e. Spartan 3. It is suitable for applications where non-volatile, system integration, security, large user flash are required and is ideal for space-critical or secure applications as well as low cost embedded controllers. Some of the applications where this FPGA is used are automotive, infotainment, telematics, GPS etc.

Spartan 6 [96] is one of the latest FPGA developed by Xilinx with aim to deliver high logic densities and reduced power consumption. It is built on 45 nm low-power technology. Some of the improved features are increased Block RAM, DSP blocks, memory controllers, enhanced clock management blocks, power optimized high-speed serial transceiver blocks etc.

1.6 Literature Survey

There are few works available in the literature addressing hardware design issues of convolutional interleaver. Yang, Zhong and Yang [97] have developed a FPGA based Forward Error Correction (FEC) decoder for Advanced Television System Committee (ATSC) digital TV. This work includes the design and implementation of convolutional de-interleaver in external dual port memory due to implementation difficulty of shift registers inside the FPGA. The authors have used Finite State Machine (FSM) based address generator for accessing the memory as convolutional de-interleaver. However, FPGA implementation results are not available for comparison to fellow researchers. Kim, Lim and Lee [98] have proposed a design for DAB transceiver for implementation on FPGA platform. The authors have modelled the convolutional interleaver in external memory due to insufficiency of flip-flops in FPGA. Due to incremental memory requirement in each subsequent row, a portion of the interleaver memory to the tune of 14% remains unutilized. In their work, the authors have worked for reduction of the memory wastage.

Hardware design of convolutional interleaver for DVB application has been proposed by Asghar & Liu in [99]. This work demonstrates unified architecture for block and convolutional interleaver supporting WiMAX and DVB applications. As per the authors, the implementation of convolutional interleaver or de-interleaver using First-In-First-Out (FIFO) register cells would be hardware inefficient due to large consumption of silicon area. Consequently, a RAM based implementation by partitioning memory with appropriate read / write logic has been employed. To keep track about the addresses of next memory location, cyclic pointers are used instead of FIFO shift registers. For each branch of the convolutional interleaver write address is provided by the concerned pointer register and the next address is computed by using an addition and a comparison with the branch boundaries. The authors used on the fly computation technique for branch boundaries using an adder and a multiplier in association with a branch counter. Due this approach, the authors claim to implement a hardware efficient design on ASIC using 0.12μm standard CMOS technology.

Unnikuttan *et. al* [100] have reported a work of designing convolutional interleaver using Verilog HDL in ModelSim software. In addition, this work includes design of a ½ code rate convolutional encoder with constraint length, K = 8 with the aim to meet the specification of DVB application. In order to test the design, the authors have made a 'test_wrap' model combining interleaver and de-interleaver blocks together. Simulation results obtained using the ModelSim software are incorporated in the paper. However, exact design procedure of convolutional interleaver used in this work has not been reported by the authors.

In addition, literature review shows some works on software platform too. One of such work as reported by Gaetzi and Hawksford [101] describes about a Simulink-MATLAB simulation model implementing complete DAB system involving the convolutional interleaver. Result analysis in respect of BER performance of the DAB system has been reported with and without involving interleaver. This analysis recommends the involvement of convolutional interleaver in the DAB transmitter.

In regard to the design and implementation of block interleaver for WLAN with IEEE 802.11 a/g standard, Tell and Liu [102] propose a modified LUT based architecture in which the memory is used as a special matrix. As per the authors, intra row permutation is carried out externally before writing data in the memory. The desired intra column permutation takes place externally after reading data bits out of the memory. Due to simultaneous writing and reading of the interleaver memory, the proposed technique can work relatively faster. In addition power consumption of the modified technique is also claimed to be lower than the conventional LUT based implementation. The only drawback reported by the authors is the minor loss of generality. The paper also states about de-interleaver implementation adopting the modified LUT based approach. In order to facilitate comparison, the authors has implemented conventional LUT based technique on the same 130 nm standard cell library of ASIC.

Sghaier *et. al* [103] have presented a full FPGA implementation of the WLAN OFDM transmitter based on IEEE 802.11a through VHDL modeling. In this work, the authors have used the LUT based approach to model the interleaver address generator, which otherwise require huge multiplexer along with memory. In IEEE 802.11a based WLAN interleaver, four interleaver depths are permitted posing the requirement of four LUTs. The authors have modelled the LUTs in the internal ROM available within the FPGA. Details of memory modeling technique used to house these LUTs are not provided. The design is implemented on Xilinx Virtex-II Pro FPGA occupying approximately 25% of the total available FPGA fabric.

Another work describing design of a FPGA-based OFDM modulator for IEEE 802.11a is available in literature [104]. As per the authors, block interleaver used in IEEE 802.11a based WLAN has been implemented on FPGA platform, including other blocks like mapper, IFFT and prefix adding module required in the transceiver. However, this paper does not explain about the implementation technique adopted while designing or implementing the block interleaver. In this work, FPGA implementation platform has been preferred by the authors due to the flexibility of re-configuration feature over ASIC. The paper has reported operating speed of 92MHz for the WLAN transceiver, when implemented on Xilinx Virtex-2 FPGA.

Limited numbers of works are available in the literature describing hardware implementation of block interleaver used in IEEE 802.16d/e based WiMAX transceiver. A technique for converting 1-dimensional interleaver equations into 2-dimension is proposed by Asghar and Liu [105]. This is due to the fact that direct implementation of interleaver function in WiMAX is not hardware efficient as it contains complex functions like floor and modulus. In addition, the conventional method i.e. using memories for storing the permutation tables is silicon consuming. Due to the mathematical translation of interleaver equations from 1-dimension to 2-dimension, the authors claim to avoid the implementation difficulty of these complex functions appearing in the interleaver equations thereby facilitating low complexity hardware implementation. This design has the capability to compute the interleaver addresses on the fly. However, the derivations in [105], do not clearly explain the design issues for all modulation schemes and code rates. The authors used 0.12μm CMOS technology as their implementation platform for the experimentation on which operating speed of 200MHz for the interleaver design has been reported.

Khater *et. al* [106] have described a VHDL based implementation of interleaver address generation circuitry for IEEE 802.16e interleaver with ½ code rate. The proposed technique is basically revolved around certain patterns that evolve during the address computation of the interleaver. In order to draw comparison, the authors claim to implement FSM based and direct method of designing WiMAX interleaver. As per the authors, these two implementation techniques have consumed huge hardware resources of the target platform. The patterns that evolved in the proposed technique of this paper is basically implemented mostly using multiplexers along with few counters and registers. FPGA implementation of the proposed technique is also reported on Altera Cyclone chip with part number EP2C5Q208C. This paper claim to simulate the interleaver design using Mentor Graphics ModelSim simulation tool through a test bench with 100 OFDMA symbols, but no such results have been incorporated in the paper permitting to draw comparison by fellow researchers.

In continuation to the work described in [104], the authors have reported another work related to design of an FPGA based OFDM modulator for IEEE 802.16-2004 [107]. During the implementation of the modulator, the authors have implemented the block interleaver along with other associated blocks required in Fixed WiMAX transceiver. The work is reported to be implemented on Xilinx Virtex-2 FPGA without mention about detailed design approach. The design is claim to work with 98.376 MHz operating frequency. During the work, MATLAB-Simulink compatibility with Xilinx System Generator has been exploited to simulate the model.

Ahmadi *et. al* [108] presented a work on design and implementation of a bit interleaver for MIMO OFDM system based on IEEE 802.22 standard. IEEE 802.22 is defined for Wireless Regional Area Network (WRAN) which involves a block interleaver. The permutation steps of the interleaver includes floor and mod function similar to WiMAX interleaver. The authors claim to implement the address generator of the interleaver using two techniques viz. fully combinational and combinational-sequential. For verification purpose test data has been generated by the authors using a MATLAB program which is given as input to the VHDL program of the interleaver. Based on the simulation results, the authors conclude that the combinational-sequential technique performs better over the other technique in terms of power and area while both methods meet the IEEE 802.22 standard timing requirements. However, implementation details of the floor and mod functions are not provided in the paper.

Apart from hardware implementation, a software model implementation of the WiMAX transceiver prepared on MATLAB-Simulink has been reported by Khan and Ghauri [109]. The model implements block interleaver as prescribed in the literature supporting all modulation schemes and code rate along with other blocks of the transceiver. The performance of the design has been evaluated through BER versus SNR logarithmic plot, time scatter plot and Signal-to-Noise Ratios plot using extensive simulation inputs. These test results are presented in the paper with inference that the model works well on SNR above 20dB.

Literature survey related to interleaver/de-interleaver design for MIMO OFDM based system results in some works in recent past having special reference to IEEE 802.16 application. Chang [110] proposed a divided memory bank architecture for the implementation of the IEEE 802.16e based de-interleaver. In addition, he proposed a dual mode architecture incorporating convolutional de-interleaver within the same design. Zafar *et al.* [111] have demonstrated performance analysis and design of channel encoder followed by interleaver for IEEE 802.16-2009 based 2x2 MIMO OFDM system. The authors used four different architectures of FEC mechanism involving convolutional encoder and interleaver which were simulated in MATLAB and Simulink environment [112]. Finally, the design is implemented on FPGA with reduced memory requirement and initial delay.

A recent work on the design and hardware implementation of MIMO OFDM system receiver including interleaver is available in the literature [113]. During the implementation of the receiver, the authors have used Intellectual Property (IP) core offered by Altera for various modules. As a result top level views of the different blocks of the receiver are only available in the paper without further implementation details. As per the authors it has some advantages of occupying less hardware resources, fast running with good stability, but not supported by any implementation result.

Another recent work [114] on the design of reconfigurable address generation circuitry for Interleaver to support multiple standards systems based on IEEE 802.11a/g and IEEE 802.16e has been reported. This work demonstrates a similar approach as done in [115] to obtain efficiency in the use of FPGA resources. This work is implemented on Xilinx Spartan 3 FPGA with necessary implementation results through Verilog HDL modeling. Software simulation results obtained using Xilinx ISE are presented by the authors to verify the design.

In the context of MIMO WLAN transceiver implementation on hardware platform, the literature review unveils some research. A work as reported in [116], demonstrated the development of a prototype transceiver for IEEE 802.11a and then upgraded to 1x4 MIMO WLAN. The authors claim to implement the transceiver on Xilinx FPGA Virtex V. Setiawan *et al.* [117] have demonstrated prototyping of 2x2 MIMO WLAN system using register transfer level (RTL) design. The authors use model-based design process for developing the RTL design of the transceiver and implemented on Altera FPGA Stratix-II. ASIC implementations of MIMO-OFDM / IEEE 802.11n transceiver are described by some researchers in [118], [119]. However, the implementations of [116], [117], [118], [119] are not focused to interleaver/de-interleaver and don't contain detailed implementation results.

In [120], Zhang *et. al* have presented a de-interleaver address generator implementation on 0.13µm CMOS platform. This paper does not explain the details about transition from de-interleaver expressions prescribed in IEEE 802.11n standard into the hardware architecture. The authors claim that the implementation is also done on FPGA platform but without any implementation result. 2-D translation with recursion of the interleaver equations for hardware simplicity is proposed by Asghar and Liu [121]. Due to this translation, alternative way to implement the complex functions as available in literature becomes possible. The final expressions derived are very complex and do not clearly explain about the hardware design issues especially for 64-QAM. The implementation platform of this work is reported to be 65nm CMOS technology. Another recent work [122] reported by the authors of [120] claim betterment over their previous work in terms of reduction in complexity and improvement in maximum operating frequency keeping the same implementation platform. The improvement claimed by the authors is due to exchanging steps between interleaver and de-interleaver.

In connection with the design of address generator for QPP interleaver used in LTE / LTE-A transceiver, literature survey shows limited number of works. A unified approach of Turbo decoder design suitable for mobile WiMAX and 3GPP-LTE applications has been proposed by Kim and Park [123]. The authors have considered dual mode approach of designing Almost Regular Permutation (ARP) and QPP interleavers to avoid huge area overhead caused by separate RAM based interleavers. This work includes design of QPP interleaver with the support of radix-4 single-binary turbo decoding. The authors claim to use retiming approach to reduce the critical path delay at the cost of increased hardware requirement to enhance the operating frequency of the circuit. Unlike RAM based approach, it facilitates on the fly address generation feature. The work is claimed to be implemented on 0.13µm CMOS technology. However, separate implementation result of QPP interleaver is not available in the paper.

Sun and Cavallaro [124] proposes a low complexity QPP interleaver of LTE / LTE-A and implemented using 65nm CMOS technology. This work includes algebraic description of the QPP interleaver leading to listing of three algebraic properties which is supposed to ease the interleaver design process. Recursive method of interleaver address computation is implemented by manipulating the underlying equations of the interleaver as proposed in the literature. As per the authors, this approach is adopted to make the design to consume lesser hardware resources. In addition, this approach require some parameters to be pre-computed and to be stored in LUTs. Also, the design has the capability to compute interleaver addresses in descending order for backward address generation with the overhead of re-computation of parameters stored in LUTs. One of the drawback of the work is that it lacks clarity about the design of circuitry to compute modulus function. The authors extended the work to design Turbo code decoder to be used in 3GPP LTE / LTE-A.

Recursive approach of QPP interleaver design for 3GPP-LTE parallel turbo code decoder has also been adopted in [125]. The authors have implemented a memory based architecture of the interleaver supporting the bandwidth required by LTE. In addition, a general architecture of the QPP interleaver design approach for contention free applications has been proposed. Like the work in [123], this work also uses radix-4 Soft Input Soft Output (SISO) approach to design the Turbo code decoder for prototyping in ASIC. As per the authors, the QPP interleaver design proposed by them supports the radix-4 architecture.

Another work of designing QPP interleaver for high throughput HSPA/LTE Multi-Standard turbo decoder is proposed in [126] by the same group of researchers of [124]. The work shows improved LUT based approach where inter row and inter column parameters are pre-computed and stored in memory. Based on the input parameter, and the inputs received from various intermediate pre-processing units, desired interleaver addresses are generated. The work is reported to be implemented on 45nm CMOS technology.

1.7 Motivation Behind the Work

As discussed in Section 1.2, interleavers play an important role in reducing the effect of channel noise especially burst noise. The performance of interleaver depends on many factors like type of interleaver and their specifications. As an example, improvement to the tune of 2 dB due to use of block interleaving over no interleaving is recorded by the authors in WLAN application [127].

Different communication standards use different interleavers. As a result, design of interleaver hardware for different standards like DAB, DVB, WLAN, WIMAX, MIMO-WLAN and LTE / LTE-A is not a unique approach. Moreover, operating speed of hardware interleavers is a critical issue and need to be carefully dealt with during the design process. FPGA is one of the most preferred platforms for testing and prototyping digital hardware due to its re-configurability feature and shorter design time. Looking at the importance of interleavers in digital communication systems, and popularity of the FPGA platform to design and test newer algorithms, the author is motivated to the design and implement of different types of interleavers especially the address generators by proposing novel and improved approach. Such approaches are expected to deliver low latency and resource efficient hardware design of the different interleavers on FPGA platform.

1.8 Objective of the Research

Importance of interleaver to preserve data integrity from burst error encountered in the channel has been discussed in previous sections. Protection from such error become more significant in the context of OFDM based high data rate wireless communication system. To cater such demand, efficient design of interleaver is an important issue to be addressed. Detailed literature review as discussed in section 1.6, suggests that not much work so far has been carried out in the issue of efficient hardware design of interleaver for implementation on FPGA platform. The main bottleneck in designing convolutional interleaver for DAB / DVB or other similar communication systems is to have low latency and memory efficient model. One of the objectives of this work is to remove such deficiency from convolutional interleaver. The block interleavers employed in applications like WLAN, WiMAX, MIMO WLAN, LTE/LTE-A can be divided into two parts: address generator and interleaver memory. Working principle of the address generator is guided by certain permutation patterns as prescribed in the literature. Resource efficient design of the interleaver address generator for FPGA implementation with improved speed performance is an important objective to be fulfilled through this research work. Alongside, judicious use of embedded memory blocks of FPGA while designing interleaver memory module is another important objective to be satisfied. Such objective if fulfilled properly, reduces the memory requirement which may be utilized by other sub-blocks of the transceiver, thereby providing opportunity for System on Chip (SOC) implementation of the BWA transceiver on the same FPGA.

1.9 Challenges Faced During the Research Work

Increasing use of multimedia services and growth of graphics based internet related contents lead to rising demand of high speed broadband wireless systems. Newer standards are being proposed to cater this demand. There are many issues related to the implementation of these standards. Software platforms like MATLAB [128] etc. are convenient for implementation but the desired performance may not be achieved due to the constraints like maximum processor clock frequency etc. In addition, the processor architectures are usually meant for general applications and may not yield the desired performance for certain high speed applications. For such high speed applications, hardware platforms are most practical solution. Presently FPGA has been considered to be the most preferred hardware platform for testing and implementation of such standards due to its shorter Turn Around Time (TAT), ease of future up-gradation, obsolescence free design etc.

An interleaver / de-interleaver comprises of two sub-sections namely address generator and interleaver memory. Design of digital hardware of the interleaver address generator used in OFDM based wireless standards like DAB, DVB, WLAN, WIMAX, MIMO-OFDM based WLAN and LTE are challenging due to the presence of complex functions like floor and modulus. These complex functions do not have any corresponding digital hardware for implementation. In addition, VHDL doesn't support such functions directly. As a result, challenges are faced in preparing the VHDL model of the interleaver / de-interleaver circuitry due to unavailability of such functions. Conventional LUT based approaches are found to be consuming large amount of logic resources apart from slowness in operation. This leads to low speed design with inefficient use of resources. For example, the LUT based address generator for WiMAX de-interleaver consumes approximately 80% more logic resources and works at half of speed than an improved technique proposed in [115]. During literature survey it has been noticed that not much work so far has been carried out in designing hardware efficient digital circuit for implementation of the interleavers especially the address generator on reconfigurable platform like FPGA. Efficient implementation on FPGA platform offers advantages like reduced logic circuit requirement, better operating speed than the conventional approaches. Taking the opportunity, the author during his investigation, has designed several hardware efficient interleavers including the address generator for the aforesaid applications on FPGA platform. The efficiency has been established by reducing the logic circuit requirement and slowness in operation.

1.10 Major Contributions in Wireless Communication Systems

The principal focus of the work is to design hardware efficient interleaver for various OFDM based high speed wireless communication systems. In this thesis work, efficient hardware implementations of both types of interleavers, i.e. convolutional and block, have been carried out. Firstly, block level representation of the designs is prepared. Each block is decomposed into most suitable digital circuits which thereafter are converted into appropriate VHDL models using Xilinx Integrated Software Environment (ISE) for FPGA implementation.

The contributions made by the author as embodied in the thesis are highlighted as follows.

- ♦ In one of the works, much effort has been given to design efficient convolutional interleaver for DAB applications. The proposed design utilizes FPGA's embedded Shift registers (SRLC16) to model the incremental memory. This modelling provides significant improvement in the operating speed of the convolutional interleaver followed by lower power consumption, requirement of lesser hardware resources and memory wastage compared to existing implementations.

- ♦ Two approaches have been followed by the author in designing the hardware for the IEEE 802.11 a/g based WLAN interleaver namely improved LUT based and FSM based. The former technique demonstrates significant reduction in resource utilization like slices, flip-flop and LUTs over the conventional LUT based approach. Similar results are also obtained for FSM based implementation. In addition, the FSM based technique offers further fast performance over the conventional LUT based method.

- ♦ WiMAX is another BWA based on IEEE 802.16 d/e standard which uses special type of block interleaver. Conventionally, LUTs are used to generate the interleaver addresses. The author has proposed improved LUT based technique to generate de-interleaver addresses. The improvement in terms of saving above 81% of memory blocks and 30% faster circuit operation than the conventional LUT based approach have been achieved. The author has also proposed FSM based interleaver address generator for WiMAX system. The work has been carried forward to design the complete FSM based interleaver (with memory) for the WiMAX application. Finally a low-complexity and novel technique is proposed to efficiently implement the address generation circuitry of the 2-D de-interleaver used in the WiMAX transceiver using the Xilinx FPGA with significantly lesser amount of hardware resources followed by higher speed improvement for different modulations and code rates.

- ♦ In the work related to speed power improved hardware design of interleaver address generator for use in MIMO WLAN, significant contributions have been made for the design of hardware efficient model of MIMO WLAN interleaver eliminating the need for floor and modulus functions for various higher order modulations and code rates. The fundamental aspect behind the development of such efficient hardware lies in the removal of these two functions during implementation phase. The work is also extended to model the interleaver memory using FPGA's embedded memory and thus provides complete hardware interleaver solution. The proposed design when compared with recent works shows noticeable betterment in terms of maximum operating frequency, power consumption and hardware resources.

- Finally the work related to the design of hardware efficient Quadratic Permutation Polynomial (QPP) interleaver address generator for LTE / LTE-A communication system is taken up. A novel algorithm has been proposed to eliminate the need of squarer and modulus functions. The algorithm is converted into digital hardware which is implemented on a reconfigurable platform with improved test results in terms of FPGA resource utilization including lesser requirement of BRAM and speed of operation in comparison with conventional implementations.

1.11 Methodology

The principal focus of the works carried out in this research is to design low complexity; hardware efficient models of interleavers deployed in OFDM based BWA. The algorithms used in interleaving processes have been reviewed. Conventional approaches like LUT based and others have been surveyed. The algorithms have been re-designed to obtain efficiency in structure. Proposed hardware structures were modelled into VHDL employing Xilinx ISE. While designing VHDL model, effort was given to use the target FPGA's embedded resources like shift register, memory, multiplier etc. to make the design faster and more resource efficient.

The proposed designs were primarily implemented and tested on three FPGA platforms namely Spartan 3, Spartan 3AN and Spartan 6, all from Xilinx Inc. Software simulation of the works were carried out using ModelSim software. MATLAB has been used for initial testing of the proposed algorithms on software platform.

1.12 Organization of Thesis

This thesis is divided into eight Chapters. **Chapter 2** gets started with brief historical background of VHDL followed by its advantages. Similarly evolution of hardware implementation platforms of digital circuits right from discrete ICs to SPLDs, to CPLDs and finally to FPGA has been narrated. It gives idea about components of a VHDL model and describes about different methodology used while preparing VHDL a model. Next FPGA fundamentals, its architecture with details has been presented. Finally, brief overviews of Xilinx Spartan 3, Spartan 3AN and Spartan 6 FPGA used in the research work have been reported.

Chapter 3 describes about design and implementation of convolutional interleaver and de-interleaver for DAB application. Interleaving operation of convolutional interleaver with necessary diagrams has been presented elaborately. The chapter provides brief introduction about embedded shift register of FPGA used to model the incremental memory of the interleaver. Use of such shift register has reduced hardware resource requirement of FPGA in addition to reduction in memory wastage. VHDL model of the proposed interleaver and de-interleaver pair along with simulation result in the form of timing diagram have been presented.

Chapter 4 introduces WLAN fundamentals along with the block interleaver used in the transceiver. It describes technique of modeling distributed and block RAM available inside Xilinx FPGA. In this work, two approaches namely improved LUT based and FSM based have been followed in designing the hardware for the IEEE 802.11 a/g based WLAN interleaver. Both design approaches along with their hardware models, simulation diagrams have been presented. Modeling of interleaver memory in FPGA using block RAM and critical analysis of FPGA implementation results of the two techniques have also been described.

Design and efficient implementation issues of WiMAX block interleaver has been described in **Chapter 5**. Overview of WiMAX transceiver along with interleaver / de-interleaver background has been discussed. In this chapter, three works related to design of WiMAX hardware interleaver and de-interleaver are presented. The first work is all about the design of a novel FSM based multimode, high speed and hardware efficient technique to implement the address generation circuitry of WiMAX interleaver based on IEEE 802.16e standard on FPGA platform. An LUT based de-interleaver design approach is presented next. In this approach, the conventional LUT based technique for address generation has been re-designed to use the FPGA memory blocks efficiently. The third technique is about design of a low complexity and resource efficient hardware de-interleaver for use in IEEE 802.16e based WiMAX. Transformation of address generator algorithms into digital circuits, based on mathematical formulations and the relevant simulation results have also been described in this chapter. Comparative study of implementation results with previous researcher / conventional techniques has been incorporated for each of the approaches.

Chapter 6 reports about efficient design and implementation of MIMO WLAN interleaver on Xilinx Spartan 6 FPGA. It presents back ground information about MIMO WLAN followed by brief description on MIMO WLAN transceiver employing multi stream block interleaver. Interleaving operation along with its detailed specification has been discussed next. Novel algorithm for the address generator of the interleaver including its mathematical formulation, transformation into digital hardware is presented. Like previous investigations, timing simulation and FPGA implementation results have been discussed at length.

The concluding research work of this doctoral thesis have been carried out on the design of hardware efficient QPP interleaver address generator for LTE/LTE-A communication system and has been described in **Chapter 7.** Working principle followed by proposed algorithm of the interleaver has been discussed. Hardware realization along with software simulation has been presented next. FPGA implementation result along with comparative analysis with conventional technique is described to demonstrate supremacy of the proposed design.

Finally a conclusive remark has been drawn for incorporation in the Conclusion section of the thesis work with a direction to future work in the areas of interleaver design for dual mode operation between MIMO WLAN - LTE/LTE-A, speech signal processing, advanced image transmission, 5G MIMO, Optical/Quantum Wireless systems and Massive MIMO Signal Processing applications. An exhaustive Bibliography has also been included.

Chapter 2
VHDL & FPGA Fundamentals

Outline of this Chapter

The core objective of this chapter is to disseminate fundamental concepts of VHDL and FPGA owing to their ever increasing importance as design platform in general and also in this particular research work. The chapter begins with introductory remarks on VHDL and FPGA in continuation with Chapter 1. In the first part of the chapter, it discusses fundamental elements of a VHDL model. In addition, it presents various types description that are usually employed while preparing the VHDL model of a given digital circuit. The later part of the chapter deals with the discussion on general architecture of FPGA. The three essential blocks of the architecture i.e. Programmable Logic Block, I/O Block and Programmable Interconnect Block have been described with supporting figures. The chapter ends with brief overview of three different latest Xilinx FPGAs used in the research work.

2.1. Introduction

VHDL is the language for describing digital electronic systems [92]. It arose out of the United States Government's Very High Speed Integrated Circuits (VHSIC) program, initiated in 1980. During the course of this program, it became clear that there was a definite need for a standard language for describing the structure and function of integrated circuits (ICs). As a consequence, the VHDL was developed, and subsequently adopted as a standard by the Institute of Electrical and Electronic Engineers (IEEE) in the US.

VHDL is designed to fill a number of needs required in the design process. Firstly, it allows description of the structure of a design i.e. how it is decomposed into sub-designs [129], and how those sub-designs are interconnected. Secondly, it allows the specification of the function of designs using popular programming language forms. Thirdly, as a result, it allows a design to be simulated before being manufactured, so that designers can quickly compare possible alternatives and test for correctness without the delay thereby reducing the expense of hardware prototyping as well as design turn around time [130].

In the beginning, digital circuits were designed with 74XX series ICs. Designing moderately large circuit with these ICs was not an efficient way due to numerous reasons like larger real estate occupancy of the circuit board, increased power consumption, lack of compactness etc. Programmable Logic Devices (PLDs) [131] were introduced to solve the problem. A PLD is supplied to the user with no logic function programmed in it. It is up to the designer to make the PLD to perform whatever way a design requires. Only the resources required by the design are utilized. Since several functions can usually be combined in the design and programmed on to a single chip, the chips count, real estate occupancy of PCB and power consumption, all are reduced considerably. Being reprogrammable, any change required during the design can be incorporated, often without removing it from the circuit.

PLDs such as PROM, PLA and PAL, also known as Simple Programmable Logic Devices (SPLDs) [132] have limited number of inputs, product terms, and outputs, which are insufficient to implement fairly complex logic circuits. A new sophisticated type chips, called Complex Programmable Logic Devices (CPLDs) [132] were developed to cater the increasing requirement. CPLDs consist of multiple SPLD like blocks connected together by a programmable switching matrix housed altogether inside a single chip. Though

CPLDs provide logic capacity which is higher than 50 typical SPLD devices, but increasing the logic density of CPLD further becomes difficult due to interconnection complexity. To increase the logic density and to add more functionality in a single programmable device, alternative architecture have been developed which are known as Field Programmable Gate Arrays (FPGAs). FPGAs comprise of an array of unconnected circuit elements and interconnect resources which are utilized for the implementation of logic functions by end user through programming.

2.2. Components of a VHDL Model

The purpose of VHDL descriptions is to provide a model for digital circuits and systems. This abstract view of the real physical circuit is referred to as entity [133]. An entity normally consists of five basic elements, or design units, as shown in Fig. 2.1 below. In VHDL, one generally distinguishes between the external view of a module and its internal description. The external view is reflected in entity declaration which represents an interface description of a 'black box'. The important part of this interface description consists of signals over which the individual modules communicate with each other. Fig. 2.2 explain the format of entity declaration with an example of 4 to 1 multiplexer.

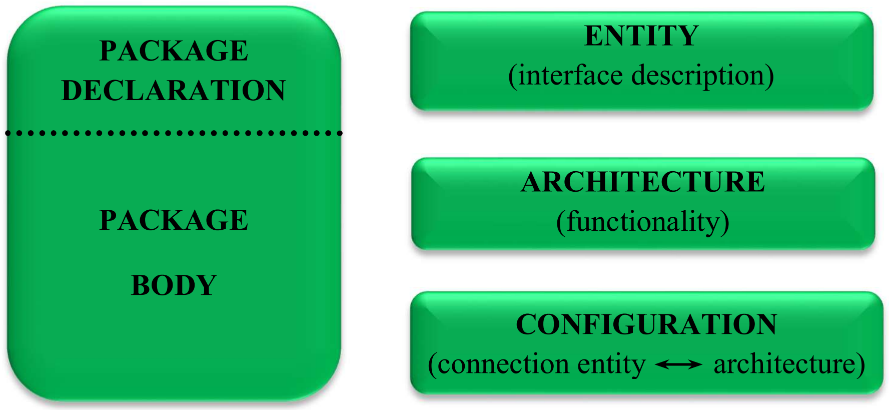
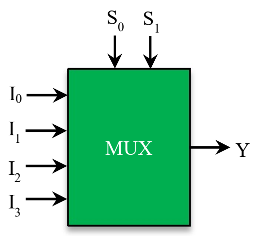
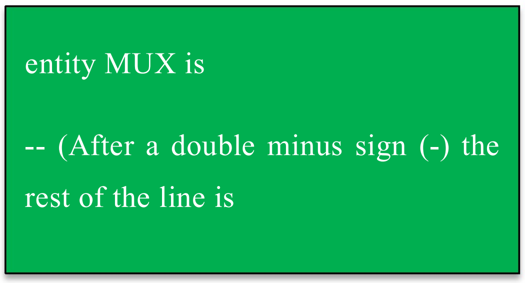


Fig. 2.1 Basic Elements of A VHDL Model.

The internal view of a module and, therefore its functionality is described in the architecture body. This can be achieved in various ways. One possibility is given by coding of a behavioral description with a set of concurrent or sequential statements. Another possibility is a structural description which serves as a base for the hierarchically designed circuit architectures. Fig. 2.3 explains the two types of architectures with the example of full adder. Naturally these two kinds of architectures can also be combined. The lowest hierarchy level however must consist of behavioral descriptions.



(a)



```
entity MUX is

-- (After a double minus sign (-) the
rest of the line is
```

(b)

Fig. 2.2 (a) Block Diagram of a 4 to 1 Multiplexer (MUX) (b) Its Entity Declaration

One of the major VHDL features is the capability to deal with multiple different architectural bodies belonging to the same entity declaration. In this case, it is necessary to bind one of the architectures to the entity in order to have a unique hierarchy for simulation or synthesis. Being able to investigate different architectural alternatives, the development of the systems could be done in an efficient top-down manner. The ease of switching between different architectures has another advantage, namely quick testing. This also includes switching between behavioral descriptions based on the different algorithms, as well as switching to gate level net lists, for example, after a partial synthesis is performed. Which architecture should be used for simulation or synthesis in conjunction with a given entity is specified in the configuration section. If the architecture body consists of structural description, then the binding architecture and entities of the instantiated sub modules, the so called components, can also be fixed by the configuration statement.

The package is the last element mentioned here. It contains declarations of frequently used data types, components, functions and so on. The package consists of a package declaration and a package body. The declaration is used, like the name implies, for declaring the above mentioned objects. This means, they become visible to other design units. In the package body, the definition of these objects can be carried out, for example the definitions of functions or the assignments of a value to a constant. Packages are language elements which can be compared with header files and the belonging codes, or objects files, found in programming language C. The portioning of a package into its declaration and body provides advantages in compiling the model descriptors.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity full_adder is
   port(in1, in2, c_in: in std_logic;
   sum, c_out: out std_logic);
end full_adder;

architecture behavioural of full_adder is

begin
  process(in1, in2, c_in)
      begin
          if (in1=’0’ and in2=’0’ and c_in=’0’) then
              sum <= ‘0’;
              c_out <=’0’;
          elsif (in1=’0’ and in2=’0’ and c_in=’1’) then
              sum <= ‘1’;
              c_out <=’0’;
          elsif (in1=’0’ and in2=’1’ and c_in=’0’) then
              sum <= ‘1’;
              c_out <=’0’;
          elsif (in1=’0’ and in2=’1’ and c_in=’1’) then
              sum <= ‘0’;
              c_out <=’1’;
          elsif (in1=’1’ and in2=’0’ and c_in=’0’) then
              sum <= ‘1’;
              c_out <=’0’;
          elsif (in1=’1’ and in2=’0’ and c_in=’1’) then
              sum <= ‘0’;
              c_out <=’1’;
          elsif (in1=’1’ and in2=’1’ and c_in=’0’) then
              sum <= ‘0’;
              c_out <=’1’;
          else
              sum <= ‘1’;
              c_out <=’1’;
          end if;
      end process;
end behavioural;
```

(a)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity full_adder is
   port(in1, in2, c_in: in std_logic;
   sum, c_out: out std_logic);
end full_adder;

architecture structural of full_adder is

component half_adder is
  port (x,y: in std_logic;
  sum, carry: out std_logic);
end component;

component or_2 is
  port (x,y: in std_logic;
  z: out std_logic);
end component;

signal s1, s2, s3: std_logic;

begin

H1: half_adder port map(x=>in1, y=>in2, sum=>s1,
carry=>s3);

H2: half_adder port map(x=>s1, y=>c_in,
sum=>sum, carry=>s2);

O1: or_2 port map(x=>s2, y=>s3, z=>c_out);

end structural;
```

(b)

Fig. 2.3 VHDL Modeling of a Full Adder in (a) Behavioral (b) Structural

2.3 FPGA Fundamentals

Fig. 2.4 shows the typical FPGA architecture [93]. There are three key parts of its structure: Programmable Logic Blocks, I/O Blocks, and Programmable Interconnect. The I/O Blocks form a ring around the outer edge of the part. Each of these provides individually selectable input, output, or bi-directional access to one of the general-purpose I/O pins on the exterior of the FPGA package. Inside the ring of I/O Blocks lies a rectangular array of logic blocks. Programmable interconnect steers the output of one logic block to the input of another logic block or I/O Blocks to logic blocks and vice versa. The logic blocks within an FPGA can be as small and simple as the macrocells in a PLD called Fine Grained or larger and more complex called Coarse Grained [134]. However, they are never as large as an entire PLD, as the logic blocks of a CPLD are.

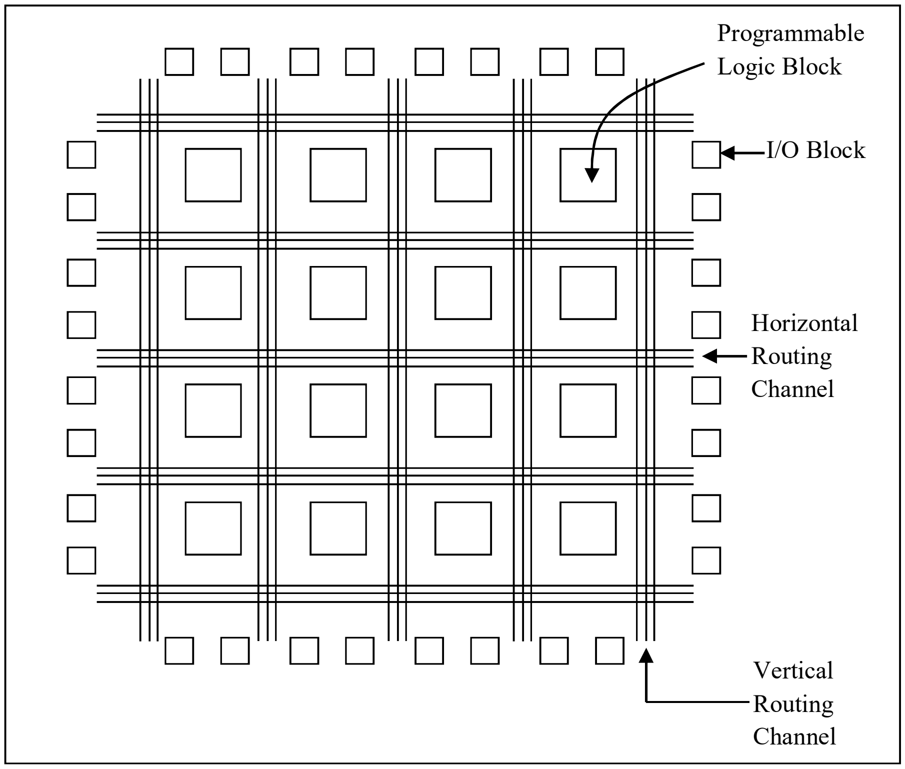


Fig. 2.4 General Architecture of FPGA.

2.3.1 Programmable Logic Block

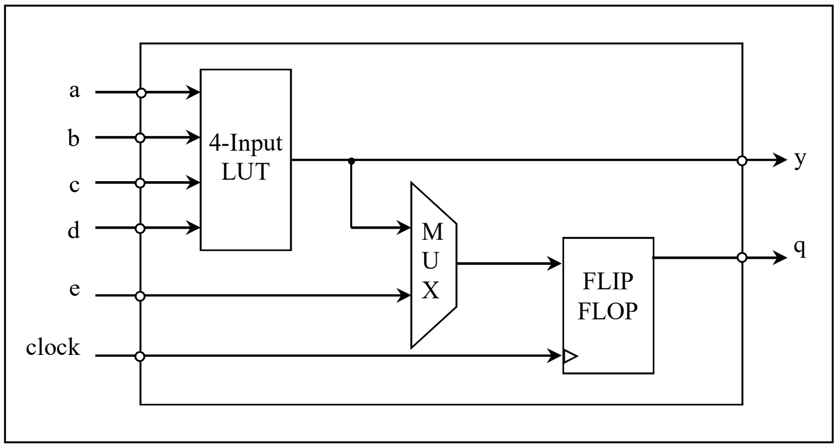
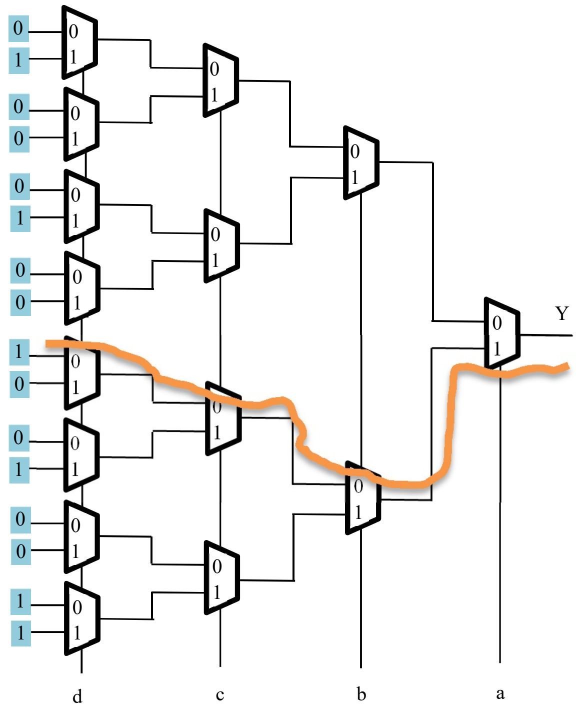


Fig. 2.5 Simplified Diagram of a Typical Programmable Logic Block

A typical Programmable Logic Block of FPGA is shown in Fig. 2.5. The Logic Block consists of a 4-input Look Up Table (LUT), a register, a clock signal and a user programmable multiplexer (MUX) [135]. The 4-input LUT is basically used as function generator which is capable of realizing any arbitrarily defined Boolean function of four inputs. Next paragraph will discuss about a 4-input LUT in detail. Each register could be configured to initialize with logic 0 or logic 1 and also to act as a flip-flop or latch. If the flip-flop option is selected, the register can further be configured to be triggered by positive edge or negative edge of the clock. The MUX feeding the flip-flop can be configured to accept the output from the LUT or a separate input to the logic block. All these programming can be done by configuring the SRAM cell or the EEPROM cell or the antifuse whatever technology is implemented. Most of the FPGAs available in the market are SRAM based. Combinatorial output is available at y whereas registered output is at q.

Fig. 2.6. shows internal structure of a 4-input LUT. It shows implementation of the four input Boolean function, $Y(a, b, c, d) = \sum m\{1, 5, 6, 8, 11, 14, 15\}$. The truth table representation of the Boolean function is to be implemented in the LUT is provided. As shown in Fig. 2.6, 15 numbers of 2-input multiplexers are required to implement the LUT. Inputs of the first level multiplexers are set or reset as per the output to be generated and should be identical with the output Y. For example, Y = 1, for a=1, b=0, c=0 and d=0. This implementation in the LUT has been shown with a free hand solid line.

Boolean Function to be implemented: $Y(a,b,c,d) = \sum m\{1,5,6,8,11,14,15\}$



a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Fig. 2.6 internal structure of a 4-input LUT implementing
$Y(a,b,c,d) = \sum m\{1,5,6,8,11,14,15\}$

2.3.2 Input - Output Block

The Input Output Block (I/O Block) provides a programmable bi-directional interface between an I/O pin and FPGA's internal logic. Fig. 2.7 shows the simplified block diagram of I/O Block. There are three signal paths available in the I/O Block: Input path, Output path and Control path. The Input path carries data from the I/O pin to FPGA's internal logic through Buffer. Data from FPGA's internal logic to the I/O pin is carried by the output path. The Control path determines when the output driver would function in natural mode or in high impedance state. When it is in high impedance state (Control = Low) the I/O pin works as input line otherwise the pin works as output line. It is evident from Fig. 2.7 that output is active low.

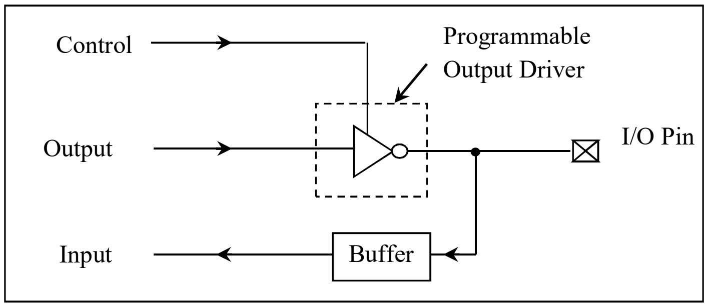


Fig. 2.7 Simplified Block Diagram of I/O Block.

2.3.3 Programmable Interconnect

In addition to its logic, an important feature that distinguishes individual FPGAs is the Programmable Interconnect structure. As shown in Fig. 2.8, the Programmable Interconnect structure is basically horizontal and vertical routing channels [136], [137]. Each channel contains short wire segments (singles) that span a single Logic Block (LB) [138] and longer segments spans two LBs (doubles). In addition, there are some very long segments (not shown in Fig. 2.8) that span's the entire FPGA length or width [139].

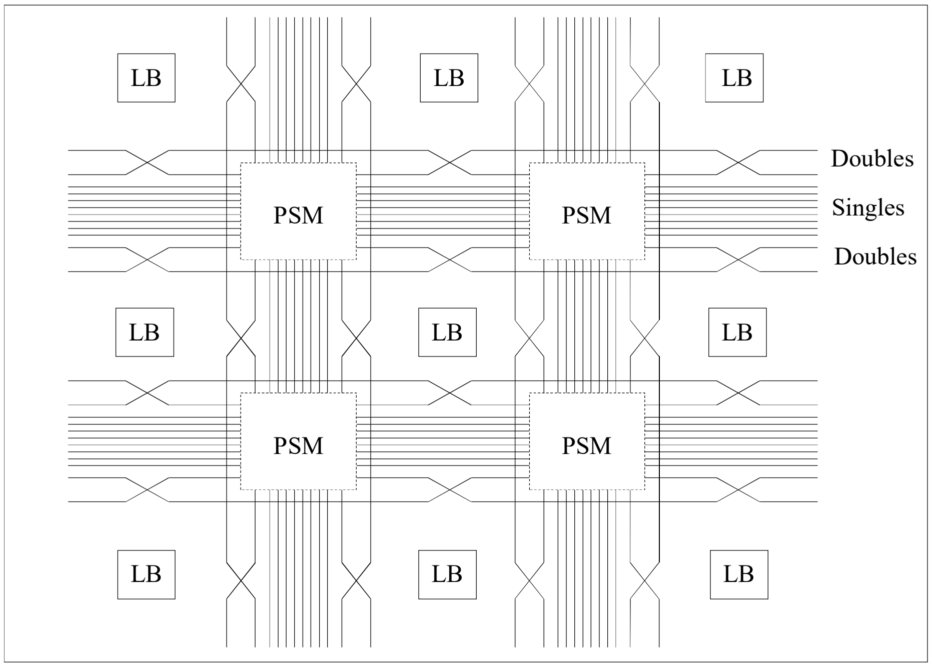


Fig. 2.8 Detailed view of interconnection routing between Logic Blocks.

Programmable Switch Matrix is used to connect LB's inputs and outputs to the wire segments or to connect one wire segment with the other. Inside the Switch Matrix, each wire can connect the other three wires as shown in Fig. 2.9. Fig. 2.10 shows an interconnect point implemented using SRAM based technology fuse [134]. In this technique, an SRAM cell controls the ON/OFF status of the transistor. During programming of the FPGA, the desired SRAM through which connection is to be established, receives a '1' whereas others will receive '0'.

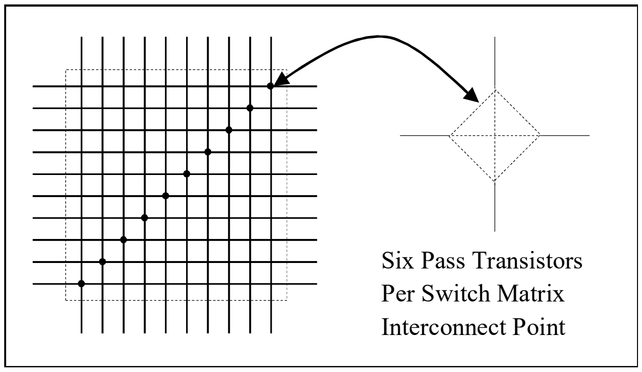


Fig. 2.9 Detailed view of Programmable Switch Matrix Interconnection of FPGA

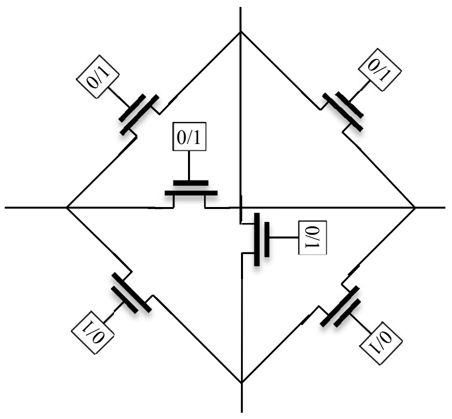


Fig. 2.10 Detailed view of Switch Matrix Interconnect Point implemented using SRAM Technology

2.4 FPGA platform used in the experimentation of Interleaver

This section describes some of important features of Xilinx FPGAs used in relevant experimentations during the research studies.

2.4.1 Spartan 3

The Spartan-3 is one of the low cost FPGAs produced on the 90nm process technology [94] whose design methodologies, tools, and architecture are aimed at addressing high-density consumer oriented applications such as Set Top Box (STB), MP3 based personnel digital player, vending machine etc. The eight-member family offers densities ranging from 50,000 to five million system gates, as shown in Table 2.1. Fig. 2.11 shows one of the package marking of Xilinx Spartan 3 FPGA with part number XC3S400-4PQ208C.

Table 2.1: Summary of Spartan-3 FPGA Attributes

Attributes	**XC3S 50**	**XC3S 200**	**XC3S 400***	**XC3S 1000**	**XC3S 1500**	**XC3S 2000**	**XC3S 4000**	**XC3S 5000**
System Gates	50K	200K	400K	1M	1.5M	2M	4M	5M
Logic Cells (LC)	1,728	4,320	8,064	17,280	29,952	46,080	62,208	74,480
CLBs	192	480	896	1,920	3,328	5,120	6,912	8,320
Dedicated Multipliers	4	12	16	24	32	40	96	104
Block RAM Bits	72K	216K	288K	432K	576K	720K	1,728 K	1,872 K
Distributed RAM	12K	30K	56K	120K	208K	320K	432K	520K
Digital Clock Manager (DCM)	2	4	4	4	4	4	4	4
Maximum User I/O	124	173	264	391	487	565	712	784

* The author has used FPGA kit in the Laboratory based on XC3S400 device of Spartan-3 Family.

A Logic Cell (LC) as shown in Fig. 2.5, also known as LB contains RAM based 4-input Look-Up Table (LUT) and a D flip flop. Configurable Logic Blocks (CLBs) comprises of eight such LCs to implement logic and storage elements that can be used as flip-flops or latches. CLBs can be programmed to perform a wide variety of logical functions as well as to store data.

Spartan 3 devices supports on chip dedicated 18x18 multipliers to enhance the performance of computing operation. Use of dedicated multiplier improves the performance of a FPGA based design by reducing interconnection delay in the CLBs and makes the design resource efficient by permitting CLBs to use by other circuitry. In addition to basic multiplication functions, the embedded multiplier block can be used as a shifter or to generate magnitude or two's-complement return of a value. The multipliers can be cascaded with each other or CLB logic for larger or more complex functions.

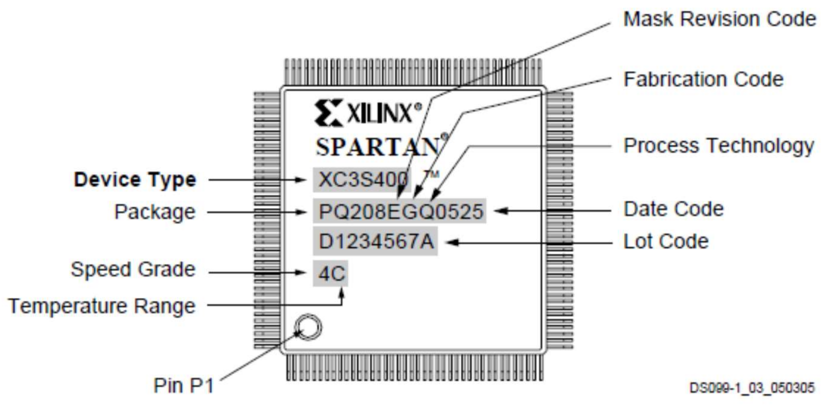


Fig. 2.11 Package marking of Xilinx Spartan 3 FPGA with part number XC3S400-4PQ208C (Courtesy Xilinx Inc.)

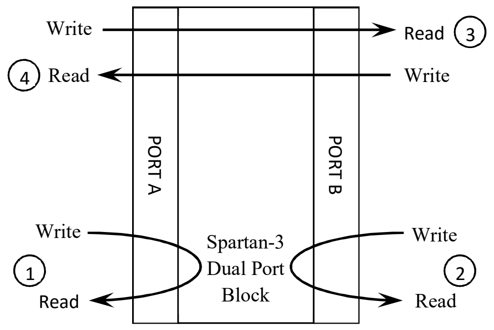


Fig. 2.12 Single port and dual port data transfer of BRAM

In order to support requirement of large, on-chip memories for various applications, Spartan-3 Generation FPGAs provides memory blocks namely Block RAM. Using various configuration options [94], these embedded memory blocks can be used to utilize the Block RAM fully as RAM, ROM, FIFOs, large look-up tables, data width converters, circular buffers, and shift registers. The Block RAMs support dual port feature as well. Fig. 2.12 shows the internal structure of a dual port BRAM permitting independent access to common RAM block which has maximum capacity of 18KB or 16KB when no parity lines are used. Each port has its own dedicated set of data, control and clock line for synchronous read and write operations. The BRAM of Spartan-3 supports the features of dual-port memory as well as all data flow operations simultaneously. The four possible schemes (Fig. 2.12) of data transfer to/from the BRAM [94] are as follows:

a) Port A behaves as an independent single-port RAM supporting simultaneous read and write operations using a single set of address lines.
b) Port B behaves as an independent single-port RAM supporting simultaneous read and write operations using a single set of address lines.
c) Port A is the write port with a separate write address and Port B is the read port with a separate read address. The data widths for port A and Port B need not be same also.
d) Port B is the write port with a separate write address and Port A is the read port with a separate read address. The data widths for port A and Port B need not be same also.

CLBs of Spartan 3 FPGA contain up to 64 bits of single-port RAM or 32 bits of dual-port RAM. This RAM is distributed throughout the FPGA and is commonly called “distributed RAM” to distinguish it from block RAM. Distributed RAM is fast, localized, and ideal for small data buffers, FIFOs, or register files.

Digital Clock Managers (DCMs) provide advanced clocking capabilities to Spartan-3 FPGA applications. DCMs optionally multiply or divide the incoming clock frequency to synthesize a new clock frequency. DCMs also eliminate clock skew, thereby improving system performance. Similarly, a DCM optionally phase shifts the clock output to delay the incoming clock by a fraction of the clock period.

Input / Output Blocks (IOBs) control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow with 3-state operation. Double Data-Rate (DDR) registers are included. The Digitally Controlled Impedance (DCI) feature provides automatic on-chip terminations, simplifying board designs.

2.4.2 Spartan 3AN

Spartan 3AN is the family of Xilinx FPGA which combines all the features of the Spartan-3A FPGA family with additional features like in-system flash memory for configuration and non-volatile data storage. It provides up to 11MB of flash memory which can be used for both device configuration as well as a valuable system resource. It is suitable for applications like automotive, infotainment, telematics, GPS etc. It contains five devices with the attributes listed in Table 2.2

Table 2.2 Summary of Spartan-3AN FPGA Attributes

Attributes	**XC3S 50AN**	**XC3S 200AN**	**XC3S 400AN**	**XC3S 700AN**	**XC3S 1400AN***
System Gates	50K	200K	400K	700K	1400K
Logic Cells (LC)	1,584	4,032	8,064	13,284	25,344
CLBs	176	448	896	1,472	2,816
Dedicated Multipliers	3	16	20	20	32
Block RAM Bits	54K	288K	360K	360K	576K
Distributed RAM	11K	28K	56K	92K	176K
Digital Clock Manager (DCM)	2	4	4	8	8
Maximum User I/O	108	195	311	372	502

* The author has also used FPGA kit in the Laboratory based on XC3S1400AN device of Spartan-3AN Family.

2.4.2 Spartan 6

The Spartan-6 is a thirteen-member family of FPGA that aims to delivers expanded densities ranging from 3,840 to 147,443 logic cells, with half the power consumption of previous Spartan families, and faster, more comprehensive connectivity. It is built on 45 nm low-power copper process technology. The Spartan-6 family offers a new, more efficient, dual-register 6-input LUT logic and a rich selection of built-in system-level blocks. These include 18 Kb (2 x 9 Kb) block RAMs, second generation DSP48A1 slices, SDRAM memory controllers, enhanced mixed-mode clock management blocks, power optimized high-speed serial transceiver blocks etc. These features provide a low cost programmable alternative to custom ASIC products with relatively ease of use. Spartan-6 FPGAs offer the better solution for high-volume logic designs, consumer-oriented DSP designs, and cost-sensitive embedded applications. Some of the important attributes of Spartan 6 FPGA family is listed in Table 2.3.

Spartan 6 FPGAs contain up to six number of Clock Management Tile (CMT). A CMT is consisting of two Digital Clock Managers (DCMs) and one Phase Locked Loop (PLL), which can be used individually or cascaded. The PLL can serve as a frequency synthesizer for a wider range of frequencies and as a jitter filter for incoming clocks in conjunction with the DCMs.

Table 2.3 Summary of Spartan-6 FPGA Attributes

Attributes	Logic Cells (LC)	Configurable Logic Blocks (CLBs)			DSP48A 1 Slices	Block RAM Blocks		CMT	Memory Controller Blocks	Max user I/O
		slices	Flip flop	Max. dist. RAM		18 Kb	Max Kb			
XC6S LX4	3840	600	4800	75	8	12	216	2	0	132
XC6S LX9	9152	1430	11440	90	16	32	576	2	2	200
XC6S LX16	14579	2278	18224	136	32	32	576	2	2	232
XC6S LX25	24051	3758	30064	229	38	52	936	2	2	266
XC6S LX45	43661	6822	54576	401	58	116	2088	4	2	358
XC6S LX75	74637	11662	93296	692	132	172	3096	6	4	408
XC6S LX100	101261	15822	126576	976	180	268	4824	6	4	480
XC6S LX150	147443	23038	184304	1355	180	268	4824	6	4	576
XC6S LX25T	24051	3758	30064	229	38	52	936	2	2	250
XC6S LX45T	43661	6822	54576	401	58	116	2088	4	2	296
XC6S LX75T	74637	11662	93296	692	132	172	3096	6	4	348
XC6S LX100T	101261	15822	126576	976	180	268	4824	6	4	498
XC6S LX150T	147443	23038	184304	1355	180	268	4824	6	4	540

* The author has also used FPGA kit in the Laboratory based on XC6SLX25 device of Spartan-6 Family.

Most Spartan-6 devices include dedicated memory controller blocks (MCBs), each targeting a single-chip DRAM (either DDR, DDR2, DDR3, or LPDDR), and supporting access rates of up to 800 Mb/s. The MCB has dedicated routing to predefined FPGA I/Os. If the MCB is not used, these I/Os are available as general purpose FPGA I/Os.

DSP applications use many binary multipliers and accumulators, best implemented in dedicated DSP slices. All Spartan-6 FPGAs have many dedicated, full-custom, low-power DSP slices, combining high speed with small size, while retaining system design flexibility. Each DSP48A1 slice consists of a dedicated 18 × 18 bit two's complement multiplier and a 48-bit accumulator, both capable of operating at up to 390 MHz. The DSP48A1 slice provides extensive pipelining and extension capabilities that enhance speed and efficiency of many applications, even beyond digital signal processing, such as wide dynamic bus shifters, memory address generators, wide bus multiplexers, and memory-mapped I/O register files. The accumulator can also be used as a synchronous up/down counter. The multiplier can perform barrel shifting.

2.5 Discussion

VHDL has proven to be a standard language describing structure and function of Digital ICs. It offers multiple advantages like ability to decompose a design into sub-designs with their interconnections, provision for simulation of a design before being manufactured, thus reducing the hardware prototype expenses, capability to deal with multiple different architectural bodies belonging to the same entity declaration etc. Elaborate discussion on these aspects has been carried out in the first part of the chapter. Fundamental architecture of FPGA with its three key parts namely Programmable Logic Blocks, I/O Blocks, and Programmable Interconnect have been discussed in second part of the chapter. Attributes of latest Xilinx Spartan 3, Spartan 3AN and Spartan 6 which are used as implementation platform during this research work have been discussed in detail. Subsequent chapters will refer to these discussions while developing the VHDL models of different interleavers including the convolutional one and their efficient FPGA implementations.

Chapter 3

Convolutional Interleaver for D A B

Outline of this Chapter

Design and implementation of convolutional interleaver for DAB application using VHDL/FPGA has been discussed in this chapter. Working principle of convolutional interleaver with progressively increasing delay units has thoroughly been explained. In continuation with the discussion made in Chapter 2, brief introduction to embedded shift register of Xilinx FPGA has been incorporated. The work utilizes FPGA's embedded Shift registers (SRLC16) to model the incremental memory of the interleaver. Such approach helps to reduce the hardware resource requirement of FPGA in addition to reduction of memory wastage over the existing implementation. Exhaustive simulations have been carried out to verify the functionality of the convolutional interleaver. Simulations results have critically been observed and analysed.

3.1. Introduction

ECCs play very important role in modern digital communication systems. BER of the transmitted data can be minimized using a good ECC, of course at the cost of redundancy [80]. Interleaving technique is traditionally used to enhance the quality of digital transmission over a bursty channel [81]. Interleaving is a process to rearrange code symbols so as to spread burst of errors into random like errors and thereafter ECC can be applied to correct them. Interleaving improves [140] the performance of digital transmission at the cost of increased memory requirement, system complexity, and delay. In most of the applications increased memory requirement and system complexity can be accommodated with advancement in technology. However, the increased delay as a result of increased memory requirement may make interleaving a non-practical solution in some applications. So, an interleaver with low delay is a practical way to deal with the error burst.

DAB is an audio broadcasting system in which analog audio signal is converted into a digital signal and transmitted in the assigned Amplitude Modulation (AM) / Frequency Modulation (FM) frequency band. DAB offers compact disc (CD) quality audio on the FM. It is very well suited for mobile reception and provides very high robustness against multipath reception [90]. The working principle of DAB is completely different from that of conventional broadcast system. The fundamental blocks of a DAB transmitter may be described by Fig. 3.1.

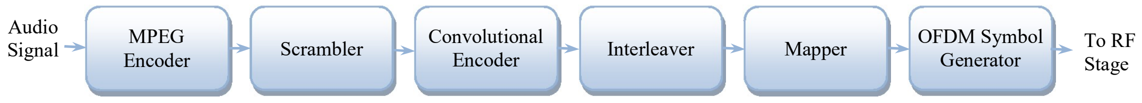


Fig. 3.1 Basic blocks of a DAB transmitter

DAB system uses Moving Pictures Experts Group (MPEG) Audio Layer II encoding. The encoder receives input audio signal in Pulse Coded Modulation (PCM) format, sampled at 48 kHz or 24 kHz, and produces the compressed audio bit stream of different bit rates ranging from 8 kbps to 384 kbps [141]. The next block, scrambler permits the signal to be made available only to the authorized users by incorporating Conditional Access (CA) feature. Punctured convolutional codes with different code rates are used to provide protection against the channel noise. Interleavers are used to save the data frame from error burst arising out of deep fade in the channel. The performance of convolutional code gets improved working in association with interleavers. DAB system uses convolutional interleaver with progressive delay elements as shown in Fig. 3.2. Convolutional interleaver offers dual advantage over block interleaver such as reduced latency and lesser memory requirement. The mapper block converts the interleaved code words into QPSK symbols. The final block is responsible for the generation of OFDM symbols involving the processes of Differential Quadrature Phase Shift Keying (D-QPSK), frequency interleaving, and D-QPSK symbols frequency multiplexing.

The work presented in this chapter is directed towards the efficient FPGA implementation of the convolutional interleaver being used in the DAB application by utilizing embedded FPGA resources. The convolutional interleaver requires progressively increasing memory units to model the delay unit. The author used embedded shift register (SRL16) of Xilinx Spartan 3 FPGA to model the delay unit of the interleaver. Such approach has resulted in two noticeable improvements over external memory based approach [98]: reduction in wastages of memory being used in the delay unit and higher operating speed due to reduced interconnection delay. Initially, 8-bit and 32-bit versions of general purpose convolutional interleavers have been modeled in VHDL and implemented on FPGA using SRL16. Comparison in terms of FPGA slice utilization by these implementations and that without use of SRL16 shows noticeable saving in favour of the former.

3.2. Convolutional Interleaver

A convolutional interleaver [140], [142] consists of N rows of shift registers, with different delay in each row. In general, each successive row has a delay which is J symbols duration higher than the previous row as shown in Fig. 3.2. The zeroth row has no delay elements. The code word symbol from the encoder is fed into the array of shift registers, one code symbol to each row. With each new code word symbol the commutator switches to a new register and the new code symbol is shifted out to the channel. The *i-th* ($1 \leq i \leq N-1$) shift register has a length of $(i-1)J$ stages where $J = M/N$ and the last row has $M-1$ numbers of delay elements.

The convolutional de-interleaver performs the inverse operation of the interleaver and differs in structure of the arrangement of delay elements. Zeroth row of interleaver becomes the $N-1$ row in the de-interleaver. First row of the former becomes $N-2$ row of latter and so on. Minimum end to end delay (t_{ee}) and memory requirement (m_r) due to the convolutional interleaver and de-interleaver pair are

$$t_{ee} = M(N\text{-}1) \text{ code symbol} \quad (3.1)$$

and

$$m_r = M(N\text{-}1) \text{ symbols} \quad (3.2)$$

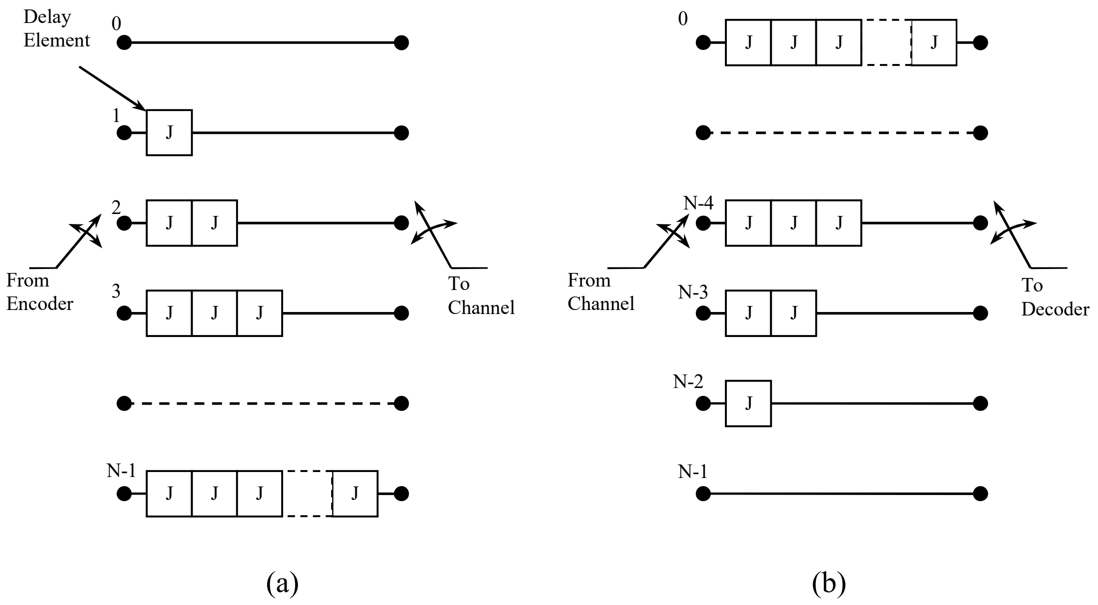


Fig. 3.2 Generic structure of a) Convolutional interleaver b) Convolutional de-interleaver

3.3. Hardware Description of FPGA

In our experimentation, Spartan-3 (device XC3S 400) with 400K gate count FPGA has been used [94]. It has total 896 numbers of configurable logic blocks (CLBs) arranged in 32 x 28 matrix fashion. Each CLB has four slices and two of them are named as SLICEM and rest two as SLICEL. Each of these slices is having logic function generator, flip-flop, multiplexer carry logic and arithmetic gates. Besides these, SLICEM supports two additional functions: storing data using distributed RAM (DRAM) and 16-bit shift register (SRL16). So, total 896 x 2 = 1792 numbers of SRL16 (embedded shift register) are available in addition to other logic resources. DAB application requires a convolutional interleaver of array size [98] of 17 x j (j = 0, 1, …, 11) = 1122 numbers of delay elements. Numbers of SRL16 required to implement the interleaver is 77 which is only 4.3% of available SRL16. Because of our efficient FPGA implementation technique, sufficient FPGA resources are made available for implementing other circuitry of the transmitter/receiver.

An SRLC16 [143] which is cascadable version of SRL16 is constructed from a 4-bit LUT of Xilinx Spartan 3 FPGA. The internal structure of SRLC16 is presented in Fig. 3.3

which is basically a 16-bit shift register but its length can be dynamically varied by changing value in MUX select input (i.e. ADDR). Our proposed design of convolutional interleaver / de-interleaver utilizes the SRLC16 to implement a progressive delay elements of Fig. 3.2(a) and (b).

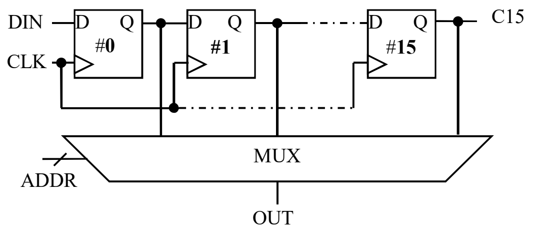


Fig. 3.3 Internal structure of SRLC16.

3.4. Proposed Model of Convolutional Interleaver

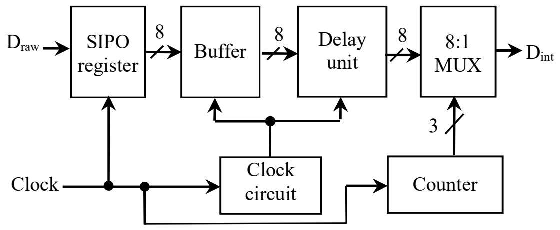


Fig. 3.4 Block diagram of proposed 8-bit convolutional interleaver.

The proposed model of an 8-bit convolutional interleaver with $J = 1$ is presented in Fig. 3.4. The code word symbols (D_{raw}) received in serial form from an encoder is converted into an 8-bit parallel code word by a Serial Input Parallel Output (SIPO) register. The 8-bit code word is then supplied to a delay unit through a buffer register. The SIPO output changes its value with each clock which is not desirable at the input of the delay unit. The buffer unit delivers a word to the delay unit after every 8 clock cycles. The delay unit is comprised of eight rows and is having the structure as narrated in Fig. 3.2(a). Embedded shift registers with casacdable feature, viz. SRLC16, available in Xilinx Spartan 3 FPGA (as described in Section 3.3) have been utilized to model the memory elements with progressively increasing delay unit of the convolutional interleaver. Approaches which do not use such feature (i.e. without SRLC16) require more FPGA slices to model the delay unit. This is because, each slice contains a flip-flop as shown in Fig. 2.5. If the flip-flop of a slice is used, the rest of the resources of that particular slice cannot be used again for some other purpose. Each code symbols of the 8-bit code word is applied to the respective row of the delay unit. The code word gets scrambled with every clocking events (T_x) as it progresses through the delay unit. Table 3.1 shows the scrambling operation of the delay unit where the input code word applied is 11111111_2 before any clock is applied. The subsequent code words are assumed to be 00000000_2 for clarity. The scrambled code word then applied to the input of an 8 line to 1 line multiplexer (MUX) which converts it into stream of serial data (D_{int}). The interleaver circuit requires a clock signal to drive the SIPO register, a clock circuit and a 3-bit counter. The clock circuit basically divides the system clock frequency by 8 which is used to drive the buffer and delay unit. The 3-bit counter generates the select input for the MUX.

Table 3.1 Scrambling Operation in Delay Unit of Convolutional Interleaver

Inputs to Delay Unit		Outputs of Delay Unit
Clock event	$D_7D_6D_5D_4D_3D_2D_1D_0$	$O_7O_6O_5O_4O_3O_2O_1O_0$
Before T_1	11111111	1xxxxxxx
After T_1	00000000	01xxxxxx
After T_2	00000000	001xxxxx
After T_3	00000000	0001xxxx
After T_4	00000000	00001xxx
After T_5	00000000	000001xx
After T_6	00000000	0000001x
After T_7	00000000	00000001

The block diagram representation of the de-interleaver is exactly similar to Fig. 3.4 except the use of delay unit of Fig. 3.2(b) in place of Fig. 3.2(a). The functional description of interleaver and de-interleaver can easily be extended to higher number of bits with or without higher values of *J*.

3.5. VHDL Modeling

This section describes the VHDL modeling [92] of an 8-bit interleaver, de-interleaver and the interleaver & de-interleaver pair together using Xilinx ISE software [144] and is presented in the form of flow charts.

3.5.1 Interleaver

In Fig. 3.5, the entity of the interleaver model contains D_IN (input code word stream) and CLK (clock) as input signal and D_OUT (scrambled code word stream) as output signal. The input code word stream enters the SIPO_I block one bit at a time in synchronization with clock. The CLK signal is also read by two VHDL programs; one for generating CLK8 (= CLK÷8) synchronization signal and the other for generating COUNT3BIT, functions as select input signal to MUX. BUFFER_I is another VHDL program to implement the 8-bit buffer register and is synchronized by CLK÷8 signal. The output from the BUFFER_I block is supplied to DEL_UNIT_I block, a VHDL program to realize the delay unit required in the interleaver. This is the heart of the interleaver. It consists of seven VHDL program internally to implement the variable length shift registers. SRLC16s have been utilized to model the variable length shift register. It is synchronized by CLK8 signal. The output of the DEL_UNIT_I block is supplied to the VHDL program to implement 8:1 MUX (MUX_8x1) which converts the 8-bit scrambled code word into serial stream of code symbols and is finally taken out from D_OUT line.

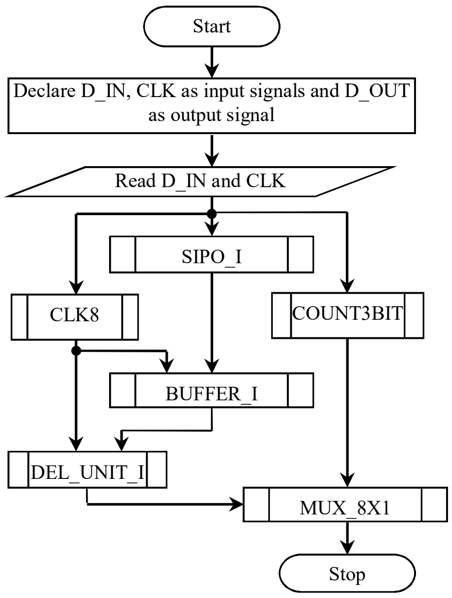


Fig. 3.5 Flow chart of 8-bit Convolutional Interleaver

3.5.2 De-interleaver

Externally the VHDL model of the 8-bit convolutional de-interleaver is identical to that of the interleaver. But internally the two models differ in the structure of the delay unit (for de-interleaver it is DEL_UNIT_D). The shift register for row N-1 in DEL_UNIT_I is used in zeroth row of the DEL_UNIT_D. Similarly shift register of N-2 row in DEL_UNIT_I is connected to 1st row of the DEL_UNIT_D and so on. The delay unit of proposed de-interleaver model also utilizes SRLC16 owing to its advantage in saving FPGA slices in a similar manner as done for the proposed convolutional interleaver.

3.5.3 Interleaver – De-interleaver pair

This section describes about a VHDL model prepared by combining the proposed convolutional interleaver and de-interleaver described in previous two sections. The objective of such combination is to verify the functionality of proposed interleaver and de-interleaver models. As the two constituents of this combined model utilize SRLC16, the advantage of lesser resource utilization of FPGA is also available in the combined model too. The combined VHDL model is designated as INTERLEAVER_DEINTERLEAVER and is presented in the form of flowchart in Fig. 3.6. INTERLEAVER and DEINTERLEAVER are the designations used to refer our proposed interleaver and de-interleaver in Fig. 3.6. The INTERLEAVER block receives raw data from input source which get spread out when progressing through it. The scrambled code words from the output of the INTERLEAVER are then applied as input to the DEINTERLEAVER block along with CLK as synchronization signal. It has been observed that the scrambled code word is converted into its original (raw) form at the output of the DEINTERLEAVER block thus verifying the functionality of the proposed convolutional interleaver and de-interleaver. Simulation results in Section 3.6 present such verification. The author has repeated the entire design of proposed convolutional INTERLEAVER, DEINTERLEAVER and the INTERLEAVER_DEINTERLEAVER pair using SRLC16 as discussed in this section (Section 3.5) with 32-bit word length whose FPGA implementation results are used in Section 3.7 for the purpose of comparison.

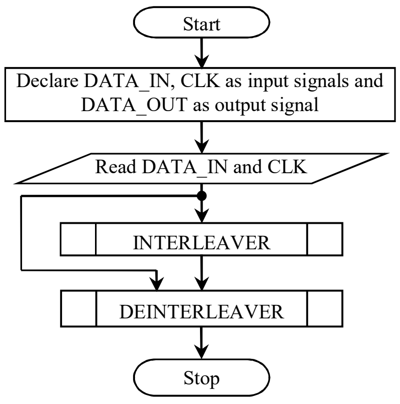
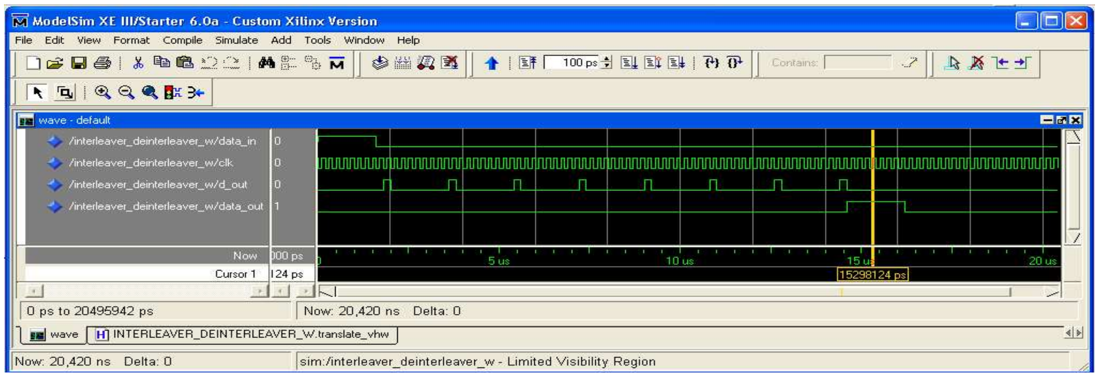

Start
Declare DATA_IN, CLK as input signals and DATA_OUT as output signal
Read DATA_IN and CLK
INTERLEAVER
DEINTERLEAVER
Stop

Fig. 3.6 VHDL model of 8-bit Convolutional INTERLEAVER_DEINTERLEAVER pair.

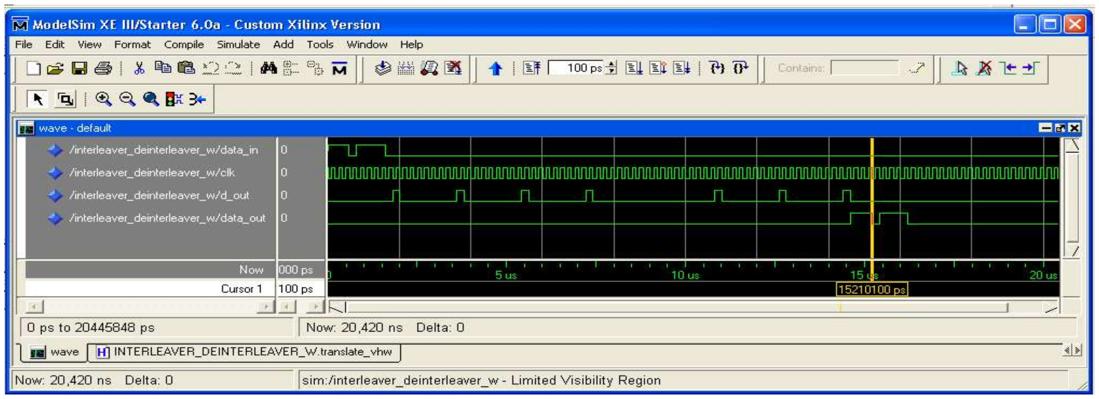
3.6 Simulation Result

This section verify the functionality of the proposed convolutional interleaver and de-interleaver (8-bit) using timing simulation obtained from ModelSim Xilinx Edition-III, version 6.0a shown in Fig. 3.7(a) and (b). The system clock frequency applied to the model is 5 MHz for simulation. Input set up time and output valid delay time are chosen to be 10ns each. The 8-bit (=11111111_2) input signal is applied at data_in input of the interleaver as shown in Fig. 3.7(a). This data word when passes through the interleaver gets scrambled and can be observed in Fig. 3.7(a) at d_out. This clearly verify the interleaver operation taking place in the convolutional interleaver. In addition, the timing diagram verify the operation of convolutional de-interleaver as well. The interleaved code word as available at the output of interleaver (at d_out) is applied as input to the de-interleaver which rearranges them in such a way that the original code word is generated at its output (data_out). Figure 3.7(b) further endorses the working of convolutional interleaver and de-interleaver with input code word = 11101111_2.


ModelSim XE III/Starter 6.0a - Custom Xilinx Version
File Edit View Format Compile Simulate Add Tools Window Help
100 ps
Contains:
wave - default
5 us
10 us
15 us
20 us

/interleaver_deinterleaver_w/data_in 0
/interleaver_deinterleaver_w/clk 0
/interleaver_deinterleaver_w/d_out 0
/interleaver_deinterleaver_w/data_out 1
Now 000 ps
Cursor 1 124 ps
15298124 ps
0 ps to 20495942 ps
Now: 20,420 ns Delta: 0
wave INTERLEAVER_DEINTERLEAVER_W.translate_vhw
Now: 20,420 ns Delta: 0
sim:/interleaver_deinterleaver_w - Limited Visibility Region

(a)


ModelSim XE III/Starter 6.0a - Custom Xilinx Version
File Edit View Format Compile Simulate Add Tools Window Help
100 ps
Contains:
wave - default
5 us
10 us
15 us
20 us

/interleaver_deinterleaver_w/data_in 0
/interleaver_deinterleaver_w/clk 0
/interleaver_deinterleaver_w/d_out 0
/interleaver_deinterleaver_w/data_out 0
Now 000 ps
Cursor 1 100 ps
15210100 ps
0 ps to 20445848 ps
Now: 20,420 ns Delta: 0
wave INTERLEAVER_DEINTERLEAVER_W.translate_vhw
Now: 20,420 ns Delta: 0
sim:/interleaver_deinterleaver_w - Limited Visibility Region

(b)

Fig. 3.7 Simulation result with (a) input code word = 11111111_2 and (b) input code word = 11110111_2

3.7 Analysis of FPGA Implementation Results

The VHDL model of Convolutional interleaver-de-interleaver pairs (both 8-bit and 32-bit) are implemented and tested into Xilinx Spartan-3 (Device: XC3S400) FPGA platform in the laboratory. The FPGA implementation of the convolutional interleaver-de-interleaver pair without SRLC16 feature is a very hardware-intensive application in comparison with SRLC16. Implementation without SRLC16 involves slice flip-flop to model the delay unit of the interleaver/de-interleaver. As shown in Fig. 2.5, when the flip-flop of a slice is used, the 4-input LUT of the slice remain unutilised leading to wastage of FPGA resources. Table 3.2 shows a comparative analysis of the FPGA resource requirement in the delay units of interleaver and de-interleaver taken together for the two implementations - with and without SRLC16 for both 8-bit and 32-bit versions.

Table 3.2 Comparative Analysis between Various Implementations

Interleaver word length	1-bit delay units required	FPGA slices required		Slice saving in %
		without SRLC16	with SRLC16#	
8-bit	8 x 7 = 56	56 ÷ 2 = 28	14	50.00 %
32-bit	32 x 31 = 992	992 ÷ 2 = 496	92	81.45 %

# Proposed technique.

Table 3.2 clearly signifies that our proposed implementation technique of convolutional interleaver and de-interleaver pair with SRLC16 saves 50 % and above 81 % of FPGA resources compared to the flip-flop based technique without SRLC16 for 8-bit and 32-bit cases respectively. Use of lesser slices leads to reduced delay in the interconnection network inside the FPGA. This further implies reduction in power consumption too.

Table 3.3 makes comparison of our proposed technique with Kim *et. al* [98] in the issue of reduction in memory wastage for interleaver implementation. Pictorial representation of Table 3.3 is provided through a bar chart in Fig. 3.8 including comparison between General Structure [98], Kim *et. al* and our proposed technique highlighting row wise wastage of memory bits. Our proposed work shows significant reduction in memory wastage issue over the general structure in all most all rows. This technique results in significant saving of memory bits in row no. 1 and 2 in comparison with [98]. In other rows, work in [98] has performed better mostly due to merging of rows which may lead to more complexity in addressing of the memory. However, our proposed technique reduces overall memory wastage by 30.38 % for DAB application over [98]. As shown in Table 3.3, Row no. 1 requires 17-bit delay units which is modeled using two 16-bit SRLC16 leading to non-utilization of 15-bits of the 2nd SRLC16. Similarly, to model the 34-bit delay units for Row no. 2, three numbers of SRLC16 are required. In this case, 14-bits of 3rd SRLC16 remain utilized. In similar line, with increase in row nos., the number of unutilized bits get progressively reduced.

Another bar chart comparison between the three techniques with respect to memory wastage factor is shown in Fig. 3.9. The latter chart is normalized against our proposed technique. It is evident from Fig. 3.9 that our proposed technique is most efficient as far as overall memory wastage is concerned. In addition, obviously the access time of embedded shift register is lower than that of external memory used in [98].

Table 3.3 Comparative analysis with respect to memory wastage

row no.	1-bit delay units required	using Kim et. al. technique		our proposed technique	
		RAM size	wasted memory	no. of SRLC16 required	wasted memory
1	17	128 (R1+R3)	60	2	15
2	34	256 (R2+R8)	86	3	14
3	51	merged with R1	---	4	13
4	68	256 (R4+R11)	1	5	12
5	85	256 (R5+R10)	1	6	11
6	102	256 (R6+R9)	1	7	10
7	119	128	9	8	9
8	136	merged with R2	---	9	8
9	153	merged with R6	---	10	7
10	170	merged with R5	---	11	6
11	187	merged with R4	---	12	5
Total wastage			158		110

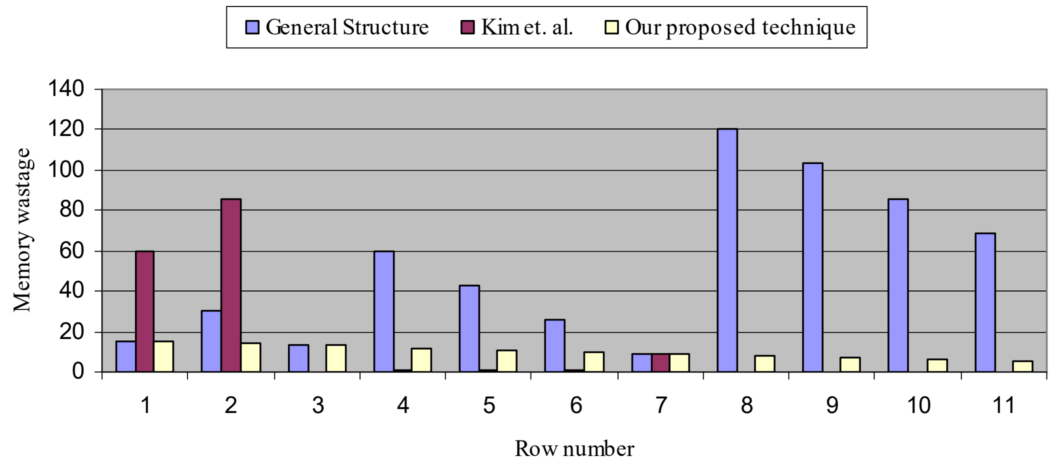


Fig. 3.8. Bar chart showing row wise memory wastage of the three implementation techniques

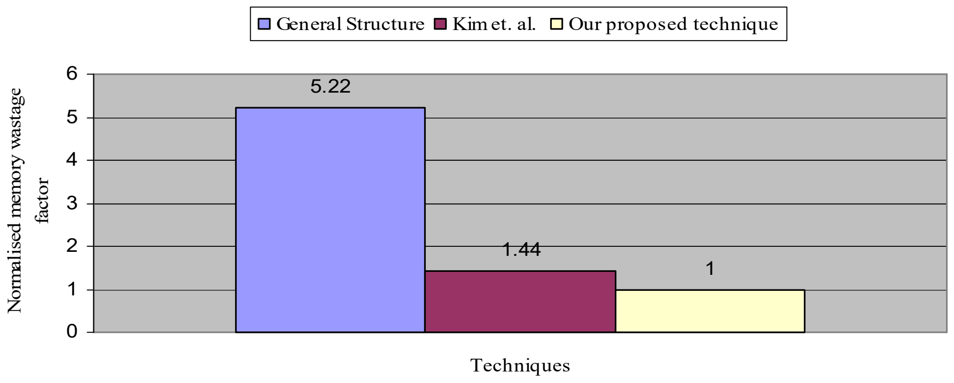


Fig. 3.9 Memory wastage factors of the three implementation techniques

The HDL Synthesis Report and Device utilization summary generated using XST (Xilinx Synthesis Technology), version G. 35, a Xilinx tool that synthesizes HDL designs for the VHDL models (both 8-bit and 32-bit) are given in Table 3.4 and 3.5 respectively. The 32-bit design needs two 5-bit adders in the 5-bit counters of interleaver and de-interleaver each. As evident from Table 3.4 that 8-bit and 32-bit interleaver and de-interleaver pair needs 14 and 92 numbers of SRLC16, which matches with Table 3.2. Other registers are required for constructing SIPOs, internal storage in counters and in clock circuits.

Table 3.4 HDL synthesis report

For 8-bit		For 32-bit	
# Adder/Subtractor	2	# Adder/Subtractor	2
3-bit adder	2	5-bit adder	2
# Registers	28	# Registers	80
1-bit register	24	1-bit register	76
8-bit register	2	32-bit register	2
3-bit register	2	5-bit register	2
# Shift Registers	14	# Shifter Register	92
SRLC16_1	14	SRLC16_1	92
# Multiplexer	2	# Multiplexer	2
8-to-1 multiplexer	2	32-to-1 multiplexer	2

The Device Utilization Summary shows that the Convolutional interleaver and de-interleaver pair uses very few FPGA resources thus making room for other associated circuitry to be implemented on the same FPGA chip. The estimated power consumption of the 32-bit model is found to be 125mW (using Xilinx XPower SoftwareVersion:G.35) making the circuit suitable for battery powered applications also.

Table 3.5 Device utilization summary

Selected Device : 3s400pq208-5		
FPGA Resources	For 8-bit	For 32-bit
Number of Slices:	31 out of 3584 (0.86%)	133 out of 3584 (3.71%)
Number of Slice Flip Flops:	46 out of 7168 (0.64%)	151 out of 7168 (2.11%)
Number of 4 input LUTs:	34 out of 7168 (0.47%)	146 out of 7168 (2.04%)
Number of bonded IOBs:	3 out of 141 (2.12%)	3 out of 141 (2.12%)
Number of GCLKs:	1 out of 8 (12.5%)	1 out of 8 (12.5%)

3.8 Discussion

This chapter emphasized the use of convolutional interleaving techniques to maintain data fidelity against burst errors in digital communication. An efficient design of convolutional interleaver and de-interleaver utilising SRLC16 of Xilinx FPGA has been proposed. VHDL model of the proposed design is prepared using Xilinx ISE and is implemented on Spartan 3 FPGA. Simulation results in the form of timing diagram obtained using ModelSim software is presented which verify the functionality of the proposed interleaver design. Reduction in FPGA resource utilization up to 81 % compared to other implementation technique has been recorded due to our efficient design utilising SRLC16. Lesser power consumption and reduced FPGA interconnection delay are the obvious implications of this technique. It also lowers the overall memory wastage by 30 % compared to a popular implementation technique for DAB application. Encouraged with the results obtained while implementing convolutional interleaver, the researcher undertook the design of block interleaver for WLAN application using the same FPGA platform. The design issues for the block interleaver are presented in the next chapter.

Chapter 4
Interleaving in WLAN

Outline of this Chapter

After successful implementation of convolutional interleaver with embedded shift register SRLC16 on FPGA platform providing better results in terms of lesser memory wastage and lesser hardware resources, effort has now been made to develop improved design and efficient implementation of block interleaver used in IEEE 802.11 a/g based WLAN. This chapter initially describes the necessary background of the block interleaver used in WLAN transceiver. In this thesis work two approaches namely improved LUT based and Finite State Machine based have been followed in designing the hardware for the block interleaver. The former technique demonstrates reduction in resource utilization like slices, flip-flop and LUTs over the conventional LUT based approach. Similar results are also obtained for FSM based implementation but with faster performance.

4.1. Introduction

BWA is the most challenging segment of the wireless revolution since it has demonstrated a viable alternative to the cable modem and digital subscriber line in the last mile access environment [145]. High processing speed, design flexibility and fast design TAT are the important requirements of BWA to meet the challenges poised to it. These requirements force the designers to choose reconfigurable hardware platform like FPGA. A product implemented on FPGA can easily be upgraded by making necessary changes in the HDL code and thus becomes obsolescence free. In addition, the TAT of FPGA based circuits is almost instantaneous meaning prototyping and physical validation of a digital design in real world conditions, when compared to the weeks-long wait required to manufacture the design as ASIC [146].

A WLAN interconnects two or more communicating devices using some wireless distribution method and usually provides a connection through an AP [44] to the wider internet. During the past few years, the IEEE 802.11 WLAN has emerged as a prevailing broadband indoor wireless networking technology [145]. IEEE 802.11a [74] and IEEE 802.11g [147] based WLAN use OFDM [67] PHY layer that greatly increases the overall throughput at the AP. OFDM technique is gaining popularity due to its high transmission capability and also for alleviating the adverse effects of ISI and ICI.

In many communication channels, two types of errors namely random and burst occur. Random errors are dealt with FEC codes like Convolutional, Turbo etc. A burst error or error burst is a contiguous sequence of bits or symbols, received in erroneous condition over a data transmission channel. Burst error causes performance degradation of the communication system with increased value of BER. Re-transmission of erroneous frames using conventional techniques like Automatic Repeat Request (ARQ) may be employed but may not be suitable in many applications as it demands duplex channel [148]. The effect of burst error is more efficiently mitigated by interleaving technique [24]. Interleaving [80] plays a vital role in improving the performance of FEC codes in terms of BER. Interleaving is a process to rearrange code symbols so as to spread the burst of errors into random like errors. Hence FEC techniques could be applied to correct them. Block interleaving [19] is one of the widely used techniques in which the bits received from the encoder are stored row wise in the interleaver's own memory and read column wise. WLAN based on IEEE 802.11a and IEEE 802.11g uses special type of block interleaver [102] of various specifications depending on the modulation type to combat error burst.

In this chapter, we describe two techniques involving LUT and FSM to model multimode interleaver for OFDM based WLAN. As per IEEE 802.11a and IEEE 802.11g standard, ½, ⅔ and ¾ are the allowed code rates whereas BPSK, QPSK, 16-QAM and 64-QAM are the permitted modulation schemes. Our work includes multimode interleaver design on Xilinx Spartan 3 FPGA with all possible modulation scheme permitted as per [44],[74]. The address generator of the interleaver is governed by two equations which includes complex functions like modulus and floor. Due to the absence of corresponding digital hardware for these functions, hardware design of the interleaver is more challenging. Moreover, VHDL does not support such functions directly as well. Consequently, the LUT based technique is conventionally used in which the 'address LUTs' are usually housed in external memory. Use of external memory makes the design slower due to long memory access time. Our work describes improved LUT based technique employing FPGA's internal memory to house the addressing LUTs. Memory partitioning is employed to reduce the memory wastage. As a result, the proposed LUT based technique shows better performance in terms of operating frequency with efficient resource utilization. On the contrary, the FSM based technique shows two different approaches involving BRAM and DRAM of FPGA to model the interleaver memory. Critical analysis of the results of FPGA implementation including software simulation of both approaches has been made.

4.2. Interleaving in WLAN

IEEE 802.11a and IEEE 802.11g based WLAN uses identical interleaving technique in which a special type of block interleaver [102] is used. Specification of a block interleaver is referred as it interleaver depth, computed by multiplying number of rows with number of columns of the memory block used as the block interleaver. The interleaver depth varies with modulation scheme. The interleaver action can be expressed in terms two sets of equations which ensures the following two design rules:

i) The adjacent coded bits are mapped into non-adjacent subcarriers.
ii) Adjacent coded bits are mapped alternately into less and more significant bits of the constellation to avoid long run of lowly reliable bits.

Let N_{cbps} is the block size corresponding to the number of coded bits per allocated sub-channels per OFDM, d represents number of columns of the block interleaver which is typically chosen to be 16 [44]. m_k is the output after first level of permutation and k varies from 0 to N_{cbps}-1. s is a parameter defined as s=max{1, $N_{cpc}/2$}, where N_{cpc} is the number of coded bits per sub-carrier as shown in Table 4.1.

$$m_k = (\frac{N_{cbps}}{d})(k \% d) + \left\lfloor \frac{k}{d} \right\rfloor \tag{4.1}$$

$$j_k = sx\left\lfloor \frac{m_k}{s} \right\rfloor + (m_k + N_{cbps} - \left\lfloor \frac{d\,x\,m_k}{N_{cbps}} \right\rfloor)\% s \tag{4.2}$$

where % and $\lfloor\ \rfloor$ signify modulo and floor functions respectively.

Table 4.1 Specifications of IEEE 802.11a and IEEE 802.11g based WLAN Interleaver

Modulation Scheme	N_{cpc}	s	N_{cbps}	No. of Rows in interleaver memory
BPSK	1	1	48	3
QPSK	2	1	96	6
16-QAM	4	2	192	12
64-QAM	6	3	288	18

4.3 Modeling Memory in FPGA

SRAM based FPGAs [149] offer internal (embedded) storage for potential applications like local storage, FIFO, data buffers, stack, large LUT etc. Xilinx offers two types of such internal storage called Distributed RAM (DRAM) and Block RAM (BRAM) in its FPGAs [94],[150].

4.3.1 Distributed RAM

In our experimentation we have used Xilinx Spartan-3 FPGA (device XC3S400) [94] having 896 CLBs. Each CLB contains four slices and each slice contains two LCs. Each LC contains a 4-input LUT. The LUT performs any possible logic function of its four inputs and forms the basis of the Spartan-3 logic architecture. Two slices of a CLB are termed as SLICEM and the other two as SLICEL as shown in Fig. 4.1. The two LCs of a SLICEM slice contain storage elements and can be utilized as two 16 x 1 bit DRAM in addition to using it as 16-bit shift register (SRL16) or only as logic generator. The LCs of slice, SLICEL can be used as ROM/logic generator. Each 16 x 1 RAM can be cascaded for deeper and wider memory applications. Spartan-3, Device XC3S400 FPGA offers 56Kbits of DRAM.

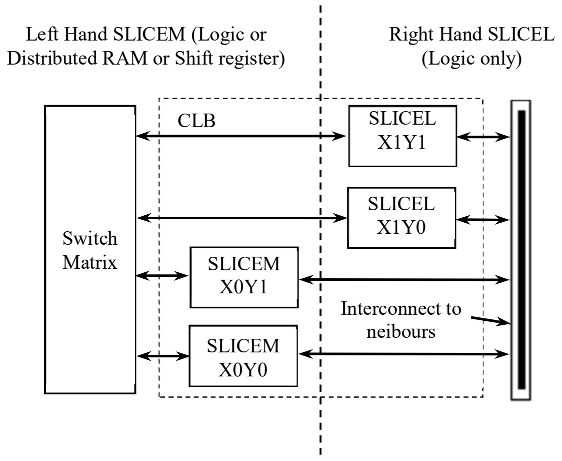

Left Hand SLICEM (Logic or Distributed RAM or Shift register)
Right Hand SLICEL (Logic only)
CLB
Switch Matrix
SLICEL X1Y1
SLICEL X1Y0
SLICEM X0Y1
SLICEM X0Y0
Interconnect to neibours

Fig. 4.1 Internal structure of a CLB in Spartan 3 FPGA

4.3.2 Block RAM

The Block RAM available in Spartan 3 FPGA can be configured to work as single port or dual port memory. Single port memory can be either be read or written depending on the control signal but not simultaneously. Dual port BRAM has the advantage of performing both read as well as write operation on a single memory block simultaneously using two different ports as shown in Fig. 4.2. Table 4.2 lists all the interface signals of a dual port BRAM, their direction and the port to which they are associated with. Pictorial view and interface signals of single port BRAM are implied from Fig 4.2. In our experimentation we have used Xilinx Spartan-3 FPGA (device XC3S400) [94] having 16 nos. of 18KB (16KB data and 2KB parity) memory size each. The memory blocks can be organized in various ways as shown in Table 4.3 using VHDL programming. The proposed design uses single port memory with 1K x 16-bit organization to store interleaver addresses whereas interleaver memory is modeled using a dual port BRAM with 16K x 1-bit organization.

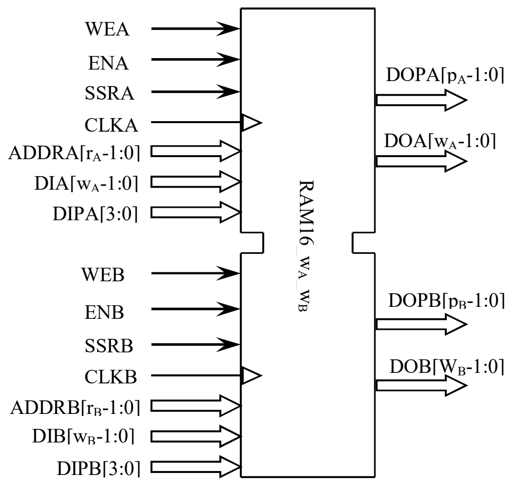


Fig. 4.2 Dual port BRAM in Xilinx FPGA

Table 4.2 Dual port BRAM interface signal

Signal Function	Port A	Port B	Direction	Brief description
Data Input Bus	DIA	DIB	Input	The memory block receives input data to be written in the selected location of Port A / Port B through these lines.
Parity Data Input Bus	DIPA	DIPB	Input	The memory block receives parity data input to be written in the selected location of Port A / Port B through these lines.
Data Output Bus	DOA	DOB	Output	The memory block transmits data from a selected location of Port A / Port B through these lines.
Parity Data Output	DOPA	DOPB	Output	The memory block transmits parity data from a selected location of Port A / Port B through these lines.
Address Bus	ADDRA	ADDRB	Input	Through these lines, a memory location is addressed for either read or write operation of Port A / Port B.
Write Enable	WEA	WEB	Input	This signal when made active (logic 1) permits the data write operation in a selected memory location of Port A / Port B.
Clock Enable	ENA	ENB	Input	This signal when made active (logic 1) enables the memory block. This signal can be treated as master control of the memory block.
Synchronous Set/Rest	SSRA	SSRB	Input	The synchronous set/reset input, SSR, forces the data output latches to the value specified by the SRVAL attribute. When SSR and the enable signal, EN, are High, the data output latches for the DO and DOP outputs are synchronously set to a '0' or '1' according to the SRVAL parameter.
Clock	CLKA	CLKB	Input	This signal clocks Port A / Port B for all synchronous operations. Clock polarity is configurable and is rising edge triggered by default.

Table 4.3 Organization of BRAM in Spartan 3 FPGA

Total RAM bits, including parity	18,432 (16K data + 2K parity)
Memory Organizations	16Kx1 8Kx2 4Kx4 2Kx8 (no parity) 2Kx9 (x8 + parity) 1Kx16 (no parity) 1Kx18 (x16 + 2 parity) 512x32 (no parity) 512x36 (x32 + 4 parity) 256x72 (single-port only)

4.4. Hardware Models of Interleaver

The proposed hardware models of OFDM based WLAN interleaver consist of two sections: address generator and interleaver memory as shown in Fig. 4.3. The address generator is basically the simultaneous implementation of (4.1) and (4.2) which is the write address along with provision for generation of read address for interleaver memory. Block interleaver uses two memory blocks out of which one memory block is written and the other is read based on the value of select (sel) signal. In this work, two different interleaver design approaches namely LUT and FSM based for IEEE 802.11a and IEEE 802.11g WLAN have been proposed. In both approaches, a MATLAB program is developed implementing (4.1) and (4.2) to generate the interleaver write addresses. Part of such addresses (first 32) with four different modulation schemes are shown in Table 4.4.

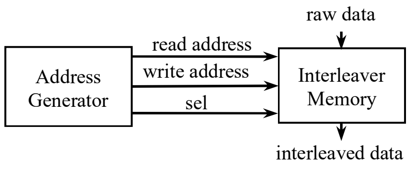


Fig. 4.3 Top level view of interleaver

Table 4.4 First 32-Write Addresses for Four Modulation Schemes and Their Encoding

N_{cbps}=48 bits, BPSK (mod_typ =00)	0	3	6	9	12	15	18	21
	24	27	30	33	36	39	42	45
	1	4	7	10	13	16	19	22
	25	28	31	34	37	40	43	46
N_{cbps}=96 bits, QPSK (mod_typ =01)	0	6	12	18	24	30	36	42
	48	54	60	66	72	78	84	90
	1	7	13	19	25	31	37	43
	49	55	61	67	73	79	85	91
N_{cbps}=192 bits, 16-QAM (mod_typ =10)	0	13	24	37	48	61	72	85
	96	109	120	133	144	157	168	181
	1	12	25	36	49	60	73	84
	97	108	121	132	145	156	169	180
N_{cbps}=288 bits, 64-QAM (mod_typ =11)	0	20	37	54	74	91	108	128
	145	162	182	199	216	236	253	270
	1	18	38	55	72	92	109	126
	146	163	180	200	217	234	254	271

4.4.1 LUT based Interleaver

In this approach of interleaver design, the write addresses are pre-computed implementing (4.1) and (4.2) through the MATLAB program described in the form of flow chart in Fig. 4.4 and stored in LUTs. The program accepts N_{cbps}, s, k and d (defined in Section 4.2) as inputs, computes one set of values of b, c, g, m_k, a, e, h, f, i and j_k in every iteration till k < N_{cbps} as described in Fig. 4.4. The values of j_k obtained in every iteration represents the interleaver addresses which are to be stored in respective LUTs. As shown in Table 4.1, WLAN supports four different interleaver depths one for each modulation scheme with the dimension described there. Conventionally, four separate memory modules are required to house these four LUTs. In this work, the author has used embedded single and dual port BRAM memory available in the target FPGA. One of such single port BRAM module having dimension 1K x 16-bit is partitioned as shown in Fig. 4.4 to model these LUTs. The first partition having the address range 0-02FH holds the interleaver addresses with depth N_{cbps} = 48-bits. Similarly, the interleaver addresses with N_{cbps} = 96-bits are also stored in memory locations with address ranges 030H-08FH and so on. An up-counter is used to read these addresses from the appropriate LUT, based on the value of N_{cbps} stored in the variable MOD_TYP as shown in Table 4.5. Partitioning of the memory eliminates the need of four different memory blocks to model the four LUTs. Table 4.5 shows the complete range of addresses of each LUT inside a single port BRAM.

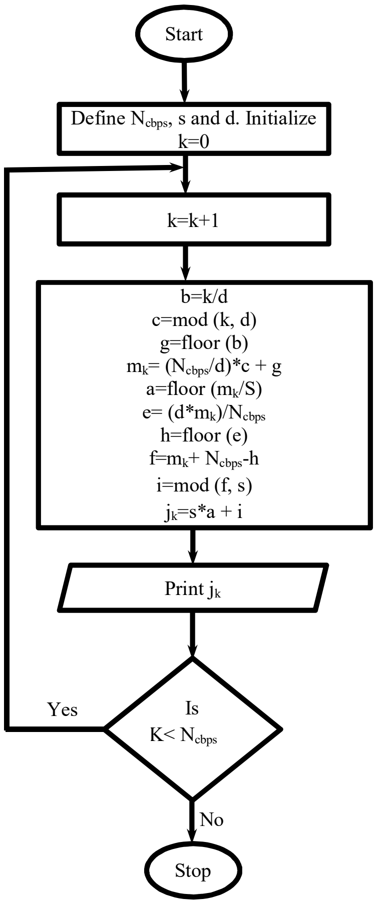


Fig. 4.4 Flow chart of MATLAB program used to pre-compute WLAN interleaver addresses

The memory requirement of block interleaver has been modelled using a dual port BRAM of FPGA in the proposed work. The use of memory is optimized in the sense that one dual port memory have been utilized to model the two memory blocks required in OFDM based WLAN block interleaver. As shown in Fig. 4.6, the memory block is configured in such a manner that when one port is in write mode the other is in read mode and vice versa. The first memory module occupies 0-287 bit locations whereas the other module is placed from 512 to 799 bit location in the 16K x 1 dual port BRAM.

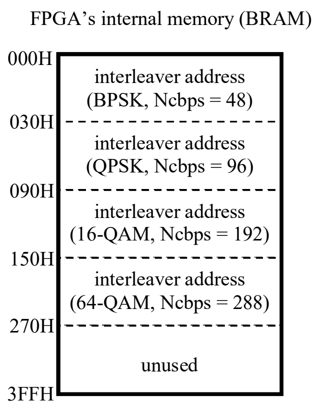

FPGA's internal memory (BRAM)
000H
interleaver address (BPSK, Ncbps = 48)
030H
interleaver address (QPSK, Ncbps = 96)
090H
interleaver address (16-QAM, Ncbps = 192)
150H
interleaver address (64-QAM, Ncbps = 288)
270H
unused
3FFH

Fig. 4.5 Modeling of LUT in FPGA's internal memory

Table 4.5 Address Ranges of Various LUT Inside BRAM

Interleaver Depth (N_{cbps})	Modulation type (MOD_TYP)	Address Range (10 bit)	Memory Size in bit
48	BPSK (00B)	000h-02FH	48x16
96	QPSK(01B)	030h-08FH	96x16
192	16-QAM (10B)	090h-14FH	192x16
288	64-QAM (11B)	150h-26FH	288x16

The interleaver action can be described in terms of a local FSM as shown in Fig. 4.7. This FSM enters in the START state on reset. Based on the value of modulation type (mod_typ) the counter is initialized to a preset value (e.g. for mod_typ = 01B, counter_preset = 030H; for mod_typ = 11B, counter_preset = 150H etc.). The FSM thereafter allows the counter to progress through its natural count sequence till the terminal value. On reaching the terminal value the FSM alters the state of rw_sel signal causing the read and write mode of two memory blocks to swap. The counter gets auto initialized to its respective preset value and then starts counting up again.

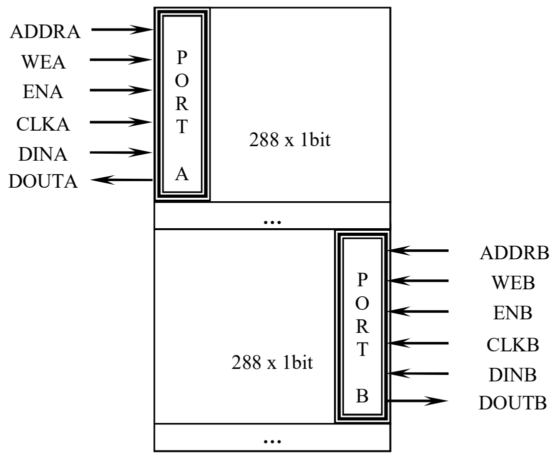


Fig. 4.6 Modeling of interleaver memory using dual port BRAM in FPGA

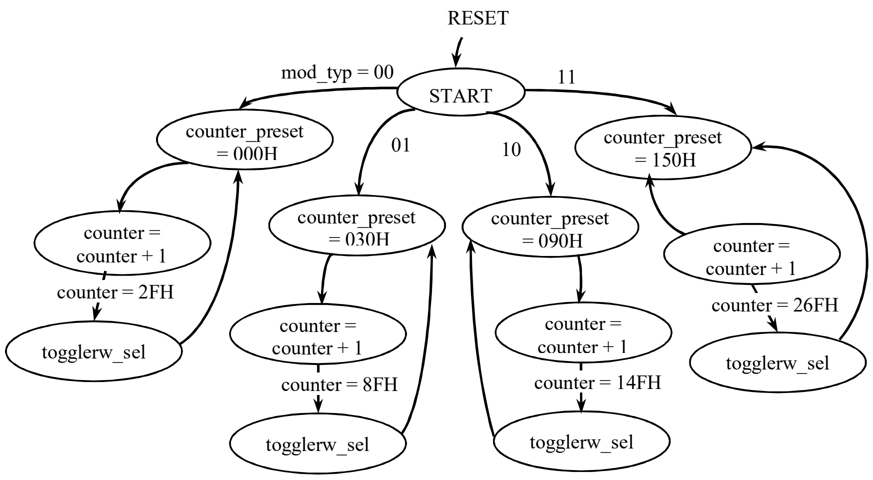


Fig. 4.7 State diagram representation view of proposed interleaver

Fig. 4.8 shows the detailed picture of the proposed interleaver. The Address Generator Block (AGB) is responsible for generating two types of addresses, one for writing and the other for reading the interleaver memory in Interleaver Memory Block (IMB). The read addresses (rd_address) are obtained from a 10 bit counter. In order to generate the write addresses the appropriate LUT for a mod_typ is to be read as per Table 4.5. The 10-bit counter output is added with a preset value in the adder A_1. The appropriate preset value is selected by a multiplexer (M_1) based on the value of mod_typ as shown in Fig. 4.7. For example, with mod_type = 01B, preset = 030H which is the starting address of the LUT with interleaver depth 96-bits. The counter gets reset signal with the help of a comparator after the last address of the said LUT is read. With mod_typ = 01B, the multiplexer (M_2) attached to the comparator select the input value 05FH. The comparator generates a high reset pulse when the counter output reaches 05FH.

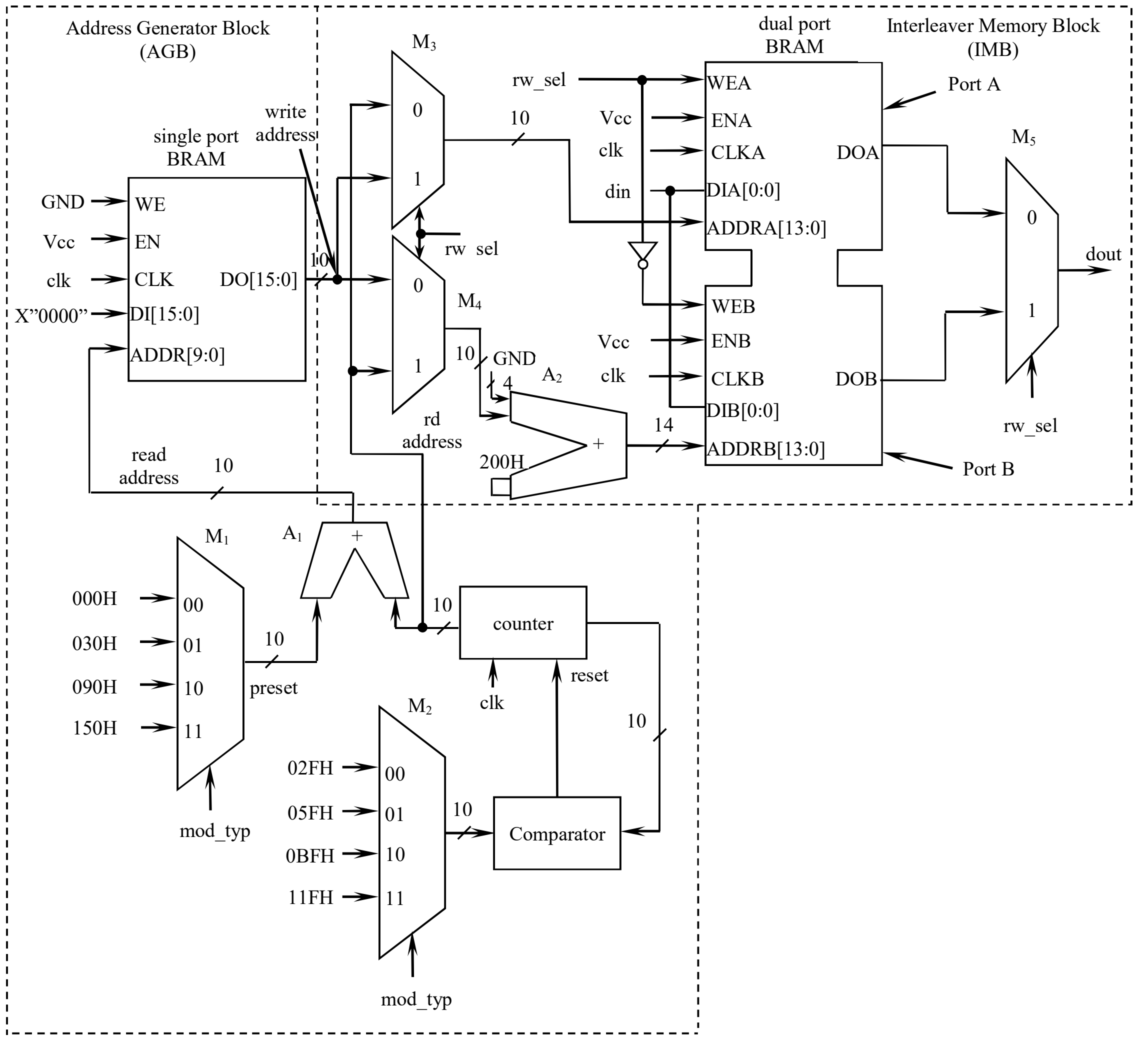


Fig. 4.8 Detailed view of proposed LUT based interleaver

The lower 10-bit of the output received from the single port BRAM is used as the write address of the interleaver. The multiplexers (M_3 and M_4) attached to the dual port BRAM with rw_sel input sends the read and write addresses to the appropriate ports. For example, when rw_sel = 1, Port A receives write address where as Port B gets the read address. When writing/reading of Port A/Port B is over, status of rw_sel gets changed to 0. As a result Port A now receives read address and Port B receives write address. This phenomenon has been explained with the help of timing diagram in Fig. 4.9. In this manner every time when rw_sel status gets changed, the read/write operation between Port A and Port B gets swapped. A toggle flip-flop is used to generate the rw_sel signal and is synchronized to the reset input of the counter. As shown in Fig. 4.6, the two memory blocks required for interleaver memory are housed in a single dual port BRAM of Spartan 3 FPGA with a capacity of 1K x 16-bit. The first memory block is placed through Port A within the address space of 0-287. A bias value of 200H needs to be added to the read/write address of Port B as the second memory block is placed in the address space of 512-800 with starting address 200H. The multiplexer (M_5) attached to the dual port BRAM output sends the interleaved data bit out from the memory block being read.

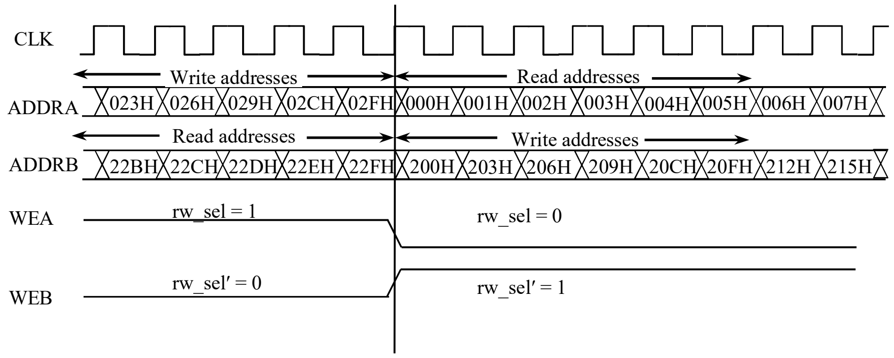


Fig. 4.9 Timing diagram showing swapping of read/write operation between Port A and Port B using rw_sel

4.4.2 FSM based Interleaver

This section describes FSM based hardware interleaver especially the address generator design for IEEE 802.11a and IEEE 802.11g based WLAN. Careful examination of the write addresses in Table 4.4 reveals that the subsequent addresses are not equally spaced for all the cases. Within a particular modulation scheme, the increment values follow a fixed type of pattern. In case of BPSK and QPSK (with s = 1) the increments are linear having the values 3 and 6 respectively. 16-QAM and 64-QAM have nonlinear increments e.g. 13, 11 and 20, 17, 17 respectively.

Our proposed design of address generator block is described in the form of schematic diagram in Fig. 4.10. Bulk of the circuitry is used for the generation of write address. It contains three multiplexers (muxs): mux-1 and mux-2 implement the unequal increments required in 16-QAM and 64-QAM whereas mux-3 routes the outputs received from mux-1 and mux-2 along with equal increments of BPSK and QPSK. The select input of mux-1 is driven by a T flip-flop named qam16_sel whereas that of mux-2 is controlled by a mod-3 counter, qam64_sel. The two lines of mod_typ (modulation type) are used as select input of mux-3. The 6-bit output from the mux-3 acts as one input of the 9-bit adder after zero padding. The other input of the adder comes from accumulator, which holds the previous address. After addition a new address is written in the accumulator.

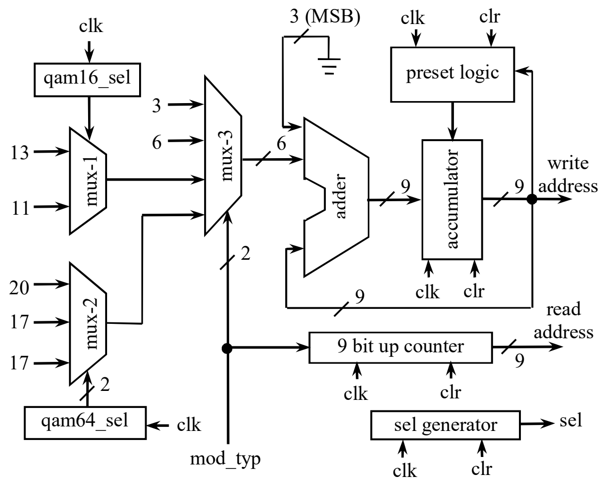


Fig. 4.10 Schematic diagram of address generator

The preset logic is a hierarchical FSM whose principal function is to generate the correct beginning addresses for all subsequent iterations and is shown in the form of state diagram in Fig. 4.11. This block contains a 4-bit counter keeping track of end of states during the iteration. The FSM enters into the first state (S_F) with clr = 1. Based on the value in mod_typ it makes transition to one of the four possible next states (S_{MT0}, S_{MT1}, S_{MT2} or S_{MT3}). Each state in this level represents one of the possible modulation schemes. The FSM thereafter makes transition to the next level of states (e.g. S_{000}, S_{001} and so on) based on the value in the accumulator. When the FSM at this level reaches to the terminal value of that iteration (e.g. 45 in S_{MT0}), it makes transition to a state (e.g. S_{000}) in which it loads the accumulator with the initial value (e.g. preset=1) of the next iteration. This process continues till all the interleaver addresses are generated for the selected mod_typ. If no changes take place in the values of mod_typ, the FSM will follow the same route of transition and the same set of interleaver addresses will be continually be generated. Any change in mod_typ value causes the interleaver to follow a different path. In order to facilitate the address generator with on the fly address computation feature, we have made the circuit to respond to clr input followed by mod_typ inputs at any stage of the FSM. With clr=1 it comes back to S_F state irrespective of its current position and there after transits to the desired states in response of new value in mod_typ.

The read addresses are linear in nature and are generated using a 9-bit up counter as shown in Fig. 4.10. The counter is reset whenever it reaches to the terminal count for a desired modulation scheme. For example, in case of 16-QAM, the counter counts from 0 to 191 and then repeats. The sel generator is basically a T flip-flop used to generate the select (sel) signal and is initialized to zero using clr input.

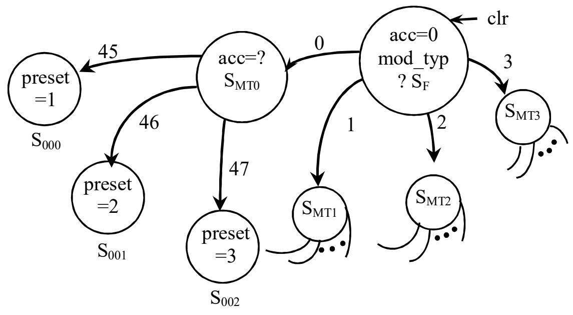


Fig. 4.11 State diagram of preset logic

4.4.2.1 Interleaver Memory

The interleaver memory block comprises of two memory modules (RAM-1 and RAM-2), three muxs and an inverter as shown in Fig. 4.12. In block interleaving when one memory block is being written the other one is read and vice-versa. Each memory module receives either write address or read address with the help of the mux connected to their address inputs (A) and sel line. RAM-1 at the beginning receives the read address and RAM-2 gets the write address with write enable (W_E) signal of RAM-2 active. After a particular memory block is read / written up to the desired location, the status of sel changes and the operation is reversed. The mux at the output of the memory modules routes the interleaved data stream from the read memory block to the output.

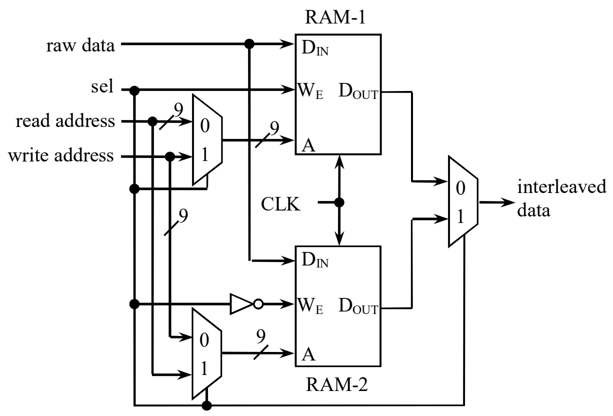


Fig. 4.12 Schematic view of FSM based Interleaver Memory block

The maximum memory required for OFDM based WLAN interleaver is 288 bits. Two identical memory blocks each of capacity 288-bits are required for the implementation of block interleaver. To model this memory in FPGA we have followed two techniques: one using DRAM and the other using BRAM. To model 288-bits memory we require four 64 x 1-bit and one 32 x 1-bit DRAM as shown in Fig. 4.13. The write enable (WE) logic is designed with the help of a 3 to 5 decoder as shown Fig. 4.14. Table 4.6 shows the conditions in which the WE signals for various DRAM blocks are generated. Modeling the interleaver memory using BRAM is relatively simpler than DRAM approach. BRAM of 16K x 1 bit has been utilized to model the memory.

Both approaches have their own merits and demerits. The DRAM technique makes the embedded memory free for other requirement in the system. In this technique exact amount of memory requirement can be modeled. DRAM is available at the cost of slices which otherwise used to implement digital logic. As a result FPGA resources available to implement other logic functions, if required are crunched. However, this drawback can be mitigated with modern day FPGAs which contains abundance of logic resources due to advancement in VLSI Technology. On the contrary, the BRAM technique uses dedicated memory leaving slices free to implement digital logic. It does not require logic circuit like that of Fig. 4.14 for WE. Moreover BRAM based interleaver can operate at higher frequency than its DRAM counterpart. The only drawback is that out of 16Kbits only 288-bits are used keeping rest unutilized.

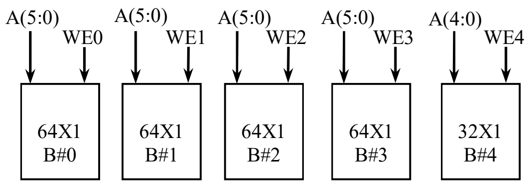


Fig. 4.13 Organization of 288 bit DRAM

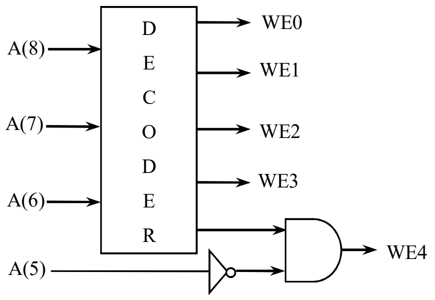


Fig. 4.14 Write Enable signal generation for various DRAM blocks

Table 4.6 Condition for Generation of Write Enable Signals

B#0	B#1	B#2	B#3	B#4
$A_8A_7A_6$ = 000	$A_8A_7A_6$ = 001	$A_8A_7A_6$ = 010	$A_8A_7A_6$ = 011	$A_8A_7A_6A_5$ = 1000
WE0=1	WE1=1	WE2=1	WE3=1	WE4=1

4.5 Simulation Results

Simulation results of the proposed LUT and FSM based interleavers for OFDM based WLAN are presented in the form of timing diagram in Fig. 4.15 (a)-(d) and Fig. 4.16 (a)-(d) respectively. In both cases, the diagrams are obtained using ModelSim Xilinx Edition-III, version 6.0a. In these figures, 4.15(a) is identical with 4.16(a) as both displays interleaver output for BPSK (mod_typ = 00). Similarly, Fig. 4.15(b) and 4.16(b) are identical as both the simulation results are obtained with mod_typ = 01 (i.e. QPSK) and so on. In all the figures of 4.15 and 4.16, first 16-bits of raw data input (data_in) are held high. The effect of interleaver is visible as the consecutive 1's are dispersed by certain bit positions in the data output (data_out) line. In case of BPSK, the spread is uniform and by three positions. For QPSK, the bits are spread by six positions uniformly. On the contrary, 16-QAM and 64-QAM show non-uniform spread by 13/11 and 20/17/17 respectively. This is because the write address sequences in BPSK and QPSK modulation schemes are uniformly distributed whereas, for 16-QAM and 64-QAM they are non-uniform as highlighted in Table 4.4. The CLR signal is used to reset (= 1) the interleaver at the beginning of an operation.

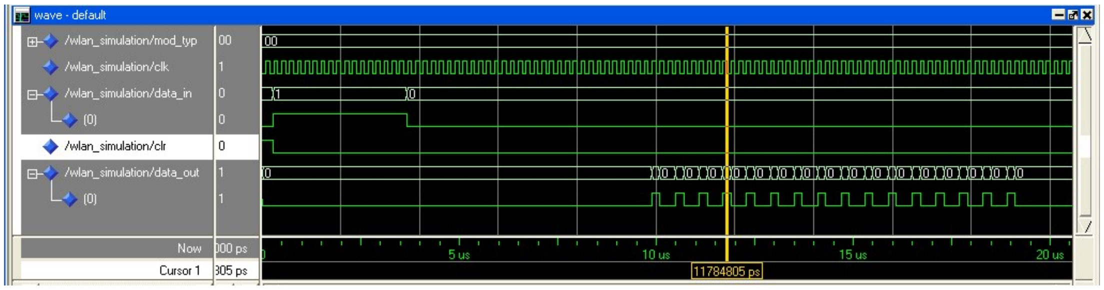


Fig. 4.15(a) Simulation result for BPSK (mod_typ = 00) in LUT based interleaver

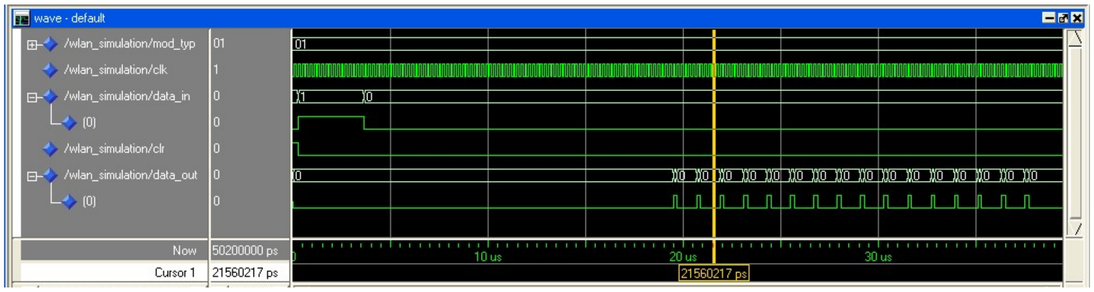


Fig. 4.15(b) Simulation result for QPSK (mod_typ = 01) in LUT based interleaver

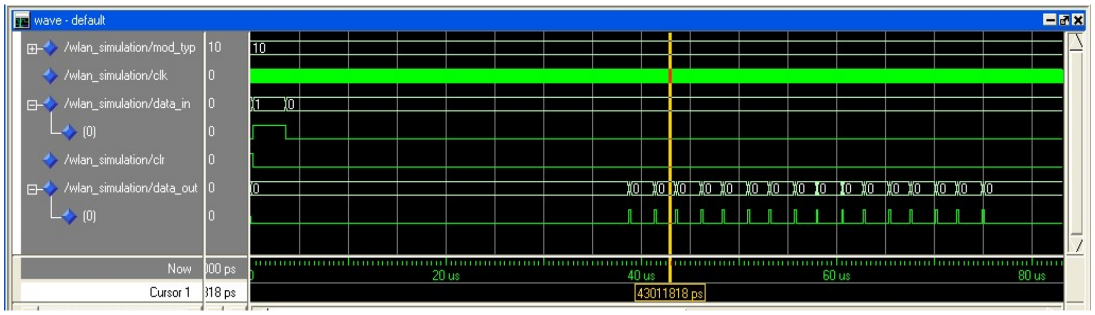


Fig. 4.15(c) Simulation result for 16-QAM (mod_typ = 10) in LUT based interleaver

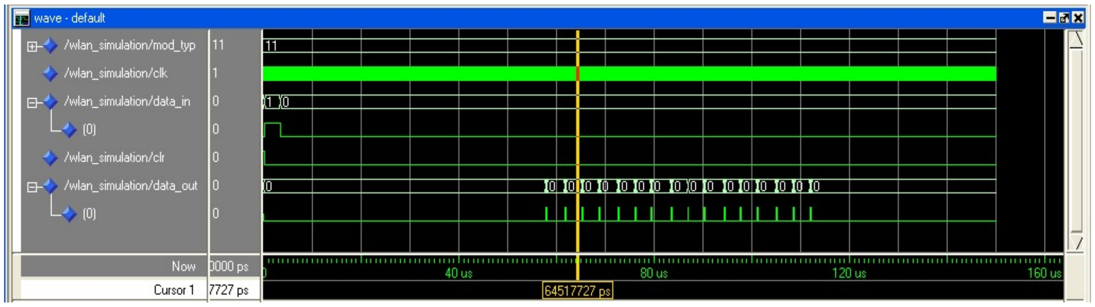


Fig. 4.15(d) Simulation result for 64-QAM (mod_typ = 11) in LUT based interleaver

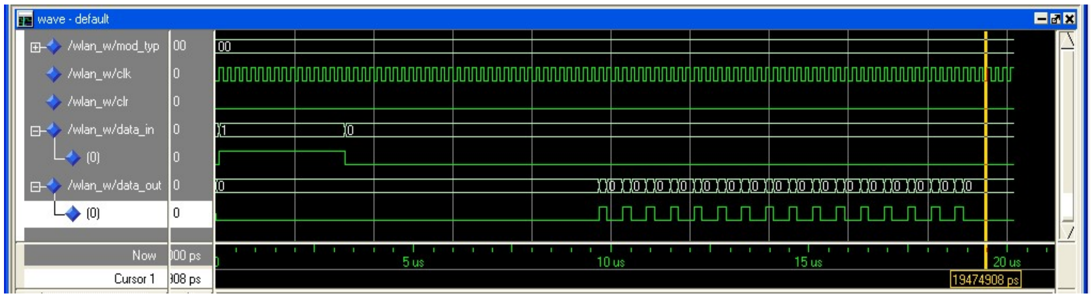


Fig. 4.16(a) Simulation result for BPSK (mod_typ = 00) in FSM based interleaver

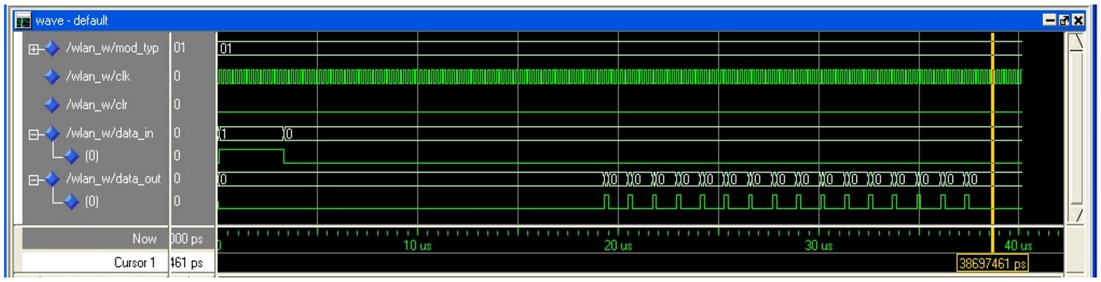


Fig. 4.16(b) Simulation result for QPSK (mod_typ = 01) in FSM based interleaver

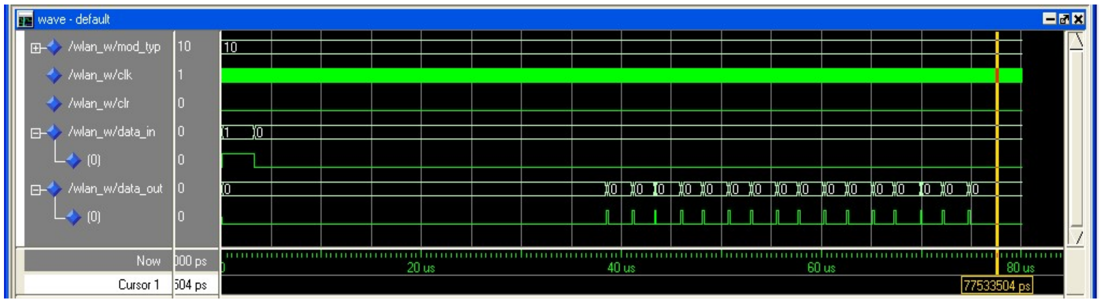


Fig. 4.16(c) Simulation result for 16-QAM (mod_typ = 10) in FSM based interleaver

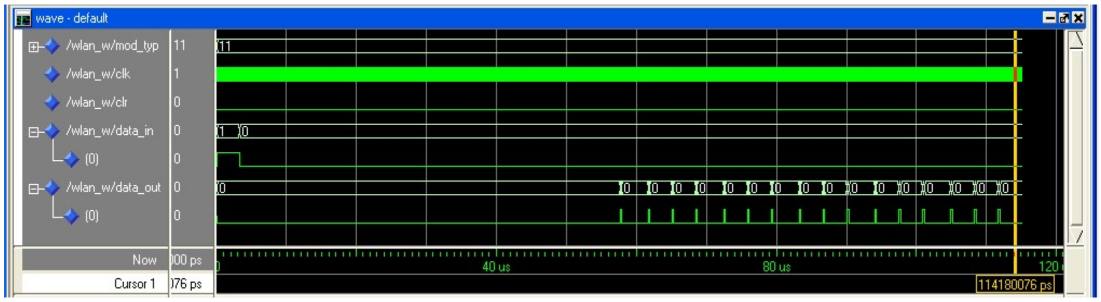


Fig. 4.16(d) Simulation result for 64-QAM (mod_typ = 11) in FSM based interleaver

4.6 Critical Analysis of FPGA Implementation

This section describes FPGA implementation results and their analysis for the proposed two techniques of interleaver design.

4.6.1 FPGA Implementation of LUT based Interleaver

The proposed VHDL model of the LUT based interleaver is prepared using Xilinx ISE and is implemented on Xilinx Spartan-3 FPGA. Table 4.8 shows the HDL synthesis report for the implementation. The M_1 of Fig. 4.8 has been modeled in a ROM of size 4 x

10-bit. Three adders are being used by the circuit, two as A_1 and A_2 and the third in the counter. A_2 is 14-bit while other two are 10-bit. Three 10-bit registers are used to hold the read address, rd address and the count value in counter respectively in addition to one d flip-flop as rw_sel. The 10-bit 4-to-1 multiplexer models the M_2 mux. Device utilization summary of this implementation on Xilinx Spartan-3 FPGA (XC3S400) has been described in Table 4.8. As seen, the proposed technique utilizes only 1.53% of available slices, 0.42% of available slice flip-flop, and 1.42% of available 4 input LUTs. The estimated power consumption of the implementation is found to be as low as 56mW.

Table 4.7 HDL Sythesis Report of LUT based WLAN Interleaver

Logic Circuit used	Quantity
4x10-bit ROM	1
10-bit adder	2
14-bit adder	1
10-bit register	3
1-bit register	1
10-bit 4-to-1 multiplexer	1

Table 4.8 Device Utilization Summary of LUT based WLAN Interleaver

FPGA Resources	Utilization in Number	Utilization in %
Number of Slices	55 out of 3584	1.53
Number of Slice Flip-flops	30 out of 7168	0.42
Number of 4 input LUTs	102 out of 7168	1.42
Number of Bonded IOBs	06 out of 141	3.55
Number of BRAMs	2 out of 16	12.50
Number of GCLKs	1 out of 8	12.50

4.6.2 FPGA Implementation of FSM based Interleaver

The proposed VHDL model of the FSM based interleaver has been developed using Xilinx ISE and has also been implemented on Xilinx Spartan-3 FPGA. Two versions of the memory model, (i.e. BRAM and DRAM) along with FSM based common address generator have been implemented. Table 4.9 shows the HDL synthesis report of both the implementations. It is evident that except the use of embedded memory, logic circuit requirement is lesser in case of the technique using BRAM. The DRAM technique uses two ROM, some register/latches in excess compared to the technique using BRAM. The write enable logic for the two sets of memory module of Fig. 4.13 are modelled by the two 4 x 1-bit ROM. The 2-bit adder is used in the 2-bit counter, qam64_sel whereas the 9-bit adder is used to generate the write address of interleaver as shown in Fig. 4.10. The 4-

bit up counter in preset logic keeps track of the end of iteration. Read addresses for the interleaver are generated by the 9-bit up counter. The accumulator is implemented by the 9-bit register whereas the other registers are required for qam64_sel and qam16_sel. The latches are used to store some internal signals. The mux-3 of Fig. 4.10 is implemented using the 5-bit 4-to-1 mux whereas 9-bit 4-to-1 mux is required in preset logic and the 1-bit 4-to-1 mux implements the switch over condition between reading and writing of RAM blocks.

Device utilization summary of both the implementations has been described in Table 4.10. As seen the DRAM technique utilizes 37.27% excess slices, 32.81% excess slice flip-flops and 47.39% excess 4 input LUTs in comparison with BRAM technique. Out of the 206 nos. of 4 input LUTs, 170 nos. (82.52%) are used in the logic circuits of the entire interleaver and rest 36 nos. (17.48%) are used to model the two numbers of 288-bit RAM modules. The BRAM technique uses 2 out of available 16nos. of BRAM leaving rest 14 blocks for other uses if required in the associated circuits. The BRAM technique can operate at a maximum frequency of 154.879 MHz (propagation delay of 6.457ns) whereas that of DRAM technique is 116.21 MHz (propagation delay of 8.605ns). The former technique provides 24.97% faster performance over the later. As far as estimated power consumption is concerned both the techniques show similar results and each consumes 56mW of power. Low power consumption is an important advantage for the equipments used in wireless communication as they are being run by battery power.

Table 4.9 HDL Sythesis Report of FSM based WLAN Interleaver

WITH BRAM	WITH DRAM
	4x1-bit ROM : 2
2-bit adder : 1 9-bit adder : 1	2-bit adder : 1 9-bit adder : 1
4-bit up counter : 1 9-bit up counter : 1	4-bit up counter : 1 9-bit up counter : 1
1-bit register : 1 2-bit register : 1 9-bit register : 1	1-bit register : 1 2-bit register : 1 9-bit register : 1
1-bit latch : 1 9-bit latch : 1	1-bit latch : 3 9-bit latch : 1
1-bit 4-to-1 mux : 1 5-bit 4-to-1 mux : 1 9-bit 4-to-1 mux : 1	1-bit 4-to-1 mux : 1 5-bit 4-to-1 mux : 1 9-bit 4-to-1 mux : 1

Table 4.10 Device Utilization Summary of FSM based WLAN Interleaver

WITH BRAM			WITH DRAM			% of excess use by DRAM method
FPGA Resources	**Utilization in No.**	**Utilization in %**	**FPGA Resources**	**Utilization in No.**	**Utilization in %**	
Number of Slices	61 out of 3584	1.70	Number of Slices	97 out of 3584	2.71	37.27
Number of Slice Flip-flops	31 out of 7168	0.43	Number of Slice Flip-flops	46 out of 7168	0.64	32.81
Number of 4 input LUTs	108 out of 7168	1.51	Number of 4 input LUTs	206 out of 7168	2.87	47.39
Number of bonded IOBs	6 out of 141	4.26	Number used as logic	170 out of 206	82.52	--
			Number used as RAM	36 out of 206	17.48	--
Number of BRAMs	2 out of 16	12.50	Number of bonded IOBs	6 out of 141	4.26	nil
Number of GCLKs	1 out of 8	12.50	Number of GCLKs	1 out of 8	12.50	nil

Comparative study of the proposed implementations in terms of FPGA resources and operating speed is shown in Table 4.11. The LUT and BRAM based techniques show better performance over the DRAM technique in terms of FPGA resources along with the operating frequency. Comparison between LUT and BRAM techniques shows very competitive result in all parameters with some betterment in favour of the BRAM technique. The improvement obtained is due to optimization in BRAM requirement and hence eliminating the associated logic circuits. The BRAM technique shows marginal betterment in terms of operating speed by a factor of 8.56%. In BRAM technique the address generator is implemented using logic circuits whereas in LUT based implementation the address generator is LUT based implemented in FPGA’s internal memory. DRAM technique does not require any BRAM.

Efforts have been made by the author to compare the FPGA implementation results of our proposed work with that of other researchers. Direct comparison with [103] is not possible as the author describes the FPGA implementation of the complete OFDM transmitter for IEEE 802.11a based WLAN. However, in [103] interleaver address generation is done by modeling LUT using FPGA based single port DRAM. The DRAM based technique consumes larger FPGA resources with single advantage that it does not use the dedicated FPGA memory (BRAM).

Table 4.11 Comparison Between Various Implementations

FPGA parameter	Performance of LUT technique	Performance of BRAM technique	Performance of DRAM technique
Slices	1.53% used	1.70% used	2.71% used
Flip-flop	0.42% used	0.43% used	0.64% used
LUT	1.42% used	1.51% used	2.87% used
BRAM	12.5% used	12.5% used	nil
Operating frequency	141.63 MHz	154.88 MHz	116.21 MHz

4.7 Discussion

In this chapter two novel LUT and FSM based techniques have been proposed to model the block interleavers used in IEEE 802.11a and IEEE 802.11g based WLAN. The proposed hardware models of the interleaver are completely implemented in Spartan-3 FPGA. Unlike the conventional technique which uses external memory, the LUT based technique uses FPGA's own internal memory to house the addressing sequences. Single memory module is partitioned to eliminate the requirement of four memory blocks. Due to this partitioning, the proposed technique shows better result in terms of operating frequency and hardware resources. In the FSM based approach, two different techniques to model the required memory in the interleaver using internal resources of FPGA have been shown. Critical analysis of implementation results of both approaches has been made to ease the decision making of a system designer regarding the technique to adopt in WLAN applications. Both the techniques make efficient use of FPGA's internal resources. Finally, all the approaches have been compared and concluded that BRAM-FSM based technique shows better result among them. Methodology adopted in this work is extended further to develop improved design and implementation of WiMAX interleavers on reconfigurable platform.

Chapter 5
Interleaving in WiMAX

Outline of this Chapter

This chapter presents in depth the analysis of various issues related to the design and implementation of WiMAX interleaver / de-interleaver on FPGA platform in maintaining logical extension of the similar work on WLAN as reported in the previous chapter. Multiple designs supported with algorithmic and mathematical background have been proposed. Importance of WiMAX with brief technical specifications, system overview and interleaver specifications have also been incorporated. Detailed discussion on the proposed FSM based interleaver, improved LUT based de-interleaver and low complexity de-interleaver along with their hardware models, simulation waveforms and FGPA implementation results have been made. This discussion includes design of address generator of the interleaver / de-interleaver. Noticeable performance improvement in terms of FPGA resource utilization and operating speed in comparison with existing implementation available in literature have been recorded.

5.1. Introduction

Tremendous increase in the use of internet in the last decade has put the quest of BWA. It is increasingly gaining popularity as an alternative solution to DSL or cable modem for internet access. BWA has stringent requirements like high processing speed, flexibility and fast design TAT. These requirements make the choice of the reconfigurable hardware platform like FPGA as the obvious option. Moreover, any new technology like WiMAX needs some time to mature. Thus a product implemented on FPGA can easily be upgraded by making necessary changes in the HD code only and thus becomes obsolescence free. In addition, the TAT of FPGA based circuits is much shorter compared to ASIC based design. Design flexibility is another important advantage of FPGA based implementations. The proposed system could have also been implemented using software. The principal drawback of such approach is that a powerful computer is to be used to run the program for achieving high processing speed which is a prerequisite for WiMAX system. Employing such powerful computer may be a costly solution which may be detrimental to the popularity of WiMAX.

WiMAX is based on the IEEE 802.16 standard for BWA system. IEEE 802.16d, now known as, IEEE 802.16-2004 defines fixed WiMAX in the frequency band of 2 to 11GHz [49]. Amended IEEE 802.16e adds mobility support to IEEE 802.16 and defines standard for mobile WiMAX in the frequency band 2-6 GHz. Typical data rate in IEEE 802.16e is 5Mbps with bandwidth 1.25 to 20 MHz up to 2048 sub-carriers, as opposed to the OFDM version with 256 sub-carriers (of which 200 are used) in 802.16-2004. Both IEEE 802.16-2004 and IEEE 802.16e permit NLOS connectivity [67]. The WiMAX air interface adopts Orthogonal Frequency Division Multiple Access (OFDMA) for improved multi-path performance. Scalable OFDMA (SOFDMA) [151] is introduced in the IEEE 802.16e amendment to support scalable channel bandwidths from 1.25 to 20 MHz.

OFDM [67] technique offers promising solution that has gained tremendous research interest in recent years due to its high transmission capability and alleviating the adverse effects of ISI and ICI. In an OFDM system, the data is divided into multiple parallel sub-streams at a reduced data rate, and each is modulated and transmitted on a separate orthogonal subcarrier. This increases symbol duration and thereby improves system robustness. OFDM is achieved by providing multiplexing on users' data streams on both uplink and downlink transmissions. OFDM is the fundamental building block of the IEEE 802.16 standard.

In digital communication systems, presence of interleavers improve the performance of FEC codes in terms of bit error rate. Interleaving process basically changes the arrangement of an input code symbol into a new one so that occurrences of burst errors will be spread out and FEC techniques of random error correction become effective. Block interleaving is one of the popular techniques to counter burst error in the channel and are being employed in many modern day wireless communication system applications. In a block interleaver, input bit streams are stored row wise in the interleaver memory and read column wise and vice versa. WiMAX uses a special type of block interleaver in which the interleaver depth and pattern vary depending on the code rate and modulation type.

In this chapter, three works related to the design of WiMAX hardware interleaver and de-interleaver are being presented. The interleaver / de-interleaver contains complex functions like modulus and floor due to which the design is challenging. These complex functions do not have any corresponding digital hardware for implementation. In addition, VHDL doesn't support such functions directly. Consequently, challenges are faced in preparing the VHDL model of the interleaver / de-interleaver circuitry due to unavailability of such functions. Conventional LUT based approaches are found to be consuming large amount of logic resources apart from slowness in operation. This leads to low speed design with inefficient use of resources. The first work is all about the design of a novel FSM based multimode, high speed and hardware efficient technique to implement the address generation circuitry of WiMAX interleaver based on IEEE 802.16e standard on FPGA platform. An LUT based de-interleaver design approach is presented next. In this approach, the conventional LUT based technique for address generation has been re-designed to use the FPGA memory blocks efficiently. The third technique is about design of a low complexity and resource efficient hardware de-interleaver for use in IEEE 802.16e based WiMAX. This work includes design of a novel algorithm for the de-interleaver with user-friendly mathematical representation and its general validity. Use of FPGA's embedded multiplier provides performance improvement by reducing interconnection delay, efficient resource utilization and lesser power consumption compared to CLB based multiplier. This work shows betterment over LUT technique in terms of maximum operating frequency.

5.2 System Description

The system level overview of IEEE 802.16e based WiMAX system is described in Fig. 5.1. In this system, the input binary data stream obtained from a source is randomized to prevent a long sequence of 1s and 0s, which will cause timing recovery problem at the receiver. The randomized data bits are thereafter encoded using Reed Solomon (RS) encoder followed by Convolutional Coder (CC). The former is suitable for correction of burst type of errors [152] whereas the latter is for random errors [153]. After RS-CC encoding, all encoded data bits are to be interleaved by a special type block interleaver. In the block interleaver of WiMAX system, data is written in a random manner based on certain permutation in the memory and read sequentially [154]. Thereafter data passes through the mapper block in which modulation takes place. The resulting data symbols are used to construct one OFDM symbol by performing Inverse Fast Fourier Transform (IFFT). CP is used to reduce ISI and ICI [67]. The receiver section as shown in Fig. 5.1 works exactly in reverse order.

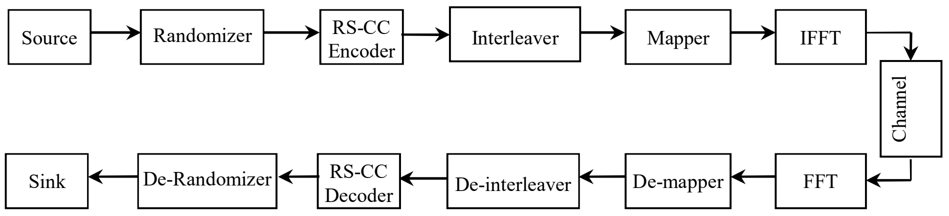


Fig. 5.1 Overview of WiMAX system

5.3 Interleaving / De-interleaving in WiMAX System

The block interleaver used in WiMAX system has different interleaving patterns for different code rates and modulation schemes. Different Interleaver Depths (IDs) are required to incorporate various code rates and modulation schemes. Table 5.1 describes permitted interleaver depths in IEEE 802.16e [52]. Bits in WiMAX are interleaved in two steps. The first step ensures that the adjacent coded bits are mapped onto nonadjacent subcarriers, which provides frequency diversity and improves the performance of the decoder. The second step ensures that the adjacent bits are alternately mapped to less and more significant bits of the modulation constellations to avoid long run of lowly reliable bits.

Let N_{cbps} is the block size corresponding to the number of coded bits per allocated sub-channels per OFDM, d represents number of columns of the block interleaver which is typically chosen to be 16 for WiMAX [155]. m_k is the output after first level of permutation and k varies from 0 to N_{cbps} -1. s is a parameter defined as s = $N_{cpc}/2$, where N_{cpc} is the number of coded bits per sub-carrier, i.e., 2, 4 or 6 for QPSK, 16-QAM or 64-QAM respectively [154]. Thus for QPSK, s=1, for 16-QAM, s = 2 and for 64-QAM, s = 3. The first and second levels of permutation are given by (5.1) and (5.2) respectively are as follows:

$$m_k = (\frac{N_{cbps}}{d})(k \% d) + \left\lfloor \frac{k}{d} \right\rfloor \tag{5.1}$$

$$j_k = sx\left\lfloor \frac{m_k}{s} \right\rfloor + (m_k + N_{cbps} - \left\lfloor \frac{d x\, m_k}{N_{cbps}} \right\rfloor)\%s \tag{5.2}$$

where % and ⌊ ⌋ signify modulo and floor functions respectively.

The de-interleaver, which performs the inverse operation, is also defined by two permutations. Within a received block of N_{cbps} bits, let j be the index of received bits before the first permutation; m_j be the index of that bit after the first and before the second permutation; and let k_j be the index of that bit after the second permutation, just prior to delivering the block to the decoder. Equation (5.3) and (5.4) define the first and second level of permutations for de-interleaver.

$$m_j = s.\left\lfloor \frac{j}{s} \right\rfloor + \left(j + \left\lfloor \frac{d.j}{N_{cbps}} \right\rfloor\right)\%s \tag{5.3}$$

$$k_j = d.m_j - (N_{cbps} - 1).\left\lfloor \frac{d.m_j}{N_{cbps}} \right\rfloor \tag{5.4}$$

As per [52], (5.3) and (5.4) performs inverse operation of (5.2) and (5.1) respectively.

Table 5.1 Permitted interleaver/de-interleaver depths in IEEE 802.16e for all code rates and modulation schemes

Modulation Scheme	QPSK (s=1)		16-QAM (s=2)		64-QAM (s=3)		
Code Rate	½	¾	½	¾	½	⅔	¾
Interleaver Depth N_{cbps} in bits	96	144	192	288	288	384	432
	192	288	384	576	576	-	-
	288	432	576	-	-	-	-
	384	576	-	-	-	-	-
	480	-	-	-	-	-	-
	576	-	-	-	-	-	-

5.4 Hardware Modeling of FSM based Address Generator

The address generation circuit of the block interleaver for WiMAX system is basically the simultaneous implementation of (1) and (2). A MATLAB program has been developed implementing (1) and (2) for pre-computation of the interleaver address sequences and is being described by an algorithm shown in Fig. 5.2. Execution of this program with permissible values of N_{cbps} for different modulations, we find all the values of interleaver memory addresses, designated by j_k, out of which first 32 of each category are only listed in Table 5.2. Careful examination of the values of j_k, confirms that the subsequent values are not equally spaced for all cases. Within a modulation scheme, the increment values follow a fixed type of pattern irrespective of coding rate. Encoding of ID, Modulation Type (MOD_TYP) and increment values from implementation point of view are presented in Table 5.3.

```
          Input $N_{cbps}$, s and d.
          Initialize $k=0$
Rep:      Compute
                $k=k+1$
                $b=k/d$
                $c=mod (k, d)$
                $g=floor (b)$
                $m_k= (N_{cbps}/d)*c + g$
                $a=floor (m_k/S)$
                $e= (d*m_k)/N_{cbps}$
                $h=floor (e)$
                $f=m_k+ N_{cbps}-h$
                $i=mod (f, s)$
                $j_k=s*a + i$
          Print
                $j_k$
          If $k< N_{cbps}$ then go to Rep
          Else End
```

Fig. 5.2 Algorithm of MATLAB program used to pre-compute WiMAX interleaver addresses

Table 5.2 First 32-permutation sample addresses for three code rates and modulation schemes

N_{cbps}=96 bits, ½ code rate, QPSK	0	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90
	1	7	13	19	25	31	37	43	49	55	61	67	73	79	85	91
N_{cbps}=288 bits, ¾ code rate, 16-QAM	0	19	36	55	72	91	108	127	144	163	180	199	216	235	252	271
	1	18	37	54	73	90	109	126	145	162	181	198	217	234	253	270
N_{cbps}=384 bits, ⅔ code rate, 64-QAM	0	26	49	72	98	121	144	170	193	216	242	265	288	314	337	360
	1	24	50	73	96	122	145	168	194	217	240	266	289	312	338	361

The address generation concept of the proposed block interleaver is described in the form of schematic diagram as shown in Fig. 5.3. Unlike [106], our design includes all possible code rates and modulation type permitted under IEEE 802.16e. As shown in Fig. 5.3, the design concept contains three levels of multiplexer (MUX). The first level MUXs implement the unequal increments required in 16-QAM and 64-QAM. The four-interleaver depths of 16-QAM as shown in Table 5.3 are implemented by the first four MUXs from the top in level 1. The select inputs of these four MUXs are tied together and are driven by a T flip-flop named QAM16_SEL. Similarly, the last four MUXs are for 64-QAM modulation. The select inputs are driven by a mod-3 counter, QAM64_SEL. The second level MUXs basically pick up one inputs based on the values of ID. The topmost MUX in level 2 implements the eight interleaver depths of QPSK modulation scheme available by concatenation of sub-channels [52]. The second and third MUXs in level 2 are for 16-QAM and 64-QAM respectively. The outputs from level 2 MUXs are routed to the next section by level 3 MUX based on MOD_TYP value. The 7-bit output from the level 3 MUX acts as one input to the 10-bit adder circuit after zero padding. The other input of the adder comes from Accumulator, which holds the previous address. After addition a new address is written in the Accumulator. The preset logic is a FSM whose principal function is to generate the correct beginning addresses for all subsequent iterations and is described at length in the next section.

Table 5.3 Increment values for various interleaver depths and modulation schemes with their encoding

Modulation	MOD_TYP	Interleaver Depth	ID*	Increment values	Whether equally spaced
QPSK	00	96	000	6	Yes
		144	001	9	Yes
		192	010	12	Yes
		288	011	18	Yes
		384	100	24	Yes
		432	101	27	Yes
		480	110	30	Yes
		576	111	36	Yes
16-QAM	01	192	X00	13,12	No
		288	X01	19,17	No
		384	X10	25,23	No
		576	X11	37,35	No
64-QAM	1X	288	X00	20,17,17	No
		384	X01	26,23,23	No
		432	X10	29,26,26	No
		576	X11	38,35,35	No

* Also referred as CODE_RATE.

As a case study the algorithm for modeling of QPSK MUX in VHDL is described below.

```
If ID = 0 then MUX_OUT <= 6
else if ID = 1 then MUX_OUT <=
9
else if ID = 2 then MUX_OUT <=
12
…
else MUX_OUT <= 36.
```

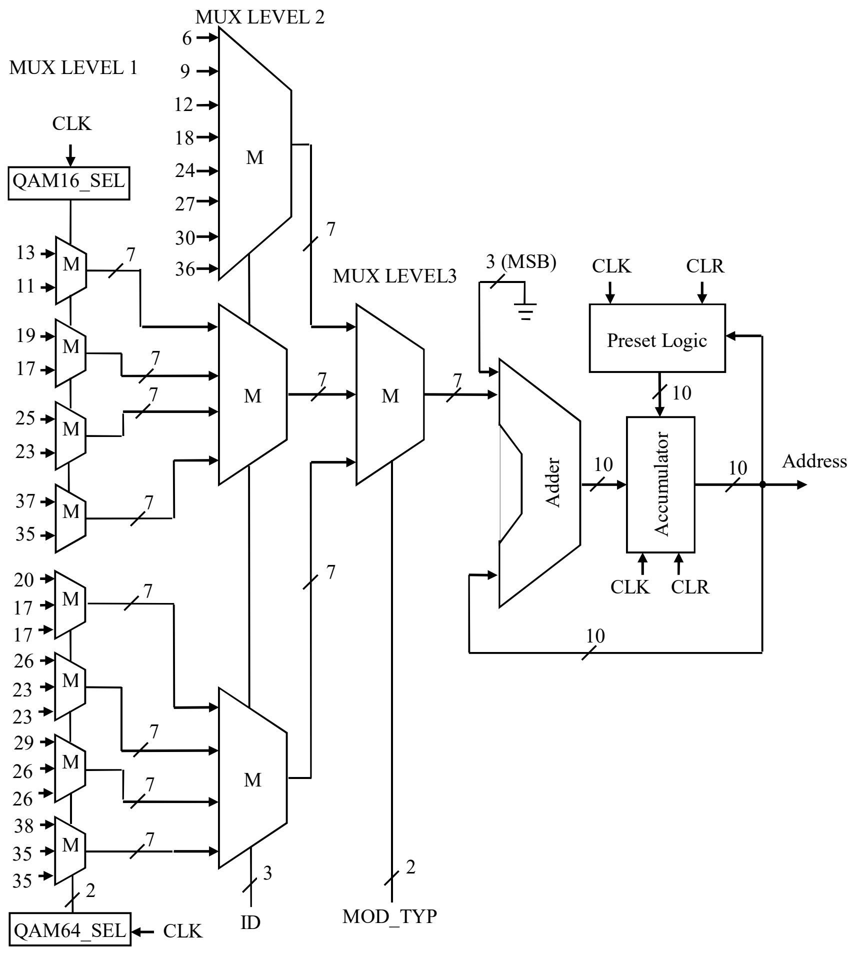


Fig. 5.3 FSM based Address Generation scheme

5.4.1 Preset Logic as Finite State Machine

The Preset Logic block of Fig. 5.3 is the heart of the Address Generator of WiMAX interleaver. It is basically a hierarchical FSM and the state diagram is shown in

Fig. 5.4. This block contains a 4-bit counter which keeps track of end states during an iteration. The FSM enters into the first state (S_F) with CLR=1. Based on the value in MOD_TYP it makes transition to one of the three possible next states (S_{MT0}, S_{MT1} or S_{MT2}). Each state in this level represents one of the possible modulation schemes. The FSM thereafter makes transition to one of the next level states (S_{ID0} to S_{ID7} from S_{MT0}, S_{ID0} to S_{ID3} from S_{MT1} or S_{MT2}) based on the value in ID. The various states of this level signify one of the interleaver depths. From these states it branches to the next level of states based on the value in the accumulator. When the FSM at this level reaches to the terminal value of that iteration (e.g. 90 in S_{ID0} of S_{MT0}), it makes transition to a state (e.g. S_{000}) in which it loads the accumulator with the initial value (e.g. Preset=1) of the next iteration. This process continues till all the interleaver addresses are generated for the selected ID and MOD_TYP. If no changes take place in the values of ID and MOD_TYP, the FSM will follow the same route of transition and the same set of interleaver addresses will be continually generated. Any change in ID and MOD_TYP value causes the interleaver to follow a different path. In order to facilitate the address generator with on the fly address computation feature, the designed circuit responds to CLR followed by ID and MOD_TYP inputs at any stage of the FSM. With CLR = 1 it comes back to S_F state irrespective of its current position and there after transits to desired states in response to new values in ID and MOD_TYP.

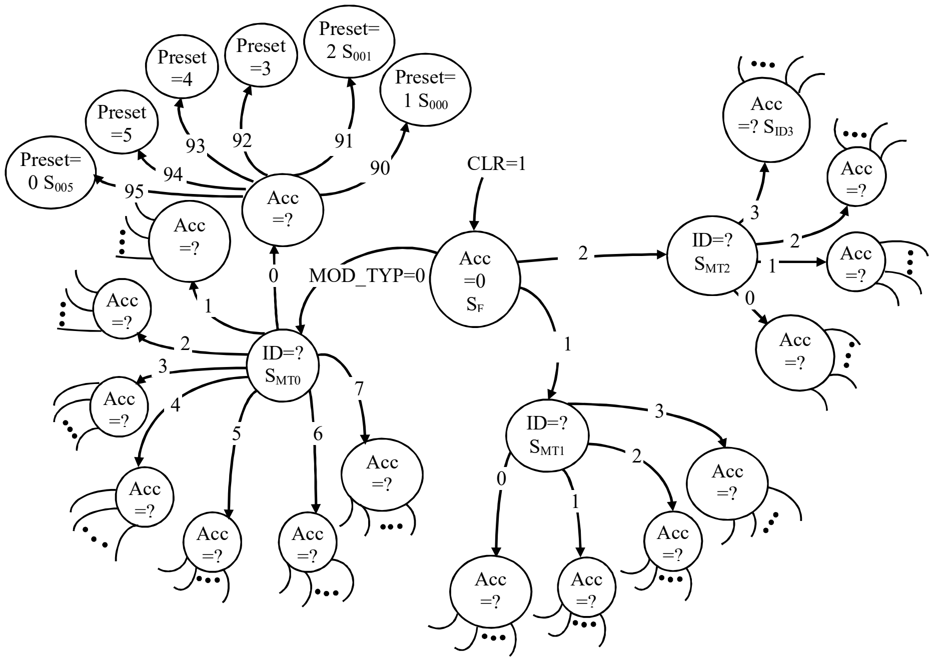


Fig. 5.4 States in preset logic

5.5 Modeling Memory in FPGA

Modern FPGAs are equipped with different types of embedded resources to support efficient implementation of circuitry related to various applications like local storage, FIFO, data buffers, stack, large LUT etc. One of such internal resource offered in Xilinx FPGAs is BRAM [150]. Table 5.4 and Fig. 5.5 list all the interface signals of a single port BRAM and their directions. In our experimentation, Xilinx Spartan-3/Spartan-3AN FPGA (device XC3S1400AN) [94] having 16/32 nos. of 18KB (16KB data and 2KB parity) single port BRAM block is used. Out of these, 3 BRAM blocks are used to store the address LUTs for three different modulation schemes of WiMAX de-interleaver address generator.

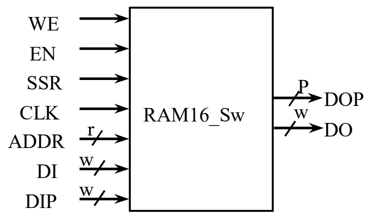


Fig. 5.5 Single Port BRAM in Xilinx Spartan-3AN FPGA

Table 5.4 Single Port BRAM Interface Signal

Signal Description	Port Name	Direction	Brief Description
Data Input Bus	DI	Input	The memory block receives input data to be written in the selected location through these lines.
Parity Data Input Bus	DIP	Input	The memory block receives parity data input to be written in the selected location through these lines.
Data Output Bus	DO	Output	The memory block transmits data from a selected location through these lines.
Parity Data Output	DOP	Output	The memory block transmits parity data from a selected location through these lines.
Address Bus	ADDR	Input	Through these lines, a memory location is addressed for either read or write operation.
Write Enable	WE	Input	This signal when made active (logic 1) permits the data write operation in a selected memory location.
Clock Enable	EN	Input	This signal when made active (logic 1) enables the memory block. This signal can be treated as master control of the memory block.
Synchronous Set/Rest	SSR	Input	The synchronous set/reset input, SSR, forces the data output latches to the value specified by the SRVAL attribute. When SSR and the enable signal, EN, are High, the data output latches for the DO and DOP outputs are synchronously set to a '0' or '1' according to the SRVAL parameter.
Clock	CLK	Input	This signal clocks for all synchronous operations. Clock polarity is configurable and is rising edge triggered by default.

5.6 Hardware Model of LUT based De-interleaver

5.6.1 Methodology of proposed design

In general, the design methodology of hardware interleaver / de-interleaver is classified into two categories, LUT based and incremental address generation based FSM. The former technique is relatively simple but consumes large logic resources, particularly memory, whereas the latter involves complex design methodology but requires relatively less logic resources. In this work, the author proposes an improved design methodology to implement the LUT based address generator for WiMAX de-interleaver on reconfigurable platform. As per IEEE 802.16e standard [53], ½, ⅔ and ¾ are the allowed code rates where as QPSK, 16-QAM and 64-QAM are the permitted modulation schemes. Accordingly, there are eight, four and four interleaver depths in QPSK, 16-QAM and 64-QAM modulation schemes respectively [155] to implement all the permissible code rates and modulation schemes. In conventional LUT based approach, to implement the de-interleaver address generator, 16 numbers of memory blocks of varying size are required to house all the interleaver addresses. During this work, a relationship between the de-interleaver memory addresses of various N_{cbps} within a modulation scheme is identified. It has been found that, the memory addresses of a larger N_{cbps} encompass the same of smaller N_{cbps}. This relationship between the address LUTs is exploited to propose a memory efficient LUT based address generator for WiMAX de-interleaver. Using our proposed design, the number of memory blocks used has been reduced to 3 only ensuring saving of 81.25% critical resource.

A MATLAB program is developed using (5.3) and (5.4) of to determine the write addresses of the de-interleaver for all code rates and modulation schemes. Flow chart representation of the program is presented in Fig. 5.6. Among these addresses, first 5 rows for each modulation schemes with N_{cbps} = 576-bits are presented in Table 5.5(a)-(c). The first 6 columns in Table 5.5(a), describe the memory addresses of first 5 rows with N_{cbps} = 96-bits and QPSK modulation scheme. Similarly, the first 5 rows of interleaver memory addresses with N_{cbps} = 144-bits and QPSK modulation scheme are represented by the first 9 columns in Table 5.5(a). Likewise, the addresses for other N_{cbps} with QPSK modulation scheme can be determined from the same Table where the number of columns is defined as N_{cbps}/d ($d = 16$). A similar approach can be applied in Table 5.5(b) and (c) to determine the memory addresses with various interleaver depths for 16-QAM and 64-QAM modulation schemes respectively. Pictorial representation of this relationship of memory addresses between various interleaver depths are given in Fig. 5.7(a)-(c). For example, all the 32 numbers of columns in Table 5.7(a) are represented by the entire cylinder of Fig. 5.7(a). First six columns in Table 5.7(a) represents the sample de-interleaver addresses for N_{cbps} = 96-bits and are marked with pink colour. Similarly, the first section of the cylinder in Fig. 5.7(a) also being marked with pink colour to signify the representation the same de-interleaver addresses. Likewise, next three columns in Table 5.7(a) are marked with black colour, which represents the sample de-interleaver addresses for N_{cbps} = 144-bits along with the first six pink coloured columns. As seen in Fig. 5.7(a), next portion of the cylinder is also painted with black colour. It is evident from Fig. 5.7(a) that the pink followed by black coloured cylinder actually signify the first nine columns of Table 5.7(a) representing the de-interleaver addresses for N_{cbps} = 144-bits. This discussion can be extended further to explain the relationship between remaining part of the Fig. 5.7(a)-(c) with rest part Table 5.7(a)-(c).

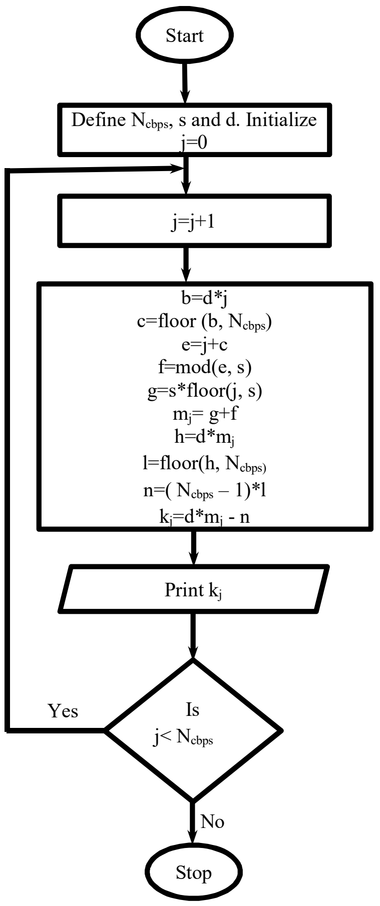


Fig. 5.6 Flow chart of MATLAB program used to determine WiMAX de-interleaver addresses

Table 5.5(a) First Five Rows of Addresses for N_{cbps} = 576, ¾ Code Rate, QPSK

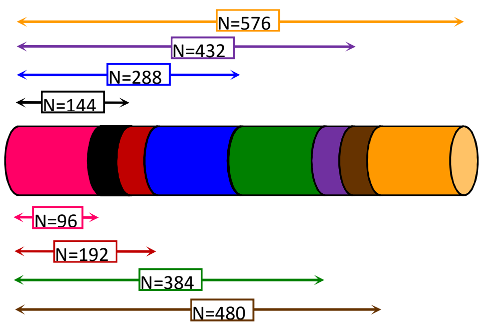
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320	336	352	368	384	400	416	432	448	464	480	496	512	528	544	560
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241	257	273	289	305	321	337	353	369	385	401	417	433	449	465	481	497	513	529	545	561
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242	258	274	290	306	322	338	354	370	386	402	418	434	450	466	482	498	514	530	546	562
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243	259	275	291	307	323	339	355	371	387	403	419	435	451	467	483	499	515	531	547	563
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244	260	276	292	308	324	340	356	372	388	404	420	436	452	468	484	500	516	532	548	564

Table 5.5(b) First Five Rows of Addresses for N_{cbps} = 576, ¾ Code Rate, 16-QAM

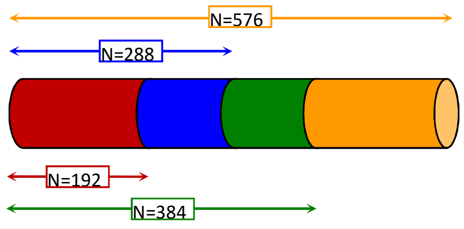
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320	336	352	368	384	400	416	432	448	464	480	496	512	528	544	560
17	1	49	33	81	65	113	97	145	129	177	161	209	193	241	225	273	257	305	289	337	321	369	353	401	385	433	417	465	449	497	481	529	513	561	545
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242	258	274	290	306	322	338	354	370	386	402	418	434	450	466	482	498	514	530	546	562
19	3	51	35	83	67	115	99	147	131	179	163	211	195	243	227	259	275	307	291	339	323	371	355	387	403	435	419	467	451	499	483	515	531	563	547
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244	260	276	292	308	324	340	356	372	388	404	420	436	452	468	484	500	516	532	548	564

Table 5.5(c) First Five Rows of Addresses for N_{cbps} = 576, ½ Code rate, 64-QAM

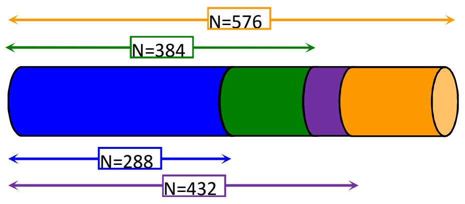
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320	336	352	368	384	400	416	432	448	464	480	496	512	528	544	560
17	33	1	65	81	49	113	129	97	161	177	145	209	225	193	257	273	241	305	321	289	353	369	337	401	417	385	449	465	433	497	513	481	545	561	529
34	2	18	82	50	66	130	98	114	178	146	162	226	194	210	274	242	258	322	290	306	370	338	354	418	386	402	466	434	450	514	482	498	562	530	546
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243	259	275	291	307	323	339	355	371	387	403	419	435	451	467	483	499	515	531	547	563
20	36	4	68	84	52	116	132	100	164	180	148	212	228	196	260	276	244	308	324	292	356	372	340	404	420	388	452	468	436	500	516	484	548	564	532



(a)



(b)



(c)

Fig. 5.7 Relationship between de-interleaver memory address with various N_{cbps} (= N) and (a) QPSK modulation scheme (b) 16-QAM modulation scheme (c) 64-QAM modulation scheme

5.6.2 Proposed hardware for the address generator

The hardware structure of the proposed LUT based address generator for WiMAX de-interleaver is shown in Fig. 5.8. The complete hardware is divided into two parts: LUT address generator block and LUT block. The former consists of ROMs, multiplexers and an up counter responsible for generating the memory address (*icount*) required to read the address LUTs. The ROMs store the terminal values of each row as input and the starting values of the next row as the output. The column counter counts up to the desired column and then gets reloaded with another *preset* value representing the starting memory address of the next row from the appropriate ROM selected by *mod typ* and *code rate* signals. The content of ROMs used to implement N_{cbps} = 96 of QPSK (ROM_00_000) and N_{cbps} = 192 of 16-QAM (ROM_01_X00) are presented in Table 5.6 (a) and (b). Similar contents are available in other ROMs.

The latter block contains the three address LUTs storing the de-interleaver addresses for the three modulation schemes. The multiplexer arrangement along with values in the *mod typ* ensures selection of proper address LUT for a particular modulation scheme. The selected address LUT is read using *icount* and the de-interleaver addresses are made available at the *address* output line.

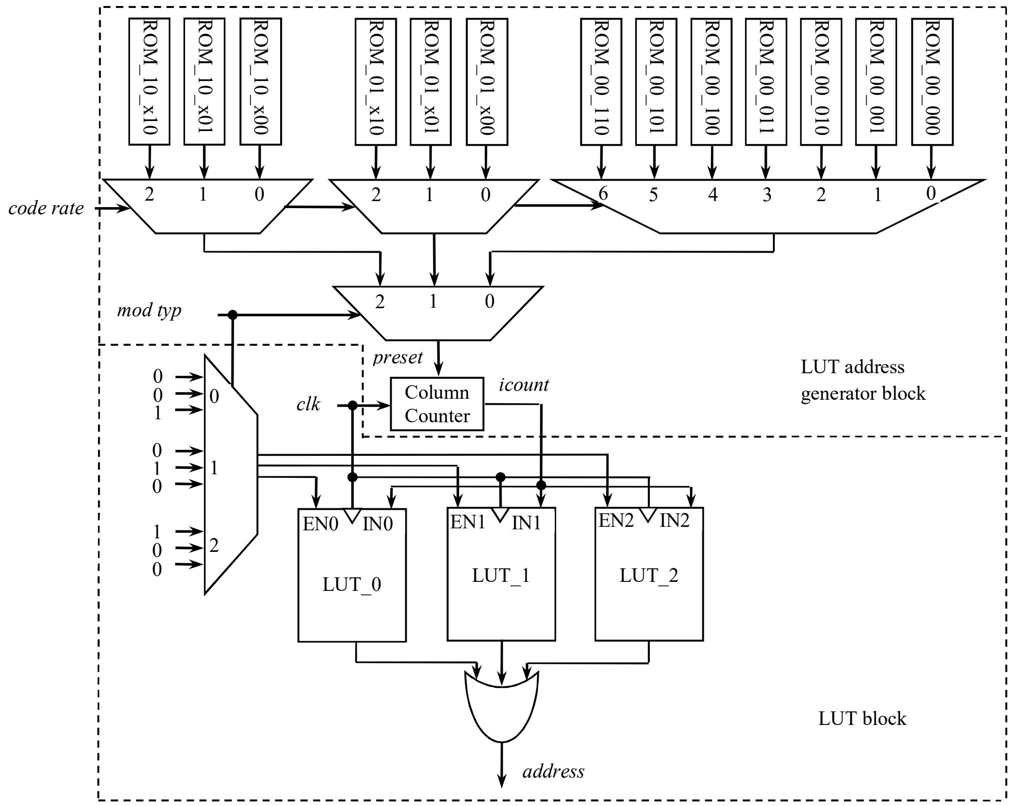


Fig. 5.8 Detailed hardware structure of proposed address generator

Table 5.6 (a) Content of ROM_00_000

Input	Output	Input	Output
5	36	293	324
41	72	329	360
77	108	365	396
113	144	401	432
149	180	437	468
185	216	473	504
221	252	509	540
257	288	545	0

Table 5.6 (b) Content of ROM_01_x00

Input	Output	Input	Output
11	36	299	324
47	72	335	360
83	108	371	396
119	144	407	432
155	180	443	468
191	216	479	504
227	252	515	540
263	288	551	0

5.7 Proposed Algorithm for Low complexity De-interleaver

In this section, the proposed algorithm for Address Generator of WiMAX de-interleaver along with its mathematical background has been described. The MATLAB program as done in LUT based approach and also as described by Fig. 5.6 is used to get the de-interleaver addresses for all modulation schemes and code rates. Due to the presence of floor function in (5.3) and (5.4), direct implementation of them on FPGA chip is not feasible. Table 5.7 shows the de-interleaver addresses for first 4 rows and 5 columns of each modulation type. As d =16 [155] is chosen, the number of rows are fixed (=d) for all N_{cbps} whereas the number of columns are given by N_{cbps}/d.

Close examination of the addresses in Table 5.7 reveals that the co-relation between them follows the manner as shown in Table 5.8. The mathematical foundation of the co-relation between the addresses, as derived in this work is represented by (5.5)-(5.7).

$$k_{n,QPSK} = \{ d * i + j \qquad \text{for } \forall\, j \text{ and } \forall\, i \tag{5.5}$$

$$k_{n,16\text{-}QAM} = \begin{cases} d * i + j & \text{for } j\%2 = 0 \text{ and for } \forall\, i \\ d * (i+1) + j & \text{for } j\%2 = 1 \text{ and for } i\%2 = 0 \\ d * (i-1) + j & \text{for } j\%2 = 1 \text{ and for } i\%2 = 1 \end{cases} \tag{5.6}$$

$$k_{n,64\text{-}QAM} = \begin{cases} d * i + j & \text{for } j\%3 = 0 \text{ and for } \forall\, i \\ d * (i-2) + j & \text{for } j\%3 = 1 \text{ and for } i\%3 = 2 \\ d * (i+1) + j & \text{for } j\%3 = 1 \text{ and for } i\%3 \neq 2 \\ d * (i+2) + j & \text{for } j\%3 = 2 \text{ and for } i\%3 = 0 \\ d * (i-1) + j & \text{for } j\%3 = 2 \text{ and for } i\%3 \neq 0 \end{cases} \tag{5.7}$$

where j = 0,1,…d-1 and i = 0,1,…,$\left(\frac{Ncbps}{d}\right)$-1 represent the row and column numbers respectively of Table 5.8. Also, k_n represents the de-interleaver addresses.

General validity of (5.5)-(5.7) to represent the co-relation between the addresses of Table 5.8 has formally been proved using the algebraic analysis in [155] which lacks the involvement of (5.5)-(5.7). The outcome of this analysis using (5.5)-(5.7) provides the same result as shown in Table 5.8. Thus (5.5)-(5.7) play the pivotal role in establishing formal mathematical foundation of our proposed algorithm.

From Table 5.8 and mathematical representation by (5.5)-(5.7), following three algorithms for the three modulation schemes are proposed. These algorithms eliminate the requirement of floor function while generating write addresses and have also been tested on MATLAB. Results obtained are verified with the previous MATLAB program for all code rates and modulation schemes of WiMAX de-interleaver.

Table 5.7 First 4-rows and 5-columns of De-interleaver Sample Addresses for Three Code Rates and Modulation Types

N_{cbps}, code rate and modulation type	De-interleaver addresses				
N_{cbps} = 96-bits, ½ code rate, QPSK	0	16	32	48	64
	1	17	33	49	65
	2	18	34	50	66
	3	19	35	51	67
N_{cbps} = 192-bits, ½ code rate, 16-QAM	0	16	32	48	64
	17	1	49	33	81
	2	18	34	50	66
	19	3	51	35	83
N_{cbps} = 576-bits, ¾ code rate, 64-QAM	0	16	32	48	64
	17	33	1	65	81
	34	2	18	82	50
	3	19	35	51	67

Table 5.8 Determination of Co-relation between Addresses

Row no.(j)	Column no. (i) →	0	1	2	3	4
0	N_{cbps} = 96-bits, ½ code rate, QPSK	*d*.0+0=0	*d*.1+0=16	*d*.2+0=32	*d*.3+0=48	*d*.4+0=64
1		*d*.0+1=1	*d*.1+1=17	*d*.2+1=33	*d*.3+1=49	*d*.4+1=65
2		*d*.0+2=2	*d*.1+2=18	*d*.2+2=34	*d*.3+2=50	*d*.4+2=66
3		*d*.0+3=3	*d*.1+3=19	*d*.2+3=35	*d*.3+3=51	*d*.4+3=67
0	N_{cbps} = 192-bits, ½ code rate, 16-QAM	*d*.0+0=0	*d*.1+0=16	*d*.2+0=32	*d*.3+0=48	*d*.4+0=64
1		*d*.1+1=17	*d*.0+1=1	*d*.3+1=49	*d*.2+1=33	*d*.5+1=81
2		*d*.0+2=2	*d*.1+2=18	*d*.2+2=34	*d*.3+2=50	*d*.4+2=66
3		*d*.1+3=19	*d*.0+3=3	*d*.3+3=51	*d*.2+3=35	*d*.5+3=83
0	N_{cbps} = 576-bits, ¾ code rate, 64-QAM	*d*.0+0=0	*d*.1+0=16	*d*.2+0=32	*d*.3+0=48	*d*.4+0=64
1		*d*.1+1=17	*d*.2+1=33	*d*.0+1=1	*d*.4+1=65	*d*.5+1=81
2		*d*.2+2=34	*d*.0+2=2	*d*.1+2=18	*d*.5+2=82	*d*.3+2=50
3		*d*.0+3=3	*d*.1+3=19	*d*.2+3=35	*d*.3+3=51	*d*.4+3=67

QPSK

```
    initialize $N_{cbps}$ and $d$
    for $j = 0$ to $d-1$, $j$++
        for $i = 0$ to $(N_{cbps}/d) - 1$, $i$++
            $k_n = d * i + j$
        end for
    end for
```

```
16-QAM
    initialize $N_{cbps}$ and $d$
    for $j$ = 0 to $d-1$, $j$++
        for $i$ = 0 to $(N_{cbps}/d) - 1$, $i$++
            if ($j$ mod 2 = 0)
                $k_n = d * i + j$
            else
                if ($i$ mod 2 = 0)

                    $k_n = d * (i+1) + j$

                else
                    $k_n = d * (i-1) + j$
                end if
            end if
        end for
    end for
```

```
64-QAM
    initialize $N_{cbps}$ and $d$
    for $j$ = 0 to $d$-1, $j$++
        for $i$ = 0 to $(N_{cbps}/d) - 1$, i++
            if ($j$ mod 3= 0)
                $k_n = d * i + j$
            elseif($j$ mod 3= 1)
                if ($i$ mod 3 = 2)
                    $k_n = d * (i-2) + j$
                else
                    $k_n = d * (i+1) + j$
                end if
            else
                if ($i$ mod 3 = 0)
                    $k_n = d * (i+2) + j$
                else
                    $k_n$ = d * $(i-1) + j$
                end if
            end if
        end for
    end for
```

5.7.1 Transformation into Circuit

In order to test the proposed algorithms for Address Generator of WiMAX de-interleaver with all modulation schemes, transformation of these algorithms into digital circuits are made and are shown in Fig. 5.9(a)-(c). The QPSK hardware shown in Fig. 5.9(a), has a row counter (RWC_0) to generate row numbers between 0 to d-1. A column counter (CLC_0) with multiplexer (M_0) and comparator (C_0) generate the variable column numbers to implement permissible N_{cbps}. A multiplier (ML_0) and an adder (A_0) perform the desired operations to implement (5.5). The Address Generator for 16-QAM follows similar structure, like that of QPSK with few additional modules. These modules are designed with an incrementer, a decrementer, two modulo-2 blocks and two multiplexers as shown in Fig. 5.9(b). As per Table 5.7, in 64-QAM modulation scheme, the Address Generator has to implement three different progressive patterns for the column numbers. The design procedure used in 16-QAM is extended in 64-QAM to meet this requirement with the use of additional hardware and is shown in Fig. 5.9(c). A simple up counter generates the read addresses for the 2D-de-interleaver.

The top-level structure of the de-interleaver Address Generator is shown in Fig. 5.10. Logic circuits shown inside the dashed line in Fig. 5.9(a)-(c) are presented here as QPSK block, 16-QAM block and 64-QAM block. Our design is optimized in the sense that common logic circuits like multiplier, adder, row counter and column counter are shared while generating addresses for any modulation type. In addition, the design also shares the incrementer and decrementer required in 16-QAM and 64-QAM blocks.

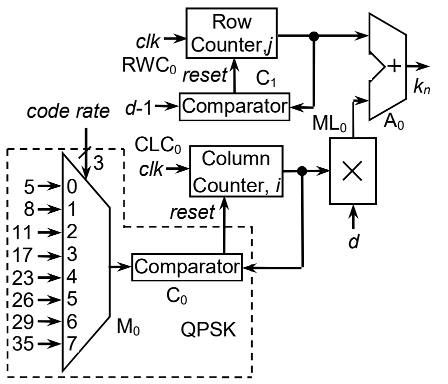


Fig. 5.9(a) Hardware structure of Address Generator for QPSK

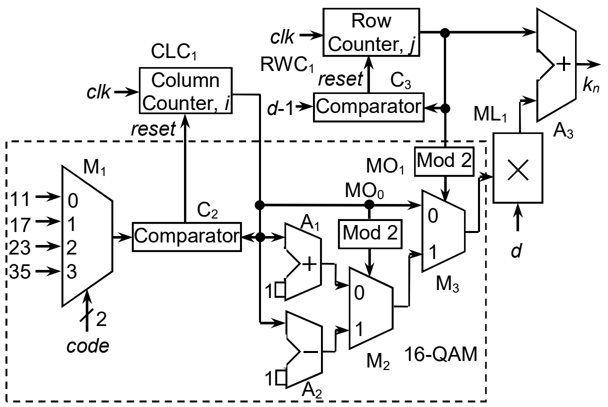


Fig. 5.9(b) Hardware structure of Address Generator for 16-QAM

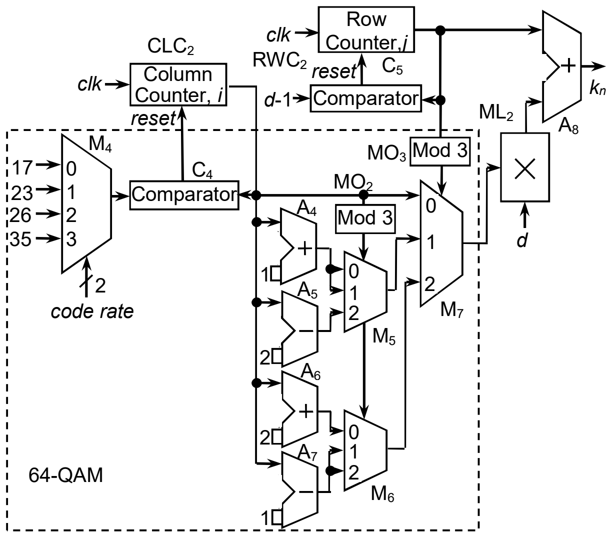


Fig. 5.9(c) Hardware structure of Address Generator for 64-QAM

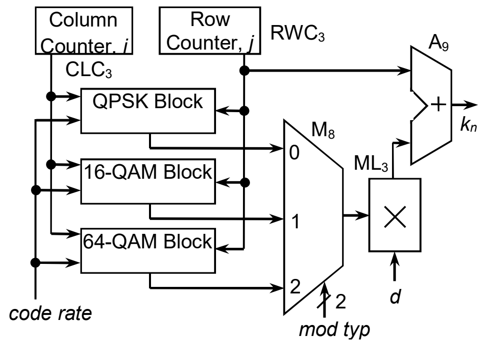


Fig. 5.10 Top level view of complete de-interleaver Address Generator

5.8 Simulation Results

5.8.1 FSM based Address Generator of Interleaver

The simulation results are obtained in the form of timing diagram using ModelSim Xilinx Edition-III version 6.0a software. In order to have a clear picture of the proposed technique the simulation result of the address generator and the complete interleaver have been presented separately.

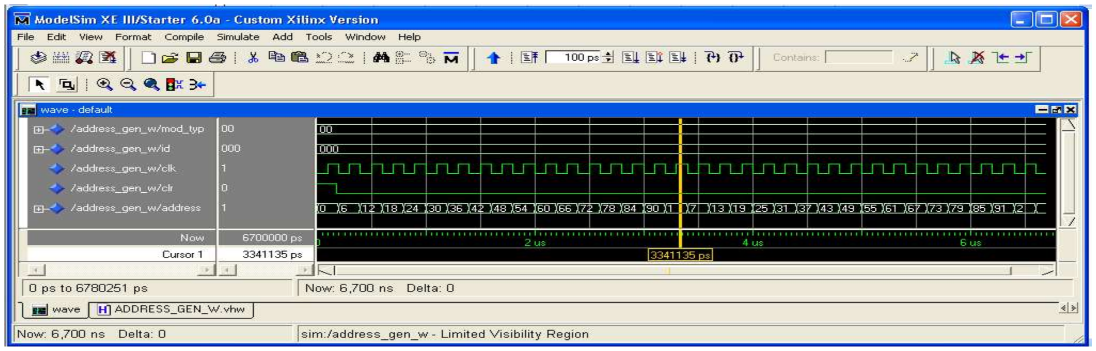
5.8.1.1 Address Generator

Simulation results of address generator are described in Fig. 5.11(a), (b) and (c). Fig. 5.11(a) is for MOD_TYP = 00 and ID = 000 i.e. QPSK with N_{cbps} = 96 as described in Table 5.3. In Fig. 5.11(b) simulation result of 16-QAM with N_{cbps} = 288 (MOD_TYP =

01 and ID=001) is presented. Similarly address generation for 64-QAM with N_{cbps} = 384 (MOD_TYP = 10 and ID=001) is shown in Fig. 5.11(c). In all the figures, initially CLR = 1 to ensure that the counter in preset logic and accumulator are reset. In order to maintain clarity, only first two iterations for the three situations have been presented. Addresses generated in Fig. 5.11(a), (b) and (c) clearly conform to Table 5.2. The author has simulated and tested the address generation circuitry for all other values of ID and MOD_TYP, however to avoid repetition other situations are not shown.

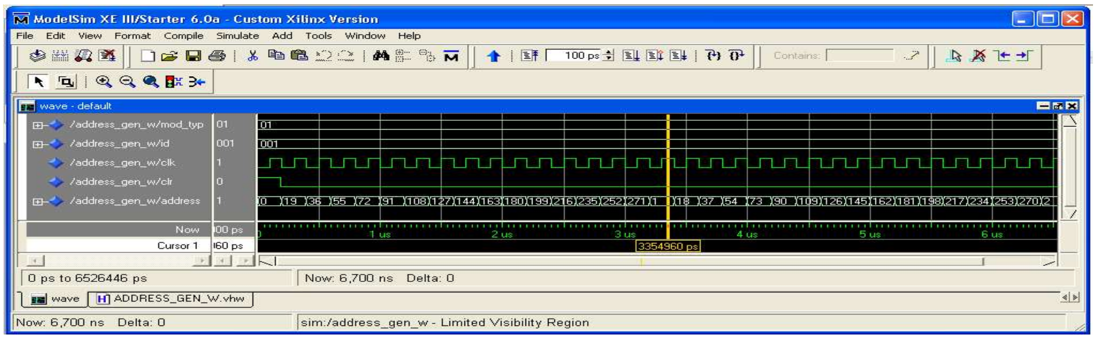
5.8.1.2 Complete Interleaver

Fig. 5.12(a), (b) and (c) explain the interleaving operation of the proposed interleaver for WiMAX system. In these figures the modulation types and interleaver depths chosen are identical with Fig. 5.11(a), (b) and (c) respectively. The raw data input (data_in) into the interleaver in Fig. 5.12(a), (b) and (c) are held high for first 16 consecutive bit duration and made low thereafter to have clear view of the interleaving operation. As seen in figures these consecutive bits are dispersed by a predefined interval which is 6 in Fig. 5.12(a), 19, 17 in Fig. 5.12(b) and 26, 23, 23 in Fig. 5.12(c) and conforms to Table 5.7.



```
/address_gen_w/mod_typ   00     00
/address_gen_w/id        000    000
/address_gen_w/clk       1
/address_gen_w/clr       0
/address_gen_w/address   1      0 6 12 18 24 30 36 42 48 54 60 66 72 78 84 90 1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91 2
Now        6700000 ps
Cursor 1   3341135 ps     3341135 ps
0 ps to 6780251 ps        Now: 6,700 ns   Delta: 0
Now: 6,700 ns   Delta: 0   sim:/address_gen_w - Limited Visibility Region
```

Fig. 5.11(a) Generation of first 32 write addresses with MOD_TYP = 00, ID = 000



```
/address_gen_w/mod_typ   01     01
/address_gen_w/id        001    001
/address_gen_w/clk       1
/address_gen_w/clr       0
/address_gen_w/address   1      0 19 36 55 72 91 108 127 144 163 180 199 216 235 252 271 1 18 37 54 73 90 109 126 145 162 181 198 217 234 253 270 2
Now        00 ps
Cursor 1   60 ps          3354960 ps
0 ps to 6526446 ps        Now: 6,700 ns   Delta: 0
Now: 6,700 ns   Delta: 0   sim:/address_gen_w - Limited Visibility Region
```

Fig. 5.11(b) Generation of first 32 write addresses with MOD_TYP = 01, ID = 001

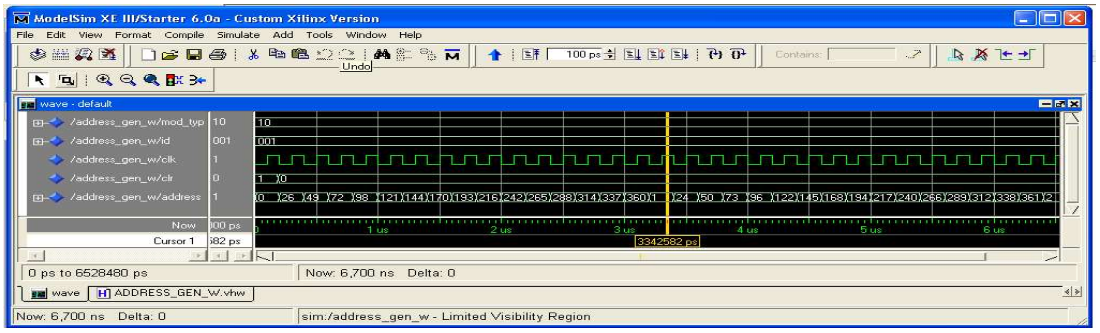


Fig. 5.11(c) Generation of first 32 write addresses with MOD_TYP = 10, ID = 001

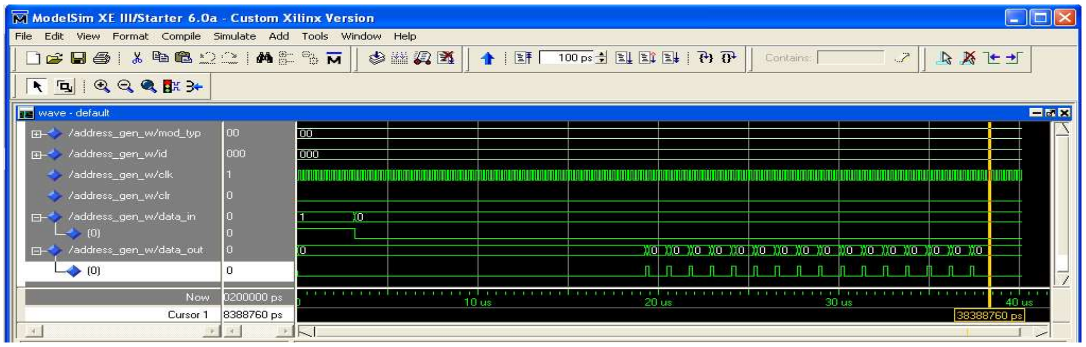


Fig. 5.12(a) Interleaving operation with MOD_TYP = 00, ID = 000

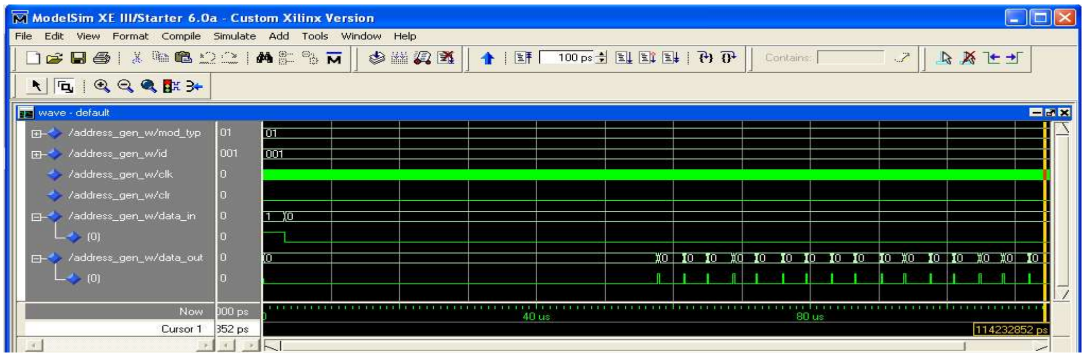


Fig. 5.12(b) Interleaving operation with MOD_TYP = 01, ID = 001

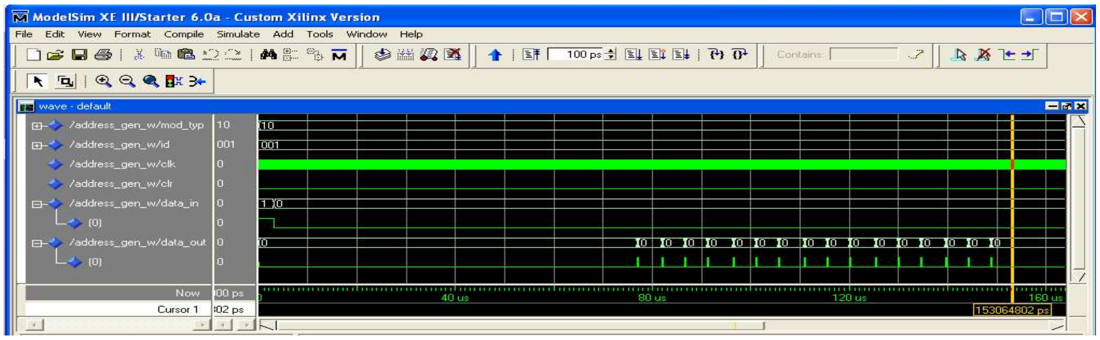
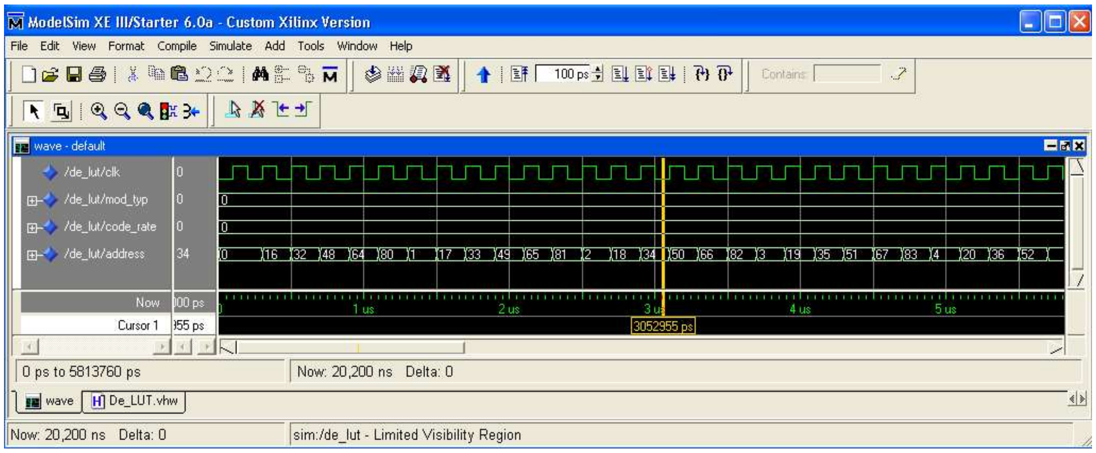


Fig. 5.12(c) Interleaving operation with MOD_TYP = 10, ID = 001

5.8.2 LUT based Address Generator of De-interleaver

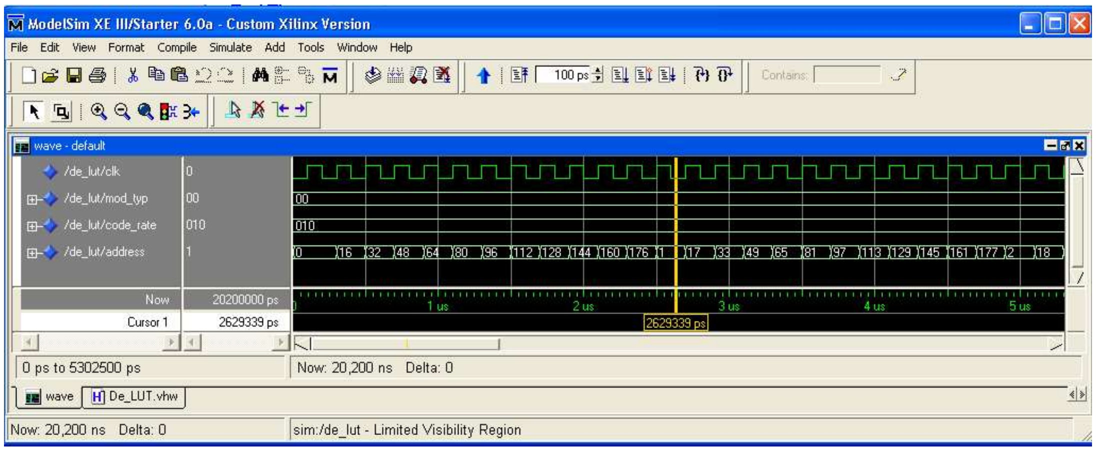
The simulation results in the form of timing diagram obtained using ModelSim Xilinx Edition-III, version 6.0a of LUT based address generator of WiMAX de-interleaver are shown in Fig. 5.13(a)-(c). In Fig. 5.13(a), MOD_TYP = 0 (00_2) and CODE_RATE = 0 (000_2). The sequence of addresses generated are 0, 16, 32, 48, 64, 80, 1, 17, 33, 49, 65, 81, 2 …. which clearly conform to N_{cbps} = 96 with QPSK modulation and ½ code rate. Similarly, Fig. 5.13(b) and (c) show generation of de-interleaver address sequence for N_{cbps} = 192 with QPSK modulation, ½ code rate (MOD_TYP = 00, CODE_RATE = 010) and N_{cbps} = 288 with QPSK modulation, ¾ code rate (MOD_TYP = 00, CODE_RATE = 011) respectively.

The author has simulated and tested the address generation circuitry for all other values of CODE_RATE, MOD_TYP, however in order to avoid repetition, other situations are not shown.


ModelSim XE III/Starter 6.0a - Custom Xilinx Version
File Edit View Format Compile Simulate Add Tools Window Help
100 ps
Contains:
wave - default
wave
De_LUT.vhw

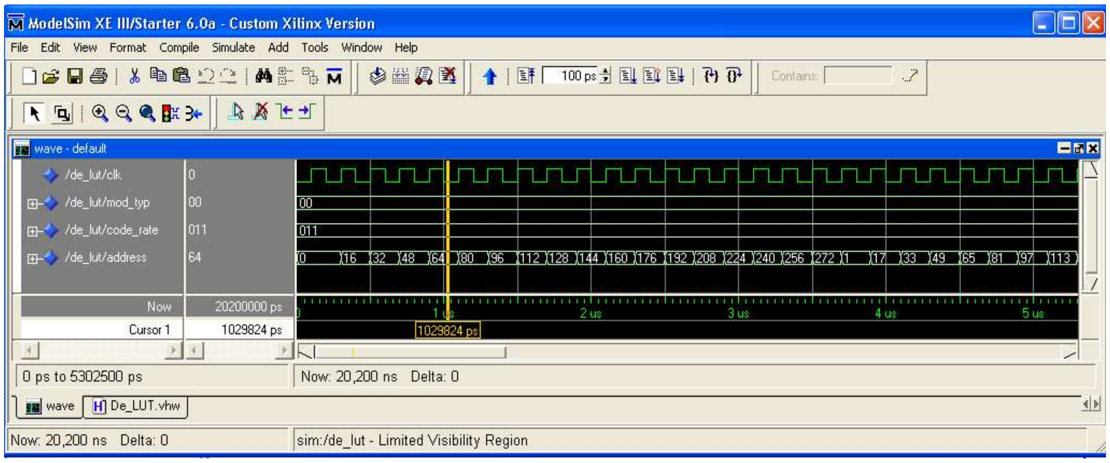
/de_lut/clk 0
/de_lut/mod_typ 0 — 0
/de_lut/code_rate 0 — 0
/de_lut/address 34 — 0 16 32 48 64 80 1 17 33 49 65 81 2 18 34 50 66 82 3 19 35 51 67 83 4 20 36 52
Now 00 ps — 1 us 2 us 3 us 4 us 5 us
Cursor 1 55 ps — 3052955 ps
0 ps to 5813760 ps — Now: 20,200 ns Delta: 0
Now: 20,200 ns Delta: 0 — sim:/de_lut - Limited Visibility Region

(a)


ModelSim XE III/Starter 6.0a - Custom Xilinx Version
File Edit View Format Compile Simulate Add Tools Window Help
100 ps
Contains:
wave - default
wave
De_LUT.vhw

/de_lut/clk 0
/de_lut/mod_typ 00 — 00
/de_lut/code_rate 010 — 010
/de_lut/address 1 — 0 16 32 48 64 80 96 112 128 144 160 176 1 17 33 49 65 81 97 113 129 145 161 177 2 18
Now 20200000 ps — 1 us 2 us 3 us 4 us 5 us
Cursor 1 2629339 ps — 2629339 ps
0 ps to 5302500 ps — Now: 20,200 ns Delta: 0
Now: 20,200 ns Delta: 0 — sim:/de_lut - Limited Visibility Region

(b)



(c)

Fig.5.13 Simulation result of LUT based De-interleaver Address Generator with (a) MOD_TYP = 00, CODE_RATE = 000 (b) MOD_TYP = 00, CODE_RATE = 010 and (c) MOD_TYP = 00, CODE_RATE = 011

5.8.3 Low Complexity Address Generator of De-interleaver

The proposed hardware of the low complexity address generator is converted into a VHDL program using Xilinx ISE. Simulation results are obtained for all permissible modulation types and code rates using ModelSim XE-III and a part of the same for N_{cbps} = 576-bits, ¾ code rate, 64-QAM has been presented in Fig. 5.14. The initial portion of Fig. 5.14 shows the last part of addresses for first row (j=1) and the later part (from ruler) shows the addresses for second row (j=2). The simulation results are verified with the output obtained from the MATLAB program described in Table 5.2.

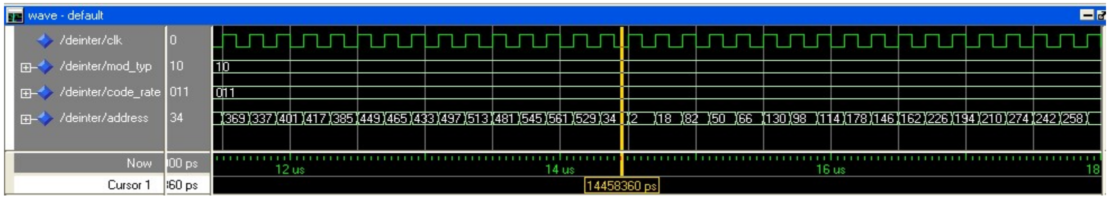


Fig. 5.14 Simulation result showing the addresses of last part of first row (j=1) and first portion of second row (j=2) for N_{cbps} = 576-bits, ¾ code rate, 64-QAM

5.9 Critical Analysis of FPGA implementation Results

5.9.1 FSM based Address Generator of Interleaver

The proposed hardware model of FSM based WiMAX interleaver is implemented and tested on Xilinx Spartan-3 (Device: XC3S400) FPGA platform in the laboratory. The

FPGA implementation of the interleaver is carried out in two phases; firstly the address generator and thereafter the complete interleaver and presented accordingly.

5.9.1.1 Address Generator

A VHDL model of the proposed FSM based address generation hardware is prepared using Xilinx ISE 8.1i and thereafter implemented in the said FPGA. In order to make comparative analysis we have also designed and implemented address generator circuitry for the interleaver depths listed in [106] and result is presented in Table 5.9. Our approach shows approximately 30% improvement in terms of maximum operating clock frequency, approximately 46% improvement in FPGA flip-flop used with negligible (less than 3%) loss in terms of Logic Cells (LCs) used. Careful design of the preset logic provides this improvement. Table 5.10 and 5.11 shows the HDL synthesis report and device utilization summary corresponding to the implementation of the circuit shown in Fig. 5.3. Minimum propagation delay of the circuit is measured to be 5.234ns and maximum operating frequency is 191.05MHz. The estimated power consumption of the circuit is found to be 56mW using Xilinx XPower I.25.

The address generator circuit when implemented on recent FPGAs like Virtex 4 shows further betterment in terms of operating frequency (278.30MHz) but at the cost of increased power consumption (224mW). As these FPGAs offer a large number of resources the utilization percentage as shown in Table 5.11 further goes down.

Fig. 5.15(a), (b) and (c) show the moments captured (second addresses) during the progress of address generation circuitry on FPGA platform with ID and MOD_TYP shown in Fig. 5.15(a), (b) and (c) respectively. As shown in Fig. 5.15(b), the first toggle switch is used as CLR (clear) input, next three implements ID (interleaver depth) and last two represents MOD_TYP (modulation type) whose values are described in Table 5.3. Similarly, the first 10 LEDs from the left represents the address generated with rightmost LED representing the MSB.

Table 5.9 Comparative analysis of similar implementations of address generator

Implementation Technique	Number of LCs used	Number of flip-flops used	Improvement in flip-flop used	Maximum Clock Frequency (MHz)	Improvement in Max. Clock Frequency
Khater et. al. [106]	105	54	45.95%	147.9	29.14%
Our implementation	108	37		191.05	

Table 5.10 HDL synthesis report of FSM based address generator

Logic Circuit used	Quantity
8x7-bit ROM	1
10-bit adder	2
2-bit adder	1
4-bit up counter	1
Flip-flops	18
10-bit latch	1
7-bit latch	1
7-bit 4-to-1 multiplexer	2
7-bit 8-to-1 multiplexer	1

Table 5.11 Device utilization summary of FSM based address generator

FPGA Resources	Utilization in Number	Utilization in %
Number of Logic Cell (LC)	242 out of 3584	6.75
Number of Flip-flops	48 out of 7168	0.67
Number of Bonded IOBs	17 out of 141	12.06
Number of GCLKs	2 out of 8	25.00

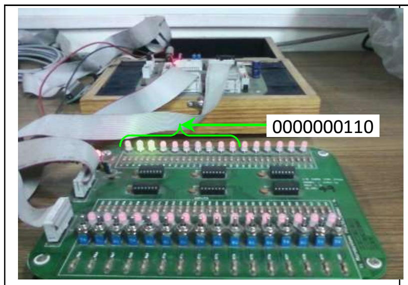


Fig. 5.15(a) Photograph with ID=000, MOD_TYP=00

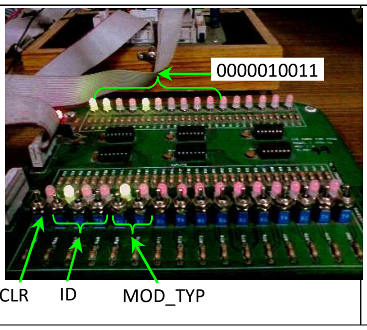


Fig. 5.15(b) Photograph with ID=001, MOD_TYP=01

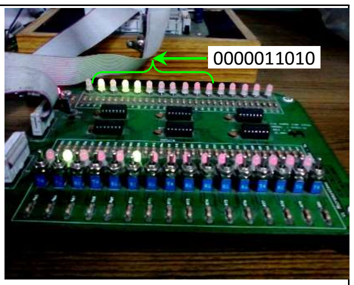


Fig. 5.15(c) Photograph with ID=001, MOD_TYP=10

5.9.1.2 Complete Interleaver

This section makes the critical analysis of FPGA implementation results of the entire interleaver including the proposed FSM based address generator. The HDL synthesis report of the complete interleaver is presented in Table 5.12. It shows additional requirement of few flip-flops/latches and multiplexers which are used in designing the memory module of the interleaver.

Table 5.12 HDL synthesis report of the complete interleaver

Logic Circuit used	Quantity
8x7-bit ROM	1
10-bit adder	2
2-bit adder	1
4-bit up counter	1
Flip-flops	23
10-bit latch	1
7-bit latch	3
1-bit latch	1
7-bit 4-to-1 multiplexer	2
7-bit 8-to-1 multiplexer	1
1-bit 4-to-1 multiplexer	5

Device utilization summary of the complete interleaver implementation is described in Table 5.13. The utilization percentage of LCs and flip-flops are marginally increased because of the associated circuitry in the memory module of the interleaver. Number of Input Output Blocks (IOBs) has been dropped by 8 as because the 10-bit address output lines of address generator have been replaced by 2 lines; one carrying raw input data and the other sending out the interleaved data. The interleaver utilizes two BRAMs which is 12.5% of the available BRAM blocks in Spartan-3 FPGA. Minimum propagation delay and maximum operating frequency of the FPGA based interleaver is found to be 7.442ns and 134.381MHz respectively. Due to efficient modeling, the interleaver circuitry uses very few FPGA resources thereby making room for other associated circuitry like randomizer, encoder etc to be implemented on the same FPGA chip. Because of the presence of floor and mod function in (5.1) and (5.2), direct implementation of the address generation circuitry is very complex and consumes large amount of logic resources. Instead, our state machine based approach provides a faster and resource efficient implementation of WiMAX interleaver on FPGA platform.

Table 5.13 Device Utilization Summary of Complete Interleaver

FPGA Resources	Utilization in Number	Utilization in %
Number of LCs	267 out of 3584	7.45
Number of Flip-flops	54 out of 7168	0.75
Number of Bonded IOBs	9 out of 141	12.06
Number of GCLKs	2 out of 8	25.00
Number of BRAMs	2 out of 16	12.50

5.9.2 LUT based Address Generator of Interleaver

The proposed hardware structure of LUT based de-interleaver address generator is transformed into VHDL model using Xilinx Integrated Software Environment (ISE 8.1) and is implemented on Xilinx Spartan 3 FPGA (XC3S400). Additionally, the hardware structure is also implemented on Xilinx Spartan-3AN FPGA (XC3S1400AN) using ISE 12.1. Table 5.14 shows the device utilization summary for both implementations. The two implementations are almost identical in terms of FPGA resource utilizations, but differ significantly in operating frequency and estimated power consumption. It is observed that the design implemented on advanced FPGA (Spartan-3AN) works faster by 30% than the other, but also consumes double amount of power. The principal advantage of our proposed technique is that it requires only 3 BRAMs of capacity 18KB instead of 16, saving 81.25% of critical FPGA internal resource.

Based on the equivalence drawn between FPGA and ASIC implementations in [156] our work is compared with that of [105] by converting the later in FPGA equivalent implementation. This comparison shows our implementation on Spartan 3 FPGA is at par with [105] in terms of operating frequency. But, the implementation on Spartan 3AN shows improvement of almost 30% over [105] as FPGA equivalent maximum frequency of the later is found to be 62.5MHz.

Table 5.14 Device Utilization Summary of LUT based Address Generator of WiMAX De-interleaver

FPGA Resources / Parameters	Resource Utilization / Parameters in Spartan 3	Resource Utilization / Parameters in Spartan 3AN
Number of slices	633 out of 3584	626 out of 11264
Number of slice Flip-flops	56 out of 7168	41 out of 22528
Number of 4 input LUTs	1229 out of 7168	1205 out of 22528
Number of bonded IOBs	16 out of 141	16 out of 502
Number of BRAMs	3 out of 16	3 out of 32
Number of GCLKs	1 out of 8	1 out of 24
Maximum clock speed	62.5 MHz	88.72 MHz
Power consumption	32mW	68mW

5.9.3 Low Complexity Address Generator of Interleaver

The VHDL program developed for the proposed WiMAX de-interleaver Address Generator is downloaded on Xilinx Spartan-3 (Device XC3S400) FPGA [94]. Table 5.15 shows the HDL synthesis report. The two blocks, MO_0 and MO_1 of Fig. 5.9(b) are implemented using mod 2^n function of VHDL. Requirement of i mod 3 (MO_2) and j mod 3 (MO_3) functions in 64-QAM circuit of Fig. 5.9(c) are fulfilled by designing two small ROMs of dimension 16x3-bit and 64x3-bit respectively as MOD 3 function is not supported in VHDL. The use of rest of the logic circuits is obvious in the design.

As FPGA based implementation of WiMAX de-interleaver Address Generator has not been found in the literature, direct comparison of the results of our proposed work could not be carried out. However, implementation of the conventional LUT based technique of address generation for WiMAX 2D-de-interleaver on the same FPGA is made in the similar manner as proposed in [157]. In the latter case, the LUTs are modeled using FPGA's embedded memory, Block RAM [94], to reduce the memory access time. For fairness of comparison, three Block RAMs are used, one for each modulation scheme to house the address LUT of various interleaver depths. Efficient use of Block RAMs is made possible by exploring the feature that, within a modulation scheme the address LUT of a smaller N_{cbps} is the subset of the address LUT of larger N_{cbps}.

Table 5.16 shows the comparison between the two implementations in respect to FPGA resources. In spite of smart use of Block RAM in LUT based approach, the present work results in significant reduction in occupancy of FPGA slices (by 80.24%), flip-flops (by 35.9%) and 4 input LUTs (by 80.47%). This comparison clearly proves the low complexity and hardware efficiency of our design over the conventional technique. Further, to make the design more hardware efficient, embedded multiplier of Xilinx Spartan-3 FPGA is used to implement the ML_3 block of Fig. 5.10. In addition, the Address Generator using the proposed technique can work 48.69% faster than the later.

Further, based on the equivalence drawn between FPGA and ASIC implementations in [156], our work is again compared with that of [105], by converting the later in FPGA equivalent implementation. This comparison also shows almost 48.69% improvement in our work with respect to operating frequency over [105] as FPGA equivalent maximum frequency of the later is found to be 62.5MHz. The reasons behind these improvements are low complexity, optimized and shared hardware design and use of FPGA's embedded multiplier which in turn reduces interconnection delay inside FPGA. Similar comparison with [158] is not useful as this work is not focused on our target design only. Both CTC and Long Term Evolution (LTE) interleavers do not use floor function for FPGA implementation, while the present work involves use of floor function for such realization in an efficient manner. The work in [159] is based on FSM based technique for designing Address Generator using FPGA for channel interleaver employed in WiMAX transmitter. However, this work involves multiplier based Address Generator in FPGA platform for designing channel de-interleaver in WiMAX receiver. The conventional LUT based approach and our proposed algorithm, both targeting same Address Generator is implemented on the identical FPGA platform and accordingly effective comparison as in Table 5.16 becomes possible.

Table 5.15 HDL Synthesis Report of Low Complexity Address Generator

Logic Circuits Used	Quantity	Logic Circuits Used	Quantity
16x3-bit ROM	1	4-bit register	1
64x3-bit ROM	1	6-bit register	1
10-bit adder	1	4-bit 4-to-1	3
18-bit adder	2	4-bit 8-to-1	1
18-bit subtractor	2	6-bit 4-to-1	3
4-bit adder	1	6-bit 8-to-1 multiplexer	1
6-bit adder	1		

Table 5.16 Comparison Between Proposed and LUT Based Technique

FPGA Parameters	Performance of proposed technique	Performance of LUT based technique	% Reduction / improvement in resource utilization	Remarks
Slices	3.49 %	17.66 %	-80.24	Significant reduction
Flip-flops	0.50 %	0.78 %	-35.90	Reduction
4 input LUTs	3.35 %	17.15 %	-80.47	Significant reduction
Operating frequency	121.82 MHz	62.51 MHz	94.88	Significant improvement

5.10 Discussion

This chapter describes three different techniques of modelling hardware interleaver /de-interleaver used in IEEE 802.16e based WiMAX transceiver. An interleaver/de-interleaver comprises of two sections: Address generator and Memory module. Due to the presence of modulo and floor functions implementing the address generator, design of hardware module for the same is a difficult task. This is due to the fact that corresponding digital hardware for the two complex functions are not available. Conventionally, LUT based approach is used in which all the addressing sequences are precomputed and stored in external memory. Such approaches consumes external memory blocks and the slow in operation.

In this work, firstly design of a FSM based high speed and hardware efficient technique to implement the address generation circuitry of WiMAX interleaver on FPGA platform has been demonstrated. Secondly, an improved LUT based de-interleaver address generator circuitry is proposed. In this approach, the conventional LUT based technique for address generation has been re-designed to use the FPGA memory blocks efficiently. Design of a low complexity and resource efficient hardware de-interleaver including a novel algorithm for the de-interleaver with user-friendly mathematical representation and its general validity is presented thereafter. This work shows significant performance improvement over LUT technique in terms of enhanced maximum operating frequency and reduced FPGA resource utilization. The low complexity model of interleaver design is carried forward in Chapter 6 for the implementation on MIMO WLAN interleaver due to its attractive performance.

Chapter 6
Interleaving in MIMO WLAN

Outline of this Chapter

The approach described in Chapter 5 while designing low complexity model of WiMAX de-interleaver is adopted in this chapter for the design and implementation of novel interleaver hardware on FPGA platform to be used in OFDM based MIMO WLAN applications. After initial remarks on MIMO WLAN, the chapter briefs about the work done with important contribution made through this research. Novel algorithm with mathematical formulation for the address generator of the interleaver is the key contribution of this chapter. The chapter thereafter describes hardware transformation of the novel algorithm, its timing simulation and FPGA implementation results using the DSP blocks of FPGA unlike the previous implementations. Comparative analysis of the implementation results demonstrates superiority of the proposed design in terms of operating frequency, throughput and power consumption/resource occupancy.

6.1 Introduction

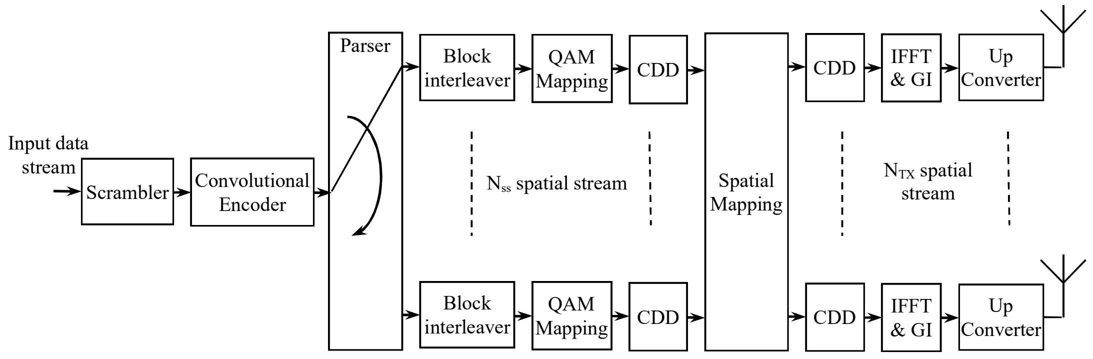
Increasing use of multimedia services and growth of graphics based internet related contents lead to the rising demand of high speed broadband wireless systems. Use of more than one antenna at the transmitter and / or at the receiver aims to improve the transmission / reception rate substantially. OFDM is becoming a popular technique for high data rate wireless transmission [160]. OFDM may be combined with multiple antennas at both the access point and mobile terminal to increase the diversity gain and/or enhance system capacity on a time-varying multipath fading channel, resulting in a MIMO OFDM system [66].

The IEEE 802.11n, an amendment to IEEE 802.11 standard, is based on MIMO-OFDM transmission techniques to enable high speed data communication with maximum throughput of 600 Mbps [161]. The aim of interleaving [162] is to reorder the incoming data and make the adjacent bits non-adjacent by a factor, to cope with the burst errors occurring during the high throughput transmission of data over the channel. Such rearrangement of data bits helps to improve the performance of FEC techniques. In a fading channel, diversity is the technique adopted to improve the performance of a communication system. In such cases, interleavers are used to improve the system performance by exploiting spatial and frequency diversities.

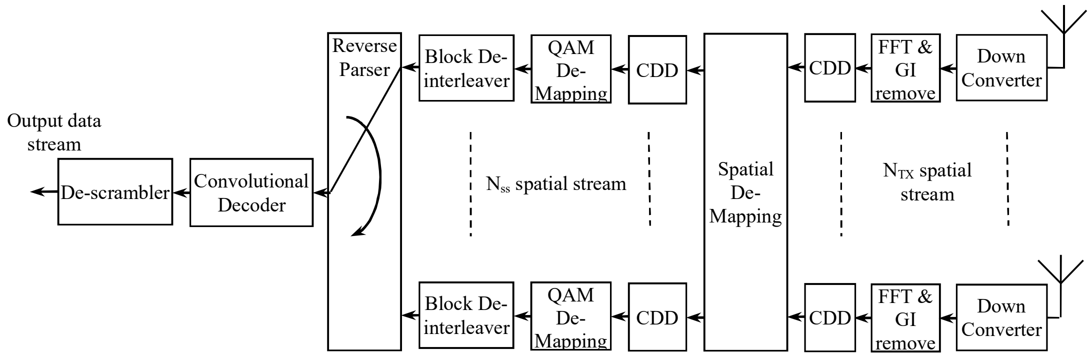
In this chapter, novel design of interleaver used in 4 x 4 MIMO WLAN transceiver has been described. In general term, an interleaver consists of two parts; address generator and interleaver memory. Literature [75] recommends three steps of permutation involving floor function for the implementation of the address generator. Conventional approach of MIMO WLAN interleaver implementation is LUT based [163] due to the non-availability of corresponding digital hardware for floor function. The LUT based technique is in general unattractive [115], as it requires large number of memory blocks to house the address LUTs with various modulation schemes, bandwidths (BWs) and spatial streams. In addition, large access time of memory results in slower operation of the address generator using LUT. In this work, we propose a novel algorithm with necessary mathematical background including its general validity for the address generation of MIMO WLAN interleaver which eliminates the requirement for floor function. The proposed algorithm is also transformed into digital hardware and is modelled in VHDL using Xilinx ISE 12.1. The model utilizes embedded Digital Signal Processing (DSP) blocks of Xilinx Spartan 6 FPGA [96] to implement the multiplier. The memory requirement of the interleaver is also met by configuring the available embedded memory (Block RAM) [96] within the target FPGA. The use of DSP blocks and Block RAM of FPGA makes our design novel and highly resource efficient in comparison with other similar implementations [121], [120], [122]. Further, LUT based technique of interleaver design is modelled and implemented on the same FPGA platform for comparison. In this case also, our proposed design shows improvement in terms of operating frequency and memory utilization over LUT based technique. Necessary hardware required for read address generation is also designed and included in the VHDL model. Simulation results in the form of timing diagram for the address generator is obtained using ModelSim XE-III software and are also verified with the theoretical results. Our design performs better in terms of operating frequency, throughput, and power consumption, compared to the few recent ASIC implementations [121], [120], [122] when they are converted into equivalent FPGA counterparts using [156]. Our proposed design satisfies the maximum throughput requirement of MIMO WLAN of IEEE 802.11n.

6.2 System Description of MIMO WLAN Transceiver

Essential blocks of an MIMO WLAN transceiver are shown in Fig. 6.1(a)-(b) [161], [164]. In the transmitter, input data stream is randomized using a scrambler. The scrambled data passes through a convolutional encoder to reduce the effect of random error in the channel. A parser [164] routes the consecutive data bits based on the number of coded bits per sub-carrier into four (or less) different spatial streams. Every spatial data stream uses one block interleaver to reduce the effect of burst error in the channel. The bit stream thereafter are mapped into QAM symbols. A spatial stream dependent Cyclic Delay (CD) followed by spatial mapping matrix has been applied to each subcarrier to convert N_{ss} spatial stream inputs into N_{tx} transmitter outputs. To provide transmit Cyclic Delay Diversity (CDD) and to prevent undesired beam-forming effects, an additional cyclic delay per transmitter can be applied. Each transmitter thereafter applies an IFFT, inserts a Guard Interval (GI), up-converts and transmits the signal. In the receiver of MIMO WLAN, blocks with reverse function are applied to obtain the original data stream.



(a)



(b)

Fig. 6.1 Block diagram of MIMO WLAN (a) transmitter and (b) receiver

6.3 Interleaving in IEEE 802.11n

The encoded data bits received from convolutional encoder and parser are interleaved by a special type of block interleaver. Interleaving in 802.11n is a three step process in which the first two steps provide spatial interleaving and the final step performs frequency interleaving [161]. The interleaving steps are defined in the form of three blocks shown in Fig. 6.2. The first step (B_1) ensures that adjacent coded bits are mapped onto non-adjacent subcarriers, while the second step (B_2) is responsible for alternating mapping of adjacent coded bits onto less or more significant bits of constellation, thus avoiding long runs of lowly reliable bits. If more than one spatial stream exists in the 802.11n physical layer, the third step called frequency rotation (B_3) would be applied to the additional spatial streams. The frequency rotation ensures that the consecutive carriers used across the spatial streams are not highly correlated.

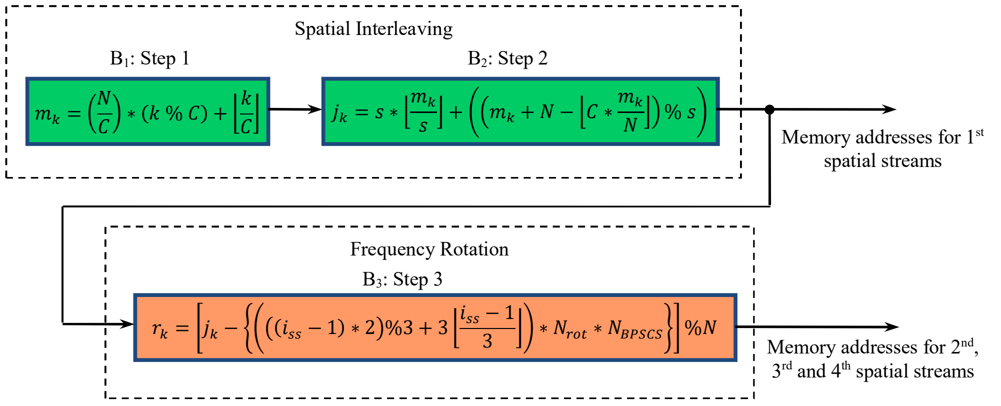

Spatial Interleaving
B1: Step 1
B2: Step 2
$m_k = \left(\frac{N}{C}\right) * (k \% C) + \left\lfloor \frac{k}{C} \right\rfloor$
$j_k = s * \left\lfloor \frac{m_k}{s} \right\rfloor + \left(\left(m_k + N - \left\lfloor C * \frac{m_k}{N} \right\rfloor\right) \% s\right)$
Memory addresses for 1st spatial streams
Frequency Rotation
B3: Step 3
$r_k = \left[j_k - \left\{\left(((i_{ss} - 1) * 2)\%3 + 3\left\lfloor \frac{i_{ss} - 1}{3} \right\rfloor\right) * N_{rot} * N_{BPSCS}\right\}\right] \%N$
Memory addresses for 2nd, 3rd and 4th spatial streams

Fig. 6.2 Block diagram of steps involved in interleaving process for MIMO WLAN

Here N is the block size corresponding to number of coded bits per allocated sub-channels per OFDM symbol. C represents number of columns in the interleaver, whose value is 13 and 18 for 20MHz and 40MHz BW [162] respectively. The parameter s is defined as s = max (1, N_{BPSCS}), whereas N_{BPSCS} is the number of coded bits per sub-carrier, and takes value 1, 2, 4 or 6 for BPSK, QPSK, 16-QAM or 64-QAM respectively. I_{ss} is the index of the spatial stream and N_{rot} is the parameter used for defining different rotation with value 13 and 29 for 20MHz and 40MHz BW respectively. The operators **%** and $\lfloor\ \rfloor$, represent the modulo and floor functions respectively.

6.4 Proposed Algorithm for Address Generator of Interleaver

The permutation steps as described in B1-B3 blocks of Fig. 6.2 involve floor function. LUT based technique is conventionally used due to unavailability of appropriate logic circuit for floor function. Such technique being memory intensive makes the implementation resource inefficient [163]. Also, the interleaver involving LUT based address generator may exhibit slower operation due to large memory access time. In this work, a novel low complexity algorithm for the address generator of channel interleaver used in 4 x 4 MIMO WLAN eliminating the need for floor function has been proposed. The algorithm aims at offering efficient hardware design of the address generator on FPGA platform with an objective to satisfy the throughput requirement for the application.

Table 6.1 shows complete interleaver specifications with all permissible values of modulation schemes, spatial streams and BWs as per IEEE 802.11n [75], [121]. Initially, a MATLAB program has been developed by implementing B1-B3 blocks of Fig. 6.2 to determine the interleaver addresses in similar manner as described in previous two chapters. Table 6.2(a)-(c) show such interleaver write addresses for three cases out of the entire set of 32, e.g. N_{bpscs}=1, N=52, i_{ss}=4; N_{bpscs}=4, N=208, i_{ss}=2; and N_{bpscs}=6, N=312, i_{ss}=3; all with 20MHz BW respectively. Careful examination of these write addresses reveals the appropriate correlation among them, which is being utilized to develop new algorithms for the purpose of successful implementation of write address generator required for MIMO WLAN interleaver as described in Table 6.3(a)-(c). The read addresses could be generated in a conventional manner.

Table 6.1 Interleaver specification of IEEE 802.11n based MIMO WLAN

Modulation Scheme	**Spatial Stream (i_{ss})**	***BW***	**Interleaver Depth (N)**
BPSK (N_{bpscs}=1)	1,2,3,4	20MHz	52
		40MHz	108
QPSK (N_{bpscs}=2)	1,2,3,4	20MHz	104
		40MHz	216
16-QAM (N_{bpscs}=4)	1,2,3,4	20MHz	208
		40MHz	432
64-QAM (N_{bpscs}=6)	1,2,3,4	20MHz	312
		40MHz	648

Table 6.2(a) Interleaver write addresses with N_{bpscs}=1, N=52, i_{ss}=4, BW=20MHz

	Column no(i)							
Row no(j)	0	1	2	…	9	10	11	12
0	13	17	21	…	49	1	5	9
1	14	18	22	…	50	2	6	10
2	15	19	27	…	51	3	7	11
3	16	20	28	…	0	4	8	12

Table 6.2(b) Interleaver write addresses with N_{bpscs}=4, N=208, i_{ss}=2, BW=20MHz

	Column no(i)								
Row no(j)	0	1	2	…	6	7	8	…	12
0	104	121	136	…	200	9	24	…	88
1	105	120	137	…	201	8	25	…	89
2	106	123	138	…	202	11	26	…	90
…	…	…	…	…	…	…		…	…
7	111	126	143	…	207	14	31	…	95
8	112	129	144	…	0	17	32	…	96
9	113	128	145	…	1	16	33	…	97
10	114	131	146	…	2	19	34	…	98
…	…	…	…	…	…	…		…	…
15	119	134	151	…	7	22	39	…	103

Table 6.2(c) Interleaver write addresses with N_{bpscs}=6, N=312, i_{ss}=3, BW=20MHz

	Column no(i)								
Row no(j)	0	1	2	3	4	5	6	…	12
0	234	260	283	306	20	43	66	…	210
1	235	258	284	307	18	44	67	…	211
2	236	259	282	308	19	42	68	…	212
…	…	…	…	…	…	…		…	…
5	239	262	285	311	22	45	71	…	215
6	240	266	289	0	26	49	72	…	216
7	241	264	290	1	24	50	73	…	217
8	242	265	288	2	25	48	74	…	218
9	243	269	292	3	29	52	75	…	219
…	…	…	…	…	…	…	…	…	…
23	257	280	303	17	40	63	89	…	233

Table 6.3(a) Proposed algorithm for N_{bpscs}=1/2 (BPSK/QPSK) with all N, i_{ss} and BW

Column no. (i) →		0	1	2	3	...	C-4	C-3	C-2	C-1
↓ Row no. (j)		i<(C-I)					i>=(C-I)			
0	j<(D-J)	D*(i+I)+(j+J)				...	D*{i-(C-I)}+(j+J)			
1										
2										
3										
...		i<(C-I-1)					i>=(C-I-1)			
D-4	j>=(D-J)	D*(i+I+1)+ {j-(D-J)}				...	D*{i-(C-I-1)}+{j-(D-J)}			
D-3										
D-2										
D-1										

Table 6.3(b) Proposed algorithm for N_{bpscs}= 4 (16-QAM) with all N, i_{ss} and BW

Column no. (i) →		0	1	2	3	...	C-4	C-3	C-2	C-1
↓ Row no. (j)		{i<(C-I)} &(i%2=0)		{i<(C-I)} &(i%2=1)			i>=(C-I) &(i%2=0)		i>=(C-I) &(i%2=1)	
0	{j<(D-J)} &(j%2=0)	D*(i+I)+ (j+J)		D*(i+I)+ (j+J+1)		...	D*{i-(C-I)}+ (j+J)		D*(i-(C-I))+ (j+J+1)	
1										
2	{j<(D-J)} &(j%2=1)	D*(i+I)+ (j+J)		D*(i+I)+ (j+J-1)			D*(i-(C-I))+ (j+J)		D*(i-(C-I))+ (j+J-1)	
3										
...		i < (C-I-1) &(i % 2=0)		{i < (C-I-1)} &(i % 2=1)			i >= (C-I-1) &(i % 2=0)		i >= (C-I-1) &(i % 2=1)	
D-4	{j>=(D-J)} &(j%2=0)	D*(i+I+1)+ {j-(D-J)}		D*(i+I+1)+ {j-(D-J-1)}		...	D*{i-(C-I-1)}+ {j-(D-J)}		D*{i-(C-I-1)}+ {j-(D-J-1)}	
D-3										
D-2	{j>=(D-J)} &(j%2=1)	D*(i+I+1)+ (j-(D-J))		D*(i+I+1)+ {j-(D-J+1)}			D*{i-(C-I-1)}+ {j-(D-J)}		D*{i-(C-I-1)}+ {j-(D-J+1)}	
D-1										

Table 6.3(c) Proposed algorithm for N_{bpscs}=6 (64-QAM) with all N, i_{ss} and BW

Column no. (i) →		0 1	2	3	…	C-4	C-3 C-2	C-1
↓ Row no. (j)		{i<(C-I)} &(i%3=0)	{i<(C-I)} &(i%3=1)	{i<(C-I)} &(i%3=2)		i>=(C-I) &(i%3=0)	i>=(C-I) &(i%3=1)	i>=(C-I) &(i%3=2)
0 1	{j<(D-J)} &(j%3=0)	D*(i+I)+ (j+J)	D*(i+I)+ (j+J+2)	D*(i+I)+ (j+J+1)		D*{i-(C-I)}+ (j+J)	D*{i-(C-I)}+ (j+J+2)	D*{i-(C-I)}+ (j+J+1)
2	{j<(D-J)} &(j%3=1)	D*(i+I)+ (j+J)	D*(i+I)+ (j+J-1)	D*(i+I)+ (j+J+1)	…	D*{i-(C-I)}+ (j+J)	D*{i-(C-I)}+ (j+J-1)	D*{i-(C-I)}+ (j+J+1)
3	{j<(D-J)} &(j%3=2)	D*(i+I)+ (j+J)	D*(i+I)+ (j+J-1)	D*(i+I)+ (j+J-2)		D*{i-(C-I)}+ (j+J)	D*{i-(C-I)}+ (j+J-1)	D*{i-(C-I)}+ (j+J-2)
…		i<(C-I-1) &(i%3=0)	{i<(C-I-1)} &(i%3=1)	{i<(C-I-1)} &(i%3=2)		i>=(C-I-1) &(i%3=0)	i>=(C-I-1) &(i%3=1)	i>=(C-I-1) &(i%3=2)
D-4 D-3	{j>=(D-J)} &(j%3=0)	D*(i+I+1) + {j-(D-J)}	D*(i+I+1)+ {j-(D-J-2)}	D*(i+I+1)+ {j-(D-J-1)}		D*{i-(C-I-1)}+ {j-(D-J)}	D*(i-(C-I-1))+ (j-(D-J-2))	D*(i-(C-I-1))+ (j-(D-J-1))
D-2	{j>=(D-J)} &(j%3=1)	D*(i+I+1) + (j-(D-J))	D*(i+I+1)+ (j-(D-J+1))	D*(i+I+1)+ (j-(D-J-1))	…	D*(i-(C-I-1)+ (j-(D-J))	D*(i-(C-I- 1))+(j-(D-J+1))	D*(i-(C-I- 1))+(j-(D-J-1))
D-1	{j>=(D-J)} &(j%3=2)	D*(i+I+1) +(j-(D-J))	D*(i+I+1)+ (j-(D-J+1))	D*(i+I+1)+ (j-(D-J+2))		D*(i-(C-I-1))+ (j-(D-J))	D*(i-(C-I-1))+ (j-(D-J+1))	D*(i-(C-I-1))+ (j-(D-J+2))

The mathematical formulation of the proposed algorithms in Table 6.3(a)-(c) including all modulation schemes, spatial streams and BWs are represented by (6.2)-(6.4).

$$k_{n(QPSK\text{-}BPSK)} = \begin{cases} D*(i+I)+(j+J) & \text{when } j < (D\text{-}J) \text{ and } i < (C\text{-}I) \\ D*\{i\text{-}(C\text{-}I)\}+(j+J) & \text{when } j < (D\text{-}J) \text{ and } i \geq (C\text{-}I) \\ D*(i+I+1)+\{j\text{-}(D\text{-}J)\} & \text{when } j \geq (D\text{-}J) \text{ and } i < (C\text{-}I\text{-}1) \\ D*\{i\text{-}(C\text{-}I\text{-}1)\}+\{j\text{-}(D\text{-}J)\} & \text{when } j \geq (D\text{-}J) \text{ and } i \geq (C\text{-}I\text{-}1) \end{cases} \tag{6.2}$$

$$k_{n(16\text{-}QAM)} = \begin{cases} D*(i+I)+(j+J) & \text{when } \{j<(D\text{-}J)\}\&[\{i<(C\text{-}I)\}\&(i\%2=0)] \\ D*(i+I)+(j+J+1) & \text{when } [\{j<(D\text{-}J)\}\&(j\%2=0)]\&[\{i<(C\text{-}I)\}\&(i\%2=1)] \\ D*\{i\text{-}(C\text{-}I)\}+(j+J) & \textit{when } \{j<(D\text{-}J)\}\& [\{i \geq(C\text{-}I)\}\&(i\%2=0)] \\ D*(i\text{-}(C\text{-}I))+(j+J+1) & \text{when } [\{j<(D\text{-}J)\}\&(j\%2=0)]\& [\{i\geq (C\text{-}I)\}\&(i\%2=1)] \\ D*(i+I)+(j+J\text{-}1) & \text{when } [\{j<(D\text{-}J)\}\&(j\%2=1)]\&[\{i<(C\text{-}I)\}\&(i \%2=1)] \\ D*(i\text{-}(C\text{-}I))+(j+J\text{-}1) & \text{when } [\{j<(D\text{-}J)\}\&(j\%2=1)] \& [\{i\geq(C\text{-}I)\}\&(i\%2=1)] \\ D*(i+I+1)+\{j\text{-}(D\text{-}J)\} & \text{when } \{j\geq (D\text{-}J)\} \& [\{i<(C\text{-}I\text{-}1)\}\&(i\%2=0)] \\ D*(i+I+1)+\{j\text{-}(D\text{-}J\text{-}1)\} & \text{when } [\{j\geq(D\text{-}J)\}\&(j\%2=0)] \& [\{i<(C\text{-}I\text{-}1)\}\&(i\%2=1)] \\ D*\{i\text{-}(C\text{-}I\text{-}1)\}+\{j\text{-}(D\text{-}J)\} & \text{when } \{j>=(D\text{-}J)\} \& [\{i\geq(C\text{-}I\text{-}1)\}\&(i\%2=0)] \\ D*\{i\text{-}(C\text{-}I\text{-}1)\}+\{j\text{-}(D\text{-}J\text{-}1)\} & \text{when } [\{j\geq(D\text{-}J)\}\&(j\%2=0)]\& [\{i\geq(C\text{-}I\text{-}1)\}\&(i\%2=1)] \\ D*(i+I+1)+\{j\text{-}(D\text{-}J+1)\} & \text{when } [\{j\geq(D\text{-}J)\} \&(j\%2=1)] \& [\{i<(C\text{-}I\text{-}1)\}\&(i\%2=1)] \\ D*\{i\text{-}(C\text{-}I\text{-}1)\}+\{j\text{-}(D\text{-}J+1)\} & \text{when } [\{j\geq(D\text{-}J)\}\&(j\%2=1)] \& [\{i\geq(C\text{-}I\text{-}1)\}\&(i\%2=1)] \end{cases} \tag{6.3}$$

$$k_{n(64\text{-}QAM)}=\begin{cases}
D*(i+I)+(j+J) & \text{when } \{j<(D-J)\}\&[\{i<(C-I)\}\&(i\%3=0)] \\
D*(i+I)+(j+2) & \text{when } [\{j<(D-J)\}\&(j\%3=0)]\&[\{i<(C-I)\}\&(i\%3=1)] \\
D*(i+I)+(j+J+1) & \text{when } [\{j<(D-J)\}\&(j\%3\neq2)]\&[\{i<(C-I)\}\&(i\%3=2)] \\
D*\{i-(C-I)\}+(j+J) & \text{when } \{j<(D-J)\}\&[\{i\geq(C-I)\}\&(i\%3=0)] \\
D*\{i-(C-I)\}+(j+J+2) & \text{when } [\{j<(D-J)\}\&(j\%3=0)]\&[\{i\geq(C-I)\}\&(i\%3=1)] \\
D*\{i-(C-I)\}+(j+J+1) & \text{when } [\{j<(D-J)\}\&(j\%3\neq2)]\&[\{i\geq(C-I)\}\&(i\%3=2)] \\
D*(i+I)+(j+J-1) & \text{when } [\{j<(D-J)\}\&(j\%3\neq0)]\&[\{i<(C-I)\}\&(i\%3=1)] \\
D*\{i-(C-I)\}+(j+J-1) & \text{when } [\{j<(D-J)\}\&(j\%3\neq0)]\&[\{i\geq(C-I)\}\&(i\%3=1)] \\
D*(i+I)+(j+J-2) & \text{when } [\{j<(D-J)\}\&(j\%3=2)]\&[\{i<(C-I)\}\&(i\%3=2)] \\
D*\{i-(C-I)\}+(j+J-2) & \text{when } [\{j<(D-J)\}\&(j\%3=2)]\&[\{i\geq(C-I)\}\&(i\%3=2)] \\
D*(i+I+1)+\{j-(D-J)\} & \textit{when } \{j\geq(D-J)\}\&[\{i<(C-I-1)\}\&(i\%3=0)] \\
D*(i+I+1)+\{j-(D-J-2)\} & \text{when } [\{j\geq(D-J)\}\&(j\%3=0)]\&[\{i<(C-I-1)\}\&(i\%3=1)] \\
D*(i+I+1)+\{j-(D-J-1)\} & \text{when } [\{j\geq(D-J)\}\&(j\%3\neq2)]\&[\{i<(C-I-1)\}\&(i\%3=2)] \\
D*\{i-(C-I-1)\}+\{j-(D-J)\} & \text{when } \{j\geq(D-J)\}\&[\{i\geq(C-I-1)\}\&(i\%3=0)] \\
D*\{i-(C-I-1)\}+\{j-(D-J-\ )\} & \text{when } [\{j\geq(D-J)\}\&(j\%3=0)]\&[\{i\geq(C-I-1)\}\&(i\%3=1)] \\
D*\{i-(C-I-1)\}+\{j-(D-J-1)\} & \text{when } [\{j\geq(D-J)\}\&(j\%3\neq2)]\&[\{i\geq(C-I-1)\}\&(i\%3=2)] \\
D*(i+I+1)+\{j-(D-J+1)\} & \text{when } [\{j\geq(D-J)\}\&(j\%3\neq0)]\&[\{i<(C-I-1)\}\&(i\%3=1)] \\
D*\{i-(C-I-1)\}+\{j-(D-J+1)\} & \text{when } [\{j\geq(D-J)\}\&(j\%3\neq0)]\&[\{i\geq(C-I-1)\}\&(i\%3=1)] \\
D*(i+I+1)+\{j-(D-J+2)\} & \text{when } [\{j\geq(D-J)\}\&(j\%3=2)]\&[\{i<(C-I-1)\}\&(i\%3=2)] \\
D*\{i-(C-I-1)\}+\{j-(D-J+2)\} & \text{when } [\{j\geq(D-J)\}\&(j\%3=2)]\&[\{i\geq(C-I-1)\}\&(i\%3=2)]
\end{cases} \tag{6.4}$$

The general validity of the proposed mathematical formulation could be established with the help of [105]. As far as spatial permutation is concerned, the steps involved in IEEE 802.16e [105] and in IEEE 802.11n [75] are identical. Additionally, the spatial streams of the latter undergoes frequency rotation using B_3 of Fig. 6.2, except the first stream. Further, analysis of the 3rd step results that the entire term beyond j_k (i.e. J_{rot}) remains constant for a particular spatial stream and can be expressed as [121]

$$r_k = [j_k - J_{rot}]\%N \tag{6.1}$$

$$\text{where } J_{rot} = \left[\{(i_{ss}-1)*2\}\%3 + 3\left\lfloor\frac{i_{ss}-1}{3}\right\rfloor\right] * N_{rot} * N_{BPSCS}$$

As the first stream for all modulation schemes undergoes no frequency rotation, hence

$$r_k = [j_k - 0]\%N = [j_k]\%N = j_k$$

For subsequent streams, the value of J_{rot} as shown in Table 6.4, differs for each spatial streams, modulation schemes and BWs. The expression of j_k so derived for all modulation schemes in [105] if substituted in (6.1) gives three new equations. The final expressions so obtained and the proposed mathematical formulations developed in this work, generate the same results which are identical with results obtained through direct implementation of B_1-B_3 steps.

Table 6.4 Values of J_{rot} for all modulation schemes, spatial streams and BWs

Modulation Scheme (N_{cbpsc})	BW=20MHz				BW=40MHz			
	I_{ss}=1	I_{ss}=2	I_{ss}=3	I_{ss}=4	I_{ss}=1	I_{ss}=2	I_{ss}=3	I_{ss}=4
BPSK (N_{cbpsc}=1)	0	26	13	39	0	58	29	87
QPSK (N_{cbpsc}=2)	0	52	26	78	0	116	58	174
16-QAM (N_{cbpsc}=4)	0	104	52	156	0	232	116	348
64-QAM (N_{cbpsc}=6)	0	156	78	234	0	348	174	522

6.5 Transformation into Hardware

This section describes the transformation of the proposed algorithm into digital hardware for the address generator of block interleaver in connection with IEEE 802.11n based WLAN. The top level view of the complete interleaver consisting of proposed address generator and memory block is shown in Fig. 6.3(a).

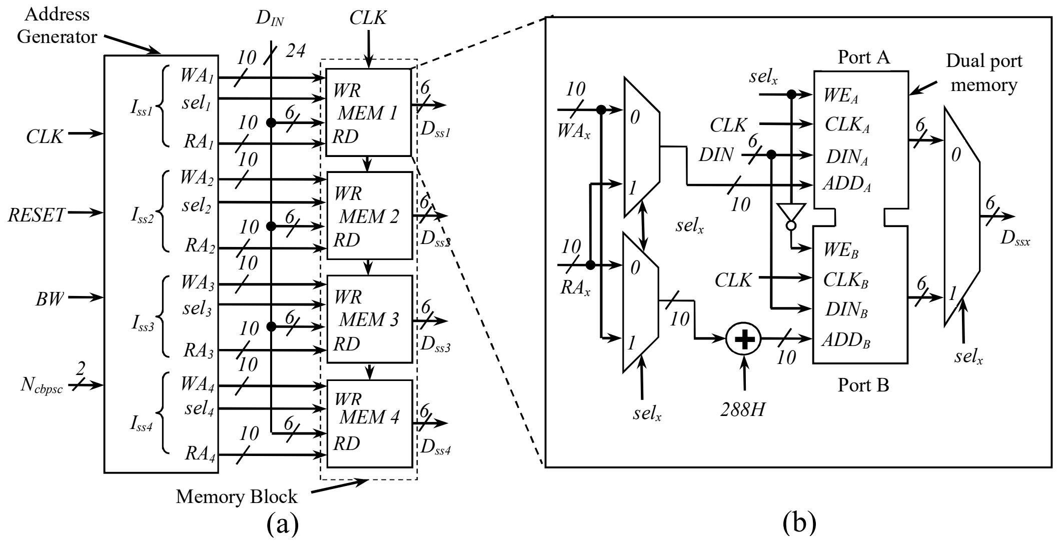


Fig. 6.3 a) Top level view of complete interleaver b) arrangement of memory block

6.5.1 Memory Block

The detailed arrangement of the memory block for one spatial stream having similar structure as in [157] is shown in Fig. 6.3(b). The structure is generic and is applicable to all spatial streams. It receives three inputs from the address generator block; write address (WA_x), read address (RA_x) and sel_x. The requirement of two memory blocks for block interleaving is accomplished with the help of a dual port memory (with Port A and B) where read and write operation can be performed simultaneously. As a result, the interleaver memory block design is lesser complex than [111]. As seen in Fig. 6.3(b), the first 288H locations are used as Port A and next 288H locations as Port B. An adder is used to insert the bias of 288H while generating addresses for Port B. When one port is being written, other one is read and vice versa. Swapping between read/write operations at the end of a cycle is performed using the signal sel_x which is generated using a toggle flip-flop.

6.5.2 Address Generator

The address generator is the heart of the interleaver. The encoding schemes used in this work for the two inputs, BW and N_{cbpsc} of the address generator are described in Table 6.5. The i_{ss1}-i_{ss4} represent the four different spatial streams each consisting of write (WA_x), read (RA_x) addresses and select signal (sel_x) output. As shown in Fig. 6.4(a), a multiplexer is used in the write address generator to route the desired WA_x from four possible sources based on the value of N_{cbpsc} for a particular spatial stream, I_{ssx}.

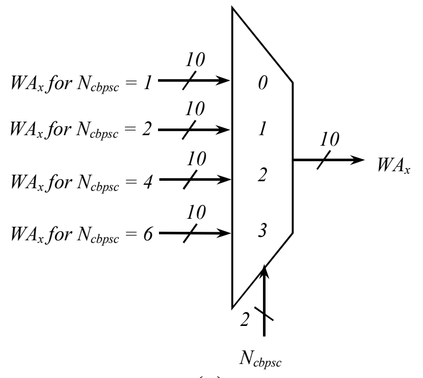
Fig. 6.4(b) and (c) show the hardware used for generation of row count (*JCOUNT*) and column count (*ICOUNT*) respectively using up-counters and comparators. Circuit arrangement for generation of row number, D using BW and N_{cbpsc} is shown in Fig. 6.5(a). Similarly, Fig. 6.5(b) and (c) describe hardware used for generation of $ICOUNT<(C-I_x)$, $ICOUNT\geq(C-I_x)$, $JCOUNT<(D-J_x)$ and $JCOUNT\geq(D-J_x)$ signals. Here I_x and J_y is the column and row offset value respectively, used while computing the addresses and is defined in Table 6.6.

Table 6.5(a) Encoding of BW

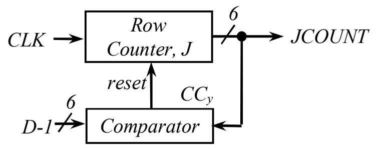
Bandwidth (BW)	Encoded bit
20MHz	0
40MHz	1

Table 6.5(b) Encoding of N_{cbpsc}

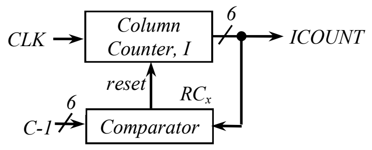
Modulation Scheme (N_{cbpsc})	Encoded bits
BPSK (N_{cbpsc}=1)	00
QPSK (N_{cbpsc}=2)	01
16-QAM (N_{cbpsc}=4)	10
64-QAM (N_{cbpsc}=6)	11



(a)

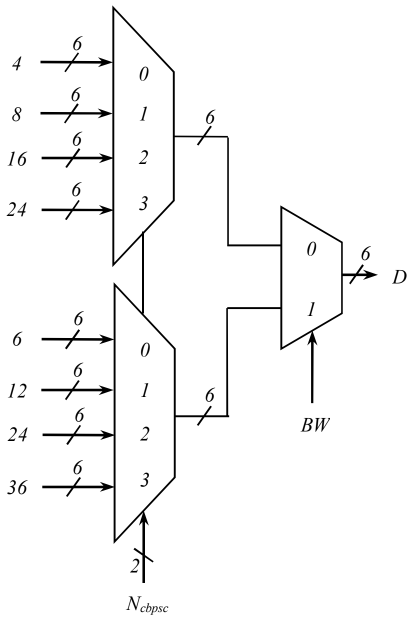


(b)

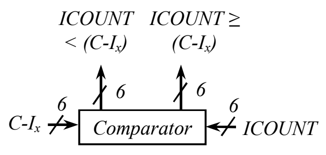


(c)

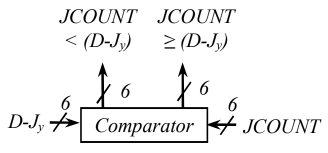
Fig. 6.4 Scheme showing generation of (a) write address (b) row count and (c) column count



(a)



(b)



(c)

Fig. 6.5 Arrangement showing generation of (a) number of rows, (b) ICOUNT<(C-I$_x$) and ICOUNT≥(C-I$_x$) (c) JCOUNT<(D-J$_y$) and JCOUNT≥(D-J$_y$)

Table 6.6 Definition of I_x and J_y for all streams and *BW*

Stream	*BW=20MHz, C=13*	*BW=40MHz, C=18*
I_{ss1}	I_1=0, J_1=0	I_1=0, J_1=0
I_{ss2}	I_2=6, J_2=NBPSC*2	I_2=8, J_2=NBPSC*2
I_{ss3}	I_3=9, J_3=NBPSC*3	I_3=13, J_3=NBPSC
I_{ss4}	I_4=3, J_4=NBPSC	I_4=3, J_4=NBPSC*3

Hardware required for the generation of read addresses (RA_x) is shown in Fig. 6.6. Like the write address generator, the structure developed for generation of RA_x is also generic and is applicable to all the spatial streams. The first and second level multiplexers select one of the values of interleaver depth from the inputs with the help of *BW* and *mod_typ* signal. The *rd_count* is a 10-bit up counter and generates RA_x. While progressing through the count values, when the *rd_count* value equals the output of M$_1$, a *reset* pulse is generated by the comparator and *rd_count* goes to initial state to start another cycle.

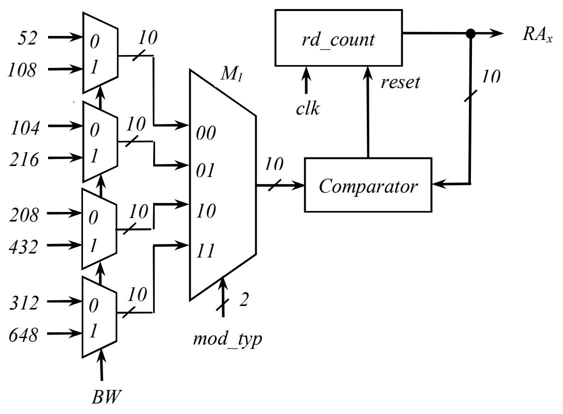


Fig. 6.6 Circuit for generation of read address (RA$_x$)

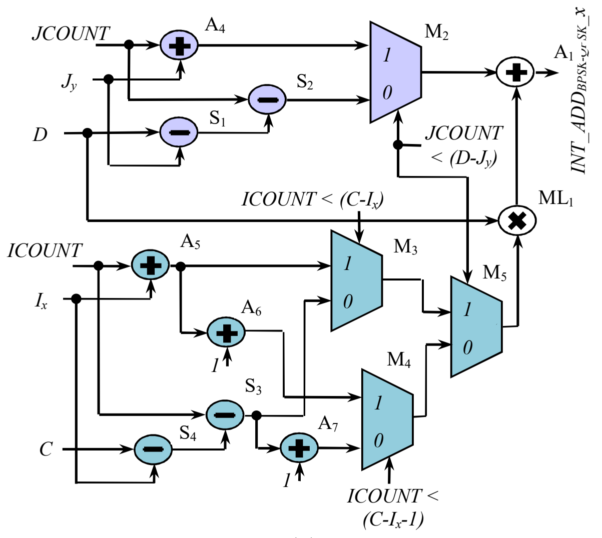
Fig. 6.7(a) and (b) show the rest of the circuit details required to generate interleaver write addresses with BPSK/QPSK, 16-QAM and 64-QAM modulation schemes. In these figures, the adders (A_1-A_3) receive two inputs; one from the row count part (purple coloured) and the other from the column count part (blue coloured) of the circuit. In Fig. 6.7(a), the *JCOUNT+J_y* signal is generated by adder (A_4) whereas the two subtractors (S_1 and S_2) generate the signal *JCOUNT-(D-J_y)*. Based on the value of *JCOUNT<(D-J_y)* signal, the multiplexer (M_2) routes one of these signals to the input of the A_1. Similar hardware structure can be found for generation of signals like *ICOUNT+I_x*, *ICOUNT+I_x+1*, *ICOUNT–(C-I_x)* etc. in the column count part. The output of the column count part gets multiplied with *D* in the multiplier (ML_1) to generate the second input of A_1. In Fig. 6.7(b) and (c), the circuit details for generation of signals like *ICOUNT+I_x*, *ICOUNT–(C-I_x)*, *JCOUNT+J_y*, *JCOUNT-(D-J_y)* etc. are not shown to avoid repetition and clumsiness. The condition for generation of select inputs (*II4*, *JJ4*, *II6* and *JJ6*) for the multiplexers of Fig. 6.7(b) and (c) are described and encoded in Table 6.7(a) and (b).

Table 6.7(a) Encryption of signals *II4* and *JJ4*

Condition	*II4*	Condition	*JJ4*
ICOUNT<(C-I_x) and iXMOD = 0	00	*JCOUNT<(D-J_y) and jXMOD=0*	00
ICOUNT<(C-I_x) and iXMOD = 1	01	*JCOUNT<(D-J_y) and jXMOD=1*	01
ICOUNT≥(C-I_x) and iXMOD = 0	10	*JCOUNT≥(D-J_y) and jXMOD=0*	10
ICOUNT≥(C-I_x) and iXMOD = 1	11	*JCOUNT≥(D-J_y) and jXMOD=1*	11

Table 6.7(b) Encryption of signals *II6* and *JJ6*

Condition	*II6*	Condition	*JJ6*
ICOUNT<(C-I_x) and iXMOD=0	000	*JCOUNT<(D-J_y) and jXMOD=0*	000
ICOUNT<(C-I_x) and iXMOD=1	001	*JCOUNT<(D-J_y) and jXMOD=1*	001
ICOUNT<(C-I_x) and iXMOD=2	010	*JCOUNT<(D-J_y) and jXMOD=2*	010
ICOUNT≥(C-I_x) and iXMOD=0	011	*JCOUNT≥(D-J_y) and jXMOD=0*	011
ICOUNT≥(C-I_x) and iXMOD=1	100	*JCOUNT≥(D-J_y) and jXMOD=1*	100
ICOUNT≥(C-I_x) and iXMOD=2	101	*JCOUNT≥(D-J_y) and jXMOD=2*	101


JCOUNT
J_y
D
A_4
S_2
S_1
M_2
1
0
A_1
$INT\_ADD_{BPSK\text{-}QPSK\_x}$
JCOUNT < (D-J_y)
ICOUNT < (C-I_x)
ML_1
ICOUNT
I_x
A_5
M_3
M_5
A_6
1
M_4
S_3
S_4
A_7
C
ICOUNT < (C-I_x-1)

(a)

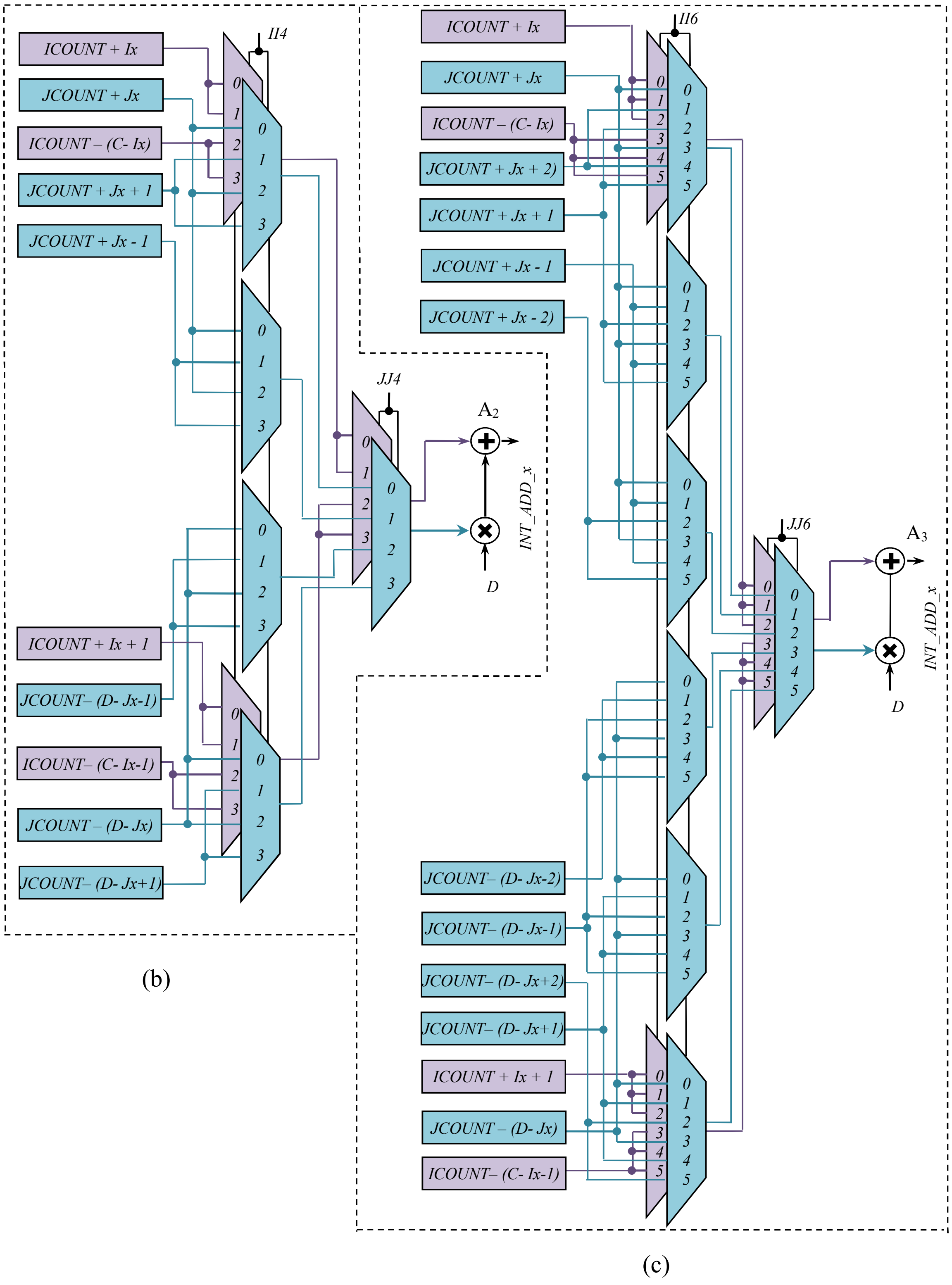
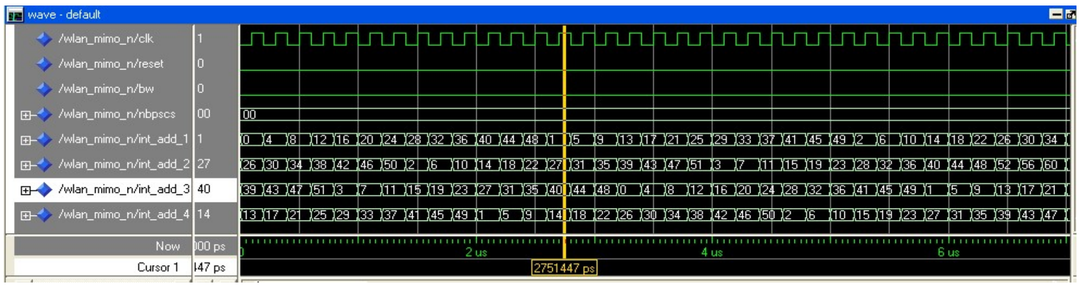


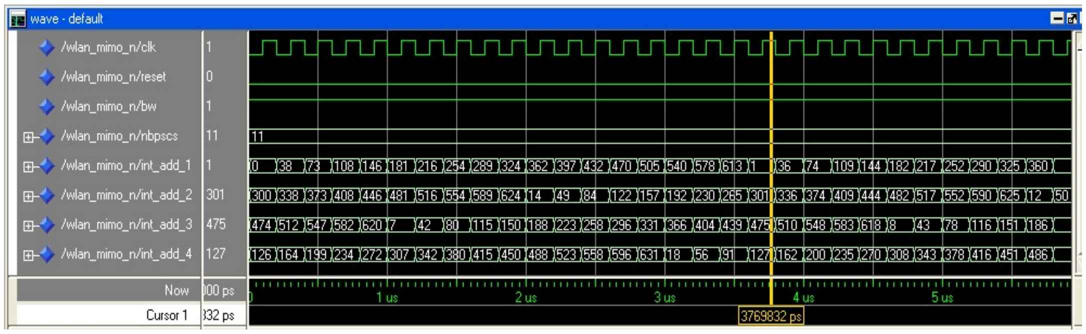
Fig. 6.7 Circuit diagram for the generation of interleaver write addresses with (a) N_{cbpsc}=1 or 2 (b) N_{cbpsc}=4 and (c) N_{cbpsc}=6

6.6 Simulation Results of MIMO WLAN Interleaver

This section describes generation of simulation result in the form of timing diagram containing the desired write address sequences of our proposed interleaver address generator. The timing simulation so obtained using ModelSim XE-III software, enables the author to verify the working of the proposed interleaver address generator with the standard document of IEEE 802.11n [75]. The address generation circuitry is tested for all BWs, spatial streams and modulation schemes, out of which one such result with $N_{bpscs} = 1$ (BPSK), $N = 52$, $BW = 20$MHz ($nbpscs = 00_2$, $bw = 0_2$,) having all four spatial streams is presented in Fig. 6.8(a). The first four signals i.e. *clk*, *reset*, *bw* and *nbpscs* are input to the address generator. All the operations of the circuit are synchronized with respect to the *clk* signal. The last four signals (*int_add_1* to *int_add_4*) of Fig. 6.8(a) are the output of the address generator displaying the sequence of write addresses generated for the four different spatial streams (i_{ss1}-i_{ss4}) of the interleaver. Among these, the bottom most signal (*int_add_4*) generates the address sequence with values 13, 17, 21,…, 49, 1, 5, 9, 14, 18, 22, …, 50, 2, 6, … which is identical with the address sequence of Table 6.2(a). This verifies the working of our proposed interleaver as per the standard [75]. Another timing simulation with $N_{bpscs} = 6$ (16-QAM), $N = 648$, $BW = 40$MHz ($nbpscs = 11_2$, $bw = 1_2$) has been presented in Fig. 6.8(b). In this case too, the address sequences displayed at the bottom four signals exactly match with the output of the MATLAB program with input parameters $N_{bpscs} = 6$ (16-QAM), $N = 648$, $BW = 40$MHz. Such verifications have been carried out exclusively with address sequences for all possible combination of the MIMO WLAN interleaver specification as per Table 6.1. However, comparison of these simulation results with other works could not be made as timing simulations have not been provided by the others.



(a)


wave - default
/wlan_mimo_n/clk 1
/wlan_mimo_n/reset 0
/wlan_mimo_n/bw 1
/wlan_mimo_n/nbpscs 11 11
/wlan_mimo_n/int_add_1 1 0 38 73 108 146 181 216 254 289 324 362 397 432 470 505 540 578 613 1 36 74 109 144 182 217 252 290 325 360
/wlan_mimo_n/int_add_2 301 300 338 373 408 446 481 516 554 589 624 14 49 84 122 157 192 230 265 301 336 374 409 444 482 517 552 590 625 12 50
/wlan_mimo_n/int_add_3 475 474 512 547 582 620 7 42 80 115 150 188 223 258 296 331 366 404 439 475 510 548 583 618 8 43 78 116 151 186
/wlan_mimo_n/int_add_4 127 126 164 199 234 272 307 342 380 415 450 488 523 558 596 631 18 56 91 127 162 200 235 270 308 343 378 416 451 486
Now 000 ps
Cursor 1 332 ps
1 us
2 us
3 us
4 us
5 us
3769832 ps

(b)

Fig. 6.8 Write addresses (WA_x) of interleaver for (a) N_{bpscs}=1 (BPSK), N=52, BW=20MHz (nbpscs=00_2, bw=0_2) and (b) N_{bpscs}=6 (64-QAM), N=648, BW=40MHz (nbpscs=11_2, bw=1_2)

6.7 FPGA Implementation Results

The proposed design of the interleaver is transformed into a VHDL model using Xilinx ISE 12.1 and is implemented on Xilinx Spartan-6 FPGA. Table 6.8 shows the minimum hardware requirement for the implementation of the proposed design obtained by HDL synthesis irrespective of implementation platform e.g. FPGA or ASIC. The two ROMs of sizes 64 x 2 bit are used to store initial value of I_x and J_y as per Table 6.6. The 10-bit adders are used at the final stage of the address generator one at each bit stream for addition of row count value with the column count value. For the computation of signals like $ICOUNT+I_x$, $ICOUNT+I_x+1$, $ICOUNT-(C-I_x)$ etc, the design uses 6-bit adder, 6-bit subtractor with borrow input & 6-bit subtractor without borrow input circuits. The design models two 6-bit counters for the implementations of Fig. 6.4(b) and (c). Internal storage requirement is met up by the 1-bit and 6-bit latches as described in Table 6.8. In order to implement the less than and greater than equal to condition as listed in Table 6.7 (a) and (b), the design uses the 6-bit less and great-equal circuits. The proposed hardware structure of the address generator requires large number of multiplexers of different widths. To implement them on FPGA platform, the design requires the multiplexers as mentioned in Table 6.8.

In spite of our exhaustive literature survey, similar implementations on FPGA platform have not yet been noticed for the purpose of comparison. As a result, the conventional LUT based approach [157], [103], [165] has been redesigned and implemented for MIMO WLAN interleaver on the same FPGA platform utilizing BRAM to house the address LUTs for the sake of comparison only. Four dual port BRAM memory blocks have been used to implement the interleaver memory in both the designs in a manner similar to [115]. Comparative analysis of the two implementations in terms of device utilization is made in Table 6.9 wherein betterment of the proposed novel technique can be quantified in terms of embedded memory utilization (88.9% memory saving) and operating speed (27.43% speed improvement) with approximately 4% overuse of slice LUTs. Such marginal overuse occurs as the logic circuit associated with LUT based approach is relatively simpler [157] than our proposed technique. As modern FPGAs like Spartan 6 contain abundance of such slice LUTs, minor overuse does not affect the design in comparison with the use of critical and limited resources like BRAM. Significant reduction in BRAM use by our proposed design, enables the designer to meet other memory requirement while implementing the complete MIMO WLAN transceiver on the same FPGA. Use of DSP blocks as multiplier improves the performance of the circuit by reducing the delay. The circuit works at maximum clock speed of 208.7MHz with 28.62mW of power consumption. As the design has four parallel spatial streams, the throughput of the proposed interleaver may reach upto 834.8Mbps on Spartan 6 FPGA thereby capable of delivering 28.14% higher throughput than the maximum requirement [75].

Table 6.8 Minimum hardware requirement for the interleaver

Logic Circuits Used	Quantity	Logic Circuits Used	Quantity
64x2-bit ROM	2	1-bit latch	26
10-bit adder	4	6-bit comparator great	12
6-bit adder	22	6-bit comparator less equal	1
6-bit sub borrow in	4	1-bit 2-to-1 multiplexer	659
6-bit subtractor	50	6-bit 4-to-1 multiplexer	178
6-bit up counter	2		

Table 6.9 Device Utilization Summary

FPGA Resources	This work		LUT Based technique [157], [103], [165]	
	Utilization in Number	Utilization in %	Utilization in Number	Utilization in %
Number of Slices Registers	30 out of 30064	0.10	35 out of 30064	0.12
Number of Slices LUTs	864 out of 15032	5.75	201 out of 15032	1.34
Number of BRAMs	4 out of 52	7.69	36 out of 52	69.23
Number of DSP48A1s	4 out of 38	10.53	0 out of 38	0 %
Number of BUFG/BUFGCTRLs	2 out of 16	12.50	2 out of 16	12.50

In addition, comparison with few works has been done based on the equivalence drawn between FPGA and ASIC implementations in [156]. The comparative study of the proposed implementation in respect of key FPGA parameters shows betterment over other similar recent works and is presented in Table 6.10. The proposed circuit shows betterment over [121], [120], [122] and LUT based technique in terms of maximum operating frequency. Our implementation has been found to be the most efficient among the designs in references [121], [120], [122] of Table 6.10 in terms of power consumption. Similar comparison has been drawn by Zafar *et al.* [111] with previous work [110] in terms of resource requirement while implementing interleaver for 2 x 2 MIMO WiMAX system. However, direct comparison between our proposed and work in [110], [111] may not be possible, as the interleaver specification for both standards are not identical, especially the former involves a third step of permutation called frequency rotation.

Table 6.10 Comparative study between similar works

FPGA Parameters	This work	[120]	[121]	[122]	LUT Based [157], [103], [165]
Maximum clock frequency	208.7 MHz	109.38MHz (Improvement over [120]: 47.59%)	70.31MHz (Improvement over [121]: 66.31%)	125MHz (Improvement over [122]: 40.1%)	151.45MHz (Improvement over LUT method: 27.43%)
Power consumption	28.62mW	111.24mW (Reduction over [120]: 74.27%)	48mW (Reduction over [121]: 40.38%)	Not available	28.62mW (at par with LUT based method)

6.8 Discussion

This chapter demonstrates the design and implementation of novel interleaver hardware on FPGA platform to be used in OFDM based MIMO WLAN applications. New algorithm has been proposed for the address generator of the interleaver eliminating the requirement of floor function and is supported by mathematical foundation with general validity. The algorithm is transformed into digital circuit and is modeled using VHDL software. Simulation results verify the functionality of the proposed algorithm. Hardware implementation of the VHDL model using Xilinx ISE has also been done as well as tested on Xilinx Spartan 6 FPGA. Efficient design and use of FPGA's embedded resources during the implementation enables betterment over few recent similar works and conventional design in terms of multiple FPGA parameters. This work motivates the author to design the QPP interleaver used in latest wireless broadband technology-LTE/LTE-A and is presented in the next chapter.

Chapter 7
Implementation of QPP interleaver

Outline of this Chapter

After detailed discussion on the design and implementation of resource efficient and high speed interleavers / de-interleavers in the previous chapters, this chapter incorporates QPP interleaver implementation on Xilinx Spartan 6 FPGA. The address generator of the interleaver contains a quadratic expression having square and modulus function whose direct digital hardware is not yet available in the literature. A novel algorithm has now been proposed which can provide low complexity hardware solution to implement the interleaver address generator. This chapter describes VHDL model and timing simulation of the proposed address generator using ModelSim XE-III software. Due to absence of implementation results in the literature, comparison of this work is made by implementing conventional LUT based technique on the same FPGA. Such comparison shows better FPGA resource utilization and improved operating speed in favour of the novel proposed technique.

7.1. Introduction

The demand for ubiquitous mobile internet services requires high bandwidth connectivity. To cater this demand, new technologies like the LTE of 3rd Generation Partnership Project (3GPP) standards [76] have been developed. LTE is rapidly becoming the dominant global standard for fourth generation cellular networks [56]. It has brought together many technological innovations from different areas of research such as digital signal processing, internet protocols, network architecture and security, and is also poised to dramatically change the way we use the world wide mobile network in future. LTE-A [77] is the project name of the evolved version of LTE that is being developed by 3GPP [166]. LTE-A will meet or exceed the requirements of the International Telecommunication Union (ITU) for the fourth generation (4G) radio communication standard known as International Mobile Telecommunication (IMT)-Advanced [167].

LTE / LTE-A uses Turbo coding as channel coding scheme. Turbo encoder and decoder are one of the major blocks in a LTE wireless transceiver. Turbo encoder/decoder employs interleaver to reduce the effect of burst error in the channel. Turbo decoders provide best performance but suffer from high decoding latency due to the iterative decoding process [168]. This is due to the forward–backward recursion in the maximum a posteriori (MAP) decoding algorithm and the interleaving /de- interleaving between the iterations [124].

The QPP interleaver is defined in the new 3GPP LTE standard. The function of the QPP interleaver is to take a block of N-bit data and produce a permutation of the input data block. From the coding theory perspective, the performance of a Turbo code depends critically on the interleaver structure. The structure of the QPP interleaver differs from previous 3G interleavers in sense that it is based on algebraic constructions via permutation polynomials over integer rings. It is known that permutation polynomials generate contention-free interleavers.

In this Chapter, we propose an efficient algorithm to model the interleaver address generator used in LTE/LTE-A. The address generator of QPP interleaver involves a quadratic expression having square and modulus function whose corresponding digital hardware is not available. In our approach, the said expression is divided into two parts. The first part generates raw addresses which is described by a novel algorithm. The second part computes modulus on these raw addresses and employs a modified technique over [169]. Both algorithms are transformed into digital circuits for efficient FPGA implementation. VHDL model has been prepared to implement the hardware in Xilinx Spartan 6 FPGA. Functionality of the interleaver address generator is verified through timing simulation using ModelSim XE-III software. In spite of best possible effort in literature survey, the author could not find similar works implemented on FPGA platform having implementation results. Consequently, comparison could not be drawn with existing works implemented on CMOS and other platforms. As a result, for the sake of comparison, the author implemented the conventional LUT based technique with improved memory modeling on the same FPGA. Comparative analysis of our proposed work with improved LUT based technique in terms FPGA resource utilization (Block RAM) and operating speed shows betterment by 71.16% and 82.26% respectively at the negligible cost of FPGA slice requirement.

7.2. Interleaving in LTE/LTE-A

Turbo coding is employed in many standards for forward error correction techniques due to its impressive performance [167]. The interleavers are used in turbo coding / decoding to improve the error performance. In order to increase the throughput, parallel MAP decoding technique is adopted in Turbo code decoder of LTE/LTE-A transceiver [170]. As a result, more than one MAP processor may store their outputs in the same memory block of the interleaver simultaneously, if the interleaver is not designed properly. In that case a contention of memory access [171] occurs and additional circuit with decoding latency will be required to resolve the contention. The 3GPP LTE and LTE-A standards incorporate the use of the QPP interleaver [172]. Such interleaver poses contention free property and allows parallel decoders to decode one codeword with improved throughput [173].

The interleaving operation in a QPP interleaver defined for LTE/LTE-A may be expressed as

$$\Pi(i) = (f_1 * i + f_2 * i^2) \bmod K \tag{7.1}$$

where i stands for the original address, and $\Pi(x)$ is the interleaved address. The parameters f_1 and f_2 is related to the block size K and is defined in [76]. In the 3GPP LTE/LTE-A standard, there are 188 different block sizes ranging from 40 to 6144, and each size has its different interleaver parameters f_1 and f_2.

7.3. Proposed Algorithm for QPP interleaver

This section describes the formulation of the proposed algorithm for the address generator of interleaver used in LTE/LTE-A. Firstly, a MATLAB program has been developed using (7.1) to determine sequence of addresses to be generated against each value of K, f_1 and f_2 parameters. Table 7.1 shows certain portion of such address sequences obtained for three instances with (a) $K = 40$, (b) $K = 1008$ and (c) $K = 6144$.

Table 7.1(a) Address Sequences with $K = 40, f_1=3, f_2=10$

13	6	19	12	25	18	31	24	37	30	3	36
9	2	15	8	21	14	27	20	33	26	39	32
…											
21	14	27	20	33	26	39	32	5	38	11	4

Table 7.1(b) Address Sequences with $K = 1008, f_1 = 171, f_2 = 204$

375	150	333	924	915	306	105	312	927	942	357	180
411	42	81	528	375	630	285	348	819	690	969	648
…											
423	438	861	684	915	546	585	24	879	126	789	852

Table 7.1 (c) Address Sequences with $K = 6144, f_1 = 263, f_2 = 480$

743	2446	5109	2588	1027	426	785	2104	4383	1478	5677	4692
4667	5602	1353	4208	1879	510	101	652	2163	4634	1921	168
…											
951	5726	5317	5868	1235	3706	993	5384	4591	4758	5885	1828

Due to non-availability of corresponding hardware to implement (7.1), we propose a novel algorithm which leads to low complexity implementation of the address generator. In our approach, (7.1) is divided into two sub-parts out of which the first part computes the raw addresses. These raw addresses pass through the mod function in the second part. Both parts are described by (7.2) and (7.3) respectively. Accordingly, previous MATLAB program is partially modified to generate the raw addresses without the mod function, i.e. implementation of (7.2). The raw addresses so generated for the same three cases of Table 7.1 are shown in Table 7.2(a)-(c).

$$y(i) = (f_1 * i + f_2 * i^2) \tag{7.2}$$

$$\Pi(i) = y(i) \bmod K \tag{7.3}$$

Table 7.2(a) Raw Address Sequences with $K = 40, f_1=3, f_2=10$

13	46	99	172	265	378	511	664	837	1030	1243	1476
1729	2002	2295	2608	2941	3294	3667	4060	4473	4906	5359	5832
…											
320947	324540	328153	331786	335439	339112	342805	346518	350251	354004	350251	354004

Table 7.2(b) Raw Address Sequences with $K = 1008, f_1 = 171, f_2 = 204$

375	1158	2349	3948	5955	8370	11193	14424	18063	22110	26565	31428
36699	42378	48465	54960	61863	69174	85020	93555	102498	111849	121608	131775
…											
6421383	6493974	6566973	6640380	6714195	6788418	6863049	6938088	7013535	7089390	7165653	7242324

Table 7.2(c) Raw Address Sequences with $K = 6144, f_1 = 263, f_2 = 480$

743	2446	5109	8732	13315	18858	25361	32824	41247	50630	60973	72276
84539	97762	111945	127088	143191	160254	178277	197260	217203	238106	259969	282792
…											
15084471	15255134	15426757	15599340	15772883	15947386	16122849	16299272	16476655	16654998	16834301	17014564

Careful examination of the sequence of raw addresses in Table 7.2(a)-(c) shows certain correlation between them which may be expressed by the following novel algorithm for address generator of LTE/LTE-A interleaver:

Define:

$F = f_1 + f_2;$

Fixed Increment: $INC = 2*f_2$

Initial condition: $PA = PI = F$

Subsequent address: Previous address (*PA*) + {previous increment (*PI*) + *INC*}

e.g. (a) for $K = 40, f_1 = 3, f_2 = 10$

Initial Conditions:

$PA = PI = F = 13, INC = 20,$

Subsequent addresses: $13 + (13+20) = 46; 46 + (33+20) = 99; 99 + (53+20) = 172$ and so on.

(b) For $K = 1008, f_1 = 171, f_2 = 204$

Initial Conditions:

$PA = PI = F = 375, INC = 408,$

Subsequent addresses: $375 + (375+408) = 1158; 1158 + (783+408) = 99; 99 + (1191+20) = 1290$ *and so on*.

(c) For $K = 6144, f_1 = 263, f_2 = 480$

Initial Conditions:

$PA = PI = F = 743$, $INC = 960$,

Subsequent addresses: *$743+ (743+960) = 2446$; $2446 + (1703 + 960) = 5109$; $5109 + (2663+960) = 8732$ and so on*.

The proposed algorithm computes the raw addresses recursively without involving multiplier and squarer circuit thus ensuring low complexity implementation.

Implementation of (7.3), i.e. computation of modulus on raw addresses is done by suitably modifying the algorithm proposed by Butler & Sasao [169]. This algorithm computes x mod z as a modulo reduction process, where at each stage, the magnitude of x is reduced, but the residue remains the same which is continued until only the residue remains. As the interleaver block size K (divisor) has 188 different values ranging from 40 to 6144, the second method of Butler & Sasao [169] where the divisor is an independent variable is adopted. In the proposed case, as shown in Table 7.2(c), the maximum value whose modulus is to be computed is 17014564 which in binary requires 25-bits representation. As a result value of $i = 25$ in computation of $\Theta = z*2^i$ where $i-1$ represents number of comparison stages, z represents the divisor ($= K$) and Θ is defined to be the first value to be subtracted from the dividend, X. Subsequent values of Θ are computed by dividing present value of Θ by 2. In our work, division is accomplished by right shift which is more resource efficient than direct division technique.

7.4. Hardware Realization

In order to test the functionality of the proposed algorithm, corresponding digital hardware is designed. Top level view of the hardware is presented in Fig. 7.1 The first block is Raw Address Generator, implementing (7.1), receives initiation pulse (*INIT*), Clock signal (*CLK*) and a memory pointer (*I*) to retrieve the corresponding values of K, f_1 and f_2 stored in an LUT [76]. The raw addresses (X), and K values are passed on to the MOD circuit which finally generates the desired interleaver addresses.

The Raw Address Generator as shown in Fig. 7.2 comprises of two subsidiary units namely, an LUT (a), a START signal generator (b) along with the main unit (c). The LUT stores the values of K, f_1 and f_2 and has 32-bit width. The organization of data in the LUT is described in Table 7.3. A desired combination of the block size K, f_1 and f_2 is read from the LUT by supplying appropriate address of LUT in I. An adder is used to generate $F = f_1 + f_2$. A *START* signal is generated by the hardware shown in Fig. 7.2(b) either at the beginning of the operation or after completion of one period of generating 188 interleaver addresses. *INIT* pulse is responsible for generating the *START* signal at the beginning whereas an active *reset* signal generates the subsequent *START* signal on completion of each addressing cycles. The main circuit responsible for raw address generation receives signal F, f_2 and *START* from other two ancillary units and the system *CLK*. Active *START* pulse causes $M_1 = F$, $M_2 = INC = 2^* f_2$, $M_3 = F$ and $M_4 = 0$ representing the initial condition of the circuit. Registers R_1 and R_2 store the output of A_1 and M_3 respectively.

Fig. 7.3 describes the modified hardware used to compute X mod K. In this application, i = 25 and z is a 13-bit number resulting in Θ to be a 38-bit number. This consequents 25-bit X to be converted into a 38-bit number by appending 13 zeros in the MSBs so that X and Θ may be compared. In order to make our design resource efficient we computed Θ by appending 25 number of zeros in the LSB of Z instead of using a multiplier directly. The left most comparator (C_{24}) as shown in Fig. 7.3 compares A_{24} (38-bit version of X) with Θ_{24}. If $A_{24} \geq \theta_{24}$, the select input of the multiplexer (M_{24}) receives a 0, thus routing Θ_{24} to the output (= B_{24}). The subtractor (S_{24}) performs first stage reduction by computing $A_{23} = A_{24} - B_{24}$. In case, $A_{24} \ngeq \theta_{24}$, M_{24} receives $B_{24} = 0$, the stage performs *pass through* operation. Similar operation is carried out in 23 subsequent stages as shown in Fig. 7.3 till A_0 is computed which is converted into 13-bit number, Y removing 25 zeros from MSBs before sending it as output of the address generator. Subsequent Θs (i.e. Θ_{23}, $\Theta_{22}, ..., \Theta_1$) are computed from previous Θ values dividing by 2. Instead of using a divisor, the author employed shifter to perform ÷2 operation thereby making design more resource efficient.

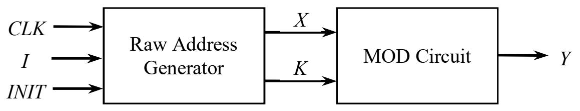


Fig. 7.1 Top level view of the QPP interleaver address generator

Table 7.3 Organization of LUT

I (9 bit)	*K (13 bit)*	*f_1 (9 bit)*	*f_2 (10 bit)*
000H	01400C0AH		
001H	01801C0CH		
002H	01C04C2AH		
…	…		
05BH	01F80DC54H		
…	…		
0BBH	C0041DE0H		

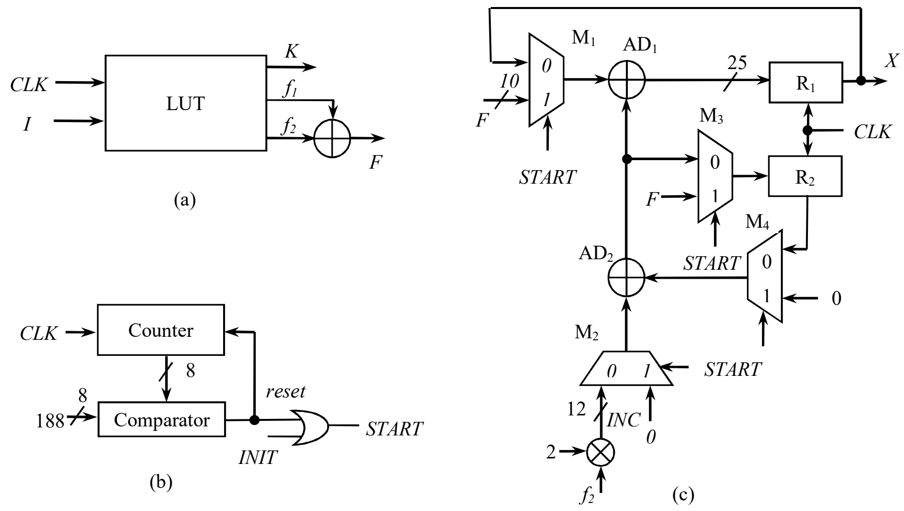


Fig. 7.2 Proposed Hardware of Raw Address Generator
(a) LUT (b) *START* signal Generator (c) Main Unit

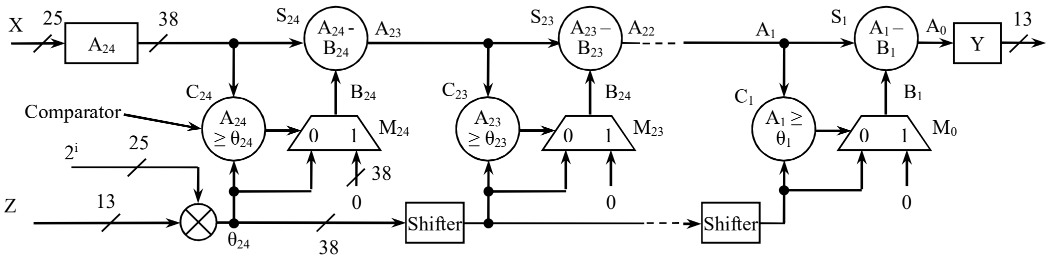
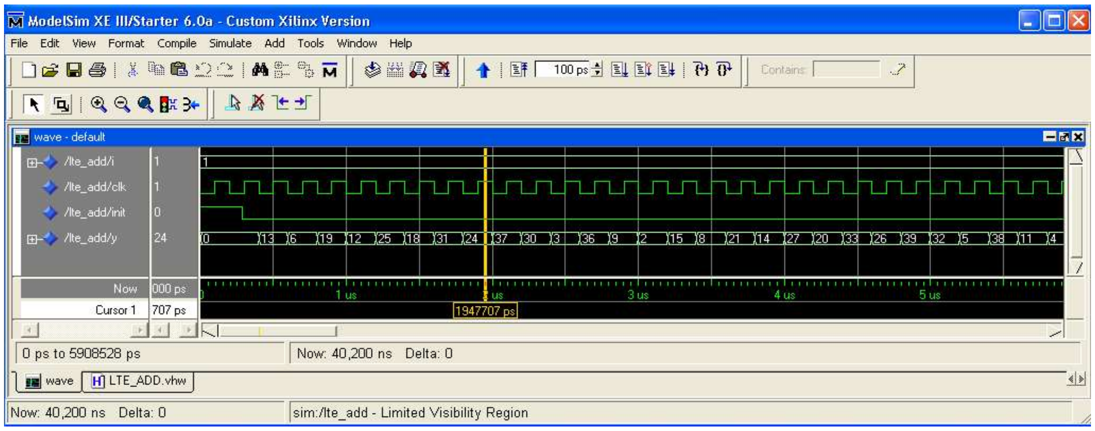


Fig. 7.3 Modified MOD circuit to compute X mod Z

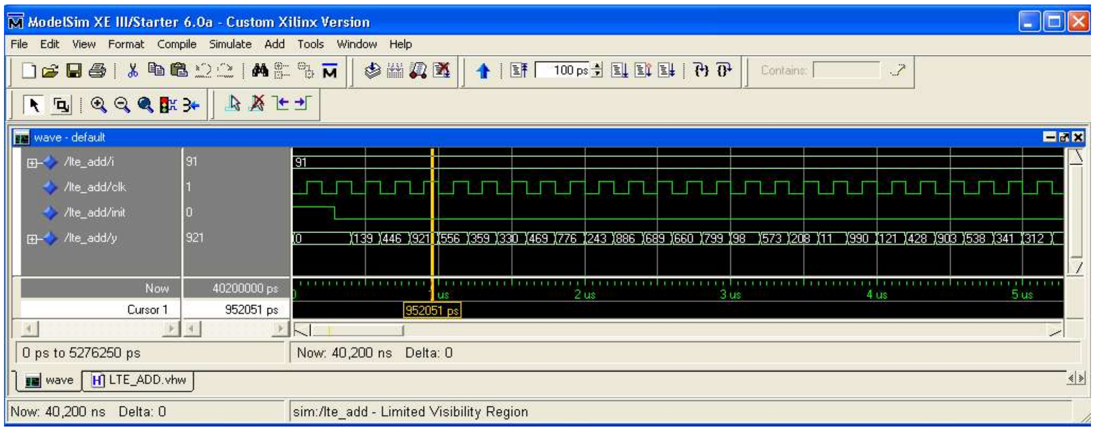
7.5. Simulation Results for QPP Interleaver

The simulation results in the form of timing diagram obtained using ModelSim Xilinx Edition-III for *i = 1(K=40), 91 (K=1008) and 188 (K=6144)*, are shown in Fig. 7.4(a), (b) and (c) respectively. The captured portion show the interleaver addresses generated for the first few cases and are identical with Table 7.1(a), (b) and (c) respectively. The circuit fetches $K=40$, $f_1=3$ and $f_2=10$ from the LUT of Fig. 7.2(a) when supplied with $i=1$. As shown in Fig. 7.4(a)-(c), *init = 1* in the beginning for one clock pulse to enable hardware of Fig. 7.2(b) to generate *START* pulse. On completion of an iteration of address generation, the *START* pulse for subsequent iterations is generated by the *reset* signal. Based on the values received against the signals *START, F and* f_2, the main unit of the address generator as shown in Fig. 7.2(c) computes the desired addresses and are made available at the output as y in Fig. 7.4(a)-(c). The author has generated and verified addresses for all values of *i,* however to avoid clumsiness other results are not included.


ModelSim XE III/Starter 6.0a - Custom Xilinx Version
File Edit View Format Compile Simulate Add Tools Window Help
100 ps
Contains:
wave - default
Now
Cursor 1
1 us
3 us
4 us
5 us
wave
LTE_ADD.vhw

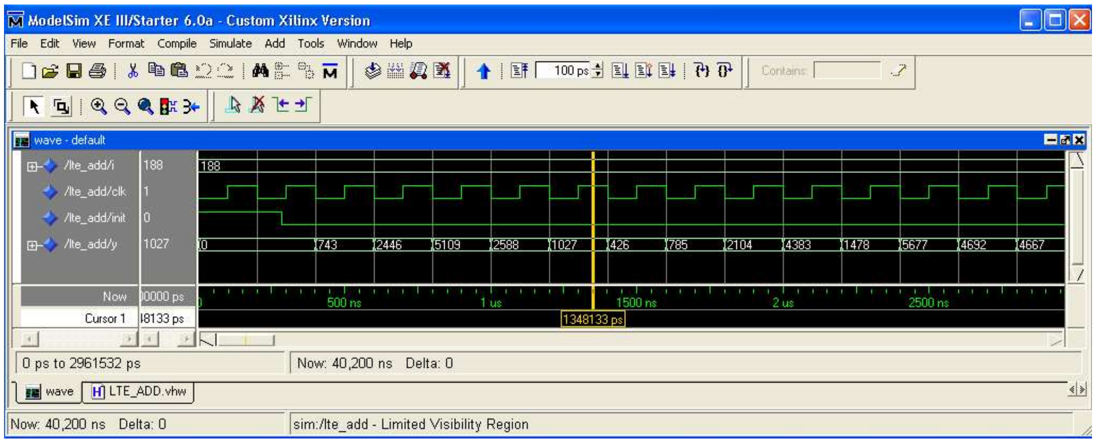
```
/lte_add/i      1     1
/lte_add/clk    1
/lte_add/init   0
/lte_add/y      24    0 13 6 19 12 25 18 31 24 37 30 3 36 9 2 15 8 21 14 27 20 33 26 39 32 5 38 11 4
Now             000 ps
Cursor 1        707 ps    1947707 ps
0 ps to 5908528 ps        Now: 40,200 ns  Delta: 0
Now: 40,200 ns  Delta: 0  sim:/lte_add - Limited Visibility Region
```

Fig. 7.4(a) Timing simulation showing initial addresses for *i=1* (K = 40, f_1=3, f_2=10)


ModelSim XE III/Starter 6.0a - Custom Xilinx Version
File Edit View Format Compile Simulate Add Tools Window Help
100 ps
Contains:
wave - default
Now
Cursor 1
us
2 us
3 us
4 us
5 us
wave
LTE_ADD.vhw

```
/lte_add/i      91    91
/lte_add/clk    1
/lte_add/init   0
/lte_add/y      921   0 139 446 921 556 359 330 469 776 243 886 689 660 799 98 573 208 11 990 121 428 903 538 341 312
Now             40200000 ps
Cursor 1        952051 ps   952051 ps
0 ps to 5276250 ps        Now: 40,200 ns  Delta: 0
Now: 40,200 ns  Delta: 0  sim:/lte_add - Limited Visibility Region
```

Fig. 7.4(b) Timing simulation showing initial addresses for *i=91*
(K = 1008, f_1 = 171, f_2 = 204)


ModelSim XE III/Starter 6.0a - Custom Xilinx Version
File Edit View Format Compile Simulate Add Tools Window Help
100 ps
Contains:
wave - default
Now
Cursor 1
500 ns
1 us
1500 ns
2 us
2500 ns
wave
LTE_ADD.vhw

```
/lte_add/i      188   188
/lte_add/clk    1
/lte_add/init   0
/lte_add/y      1027  0 743 2446 5109 2588 1027 426 785 2104 4383 1478 5677 4692 4667
Now             )0000 ps
Cursor 1        )8133 ps   1348133 ps
0 ps to 2961532 ps        Now: 40,200 ns  Delta: 0
Now: 40,200 ns  Delta: 0  sim:/lte_add - Limited Visibility Region
```

Fig. 7.4(c) Timing simulation showing initial addresses for *i=188*
(K= 6144, f_1 = 263, f_2 = 480)

7.6. FPGA Implementation Result and Analysis

The proposed hardware structure of LTE / LTE-A interleaver address generator is transformed into VHDL model using Xilinx Integrated Software Environment (ISE 12.1) and has been implemented on Xilinx Spartan 6 FPGA. Table 7.4 shows the HDL synthesis report for the implementation. The adder of Fig. 7.2(a) is realized by the 10-bit adder. The 13-bit and 38-bit subtractors are used in the mod circuit to determine the *Y* and $A_{x-1}=A_x-B_x$. AD_1 and AD_2 adders of Fig. 7.2(c) are implemented through two 25-bit adders. The input applied through *I* is converted into LUT address with the help of the 9-bit subtractor to access the content of LUT described by Table 7.3. An 8-bit up-counter is used to implement the counter of Fig. 7.2(b). The 1-bit, 25-bit registers are used to implement reset signal, R1 and R2 register respectively. Fig. 7.2(b) uses the 8-bit greater than comparator to generate the reset pulse whereas the 24 numbers of 38-bit less than equal to comparators are used in the mod circuit to implement C_{24}, C_{23}, C_{22} .....,C_1. Similarly the mod circuit of Fig. 7.3 also uses 24 number of 38-bit 2-to-1 multiplexers (M_{24}, M_{23}, M_{22} .....,M_1). The 25-bit 2-to-1 multiplexers implements M1-M4 of Fig. 7.2(c)

Table 7.4 HDL Synthesis Report

Logic Circuits Used	Quantity	Logic Circuits Used	Quantity
10-bit adder	1	1-bit register	1
13-bit subtractor	1	25-bit registers	2
25-bit adder	2	8-bit comparator greater	1
38-bit subtractor	24	38-bit comparator less equal	24
9-bit subtractor	1	25-bit 2-to-1 multiplexer	4
8-bit up counter	1	38-bit 2-to-1 multiplexer	24

Direct comparison of our proposed work with the existing works in [124], [123], [126] is not possible due to dissimilarity in implementation platform and non-availability of implementation result for interleaver alone. However, in order to compare the implementation results, we have modelled and implemented the conventional LUT based technique in the same Spartan 6 FPGA platform. As per [76], there are 188 permissible block sizes leading to 188 memory blocks required to house the address LUTs. Such requirement cannot be catered by the target FPGA which only possess 52 BRAM blocks. To solve the problem, the author partitioned the available memory blocks which in turn helped to reduce memory wastage as well. Implementation results of both the techniques have been shown in Table 7.5. Pictorial representation of Table 7.5 is also incorporated in Fig. 7.8 for quick comparison and analysis of the implementation results. Our proposed technique shows significant reduction in Block RAM requirement by 71.16% in comparison with improved LUT based technique. Similarly our proposed design shows significant improvement by 82.26% in terms of maximum operating frequency. This improvements are at the cost of minor increase in FPGA Slice LUT requirement.

Table 7.5 Comparative Device Utilization Summary

FPGA Resources / Parameters	This work		Improved LUT based implementation		Remarks
	Utilization in Number	Utilization in %	Utilization in Number	Utilization in %	
Number of Slice Registers	59 out of 30064	0.19	63 out of 30064	0.21	Reduction by 0.02%
Number of Slice LUTs	1128 out of 15032	7.50	433 out of 15032	2.88	Increase by 4.62%
Number of Bonded IOBs	23 out of 240	9.58	23 out of 240	9.58	No change
Number of Block RAM	1 out of 52	1.92	38 out of 52	73.08	Reduction by 71.16%
Maximum clock speed	260.92MHz		143.16MHz		Speed improvement by 82.26%

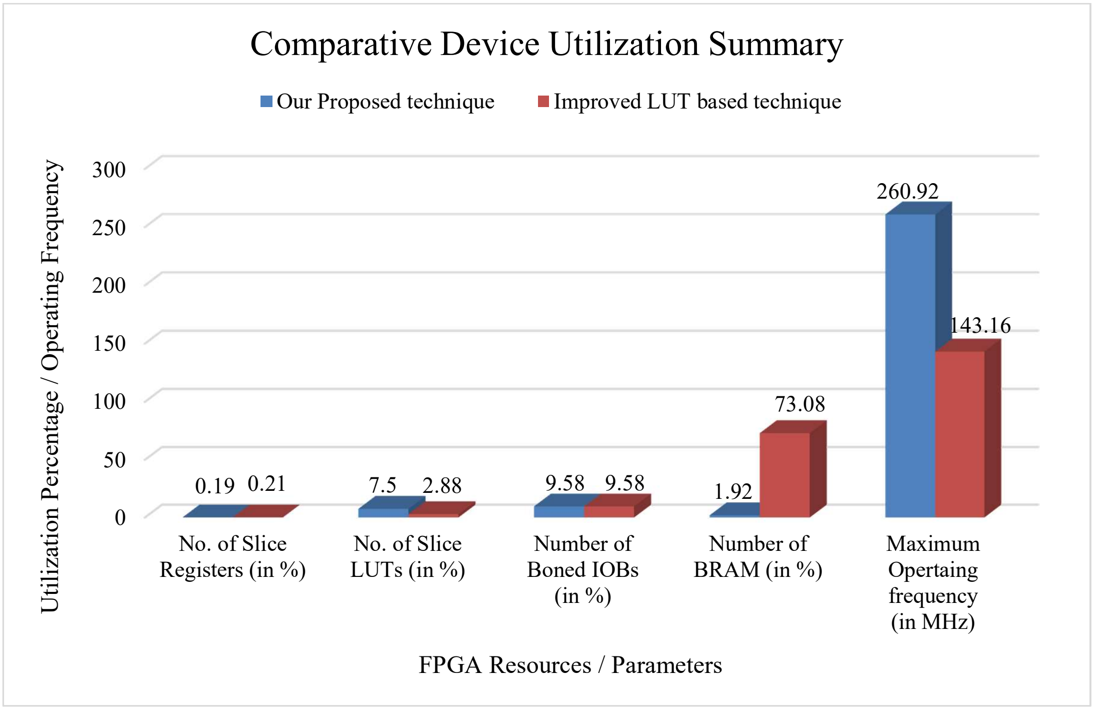


Fig. 7.8 Bar Chart Representation of FPGA Device Utilization Summary

7.7. Discussion

This work describes a novel and efficient algorithm to model the interleaver address generator used in LTE/LTE-A. The algorithm exploits the correlation between the consecutive addresses of the address generator of QPP interleaver to efficiently model and implement on reconfigurable hardware. The algorithm is converted into digital hardware and implemented on Xilinx Spartan 6 FPGA using Xilinx ISE 12.1. Functionality of the address generator has been tested through timing simulation using ModelSim XE-III software. The proposed work when compared with conventional LUT based work shows significant improvement in terms of embedded memory utilization and operating speed. Next chapter summarises the complete work done during this doctoral research activity including potential areas for future research in continuation to this work.

Chapter 8

Conclusion and Future Works

8.1 Conclusive Remarks

In digital communication systems, interleavers play an important role in reducing the effect of burst error encountered in the channel during transmission. Design of digital hardware for the interleaver address generator used in OFDM based wireless standards like DAB, DVB, WLAN, WiMAX, MIMO-WLAN and LTE/ LTE-A are **highly challenging due to the presence of complex functions like floor, modulus and square.** The principal aim of this research work was to design hardware efficient interleaver for various OFDM based high speed wireless communication systems. FPGAs are most preferred reconfigurable hardware platform for implementation of newer algorithms due to advantages like shorter turnaround time, ease of future upgradation, obsolescence free design and direct linkage of MATLAB with software tools for HDL designs like Xilinx Integrated Software Environment (ISE). In this work, efficient hardware implementations of both types of interleavers, i.e. convolutional and block, involving all permissible code rates and modulation types have been carried out. MATLAB programs have been developed to generate the desired interleaver addresses. At first, block level representation of the designs is prepared. Novel algorithms have been proposed to model these blocks. Verification of the proposed algorithms has been accomplished through MATLAB programs. Each block is decomposed into most suitable digital circuits which thereafter are converted into appropriate VHDL models using Xilinx ISE. Finally, the VHDL models are implemented on Xilinx FPGAs like Spartan 3, 3AN and 6. Timing simulations of the interleaver address generators / interleavers have been extensively carried out to verify functionality of the proposed designs.

At the outset, design and implementation of convolutional interleaver for DAB application has been considered. The work utilizes FPGA’s embedded shift registers SRLC16 to model the incremental memory of the interleaver. **This modeling lowers the hardware resource occupancy of FPGA by 81% over implementation without embedded shift registers.** The power consumption of the convolution interleaver hardware is found to be as low as 125mW. Due to the use of SRLC16, interconnection delay inside the FPGA is reduced thereby improving the operating speed of the convolutional interleaver. **Reduction in memory wastage by 30% over an existing implementation is another important contribution of the work.**

Conventionally Look-up Table (LUT) based technique is employed in designing the block interleaver used in IEEE 802.11 a/g based WLAN transceiver. Two approaches namely improved LUT based and Finite State Machine (FSM) based have been adopted by the author in designing the hardware for the interleaver. **The former technique demonstrates reduction in resource utilization like slices, flip-flop and LUTs by 43%, 34% and 50% respectively over the conventional LUT based approach. Similar results have also been obtained for FSM based implementation. In addition, the FSM based technique offers 25% faster performance over the conventional LUT based method.**

WiMAX is another BWA based on IEEE 802.16 d/e standard which uses special type of block interleaver. Conventionally, LUTs are used to generate the interleaver addresses. The author has proposed improved LUT based technique to generate de-interleaver addresses. **The improvement in terms of memory saving above 81% of memory blocks and 30% faster circuit operation over the conventional LUT based approach have been achieved.** Next, FSM based interleaver address generator for WiMAX system has been proposed. The work has been carry forward to design the complete FSM based interleaver (with memory) for the WiMAX application. **This work too shows approximately 30% improvement in terms of maximum operating clock frequency, approximately 46% improvement in FPGA flip- flop used with negligible (less than 3%) loss in terms of LCs used over the existing implementation.** Finally design of a low complexity and resource efficient hardware for de-interleaver has been proposed. It includes a novel algorithm for the de-interleaver with user-friendly mathematical representation followed by general validity. **Use of FPGA's embedded multiplier and sharing of resources among the QPSK, 16-QAM and 64-QAM blocks exhibit significant reduction in occupancy of FPGA slices (by 80.24%), flip-flops (by 35.9%) and 4 input LUTs (by 80.47%) along with 95% faster operation than LUT based approach.**

Another important work related to speed power improved hardware design of interleaver address generator for use in MIMO WLAN has been carried out. This work contributes hardware efficient model of MIMO WLAN interleaver completely eliminating the need for floor and modulus functions. The work is also extended to model the interleaver memory using FPGA's embedded memory and thus provides complete hardware interleaver solution. **The proposed design when compared with recent works shows noticeable betterment in terms of maximum operating frequency by 12.56% and power consumption by 65.64%.**

Finally the work related to the design of hardware efficient Quadratic Permutation Polynomial (QPP) interleaver address generator for LTE / advanced LTE communication system has been taken up. The address generator involves a quadratic equation and modulus function which do not have corresponding digital hardware. A novel algorithm has been proposed to eliminate the need of squarer and modulus functions. The algorithm is converted into digital hardware which is also implemented on a reconfigurable platform. **This approach shows significant reduction in Block RAM requirement by 71.16% in comparison with improved LUT based technique along with the significant improvement of maximum operating frequency by 82.26%.**

A summary of wireless applications for which various interleaver designs have been carried out along with the implementation platforms used against them in the entire doctoral research work is presented in Table 8.1. This table also highlights the embedded resources of the target FPGAs used in the work.

Table 8.1 Summary of Interleaver Applications and FPGA Implementation Platforms used in this Research Work

Application	Xilinx FPGA Generations	Devices	Embedded Resources used
Convolutional Interleaver for DAB	Spartan-3	XC3S 400	SRLC16 (Embedded Shift Register)
Block Interleaver for WLAN	Spartan-3	XC3S 400	Block RAM, Distributed RAM
Block Interleaver for WiMAX	Spartan-3, Spartan-3AN	XC3S 400, XC3S1400AN	Block RAM, MULT18X18 (Embedded Multiplier)
Block Interleaver for MIMO WLAN	Spartan-6	XC6SLX25	Block RAM, DSP48A1s (DSP Block)
QPP Interleaver for LTE/LTE-A	Spartan-6	XC6SLX25	Block RAM

8.2 Prospective Future Work

Like “space never ends”, research never ends either. Though the works incorporated in this thesis provide efficient solution to interleaver design and implementation issues on FPGA platform through VHDL modeling, following points may be considered as potential candidate for future development in continuation to this work.

- Development of FPGA based Dual mode ARP and QPP block interleavers supporting MIMO WLAN as well as LTE/LTE-A standard simultaneously.
- Implementation of convolutional interleaver for advanced Speech Signal Processing.
- Design and implementation of Multi-Dimensional Interleavers for advanced image transmission.
- Interleavers for high speed successful data transmission over severe burst error communication channel.
- Advanced Interleavers for Multi-functional MIMO systems in 5G Wireless Communication.
- Application of the knowledge of interleavers in RF Wireless system for corresponding transformation to Optical/Quantum Wireless systems.
- Massive MIMO Signal Processing techniques utilizing the principles of the interleavers incorporated in this thesis for Optical Wireless Communication.

Bibliography

[1] J. G. Proakis, Digital Communications, 3rd ed. New York: McGraw-Hill, 1995.

[2] F. B. Samuel and LL. D, Morse, “Examination of the Telegraphic Apparatus and the Processes in Telegraphy”, Philp & Solomons, Washington, 1869, pp. 7-36.

[3] http://www.elon.edu/e-web/predictions/150/1830.xhtml

[4] L. Coe, “The Telegraph: A History of Morse’s Invention and its Predecessors in the United States”, McFarland & Co., 1993.

[5] A. Odlyzko, “The history of communications and its implications for the Internet” available at http://dspace.mit.edu / bitstream / handle / 1721.1/1525/ history_communications2.pdf.

[6] C. S. Fischer, “America Calling: A Social History of the Telephone to 1940”, University of California Press, USA, 1948, pp. 1-32.

[7] Alexander Graham Bell Family Papers at the Library of Congress available at http:// www.loc.gov/ collections/ alexander-graham-bell-papers/ articles-and-essays/ telephone-and- multiple - telegraph/

[8] LRF. Harris and J. Martin “Evolution of switching system architecture” The Post Office Electrical Engineers' Journal, vol. 74, Oct. 1971, pp.187-193.

[9] C. Breen and C. A. Dahlbom, “Signalling Systems for Control of Telephone Switching”, B.S.T.J. 39, no.9, pp. 1381-1444.

[10] J. E. Flood (ed.) “Telecommunication Networks”, IEE, 2nd Edition, Chapter 1.

[11] S. F. Smith,. “Telephony and Telegraphy: An Introduction to Instruments and Switching Systems”, Oxford University, 1978.

[12] http://www.thg.org.uk/index.php/component/content/article?id=115.

[13] F. Redmill and A. R. Valdar, “SPC Digital Exchanges”, Peter Peregrinus Publisher, 1994.

[14] T. Viswanathan, “Telecommunication Switching Systems and Networks”, PHI Publication, 2006, Chapter 1.

[15] http://en.wikipedia.org/wiki/Invention_of_the_telephone

[16] G. Kennedy and B. Davis, “Electronic Communication System”, Tata Mcgraw Hill Edition, New Delhi, 1999, pp 2-5.

[17] R. R. Gulati, “Monochrome and Color Television” Wiley Eastern Limited, 2nd Edition, Chapter 1.

[18] R. P. Singh and S. D. Sapre, "Communication Systems: Analog & Digital", McGraw-Hill Publisher, New Delhi, 2007.

[19] A. Goldsmith, "Wireless Communications", Cambridge University Press, New York, USA, 2005.

[20] T. S. Rappaport, "Wireless Communications: Principles and Practice", Prentice-Hall, Englewood Cliffs, NJ, 1996

[21] W. Stalling, "Wireless Communications and Networks", Pearson Publication, New Delhi, India, 2005

[22] V. T. Yadugiri, "Jagadish Chandra Bose", Current Science, vol. 98, no. 7, April, 2010, pp. 975-77.

[23] A. S. Molisch, "Wireless Communication", John Wiley and Sons Publication, UK, 2005

[24] K. L. Du and M. N. S. Swamy, "Wireless Communication Systems: From RF Subsystems to 4G Enabling Technologies", Cambridge University Press, UK, 2010.

[25] T. L. Singhal, "Wireless Communications", Tata McGraw Hill Publication, New Delhi, 2010.

[26] B. Leung, "VLSI for Wireless Communication", 2nd Edition, Springer, USA 2011.

[27] D. Markovic´, B. Nikolic´, and R. W. Brodersen, "Power and area efficient VLSI architectures for communication signal processing," in Proc. Int. Conf. Communications, Jun. 2006.

[28] D. Markovic, B. Nikolic, and R.W. Brodersen, "Power and area minimization for multidimensional signal processing," IEEE J. Solid-State Circuits, vol. 42, no. 4, April 2007, pp. 922–934.

[29] A. P. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Brodersen, "HYPER-LP: a system for power minimization using architectural transformations," in Computer-Aided Design, 1992. ICCAD-92. Digest of Technical Papers, 1992 IEEE/ACM International Conference on, Santa Clara, CA, Nov. 1992, pp. 300–303.

[30] P. Banerjee, M. Haldar, A. Nayak, V. Kim, V. Saxena, S. Parkes, D. Bagchi, S. Pal, N. Tripathi, D. Zaretsky, R. Anderson, and J. Uribe, "Overview of a compiler for synthesizing MATLAB programs onto FPGAs," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 3, Mar. 2004, pp. 312–324.

[31] http://www.corp.att.com/attlabs/reputation/timeline/46mobile.html

[32] W. R. Young, “Introduction, Background and Objectives”, Bell Systems Technical Journal, vol. 58, January, 1979, pp. 1–14.

[33] T. R. Anderson, T. U. Daim, and J. Kim, “Technology forecasting for wireless communication” Technovation, 28(9), 2008, pp. 602–614.

[34] V. H. McDonald, “The Cellular Concept”, Bell Systems Technical Journal, vol. 58, January,1979, pp. 15–49.

[35] A. Jamalipour, T. Wada, T. Yamazato, “A Tutorial on Multiple Access Technologies for Beyond 3G Mobile Networks”, IEEE Communication Magazine, vol. 43, February, 2005.

[36] A. Furaskär, S. Mazur, F. Müller, and H. Olofsson, “Edge: Enhanced data rates for GSM and TDMA/136 evolution,” IEEE Personal Commun., vol. 6, June 1999, pp. 56–66.

[37] I. S. Misra, “Wireless Communications and Networks: 3G and Beyond”, McGraw Hill (India) Publication, New Delhi, 2013

[38] Third Generation (3G) Wireless White Paper, Trillium Digital System, March 2000.

[39] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi, “QUALCOMMCDMA/HDR: A Bandwidth-Efficient High-Speed Wireless Data Service for Nomadic Users”, IEEE Communication Magazine, July, 2000.

[40] D. Tse, and P. Biswanath, “Fundamentals of Wireless Communications”, Cambridge University Press, UK, 2005.

[41] D. E. Comer, “Computer Networks and Internet”, 5e, Upper Saddle River, NJ: Prentice Hall, Chapter 12.

[42] G. Ginis, and J. M. Cioffi, “Vectored Transmission for Digital Subscriber Line Systems”, IEEE Journal on selected areas in Communications, vol. 20, no. 5, June 2002, pp. 1085-1104.

[43] J. A. Hausman, J. G. Sidak and H. J. Singer “Cable Modems and DSL: Broadband Internet Access for Residential Customers, American Economic Review”, vol. 91 no. 2, 2001, pp. 302-307.

[44] M. S. Kuran and T. Tugcu, “A Survey on Emerging Broadband Wireless Access Technologies,” Computer Networks, vol. 51, no. 11, Aug. 2007, pp. 3013–3046.

[45] P. Roshan and J. Leary, “802.11 Wireless LAN Fundamentals” CISCO Press, USA, 2004.

[46] Q. Ni, L. Romdhani, and T. Turletti, “A Survey of QoS Enhancements for IEEE 802.11 Wireless LAN”, Journal of Wireless Communications and Mobile Computing, Wiley, vol. 4, issue 5, 2004, pp. 547-566.

[47] "Radio Regulations, Edition of 2012". ITU-R. Retrieved 2014-11-10.

[48] C. Eklund, R. B. Marks, K. L. Stanwood and S. Wang, “IEEE Standard 802.16: A Technical Overview of the WirelessMAN™ Air Interface for Broadband Wireless Access”, IEEE Communication Magazine, June, 2002, pp. 98-107.

[49] A. Ghosh, D. R. Wolter, J. G. Andrews, R. Chen, “Broadband Wireless Access with WiMAX/802.16: Current Performance Benchmarks and Future Potential”, IEEE Communication Magazine, vol. 43, February, 2005, pp. 129-136.

[50] L. Nuaymi, “WiMAX: Technology for Broadband Access”, John Wiley and Sons Publication, UK, 2007

[51] K. C. Chen and J. R. B. D. Marca (eds.), “Mobile WiMAX”, John Wiley and Sons Publication, UK, 2008

[52] IEEE 802.16-2004, Local and Metropolitan Networks — Part 16: Air Interface for Fixed Broadband Wireless Access Systems, 2004.

[53] IEEE 802.16e-2005, IEEE standard for local and metropolitan area networks—part 16: air interface for fixed Broadband Wireless Access Systems—Amendment 2, 2005.

[54] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj and H. V. Poor, “MIMO Wireless Communication”, Cambridge University Press, UK, 2007.

[55] A. Ashtaiar, “MIMO-Based Wireless Local Area Networks”, Lambert Academic Publishing, Germany, 2010.

[56] A. Ghosh, J. Zhang, J. G. Andrews, and R. Muhamed, “Fundamentals of LTE”, Prentice-Hall, 2010.

[57] S. Parkvall, E. Dahlman, A. Furuskar, Y. Jading, M. Olsson, S. Wanstedt, and K. Zangi, “LTE-advanced Evolving LTE towards IMT-advanced”, Proc. 68th IEEE Vehicular Technology Conference, Sep. 2008.

[58] A. Ghosh, R. Ratasuk, B. Mondal, N. Mangalvedhe, and T. Thomas, “LTE-Advanced: Next-Generation Wireless Broadband Technology”, IEEE Wireless Communication Magazine, June, 2010, pp. 10-22.

[59] E. Dahlman, S. Parkvall and J. Skold. "4G: LTE/LTE-Advanced for Mobile Broadband", Academy Press, UK, 2014

[60] A. Ghosh and R. Ratasuk, "Essentials of LTE and LTE-A", Cambridge University Press, UK, 2011

[61] C. Cox, "An Introduction to LTE", John Wiley Publication, UK, 2012.

[62] M. Sauter, "From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband", John Wiley and Sons Publication, UK, 2011

[63] M. O. Kolawole, "Satellite Communication Engineering", Chapter 1, Marcel Dekkar Publication.

[64] K. N. R. Rao, "Satellite Communication: Concept and Applications", PHI Publication, New Delhi, 2013.

[65] R. Prasad, "OFDM for wireless communications systems," Artech House Publishers, Boston, 2004.

[66] T. Hwang, C. Yang, G. Wu and G. Y. Li, "OFDM and Its Wireless Applications: A Survey", IEEE Transaction on Vehicular Technology, vol. 58, no. 4, May, 2009, pp. 1673-1694.

[67] U. S. Jha and R. Prasad, "OFDM Towards Fixed and Mobile Broadband Wireless Access" Artech House Publisher, London, 2007.

[68] G. M. Babler, "A Study of Frequency Selective Fading for Microwave Line of Sight Narrow Band Radio Channel" B. S. T. J. 51 no. 3 (March 1972), pp. 731-57.

[69] A. Labouebe and Y. Gouville, "60 GHz Wireless Communication Characteristics, system performance and hardware requirements" Masters Thesis, Chalmers University of Technology, CTH, Sweden.

[70] T. D. Chiueh and P. Y. Tsai, OFDM Baseband Receiver Design for Wireless Communications. Hoboken, NJ: Wiley, 2007.

[71] Digital Audio Broadcasting (DAB); Data Broadcasting – MPEG-2 TS Streaming, ETSI TS 102 427 (V1.1.1) (2005-07).

[72] Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television, ETSI TS 300 744 (V.1.6.1) (2009-01).

[73] Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H), ETSI EN 302 304 (V.1.1.1) (2004-11).

[74] IEEE std. 802.11a-1999, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High Speed Physical Layer in the 5 GHz Band,” July 1999.

[75] IEEE 802.11n-2009: ‘Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: enhancements for higher throughput,’ The IEEE Standards Association, New York, NY, USA, 2009.

[76] “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3GPP System Architecture Evolution: GPRS Enhancements for LTE Access; Release 8”, 3GPP, 3GPP TS 23.401.

[77] 3GPP TR 136.912, “LTE; Feasibility study for Further Advancements for E-UTRA (LTE-Advanced),” V10.0.0, 2011; http://www.etsi.org/

[78] H. Rohling (ed.), “OFDM: Concept for Future Communication Systems”, Springer, Heidelberg, Germany, 2011.

[79] R. W. Hamming, “Error Detecting and Error Correcting Codes”, B. S. T. J. vol 29, no. 2 (March 1950), pp. 147-60.

[80] S.B. Wicker, “Error Control System for Digital Communication and Storage”, Englewood Cliffs, NJ: Prentice-Hall, Inc, 1995.

[81] Yun Q. Shi, Xi Min Zhang, Zhi-Cheng Ni, and Nirwan Ansari, “Interleaving for Combating Bursts of Errors”, IEEE Circuits and System magazine, first quarter, 2004, pp. 29-42.

[82] B. Sklar, Digital Communications: Fundamental and Applications, 2/e, Pearson Education, New Delhi, 2001.

[83] S. A. Hanna, ‘Convolutional interleaving for digital radio communications’, Second IEEE International Conference on Personal Communications: Gateway to the 21st Century, vol.: 1, pp. 443-447, 1993.

[84] S. Nag, S. Shenoy and B. U. Chandrashekar, “Hardware Implementation of a Combined Interleaver and DeInterleaver”, Design and Re-use, IPSOC, 2006.

[85] G. C. Clark Jr. and J. B. Cain, “Error-Correction Coding for Digital Communications” Springer, 1981

[86] Seagate Technology Llc, “Low complexity pseudo-random interleaver” US Patent, US7395461, July 2008.

[87] Divsalar and Pollara, “Turbo code for PCS applications”, in Proc. of ICC 1995, Seattle, WA, June, 1995, pp. 54-59.

[88] T. Richter and G. Fettweis, “Parallel Interleaving on Parallel DSP Architectures,” Proc. 2002 Workshop on Signal Processing Systems (SiPS ’02), Oct. 2002, pp. 195–200.

[89] L. Hanzo, J. Akhtman, L. Wang, and M. Jiang, “MIMO-OFDM for LTE, WIFI and WIMAX: Coherent Versus non-Coherent and Cooperative Turbo-Transceivers”, Chichester, U.K.: Wiley/IEEE Press, 2010.

[90] W. Hoeg and T. Lauterbach, “Digital Audio Broadcasting: Principles and Applications of DAB + and DMB”, John Wiley Publication, UK, 2009.

[91] D. J. MacKay, “Good Error-Correcting Codes Based on Very Sparse Matrices”, IEEE Transaction on Information Theory, vol. 45, no. 2, Mar. 1999, pp. 399–431.

[92] Douglas L. Perry, “VHDL: Programming by Example”, Tata McGraw-Hill, New Delhi, 2002.

[93] J. Rose, R. J. Francis, D. Lewis and P. Chow, “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency,” IEEE J. Solid-State Circuits, vol. 25, no. 5, Oct. 1990, pp. 1217-1225.

[94] Xilinx ‘Spartan-3 Date sheet’ available at www.xilinx.com

[95] Xilinx ‘Spartan-3AN Date sheet’ available at www.xilinx.com

[96] Xilinx ‘Spartan-6 Date sheet’ available at www.xilinx.com

[97] H. Yang, Y. Zhong, and L. Yang, “An FPGA Prototype of a Forward Error correction (FEC) Decoder for ATSC Digital TV”, IEEE Transaction on Consumer Electronics, vol. 45, no. 2, 1999, pp. 387-395.

[98] J. B. Kim, Y. J. Lim, and M. H. Lee, “A Low Complexity FEC Design for DAB”, IEEE ISCAS, Sydney, Australia, 2001, pp. 522-525.

[99] R. Asghar and D. Liu, “Low Complexity Multi Mode Interleaver Core for WiMAX with Support for Convolutional Interleaving”, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering vol.3, no.4, 2009, pp. 935-944.

[100] A. Unnikuttan, M. Rathna, P. R. Rekha, and R. Nandakumar, “Design of Convolutional Interleaver”, International Journal of Innovative Research in Information Security, vol. 1, issue 4, Nov., 2014, pp. 33-39.

[101] L. M. Gaetzi and M. O. J. Hawksford, “Performance prediction of DAB modulation and transmission using Matlab modeling”, Proceedings of IEEE International Symposium on Consumer Electronics, 2004, pp. 272-277.

[102] E. Tell, and D. Liu, “A hardware architecture for a multimode block interleaver”, ICCSC, Moscow, Russia, June 2004.

[103] A. Sghaier, S. Ariebi, and B. Dony, “A pipelined implementation of OFDM transmission on reconfigurable platforms”, CCECE08 Conference, 2008, pp. 801-804.

[104] J. Garcia and R. Cumplido, “On the design of an FPGA-Based OFDM modulator for IEEE 802.11a”, 2nd International Conference on Electrical and Electronics Engineering (ICEEE) and XI Conference on Electrical Engineering (CIE 2005), Mexico, 2005, pp. 114-117.

[105] R. Asghar, and D. Liu, “2D realization of WiMAX channel interleaver for efficient hardware implementation” in Proc. World Academy of Sc. Engineering and Technology, vol 51., Hong Kong, 2009, pp. 25-29.

[106] A. A. Khater, M.M. Khairy and S. E.-D. Habib, “Efficient FPGA Implementation for the IEEE 802.16e Interleaver”, International Conference on Microelectronics, Morocco, 2009, pp. 181-184.

[107] J. Garcia and R. Cumplido, “On the design of an FPGA-Based OFDM modulator for IEEE 802.16-2004”, Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2005), 2005.

[108] M. Ahmadi, A. Azarpeyvand and S. M. Fakhraie, “Hardware Implementation of Bit Interleaver for the IEEE 802.22 Standard”, 20th Iranian Conference on Electrical Engineering (ICEE), 2012, pp. 1228-1231.

[109] M. N. Khan and S. Ghauri. “The WiMAX 802.16e Physical Layer Model”, IET International Conference in Wireless, Mobile and Multimedia Networks, 2008, pp. 117 –120.

[110] Y.–N. Chang, “A low-cost dual-mode deinterleaver design”, IEEE Trans. Cons. Electr., vol. 54, no. 2, 2008, pp. 326-332.

[111] Z. Iqbal, S. Nooshabadi and H.–N. Loo, “Analysis and design of coding and interleaving in a MIMO-OFDM system”, IEEE Trans. Cons. Electr., vol. 58, no. 3, 2012, pp. 758-766.

[112] Z. Iqbal and S. Nooshabadi, “Effects of channel coding and interleaving in MIMO-OFDM systems,” IEEE Int. Midwest Symp. on Circuit and Syst., Seoul, 2011, pp. 1-4.

[113] J. Yang, H. Li, Q. Han and W. Li, "The Design and Hardware Implementation of OFDM System Receiver based on FPGA", International Conference on Chemical, Material and Food Engineering (CMFE-2015), 2015, Yunnan, China, pp. 819-821.

[114] N.K. Venkatachalam, L. Gopalakrishnan and M. Sellathurai, "Low complexity and area efficient reconfigurable multimode interleaver address generator for multistandard radios", IET Computers & Digital Techniques, available online September, 2015.

[115] B. K. Upadhyaya and S. K. Sanyal, "Efficient FPGA Implementation of Address Generator for WiMAX De-interleaver", IEEE Transactions on Circuits and Systems – II, vol. 60, issue 8, USA, August, 2013, pp. 492-496.

[116] R. Eickhoff, K. Tittelbach-Helmrich, M. Wickert, *et. al* "Physical layer amendments for MIMO features in 802.11a", Conf. on Future Network & Mobile Summit, 2011, pp.1-8.

[117] H.Setiawan, Y. Nagao, M. Kurosaki and H. Ochi, "IEEE 802.11n Physical Layer Implementation on Field Programmable Gate Array", TELKOMNIKA Indonesian J. Electri. Engg., 2011, vol. 10, no. 1, pp. 67-74.

[118] D. Perels, S. Haene, P. Luethi, *et. al* "ASIC implementation of a MIMO-OFDM transceiver for 192 mbps WLANs", 31st European Solid State Conf., France, 2005, pp. 215-218.

[119] T.H. Tran, Y. Nagao, M. Kurosaki, B. Sai and H. Ochi "ASIC design of 600 Mbps 4 × 4 MIMO wireless LAN system" 14th IEEE Int. Conf. on Advan. Commu. Tech., PyeongChang, 2012, pp. 360–363.

[120] Z.-D. Zhang, B. Wu, Y.-X. Zhu, and Y.-M. Zhou, "Design and implementation of a multi-mode interleaver/deinterleaver for MIMO OFDM systems," in Proceedings of the 8th IEEE International Conference on ASIC (ASICON '09), Changsha, China, pp. 513–516.

[121] R. Asghar and D. Liu, "Low complexity hardware interleaver for MIMO-OFDM based wireless LAN," in Proc. 2009 Intl. IEEE Symp. on Circuits and Systems (ISCAS), pp.1747-1750.

[122] Z. Zhang B. Wu, Y. Zhou and X. Zhang, "Low-Complexity Hardware Interleaver / Deinterleaver for IEEE 802.11a/g/n WLAN" *VLSI Design, Hindwai Publisher*, 2012, Article ID 948957, pp. 1-7.

[123] J.-H. Kim and I.-C. Park, "A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE," in Proc. CICC, San Jose, CA, USA, Sep. 2009, pp. 487–490.

[124] Y. Sun, J. R. Cavallaro, "Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decoder", Integration, the VLSI Journal, Elsevier, 2011, pp. 305-315.

[125] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE", IEEE Journal of Solid-State Circuits, vol. 46, no. 1, Jan, 2011, pp. 1-10.

[126] G. Wang, H. Shen, Y. Sun, J. R. Cavallaro, A. Vosoughi, and Y. Guo, "Parallel Interleaver Design for a High Throughput HSPA /LTE Multi-Standard Turbo Decoder", IEEE Transactions on Circuits and Systems—I: Regular Papers, vol. 61, no. 5, May 2014, pp. 1376-1389.

[127] H Nakase, H Oguma and S Kameda, "Improvement of bit error rate using channel interleaving for channel binding WLAN prototype" IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, 2008, Canes, 2008, pp. 1-5.

[128] S. J. Chapman, "MATLAB Programming for Engineers", Thomson Publisher, 2012.

[129] J. Bhaskar, "A VHDL Primer", Pearson Education, New Delhi, 2003

[130] D. G. Bailey, "Design for Embedded Image Processing on FPGAs", Wiley-Blackwell Publisher, Singapore, 2011.

[131] R. S. Sandige, "Modern Digital Design" New York, McGraw-Hill, 1990

[132] M. Ferdjallah, "Introduction to Digital Systems: Modeling, Synthesis, and Simulation Using VHDL", John Wiley & Sons, USA, 2011

[133] G. Richard and B. Slavek, "VHDL Manual", University of Ulm, Department of Microelectronics, 1998.

[134] Clive "Max" Maxfield, "Design Warrior's Guide to FPGAs: Devices Tools and Flows", Elsevier Publication, Oxford, U. K., 2004.

[135] V. Betz and J. Rose, "How Much Logic Should Go in an FPGA Logic Block?," IEEE Design & Test of Computers, January-March 1998, pp. 10-15.

[136] J. S. Rose and S. Brown, "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays," IEEE J. Solid-State Circuits, vol. 26, no. 3, Mar. 1991, pp. 277-282.

[137] S. Brown, J. Rose, and Z. G. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays," IEEE Trans. on Computer-Aided Design, vol. 11, no. 5, May 1992, pp. 620-628.

[138] S. M. Kang and Y. Leblebici, "CMOS Digital Integrated Circuits, Analysis and Design", 3/e, Tata McGraw-Hill, New Delhi, 2003.

[139] S. Trimberger, "Effects of FPGA Architecture on FPGA Routing", IEEE Conf. on Design Automation, 1995, pp. 574-578.

[140] G. D. Forney, "Burst-Correcting Codes for the Classic Bursty Channel", IEEE Transaction on Communication Technology, vol. COM-19, 1971, pp. 772-781.

[141] Final Draft ETSI EN 300 401 V1.4.1 (2006-01), Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers.

[142] J. L. Ramsey, "Realization of Optimum Interleavers", IEEE Transaction on Information Theory, vol. IT-16, no. 3, 1970, pp. 338-345.

[143] Xilinx, "Using Look-Up Tables as Shift Registers (SRL16) in Spartan-3 Generation FPGA-XAPP 465", 2005 available at www.xilinx.com.

[144] Vi Microsystems, "Xilinx Spartan 3 FPGA Trainer (VVSM-07) User Manual", Version –1.0, Chennai, India, 2004.

[145] W. Konhauser, "Broadband wireless access solutions -progressive challenges and potential value of next generation mobile networks", Wireless Personal Communications, vol. 37, May 2006, pp.243–259.

[146] Y. S. Iskander, "Improved abstractions and turnaround time for FPGA design validation and debug" Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 2012.

[147] IEEE std. 802.11g-2003, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band" July 2003.

[148] J. D. Gibson, "The Communications Handbook", CRC Press, 2002

[149] S. Brown and J. Rose, "FPGA and CPLD Architectures: A Tutorial", IEEE Design & Test of Computers, Summer 1996, pp. 42-57.

[150] Xilinx, "Using Block RAM in Spartan 3 FPGAs," XAPP463, available at www.xilinx.com.

[151] H. Yagoobi, “Scalable OFDMA Physical Layer in IEEE 802.16 WirelessMAN”, Intel Technology Journal, vol 08, August 2004.

[152] R. Johannesson and K. S. Zigangirov, “Fundamentals of Convolutional Coding” IEEE Publication, 2015.

[153] S. B. Wicker and V. K. Bhargava, “Reed-Solomon Codes and Their Applications” IEEE Press, 1994.

[154] J G. Andrews , A. Ghosh and R. Muhamed, “Fundamentals of WiMAX: Understanding Broadband Wireless Networking” (Prentice Hall Communications Engineering and Emerging Technologies Series), Prentice Hall PTR, Upper Saddle River, NJ, 2007.

[155] R. Asghar, and D. Liu, “Low complexity multimode interleaver core for WiMAX with support for convolutional interleaving”, International Journal of Electronics, Communication and Computer Engineering, vol.1, no.1 Paris, 2009, pp. 20-29.

[156] I. Kuon and J. Rose, “Measuring the gap between FPGAs and ASICs,” in Proc. of Int. Symposium on Field Programmable Gate Arrays (FPGA ’06), Monterey, California, ACM Press, New York, Feb. 22–24, 2006, pp. 21–30.

[157] B. K. Upadhyaya, and S. K. Sanyal, “An improved LUT based reconfigurable multimode interleaver for WLAN application,” Int. J. Recent Trends in Engneering and Tech., ACEEE, vol. 6, no. 2, 2011, pp. 183-188.

[158] Y. N. Chang and Y. C. Ding: “A Low-Cost Dual Mode De-interleaver Design,” in Proc of Int. Conf. on Consumer Electronics, 2007.

[159] B. K. Upadhyaya, I. S. Misra, and S. K. Sanyal, “Novel design of address generator for WiMAX multimode interleaver using FPGA based finite state machine,” in Proc. of 13th Int. Conf. Computer and Information Technology*,* Dhaka, 2010, pp. 153-158.

[160] H. Yang, “A road to future broadband wireless access: MIMO-OFDM-based air interface”, IEEE Comm. Mag., 2005, vol. 43, no. 1, pp. 53-60.

[161] T. K. Paul and T. Ogunfunmi, “Wireless LAN comes of age: understanding the IEEE 802.11n amendment” IEEE Circuit and Sys. Mag., 2008, vol. 8, no. 1, pp. 28-54.

[162] H. Niu, X. Ouyang and C. Ngo, “Interleaver design for MIMO OFDM based wireless LAN”, IEEE Wireless Comm. and Net. Conf., vol. 4, USA, 2006, pp. 1825-1829.

[163] R. Asghar, “Flexible interleaving subsystems for FEC in baseband processors”, PhD thesis, Linköping University, Sweden, 2010.

[164] R. Van Nee, V. K. Jones, G. Awater, et al. “The 802.11n MIMO-OFDM standard for Wireless LAN and beyond”, Int. J. of Wireless Personal Communications, 2006, vol. 37, pp. 445-453.

[165] B. K. Upadhyaya, P. K. Goswami and S. K. Sanyal, “Memory efficient LUT based address generator for OFDM-WiMAX de-interleaver”, Int. J. of Electronics and Electrical Engineering, vol. 2, no. 1, USA, 2014, pp. 31-35.

[166] Agilent, “Introducing LTE Advanced - Application Note”, available in http://cp.literature.agilent.com/litweb/pdf/5990-6706EN.pdf

[167] Qualcomm, “LTE Advanced: An evolution built for the long-haul”, October, 2013, available in https://www.qualcomm.com/documents/lte-advanced-evolution-built-long-haul

[168] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo Codes”, IEEE Proceedings of the Int. Conf. on Communications, Geneva, Switzerland, May 1993, pp. 1064-1070.

[169] J. T. Butler and T. Sasao, “Fast hardware computation of x mod z,”18th Reconfigurable Architectures Workshop (RAW 2011), May 16-17, 2011, Anchorage, Alaska, USA.

[170] S. G. Lee, C. H. Lee, C. H. Wang and W. H. Sheen, “Architecture Design of QPP Interleaver for Parallel Turbo Decoding”, Vehicular Technology Conference (VTC 2010- Spring), 2010, pp. 1-5.

[171] O. Y. Takeshita, “On maximum contention-free interleavers and permutation polynomials over integer rings,” IEEE Trans. Inf. Theory, vol. 52, no. 3, Mar. 2006, pp. 1249–1253.

[172] C. C. Wong and H. C. Chang, “Reconfigurable Turbo Decoder with Parallel Architecture for 3GPP LTE System” IEEE Transactions on Circuit and Systems-II: Express Briefs, vol. 57, no.7 July, 2010

[173] Technical Specification Group Radio Access Network, the 3rd Generation Partnership Project (3GPP), available at http://www.3gpp.org.