Studies on Development of Differential Evolution based Spectrum Allocation Algorithms and Field Programmable Gate Array Implementation for Cognitive Radio Networks

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

DOCTOR OF PHILOSOPHY
in
Electronics Science

By
Kiran Kumar A.
(10PHPE01)

Centre for Advanced Studies in Electronics Science and Technology
School of Physics
University of Hyderabad
Hyderabad-500 046
India

November - 2015

Centre for Advanced Studies in Electronics Science and Technology
School of Physics
University of Hyderabad
Hyderabad, India

DECLARATION

I Mr. A. Kiran Kumar hereby declare that this thesis entitled "Studies on Development of Differential Evolution based Spectrum Allocation Algorithms and Field Programmable Gate Array Implementation for Cognitive Radio Networks" submitted by me under the guidance and supervision of Dr.Samrat L. Sabat is a bonafide research work which is also free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.
A report on plagiarism statistics from the University Librarian is enclosed.

A. Kiran Kumar,
Ph.D (Electronics Science)
Reg. No: 10PHPE01

Dr. Samrat L. Sabat,
Associate Professor,
CASEST,
School of Physics,
University of Hyderabad.

Centre for Advanced Studies in Electronics Science and Technology
School of Physics
University of Hyderabad
Hyderabad, India

CERTIFICATE

This is to certify that the work described in this thesis entitled "Studies on Development of Differential Evolution based Spectrum Allocation Algorithms and Field Programmable Gate Array Implementation for Cognitive Radio Networks" has been carried out by A. Kiran Kumar bearing the Reg. No. 10PHPE01 under my direct supervision and this has not been submitted for any degree or diploma at this or any other university.

Dr. Samrat L. Sabat,
Associate Professor,
CASEST,
School of Physics,
University of Hyderabad.

Dean,
School of Physics,
University of Hyderabad.

Head, CASEST,
School of Physics,
University of Hyderabad.

To

My family

Acknowledgements

I would like to express my sincere gratitude to my research supervisor Dr. Samrat L. Sabat for his constant support, encouragement, introducing me to the exciting research topic "Cognitive Radio", and for giving me the opportunity to work. He guided me with his valuable suggestions and took keen personal interest throughout the progress of my research work.

I take this opportunity to thank the Head of Centre for Advanced Studies in Electronics Science and Technology, Prof. G. Rajaram, Prof Siba. K. Udgata, Prof. M. Ghanashyam Krishna, Prof. K.C. James Raju and Dr. S.V.S. Nageswara Rao for their valuable suggestions during the research work. I wish to thank the Dean of School of Physics, for providing all the necessary facilities to carry out my work. I would also like to thank University Grants Commission (UGC), Government of India for providing the financial support during my research period.

I wish to thank Abraham for his help in all my administration works. I would like to thank P.Ravibabu, A. Subbarami Reddy, Bansilal and other technical and non-teaching staff members of the School of Physics for their help and co-operation, during my research tenure. I am also thankful to my friends Rangababu, Shravan, Narasimhappa, Sivaram, Bharadwaj, Deepak Tosh, S.Srinu, J.Anusha for their co-operation and suggestions in my research. I would also like to thank Xilinx for providing the tools under the XUP Program.

I wish to express my sincere gratitude to my parents for their unconditional support and encouragement throughout my life. I would like to thank everyone who has helped me knowingly or unknowingly during my whole research work.

Kiran Kumar Anumandla

Abstract

Recent trends in wireless communication technologies claim a rapid increase in demand of radio spectrum. In the current spectrum allocation scheme, it is difficult to accommodate the demand of radio spectrum. Moreover the designated spectrum are not efficiently exploited, resulting its poor utilization. Studies have demonstrated that reuse of the un-utilized spectrum provides a significant improvement in network capacity. Recently, a new dynamic spectrum access paradigm called Cognitive Radio (CR) has gained popularity to solve the shortcomings of spectrum under-utilization and spectrum scarcity. In CR technology, unlicensed users (secondary users) make use of the unused spectrum of licensed users (primary users), thereby discovering a new capacity and commercial value from the existing unused spectrum. The main functions of the CR are spectrum sensing, spectrum management, spectrum mobility and spectrum sharing. Spectrum sensing deals with the detection of vacant spectrum bands known as spectrum holes and these detected holes are assigned to the secondary users (SUs) during spectrum management phase. It uses different spectrum allocation (SA) algorithms for allocating spectrum to SUs. The present thesis mainly concentrates on spectrum allocation phase. The objectives of SA phase are a) maximize the spectrum utilization, b) minimize interference to primary users (PUs) and neighbor secondary users and c) maintain fairness across the users.

To achieve these goals, an efficient SA technique is required for making decisions within a stipulated time. For this purpose, various techniques like graph coloring, game theory, evolutionary algorithms, local bargaining, auction and pricing mechanisms and stochastic search methods have been reported in the literature. The problem of allocating channels amongst the secondary users in the network is considered as a NP-hard problem. In this work, evolutionary algorithms, namely Differential Evolution (DE), firefly and particle swarm intelligence are applied to find an efficient channel assignment solution. Further, the performance of three algorithms in terms of quality of solution and time complexity are compared to find the best solution.

In the distributed approach, each SU device has an embedded platform to perform the SA task. The allocation process needs to be executed in a short time after obtaining request from secondary users or preemption of the channel by the primary user. Otherwise, it degrades the quality of service and interrupt the communication. Thus, it is required to implement the SA algorithm on hardware platform like microcontroller, digital signal processor, field programmable gate array (FPGA) or application specific integrated circuit (ASIC) to improve the execution speed for performing the spectrum allocation. In this work, hardware Intellectual Property (IP) of DE based SA is developed targeting to a FPGA platform. The DE-based SA IP is developed as a coprocessor, and it is interfaced to PowerPC 440 embedded processor via Auxiliary Process Unit controller.

The spectrum allocation needs to simultaneously optimize multiple objectives like maximize network utilization, maximize fairness across the users and minimize forced termination probability by satisfying the constraints posed by primary and secondary users. Hence in this work, Multi-Objective Differential Evolution (MODE) algorithm is used to optimize the network utility functions simultaneously to provide best channel to the secondary users. In literature, forced termination probability is used to analyze the performance of spectrum allocation technique. In the present work, forced termination probability is formulated as an objective function and optimized along with the network utilization functions to find best channels for secondary users. To increase the efficiency of SA technique, a joint spectrum and power allocation algorithm is used to maximize the total network utilization and sum capacity of a user by satisfying the interference and power constraints imposed by both PUs and SUs. Further, a MODE-based SA IP is developed and implemented on FPGA platform to improve the execution speed of the MODE-based spectrum allocation. The network utility functions are integrated with the MODE IP and developed as a MODE-based SA coprocessor to solve the SA problem in cognitive radio.

Contents

List of Figures

List of Tables

Abbreviations

ABC - Artificial Bee Colony
ACO - Ant Colony Optimization
ACS - Ant Colony System
ADE - Adaptive Differential Evolutionary
AF - Acceleration Factor
ASP - Advanced Simple Profile
API - Application Programming Interface
APU - Auxiliary Processor Unit
ASIC - Application-Specific Integrated Circuit
ASSP - Application-Specific Standard Part
BFA - Binary Firefly Algorithm
BRAM - Block Random Access Memory
BS - Base Station
BSB - Base System Builder
BUFG - Global Buffer
CA - Channel Assignment
CD - Compact Disk
CDMA - Central Direct Memory Access
CLB - Combinational Logic Block
CPEs - Consumer Premise Equipments
CPLD - Complex Programmable Logical Devices
CPU - Central Processing Unit
CR - Cognitive Radio
CRC - Cyclic Redundancy Check
CRN - Cognitive Radio Network
CSGC - Color Sensitive Graph Coloring
CU - Cognitive User
DCM - Digital Clock Manager
DCR - Device Control Register
DDR2 - Double Density RAM
DDR-SRAM- Double Data Rate Static Random Access Memory

DE - Differential Evolution
DE-SA - Differential Evolution based Spectrum Allocation
DMA - Direct Memory Access
DEMUX - Demultiplexer
DRAM - Dynamic Random Access Memory
DSP - Digital Signal Processing
EA - Evolutionary Algorithm
ECMA - European Computer Manufacturers Association
EDK - Embedded Development Kit
EPROM - Erasable Programmable Read-Only Memory
ETSI - European Telecommunications Standards Institute
FA - Firefly Algorithm
FCB - Fabric Coprocessor Bus
FCC - Federal Communication Commission
FCM - Fabric Coprocessor Module
FF - Flip Flop
FFT - Fast Fourier Transform
FIFO - First In-First-Out
FIR - Finite Impulse Response
FP - Floaing Point
FPGA - Field Programmable Gate Array
FPU - Floating Point Unit
FSL - Fast Simple Link
FSM - Finite State Machine
FT - Forced Termination Probability
GA - Genetic Algorithm
GCP - Graph Coloring Problem
GE - Genetic Evolution
GPIO - General Purpose Input and Output
GPP - General Purpose Processor
GSM - Global System for Mobile Communications
HDL - Hardware Description Language
HIL - Hardware-in-the-Loop
HMCR - Harmony Memory Considering Rate
HS - Harmonic Search

HW - Hardware
I/O - Input-Output
IBM - International Business Machines
IC - Integrated Circuit
IDCT - Inverse Discrete Cosine Transform
IDE - Integrated Development Environment
IEEE - Institute of Electrical and Electronics Engineering
ILP - Integer Linear Programming
IOB - Input/output Block
IP - Intellectual Property
IPIF - Intellectual Peripheral Interface
IPIC - Intellectual Peripheral Interconnect
ISE - Integrated Software Environment
ISM - Industrial, Scientific and Medical
ISO - International Standard Organization
ITU - International Telecommunication Union
JTAG - Joint Test Action Group
LMB - Local Memory Block
LFSR - Linear Feedback Shift Register
LUT - Look Up Table
MAC - Medium Access Control
MATLAB - Matrix Laboratory
MB - MicroBlaze
μC - Micro-Controllers
MCI - Memory Controller Interface
MHz - Mega Hertz
MINLP - Mixed Integer Non-Linear Programming
MMR - Max-Min-Reward
MODE - Multi-Objective Differential Evolution
MODE-SA - Multi-Objective Differential Evolution based Spectrum Allocation
MOGA - Multi-Objective Genetic Algorithm
MPF - Max-Proportional-Fair
MPMC - Multi Port Memory Controller
MSR - Max-Sum-Reward
MUX - Multiplexer

NP - Neyman-Pearson
NSGA-II - Nondominated Sorting Genetic Algorithm II
OFDM - Orthogonal Frequency Division Multiplexing
OSI - Open Systems Interconnection
OTS - Off-the-Shelf
PAR - Pitch Adjustment Rate
PC - Personal Computer
PDA - Personal Digital Assistant
PHY - Physical Layer
PLB - Processor Local Bus
PLL_ADV- Phase Locked Loop
PPC440 - PowerPC440
PPSO - Pipelined PSO
pPSO - Parallel PSO
PSO - Particle Swarm Optimization
PSoC - Programmable System on Chip
PU - Primary User or licensed user
QoS - Quality of Service
QGA - Quantum Genetic Algorithm
RAM - Random Access Memory
RF - Radio Frequency
RISC - Reduced Instruction Set Computer
RNG - Random Number Generation
RRS - Reconfigurable Radio Systems
SA - Spectrum Allocation
SCC - Standards Coordinating Committee
SDK - Software Development Kit
SDRAM - Single Data Rate RAM
SDR - Software Defined Radio
SINR - Signal-to-Interference-plus-Noise Ratio
SIR - Signal-to-Interference Ratio
SNIR - Signal-to-Noise-plus-Interference Ratio
SoC - System on Chip
SoPC - System on Programmable Chip
SNR - Signal to Noise Ratio

SRAM - Static Random Access Memory
STD - Standard Deviation
SU - Secondary User or Unlicensed user or Cognitive Radio
SW - Software
TVWS - TeleVision White Space
UART - Universal Synchronous and Asynchronous Receive and Transmit
UDI - User Defined Instructions
UMTS - Universal Mobile Telecommunications System
USB - Universal Serial Bus
USRP - Universal Software Radio Peripheral
VLSI - Very Large Scale Integration
VHDL - VHSIC Hardware Description Language
VoIP - Voice Over Internet Protocol
WARP - Wireless Open Access Research Platform
WG - Working Group
WiMax - Worldwide interoperability for Microwave access
WiFi - Wireless Fidelity
WLAN - Wireless Local Area Network
WRAN - Wireless Regional Area Network
XPS - Xilinx Platform Studio
XST - Xilinx Synthesis Tool
ZDT1 - Zitzler-Deb-Thiele 1
ZDT2 - Zitzler-Deb-Thiele 2
2G - Second-Generation
3G - Third-Generation

LIST OF INTERNATIONAL JOURNALS PUBLISHED

1. **Kiran Kumar Anumandla,** Rangababu Peesapati, Samrat L. Sabat, Siba K. Udgata and Ajith Abraham, FPGA based Differential Evolution Coprocessor: A case study of spectrum allocation in cognitive radio network, **IET Computer and Digital Techniques**, 7(5), pp 221-234, 2013**.**

2. **Kiran Kumar Anumandla**, Rangababu Peesapati, Samrat L. Sabat and Siba K. Udgata, SoC based floating point implementation of Differential Evolution Algorithm using FPGA, **Design Automation and Embedded Systems, Springer**, 16(4), pp 221-240, 2012**.**

3. **Kiran Kumar Anumandla,** Rangababu Peesapati, Samrat L. Sabat, Field Programmable Gate Array implementation of Spectrum Allocation technique for Cognitive Radio Networks, **Computers & Electrical Engineering (Elsevier)**, 42(0), pp 178-192, 2015**.**

4. Rangababu Peesapati, **Kiran Kumar Anumandla,** Samrat L. Sabat, Comparative study of system on chip based solution for floating and fixed point differential evolution algorithm, **Swarm and Evolutionary Computation (Elsevier)**, 19(0), pp 68-81,2014.

LIST OF INTERNATIONAL CONFERENCE PROCEEDINGS

- **Kiran Kumar Anumandla,** Shravan Kudikala, Bharadwaj Akella Venkata, Samrat L. Sabat, "Spectrum Allocation in Cognitive Radio Networks Using Firefly Algorithm, In Proceedings of International conference on Swarm, Evolutionary and Memetic Computing (SEMCCO), 2013, pp 366-376.

- **Kiran Kumar Anumandla**, Bharadwaj Akella Venkata, Samrat L. Sabat and Siba K.Udgata, "Spectrum Allocation in Cognitive Radio Networks Using Multi-Objective Differential Evolution Algorithm, In Proceedings of IEEE 2nd International conference on Signal Processing and Integrated Networks (SPIN), 2015, pp 273-278.

Chapter 1

Introduction

There is a limited availability of radio spectrum resources for future services to users, despite the fact that the licensed spectrum is under-utilized for long periods of time. At present, the radio spectrum is divided into spectrum bands and these have been assigned to various services like mobile, broadcast, satellite services regulated by International Telecommunication Union (ITU). Particularly, the mobile telecommunication market predicted that system development and consequent spectrum allocations will grow significantly in the time span 2010 - 2020 in terms of aggregate data rate per user (ITU-R 2006). Figure 1.1 illustrates the average utilization of spectrum over seven locations in U.S in the frequency range of 30MHz - 2.9GHz. It is observed that the maximum spectrum occupancy is in the range of 5.2% to 13.1% at some frequency bands [1, 5]. Figure 1.2 shows that significant part of the spectrum is under-utilized. Hence, it depicts that the spectrum scarcity problem is not due to the lack of radio spectrum resources, but due to the inefficient spectrum assignment policies. The reports published by Federal Communications Commission (FCC) stated that fixed spectrum allocation policies in wireless applications accompanied to poor utilization of spectrum resources in time, frequency and geographical space dimensions [6].

To solve the spectrum scarcity problem, Mitola has introduced a new technology paradigm called Cognitive Radio (CR) [7]. It has the capability to sense the RF environment and intelligently find an appropriate spectrum for seamless communication. Thus, it opportunistically and adaptively uses the under-utilized frequency bands in different parts of the spectrum as shown in Figure 1.3. It maximizes the cognitive user capacity without disturbing the licensed users. Thus, a dynamic spectrum allocation technique is essential to improve the spectrum utilization. In CR network, two types of users are present a) licensed users also known as primary users (PUs) and b) unlicensed users also known as secondary users (SUs) or CR users. In CR cycle, the first phase is spectrum sensing. In this phase, the spectrum bands are scanned to obtain the information about occu-

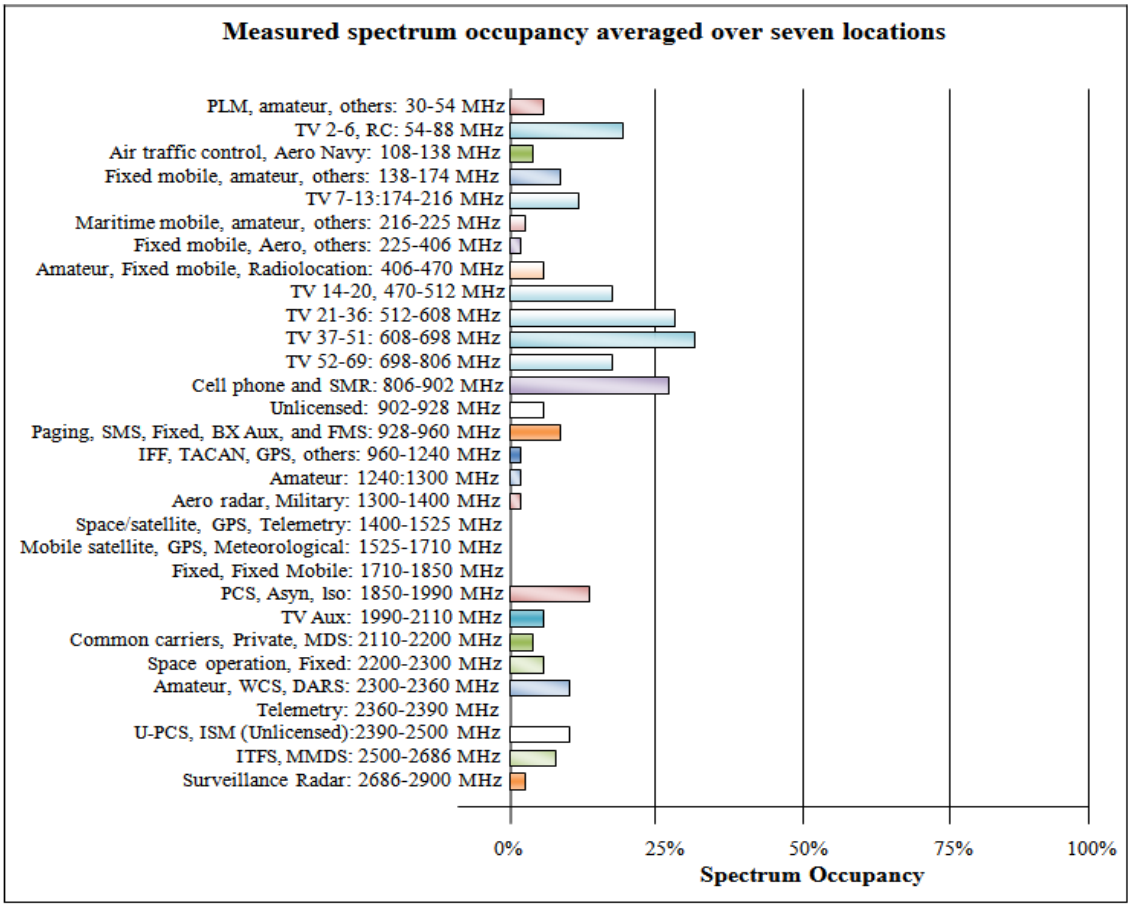

Figure 1.1: Radio Spectrum occupancy averaged over seven locations [1]

pancy in the channel. In the next phase, available channels are allocated to SUs such that the communication do not cause interference to the existing users. This is known as spectrum allocation (SA) [4]. In wireless network, interference may occur due to either environment noise or other neighborhood users. Controlling the interference is a critical task and it affects channel capacity and performance of the wireless network. The spectrum available for secondary user varies with both location and time due to mobility and traffic variations of primary user. Spectrum occupied by a cognitive user without coordinating with neighbors can cause interference to other users.

In CR network, the SA problem is more complex compared to the conventional wireless network, due to dynamic variation of spectrum holes and available frequencies with respect to both time and location. The objective of SA is to allocate a best channel (among the available) to each CR user such that the network utilization is maximum by maintaining fairness across the users subject to satisfying

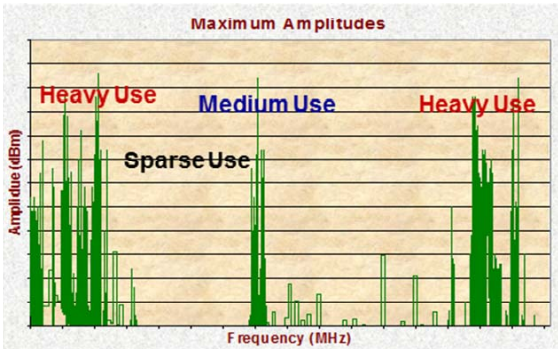

Figure 1.2: Utilization of radio spectrum [2]

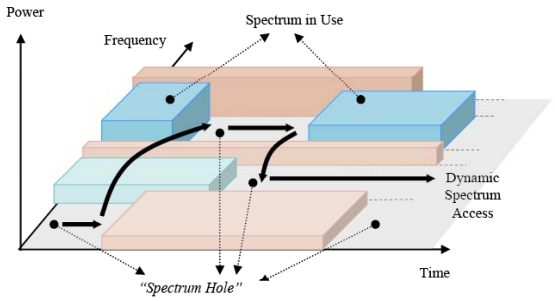

Figure 1.3: Spectrum hole and dynamic spectrum access [3]

a set of constraints imposed by both PUs and SUs. The constraints are number of channels selected for each user, data rate, transmitted power and interference to PUs. In CR network, interference can also be minimized by optimizing transmitted power allocated to each SU. Hence, a joint spectrum and power allocation technique is required to increase the network efficiency (i.e., maximum network utilization with minimum interference). An efficient SA technique will provide maximum network utilization, minimum interference, maximum data rate, maximum fairness among the users and optimal power to SUs. Optimal power to SUs ensures maximum battery life. Thus, development of efficient algorithm for SA is an important area of research.

In a distributed network, each SU uses a distributed algorithm for determining spectrum for its own communication. In this network, each SU considers the locally available information from the neighborhood users and decides its spectrum. In literature, the SA problem is solved by different techniques using high end computing platform to maximize the network utilization. However in real time, each SU implicitly have an embedded computing platform and the SA task has to be performed on it. Hence, development of architecture and hardware implementation of SA algorithm is also an important area of research.

1.1 Motivation

The spectrum allocation in CR network is a NP-hard problem and the algorithmic time complexity to solve SA exponentially increases with CR network parameters like number of primary, secondary users and number of channels [8]. Recent research has focused on solving the SA problem using evolutionary algorithms

for obtaining an optimal solution [9], [10], [11]. In recent past, a service based spectrum allocation model was solved using graph coloring method and enhanced Particle Swarm Optimization (PSO) algorithm to satisfy user demand and fairness reward [12]. It was reported that the enhanced PSO achieves better reward and efficiency compared to graph coloring method. Although the above mentioned algorithms solved the SA problem, but the performance of these algorithms were not compared among evolutionary algorithms under different network conditions like by varying the number of users and channels. Thus in this thesis, the SA problem is solved using PSO, Firefly and Differential Evolution (DE) algorithms. It also studies the effect of varying number of SUs, PUs and channels on network utilization. The performance of these algorithms is compared in terms of quality of solution and time complexity.

The SA algorithm needs to simultaneously optimize multiple objectives (like maximizing network utilization [13] and minimizing forced terminations) for finding optimal channel assignment. Thus in this thesis, the SA problem is formulated as a multi-objective optimization problem where network utility functions and forced termination probability [14],[15] are optimized simultaneously using Multi-Objective Differential Evolution (MODE) and Non-dominated Sorting Genetic Algorithm II (NSGA-II) algorithms. Power allocation to SUs also plays an important role in minimizing interference to PUs. A joint power and channel allocation algorithm was reported to maximize CR network capacity by considering Signal to Interference plus Noise Ratio (SINR) constraints posed by PUs [16]. However, in [16] maximization of capacity of an individual user, outage probability and interference constraints imposed by SUs are not considered during SA.

Thus in this thesis, a joint spectrum and power allocation algorithm is used to optimize both network capacity and individual user capacity simultaneously using MODE and NSGA-II algorithms by considering outage probability and interference constraints imposed by PUs and SUs. In a distributed network, each SU has to perform the SA task on its CR device quickly to ensure seamless communication. In literature, a hardware device for channel allocation was proposed for cellular networks to speedup the channel selection and allocation algorithm with respect to current traffic requirement and interference constraints [17]. The hardware device improved the efficiency of allocation time. In literature, there is not much work on accelerating the execution speed of SA algorithm in CR networks to provide seamless communication to users. Thus in this thesis, DE-based SA

(DE-SA) hardware Intellectual Property (IP) is developed to accelerate the execution speed of SA algorithm. This IP is used to optimize the three network utility functions independently. In addition, MODE-based SA (MODE-SA) hardware IP is also developed to optimize the network utility functions simultaneously. The main objective of this research is to study, design, implement and validate a SA technique for Cognitive Radio on a reconfigurable platform.

1.2 Challenges

Spectrum allocation is an important step in CR networks and variety of techniques have been applied to solve the SA problem. However, still there are some open issues in solving the SA problem in a CR network.

(I) Spectrum characterization: Most of the previous works select a channel for a user based on a single criterion (i.e., throughput, SNR or SINR) that represents traffic load of the spectrum. However, multiple criteria are not considered for selecting a channel, although it may give better results. There are different Quality of Service (QoS) criteria such as bit error rate, interference, throughput, jitter, end-to-end delay, transmitted power, etc., that depends on cognitive user application/service. Most of the proposed SA algorithms target to achieve only one QoS criterion, i.e. throughput. It is not applicable to other applications that are sensitive to transmission delay.

(II) Multiple channel selection: The throughput of CR network (CRN) can be increased by the use of multiple channels (contiguous or non-contiguous) or multiple radios. In literature, limited works addressed this and most of the works concentrated on single-radio device. The allocation of multiple channels to multiple radios is a complex problem in terms of algorithm and corresponding hardware design.

(III) Energy efficiency: Minimizing energy consumption of a CR device is the active research in CR technology. In literature, limited works related to the power optimization are proposed to achieve energy efficiency and non-interference to licensed users. It is a challenge to balance the QoS requirements to SUs and non-interference to PUs. The transmission power of CR device, selected channel bandwidth, frequency, link error are leading causes of energy consumption. CR users are considered to be mobile and battery power constrained. Thus, a method

for energy-aware SA techniques on a hardware platform is essential.

(IV) Spectrum Fragments/Channels: In CR technology, there is no particular depiction about the spectrum holes that a CR device can use. A CR user can use those available channels regardless of bandwidth and frequency of the channels. The previous works considered that the available channels are of fixed width with a specific central frequency and find the best channel using traditional channel assignment (CA) techniques. In CRN, there is no restriction about fixed range and width of a channel. The CR users scan the RF spectrum, detect the holes and use them by satisfying the QoS requirements like optimal bandwidth, frequency, and required throughput. Reconfiguring the hardware to attain this is a research issue.

(V) Interference Management: In CR networks, it is mandatory to limit the interference to primary users during communication. It is a crucial challenge to achieve maximum network utilization with minimum interference to other users in a network. In literature, many works considered the interference caused by SUs to PUs. However, it is important to consider the interference caused by each user to other user. Limiting the interference is usually carried out by controlling the SU transmitted power. Hence, it is required to develop a joint framework that addresses the SA and power allocation simultaneously. With this approach, a trade-off solution can be achieved that will satisfy the interference constraints posed by both PUs and SUs.

(VI) Hardware implementation: In distributed approach, each CR device has to execute the SA algorithm independently for assignment of channel to SUs. The execution of complex SA algorithm on embedded platform degrade its primary functionality. Due to dynamic variation of available channels, the SA algorithm should execute fast during call initiation and hand-off process. Hence, it is required to accelerate the execution performance of SA for fast process of allocation of channels to the SUs. Hardware implementation of the SA algorithm can meet the requirements of execution speed and power consumption specification of CR device.

1.3 Research objective

The present work has mainly two objectives: a) to develop an efficient spectrum allocation technique that maximizes network utilization subject to satisfying in-

terference and power constraints imposed by both PUs and SUs b) to implement the SA algorithm on a hardware platform to accelerate the execution speed for finding optimum channel.

1.4 Thesis contribution

In the context of a general SA framework for CR networks, efficient SA algorithm is developed to optimize the network utility functions that assures quality of service (in terms of reward assigned to SU, fairness and non-interference across the users) of each SU.

(1) The SA problem is formulated as a single objective problem and solved using Differential Evolution (DE), PSO and Firefly algorithms to maximize the network utilization subject to interference constraints imposed by SUs. The performance of these algorithms is evaluated in terms of quality of solution (fitness value of network utility functions) and time complexity. From simulation results, it is observed that the DE algorithm improved the quality of solution by 29.9% and 19.04% and the time complexity by 242.32% and 46.3% when compared to PSO and Firefly algorithms respectively.

(2) The SA task is formulated as a multi-objective optimization problem to optimize network utility functions, namely Max-Sum-Reward (MSR), Max-Min-Reward (MMR), Max-Proportional-Fair (MPF) and forced termination probability simultaneously subject to interference constraints using MODE algorithm. The performance of MODE algorithm is compared with NSGA-II for evaluating the optimal solution for fair spectrum assignment to the CR users without interference to PUs.

(3) To enhance the efficiency of SA technique, channel capacity of the individual SUs need to be optimized subject to the limit of interference and PU outage probability. Hence, a joint spectrum and power allocation model is formulated as an optimization problem and solved using DE and PSO algorithms. Further, one of the network utility function (MSR) and channel capacity of individual user are optimized simultaneously using MODE algorithm subject to PU outage probability and power constraints posed by SUs. The performance of MODE algorithm is

compared with NSGA-II algorithm in terms of quality of solution.

(4) A hardware IP of DE-SA algorithm is developed. The IP is interfaced with PowerPC 440 (PPC440) processor of Xilinx Virtex-5 FPGA using Auxiliary Processor Unit (APU) to accelerate the execution speed of SA algorithm. The acceleration of this coprocessor is compared with equivalent floating and fixed point arithmetic implementation of the SA algorithm on PPC440 processor. The implementation results show that the coprocessor accelerates the SA task by 76.79-105x and 5.19-6.91x compared to floating and fixed point implementation of the algorithm on PPC440 processor respectively.

(5) A MODE-SA hardware IP is developed to accelerate the execution speed of MODE-SA algorithm. Further, a FPGA based System on Chip (SoC) solution is developed for solving MODE-SA problem by interfacing the IP to PPC440 processor of Xilinx Virtex 5 FPGA. This IP is interfaced as a coprocessor by connecting to Auxiliary Processor Unit (APU) controller. The hardware solution attained a speedup of 50-60x compared to the time taken by the PPC440 processor to complete the allocation process.

1.5 Thesis organization

This thesis is organized as follows:

Chapter 2 presents a brief introduction to cognitive radio and mainly focused on SA. Different techniques involved in SA are discussed. It also presents a brief introduction to FPGAs, hardware accelerators, Hardware-software co-design, Programmable System-on-Chip design and the tools used in this work. The challenges involved in design and implementation of an efficient SA technique are discussed.

Chapter 3 presents the simulation study of solving SA problem using evolutionary algorithms. In this chapter, the SA problem is solved using PSO, Firefly, and DE algorithms. The algorithm performance is compared in terms of quality of solution and time complexity to solve the SA problem.

Chapter 4 presents the hardware implementation of DE algorithm on PPC440 based System on Chip platform using Virtex-5 FPGA development board. It also presents the performance of DE-IP for solving benchmark test functions and SA

problem. Further, it also presents a detail study of the IP's execution speed along with resource utilization and power analysis.

Chapter 5 presents the study of SA problem as a multi-objective optimization problem by considering network utility functions along with forced termination probability. Multi-Objective Differential Evolution and NSGA-II algorithms are used to solve the SA problem. It also presents the performance comparison of MODE with NSGA-II algorithm to solve the problem. Further, it presents a joint spectrum and power allocation algorithm to maximize the network utilization and average capacity of a user in both uplink and downlink scenarios. This problem is solved using single objective DE and PSO algorithms, MODE and NSGA-II algorithms. A detailed performance study of these algorithms is presented. Finally, it presents the performance of the MODE-SA hardware IP on Virtex-5 FPGA development board to solve the SA problem.

Chapter 6 concludes the thesis by summarizing the research contributions of the thesis. A possible future scope of the work is also presented.

Chapter 2

Background

2.1 Cognitive Radio

The recent rapid development in technological and economical background lead to reshape the design of wireless communication networks. During last decade, the usage of wireless services has increased tremendously. Thus, there is a huge demand for new spectrum to provide more services. On the other hand, large portions of the radio spectrum is under-utilized due to the fixed spectrum assignment policies made by the government and international regulatory bodies [18]. In the early 1990s, the concept of Software Defined Radio (SDR) was introduced by Joseph Mitola [19]. The SDRs typically have a reconfigurable platform in which radio frequency (RF) front end is configured according to the base band signal. Further, Mitola took one step forward and extended it to Cognitive Radio (CR). The CR is basically a SDR capable of intelligent sensing and adaptive to the RF environment. A more common definition of CR: "It is a radio that can change its transmitter parameters (must be reconfigurable) based on interaction with the environment in which it operates (must have sensing or cognitive capabilities). This interaction may involve active negotiation or communications with other spectrum users and/or passive sensing and decision making within the radio" [6].

In CR network, PUs have legacy rights to use the specific part of the spectrum, whereas secondary users termed as Cognitive Users (CUs) have no license to use the spectrum. However, they can access the unused spectrum opportunistically using CR technology. In CR domain, a spectrum hole is defined as a band of frequencies allocated to a PU, but is not being used by it, at an instant of time and specific geographic location. CR has the capability to take the best decision for achieving maximum network utilization, minimum interference to PUs and robust communication to both PUs and SUs. The two main attributes of CR are Cognitive capability and reconfigurability [3].

Cognitive capability: It helps to (i) interact with the surrounding RF environ-

ment using radio transceiver and (ii) capture the communication parameters. The radio decides the spectrum band and type of transmission strategy based on these parameters.

Reconfigurability: It is the capability to reconfigure the communication parameters of cognitive radio transceiver based on dynamic nature of radio environment during the operation. CR is flexible enough to reconfigure the transceiver parameters like operating frequency, modulation-demodulation scheme, transmission power and communication technology to exploit the unused spectrum over a wide range. It is possible to implement the cognitive radio device on a reconfigurable platform like Field Programmable Gate Array (FPGA) to achieve the reconfigurability.

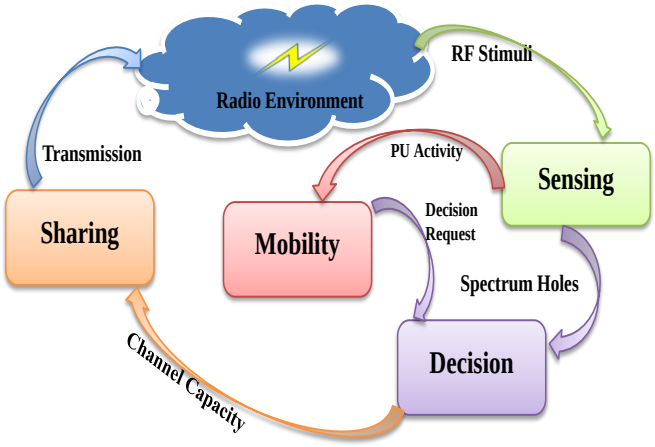

Figure 2.1: Cognitive Radio cycle [2]

The basic working principle of CR can be explained through the cognitive cycle as shown in Figure 2.1 [2, 20, 21]. It consists of four main functionalities namely spectrum sensing, spectrum management, spectrum sharing and spectrum mobility. The fist step is spectrum sensing in which each cognitive user scans the entire available RF spectrum and finds the spectrum holes [20]. This operation is performed in physical (PHY) layer of Open Systems Interconnection (OSI) model. In literature, different techniques have been employed for spectrum sensing [22, 23]. In the spectrum management step, the detected spectrum holes are analyzed/characterized with respect to transmission parameters [2]. In this phase, channel statistical properties like holding time, off time and channel characteristics like bandwidth, carrier frequency, channel error rate and interference level, etc. are extracted. Thereafter, a spectrum decision (spectrum assignment)

operation is performed. During this phase, best channels are allocated to SUs by satisfying the user Quality of Service (QoS) requirements such as throughput, minimum interference, data rate and bandwidth. This operation is performed in Medium Access Control (MAC) layer of OSI model. The detailed discussion about various spectrum allocation techniques are presented in Section 2.3.

In CR network, it is necessary to maintain coordination among the CR users during transmission through the shared wireless channels. Spectrum sharing maintains the QoS of SUs without any interference to PUs while assigning the suitable channels in the dynamic radio environment. It provides flexibility in partitioning the spectrum to the respective SUs such that the interference and collisions among the users are minimum. This function is performed in MAC layer during the communication session. After assignment of a suitable channel to SU, communication process starts. During the communication, the channel occupied by SU may be retained by a PU due to the dynamic nature of RF environment. At this instance, SU vacates the channel and look for another vacant channel. This operation is performed in spectrum mobility phase of the CR [2]. This process is also known as spectrum handover between the bands. It ensures (i) minimum interference to PU transmissions, (ii) seamless communication during switching of SUs between the bands.

2.2 Standardization attempts

It has been proven from reports [6] and measurements [18, 1] that traditional fixed spectrum assignment policy result in inefficient usage of frequency bands. Several international organizations like IEEE, Software Define Radio Forum (SDR forum), International Telecommunication Union (ITU), European Computer Manufacturers Association (ECMA) and European Telecommunications Standards Institute (ETSI) are undertaking regulatory and standardization activities for cognitive radio across the world [24]. In November 2004, the first wireless air interface standard called IEEE 802.22 Wireless Regional Area Network (WRAN) was proposed for wireless networks based on cognitive radio techniques [25]. This standard operates in TV bands (54 - 862 MHz) with the use of spectrum sensing and spectrum management. These TV band signals were sensed at a signal strength of -116dBm (-21dB) with respect to the receiver noise figure of 11dB using an omni-directional antenna. In this system, a central base station controls the Consumer Premise

Equipments (CPEs) in a cell that are associated with the base station. In 2005, the IEEE Electromagnetic Compatibility Society and IEEE Communications Society together initiated the IEEE 1900 Standards Committee, which standardizes many critical issues in the domain of policy defined radio systems, cognitive radio systems and spectrum management [26].

IEEE 1900.1 working group (WG) provided the standard definitions for spectrum management and CR-oriented terms and concepts [27]. IEEE 1900.2 WG recommended the interference and coexistence criteria and established a framework for measuring the interference between radio systems [28]. IEEE 1900.3 WG handles developing test methods for evaluation of software modules of a CR device along with its validation and certification [29]. IEEE 1900.4 defined the system architecture, functionality of terminals and the network devices [30]. The exchange of information between coordinating reconfigurable devices leads to increase the spectrum utilization and quality of service. The promising wireless services achieved by CRs in TV white band lead to various amendments in IEEE standards like IEEE 802.16h [31, 32, 33]. Additionally, the ETSI proposed regulations for Reconfigurable Radio Systems (RRS) based on CR and SDR technologies [34, 35]. These standard activities are being conducted under Standards Coordinating Committee (SCC) 41 [36] focused on dynamic spectrum access networks. New techniques are supposed to manage interference, coordinating wireless technologies, information sharing, and network management. The SCC41 addresses these issues using Software Defined Radio technologies as a key enabler for CR/DSA [37].

2.3 Cognitive Radio Network architecture

A CR network architecture primarily consists of SUs and PUs. An example of CR network architecture is shown in Figure 2.2. PUs have legacy rights to access the existing primary network infrastructures like GSM, TV broadcast, UMTS etc., that are operating at specific frequency bands. All PU operations are not effected by unlicensed users, and these activities are controlled by primary base stations. In CRN, PUs do not have any cognitive functionalities whereas SUs do not have any license to use any frequency bands.

CR networks support two types of network topologies namely: Infrastructure based (centralized) and Infrastructure-less (distributed). In a centralized CR net-

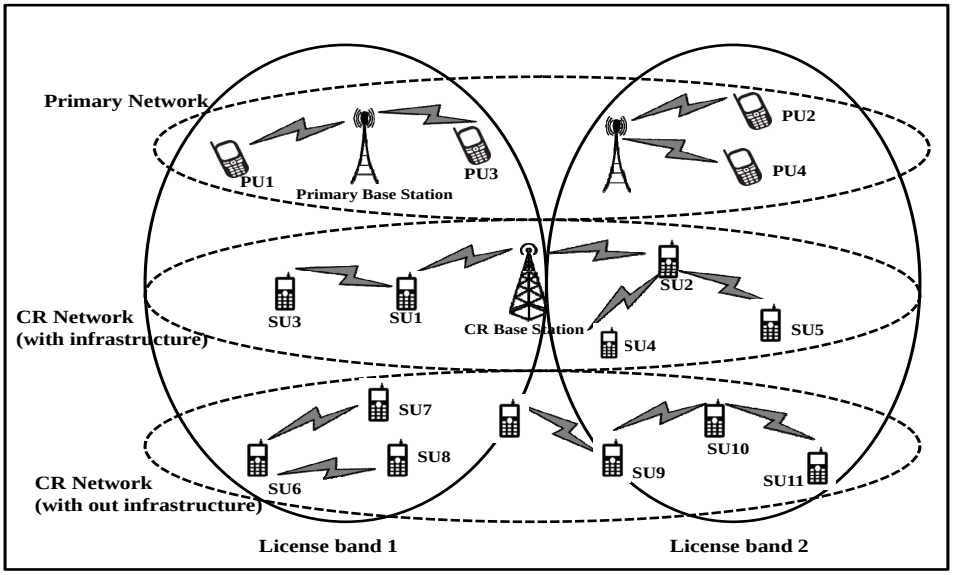

Figure 2.2: Cognitive Radio network architecture

works, the SUs are controlled through a single-hop connection to a central server. The server finds the required vacant channels and assigns the best channels to SUs without interference to PUs. In the distributed topology, SUs communicate with other users without any central server node. Each SU has a CR enabled device to detect the spectrum holes in the current RF environment. After detecting the vacant spectrum, SU selects the best channel such that it satisfies the user requirements and interference constraints. During the communication of a SU over a channel, if the licensed user retains it, then the SU need to vacate the channel and switch to another available channel. In this topology, the primary goal is to assign the best channels to SUs such that it maximizes the network utilization with minimum interference to PUs.

2.4 Spectrum Allocation

Spectrum Allocation (SA) in wireless networks aims to assign vacant channels (white spaces) to SUs, such that it maximizes the network utilization with minimum interference to PUs and SUs during communication [38]. However, maximizing the utilization of network resources and minimizing the level of interference are two conflicting objectives [39, 40]. SA can be significantly affected by the network topology and conflict among wireless links or connectivity between the wireless

devices of a network.

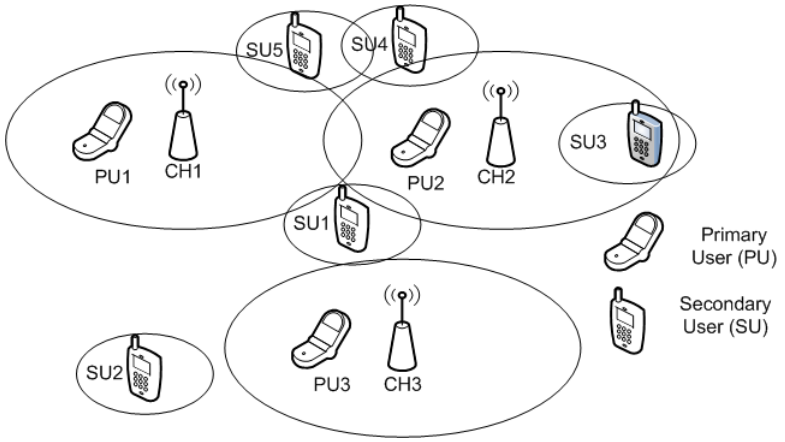

Figure 2.3: Spectrum Allocation model

Spectrum allocation provides fairness across the SUs and efficient utilization of spectrum without any performance degradation to other SUs and PUs in the network. To meet these requirements, an efficient and fast SA technique has been a key focus of research. An example of a typical SA model is shown in Figure 2.3. In the SA model, environment conditions like user position and available channels are considered as static due to slow change in RF environment. On the other hand, users can perform network-wide spectrum allocation operation quickly to adapt changes in the environment. In this section, a SA model of CRN architecture is described. The present model assumes a network of N SUs, M channels and K PUs. The primary user transmission affects not only channels available to SUs but also throughput, transmission power and transmission range of SUs. In Figure 2.3, three broadcast channels ($CH1, CH2$ and $CH3$), correspond to three PUs ($PU1, PU2$ and $PU3$) and five SUs ($SU1$ ... $SU5$). Each PU occupies a channel m with a protection area of radius D_{PR}. No secondary users that cause interference to PUs are allowed to transmit within this protective area. A SU n can change its transmission range $D_{SU}(n, m)$ by controlling its transmitted power on the channel m.

The model considered in this work is same as the model proposed in [8], wherein it is considered that a SU n can utilize the same channel m used by a PU PU_x if $D_{SU}(n, m) \leq Dist(n, PU_x) - D_{PR}$, where $Dist(n, PU_x)$ is the distance between the PU_x and SU n. The maximum and minimum transmission range of the sec-

ondary user is d_{max} and d_{min} respectively. In Figure 2.3, it is shown that primary user $PU1$ uses channel $CH1$. However, $SU1$ cannot use the channel $CH1$, since it is within the protection area of $PU1$. Since $D_{SU}(n,m) < d_{min}$, $SU2$ cannot use the channel m although it is outside the communication range of all primary users. In a CRN, each secondary user can adjust the D_{SU}, that has an impact on its transmission power. An increase in the value of D_{SU} causes an increase in interference probability with the neighboring users. If there is a conflict in the transmission range of any two SUs, then the same channel cannot be used simultaneously. The main goal of the SA algorithm is to provide maximum spectrum utilization and fairness among the users. To achieve these goals, different network utility functions are described that provide a trade-off between the fairness and spectrum utilization.

2.4.1 Problem formulation

In CRN model shown in Figure 2.3, the position of PU and SU are assumed to be static during the allocation process. The channel availability and reward values are evaluated based on the position of the user, channel utilization of PUs and neighbor SUs as specified in the Algorithm 1. The general SA model consists of a channel availability matrix $L = \{l_{n,m} | l_{n,m} \in \{0,1\}\}_{N \times M}$, where $l_{n,m} = 1$ iff channel m is available to user n, else $l_{n,m} = 0$. The channel reward matrix $B = \{b_{n,m}\}_{N \times M}$, where $b_{n,m}$ represents the reward that can be obtained by an user n using the channel m. The interference constraint matrix $C = \{c_{n,p,m} | c_{n,p,m} \in \{0,1\}\}_{N \times N \times M}$ represents the interference among SUs, where $c_{n,p,m} = 1$ if users n and p would interfere with each other if they use the channel m simultaneously and $c_{n,p,m} = 0$ otherwise. However, $c_{n,p,m} = 1 - l_{n,m}$ if $n = p$ [8]. During SA, it is assumed that the user position and available spectrum are static. Thus L, B and C matrix values are constant during the allocation period. The objective of SA is to determine the conflict free spectrum assignment matrix $A = \{a_{n,m} | a_{n,m} \in \{0,1\}\}_{N \times M}$, (where $a_{n,m} = 1$ if channel m is allocated to SU n, and $a_{n,m} = 0$ otherwise) subject to satisfying the interference constraints defined by C:

$$a_{n,m}.a_{p,m} = 0, if c_{n,p,m} = 1, \forall 1 \leq n, p \leq N, 1 \leq m \leq M \tag{2.1}$$

For the given scenario of channel availability matrix (L) and interference constraint matrix (C), the objective of spectrum allocation is to find the optimal conflict

Algorithm 1 : Pseudo-code for Evaluation of Spectrum Allocation Parameters [8]

Step 1: Define the number of SUs N and deploy them randomly. Record the location of each SU n $\forall n = 1...N$ as $\phi_n(x, y)$
Step 2: Define the number of primary users K and deploy them randomly. Record the location of each primary user k, that uses channel z_k $\forall k = 1...K$ as $\alpha_k(x, y)$
Step 3: Define the maximum and minimum transmission range of SU (d_{max} and d_{min}), protection area of PU (D_{PR}).
Step 4:
for $n = 1$ to N do
  for $m = 1$ to M do
    $D_{SU}(n, m) = min(d_{max}, min_{k=1...K, z_k=m} DIST(\phi_n(x, y), \alpha_k(x, y)) - D_{PR})$
    if $D_{SU}(n, m) > d_{min}$ then
      $B_{n,m} = D_{SU}(n, m)^2, l_{n,m} = 1$
    else
      $B_{n,m} = l_{n,m} = 0$
    end if
  end for
end for
Step 5:
for $n = 1$ to $(N - 1)$ do
  for $i = (n + 1)$ to N do
    for $m = 1$ to M do
      if $D_{SU}(n, m) + D_{SU}(i, m) \geq DIST(\phi_n(x, y), \phi_i(x, y))$ then
        $c_{n,i,m} = c_{i,n,m} = 1$
      else
        $c_{n,i,m} = c_{i,n,m} = 0$
      end if
    end for
  end for
end for

free channel assignment matrix A^* that maximizes the network utility defined as $U(A, B)$. For the given L and C matrices, $\Lambda_{(L,C)}$ is considered as the set of interference free channel assignment. Mathematically it can be expressed as:

$$A^* = \underset{A \in \Lambda_{(L,C)}}{argmax}\ U(A, B) \tag{2.2}$$

In this work, three utility functions are considered [9].

1. Max-Sum-Reward (MSR): It maximizes the total spectrum utilization in the

network regardless of fairness. This is defined as:

$$MSR : U(A, B) = \sum_{n=1}^{N} \sum_{m=1}^{M} a_{n,m}.b_{n,m} \tag{2.3}$$

2. Max-Min-Reward (MMR): It maximizes the spectrum utilization of the user with the least allotted spectrum. This is defined as:

$$MMR : U(A, B) = \min_{1 \leq n \leq N} \sum_{m=1}^{M} a_{n,m}.b_{n,m} \tag{2.4}$$

3. Max-Proportional-Fair (MPF): It maximizes the fairness for single-hop flows and the corresponding fairness-driven utility function is defined as:

$$MPF : U(A, B) = \left(\prod_{n=1}^{N} \left(\sum_{m=1}^{M} a_{n,m}.b_{n,m} \right) \right)^{1/N} \tag{2.5}$$

where $a_{n,m}$ and $b_{n,m}$ are the elements of channel assignment matrix (A) and reward matrix (B) respectively. N and M corresponds to the number of SUs and channels in the network.

2.5 Spectrum Allocation techniques

In literature, SA problem is solved based on cooperative and non-cooperative allocation behavior, underlay, and overlay spectrum access technique and distributed and centralized architecture, using techniques like graph theory, game theory, linear programming, local bargaining, heuristics, pricing and auction, fuzzy logic and evolutionary algorithms as in Figure 2.4. This section presents a brief introduction of different methods used to solve the SA problem. Game theory is a decision making framework that has been extensively used to solve many engineering design problems. In a multiuser wireless network, the action of one user/player may impact on other user's performance, so a game can be formulated to obtain a stable solution through the objective of equilibrium. The games are classified as non-cooperative and cooperative. The game choice depends on whether the users are exchanging decision information or not. In literature, game theory technique has been used to solve the SA problem in CR networks [41, 42, 43, 44].

Network graphs have been used to solve the SA problem, where the network

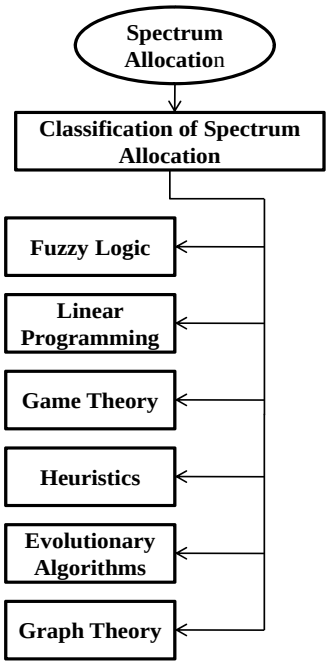

Figure 2.4: Classification of Spectrum Allocation techniques [4]

structure is assumed to be known a priori [45]. One of the most common method is network conflict graph that shows the interference between the nearby SUs [8, 46, 47]. It shows the communication and connectivity between the nodes and the links between the nodes. Dynamic conflict graphs can be used to accommodate the changes in the assignment step in the SA algorithms due to interference between the users [48, 49]. Graph coloring is another method to solve SA problem in CR network, where the problem is mapped to an either uni-directional or bi-directional graph [50, 51, 52, 8]. The allocation problem is identical to assigning a color to each vertex from the available color list. The primary constraint used in this method is that two connected SUs must not assign to the same color (same channel). When the two SUs are close to each other or using the same channel, it is necessary to include adjacent channel interference as another layer in the graph coloring method.

Fuzzy logic is generally used for decision making in various engineering problems. It is also used to solve the SA problem [53, 54]. A Fuzzy logic controller contains four modules: fuzzy inference engine, fuzzy rule base, and a fuzzification/defuzzification module. The fuzzy logic system takes decision about the selection of spectrum for SUs based on the inputs like channel availability, arrival

rate of PUs and SUs, distance and interference between the users and required quality of service. For a scalable CR network, this technique does not provide a feasible solution because it requires a large number of rules to formulate the membership function, and it may affect on results if not structured correctly. Linear Programming is another familiar method used to solve SA problem in CRNs [55, 56]. It is reported that joint power and spectrum allocation is a NP-hard problem and it has been formulated as a Mixed Integer Non-linear Programming (MINLP) problem [57].

2.5.1 Heuristic techniques

The SA problem is NP-hard. It is difficult to find an optimal solution for solving NP-hard problem using deterministic algorithm in a limited execution time. Heuristic techniques are good choice to achieve an optimal solution with in the limited time for solving NP-hard problems. The heuristic technique can provide a near-optimal solution for complex problems at modest computational cost. In literature, variety of heuristic algorithms were used to solve the SA optimization problem [58, 59, 60]. A channel assignment algorithm based on heuristic was proposed to decrease the complexity of the problem [59]. In this method, PUs and SUs are selected randomly at each step. All available channels are scanned iteratively and SU selects the suitable channel that satisfies the QoS requirements.

The SA is formulated as an Integer Linear Programming (ILP) problem and solved using heuristic method [61]. It gives less computational complexity with sub-optimal solution. In [61] , for short distance transmissions SUs are assigned with low SINR channels. For long distance transmissions, channels are assigned based on local information about the availability of spectrum holes. The SA problem was solved for both static and dynamic CRN with known and unknown traffic requirements respectively. All available channels are divided into M sets, and each SU maintains a channel list of other SUs, based on the distance to other SUs and SINR of the channel. The channels with low SINR are assigned to the closest SUs. The main advantage of heuristic methods is that it is simple and able to find near optimal and quality solutions for NP-hard problems. However, these are less sensitive to variation in data quality and problem specifications. The disadvantage of the heuristic method is that the same method cannot be applicable to solve other problems [62].

2.5.2 Evolutionary algorithms

Spectrum allocation is formulated as an optimization problem and solved using different optimization algorithms [9]. In the context of SA, the objective of optimization algorithm is to determine the best-suited channel for SUs by maximizing spectrum/network utilization that satisfies interference constraints posed by both SUs and PUs. Mathematically, the spectrum utilization is termed as fitness function and solution of the problem is termed as channel assignment matrix. Most of the traditional optimization algorithms are based on first derivative of the fitness function and tend to converge in local minima of the constrained surface. To avoid this disadvantages, many derivative free optimization algorithms were emerged for solving linear and nonlinear optimization problems [63]. Evolutionary algorithm (EA) is one such derivative free algorithm popularly being used in solving many science and engineering problems. It is a population based stochastic algorithm.

The basic principle of EA is: "given a population of individuals the environmental pressure causes natural selection (survival of the fittest) and this causes a rise in the fitness of the population" [64]. In the context of SA, the candidate solution i.e., the channel assignment matrix is randomly generated and the network utilization is evaluated as an abstract fitness measure. Some of the better candidate from the population are chosen for next generation through recombination/mutation and crossover process based on the fitness (network utilization) value. In the mutation stage, one or more population will be considered for obtaining a new candidate whereas crossover will consider two population and obtain a new candidate. This new candidate compete with old one based on the fitness value for next generation. This is iterated till a stopping criteria is met. Stopping criteria can be either the number of iteration or an optimum solution by a candidate of the population. The pseudo- code of EA algorithm is shown in Algorithm 2.

Genetic algorithm (GA) is a popular EA, where the initial population contains feasible and infeasible set of solutions of channel assignment and the constraints are based on the interference posed by SUs and PUs [9]. In GA, the genetic operators namely, crossover and recombination, mutation, and selection are play an essential role in solving the optimization problem. During the optimization, multiple offsprings are generated and behaves like independent agents and able to explore the search space in many directions simultaneously. This characteristic provides

Algorithm 2 : "Pseudo-code for general scheme of an Evolutionary algorithm [64]

```
BEGIN
  INITIALIZE population with random candidate solutions;
  EVALUATE each candidate;
  WHILE (termination condition is satisfied) DO
    1. SELECT parents;
    2. RECOMBINE pairs of parents;
    3. MUTATE the resulting offspring;
    4. EVALUATE new candidates;
    5. SELECT individuals for the next generation;
  END WHILE
END"
```

to parallelize the algorithm for implementation. Each solution is represented as a chromosome and collectively known as population. Initially, each bit in the chromosome is generated randomly and during the optimization process crossover, mutation and selection operations take place on each chromosome. Fitness values of each chromosome are evaluated by fitness function, and the number of bits in the chromosome is mapped to channel allocation matrix. This is to avoid the redundancy in encoding the number of bits into the chromosome. The fitness functions like throughput and interference are optimized to find the best solution for SA problem [9, 65, 66, 67]. The disadvantage of using GA is is that it has slow convergence rate to find an optimal solution.

Recently a new meta-heuristic algorithm namely Harmony search (HS) algorithm was proposed to solve nonlinear optimization problem [68]. It imitates the music improvisation process. In general a musician improvises one pitch, based on either by playing any one pitch from his (or her) memory, or playing an adjacent pitch of one pitch from his (or her) memory, or playing totally random pitch from the possible range of pitches [68]. Based on this principle the HS algorithm updates the candidate solution and finds an optimum solution for a given problem. HS algorithm is used to solve channel assignment problem [69]. It builds a vector of channel allocation matrices (harmonies) through intelligent combinations and mutation operations. These operations are controlled by two control parameters: Pitch Adjustment Rate (PAR) and Harmony Memory Considering Rate (HMCR). It has mainly three steps: (i) the values of channel assignment are randomly initialized without knowledge of required solution; (ii) the initialized harmonies are improvised using HMCR and PAR control parameters; (iii) improvised harmonies

are evaluated using fitness function and resultant values are stored in the harmony memory. These steps are executed till the maximum iterations are completed and the best solution is obtained.

A swarm intelligence algorithm called Ant Colony Optimization (ACO) was proposed by Dorigo et al. [70]. In ACO, the ant system simulates the behavior of real ants with artificial features like memory, visibility and discrete time. Even though the real ants are blind, they are able to find a shortest path from food source to their nest. The ants release a liquid substance called pheromone on the transiting route to exploit information of food source. The ACO algorithm was used to solve the SA problem in CR networks [71]. In this method, the broadcast message contains information about the probability of successful transmissions as pheromone [72]. The SUs adapt the channel assignment according to the received broadcast message. If the transmission probability of a channel decreases, then the SU has less chance to select that channel. A channel is selected with the use of pheromones that maximizes the total throughput, but it may not provide the required quality of service in transmission.

Another swarm intelligence technique known as Artificial Bee Colony (ABC) algorithm was proposed by Karaboga and Bastruk [73]. It simulates the foraging behavior of a bee colony. This algorithm introduces employed bees, onlooker bees and scout bees,. Initially, the employed bees are randomly distributed to search for number of food sources. Onlookers follow employed bees depending on the quantity of food at the source. Scout bees search for new food sources randomly. Finally, the optimal solution is represented by the location of food sources that has optimum amount of food detected by onlooker or employed bees. The amount of food in nectar represents the quality of the solution. The ABC algorithm is used to solve the SA problem [74]. In SA problem, the position of an onlooker or employed bee corresponds to the channel assignment matrix for SUs and nectar food is the throughput that is maximized.

2.6 Power allocation

Power allocation is an important step in spectrum allocation in wireless communication systems. In this step, optimum power is alloted to SUs to minimize interference and reduce the effect of multi-path fading. In the present CR protocol, each SU scans the wireless channel and detect the spectrum hole. Then, SU

communicates over the vacant channel. Kulkarni et al. proposed a channel and rate allocation scheme under the presence of co-channel interference to minimize the total transmitted power [75]. It was formulated as an optimization problem and solved using heuristic algorithm to provide an optimal channel and data rate to the wireless links by minimizing the total transmitted power. However, PU interference is not considered while solving the problem [75].

Hoang et al. proposed a two-phase channel and power allocation technique in which PU interference is taken into consideration to maximize the total system throughput [76]. The allocation was carried out in the centralized server and it provides a decision on assignment of channel and power to the SUs. In the two-phase resource allocation process, firstly the channel and power are assigned to the base stations for maximizing coverage and minimizing interference to PUs. In the second phase, each base station assigns the required channels to the active SUs. It achieves significant performance gain compared to methods like random allocation, non-overlapping allocation and interference graph based allocation [76]. However, the two phase allocation method is applicable only for centralized approach with infrastructure provided to SUs. Li et al. formulated a multiuser channel and power allocation problem and solved using a non-cooperative game technique to maximize the system capacity of the distributed CR network [77]. The SUs choose the optimal power need to be transmitted on each channel with respect to the payoff function that takes into account the capacity gain of themselves and the loss of others. The simulation results shown that the algorithm achieved superior performance compared to the selfish channel and power allocation scheme [77].

Haddad et al. proposed an uplink distributed binary power allocation technique in which PUs and SUs share the same spectrum to maximize the cognitive network capacity while maintaining the QoS of the PUs [78]. The thesis presents an improved power allocation algorithm based on [79]. In the distributed approach, each SU need to determine its transmission power below the threshold without affecting the QoS of PUs. In a fast fading CRN environment, centralized coordination is complex to implement. Thus in this work, a distributed CR network is considered, where each SU need to determine its transmission power below a threshold such that the total sum rate of the network is maximum while avoiding interference to the PUs. Here both type of users may communicate through the same channel by optimally selecting the SU transmission power on the corresponding channel that maximizes the network capacity.

In CR network, joint spectrum and power allocation for SUs introduces more challenges. In this scheme, the same spectrum can be alloted to the SUs that is far away from PU with an appropriate power to maintain the interference below the required limit of PUs. The main issue is to protect the PU transmissions without any interruptions. In this work, a joint spectrum and power allocation method is proposed to maximize the average capacity of the user as well as the total network utilization without any interference to PUs by considering the transmission power constraints.

2.7 Hardware platforms

There exists several hardware platforms such as microcontrollers (μC), digital signal processors (DSP), field programmable gate arrays (FPGA) and application specific integrated circuits (ASIC) for developing an embedded system. Platforms like μC and DSP are revolving around firmware development using software methodologies rather than hardware development for the application [80]. FPGA platforms support both hardware-based approach (system developed entirely in the hardware) and processor-based approach (system developed entirely in the firmware). It has the flexibility to customize the hardware design by adding any combination of peripherals and controllers, that are not available in either DSP or microcontroller based system.

2.7.1 Field Programmable Gate Array

FPGAs are equipped with programmable connectivity between the logic blocks where the programmability depends on different technologies like EPROM, anti-fuse or SRAM [81]. FPGA accommodates flexibility, shorter design time and is suitable for developing system prototype. FPGAs have embedded processors and other peripherals to develop a complete System on Chip (SoC) platform for various embedded applications. Due to this, FPGAs are used in complex DSP and embedded applications. It also offers dedicated parallel architectures to reduce the execution time of the complex algorithms. In order to meet real-time specifications of an embedded application, hardware and software need to be combined into a single FPGA platform that provides high level of integration density, flexibility and reduced communication overhead between various peripherals. Most conven-

tional approach for design and implementation of a custom hardware is to develop an Intellectual Property (IP) using Hardware Descriptive Language (Verilog or VHDL). These custom hardware IPs are interfaced to a processor core along with other I/O peripherals on a single FPGA chip. In this thesis, a Xilinx Virtex-5 FPGA development board is used for verifying the developed IPs because, it has a PPC440 microprocessor on the same silicon area, cross-platform compatibility with most common communication protocols like Ethernet and USB [82]. The FPGA core (ML507) has a variety of resources i.e., 128 DSP slices, 11,200 configurable logic block (CLB) slices, 6 Clock management blocks, 19 I/O banks and 5328 kB of RAM, suitable and enough for implementing the SA algorithm.

2.8 Hardware-Software Co-design

In an embedded system, speed and cost are the most demanding parameters, and the system performance can be improved by increasing the efficiency of both hardware and software involved in the design. In the co-design, the algorithm is partitioned into flexible software (SW) and fixed hardware (HW) modules. The partitioning of HW and SW is decided by profiling that identifies [83] the computational intense tasks/subtasks of the algorithm. Subsequently, these are designed and implemented either on a processor or using a hardware accelerator/dedicated coprocessor unit [84, 85]. The hardware is used to implement computationally intensive tasks and hence accelerate the execution speed of the algorithm/task. The HW module mostly run on customized hardware and SW module run on an embedded processor. The design flow for implementing a design on FPGA based platform using Xilinx Embedded Development Kit (EDK) is mentioned in Appendix A.1.

2.9 Hardware accelerator

Hardware accelerator is used to increase the execution speed of a specific routine in a separate custom hardware unit other than the processor. The primary benefit of an accelerator is achieved only if the total execution time of the hardware accelerator and communication overhead is less than the execution time of the same task in a processor [86]. These accelerators can be designed using an ASIC or

FPGA approach based on the application requirements. ASIC-based accelerators are more cost effective design and has longer design cycle compared to FPGA-based accelerators. In the present thesis, FPGA based accelerators are developed. The capability of on-demand FPGA reconfigurability facilitates the accelerator adapt to the actual needs of a specific application running on the processor [87, 83].

The important step in designing an accelerator is to select a proper communication/interface technique that affects the quantity of the data transfer between the processor and the accelerator [88, 89]. Selection of either handshaking or direct/interrupt mechanism to synchronize the data transfer between processor and hardware accelerator is an important issue in a design. In general, the instruction pipeline connection approach is used to connect the accelerator directly to the coprocessor port of processor via Auxiliary Processor Unit (APU) interface [90]. The coprocessor port is tightly coupled to the internal instruction pipeline of PowerPC processor. The custom hardware accelerators are executed using the coprocessor instructions, and it avoids communication overhead between the processor and coprocessor. The implementation of an accelerator unit itself is the critical task of this approach.

2.10 Programmable System on Chip design

System-on-Chip design is the recently evolved design methodology that combine various IPs and third-party cores into a single chip to minimize the design productivity gap [91]. The IPs may include memory blocks, processors, signal processing functions, custom IPs, etc. Programmable System on Chip (PSoC) design facilitates to program logic gates in an entire system as per user requirements in the field [91, 87]. FPGA based PSoC provides better flexibility to include a hard processor, soft processor, custom IPs and various peripherals to design a complex system. This methodology also has the feature of including an operating system. It offers functionality of a personal computer on a single Integrated Circuit (IC) chip. The general architecture of PSoC platform is given in Figure 2.5. In this figure, external and on-chip memories are used for executing the software programs. The Joint Test Action Group (JTAG) and Universal Synchronous and Asynchronous Receive and Transmit (UART) ports are used to debug, monitor and download the bit stream on to the FPGA.

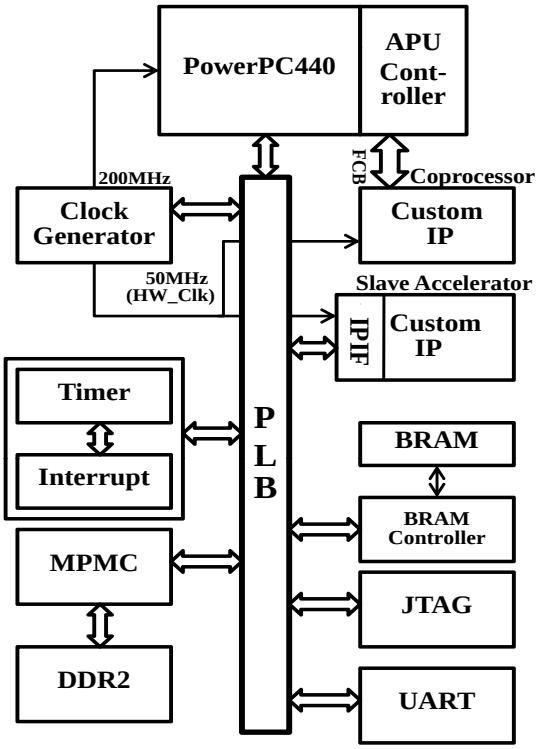

Figure 2.5: Basic System-on-Chip platform

2.10.1 Embedded processors

System-on-Chip based embedded applications demand on-chip processor core in FPGA to target a single chip solution. There are two types of processor cores embedded in FPGA, namely soft-core processor and hard-core processor. A hard-core processor like IBM PowerPC 440/405 [92, 93] in Xilinx Virtex-5 FXT / Virtex-II Pro FPGA is a dedicated physical component on the chip. A soft-core processor like Altera Nios II [94] or Xilinx MicroBlaze [95] is included in the programmable logic of the chip. If an application requires an embedded processor, then the design involves processor selection, compatibility with memory or I/O interfaces and software development.

2.10.2 Memory

Embedded system use different memories like on-chip memory, cache memory and external memories like SRAM, DRAM, SDRAM, etc. [96]. In a SoC platform, memory hierarchies play an important role to achieve energy efficiency and opti-

mal run time. FPGA-based SoC system is a memory mapped system in which each peripheral is associated with an address. In Virtex-5 FPGA, the basic primitive of internal memory is a Block RAM with a size of 148 Kbytes. However, the size is different from one FPGA to other. Decreasing the amount of on-chip memory will increase the performance and reduce the active area of SoC. Similarly, DDR2 SDRAM and SRAM are of size 256Mbyte and 1Mbyte respectively. These are used as off-chip external memories. The external memory access time is high compared to on-chip memory and SRAM [97]. These can be configured using Multi-Port Memory Controller (MPMC)/ Memory Controller Interface (MCI) in SoC platform. The use of external memory degrades the throughput of a design. Cache memory provides enough energy efficiency and run-time efficiency by maintaining local access to frequently used data and instructions. In the PSoC, if an application program fits entirely within the local memory, then the design may achieve optimal throughput performance. The program/data memory usage in SoC can be manipulated using the Linker Script of a system. It can be mapped either into internal memory such as BRAM or external memory.

2.10.3 Peripherals

In a SoC based embedded system, all peripherals can communicate with the processor through different buses like Processor Local Bus (PLB), Fabric Coprocessor Bus (FCB), etc. These peripherals are divided into two categories namely generic and custom peripherals. Generic peripherals can be either soft-core or hard-core, and these are configured during the SoC design [98]. Hard-core peripherals are implemented in silicon whereas soft/custom peripherals are configured in FPGA fabric. The custom peripherals are developed by the user with the given specifications using HDL languages. Custom peripherals like IDCT [99], FFT [100], FIR [101] filter etc., can be connected to a processor through various bus interface techniques like Slave Unit (SU) and Auxiliary Processor Unit (APU). In this thesis, custom peripherals namely DE, MODE, DE-based SA and MODE-based SA IPs are developed and interfaced to PPC440 processor using APU controller.

2.10.4 Universal Asynchronous Receiver and Transmitter (UART)

UART is a serial communication protocol. This peripheral can perform parallel to serial and serial to parallel conversion on the received data. It is used for

debugging, transferring and monitoring the input/output data between the on-chip embedded processor and serial port of a desktop system [87].

2.10.5 Digital Clock Manager (DCM)

Most of the systems have a single external clock that generates fixed clock frequency. However, a SoC platform offers various peripherals that require different clock frequencies to perform their operation [102, 103]. For example, a custom IP can run at 50MHz whereas processor and memory can run at 200MHz. Digital Clock Manager can provide various clock frequencies required for different modules of the system. It also performs elimination of clock skew to improve the system performance.

2.10.6 Bus interfaces

Different types of bus interface are used to connect the processor, memory controllers and external peripherals to the system bus. Each interface logic is unique to the corresponding bus. A bus arbiter is included in every bus interface to control the bus access. There are three different types of bus interface techniques, namely Slave Unit, Device Control Register (DCR) and Auxiliary Processor Unit (APU) to interface any external peripheral. In slave unit interfacing technique, the custom peripherals are interfaced to the Processor Local Bus (PLB). Intellectual Peripheral Interface (IPIF) is responsible to provide bi-directional interface signals between the custom peripherals and the processor. The DCR interface offers the PPC440 embedded processor to control and check the status of other peripherals. This interface is interlocked with control signals such that it can be connected to peripheral units and respective clock frequencies from the embedded processor. In the APU interfacing technique, custom peripherals are interfaced with the processor using an APU controller. Detailed description about the bus interfaces is available in [102, 104]. In this thesis, APU interfacing technique is used to connect the custom IPs with the embedded processor.

2.10.6.1 Auxiliary Processor Unit (APU) interface

The PPC440 processor in Virtex-5 FPGAs has a fabric coprocessor bus (FCB) (128 bit) through which custom peripherals are interfaced using an APU controller. The

custom peripheral is invoked using the processor extended instruction set i.e. Load and Store. This approach provides the flexibility of interfacing a coprocessor with the instruction pipeline [90].

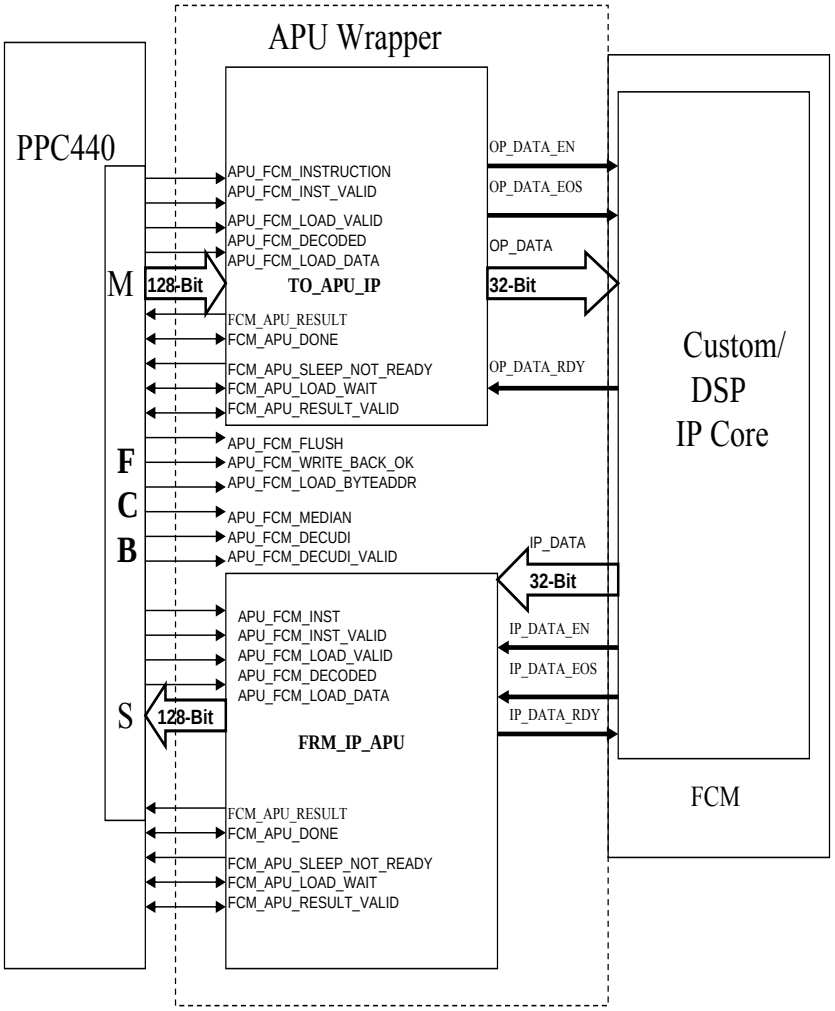

Figure 2.6: APU controller interface to PowerPC 440 processor

This coprocessor can execute the desired task concurrently with the PPC440 processor extended instructions. The APU controller synchronizes the clocks of processor and custom IP, that can run at different frequencies. The APU controller decodes the processor instructions in a pipelined manner for faster execution of overall instructions. The custom IP is designed as a fabric coprocessor module (FCM) and interfaced to PPC440 using APU controller. There are two major classes of Fabric Coprocessor Module (FCM) instructions, (a) storage instruction and (b) non-storage instruction. In this work, storage instructions i.e., load and

store are used. Non-storage instructions like floating point arithmetic instructions and User Defined Instructions (UDIs) are based on opcodes.

The APU controller interface for custom IP (i.e. either DE core or MODE core) is shown in Figure 2.6. The coprocessor interface has mainly three modules, i) PPC440 processor, ii) APU wrapper (iii) custom IP core. The processor is used to send and receive data to and from the custom IP. The APU wrapper is used to interface the custom IP with the processor. It has two different modules namely IP_APU and APU_IP. The APU_IP module receives data from the processor and sends it to custom-built IP, whereas the IP_APU module receives data from the custom-built IP and sends it to the processor (PPC440). The APU_IP receives 128-bit signal. However, the IP has only 32-bit width input, so the IP receives a full set of data in four clock cycles. Similarly, the IP_APU module receives 128 bits of data from the IP in four clock cycles. The APU wrapper is interfaced with the IP core using six control signals OP_DATA_EN ,OP_DATA_RDY, OP_DATA_EOS, IP_DATA_EN, IP_DATA_RDY, IP_DATA_EOS [105]. A Finite State Machine (FSM) with five states, i.e. load, load_valid, store, store_valid and idle states control the data transfer between the processor, IP_APU and APU_IP.

2.11 Hardware implementation of Spectrum Allocation techniques

In traditional wireless networks, a temporary channel is assigned by a base station of the corresponding cell to make communication using a wireless terminal. If the wireless terminals are in roaming from one cell to another cell, the corresponding base station has to allocate a different channel to the user to avoid interruption. It is known as spectrum hand-off. The hand-off process should be performed fast, otherwise it deteriorates the quality of transmission. Hence, the process of channel allocation need to be executed fast to establish high-quality and efficient wireless communication [17]. This task should be performed during the instances of call initiation and call hand-off. In literature, many works were concentrated on solving the channel allocation problem. However, there is a very limited work on a hardware implementation of channel allocation algorithms to accelerate the speed of execution. Integrated Channel Manager is implemented in hardware

to provide efficient channel allocation algorithm in cellular networks with multi-terminal platforms [106, 107]. The efficiency of channel allocation is improved by hardware via the high degree of parallelism. It supports single and multiple hand-offs and also provides efficient call rejection when the system is not supported.

Jelena proposed hardware implementation of channel allocator to speed up the execution of different channel allocation algorithms [17]. The hardware device is integrated into a network switch, and it is further connected to other components via a system bus. Three different algorithms namely Fixed, Quasi-Random and Semi-Fixed channel allocation algorithms were implemented in hardware, and these are located in the execution unit of channel allocator. According to the traffic demands and interference conditions, the execution unit will select the suitable algorithm to provide the desired performance. The hardware IP proved its efficiency in the process of allocation of available channels (order of nanoseconds). In the present work, a hardware implementation of SA algorithm for CR networks is proposed to speedup the execution time. In the distributed network architecture, each SU determines the spectrum availability and allocate the desired spectrum. In this scheme, each SU considers the locally available information from the neighborhood users and decides spectrum assignment. As each SU implicitly have an embedded computing platform, the SA task needs to be performed on it. However, running the SA on an embedded processor consumes most of the platform resources, thereby degrading the performance of other applications running on it. Hence, there is a requirement of a dedicated hardware peripheral for performing the SA task.

2.12 Tools used

MATLAB and C programming languages are used for algorithmic simulation and validation. Xilinx Software Development Kit (SDK) is used for profiling the algorithms on PPC440 and MicroBlaze processors and calculates the execution time of algorithm to solve the problem.

The hardware development platform uses various tools for HDL coding, simulation, synthesis and debugging. Xilinx Isim and Mentor Graphics Modelsim simulation tools are used for coding and simulation. For hardware and SoC development, Xilinx Integrated Development Environment (IDE) with Integrated Software Environment (ISE) [108] and Embedded Development Kit (EDK) [109]

are used. EDK tool is used for building a SoC system and generating custom peripherals (such as UART, JTAG, timer, IP) that are interfaced to the SoC system [110].

Chapter 3

Spectrum Allocation using Differential Evolution algorithm in Cognitive Radio Networks

In this chapter, Differential Evolution (DE) algorithm is applied to solve the spectrum allocation problem in cognitive radio networks for achieving efficient network resource utilization. The performance of DE algorithm is compared with Particle Swarm Optimization (PSO) and Firefly algorithms in terms of quality of solution and time complexity. The performance of all the three algorithms is analyzed to provide conflict free channel assignment to secondary users.

3.1 Introduction

In CR technology, each cognitive radio user (secondary user) can adapt to various technologies and utilize the vacant spectrum without any interference to the licensed users (primary users). Cognitive radio is built on a software-defined radio with intelligence that can sense, learn and adapt to statistical variations in the operating RF environment [20]. The first phase of a CR cycle is spectrum sensing. During this phase, the vacant bands (spectrum holes) are identified. Subsequently, in the second phase, SUs use the detected holes on request for communication while satisfying the interference constraints imposed by primary and secondary users. This is termed as Spectrum Allocation (SA). In CR networks, the general procedure to solve the SA problem is divided into three steps. In the first step, criteria (defines the target objective like maximize spectrum utilization) to solve the SA problem are chosen. In the second step, the approach to model the SA problem that satisfies the criteria is selected. In the final step, the most suitable technique that solves the SA problem is selected. In this work, spectral efficiency and fairness are chosen as criteria and evolutionary algorithm is selected

as technique to solve the SA problem.

In SA, the best channels are assigned to the requested users for achieving efficient channel utilization by minimizing interference between the users. This avoids performance degradation of wireless networks. It has been proven that assigning optimal channels in an arbitrary network topology is the NP-hard problem [8]. The solution of SA affects traffic load among the wireless links, network topology or connectivity between the nodes of a network. Hence, there should be a trade-off between minimizing the traffic contention, maximizing performance and connectivity. In CRN, the use of a multi-radio device is a feasible solution to increase the network capacity. However, an efficient channel allocation algorithm is necessary to avoid interference between multiple radio interfaces. The assignment algorithm needs to take various parameters like vacant channels, required transmission rate of a user, transmission power and bit error rate constraints into account for efficient allocation. Furthermore, with increasing number of users and their requirements, complexity of the SA problem increases exponentially. Traditional non-evolutionary techniques can no longer meet the demand.

In literature, evolutionary algorithms have proved to be effective to solve NP-hard problems, because of their easy implementation characteristics and low computational complexity. Abril et al. used evolutionary algorithms to obtain an effective solution for frequency assignment problem in Global Systems for Mobile (GSM) communication networks [111]. The use of evolutionary algorithms proved that they efficiently allocate frequencies to each radio cell by considering the interference constraints given by a compatibility matrix. In CR networks, SA can also be formulated as an optimization problem. Indeed, it has been solved using different evolutionary algorithms like Genetic Algorithm (GA), Quantum Genetic Algorithm (QGA), Particle Swarm Optimization (PSO) [9] and Artificial Bee Colony (ABC) [74]. During the last decade, Differential Evolution algorithm has gained popularity in solving NP-hard problems due to its inherent capability to find global optimum solutions [112, 113]. In this chapter, a simulation study of PSO, Firefly and DE algorithms to solve the SA problem is presented. The performance of the algorithms in terms of quality of solution and time complexity for solving SA problem is also compared.

3.2 Related work

In literature, different approaches such as genetic algorithm, neural network, and game theory have been used to solve the SA task of the CR [114, 115, 42, 116, 117, 8]. The SA problem of CR networks is solved by the stable matching game theory technique using Gale-Shapley theorem [118]. For any game, a steady state solution always exists for definite characteristics, and any unilateral action of a player leads to lower utility. The resultant solution is called Nash Equilibrium. The stable matching theory was developed to study the stability of marriage, i.e., it matches the preferences of men to the preferences of women. In SA problem, user and channels are assumed to be men and women, the preferences are corresponding to utility functions. Distributed version of Gale-Shapley consists of roaming and non-roaming users. For each time-slot, roaming users transmit to the best channel among others, and non-roaming users transmit to the same channel as in the previous time slot. Based on utility function, a best channel is selected for each user as non-roaming and others confirmed as roaming users. In this way, matching can be done between the users and channels, after a certain number of time slots a stable state called an equilibrium condition is reached. The main disadvantage of this technique is that the game formulation and utility functions must be designed carefully to achieve equilibrium.

Peng et al. defined a general framework for SA problem, which is mapped to a variant of graph coloring problem and solved using a general approximation methodology called vertex labeling [8]. This technique was applied on both centralized and distributed approaches, and the experimental results show that the distributed approach provides allocation assignments similar to a centralized approach. However, distributed algorithm takes less computational complexity. In graph coloring technique, when the size of the graph increases, the execution time for solving the problem increases exponentially. It means that the SA problem is very difficult to solve for real networks. Hence, approximate algorithmic methods (provide a solution that is close to the absolute minimum in a moderate time) must be used in practical applications.

In literature, different evolutionary algorithms have been used to solve the SA problem for CR networks. Zhao et al. formulated SA as an optimization problem and solved it using GA, QGA and PSO algorithms [9]. From the simulation results, it was found that PSO performs better than GA and QGA in optimizing MSR and

MPF functions. In contrast, QGA performs the best in optimizing MMR utility function. It has been shown that these evolutionary algorithms greatly outperform the color sensitive graph coloring algorithm. However, the performance of these algorithms is not studied by varying the number of users and channels. Ye et al. proposed an improved genetic spectrum assignment model, in which population of genetic algorithm was divided into two sets of feasible spectrum assignment that randomly updates the spectrum assignment strategies [119]. In [119], a penalty function was included to satisfy interference constraints. It resulted in better performance than the conventional genetic and quantum genetic assignment model in optimizing the utility function by varying the number of SUs and channels. This work was limited to maximize the network utility function called Maximum-Sum-Reward (MSR) only and does not consider the fairness based utility functions. Besides, the improved genetic algorithm performance was not compared with other popular evolutionary algorithms like PSO, DE and Firefly algorithms.

Cheng et al. used a biological inspired ABC algorithm to optimize the network utility functions by considering fairness and efficiency for cognitive users simultaneously using the weighted summation method [74]. This work defined a general framework for spectrum allocation in a CR system and optimized the allocation of spectrum for fairness and efficiency. The performance of ABC algorithm was compared with GA in terms of convergence speed and time complexity. It was proved that ABC outperforms GA. However, the algorithm performance was not compared with other popular evolutionary algorithms like PSO, Firefly, and DE. It considered only two utility functions and did not show the impact on network utility by varying the number of users and channels. Koroupi et al. proposed a new approach to solve spectrum allocation based on Ant Colony System (ACS) and Graph Coloring Problem (GCP) in CR network [10]. The performance of ACS was compared with PSO and Color Sensitive Graph Coloring (CSGC) for a number of SUs, PUs, and available channels. ACS performed better than the other algorithms, but it required more execution time to converge to the solution.

In work not presented in this thesis, the SA problem was solved by optimizing the three network utility functions, namely MSR, MMR and MPF independently using Firefly algorithm [120]. The performance of Firefly algorithm was compared with PSO and ABC algorithms in terms of time complexity and quality of the solution. It was observed from the simulation results that Firefly improved the quality of solution by 17% and 13% and the time complexity by 100% and

103%, when compared to the PSO and ABC algorithms respectively. However, the performance of these algorithms were not studied under different network parameters like a number of SUs, PUs and a number of channels. Liu et al. used Binary Firefly Algorithm (BFA) to solve the SA problem by maximizing the spectrum utilization and SU fairness under interference constraints [121]. It also compared the performance of BFA with GA and PSO under a different number of SUs and channels. Simulation results have shown that BFA outperforms GA and PSO in terms of both convergence speed and quality of the solution.

In this chapter, DE algorithm is used to optimize the network utility functions to provide the best channel assignment to secondary users. The effect of varying the number of primary, secondary users and number of channels on network utility is also presented. The performance of DE is compared with PSO and Firefly algorithms by evaluating convergence speed, time complexity and quality of the solution.

3.3 Spectrum Allocation using Differential Evolution algorithm

The SA model and the problem formulation are explained in Section 2.4 of chapter 2. This section explains about DE algorithm and the procedure to optimize the network utility functions for solving the SA problem.

3.3.1 Differential Evolution algorithm

Differential Evolution (DE) algorithm is an evolutionary computation method proposed by Storn and Price in 1995 [112]. It has been applied in diverse domains of science and engineering applications. This algorithm became a popular evolutionary algorithm because (a) it is simple to implement, (b) it has better performance, (c) It has less number of control parameters and less space complexity [113].

DE algorithm employs real-coded variables and typically relies on mutation as the search operator. It has evolved to share many features with conventional GA, like both maintain populations of potential solutions and use a selection mechanism for choosing the best individuals from the population. However, DE has more advantages than GA, like it operates directly on floating point (FP) vectors [122]. DE is a parallel direct search method that employs a population

of size NP that are FP encoded individuals. In DE, a global best solution is obtained using the direction and distance information just as differentiation of the population. The operation of searching an individual in the search space is dynamically adjusted with differentiation's direction and step length. The major steps of the algorithm are: (i) initialization of the population (ii) mutation (iii) crossover and (iv) selection process.

Algorithm 3 : Pseudo-code for the Differential Evolution algorithm based Spectrum Allocation

Step 1: Initialize the SA algorithm parameter values
Initialize number of secondary users (N), number of primary users (K), number of channels (M), Channel availability matrix $L = \{l_{n,m}|l_{n,m} \in \{0,1\}\}_{N\times M}$, Reward matrix $B = \{b_{n,m}|b_{n,m} \in \{0,1\}\}_{N\times M}$, Constraint matrix $C = \{c_{n,p,m}|c_{n,p,m} \in \{0,1\}\}_{N\times N\times M}$ and set $L_1 = (n,m)|l_{n,m} = 1$ in which elements are arranged in ascending order with n and m
Step 2: Initialize the control parameter values of the DE algorithm Initialize scale factor F, crossover rate Cr, maximum number of iterations G_{MAX}, dimension of the population $D = \sum_{n=1}^{N}\sum_{m=1}^{M} l_{n,m}$ and the population size NP.
Step 3: Set the generation number G=0 and randomly initialize a population of NP individuals using Equation 3.1. The initial population P_G=$[X_1^{(G)}, ..., X_i^{(G)}, ..., X_{NP}^{(G)}]$ is evaluated using the objective functions (Equations.(2.3), (2.4), (2.5)) where
$X_i^{(G)}$=$[x_{1,i}^{(G)}, ..., x_{3,i}^{(G)}, ..., x_{D,i}^{(G)}]$ and each individual uniformly distributed in the range $[X_{min}, X_{max}]$, where X_{min}=$\{x_1^{min}, x_2^{min}, ..., x_D^{min}\}$ and X_{max}=$\{x_1^{max}, x_2^{max}, ..., x_D^{max}\}$ with $i = [1, 2, ..., NP]$.
Step 4:

```
while maximum no. of iterations is not reached do
  for i=1 to NP //do for each individual sequentially do
    Step 4.1: Mutation Step
    Generate a mutant vector V_i^(G)={v_{1,i}^G, ...., v_{D,i}^G} corresponding to the ith target vector
    X_i^(G) via the differential mutation scheme of DE as given in Equation (3.2)
    Step 4.2: Crossover Step
    Generate a trial vector U_i^(G)={u_{1,i}^(G), ...., u_{D,i}^(G)} for the ith target vector X_i^(G) through
    binomial crossover as given in Equation (3.3)
    Step 4.3: Selection Step
    Perform the Selection operation given in Equation (3.4) by evaluating the trial vector
    U_i^(G) using Equations.(2.3), (2.4), (2.5).
  end for
  Step 4.4:Increase the iteration count
  G = G + 1
end while
Report results
Terminate
```

The operation of DE starts with initialization of population randomly in the search space, and it operates cooperatively between the individuals of the population. Each decision parameter in every vector of the initial population is assigned

with a randomly chosen value within its feasible bounds as in Equation (3.1).

$$x_{j,i} = x_j^{min} + rand_j[0,1].(x_j^{max} - x_j^{min}) \tag{3.1}$$

where $i = 1,..,NP$ and $j = 1,..,D$. $x_{j,i}$ is the initial value of the j^{th} parameter of the i^{th} population. x_j^{min} and x_j^{max} are the lower and upper bounds of the j^{th} parameter respectively. Once every vector of the population has been initialized, its corresponding fitness value is calculated and stored in a memory. The mutation, crossover and selection operations are performed to create population for the next generation $P^{(G+1)}$ using the current population $P^{(G)}$. In each generation, every vector in the population has to serve as a target vector $X_i^{(G)}$, and it is compared with a mutant vector. The mutation operator generate mutant vectors $V_i^{(G)}$ as Equation (3.2) by perturbing a randomly selected vector X_{r1} with the difference of two other randomly selected vectors X_{r2} and X_{r3}. The selected vector indices are in the range of 1 to NP. The scaling factor F is used for amplifying the difference vectors and typically chosen within the range of 0 to 1.

$$V_i^{(G)} = X_{r1}^{(G)} + F.(X_{r2}^{(G)} - X_{r3}^{(G)}) \tag{3.2}$$

Vector indices $r1$, $r2$ and $r3$ are randomly chosen in $\{1,...,NP\}$. After generating the mutant vector, crossover operations is performed to enhance potential diversity of the population. The mutant vector exchanges its components with the target vector $X_i^{(G)}$ to generate a trial vector $U_i^{(G)}$ as in Equation (3.3).

$$U_i^{(G)} = u_{j,i}^{(G)} = \begin{cases} v_{j,i}^{(G)} & if \;\; rand_j(0,1) \leq CR \; or \; j = j_{rand} \\ x_{j,i}^{(G)} & otherwise \end{cases} \tag{3.3}$$

In the next step, the algorithm uses selection operator to keep the population size constant over subsequent generations. This step decides whether the target or trial vector survives to next generation. The selection operation is given in Equation (3.4).

$$X_i^{(G+1)} = \begin{cases} U_i^{(G)} & if \;\; f(U_i^{(G)}) \leq f(X_i^{(G)}) \\ X_i^{(G)} otherwise \end{cases} \tag{3.4}$$

where $f(x)$ is an objective function to be minimized. If the new trial vector yields an equal or lower value of the objective function, it replaces the corre-

sponding target vector in the next generation; otherwise the target is retained in population. These steps are repeated till the maximum number of generations is reached or any other convergence criterion is satisfied. The pseudo-code to solve the SA problem using DE algorithm is given in Algorithm 3.

3.4 Spectrum Allocation using Particle Swarm Optimization algorithm

This section presents a brief explanation about PSO algorithm and describes the procedure to optimize the network utility functions for solving the SA problem.

3.4.1 Particle Swarm Optimization algorithm

PSO algorithm was introduced in 1995 by Kennedy and Eberhart [123],[124]. It is a stochastic algorithm that exhibits many common properties of an evolutionary algorithm for solving optimization problems. It essentially imitates the food foraging behavior of social life, such as a school of fish or swarm of birds [125]. In PSO terminology, each member of the swarm is called as a particle. Every particle in the search space represents a solution. During the search process every particle remembers its current position and self-best position found so far called as personal best (pbest). The information collected by all the particles during the search process is sorted to find the global best particle (gbest).

The position of the best particle is shared with all the particles, and their flying trajectory is changed towards the swarm's gbest and it's own pbest iteratively. Each particle's position and velocity is evaluated by a fitness function to be optimized. Since each particle search randomly in a D-dimensional search space, the position and velocity of i^{th} particle are represented as $X_i = (x_{i,1}, x_{i,2}, x_{i,3}, \cdots, x_{i,D})$ and $V_i = (v_{i,1}, v_{i,2}, v_{i,3}, \cdots, v_{i,D})$ respectively. The parameter D represents the number of variables in the fitness function to be optimized. In a D dimensional search space the pbest of the i^{th} particle is represented as $pbest_i = (p_{i,1}, p_{i,2}, p_{i,3}, \cdots, p_{i,D})$ and the $gbest$ of the whole swarm is represented as $gbest = (g_1, g_2, g_3, \cdots, g_D)$. The PSO algorithm updates the velocity and posi-

Algorithm 4 : Pseudo-code for PSO Algorithm based Spectrum Allocation

Step 1: Initialize the SA algorithm parameter values
Initialize Channel availability matrix $L = \{l_{n,m}|l_{n,m} \in \{0,1\}\}_{N\times M}$, Reward matrix $B = \{b_{n,m}|b_{n,m} \in \{0,1\}\}_{N\times M}$ and Constraint matrix $C = \{c_{n,p,m}|c_{n,p,m} \in \{0,1\}\}_{N\times N\times M}$ and set $L_1 = (n,m)|l_{n,m} = 1$ in which elements are arranged in ascending order with n and m
Step 2: Initialize the algorithmic parameter values of the PSO :
Initialize Minimum and maximum values of variables $X_{min} = 0$ and $X_{max} = 1$, dimension of the population ($D = \sum_{n=1}^{N}\sum_{m=1}^{M} l_{n,m}$), population size ($NP$) and maximum iterations (t_{max}), $t \leftarrow 0, i \leftarrow 0$ (t for iterations and i for particles)
Randomly initialize particle's position $X_i^{NP} = X_{min} + (X_{max} - X_{min}) * rand(1, D)$
Randomly initialize particle's velocity $V_i^{NP} = X_{min} + (X_{max} - X_{min}) * rand(1, D)$
Evaluate fitness function values $f_i^0..f_i^{NP}$ using Equations.(2.3), (2.4), (2.5)
$pbest_i^0 \leftarrow f_i^0$, $gbest^0 \leftarrow f_{best}^0$
Step 3: Optimization Process
Step 3.1:
while $t \leq t_{max}$ do
  Step 3.2:
  while $i \leq NP$ do
    Evaluate fitness function values f_i^{t+1} of MSR, MMR and MPF using Equations.(2.3), (2.4), (2.5)
    Update velocity and position using Equation. (3.5)
    Find $pbest_i^{t+1}$ (for minimization problem)
    if $f_i^{t+1} < pbest_i^t$ then
      $pbest_i^{t+1} \leftarrow f_i^{t+1}$
    end if
    Find $gbest_d^{t+1}$ (for minimization problem)
    if $gbest_d^{t+1} < gbest_d^t$ then
      $gbest_d^{t+1} \leftarrow gbest_d^t$
    end if
    $i \leftarrow i + 1$, go to step 3.2
  end while
  Check boundaries of population
  for $k = 1$ to NP do
    if $X_k^i > X_{max}$ or $X_k^i < X_{min}$ then
      $X_k^i = X_{min} + (X_{max} - X_{min}) * rand(1, D)$
    end if
  end for
  Check Stopping Criteria
  if $t \leq t_{max}$ then
    $t \leftarrow t + 1$, go to step 3.1
  end if
end while
Report results
Terminate

tion of each particle by the following equations (3.5) and (3.6) respectively.

$$\begin{aligned} V_{i,d}^{t+1} &= V_{i,d}^t + c_1 * rand_1 * (pbest_{i,d}^t - X_{i,d}^t) \\ &+ c_2 * rand_2 * (gbest_d^t - X_{i,d}^t) \qquad (3.5) \\ X_{i,d}^{t+1} &= X_{i,d}^t + V_{i,d}^{t+1} \qquad (3.6) \end{aligned}$$

where, c_1 and c_2 are the learning factors which determines the relative influence of cognitive and social component respectively. $rand_1$ and $rand_2$ are uniformly distributed random numbers in the range [0,1]. $V^t_{i,d}$, $X^t_{i,d}$ and $pbest^t_{i,d}$ are the velocity, position and the personal best of i^{th} particle in d^{th} dimension for the t^{th} iteration respectively. The $gbest^t_d$ is the global best of the swarm in d^{th} dimension for the t^{th} iteration. The pseudo-code to solve the spectrum allocation problem using PSO algorithm is given in Algorithm 4

3.5 Spectrum Allocation using Firefly algorithm

This section presents a brief explanation about Firefly algorithm and describes the procedure to optimize the network utility functions for solving the SA problem.

3.5.1 Firefly algorithm

In recent years, Firefly algorithm has emerged as a heuristic algorithm to solve optimization problems [126]. The use of fireflies as an optimization tool was initially proposed by Yang in 2008 [127]. This algorithm imitates the social behavior of fireflies, according to distinctive flashing and attraction properties of fireflies to protect themselves from predators and absorb their prey. Firefly produces short and rhythmic flashes. These flashes are to attract female partner (communication) and to attract potential prey.

Besides, flashing also serve as a protective warning mechanism. Light intensity from a particular distance r from a light source obeys the inverse square law. Furthermore, the air medium absorbs light, and hence the intensity becomes weaker with the increase in distance. These two combined factors make most fireflies visible only to a limited distance. This algorithm has mainly two important issues, change in light intensity and formulation of attractiveness. The attractiveness of a firefly is calculated by its brightness, which in turn corresponds to fitness value of an objective function. The light intensity and the attractiveness decrease as the distance from the source increases. So the light intensity and attractiveness are considered as monotonically decreasing functions. The light intensity is a function of distance (r) and expressed as [126].

$$I(r) = I_o e^{-\gamma r^2} \tag{3.7}$$

Algorithm 5 : Pseudo-code for Firefly algorithm based Spectrum Allocation

Step 1: Initialize the SA algorithm parameters
Initialize number of secondary users (N), number of primary users (K) and number of channels (M), Channel availability matrix $L = \{l_{n,m}|l_{n,m} \in \{0,1\}\}_{N\times M}$, Reward matrix $B = \{b_{n,m}|b_{n,m} \in \{0,1\}\}_{N\times M}$ and Constraint matrix $C = \{c_{n,p,m}|c_{n,p,m} \in \{0,1\}\}_{N\times N\times M}$ and set $L_1 = (n,m)|l_{n,m} = 1$ such that the elements in L_1 are arranged in ascending order with n and m.
Step 2: Initialize control parameters of the Firefly algorithm
Initialize light absorption coefficient γ, attractiveness β, randomization parameter α, maximum number of iterations t_{max}, the number of fireflies NP, dimension $D = \sum_{n=1}^{N}\sum_{m=1}^{M} l_{n,m}$

Step 3: Define objective function $f(\overrightarrow{x}) = MSR, MMR, MPF$, $\overrightarrow{x} = (x_1, x_2, x_3, ..., x_d) = L_1$, Generate the initial location of fireflies x_i $(i = 1, 2, ...NP)$ and set the iteration number $t = 0$.
Step 4:

```
while t ≤ t_max do
  for i=1 to NP //do for each individual sequentially do
    for j=1 to NP //do for each individual sequentially do
      Map the population x_{d,i} to assignment matrix A = {a_{n,m}}
      Evaluate A matrix using the constraint matrix C
      Compute light intensity I_i at x_i is determined by f(x_i) using Equations.(2.3), (2.4),
      (2.5)
       if I_i ≤ I_j, then
      Move firefly i towards j using Equation.(3.9)
      endif
      Evaluate the attractiveness using Equation.(3.8)
      Evaluate new solutions and update light intensity
      Check the bounds of updated solutions
    end for
  end for
  Step 4.1:
  Rank the fireflies and find the current best;
  Check Stopping Criteria
  if t ≤ t_max then
    Increase the iteration t ← t + 1,   go to step 4
  end if
end while
Report results
Terminate
```

where $I(r)$ is the light intensity at a distance r from the source, I_o is the original light intensity at the source, and γ is light absorption coefficient. The firefly's attractiveness β from a distance r is proportional to the light intensity seen by adjacent fireflies.

$$\beta(r) = \beta_o e^{-\gamma r^2} \tag{3.8}$$

where β_o is attractiveness at $r = 0$.

If a firefly i is attracted by the firefly j, then it moves towards firefly j by following the Equation (3.9).

$$x_i = x_i + \beta_o e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha\epsilon_i \tag{3.9}$$

where x_i and x_j are the locations of firefly i and firefly j respectively. α is randomization parameter and ϵ_i is a vector of random numbers with uniform distribution. The pseudo-code to solve the spectrum allocation problem using Firefly algorithm is given in Algorithm 5 .

3.6 Simulation Results

3.6.1 Experimental setup

For simulation, a desktop computer with Intel Core(TM)2 Duo CPU at 3 GHz and 2 GB of RAM is used, and the entire experiment is carried out in the MATLAB environment. To evaluate performance of the algorithm for solving SA problem, the objective functions of SA were setup by assuming that the network is noiseless, and there is a static environment. In this setup, it is considered that the network has N secondary users, K primary users and M channels in the network. Each primary user selects the channel from the available list with a protection range of d_P which is a constant. Each secondary user can adjust its communication range within the bounds of d_{min} and d_{max} to avoid interference between the secondary and primary users.

In simulation, the network parameters are set as N=20, M=20, K=20, d_P=2, d_{min}=1 and d_{max}=5. For this set of parameters, the channel availability, reward, constraint matrices are derived from the pseudo-code of Algorithm 1. For comparison, the SA problem was also solved using PSO, Firefly, and DE algorithms. The parameters of PSO algorithm are defined as follows: Number of particles NP=20, maximum iterations t_{max}=500, weighting coefficients c_1, c_2 to 0.9 and inertial weight ω=0.3. For Firefly, the number of fireflies NP=20, maximum iterations t_{max} =500, randomization parameter α=0.25, attractiveness β=0.2 and light absorption coefficient γ=1. The control parameters of DE algorithm are defined as follows: Population size NP=20, maximum number of iterations G_{MAX}=500, crossover rate Cr=0.9 and weighting factor F=0.9. Using this experimental setup,

all three algorithms were run for 20 independent runs. Three fitness functions MSR, MMR and MPF are optimized individually using PSO, Firefly and DE algorithms.

3.6.2 Results and discussions

The three evolutionary algorithms are executed with the above mentioned experimental setup to solve the SA problem. The three network utility functions are individually optimized using evolutionary algorithms, and the average rewards are plotted as convergence graphs. The convergence graphs under a fixed network topology of N=20, M=20 and K=20 are shown in Figure 3.1, 3.2 and 3.3. These figures correspond to MSR, MMR and MPF objective functions optimized by Firefly, PSO and DE algorithms. In case of MSR and MPF function DE performs the best in terms of converged value, while Firefly performs the best under the objective of MMR. Even though Firefly performs better than PSO and DE in the early stage under objectives MSR and MPF, the converged values after 500 generations by Firefly are still lower than DE. For all three objectives, DE and Firefly algorithms outperform PSO in terms of converged value and convergence rate.

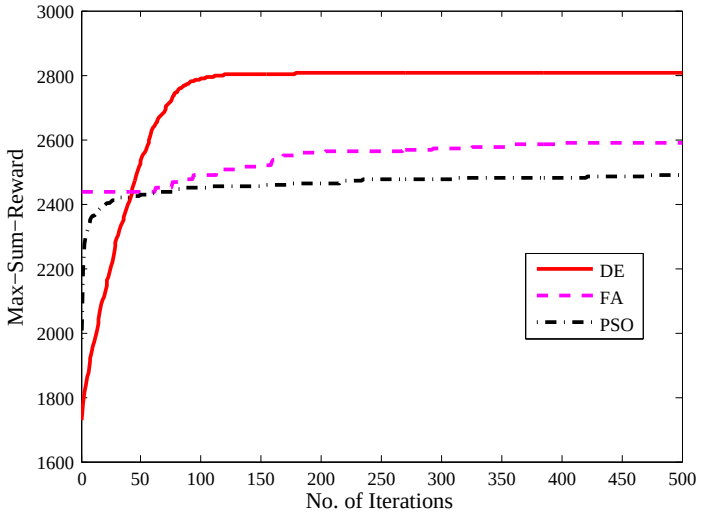

Figure 3.1: Convergence graph (Max-Sum-Reward)

In order to evaluate the convergence speed of the three algorithms, the above mentioned experimental setup is used to solve the SA problem under two different network configurations (N=10, M=10 and K=10, N=5, M=5 and K=5). Figure 3.4 shows the convergence graphs for N=10, M=10 and K=10, corresponding

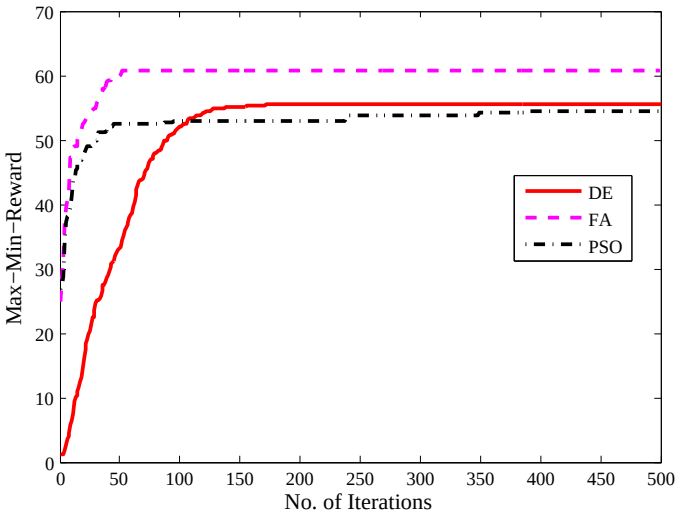

Figure 3.2: Convergence graph (Max-Min-Reward)

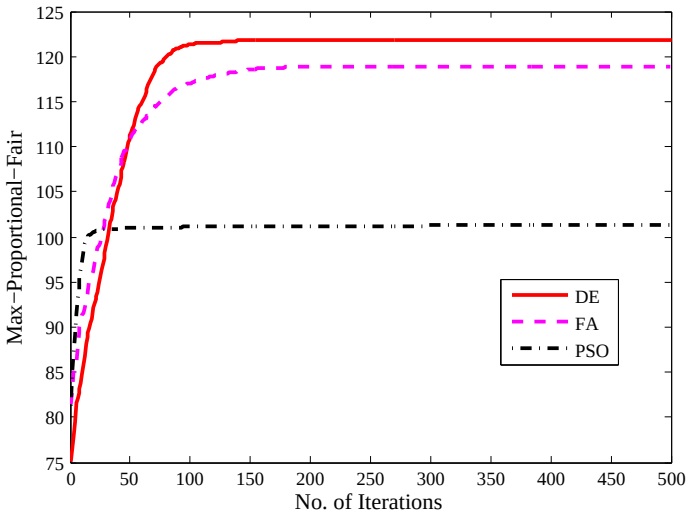

Figure 3.3: Convergence graph (Max-Proportional-Fair Reward)

to three fitness functions (MSR value is divided by number of secondary users). From this graph, it is shown that PSO performs better in the early stage under all objectives, but after 100 generations DE outperforms both Firefly and PSO in optimizing all three objectives. Figure 3.5 shows the convergence graphs for N=5, M=5 and K=5, corresponding to three fitness functions. From all the convergence graphs, it can be concluded that DE performance is superior compared to Firefly and PSO in maximizing the network utility functions.

Next, the performance of PSO, Firefly, and DE algorithms are compared while solving the SA problem by varying the number of users and channels for a given

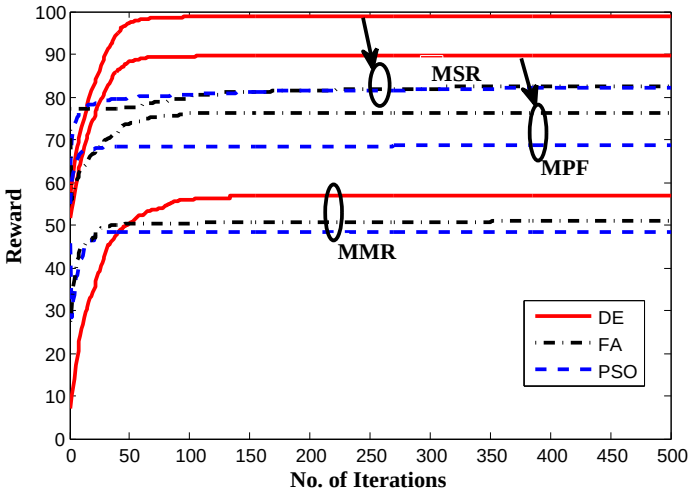

Figure 3.4: Convergence graph (N=10, M=10 and K=10)

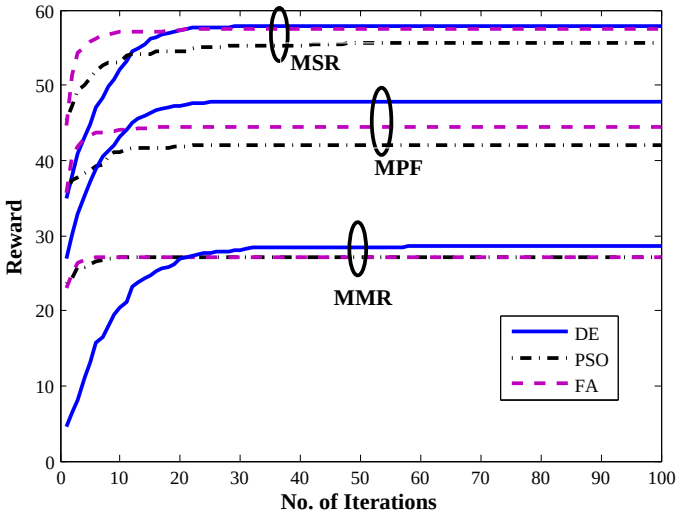

Figure 3.5: Convergence graph (N=5, M=5 and K=5)

CR network. The impact of varying number of secondary users, primary users and channels on the network performance (MSR, MMR and MPF) using DE, Firefly and PSO algorithms are studied. For simulation setup, the network parameters are set as $N = 5$ to 50, $K = 10$, $M = 5$. Figure 3.6 shows that all the three utilization function values decrease with increase in a number of secondary users N. This is because of the constraints between the secondary users to share the channel. Figure 3.7 corresponds to $N = 20$, $M = 5$, K varies between 5 and 50. From this figure, it is observed that as the number of primary users increases there is no spectrum available to secondary users leading to decrease in all the utility

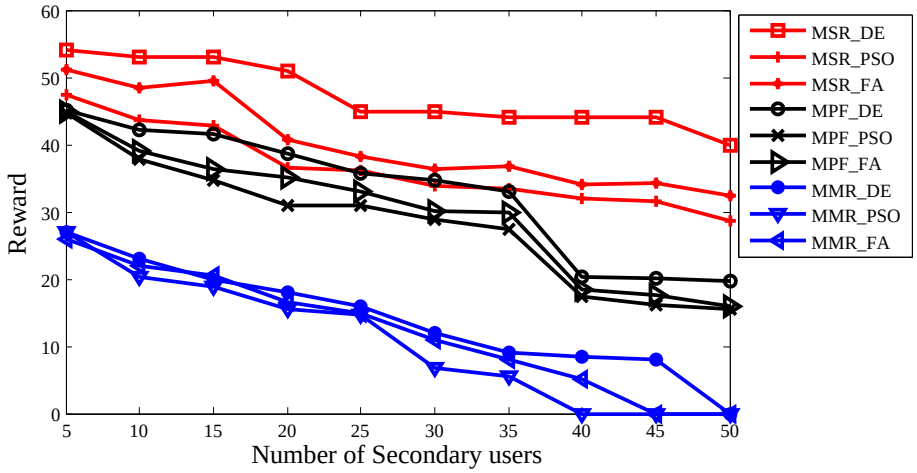

Figure 3.6: Rewards by varying number of secondary users

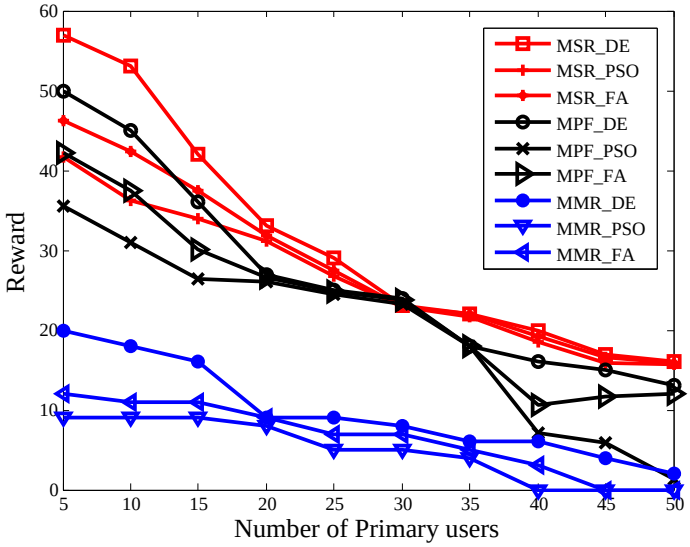

Figure 3.7: Rewards by varying number of primary users

function values. Figure 3.8 corresponds to $K = 20$, $N = 20$ with M varying between 5 to 30. This figure reveals that by increasing the number of channels, more opportunities are available to secondary users leading to increasing in all the three objective function values.

For comparison of performance, all the algorithms (PSO, DE, and Firefly) are executed for 20 independent runs, and the results are tabulated in Table 3.1, corresponding to 20 secondary users, 20 primary users and 20 channels. In this table, Time denotes time complexity, Reward corresponds to mean fitness value and $std\%$ defines standard deviation in percentage of the fitness value over 20 runs. The time complexities of PSO, Firefly, and DE algorithms are evaluated as

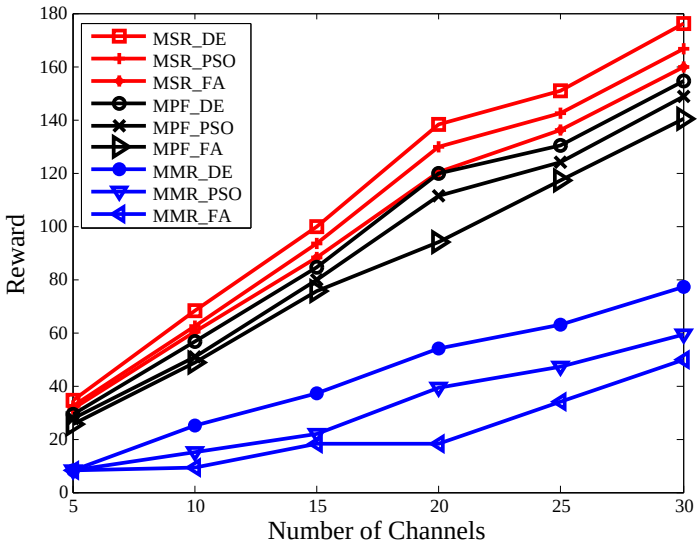

Figure 3.8: Rewards by varying number of channels

given in [128]. The following test code (Algorithm 6) was used for obtaining the time complexity. The time complexity (execution time) T of each algorithm is calculated using:

Algorithm 6 : Test Code

```
for i=1 to 1000
x= double(5.55);
x=x+x; x=x./2; x=x*x; x=sqrt(x); x=log(x);
x=exp(x); y=x/x;
end for
```

$$T = \frac{T_2 - T_1}{T_0} \tag{3.10}$$

where T_2 is the total execution time of the optimization problem, T_1 is the time required for evaluating the objective function alone and T_0 is the execution time of test code. T_1 is obtained by evaluating the objective function for 3000 iterations and T_2 is the total execution time includes 3000 function evaluations. From Table 3.1, it is observed that the DE algorithm improves the quality of solution and robustness in all the three cases. The robustness of the DE algorithm is inferred from the standard deviation of the fitness values. In the case of MPF, the improvement in quality of solution (fitness value) is approximately 19% and 30%, whereas the time complexity is improved by 46% and 242% compared to Firefly

and PSO algorithms respectively. In case of the other two utility functions, DE algorithm also performs better in terms of quality of solution and time complexity compared to PSO and Firefly algorithms. It confirms the superiority of DE algorithm in terms of convergence speed, quality of solution and time complexity with respect to Firefly and PSO algorithms for solving SA problem.

Table 3.1: Performance analysis of Firefly, PSO and DE

Fitness	Firefly			PSO			DE		
Function	Time	Reward	std %	Time	Reward	std %	Time	Reward	std %
MSR	2.908e+3	2563	2.57	6.636e+3	2408	2.65	1.938e+3	2805	2.64
MMR	2.934e+3	60.6	17.65	6.671e+3	18.2	18.34	1.980e+3	56.76	8.29
MPF	2.946e+3	102.4	3.19	6.891e+3	93.8	4.54	2.013e+3	121.90	2.11

3.7 Conclusions

Spectrum allocation (SA) for CR network is computationally complex and NP-hard optimization problem. In this work, the SA problem is solved using three evolutionary algorithms, namely PSO, Firefly and DE. This work is mainly focused on performance evaluation of these algorithms to solve the SA problem in terms of critical characteristics of optimization algorithms such as accuracy, convergence speed, and repeatability. From the simulation results, it is concluded that Firefly algorithm outperforms PSO algorithm in solving the SA problem, because firefly introduces a distance paradigm that has implicit local as well as global search motivation, thereby maintaining divergence. However in PSO, particles are oriented towards a global best particle irrespective of the distance between them. It is also concluded that DE outperforms both Firefly and PSO algorithms, because of its distinct feature of perturbing the current population with the scaled differences of randomly selected distinct population members. From the simulation results, it can be concluded that DE performance is superior to that of PSO and Firefly in providing maximum utilization of network capacity by optimizing MSR, MMR and MPF utilization objective functions and allowing a conflict free channel assignment to secondary users.

Chapter 4

FPGA implementation of Differential Evolution based Spectrum Allocation in Cognitive Radio Networks

Spectrum Allocation is a process of assigning best channels to required secondary users without any interference to primary users. In a distributed approach, each SU device has its embedded platform to perform the SA task. The computational complexity in solving the SA problem increases with the increase in the number of users in a network. However, the SA task needs to be executed in a limited time. Hence, it is required to implement the SA algorithm on a hardware platform like microcontroller, DSP, FPGA or ASIC to accelerate the execution speed of the algorithm. Among these, FPGA platform provides flexibility in customizing the hardware design and also supports processor based approach for developing a system in firmware. This chapter presents FPGA based System on Chip (SoC) implementation of DE-based SA (DE-SA) algorithm. A hardware IP of SA algorithm is developed and interfaced as a coprocessor to the PowerPC 440 processor of Xilinx Virtex-5 FPGA. The execution time of the IP is compared with its equivalent software implementation on PowerPC processor. Further, device utilization and power consumption of the complete DE-SA system is analyzed.

4.1 Introduction

In chapter 3, the SA problem was solved using PSO, Firefly and DE algorithms to maximize the network resource utilization. It was concluded that the DE algorithm performs better in terms of time complexity, convergence speed and quality of the solution. In literature, the SA problem is solved with an assumption that during the spectrum assignment process, the environmental condition of the network remains static [8]. It is also assumed that each secondary user uses a distributed

algorithm to select its channel for communication. In the distributed scheme, a secondary user considers the locally available information from neighborhood users and decides its best channel. The SA problem has been solved by different optimization algorithms using high performance computing platform to maximize the network utilization. However in real-time, SA task has to be performed on an embedded computing platform.

In an embedded platform, execution of an evolutionary algorithm demands more computation time. To meet the real-time execution speed requirement, one can either proceed with parallelization of the algorithm or implement the algorithm in a hardware. There are several hardware platforms such as microcontrollers (μC), digital signal processors (DSP), field programmable gate arrays (FPGA) and application specific integrated circuits (ASIC) for developing an embedded system. Platforms like μC and DSP are revolving around firmware development using software methodologies rather than the development of hardware for an application [80]. FPGA development platform supports both hardware-based approach (a system developed entirely in the hardware) and processor-based approach (a system developed entirely in the firmware) to develop a system. It has the flexibility to customize the hardware design by adding any combination of peripherals and controllers that are not available in microcontroller or DSP processor based system. The execution time of an evolutionary algorithm increases with the increase in complexity of the function to be optimized. Due to this, these algorithms are not suitable for implementation in low-end processors for real-time applications involving complex optimization. Thus, there is a need to define an architecture and implement the algorithm in the FPGA to meet real-time execution speed requirement. Recently, PSO and Genetic algorithms have been implemented on FPGA [129, 130].

Execution of the SA algorithm on embedded processor consumes most of the platform resources, thereby degrades the performance of other primary applications running on it. Thus, a dedicated hardware peripheral for performing the SA task is required. Hence in this chapter, a coprocessor is proposed for performing the SA task to enhance the execution speed of the allocation algorithm. Xilinx Virtex-5 FPGA is chosen as a platform for implementation of the SA task due to its features like improved DSP48E slices for complex math, six input look-up tables (LUTs) and its improved power efficiency. To implement the SA algorithm on FPGA, the critical task is to develop a suitable architecture for the complete

algorithm targeted to FPGA. In the previous chapter, it was concluded that the DE algorithm outperforms both PSO and Firefly algorithms for solving the SA problem by maximizing the network utility functions. Hence, DE-SA algorithm is being chosen for FPGA implementation to accelerate the execution speed. To achieve this, initially a fixed point DE algorithm is implemented on FPGA. Fixed point arithmetic of algorithm is selected due to its advantages of less computational complexity over floating point operations, although floating point can provide more accurate and precise results. This work mainly concentrates on accelerating the execution speed, hence fixed point arithmetic is used to implement in FPGA.

In this chapter, a coprocessor of the fixed point DE algorithm is developed and interfaced with the PPC440 processor of Virtex 5 FPGA as an auxiliary processor unit (APU). Although, there is a flexibility to choose slave interface for the same, the APU interface is chosen because it is more flexible than slave interface. This is because, with the use of extended instructions, the coprocessor can execute an instruction set in parallel with the embedded processor PPC440. The DE coprocessor is validated by optimizing benchmark test functions mentioned in Appendix A.2 [131]. Then the DE-SA IP is developed to solve the SA problem. The network utility functions, namely Max-Sum-Reward (MSR), Max-Min-Reward (MMR) and Max-Proportional-Fair (MPF) are optimized using the DE-SA IP. The IP is interfaced to the PPC440 processor through APU controller and built as a coprocessor to accelerate the execution speed of the SA task.

4.2 Related work

A limited study has been reported on hardware implementation of SA to improve the execution speed and portability of the algorithm. A hardware device for channel allocation was proposed to speed up the channel selection and allocation algorithm with respect to current traffic requirement and interference constraints [17]. It has attained high efficiency in the allocation of available channels (order of nanoseconds). FPGA-based hardware/software co-design architecture of Genetic Algorithm (GA) for reconfiguring cognitive radio parameters was proposed [114]. In [114], fitness module (for obtaining CR parameters) and GA were implemented on a processor and hard-wired blocks of FPGA's fabric respectively. It was reported that the acceleration of execution time by 6μs compared to the execution

time of the same algorithm on processor [114].

In literature, evolutionary algorithms like GA and PSO were implemented in hardware to accelerate the optimization process and achieved an optimal solution. A customized Intellectual Property (IP) of GA was implemented in the Xilinx FPGA and integrated with PPC405 processor based SoC and the speed enhancement up to 5.16x was achieved in Virtex-II Pro development board [130]. A modular co-design architecture was developed for PSO algorithm [129], in which particle positions were updated in hard-wired blocks whereas the fitness function was evaluated on a Nios-II embedded processor. In [129], the design has the flexibility to modify the fitness functions in the software depending on the applications. This approach can be used to develop various embedded applications simply by changing the fitness function. The design achieved a speedup of 20x in Altera development board [129]. Hardware architecture of pipelined PSO (PPSO) was developed along with the parallel PSO framework using multiple Nios-II processors in a System-on-a-programmable-chip (SOPC) platform and resulted speedup of 98x compared to software implementation of the PSO algorithm in Altera development board [132]. A modular, flexible and reusable multi-swarm PSO parallel hardware architecture was proposed to overcome the drawbacks of a software implementation of the PSO algorithm using a Freescale microcontroller and Xilinx MicroBlaze soft processor core [133]. A hardware accelerator for parallel PSO (pPSO) algorithm was reported and validated its performance by optimizing test bench functions on MicroBlaze processor-based SoC in a Virtex-6 development board [134]. Apart from the above works, different variants of PSO algorithms were implemented on FPGA without addressing the acceleration of execution speed [135, 136, 137, 138, 139, 140]. Although different evolutionary algorithms were implemented on FPGA for enhancing the execution speed, there is a limited work in literature that has been reported on implementation of the DE algorithm on FPGA.

DSP algorithms can be implemented in hardware using either fixed point or floating point arithmetic. Fixed point algorithm on hardware performs operations on strictly integer arithmetic only, whereas floating point algorithm on hardware performs operations on both integers and real arithmetic values. In work not presented in this thesis, a floating point DE IP was developed and interfaced to a 32-bit PPC440 (PPC440) processor using processor local bus (PLB) of Virtex-5 FPGA [141]. The developed hardware DE IP was verified by optimizing numeri-

cal benchmark functions and concluded that the DE IP accelerates the execution speed by 200x when compared to its equivalent software implementation on the PPC440. Floating point DE gives better accuracy at the expense of high computation cost. Thus, to reduce the computational complexity, fixed arithmetic is chosen for hardware implementation.

In this chapter, a coprocessor for fixed point DE algorithm is developed and interfaced to PPC440 embedded processor. The coprocessor performance is validated by solving numerical test bench functions. Further, the DE coprocessor is embedded with the three network utility functions, namely MSR, MMR and MPF in the fitness evaluation module. Finally, DE-SA IP is developed to accelerate the execution speed of the SA task.

4.3 FPGA implementation of Differential Evolution algorithm

4.3.1 Software profiling of DE algorithm

It is necessary to find computational intensive functions of the algorithm to implement it in hardware. Computational intensive functions and instructions inside these functions are identified through profiling. This information is used to decide the function(s) that need to be implemented in hardware (logic blocks of the FPGA) or software (runs on the FPGA's embedded processor). DE algorithm has three main modules a) random number generation, b) fitness function evaluation and c) DE algorithm module (i.e., mutation, crossover and selection). The DE algorithm is coded in both fixed and floating point arithmetic in C language to optimize benchmark test functions for profiling as mentioned in Appendix-1. This is termed as software DE algorithm. These software codes are executed in the embedded processor of the Virtex 5 FPGA. In this work, profiling of the DE algorithm is carried on PPC440 embedded processor operating at a clock frequency of 200 MHz and the results are tabulated in Table 4.1. The first column of the table corresponds to different test functions that are optimized using the DE algorithm. The DE algorithmic parameters are considered as maximum generations G_{MAX}=1000, population size NP=8, weighting factor F=0.6, crossover rate Cr=0.9. From this table, it is observed that for optimizing test function $Fun1$,

floating point and fixed point DE algorithm takes 4,721ms (execution time taken by DE algorithm + Objective function + RNG + Float_operations) and 70ms (execution time taken by DE algorithm + Objective function + RNG) respectively. The floating point DE takes more time due to complex floating point operations involved in the algorithm compared to the fixed point arithmetic. All the floating point operations involved in the algorithm are executed in Floating Point Unit (FPU) module of the embedded processor.

Table 4.1: Profiling results of the software (SW) DE algorithm (G_{MAX}=1000, NP=8)

	DE algorithm		Objective	function		RNG	Float_operations
Test Function	SW float (ms)	SW fixed (ms)	SW float (ms)	SW fixed (ms)	SW float (ms)	SW fixed (ms)	SW float (ms)
Fun1	50 (1%)	30 (43%)	10 (0.21%)	10 (14%)	40 (0.85%)	30 (43%)	4,621 (97.88%)
Fun2	60 (1%)	30 (43%)	60 (0.81%)	10 (14%)	40 (0.54%)	30 (43%)	7,228 (97.83%)
Fun3	90 (2%)	30 (38%)	10 (0.19%)	10 (13%)	50 (0.93%)	40 (50%)	5,220 (97.20%)
Fun4	50 (1%)	40 (44%)	20 (0.26%)	20 (22%)	40 (0.51%)	30 (33%)	7,674 (98.58%)
Fun5	1,488 (5%)	1,245 (69%)	294 (0.97%)	255 (14%)	520 (1.72%)	303 (17%)	27,944 (92.38%)
Fun6	1,824 (6%)	1,330 (29%)	3,640 (11%)	2,983 (64%)	570 (1.72%)	334 (7%)	27,042 (81.75%)

In the Table 4.1, the value inside parenthesis refers to % of the total execution time of a particular module requires during execution. For low dimension functions like $Fun1$, the execution time of the random number generator (RNG) module is comparable with DE algorithm module, whereas the objective function module takes less time compared to other two modules. Hence, to accelerate the execution speed of the total algorithm, both DE algorithm and RNG modules need to be executed on a hardware. If the objective function is implemented in the processor and the other modules are implemented in the hard-wired blocks then the on-chip bus transaction time between these modules will degrade the acceleration. To reduce the bus overhead, the DE algorithm module, RNG module and the fitness evaluation module are embedded into a single hardware IP.

4.3.2 Proposed hardware architecture of DE algorithm

The proposed DE IP core has seven main modules i.e., Memory initialization, Mutation, Crossover, Selection, Random Number Generator, Fitness evaluation and a Control Finite State Machine (FSM) module to synchronize all the six modules as shown in Figure 4.1. The FSM has idle, initialization, operation, waiting and reading states as shown in Figure 4.2. In the idle state all the modules are in the reset condition. In the initialization state, the FSM initializes the population and

fitness memories when the inputs i.e., maximum number of generations G_{MAX} and population size NP are available at the initialization module. During the operation state, control FSM enables internal modules according to the different stages of the algorithm, i.e., mutation, crossover and selection. FSM will be in a wait state until the execution of current module is completed else it will go to the next module for execution. In the reading state, FSM reads the fitness value and writes into an output register.

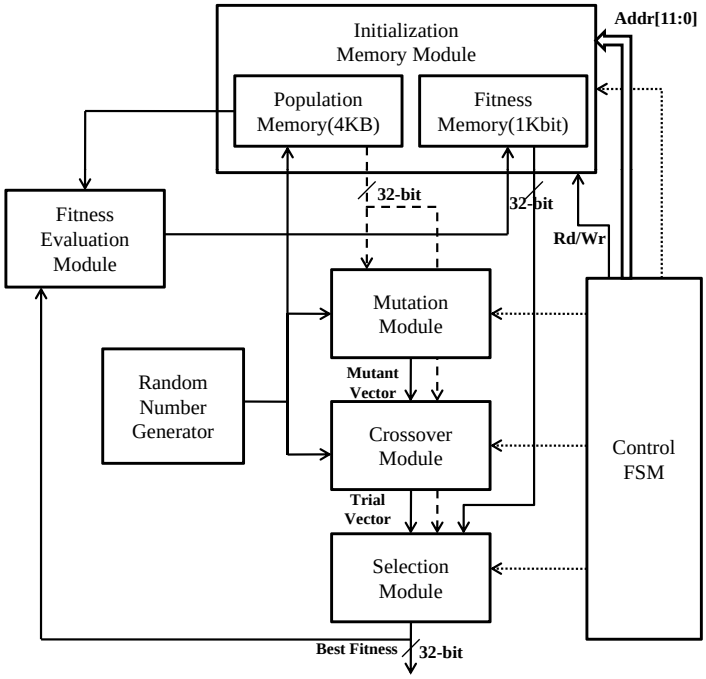

Figure 4.1: Differential Evolution IP Core

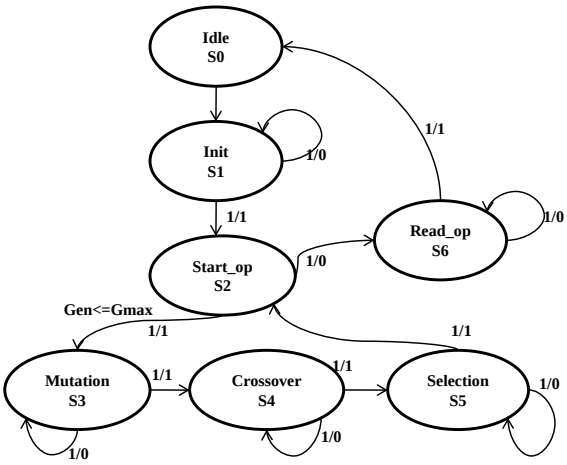

Figure 4.2: Finite State Machine diagram of DE algorithm

4.3.2.1 Memory initialization module

The memory module has two separate memories, for storing the population (Population Memory) and their fitness values (Fitness Memory). During the initialization state, population of size $NP \times D$ are randomly generated within the range of $[X_{min}, X_{max}]$, and stored in the population memory of size 4KBytes. The population members are accessed from the population memory using a 12-bit address. Each population member is of size 128 Bytes, and the maximum values of NP and D are set to 32. These values are input to the fitness evaluation module. After evaluating the fitness function, the fitness values (each of size 32-bit) are stored in the fitness memory of size 1 Kbits. This process is repeated for NP times.

4.3.2.2 Mutation module

After the initialization state, mutation operation is performed by the mutation module. In this module, a mutant vector is generated for each i^{th} target vector from the current population. Three distinct vector indices in the range of 1 to NP are generated by comparing the counter value with a value in the register. The register value contains a randomly generated number multiplied by population size. These indices are connected to the select lines of a multiplexer. Three distinct target vectors are obtained from the output of a multiplexer as shown in the Figure 4.3 and these are stored in Reg-files A, B and C, each of size 128 Bytes. The mutation operation is performed by the difference of any two of these registers scaled by a factor F and this difference is added to the third register to obtain the mutant vector. In this module, the mutant vector (128 Bytes) is generated for all the dimensions of each population member.

4.3.2.3 Crossover module

The crossover operation is mainly responsible for increasing the diversity among the mutant vectors. A trial vector is generated in the crossover module as shown in Figure 4.4. Each i^{th} trial vector is generated with a crossover rate Cr. The registers Reg 1 and Reg 2 have a randomly generated number $rand \times D$. The input mutant vector index is 5-bit, and it is padded with zeros of 27-bits to make it as 32-bit input. The trial vector (32-bit) is generated using the multiplexer logic as shown in Figure 4.4. The crossover constant controls the diversity of the

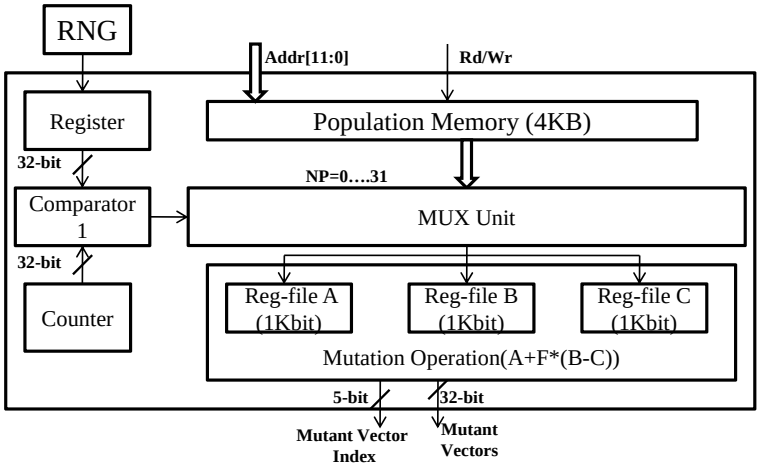

Figure 4.3: Mutation module

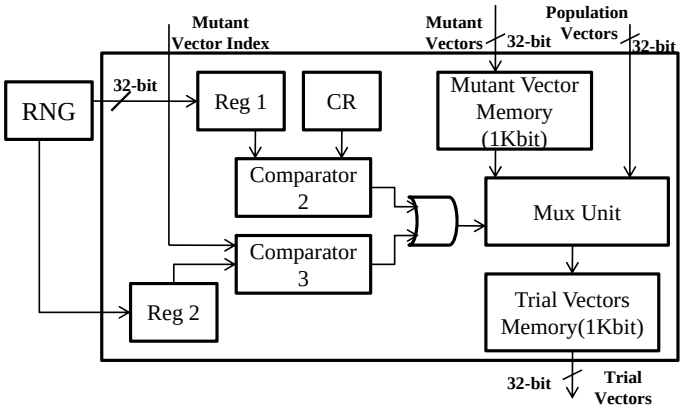

Figure 4.4: Crossover module

population and makes the algorithm escape from local optima, and ensures that the trial vector gets at least one vector from the mutant vector.

4.3.2.4 Selection module

The trial vector generated from the previous module is input to the selection module as shown in Figure 4.5. The fitness value of a trial vector is evaluated and if it is less than the fitness of the current population member then it selects the input as a trial vector otherwise it selects the current population member as the new population member. The output of the multiplexer (MUX) is the updated value of the current population memory. This process is repeated for all the generations to improve the fitness of individuals till the maximum number of generations is reached.

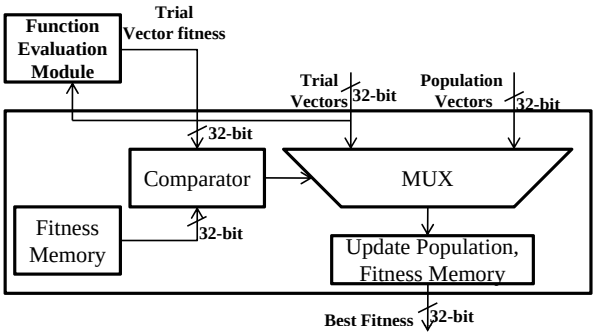

Figure 4.5: Selection module

4.3.2.5 Fitness evaluation module

The Fitness module evaluates the fitness of individual member in accordance with the defined fitness functions. Here different test bench functions like three variable Sphere function (Fun3), four variable variably dimensioned function (Fun4) and 32 variable De Jong (Fun5) and schwefel's function (Fun6) are considered to verify the performance of DE hardware. The fitness of each population and the population members for the complete population is evaluated and stored in the fitness memory module. The fitness module is designed according to the fitness function whereas the rest of the system remains unchanged.

4.3.2.6 Random Number Generator (RNG) module

Random number generation module plays an importance role in the DE algorithm. In this work, Linear Feedback Shift Register (LFSR) is used to generate random numbers, as it is easy to implement and produces fairly good pseudo-randomness. It generates the random numbers for the initial population module, mutation, crossover, and selection modules. The bit selection for mutation is also carried out using this module. The seed for the random number generator is programmable. It enables different convergence characteristics for each generation. A non-zero value is assigned for seed as an initial value, before starting the process. If all zero value appears in seed, then XOR operations continue to generate zeros and output becomes always zero. The circuit diagram for 32-bit LFSR with maximum length polynomial $X^{32} + X^{22} + X^2 + X^1 + 1$ is shown in Figure 4.6, and it generates $2^{32} - 1$ random outputs, which are large enough for this application.

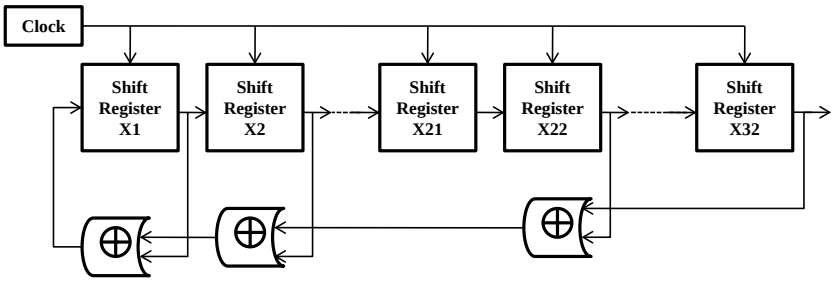

Figure 4.6: Circuit diagram of 32-bit LFSR

4.4 FPGA implementation of DE based Spectrum Allocation algorithm

4.4.1 Software profiling of SA algorithm

In the previous section profiling of both fixed point and floating point DE algorithm for solving benchmark test function targeted to PPC440 embedded processor is presented. The previous chapter concluded that the DE algorithm is a promising algorithm for solving SA problem in cognitive radio. Thus, the present section focuses on FPGA implementation of DE-SA algorithm to accelerate the execution speed. Furthermore, DE-SA algorithm is implemented on the Xilinx Virtex 5 FPGA based system on chip platform, that provides design flexibility, adaptability and ease in integration of other custom IPs. As mentioned in the earlier section, profiling of a signal processing algorithm is essential prior to its implementation in FPGA. Thus, this section presents the software profiling analysis of both fixed and floating point DE-SA algorithm on Xilinx PPC440 processor. Since the clock frequency of the target development board is 200MHz, the profiling was carried out for the clock frequency of 200 MHz.

In the SA technique the objective functions like Maximum-Sum-Reward, Max-Min-Reward and Max-Proportional-Fair are optimized using DE algorithm. In real-time, these objective functions are fixed during the allocation process. In this study, the number of secondary users, primary users and available spectrum bands, each is fixed to 20. Profiling is carried out for optimizing the Maximum-Sum-Reward (MSR), and the results are tabulated in Table 4.2. The DE algorithmic parameters are selected as G_{MAX}=300, NP=20, F=0.6 and Cr=0.9. From this table, it is observed that the floating point DE algorithm (algorithmic steps, function evaluation) takes more execution time compared to the fixed point

DE algorithm, as obvious. The execution time of floating point operations dominates the total execution time. To eliminate the computational intensive floating point operations, the algorithm is implemented in fixed point arithmetic. From this table, it is also observed as the number of users and available band increases the computational complexity of fitness evaluation module increases. The overall execution time consumed by the DE algorithm is 98%, out of which DE algorithmic steps like mutation, crossover and selection (25.42%) and fitness function evaluation (62.98%) consumes significant percentage of time.

Table 4.2: Profiling results of the software (SW) SA algorithm (% of Execution time)

	Fixed			Float					
Test Function	DE	Objective Function	RNG	DE	Objective Function	RNG	Div	floatsidf	unpack_f
MSR ($5 \times 5 \times 5$)	41.57	38.80	17.26	2.92	2.81	1.33	35.38	26.81	6.06
MSR ($20 \times 20 \times 20$)	25.42	62.98	10.06	3.57	2.84	1.35	38.11	23.97	5.36

Thus, to accelerate the total execution speed, the fitness evaluation module need to be implemented in the hardware due to its inherent advantages mentioned earlier. If the fitness function alone is implemented in the hardware and the other functionalities in the embedded processor, then the bus transaction overhead between fitness function module in hardware and DE algorithm (except fitness evaluation) in software will dominate the acceleration. This lead to degrade the acceleration performance. To avoid the performance degradation, the complete DE algorithm (including utility functions) need to be implemented as a single hardware module.

In this chapter, hardware-software co-design platform is used to study the effect on execution speed of SA algorithm. In the first case (Case-I), DE algorithm is implemented in software (on embedded processor) and fitness function (MSR) is implemented in hardware. In the second case (Case-II), DE algorithm is implemented in hardware and fitness function (MSR) is implemented in software. Later in the third case (Case-III), DE algorithm and fitness evaluation modules are implemented in hardware. System on Chip platform is developed for the above three cases individually, and the timing results are tabulated in Table 4.3. This table correspond to the network parameters as 5 secondary users, 5 primary users and 5 available channels ($5 \times 5 \times 5$). The bus transaction time to pass the values from hardware to software and vice-versa to evaluate the fitness function is 161.51ms (NP x G_{MAX} x (each bus transaction time 50μs)) whereas overall execution time

of the algorithm is 516ms and 408ms (for NP=32 and G_{MAX}=100) in Case-I and Case-II respectively.

Table 4.3: Average execution time of SA task in Hardware-software co-design platform for $(5 \times 5 \times 5)$

		NP=16			NP=32		
Test Function	G_{MAX}	Case-I Time(ms) (Std%)	Case-II Time(ms) (Std%)	Case-III Time(ms) (Std%)	Case-I Time(ms) (Std%)	Case-II Time(ms) (Std%)	Case-III Time(ms) (Std%)
	1	4.4	1.95	0.8	8.7	3.74	1.6
		(0.8)	(0.6)	(0.7)	(0.6)	(0.4)	(0.4)
	50	131	102	22	260	204	43
		(0.5)	(0.3)	(0.3)	(0.3)	(0.2)	(0.2)
MSR	100	259	205	43	516	408	85
		(0.6)	(0.4)	(0.2)	(0.2)	(0.5)	(0.2)
	300	771	615	129	1538	1225	254
		(0.6)	(0.2)	(0.1)	(0.3)	(0.6)	(0.1)

However, pure hardware implementation of DE-SA algorithm (Case-III) consumes only 85ms, due to the exclusion of bus transaction time and objective function evaluation time.

4.4.2 Hardware architecture of DE-SA IP

The proposed hardware architecture of DE-SA is shown in Figure 4.7. It consists of seven main modules, i.e., Memory initialization, Mutation, Crossover, Selection, Random Number Generator, SA fitness evaluation module and a Control Finite State Machine (FSM) Module to synchronize all six modules as explained in Section 4.3.2. In this IP, the fitness evaluation module consists of three network utilization functions, namely MSR, MMR and MPF are embedded on it. The utilization function is selected by the user input ($User\_Sel$) to DEMUX logic as shown Figure 4.7. During the fitness evaluation, the selected utilization function results are stored in the fitness memory.

The network data matrices (L, B and C) and the current DE population matrix, that are used to compute the assignment and reward matrices are stored in the internal memory. These matrices are given as inputs to the fitness module for calculating the fitness value. The FSM has idle, initialization, operation, waiting and reading states. In the idle state all the modules are in the reset condition. In the initialization state the FSM enables memory module when the inputs such as maximum number of generations G_{MAX}, population size NP, dimension D, crossover rate Cr, scaling factor F, number of secondary users N, number of pri-

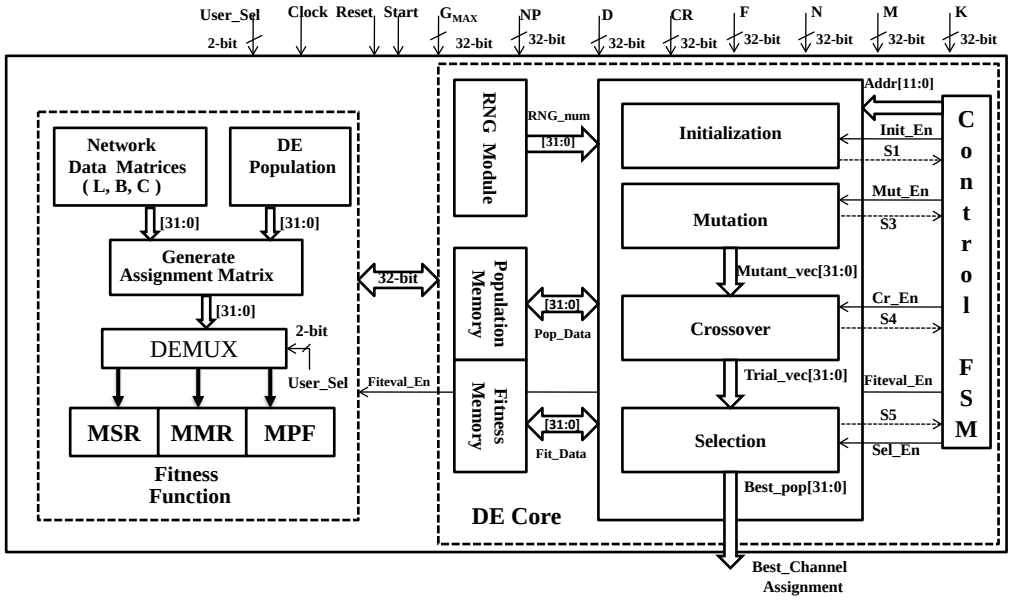

Figure 4.7: Hardware architecture of DE based Spectrum Allocation

mary users K, number of channels M and choice of utilization function $User\_Sel$ are available to the DE-SA module. During the operation state, control FSM enables internal modules according to the different stages of the algorithm, i.e., crossover, mutation and selection. FSM will be in the wait state until the execution of current module else it will go to the next module for execution. In the reading state, FSM will read the fitness value and write into the output register.

4.4.2.1 Max-Sum-Reward (MSR)

MSR module is used to calculate the total sum reward of the network. The architecture details of MSR module is shown in Figure 4.8. For each population, the channel availability matrix A and channel reward matrix (B) are input to the fitness function. These two matrices are given to a multiplier. Essentially this is a multiply-accumulate unit. The final accumulated result is stored in the 32-bit register ($Reg1$).

4.4.2.2 Max-Min-Reward (MMR)

This module is used to calculate the minimum reward of the cognitive user in the network. The same A and B matrices are given as input to the multiplier and adder. The reward value of each secondary user for the selected channels is stored in $RegFile1$ of 1Kbit size. Minimum value of $RegFile1$ is obtained using a

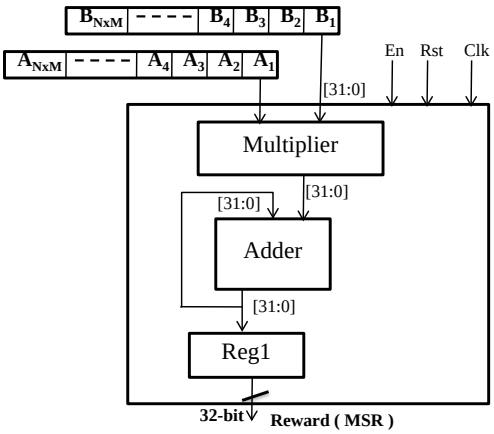

Figure 4.8: Max-Sum-Reward module

comparator and Mux logic. The resulted MMR reward value is stored in a 32-bit register ($Reg2$) as shown in Figure 4.9.

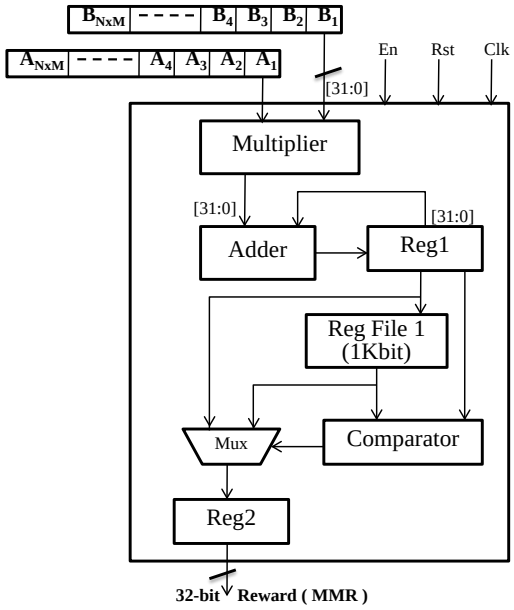

Figure 4.9: Max-Min-Reward module

4.4.2.3 Max-Proportional-Fair (MPF)

MPF module is used to calculate the fairness reward of the network. The architecture for implementing the MPF utility function is shown in Figure 4.10. The same A and B matrices are input to the multiplier and adder module of the circuit. For each secondary user, reward value is calculated for the selected channels, and it is stored in $Reg2$. This reward value is again multiplied by the reward value of

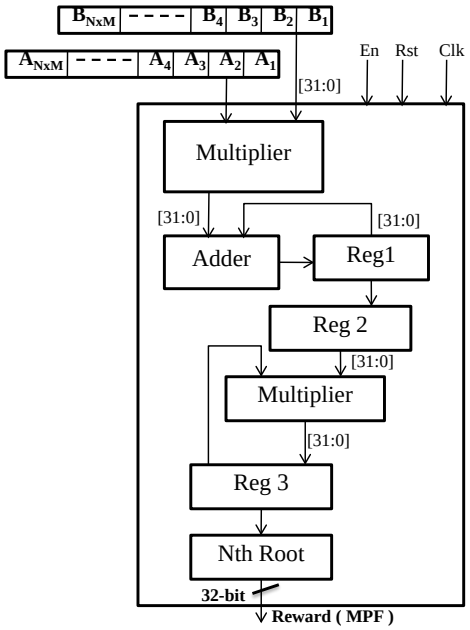

Figure 4.10: Max-Proportional-Fair module

another secondary user, and the result is stored in $Reg3$. This process continues for N times, where N is the number of secondary users. Subsequently the $Reg3$ value is input to N^{th} Root module to calculate the fair reward of the network.

4.4.3 System on Chip (SoC) implementation

In general, System on Chip implementation of the hardware is carried out for validating the prototype of a system before building the ASIC chip. SoC integrates processors, peripherals, memories, custom IP components into a single chip. The FPGA-based SoC offers hardware re-use, easier programmability to the user for developing complex systems. The proposed SoC platform uses Virtex-5 FPGA that includes PPC440 embedded processor, peripherals such as UART (RS-232) for serial communication, DDR2-SDRAM, BRAM, Timer and Interrupt modules [142]. Timer and Interrupt modules are used for software profiling.

In this work, at the first stage DE IP is developed as a Fabric Coprocessor Module and it is connected to the PPC440 processor via Fabric coprocessor bus as shown in Figure 4.11. Then the DE coprocessor is invoked using Load and Store instructions of PPC440 processor. The functionality of DE IP is validated by optimizing the benchmark test functions. In addition to the functional results, timing result and convergence of the algorithm are also observed. In the later

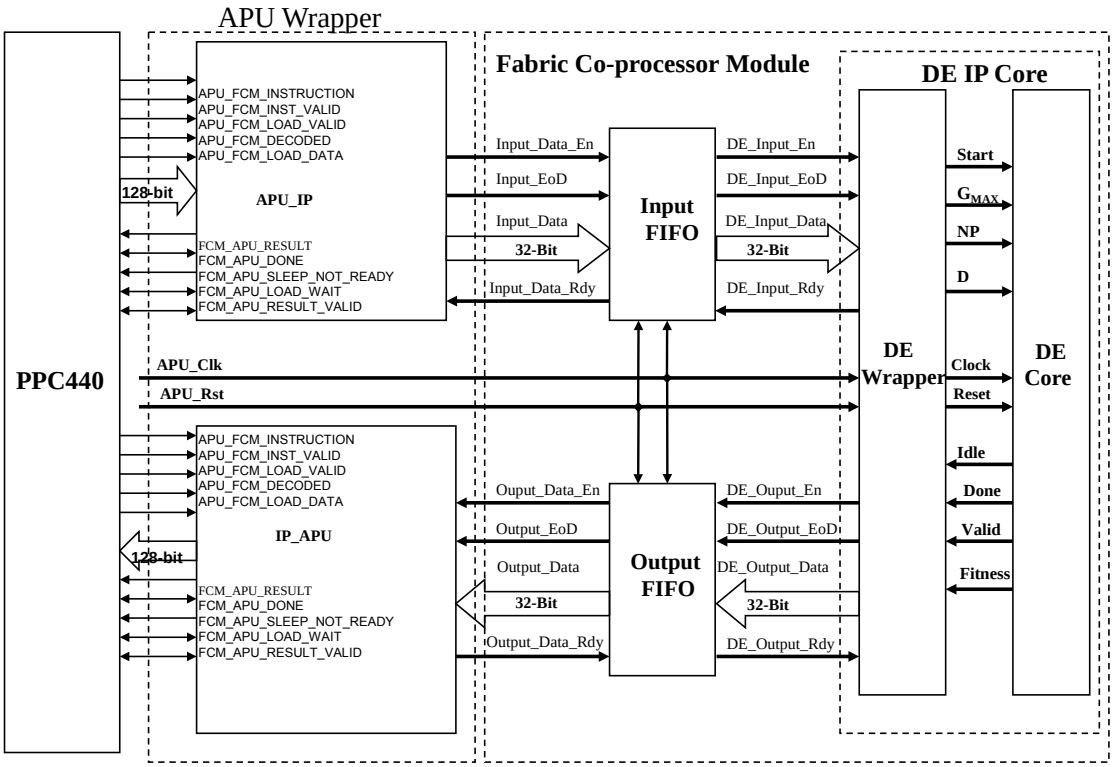

Figure 4.11: APU interface diagram of DE IP core

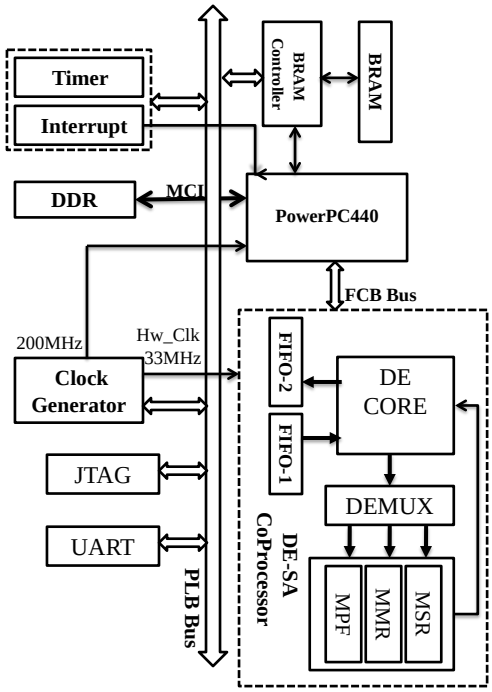

Figure 4.12: System-on-Chip setup for DE-SA

stage, the DE coprocessor for benchmark test function optimization is replaced with DE-SA coprocessor. The corresponding SoC architecture for implementing DE-SA IP is shown in Figure 4.12. In this figure, MSR, MMR, and MPF corresponds to the hardware module for evaluating three fitness functions, namely Max-Sum-Reward, Max-Min-Reward and Max-Proportional-Fair respectively. In this design, a hardcore PPC440 is used because of its natural advantages over soft-core processor along with its effective resource utilization capability. Two asynchronous FIFOs (depth of 10 and width of 32 bits) are used at the interface (input and output) of the DE-SA core. The input signal is processed as a stream of nine samples G_{MAX}, NP, D, Cr, F, $User_Sel$, N, M and K through the first FIFO ($FIFO-1$), whereas the output signal from the core is acquired through the other FIFO ($FIFO-2$). These FIFOs are interfaced with the PPC440 processor.

Table 4.4: Control parameters of the DE algorithm

Control Parameters	Value
Population Size (NP)	8,16,32
Maximum Number of Iterations(G_{MAX})	1,50,100,300
Weighting Factor (F)	0.6
Crossover rate (Cr)	0.9

4.5 Experimental setup

In the first stage, the DE algorithm is coded using fixed point C code and ported on to the PPC440 processor. The DE algorithmic parameters are tabulated in Table 4.4. The DE IP core is designed using Verilog hardware description language. The developed DE IP core is implemented and tested on a Xilinx Virtex-5 Development board (XC5VFX70T-1136). The developed DE IP core is interfaced to an auxiliary processing unit controller. Finally, a coprocessor for accelerating DE algorithm is developed. Initially, it was used for optimizing benchmark test functions. For functional verification of the DE IP core along with wrapper logic, a test bench was simulated as shown in Figure 4.13. The results shown in this figure corresponds to the function Fun6 with $G_{MAX} = 1$, $NP = 8$ and $D = 4$. It is not the optimized solution because the simulation is carried out for one iteration only. From this figure, it is ascertained that the resultant fitness value is available at DE_Output_Data port, when DE_Output_Rdy signal is logic high. It is also

evident that the IP core takes 0.08 mega clock cycles for completing one iteration with above control parameters. The optimum solution obtained for minimizing Fun6 using the DE core is "3" as expected. This is shown in Figure 4.14. It confirms the proper functionality of the DE IP core. In the next stage, the same experiment was conducted for DE-SA algorithm.

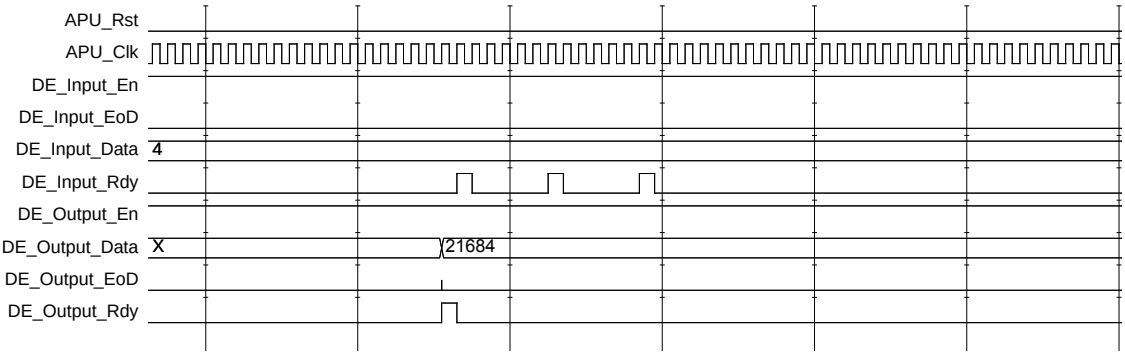

Figure 4.13: Functional simulation of DE IP core

The DE-SA algorithm is ported onto the PPC440 using 32-bit fixed point C code for software implementation. For a hardware implementation, the proposed algorithm is coded using HDL and an Intellectual Property (IP) is developed. Then the IP core is simulated and tested using Xilinx ISE and EDK 10.1.3 platform on Virtex-5 FPGA development board. The IP core frequency is set to 63.55 MHz, which is the maximum frequency obtained during implementation. The developed coprocessor is scalable (in terms of the number of secondary users, primary users, number of channels and other DE algorithmic parameters such as population size, number of generations and dimension) and these are set as user inputs to the processor. After receiving these inputs, DE-SA IP is invoked using Load/Store API call from the PPC440. Then the IP completes its execution and the channel availability matrix, and the optimum fitness values are monitored through UART module.

4.6 Results and analysis

4.6.1 Timing results

Initially, the complete DE optimization algorithm is ported to the PowerPC processor of Xilinx Virtex-5 FPGA for software implementation, and then the com-

plete DE algorithm is executed using the DE coprocessor. The execution time of the DE algorithm for different population sizes (8, 16, 32) and three different generations (1, 50, 100) is evaluated for 20 independent runs. The acceleration factor (AF) of the coprocessor with respect to software floating and fixed point execution time are tabulated as AF (float) and AF (fixed) respectively. The values in parenthesis refer to the percentage of standard deviation of execution time.

Table 4.5: Average execution time of the DE algorithm implemented on PPC440 processor (software)

		NP =8			NP=16			NP=32		
Test Function	G_{MAX}	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor
Fun1	1	4.91 (3.2)	0.15 (2.8)	32.73	9.44 (2.5)	0.26 (2.2)	36.31	18.36 (1.4)	0.52 (1.2)	35.31
	50	181.05 (1.4)	5.38 (0.9)	33.65	332.37 (0.7)	9.69 (0.4)	34.30	641.81 (0.4)	18.82 (0.2)	34.10
	100	363.17 (1.1)	10.38 (0.5)	34.99	673.21 (1.4)	19.33 (0.4)	34.83	1,301.32 (1.1)	37.51 (0.2)	34.69
Fun2	1	8.01 (1.9)	0.18 (2.2)	44.50	15.02 (1.5)	0.31 (2.3)	48.45	28.73 (0.8)	0.61 (1.2)	47.10
	50	264.53 (1.4)	5.97 (0.9)	44.31	491.39 (0.9)	10.85 (0.4)	45.29	940.12 (0.7)	19.82 (0.2)	47.43
	100	536.05 (1.5)	11.96 (0.5)	44.82	994.24 (0.9)	21.64 (0.3)	45.94	1,897.45 (0.7)	39.26 (0.2)	48.33
Fun3	1	5.12 (1.3)	0.16 (2.7)	32.00	10.13 (1.9)	0.31 (1.3)	32.68	19.88 (0.8)	0.61 (0.6)	32.59
	50	199.54 (0.8)	5.83 (0.6)	34.23	371.94 (0.4)	11.08 (0.3)	33.57	720.03 (0.2)	21.64 (0.2)	33.27
	100	397.91 (0.8)	11.62 (0.5)	34.24	740.14 (0.3)	22.08 (0.3)	33.52	1,432.64 (0.2)	43.16 (0.2)	33.19
Fun4	1	9.99 (1.9)	0.23 (2.2)	43.43	19.36 (0.9)	0.45 (1.2)	43.02	38.38 (0.5)	0.84 (0.6)	45.69
	50	305.79 (0.6)	7.08 (0.4)	43.19	584.59 (0.3)	13.48 (0.2)	43.37	1,145.25 (0.1)	26.46 (0.2)	43.28
	100	612.92 (0.7)	14.11 (0.3)	43.44	1,178.46 (0.5)	26.94 (0.2)	43.74	2,304.13 (0.3)	52.77 (0.2)	43.66
Fun5	1	41 (1.7)	6 (1.6)	6.83	81 (1.2)	11 (1.5)	7.36	162 (1.2)	23 (1.5)	7.04
	50	1,132 (1.3)	207 (1.7)	5.47	2,234 (2.1)	411 (1.6)	5.44	4,439 (2.1)	809 (1.4)	5.49
	100	2,254 (0.8)	412 (0.9)	5.47	4,435 (0.9)	825 (2.1)	5.38	8,809 (1.1)	1,638 (2.1)	5.38
Fun6	1	85 (1.8)	15 (1.9)	5.67	170 (2.1)	30 (2.3)	5.67	339 (1.1)	62 (2.2)	5.47
	50	2,251 (1.2)	446 (1.1)	5.05	4,472 (1.5)	884 (1.7)	5.06	8,916 (2.3)	1,736 (2.1)	5.14
	100	4,476 (1.3)	891 (2.1)	5.02	8,745 (1.3)	1,764 (1.1)	4.96	18,483 (0.9)	3,537 (1.1)	5.23

Table 4.5 shows the average execution time (in msec) and percentage of standard deviation (Std%) of execution time for both arithmetics of DE algorithm implemented on the PPC440 processor (SW). The average execution time and standard deviations tabulated in Table 4.5 are for 20 independent runs. The table show results with maximum iterations $G_{MAX} = 1, 50, 100$ and for different pop-

ulation sizes $NP = 8, 16, 32$. It reveals that for optimizing high dimension test functions, fixed point algorithm gives approximately 4.96 - 7.36x acceleration over the floating point algorithm implemented on PPC440 processor. For optimizing low-dimensional functions, the execution time is 32 - 48.45x faster compared to the floating point implementation. The average execution time of the algorithm using the coprocessor is tabulated in Table 4.6 and is referred as (HW) time.

Table 4.6: Average execution time of DE coprocessor and its acceleration factor (AF) over Floating and Fixed point software execution time

		NP =8			NP=16			NP=32		
Test Function	G_{MAX}	HW(ms) (Std%)	AF Float	AF Fixed	HW(ms) (Std%)	AF Float	AF Fixed	HW(ms) (Std%)	AF Float	AF Fixed
Fun1	1	0.05 (1.1)	98.20	3.00	0.1 (1.4)	94.40	2.60	0.2 (1.0)	91.80	2.60
	50	2.13 (1.2)	85.00	2.53	3.9 (0.7)	85.22	2.48	7.6 (0.2)	84.45	2.48
	100	4.27 (1.0)	85.05	2.43	8.21 (1.5)	82.09	3.94	15.2 (0.2)	85.61	2.47
Fun2	1	0.05 (1.5)	160.20	3.60	0.1 (1.3)	150.20	3.10	0.2 (1.4)	143.65	3.05
	50	2.23 (1.2)	118.62	2.68	4.06 (0.5)	121.03	2.67	7.8 (0.3)	120.53	2.54
	100	4.43 (0.7)	121.00	2.70	8.11 (0.5)	122.59	2.67	15.6 (0.2)	121.63	2.52
Fun3	1	0.07 (1.1)	73.14	2.29	0.13 (1.8)	77.92	2.38	0.26 (1.3)	76.46	2.35
	50	2.6 (1.1)	76.75	2.24	4.9 (0.5)	75.91	2.26	9.57 (0.3)	75.24	2.26
	100	5.3 (1.1)	75.08	2.19	9.9 (0.6)	74.76	2.23	19.06 (0.2)	75.16	2.26
Fun4	1	0.08 (1.6)	124.88	2.88	0.15 (1.9)	129.07	3.00	0.3 (0.9)	127.93	2.80
	50	2.9 (1.2)	105.44	2.44	5.4 (0.7)	108.26	2.50	10.5 (0.3)	109.07	2.52
	100	5.8 (1.2)	105.68	2.43	10.9 (0.7)	108.12	2.47	21.1 (0.6)	109.20	2.50
Fun5	1	0.5 (0.8)	82.00	12.00	0.8 (0.3)	101.25	13.75	1.6 (0.2)	101.25	14.38
	50	11.9 (0.2)	95.13	17.39	23.5 (0.1)	95.06	17.49	46.6 (0.1)	95.26	17.36
	100	23.7 (0.1)	95.11	17.38	46.7 (0.1)	94.97	17.67	92.6 (0.1)	95.13	17.69
Fun6	1	0.6 (0.6)	141.67	25.00	1.2 (0.3)	141.67	25.00	2.3 (0.2)	147.39	26.96
	50	16.4 (0.4)	137.26	27.20	32.4 (0.1)	138.02	27.28	64.5 (0.1)	138.23	26.91
	100	32.6 (0.5)	137.30	27.33	64.4 (0.1)	135.79	27.39	128 (0.1)	144.40	27.63

From this table it is observed that the coprocessor execution time is up to 73.14 - 160.20x faster than the software execution time for floating point DE algorithm. In contrast, it is only 2.19 - 27.63x faster compared to fixed point DE algorithm. Further, it is observed that for lower dimension functions coprocessor acceleration AF (fixed) is small as compared to higher dimension functions. It also reveals that the execution time of the coprocessor for different functions is scaling up with the

population size and maximum number of generations.

The execution time for DE-SA algorithm on PPC440 processor of Xilinx Virtex-5 FPGA is referred as software (SW) execution time whereas the time for executing the SA task using the DE-SA IP is referred as hardware (HW) execution time. The average execution time of the DE-SA implemented on PPC440 processor for optimizing the three utility functions are tabulated in Table 4.7. The tabulated results correspond to the mean execution time of 20 independent runs. Table 4.7 correspond to the network parameters as 5 secondary users, 5 primary users and 5 available channels ($5 \times 5 \times 5$). In the table, Std % refers to the standard deviation of the execution time over 20 independent runs. It is observed that the acceleration factor remains almost constant between 11 - 19x with the increase in population size and network parameters.

Table 4.7: Average execution time of Spectrum Allocation problem in Float and Fixed arithmetic (software) for ($5 \times 5 \times 5$)

		NP =8			NP=16			NP=32		
Test Function	G_{MAX}	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor
	1	42 (0.5)	2.2 (0.8)	19.09	82 (0.7)	4.3 (0.8)	19.07	163 (0.4)	8.3 (0.5)	19.64
	50	1,068 (0.3)	63 (0.3)	16.95	2,011 (0.3)	125 (0.4)	16.09	4,113 (0.1)	248 (0.3)	16.58
MSR	100	2,008 (0.2)	127 (0.2)	15.81	4,052 (0.1)	251 (0.3)	16.14	7,820 (0.1)	497 (0.2)	15.73
	300	5,786 (0.4)	381 (0.1)	15.19	11,276 (0.4)	757 (0.1)	14.90	22,364 (0.2)	1,498 (0.1)	14.93
	1	36 (0.2)	2.4 (0.1)	14.85	69 (0.1)	4.6 (0.6)	15.58	139 (0.3)	9.1 (0.5)	15.92
	50	1,109 (0.3)	67 (0.3)	16.11	2,235 (0.2)	132 (0.4)	16.38	4,492 (0.4)	263 (0.1)	16.52
MMR	100	2,133 (0.4)	134 (0.5)	15.53	4,267 (0.5)	265 (0.2)	15.67	8,842 (0.3)	528 (0.5)	16.88
	300	6,235 (0.1)	402 (0.2)	15.41	12,267 (0.3)	797 (0.1)	15.49	24,359 (0.2)	1,590 (0.1)	15.81
	1	38 (0.3)	2.5 (0.6)	12.34	73 (0.4)	4.5 (0.2)	12.87	145 (0.6)	8.5 (0.5)	13.47
	50	1,039 (0.2)	64 (0.1)	11.58	2,054 (0.3)	128 (0.5)	11.57	4,094 (0.4)	253 (0.3)	11.47
MPF	100	2,045 (0.5)	127 (0.3)	11.43	4,027 (0.2)	254 (0.6)	11.71	8,090 (0.2)	503 (0.1)	11.81
	300	6,085 (0.4)	384 (0.2)	11.52	11,925 (0.4)	762 (0.3)	11.56	24,755 (0.1)	1,590 (0.4)	11.54

It is also observed that the fixed point implementation of the algorithm in processor speeds up the design by same amount irrespective of the algorithmic complexity of the DE algorithm. This observation is true for all the three utility functions for all the generation numbers. The robustness of the implementation is supported by the low standard deviation of the execution speed over many independent runs. Similar kind of analysis is carried for the execution speed of

the algorithm by increasing the network parameters. Table 4.8 correspond to the study of execution speed for network parameter as $(10 \times 10 \times 10)$. Table 4.9 correspond to the study of execution speed for network parameter as $(20 \times 20 \times 20)$. As the network parameter increases, the complexity of SA task increases. From both the above tables, it is evident that (a) although the execution time increases in individual cases, but the acceleration factor remains almost constant with the increase in the network complexity, and (b) the fixed point implementation is the optimum choice for this application.

Table 4.8: Average execution time of Spectrum Allocation problem in Float and Fixed arithmetic (software) for $(10 \times 10 \times 10)$

		NP =8			NP=16			NP=32		
Test Function	G_{MAX}	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor
MSR	1	150 (0.3)	10.1 (0.8)	14.85	296 (0.5)	19 (0.7)	15.58	589 (0.2)	37 (0.4)	15.92
	50	4,286 (0.3)	266 (0.6)	16.11	8,583 (0.3)	524 (0.3)	16.38	17,083 (0.2)	1,034 (0.2)	16.52
	100	8,326 (0.1)	536 (0.4)	15.53	16,598 (0.3)	1,059 (0.4)	15.67	35,012 (0.5)	2,074 (0.2)	16.88
	300	24,868 (0.2)	1,614 (0.1)	15.41	49,823 (0.2)	3,216 (0.2)	15.49	100,154 (0.3)	6,335 (0.2)	15.81
MMR	1	156 (0.4)	12 (0.3)	19.09	309 (0.6)	23 (0.5)	19.07	615 (0.5)	44 (0.6)	19.64
	50	3,713 (0.5)	305 (0.1)	16.95	7,712 (0.1)	608 (0.3)	16.09	15,646 (0.3)	1,178 (0.2)	16.58
	100	7,123 (0.3)	610 (0.4)	15.81	14,632 (0.2)	1,206 (0.4)	16.14	30,059 (0.4)	2,375 (0.1)	15.73
	300	20,765 (0.3)	1,832 (0.2)	15.19	42,658 (0.5)	3,650 (0.2)	14.90	87,630 (0.2)	7,147 (0.3)	14.93
MPF	1	157 (0.6)	11 (0.3)	12.34	312 (0.4)	20 (0.5)	12.87	620 (0.5)	42 (0.3)	13.47
	50	3,824 (0.2)	270 (0.2)	11.58	7,674 (0.4)	540 (0.3)	11.57	15,421 (0.4)	1,060 (0.2)	11.47
	100	7,485 (0.3)	543 (0.2)	11.43	14,988 (0.2)	1,080 (0.1)	11.71	30,012 (0.5)	2,130 (0.3)	11.81
	300	22,455 (0.3)	1,650 (0.1)	11.52	44,964 (0.3)	3,296 (0.5)	11.56	90,038 (0.2)	6,550 (0.1)	11.54

Subsequently, the hardware IP of the SA algorithm is verified as shown in the Figure 4.12. The hardware execution time (HW) for optimizing three utility functions are measured using the timer. The execution time for three different network parameters setting $(5 \times 5 \times 5$, $10 \times 10 \times 10$ and $20 \times 20 \times 20)$ is tabulated in Table 4.10, 4.11 and 4.12 respectively.

The acceleration due to hardware implementation of the task is compared with the equivalent floating point and fixed point software implementation. The acceleration factor in the case of floating point software implementation is calculated as:

$$AF(Float) = Float\ \ SW/Fixed\ \ HW \tag{4.1}$$

Table 4.9: Average execution time of Spectrum Allocation problem in Float and Fixed arithmetic (software) for $(20 \times 20 \times 20)$

		NP =8			NP=16			NP=32		
Test Function	G_{MAX}	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Fixed SW(ms) (Std%)	Acceleration factor
	1	654 (0.5)	53 (0.4)	12.34	1,287 (0.3)	100 (0.3)	12.87	2,586 (0.7)	192 (0.4)	13.47
	50	15,028 (0.4)	1,298 (0.4)	11.58	29,925 (0.2)	2,586 (0.1)	11.57	58,974 (0.2)	5,140 (0.1)	11.47
MSR	100	29,890 (0.2)	2,614 (0.4)	11.43	60,120 (0.3)	5,135 (0.3)	11.71	120,036 (0.4)	10,161 (0.1)	11.81
	300	91,086 (0.2)	7,906 (0.3)	11.52	180,210 (0.1)	15,583 (0.4)	11.56	359,860 (0.3)	31,189 (0.2)	11.54
	1	564 (0.6)	63 (0.4)	19.09	1,120 (0.5)	120 (0.7)	19.07	2,220 (0.3)	227 (0.4)	19.64
	50	14,320 (0.5)	1,550 (0.2)	16.95	28,370 (0.4)	3,065 (0.5)	16.09	56,210 (0.4)	6,002 (0.3)	16.58
MMR	100	29,355 (0.5)	3,072 (0.3)	15.81	58,160 (0.2)	6,100 (0.2)	16.14	115,720 (0.2)	11,880 (0.4)	15.73
	300	87,480 (0.3)	9,170 (0.2)	15.19	179,319 (0.5)	18,180 (0.4)	14.90	356,685 (0.1)	35,110 (0.3)	14.93
	1	610 (0.3)	52 (0.7)	14.85	1,210 (0.6)	98 (0.7)	15.58	2,410 (0.3)	192 (0.5)	15.92
	50	16,920 (0.4)	1,294 (0.5)	16.11	33,440 (0.1)	2,550 (0.2)	16.38	66,210 (0.4)	5,021 (0.4)	16.52
MPF	100	33,320 (0.3)	2,594 (0.2)	15.53	65,870 (0.3)	5,112 (0.2)	15.67	131,070 (0.3)	9,966 (0.1)	16.88
	300	98,620 (0.4)	7,810 (0.2)	15.41	194,300 (0.3)	15,510 (0.3)	15.49	385,345 (0.2)	30,062 (0.4)	15.81

Table 4.10: Acceleration Factors of DE-SA coprocessor over DE-SA software (Float and Fixed) for $(5 \times 5 \times 5)$

		NP =8			NP=16			NP=32		
Test Function	G_{MAX}	HW(ms) (Std%)	AF Float	AF Fixed	HW(ms) (Std%)	AF Float	AF Fixed	HW(ms) (Std%)	AF Float	AF Fixed
	1	0.4 (0.3)	105.00	5.50	0.8 (0.7)	102.50	5.38	1.6 (0.4)	101.88	5.19
	50	11 (0.3)	97.09	5.73	22 (0.3)	91.41	5.68	43 (0.2)	95.65	5.77
MSR	100	22 (0.2)	91.27	5.77	43 (0.2)	94.23	5.84	85 (0.2)	92.00	5.85
	300	66 (0.1)	87.67	5.77	129 (0.1)	87.41	5.87	254 (0.1)	88.05	5.90
	1	0.3 (0.8)	88.24	5.94	0.6 (0.6)	87.06	5.59	1.1 (0.6)	87.91	5.52
	50	11 (0.4)	97.41	6.05	21 (0.3)	98.66	6.02	43 (0.3)	99.32	6.01
MMR	100	23 (0.4)	95.70	6.16	45 (0.3)	96.50	6.16	89 (0.3)	102.98	6.10
	300	65 (0.2)	94.92	6.16	129 (0.1)	96.74	6.24	263 (0.1)	98.97	6.26
	1	0.3 (0.7)	80.74	6.54	0.6 (0.6)	82.50	6.41	1.1 (0.7)	84.79	6.30
	50	12 (0.5)	77.46	6.69	22 (0.4)	77.73	6.72	45 (0.3)	76.79	6.69
MPF	100	23 (0.4)	77.64	6.79	43 (0.3)	78.90	6.74	87 (0.2)	79.08	6.69
	300	66 (0.2)	78.79	6.84	134 (0.2)	79.14	6.84	258 (0.1)	79.76	6.91

Similarly, in the case of fixed-point software implementation AF is calculated and tabulated in Table 4.10. This analysis for acceleration is carried out for all the

Table 4.11: Acceleration Factors of DE-SA coprocessor over DE-SA software (Float and Fixed) for $(10 \times 10 \times 10)$

		NP =8			NP=16			NP=32		
Test Function	G_{MAX}	HW(ms) (Std%)	AF Float	AF Fixed	HW(ms) (Std%)	AF Float	AF Fixed	HW(ms) (Std%)	AF Float	AF Fixed
MSR	1	1.7 (0.5)	88.24	5.94	3.4 (0.3)	87.06	5.59	6.7 (0.2)	87.91	5.52
	50	44 (0.4)	97.41	6.05	87 (0.2)	98.66	6.02	172 (0.1)	99.32	6.01
	100	87 (0.3)	95.70	6.16	172 (0.3)	96.50	6.16	340 (0.1)	102.98	6.10
	300	262 (0.1)	94.92	6.16	515 (0.2)	96.74	6.24	1,012 (0.1)	98.97	6.26
MMR	1	1.4 (0.4)	105.00	5.50	2.8 (0.3)	102.50	5.38	6.1 (0.2)	101.88	5.19
	50	43 (0.5)	97.09	5.73	86 (0.5)	91.41	5.68	171 (0.3)	95.65	5.77
	100	86 (0.3)	91.27	5.77	171 (0.3)	94.23	5.84	342 (0.3)	92.00	5.85
	300	263 (0.1)	87.67	5.77	516 (0.2)	87.41	5.87	1,020 (0.2)	88.05	5.90
MPF	1	1.6 (0.5)	80.74	6.54	3.1 (0.4)	82.50	6.41	6.2 (0.3)	84.79	6.30
	50	44 (0.3)	77.46	6.69	87 (0.3)	77.73	6.72	172 (0.2)	76.79	6.69
	100	87 (0.2)	77.64	6.79	173 (0.2)	78.90	6.74	343 (0.2)	79.08	6.69
	300	264 (0.1)	78.79	6.84	517 (0.1)	79.14	6.84	1,018 (0.1)	79.76	6.91

Table 4.12: Acceleration Factors of DE-SA coprocessor over DE-SA software (Float and Fixed) for $(20 \times 20 \times 20)$

		NP =8			NP=16			NP=32		
Test Function	G_{MAX}	HW(ms) (Std%)	AF Float	AF Fixed	HW(ms) (Std%)	AF Float	AF Fixed	HW(ms) (Std%)	AF Float	AF Fixed
MSR	1	8.1 (0.3)	80.74	6.54	15.6 (0.2)	82.50	6.41	30.5 (0.2)	84.79	6.30
	50	194 (0.2)	77.46	6.69	385 (0.1)	77.73	6.72	768 (0.1)	76.79	6.69
	100	385 (0.4)	77.64	6.79	762 (0.1)	78.90	6.74	1,518 (0.1)	79.08	6.69
	300	1,156 (0.2)	78.79	6.84	2,277 (0.2)	79.14	6.84	4,512 (0.1)	79.76	6.91
MMR	1	8.0 (0.2)	105.00	5.50	15.5 (0.2)	102.50	5.38	30.4 (0.2)	101.88	5.19
	50	195 (0.3)	97.09	5.73	386 (0.4)	91.41	5.68	772 (0.4)	95.65	5.77
	100	387 (0.1)	91.27	5.77	767 (0.6)	94.23	5.84	1,525 (0.3)	92.00	5.85
	300	1159 (0.1)	87.67	5.77	2,292 (0.2)	87.41	5.87	4,523 (0.2)	88.05	5.90
MPF	1	7.9 (0.3)	88.24	5.94	15.3 (0.2)	87.06	5.59	30.3 (0.2)	87.91	5.52
	50	194 (0.2)	97.41	6.05	387 (0.1)	98.66	6.02	770 (0.1)	99.32	6.01
	100	386 (0.2)	95.70	6.16	764 (0.2)	96.50	6.16	1,520 (0.2)	102.98	6.10
	300	1,158 (0.1)	94.92	6.16	2,289 (0.1)	96.74	6.24	4,525 (0.1)	98.97	6.26

utility functions with different population size and number of generations. The standard deviation (std%) of execution time for 20 independent runs is also tabulated in the Table 4.10. From this table, it is observed that DE-SA hardware IP gives an acceleration of 87.41 - 105x and 5.19 - 5.9x over float and fixed software implementation while optimizing the MSR utility function. The acceleration factor is almost same with the increase in population size and number of generations. The acceleration of the hardware IP over fixed point software implementation is due to the parallelization in FPGA. A significant acceleration $\sim$ 100x is observed over floating point software implementation. Similar trend in acceleration factor is observed while optimizing other two, i.e., MMR and MPF utility functions. The same analysis is performed by changing the network parameters as $(10 \times 10 \times 10)$, $(20 \times 20 \times 20)$ and the results are tabulated in Table 4.11 and 4.12. By comparing the Table 4.10, 4.11 and 4.12 it is observed that with the increase in the network complexity, individually the hardware IP and the processor takes more execution time to complete the SA task, however the overall acceleration factor remains almost constant. The low standard deviation concludes about the robustness of the implementation.

4.6.2 Convergence results

The convergence graph of DE algorithm for Fun2 test function is shown in Figure 4.14. This compares the convergence results of fixed point DE algorithm implemented in embedded processor (SW) and coprocessor (HW). It is observed that both the implementation gives approximately same quality of solution and convergence rate. It proves the correctness of the developed hardware IP. The DE-SA algorithm is implemented in the FPGA as shown in the Figure 4.12. The fitness values, i.e., the converged values of the utility functions are compared with the results obtained from the software, i.e., PowerPC processor. The convergence results of the three utility functions for different network configurations (i.e; $5 \times 5 \times 5$, $10 \times 10 \times 10$ and $20 \times 20 \times 20$) are shown in Figure 4.15, 4.16 and 4.17 respectively.

In these figures, (HW) and (SW) correspond to the results obtained using the DE-SA hardware IP (coprocessor) and the processor respectively. The results shown are the mean results obtained for 20 independent runs. From the Figure 4.15, it is observed that at the beginning of generation/iteration both SW and HW gives same value due to the same seed for generating random numbers, and

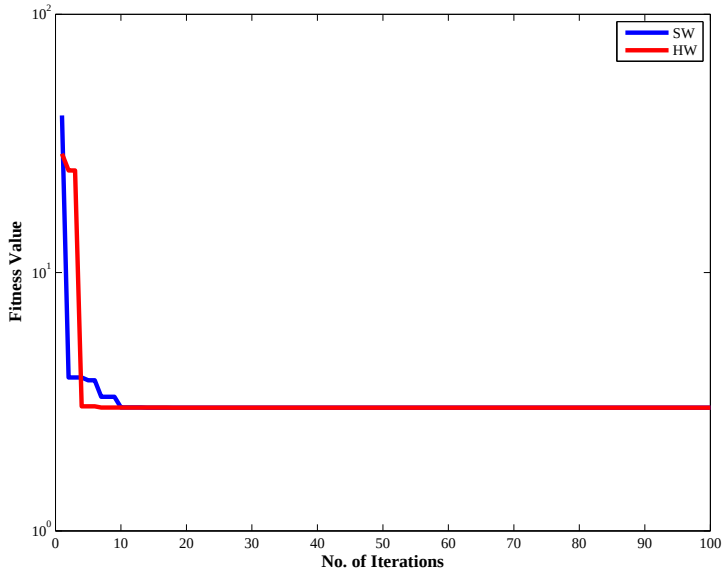

Figure 4.14: Convergence graph of Fun2 test function in hardware and software

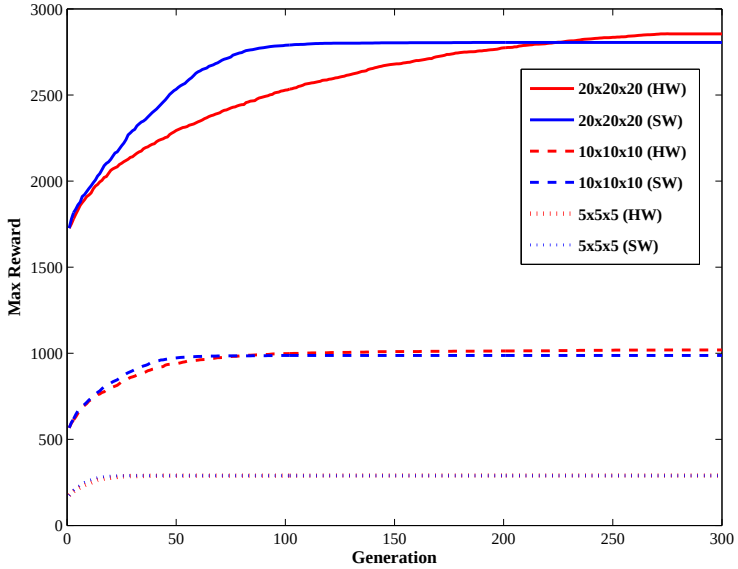

Figure 4.15: Convergence graph of MSR

afterwards there is a slight variation in the reward values (fitness values). The variation is because they generate different random numbers with iteration. It affects the performance of DE algorithm and its convergence. Similarly, Figure 4.16 and Figure 4.17 analyze the convergence behavior of the HW and SW for optimizing the Max-Min-Reward (MMR) and Max- Proportional-Fair reward (MPF) utility functions. The convergence graphs conclude that the SW and the developed IP functionally behaves almost the same.

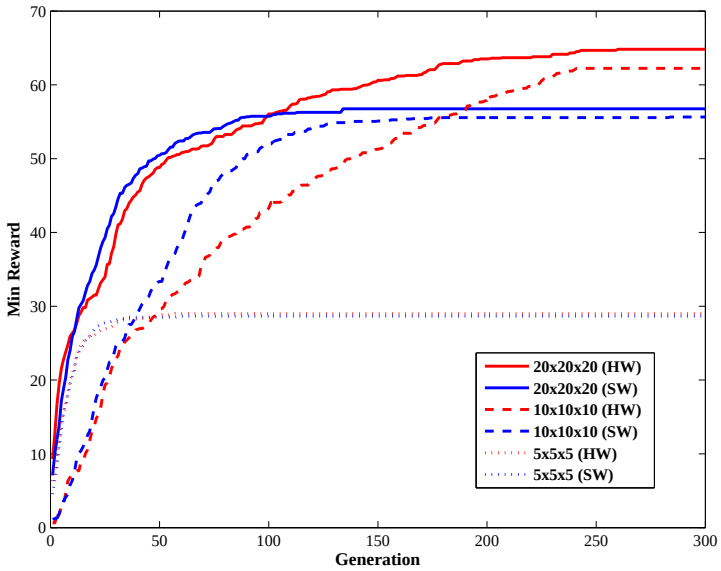

Figure 4.16: Convergence graph of MMR

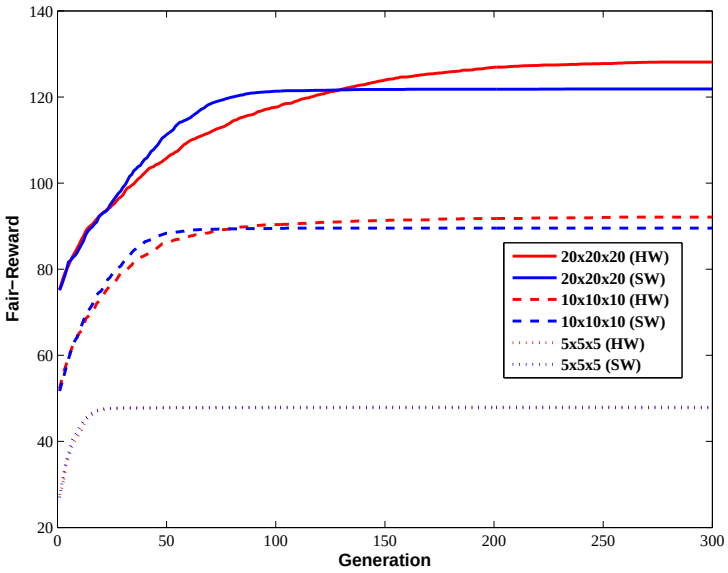

Figure 4.17: Convergence graph of MPF

4.6.3 Synthesis results

The DE-SA IP is built by interconnecting the multiple modules like DE IP and network utility functions. This IP is parameterized in terms of population size (NP), dimension (D), maximum number of generations (G_{MAX}), crossover rate (Cr), weighting factor (F), number of secondary users (N), number of primary users (K) and number of channels (M). Table 4.13 shows Xilinx Synthesis Tool (XST) syn-

thesis results (resource utilization) of individual utility functions namely, MSR, MMR, MPF and the complete DE-SA IP that includes the DE algorithm and the three network utility functions. The three network utility function modules require almost similar amount of resources. The resource utilization for the DE-SA IP (61% of BRAM, 62% of DSP48E, 13% of Slice registers, 22% of LUTs and 35% of Slices) is tabulated in the same table and observed that BRAM and DSP48E utilization is more due to computations like multiplication, division operations.

Table 4.13: Resource utilization of MSR, MMR, MPF and $DE-SA$

Test Function	BRAM	DSP48E	Slice Registers	Slice LUTs	Slices	LUT FF pairs
MSR	4(2%)	5(3%)	918(2%)	1370(3%)	545(4%)	586(34%)
MMR	4(2%)	5(3%)	793(1%)	1167(2%)	458(4%)	534(37%)
MPF	4(2%)	5(3%)	922(2%)	1365(3%)	534(4%)	613(36%)
$DE-SA$	92(61%)	80(62%)	6603(13%)	10008(22%)	3971(35%)	3851(56%)

The power analysis of the DE-SA IP is performed using XPower Analyzer tool of Xilinx ISE 10.1. Table 4.14 shows the power consumed by each resource block of the FPGA (Xilinx Virtex-5) along with its resource utilization for this application. The hierarchal power analysis of the developed DE-SA System on Chip (SoC) system is tabulated in Table 4.15. From this table, it is observed that the proposed IP consumes 26.53mW i.e., 17% of total power (152.66mW) consumed by the complete SoC system. Furthermore, device utilization of the total SoC system along with the customized DE-SA IP is tabulated in Table 4.16. Resources like Bonded IOBs, BUFG and PLL_ADVs are used only in the SoC system but not in the IP. This table reveals that DSP48E slices are used by the IP only. Hence, all the computational task is performed in the IP core. The achieved maximum operating frequency of the core is 63.5 MHz although the complete SoC system can work at 200MHz frequency.

Table 4.14: Power analysis of system

Resource	Power consumed (mW)	Used	Total available	% utilization
Clock	120.03	12	-	-
Logic	1.54	10112	44800	22.6
Signals	4.17	18946	-	
IOs	3298.22	233	722	32.3
BRAMs	21.81	90	148	58
DSPs	1.28	70	128	54.7
PPC440	44.74	1	1	100
PLL	69.24	2	6	33.3

Table 4.15: Hierarchy power analysis of $DE - SA$ SoC system

Resource Type	Power(mW)
Total SoC system	152.66
PowerPC440	44.99
Clockgen	74.91
APU	26.53
DDR2	5.66
SysACE_Compact Flash	0.29
RS232_Uart_1	0.13
PLB	0.02
proc_sys_reset_0	0.04
xps_timer_0	0.03
xps_timer_1	0.03
xps_intc_0	0.02
xps_bram_cntlr	0.01
xps_bram	0.00

Table 4.16: Device utilization of system and Core

Resource	Total SoC	Core
Slice Registers	8564(19%)	4700(10%)
Slice LUTs	11141(24%)	7859(17%)
LUT FF-Pairs	3295(23%)	2402(20%)
BRAM	88(59%)	84(56%)
DSP48E	70(54%)	70(54%)
Bonded IOBs	54(8%)	-
BUFG	9(32%)	-
PLL_ADVs	2(33%)	-
Max Freq	200 MHz	63.549 MHz

4.7 Conclusions

In this chapter, a hardware based solution for the SA problem in CR networks is proposed. Initially, a scalable hardware IP with APU interface is developed as a coprocessor for accelerating execution speed of the DE algorithm. To avoid bus overhead, the complete DE algorithm with fitness evaluation module is implemented in the hardware instead of partitioning the design into software and hardware. To validate the performance of the coprocessor, six numerical test-bench functions are optimized and compared execution time in both software and hardware implementations. The experimental results have shown that an acceleration of approximately by 25 - 27.63x and 135.79 - 147.39x is attained while optimizing a 32 dimension Fun6 complex test function compared to the fixed and floating point software implementation respectively. For optimizing less complex fitness functions i.e., Fun1 the coprocessor attained speedup of approximately by

2.43 - 3.94x over fixed point and 82.09 - 98.20x over floating point software implementation respectively.

Subsequently, the DE-SA IP is developed by integrating network utility functions to DE IP and verified its functionality on Xilinx Virtex-5 FX70T FPGA based system on chip platform. The DE-SA IP is interfaced to the PPC440 hard-core processor via auxiliary processor unit controller. It is scalable in terms of a number of primary users, secondary users, and available channels. The acceleration factor is evaluated in different network and algorithmic configurations, it is observed an acceleration of 5.19 - 6.91x and 76.79 - 105x over the fixed and floating point implementation of the SA algorithm on the PPC440 processor respectively. Resource utilization of the IP core is also reported. Maximum operating frequency and the power consumption of the IP are observed as 63.55 MHz and 26.53 mW respectively.

Chapter 5

Spectrum Allocation using Multi-Objective Differential Evolution algorithm and its FPGA implementation

In CR Network, spectrum holes are assigned to SUs using spectrum allocation (SA) algorithm by optimizing multiple objectives to provide best channels without interference to primary users. In this chapter, forced termination probability, and network utility functions (i.e., Max-Sum-Reward, Max-Min-Reward and Max-Proportional-Fair) are simultaneously optimized to provide best channels to SUs. Thus, the SA problem is formulated as a multi-objective optimization problem consisting of the above mentioned four objective functions and solved using Non-dominated Sorting Genetic Algorithm II (NSGA-II) and Multi-Objective Differential Evolution (MODE) algorithms. Further, to improve the efficiency of channel assignment solution, a joint spectrum and power allocation algorithm is used to maximize both individual and total network capacity. Finally, MODE-based SA (MODE-SA) algorithm is implemented on a SoC embedded platform to enhance the execution speed of the algorithm.

5.1 Spectrum Allocation in Cognitive Radio Networks using Multi-Objective Differential Evolutionary algorithm

5.1.1 Introduction

In chapter 4, DE-based SA IP was implemented on a FPGA and shown that the hardware IP accelerated the execution speed of SA algorithm compared to the processor implementation. In Chapter 3, the network utility functions were optimized individually to maximize the network resource utilization. This model is

valid if the quality of service (QoS) depends on only one of the utility functions. However, in reality more number of conflicting functions are associated with quality of service. To find a set of trade-off solutions, all the utility functions are need to be optimized simultaneously. In the CR operation, it is possible that a PU will arrive during a SU's operation in a vacant primary user band. It leads to forced termination of SU with a probability known as forced termination probability. It degrades the SU network capacity. In a situation, when all the channels are occupied by either SU or PU and a SU request for a channel then the requested SU will be blocked with a probability known as blocking probability. It also leads to degrade the SU network capacity. However in this work, network capacity is analyzed with respect to forced termination probability only using Markov models. Forced termination probability is considered as a performance indicator for SA technique. There exist many studies that independently addresses technique to optimize the utility functions and analyze the traffic model separately. Zhu et. al presented a Markov model to analyze the spectrum access for cognitive users in licensed bands [14]. Blocking probability, forced termination probability, and traffic throughput are formulated as performance indicators of the allocation scheme. In literature, the forced termination probability is analyzed after completion of the allocation process [15]. However, the forced termination probability depends on the allocation results, hence it is necessary to take the termination probability into account during the allocation process. Thus in this work, the forced termination probability is considered as one more objective function along with the three utility functions described in Chapter 3 and are optimized simultaneously using multi-objective optimization algorithms.

In literature, different evolutionary algorithms such as NSGA-II [143], PSO [144] and DE [145] are used for solving multi-objective optimization problems. NSGA-II is a popular technique with the advantages of fast nondominated sorting procedure and an elitist strategy to solve multi-objective problems [143]. DE is a powerful and simple evolutionary algorithm proposed by Storn and Price [112] and successfully demonstrated for solving single-objective optimization problems. DE technique is extended to solve multi-objective optimization problems and proved its superiority compared to other algorithms [146, 147, 148, 149, 150]. MODE algorithm with non-dominated sorting [151] technique finds a true pareto optimal solution [152] and provides better accuracy compared to NSGA-II [153]. In a multi-user and multi-relay cooperative cellular network, SA is formulated as a

multi-objective optimization problem that optimizes the network throughput and transmitted power of each user simultaneously [154]. In a centralized wireless sensor networks, the SA problem was solved using bi-objective mixed integer non-convex nonlinear programming method that provides maximum spectrum utilization and fairness simultaneously [155]. In the present work, MODE and NSGA-II algorithms are used to find the trade-off solutions between the network utility functions and forced termination probability.

In the cognitive network model, it is assumed that every cognitive user decides its spectrum by executing a distributed SA algorithm. Hence, each cognitive user uses its embedded computing platform to compute the assignment matrix with the information available from the neighborhood users. However solving SA using MODE algorithm on an embedded processor is a challenging task as it consumes most of the resources and degrades the execution performance of other primary applications. Hence, it demands a dedicated hardware peripheral to accelerate the execution speed of the MODE-SA task. Since MODE is a superior algorithm for solving multi-objective optimization problems, the present work proposes a hardware architecture for MODE algorithm. The proposed hardware is an extension of the architecture explained in Chapter 4. The major functionalities of MODE algorithm are mutation, crossover, selection, dominance filter and fitness evaluation (three network utility functions). The fitness evaluation module is a computationally complex module and needs to be implemented in hardware to speed up the execution of the application. In this work, the fitness functions and MODE modules are integrated into a single core. It avoids bus transaction overhead between the two different modules. The functionality of MODE IP core is validated by optimizing two benchmark test functions mentioned in Appendix A.3. Then finally MODE-SA hardware is developed and interfaced to PPC440 processor through APU controller to design a MODE-SA coprocessor.

5.1.2 Related work

In literature, Wen et al. proposed Genetic Algorithm (GA) based SA scheme using Max-Overall-Performance algorithm [156] where cognitive radio user fairness access and system's sum bandwidth are optimized by introducing a penalty function into the objective function. It resulted trade-off solutions by maximizing the overall system performance. However, the results are not compared with

any other popular evolutionary algorithms. Byun et al. used bi-objective mixed integer non-convex nonlinear programming method to solve centralized SA problem in wireless sensor network [155]. It aimed to achieve maximum fairness and maximum spectrum utilization simultaneously. The disadvantage of this approach is that the centralized approach requires more signaling between the server and the wireless nodes in the network. Devarajan et al. formulated the SA task as a multi-objective optimization problem for a multi-relay and multi-user cooperative cellular network [154] to optimize the power and throughput simultaneously. It considered the quality of service in terms of signal-to-noise ratio and fairness among users.

In literature, some of the multi-objective optimization algorithms have been implemented in hardware to accelerate the execution performance of the algorithms. Jonathan Kok et. al developed a hardware for NSGA-II and implemented on Xilinx Virtex 4 FPGA [157]. It was validated by solving four test bench functions and achieved approximately a speedup of 1300x compared to the processor implementation results. Multi-Objective Genetic Algorithm (MOGA) is also used to solve multi-objective optimization problems but requires large computation power. Tachibana et. al implemented MOGA in hardware to accelerate the execution speed with high search efficiency [158]. The hardware was validated by solving a multi-objective Knapsack problem. There is no previous work related to the hardware implementation of the MODE-SA. In this chapter, initially the SA problem is solved using different multi-objective evolutionary algorithms. Finally, FPGA based hardware accelerator is proposed to accelerate the execution performance of MODE-SA. The execution speed of the hardware is compared with equivalent PPC440 processor execution speed.

5.1.3 Multi-Objective problem formulation of Spectrum Allocation

In this chapter, three network utilization functions are optimized simultaneously by considering the two constraints: a) number of selected channels (Ch) and b) required channel capacity (R). The SA model is same as shown in Figure 2.3. The definition of utility functions i.e., MSR, MMR and MPF are also same as defined earlier in Equations.(2.3), (2.4), (2.5) of Chapter 2. Mathematically, the multi-objective spectrum allocation can be defined as:

$$\begin{aligned} & Maximize && [MSR, MMR, MPF, 1/FT] \\ subject\ to: & \sum_{m=1}^{M} a_{n,m} && = Ch \\ & \sum_{m=1}^{M} b_{n,m} && \geq R \end{aligned} \tag{5.1}$$

where FT is forced termination probability.

5.1.4 Forced termination probability

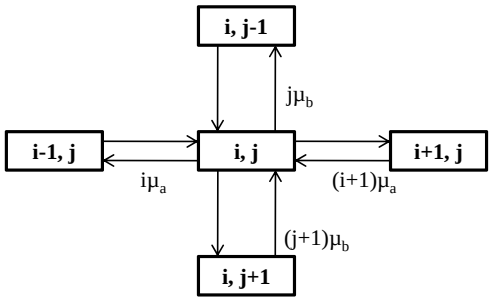

Figure 5.1: Transitions of cognitive radio from one state to another

In this work, SA is modeled as a continuous time markov chain, and it is characterized by its states and transition rates as shown in Figure 5.1 [14, 15]. Let primary users and cognitive users share M channels and state is denoted by an integer pair (i, j), where i is the total number of channels used by the secondary users and j is the total number of channels used by primary users. It is assumed that the arrival rates of secondary and primary users are λ_a and λ_b respectively, and modeled as a Poisson process. Let the corresponding service rates are μ_a and μ_b respectively. Primary users have the highest priority to occupy the band as they are the licensed users. Cognitive users are supposed to vacate the band if a primary user demands to use the particular band. Forced termination occurs when a PU occupies a channel used by SU. At this instant SU will move from the state (i, j) to $(i, j + 1)$ ... $(i - t, j + 1)$ depending on t, where t is the number of channels preempted by PU. The residual $(i - t)$ channels are distributed in remaining $(M - j - 1)$ channels. In this work, two assumptions are considered for SA.

1. Forced termination of a SU occurs only when the SU using the same channel which is preempted by a PU.

2. If the PU occupies the channel other than the same band then forced termination will not occur, only data rate will get reduced.

For the above given SA model, let us consider an example with N=5, M=5 and K=5, then the total number of states is [14]

$$1 + \sum_{x=1}^{M} (x+1) = 21$$

The channel assignment matrix corresponding to above example is A_{NXM}

$$A_{5X5} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

In the above matrix, rows represent SUs and columns represent channels that are assigned to utilize spectrum resources. Let S be the number of cognitive users for which a particular channel is available (for channel 1 S=1, for channel 3 S=4) and V is the number of channels that each cognitive user can occupy (for SU1 V=3, for SU2 V=2). Let n_c be the number of cognitive users in the present state and n_p be the number of PUs in the present state. $\gamma^{(n_c,n_p)}_{(n_c-1,n_p+1)}$ is defined as the transition rate from the state (n_c, n_p) to $(n_c - 1, n_p + 1)$ [14].

$$\gamma^{(n_c,n_p)}_{(n_c-1,n_p+1)} = \frac{\binom{S}{1}\binom{M-n_p}{V-1} * \lambda_b}{\binom{M-n_p}{V}} \tag{5.2}$$

A transition occurs when the primary user will retain its channel for communication. Forced termination of the secondary users occur when a channel occupied by a SU is retained by a PU. In this scenario, the SU has to vacate the channel. This termination may also occur if the reward value of the m^{th} channel i.e., $b_{n,m}$ is less than the threshold value. The forced termination probability (P_F) is given by:

$$P_F = \sum_{n_p=0}^{M} \sum_{n_c=0}^{N} \frac{1}{n_c} * \gamma_{(n_c-1,n_p+1)}^{(n_c,n_p)} P(n_c, n_p) \tag{5.3}$$

where $P(n_c, n_p)$ is the transition probability matrix [14].

5.1.5 Multi-Objective Differential Evolution algorithm

The multi-objective optimization problem for maximizing a set of functions can be generalized as follows:

$$\begin{aligned} Maximize \quad & [f_1(x), f_2(x), ..., f_k(x)] \\ Subject\ to \quad : \quad & g_l(x) \leq 0,\ l = 1, 2, ..., n \\ h_j(x) \quad = \quad & 0,\ j = 1, 2, ..., m \end{aligned} \tag{5.4}$$

where k is the number of fitness functions, l is the number of inequality constraints and j is the number of equality constraints. $x \in X^d$ is vector of decision variables, where d is the number of independent variables x_i. $f_i(x)$ are cost functions or objective functions. Trade-off solutions are obtained by solving the multi-objective optimization problem. A trade-off solution is said to be pareto optimal when there exists no solution that dominates it. Then a set of pareto optimal points resulting in solution space is known as pareto front. The algorithmic parameters of MODE algorithm are tabulated in Table 5.1. The pseudo-code of the multi-objective differential evolution is described in Algorithm 7.

Table 5.1: Control parameters of the MODE algorithm

Control Parameters	Value
Population Size ($XPOP$)	600
Maximum number of Iterations (Max_Gen)	200
Weight Factor (F)	0.5
Crossover rate (Cr)	0.2
Number of Objective functions ($Nobj$)	2,3
Maximum number of Function Evaluations (Fun_Eval)	30000

Algorithm 7 : Pseudo-code of Multi-Objective Differential Evolution algorithm

Step 1: Read the control parameters of the MODE algorithm given in Table 5.1: number of objectives $Nobj$, weight factor F, crossover rate Cr, maximum number of iterations Max_Gen, maximum functional evaluations Fun_Eval and the population size $XPOP$ from user.

Step2: Set the generation number G=0 & randomly initialize population of $XPOP$ individuals

for i=1 to $XPOP$ //do for each individual sequentially do
  for j=1 to $Nvar$ //do for each individual sequentially do
    $Parent(i,j) = X_{min}$ + (X_{max}-X_{min})*rand(); //each individual uniformly distributed in the range $[X_{min}, X_{max}]$ where $X_{min} = \{x_1^{min}, x_2^{min}, ..., x_{Nvar}^{min}\}$ and X_{max}=$\{x_1^{max}, x_2^{max}, ..., x_{Nvar}^{max}\}$
  end for
end for

Step 3:

while the maximum functional evaluations limit is not reached OR G< Max_Gen do
  for i=1 to $XPOP$ //do for each individual sequentially do
    Step 3.1: Mutation Step
    Generate a mutant vector $Mutant(i)^{(G)}$=$\{mutant_{1,i}^{G}, ..., mutant_{Nvar,i}^{G}\}$ corresponding to the ith target vector $Parent(i)^{(G)}$ via the differential mutation scheme of DE as: $Mutant(i)^{(G)} = Parent(r_1)^{(G)} + F * (Parent(r_2)^{(G)} - Parent_(r_3)^{(G)})$, Vector indices r_1, r_2 and r_3 are randomly chosen, where r_1, r_2 and r_3 {1,...,$XPOP$}
    Step 3.2: Crossover Step
    Generate a trial vector $Child(i)^{(G)}$=$\{child_{1,i}^{(G)}, ..., child_{Nvar,i}^{(G)}\}$ for the ith target vector $Parent(i)^{(G)}$ through binomial crossover in the following way:
    if ($rand_{i,j}[0,1] \leq CR$, then
    $Child(i)^{(G)} = Parent(i)^{(G)}$
    else
    $Child(i)^{(G)}$=$Mutant(i)^{(G)}$
    endif
    Step 3.3: Selection Step
    Evaluate the trial vector $Child(i)^{(G)}$
    if $f(Child(i)^{(G)}) \leq f(Parent(i)^{(G)})$, then
    $Parent(i)^{(G+1)}$= $Child(i)^{(G)}$
    else
    $Parent(i)^{(G+1)}$= $Parent(i)^{(G)}$
    endif
  end for
  Step 3.4: If stopping criterion is not satisfied then increase the Generation Count: G = G + 1
end while

Step 4: Dominance filter is used to get pareto optimal solution of the problem.

Report results

Terminate

5.1.6 MODE based Spectrum Allocation

In the proposed MODE-SA, each individual population represents a conflict free channel assignment to SUs. The dimension of the population is evaluated using the availability matrix L that satisfies the condition $l_{n,m} = 1$. The values of L, B and C matrices are initialized using the Algorithm 1 given in chapter 3. The proposed MODE-SA algorithm is described as follows:

1. Initialize the MODE and SA algorithmic parameters as shown in Table 5.1

2. Given $L = \{l_{n,m} | l_{n,m} \in \{0,1\}\}_{N \times M}$, $B = \{b_{n,m}\}_{N \times M}$ and $C = \{c_{n,p,m} | c_{n,p,m} \in \{0,1\}\}_{N \times N \times M}$, dimension of the population is set to $D = \sum_{n=1}^{N} \sum_{m=1}^{M} l_{n,m}$ and G_{MAX} is set to 0;

3. Generate the parent population ($Parent(g,h)$) randomly, $X_g = [x_{1,g}, ..., x_{D,g}]$ where $x_{d,g} \in 0, 1$, and $g \in (1 \ldots XPOP)$

4. Generate the channel availability matrix (A) by using L, B and C matrices obtained using Algorithm 1.

5. Calculate the fitness of each individual of the current population using Equations.(2.3), (2.4) and (2.5).

6. Perform the mutation operation to generate mutant vectors and the bounds of the population values are verified.

7. Perform crossover and selection operations and update the parent population as defined in the MODE algorithm (Algorithm 7).

8. Increment the G_{MAX} value, if it reaches the predefined maximum iterations or maximum number of function evaluations then the parent population obtained from the step (7) is given to the dominance filter and stop the process else go to step (6) and continue.

9. In the dominance filter, non-dominated sorting operation is performed to obtain non-dominated solutions as a pareto front.

5.1.7 Simulation setup and Results

The simulation is carried out by assuming the CR environment as static and noiseless. MATLAB software is used to perform the extensive simulation on Intel Dual Core processor with 2GB RAM. In this work, four objective functions namely MSR, MMR, MPF and Forced termination probability are optimized simultaneously using MODE and NSGA-II algorithms. It is assumed that in the network there are N cognitive users, M number of channels and K primary users. Each primary user has a communication range denoted as d_p. The secondary users can use the vacant channel without any interference to neighbor users within the transmission range, i.e., between d_{min} and d_{max}. The simulation parameters of SA algorithm are N=20, K=20, M=20, d_P=1, d_{min}=1, d_{max}=4, Ch=4 and R=512 Kbps. These values are used to derive the availability matrix, reward matrix, constraint matrix and channel assignment matrix [8]. The threshold value of the reward is set to 16 Kbps. The algorithmic control parameters of MODE are tabulated in Table 5.1. In the case of NSGA-II, the population size, and maximum generations are chosen as 600 and 200 respectively. In the current simulation, the service rate of secondary and primary users are set as 0.82 and 0.06 respectively for calculating the termination probability. The arrival rate of SUs is fixed to 0.3, whereas it varies in between 0.02 to 0.1 for PUs.

The performance of algorithm is measured in terms of time complexity and pareto front. Both NSGA-II and MODE algorithms are executed for 20 independent runs. The time complexity of both the algorithms is evaluated as [128]. The following test code (Algorithm 8) is used for obtaining the time complexity.

Algorithm 8 : Test Code

```
for i=1 to 1000
x= double(5.55);
x=x+x; x=x./2; x=x*x; x=sqrt(x); x=log(x);
x=exp(x); y=x/x;
end for
```

Table 5.2 shows the percentage of improvement in time complexity of MODE over NSGA-II algorithm for solving the SA problem. The pareto front between the two objective functions i.e., MSR vs. Forced termination probability, MMR vs. Forced termination probability and MPF vs. Forced termination probability are shown in Figures 5.2 - 5.4 respectively.

Table 5.2: Time complexity of NSGA-II and MODE

Fitness Function	NSGA-II	MODE	Percentage of Improvement
MMR & FT	4.66e+5	1.38e+5	237
MSR & FT	4.87e+5	1.47e+5	232
MPF & FT	4.44e+5	1.30e+5	241
MSR, MMR & MPF	4.41e+5	1.86e+5	137
MSR, MMR, MPF & FT	5.63e+5	1.96e+5	187

FT = Forced termination probability

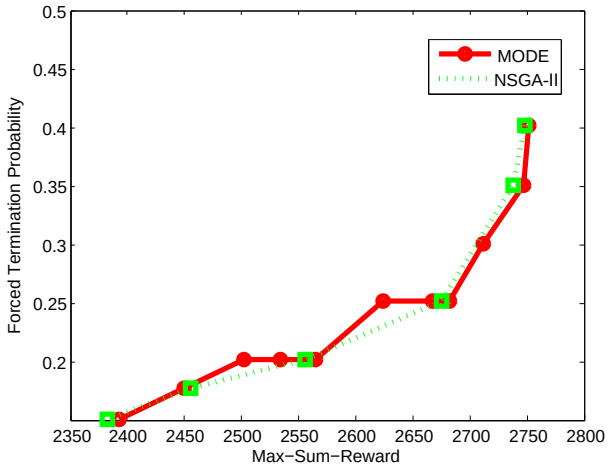

Figure 5.2: MSR vs Forced termination probability

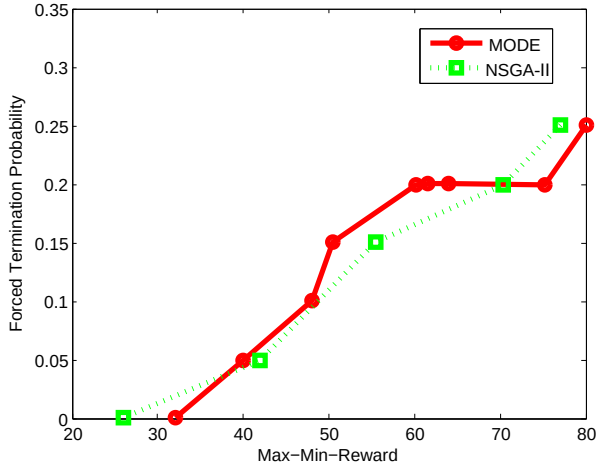

Figure 5.3: MMR vs Forced termination probability

Figure 5.2 depicts the typical nature of maximizing MSR while minimizing forced termination probability. It is observed that, as the MSR value increases,

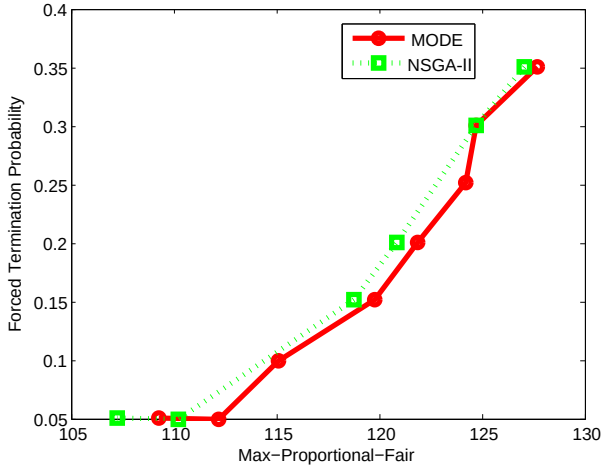

Figure 5.4: MPF vs Forced termination probability

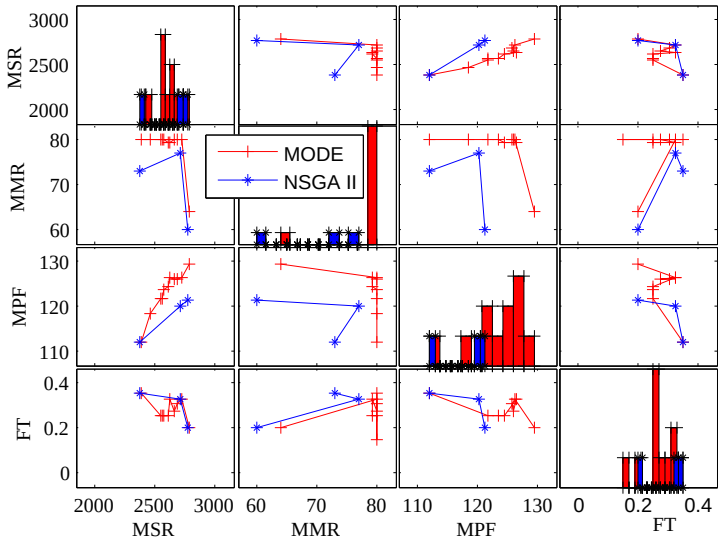

Figure 5.5: Pareo front of three and four objective functions

forced termination probability also increases. Figure 5.3 and Figure 5.4 shows the optimal front between MMR, MPF vs. forced termination probability respectively. From the above figures, it is observed that the forced termination probability increases with the increase in the values of MSR, MMR and MPF. Furthermore, the effect of including the termination probability along with three utility functions in the optimization is also studied. For this study, four objective functions are optimized simultaneously using both MODE and NSGA-II algorithms, and results are plotted as pareto fronts in Figure 5.5. To avoid the difficulty in observing the results, scatter plot matrix is used to analyze the cor-

relation between the two objective functions. Diagonal plot (histogram) shows the plot between the same objective function (i.e., MSR vs MSR). The legend MODE and NSGA-II correspond to the front using MODE and NSGA-II algorithms. From this figure, it is observed that MODE gives better front compared to NSGA-II algorithm.

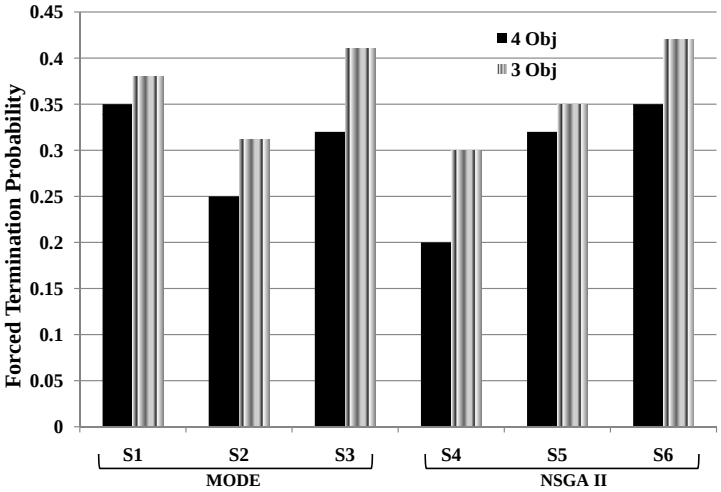

Figure 5.6: Performance of Quality of Service

In order to observe the influence of termination probability in the solution, first the three utility functions are optimized simultaneously and the forced termination probability is calculated at three different instances (S1, S2, and S3) from the front shown in Figure 5.5. Secondly, three instances that have approximately same front value as previous are selected from the MODE (with 4 objective functions) result. The calculated and optimized termination probability are plotted in Figure 5.6. From this figure, it is observed that inclusion of termination probability in allocation problem gives an efficient channel assignment solution with minimum termination probability. Thus, it is concluded that the quality of the solution is improved. The same conclusion can be drawn for NSGA-II solutions also. In Figure 5.6, the solutions S4, S5, and S6 corresponds to three different instances obtained using NSGA-II algorithm.

5.2 Joint Spectrum and Power Allocation in Cognitive Radio Networks

5.2.1 Introduction

Power allocation also plays an important role in the interference management and energy saving in mobile units. During power allocation, it is required to maintain a specific Signal-to-Interference-plus-Noise Ratio (SINR) level to the cognitive users such that it minimizes interference level to the PUs. In literature, most of the reported work solved the SA problem by maximizing the network throughput. The optimization of network throughput with power constraint guarantees minimum interference between CR to CR users and CR to PUs in the network while achieving maximum network capacity. Hence, it is required to optimize the individual transmitted power of each user during SA. Digham et. al proposed a joint power and channel allocation optimization algorithm for maximizing sum capacity of the CR network, while satisfying SINR constraints posed by PUs [159]. However, it does not consider the fairness among the CR users. Wang et. al proposed a novel SA and power control algorithm to maximize the network utility functions in CR networks [160]. In [160], transmitting power of CR node is dynamically varied according to the cost and connection degree by satisfying the quality of service requirements and interference constraints posed by SUs.

Haddad et al. proposed a downlink and uplink distributed power allocation algorithm to maximize the CR network capacity [161, 78]. In [161], a SU can decide either to communicate or silent in the channel without affecting the QoS of PUs. Zayen et. al proposed a binary power allocation scheme that consists of a single PU and multiple SUs in the network [79]. However in real-time, the SU transmission cannot be on a single channel to communicate with other users. If a SU uses multiple channels for communication, then it may affect on multiple PUs transmission due to interference. In recent studies, CR protocols are extended such that both primary and secondary users simultaneously transmit over the same channel [162, 163]. The present work considers that the SUs communicate over the same channel along with PUs at the same time with an acceptable interference from SUs.

In literature, many works have considered the centralized optimization scheme

for power control that requires the information about interference and channel conditions of the network [164, 79, 159]. To avoid this problem, distributed scheme is preferred in which each user can manage its resources based on locally available information like channel gain, interference and noise conditions. In distributed scheme, the system capacity is maximized without any interference to PUs by considering SU interference and information outage probability. The outage probability is used to define probability of mutual information channel data rate is below the transmitted code rate [165]. The mutual information of the channel must be able to provide target data rate R to achieve better communication. Hence, the SU can adapt its power level $p_{i,j}$ in between 0 and p_{max} to achieve the desired data rate by not affecting the outage probability of PUs. For SA, channel capacity is the most important factor to assign channels for the requested users. Hence, it is necessary to optimize the network capacity and transmitted power of each user (i.e., the individual average capacity of SU) simultaneously to achieve minimum interference to PUs and required QoS.

The main goal of the present work is to assign suitable channels to SUs with optimum transmitting powers for maximizing the network throughput under various constraints like noise, power, and interference. In the present work, the channel gain and network throughput are calculated using different channel characteristic parameters such as path loss, attenuation, fading of respective channels and their interference to neighbor users. A joint spectrum and power allocation function is formulated to maximize the capacity of each user. It is solved using DE and PSO algorithms to provide a fair allocation of channel and transmission power to SUs in both uplink and downlink scenarios. It is also important to study the effect of the joint spectrum and power allocation on the total network utilization. In literature, the total network utilization is analyzed using Max-Sum-Reward (MSR) utility function in a distributed network architecture. It is used to maximize the utilization of spectrum resources by assigning proper channels to SUs by satisfying the interference constraints posed by both SUs and PUs. Hence, this study is extended to optimize the total network utilization (MSR) and capacity of each user simultaneously using MODE and NSGA-II algorithms. Trade-off solutions between the two objective functions are obtained through a pareto front. The performance of two algorithms are compared in terms of quality of the solution. Finally, forced termination probability is included as one more objective function and optimized the three functions simultaneously to find the quality of service

metric of the proposed algorithm.

5.2.2 System model

It is considered that a wireless cognitive radio network consists of K number of primary users and N number of secondary users randomly deployed over a geographical area as shown in Figure 5.7. It is assumed that M number of channels are available for communication. In this model, a certain number of primary users, secondary users and channels are scheduled to communicate at a given instant of time while remaining users are silent. It is also assumed that (i) all SUs have a prior knowledge about the channel state information of their links and (ii) channel gains are independent and identically distributed random variables. Each PU is covered by a non-interference boundary and SUs do not communicate within this range in an ad-hoc manner.

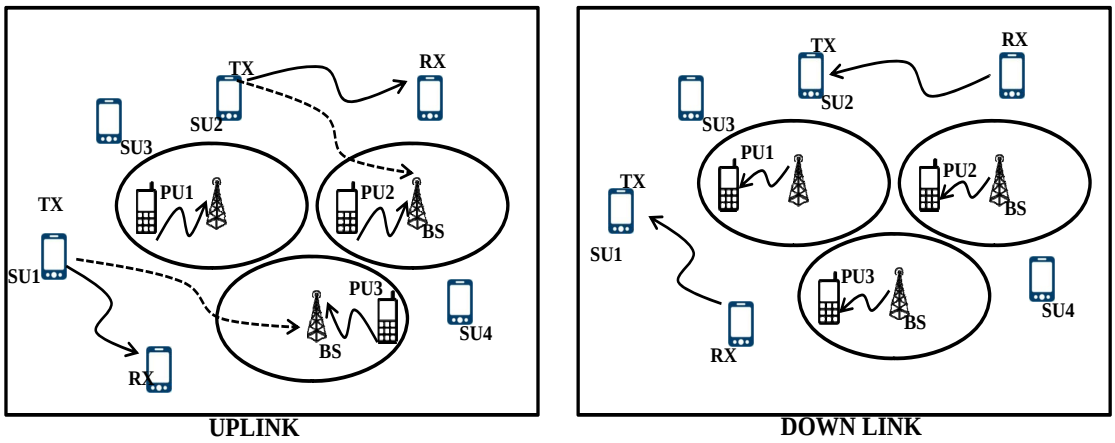

Figure 5.7: Network Model

To avoid the interference to PUs, SUs communicate outside the protection range of PUs. The protection area surrounded by each primary user is denoted by d_p and the minimum and maximum communication range of each SU is in the range of d_{min} and d_{max} respectively. Each SU calculates its availability matrix L based on the availability of channels near to it. The channel availability matrix is $L = \{l_{n,m} | l_{n,m} \in \{0,1\}\}_{N \times M}$. If m^{th} channel is available to n^{th} user then $l_{n,m} = 1$ otherwise $l_{n,m} = 0$. The channel reward matrix $B = \{b_{n,m}\}_{N \times M}$ is initialized with the capacity assigned to n^{th} user in m^{th} channel using power allocation algorithm [79]. The interference constraint matrix $C = \{c_{n,p,m} | c_{n,p,m} \in \{0,1\}\}_{N \times N \times M}$ is

determined by evaluating the channel constraints and user constraints [8]. In this distributed approach user's position and available spectrum are static during the allocation process. The final assignment matrix $A = \{a_{n,m}|a_{n,m}\ in\{0,1\}\}_{N\times M}$ is calculated using L, B and C matrices. The instantaneous capacity of i^{th} primary user is given by [79]:

$$Cap_i = log_2\left(1 + \frac{p_i\,|g_{i,i}|^2}{\sum_{j=1}^{N} p_{i,j}\,|g_{i,j}|^2 + \sigma^2}\right) \tag{5.5}$$

where p_i is the transmitted power of i^{th} primary user, $g_{i,i}$ is the channel gain between base station and primary user, $p_{i,j}$ denotes the transmitted power of the j^{th} secondary user in i^{th} channel, $g_{i,j}$ denotes the channel gain of j^{th} SU from i^{th} PU, σ^2 is the ambient noise variance. The instantaneous capacity of j^{th} secondary user over a channel i is given by [79]:

$$Cap_{i,j} = log_2\left(1 + \frac{p_{i,j}|g_{i,j}|^2}{\sum_{\substack{q=1\\q\neq j}}^{N} p_{q,j}|g_{q,j}|^2 + p_i|g_{i,j}|^2 + \sigma^2}\right) \tag{5.6}$$

In a distributed network, SUs need to adapt the required communication environment without any interference to PUs to maximize the network capacity. The average cognitive capacity of each SU over the network is given by:

$$Cap_{sum} = \frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{N} A_{i,j} * Cap_{i,j} \tag{5.7}$$

where $A_{i,j}$ is the channel assignment matrix. In this work, the cognitive radio network capacity (5.7) is maximized by controlling the transmitting power of SUs to avoid the PU interference. The information about channel characteristics and required data rate of the PU are to be known to the SU apriori, to protect the PU from interference.

5.2.3 Proposed algorithm

In this work, it is assumed that both primary and secondary users are utilizing the same channel by limiting the transmitted power levels of SUs. In communication, the SUs detect the hole, then adapts to PU transmission characteristics without any interference to PUs while satisfying the QoS requirements. In this technique, SUs opportunistically selects the channel with an acceptable transmission power to maximize the total sum rate while considering the proper outage probability of PUs. A distributed approach is considered for joint spectrum and power allocation, in which SU decides the channel and power allocated to each channel, such that the allocated power is less than the SNR threshold. Initially, the joint spectrum and power allocation problem is formulated as a single objective optimization problem and the average capacity of each user is maximized by satisfying the SINR and outage probability constraints. During the evaluation of the objective function, an iterative method is used to check the SINR constraints and PU outage probability. Based on these constraints, the solution is updated and finally a fair spectrum and power assignment solution is achieved. The joint spectrum and power allocation optimization problem is defined as:

$$\text{Find}\,\{A_{i,j}, p_{i,j}\}\; s.t \arg\max_{A_{i,j},p_{i,j}} Cap_{sum}$$

subject to:

$$\begin{aligned} &A_{i,j}\epsilon\{0,1\}\; p_{i,j}\epsilon[0,p_{th}], for\; i=1,..,M\; j=1,...,N \\ &P_{out} = Prob\{Cap_i \leq R_i \mid R_i, w\} \leq w \end{aligned} \tag{5.8}$$

where R_i denotes i^{th} PU transmitted data rate and w is the outage probability. The key idea in this technique is that each SU selects a suitable channel and limit its power level in the range of $[0, p_{th}]$ by controlling the interference to other users in the network. In this distributed algorithm, selection of proper channels and transmission power to the respective SUs should guarantee the QoS of PUs. The SINR constraints are considered to manage the SU transmission power for avoiding the interference to other users. It is also noted that, if a SU violates the PU outage constraint then the SU should vacate the channel. In the high SINR condition, all SUs are transmitting above the threshold power limit p_{th}. If all SUs are transmitting with P_{max} then n^{th} SU will be active only if [79]

$$\frac{|g_{n,n}|^2}{\sum\limits_{\substack{q\in\psi\cup\{i\}\\ q\neq n}} |g_{q,n}|^2} > \left(\frac{\tilde{N}+U-1}{\tilde{N}+U-2}\right)^{\tilde{N}-1} \tag{5.9}$$

where $\tilde{N}$ is the size of active SUs, $U = p_i/p_{max}$ and ψ represents set of indices of all active SUs with respect to the number of channels. In case of large number of secondary users distributed over a limited geographical area (i.e., dense network) n^{th} SU will be active in i^{th} channel if its signal-to-interference ratio (SIR) is greater than e [79].

$$SIR_{i,n} = \frac{p_{i,n}|g_{i,n}|^2}{p_i|g_{i,n}|^2 + \sum\limits_{\substack{q\in\psi\\ q\neq n}} p_q|g_{q,n}|^2} > e \tag{5.10}$$

In the low SINR condition, power control algorithm adapts to a suitable power level such that it avoids the interference to PUs and neighborhood SUs. The n^{th} SU will be active if it satisfies the below condition: [79]

$$SINR_{i,n} < \frac{\sum\limits_{\substack{j\in\psi\\ j\neq n}} p_{i,j}|g_{j,j}|^2}{P_{max}W^2(\tilde{N}+U-2)+\sigma^2} \tag{5.11}$$

$$\simeq \frac{P_{max}W^2(\tilde{N}-1)}{P_{max}W^2(\tilde{N}+U-2)+\sigma^2} \tag{5.12}$$

where, W^2 denotes average interference gain. In a dense network with low SINR, a SU will be active if the SIR of n^{th} user is greater than 1.

$$SIR_{i,n} = \frac{p_{i,n}|g_{n,n}|^2}{p_i|g_{i,n}|^2 + \sum\limits_{\substack{q\in\psi\\ q\neq n}} p_{i,q}|g_{q,n}|^2} > 1 \tag{5.13}$$

In this work, QoS of primary users is maintained with the use of outage constraint. In a centralized approach, there is a need of central database entity but it is hard to implement and extract the diversity gain in fast fading environment that involves overhead in signaling between the users. In distributed approach, SUs obtain information about a PU either through a band manager or by collecting the channel state information itself. The PU average channel gain $g_{i,i}$ can be

approximated to average channel gain G_i as

$$g_{i,i} = G_i * g'_{i,i} \tag{5.14}$$

where $g'_{i,i}$ is the random variable of channel gain i.e., normalized channel impulse response. The PU outage probability can be defined as:

$$P_{out} \simeq 1 - exp\left[-(2^{R_i} - 1)\left(\frac{\tilde{N}W_{su}^2 P_{max} + \sigma^2}{W_i^2 p_i}\right)\right] \leq q \tag{5.15}$$

Hence, the maximum number of active SUs ($\tilde{N}$) transmitting with required specifications and with out interference to PU transmissions is

$$0 \leq \tilde{N} \leq \frac{-log(1-q)}{(2^{R_i} - 1)} . \frac{W_i^2 p_i}{W_{su}^2 P_{max}} - \frac{1}{SNR} = \tilde{N}_{theory} \tag{5.16}$$

where $\tilde{N}_{theory}$ is the theoretical value of maximum number of active SUs. To maximize the average capacity of a user, firstly the proper channels are selected for required SUs and then the power allocation is performed in the selected channels. Optimal resource allocation to SUs is performed by maximizing each user sum rate over the assigned channels using PSO and DE algorithms. An iterative method is used to check SINR constraints ((5.10) and (5.13)) and PU outage constraint (5.15). After checking the constraints, the proposed algorithm allocates power to the SUs. It avoids interference to PUs and maximizes the average capacity of each user.

5.2.4 Joint Spectrum and Power allocation using DE and PSO algorithms

In this work, two evolutionary algorithms namely DE and PSO are used for solving the joint spectrum and power allocation problem. The complete detail of DE algorithm for channel allocation without considering power constraint is explained in Chapter 3. The pseudo-code of the proposed DE based joint spectrum and power allocation is given in Algorithm 9. In the present CRN model, if multiple channels are allocated to a SU, then it does not terminate forcefully due to either interference or PU arrival. However, the SU transmission data rate may decrease because it vacates few channels from the assigned channels. The pseudo-code of

the joint spectrum and power allocation technique is shown in Algorithm 10. For each population, Algorithm 10 is executed and finally the fitness value is evaluated using (5.7). Each SU measures the channel SIR and compares with low or high SIR value. In case of both conditions are not satisfied then $p_{i,j}$ values are assigned with the previous values. In each iteration, $p_{i,j}$ values are updated using DE/PSO algorithms. Each PU verifies its outage constraint in every iteration based on the results of channel and power allocation. This loop iterates until a maximum number of iterations is reached. The same problem is solved using PSO algorithm and compared the performance with DE algorithm.

Algorithm 9 : DE based Joint Spectrum and Power allocation in distributed approach

Step 1: Initialize the cell radius, protection area and all parameters related to the proposed algorithm.
Step 2: Define number of secondary users (N), primary users (K) and number of channels (M), initialize the optimization algorithm (DE and PSO) control parameters.
Step 3: Evaluate the channel availability matrix $L = \{l_{n,m} | l_{n,m} \in \{0,1\}\}_{N \times M}$ and calculates the dimension of the population $D = \sum_{n=1}^{N} \sum_{m=1}^{M} l_{n,m}$.
Step 4: Initialize the population $XPOP$ with the solution variables of channel and power allocation algorithm
i.e., $XPOP = A_{i,j} \cup p_{i,j}$ $A_{i,j} \epsilon \{0,1\}$ $p_{i,j} \epsilon [0, p_{th}]$, where i= 1... M and j=1...N
Step 5: Evaluate the fitness function defined in Equation (5.8) for each population using Algorithm 10.

```
while maximum no. of generations is not reached do
    Perform the Mutation, Crossover and Selection operations to find the best fitness
    value.
    Evaluate the average capacity of SU using Equation (5.8)
    Increase the Generation Count
    G = G + 1
end while
Report results
Terminate
```

Step 6: Resultant solution i.e., spectrum and power assignment values $A_{i,j}$ and $p_{i,j}$ allocated to current SUs.

5.2.5 Simulation Results

For simulation, the joint spectrum and power allocation problem defined in (5.8) is optimized using DE and PSO algorithms. MATLAB software is used to perform the extensive simulation. For simulation K number of channels (K=10),

Algorithm 10 : Joint spectrum and power allocation

for t=1:t_{max} do
  Calculate Channel Assignment matrix $A = \{a_{n,m} | a_{n,m} \in \{0,1\}\}_{N \times M}$ by satisfying the interference constraints $a_{n,m}.a_{p,m} = 0, if c_{n,p,m} = 1, \forall 1 \leq n,p \leq N, 1 \leq m \leq M$
  for i=1 to M do
    for j=1 to N do
      if $SINR_{i,j}^{(t)} > e$, then // In high SINR condition
      $p_{i,j}^{t+1} = p_{th}$
      else
      $p_{i,j}^{t+1} = p_{i,j}^{t}$
      endif
      if $SINR_{i,j}^{(t)} > 1$, then // In low SINR condition
      $p_{i,j}^{t+1} = p_{th}$
      else
      $p_{i,j}^{t+1} = p_{i,j}^{t}$
      endif
    end for
  end for
  Check outage constraint given in (5.15)
  for i=1 to M do
    if $\tilde{N}_i^{(t)} \leq \tilde{N}_{theory}^{(t)}$, then $\tilde{N}_i^{(t+1)} = \tilde{N}_i^{(t+1)} - 1$
    endif
  end for
  Evaluate the objective function defined in (5.8) to find the average capacity of SU
  Increase the iteration count
  $t = t + 1$
end for

M primary users (M=10) and N number of cognitive users ($N = 1\ to\ 10$) are deployed uniformly within an area of 1000m radius with PU protection area of radius 600m for the model shown in Figure 5.7. The channel gains are considered as defined in COST-231 model [166] with log-normal shadowing (standard deviation of 10dB) and fast fading effects in the range of (0,1). In simulation, the maximum power (P_{max}), power ratio for downlink and uplink are considered as 1W, 10 and 1 respectively. The CR network simulation parameters L, B and C matrices are initialized as described in Chapter 2.

In the DE based joint spectrum and power allocation, the control parameters are initialized as crossover rate Cr=0.9, weighting factor F=0.9, population size NP=600 and maximum generations G_{max}=200. In case of PSO, the control parameters are initialized as maximum iterations=200, number of particles NP=600, inertial weight ω=0.3 and weighting coefficients $c_1 = c_2 = 0.9$. The

dimension of population is calculated by considering the number of elements in availability matrix L and the transmitted power $p_{i,j}$ depends on number of SUs in the network. The lower and upper bound of channel assignment variables are set as 0 and 1, whereas power assignment variables ($p_{i,j}$) are set as 0 and P_{max} respectively. This subsection presents numerical results for network capacity and the average capacity of each user by assigning suitable spectrum and transmitted power to each user. For the same network settings, both DE and PSO algorithms are used to solve the spectrum and power allocation problem and the results are given in Figures 5.8 - 5.11.

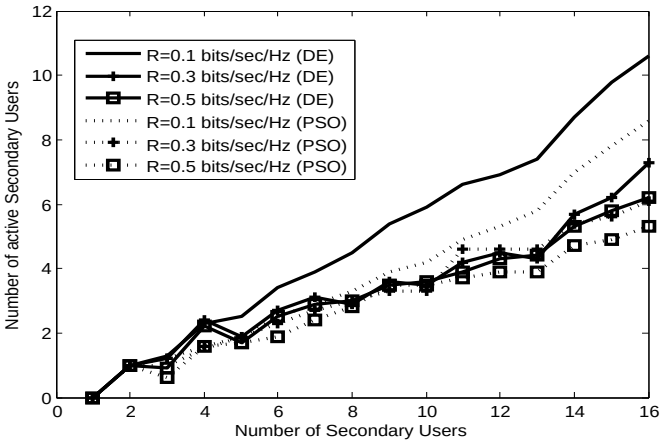

Figure 5.8: Number of active secondary users vs. number of secondary users with different rates in downlink

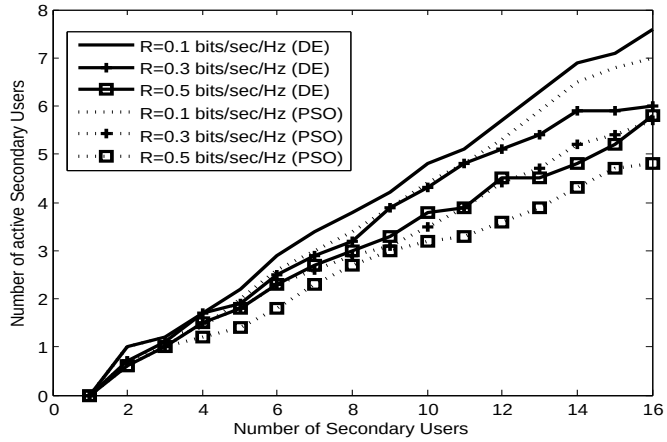

Figure 5.9: Number of active secondary users vs. number of secondary users with different rates in uplink

From Figures 5.8 and 5.9, it is observed that as the number of incoming SUs increases, the number of active SUs also increases while satisfying the QoS requirements of PUs. From this results, it is noted that the number of active SUs in downlink outruns the uplink scenario for power allocation. It is also observed that in downlink scenario base station transmitted power is ten times more than uplink power, that helps to maintain guaranteed QoS to PUs. In the case of downlink scenario, for a rate R= 0.1bits/s/Hz (DE), 68% of SUs are allowed, whereas in uplink 43% of SUs are allowed for 16 incoming SUs using DE algorithm. In case of PSO, 53% and 43% of SUs are allowed in downlink and uplink scenarios respectively. As the number of incoming SUs increases, initially the number of active SUs also increases but at a certain value the active SUs remains constant due to the interference constraints posed by SUs and PUs in multiple channels. It is also observed that the number of active SUs is decreasing with increase in R. It means that the data rate required by SU increases, but the number of active SUs decreases due to limited availability of resources.

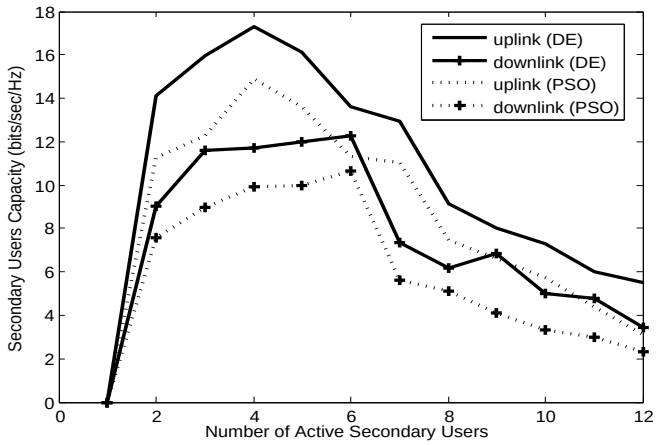

Figure 5.10: Sum of Secondary users capacity per user vs. number of secondary users with different rates for R=0.5 bits/s/Hz

Fig.5.10 and 5.11 show the affect on sum of user capacity over the available channels for both uplink and downlink configurations for R = 0.5 and 0.3 bits/s/Hz. It is observed that the capacity in uplink scenario outruns that of downlink system. It is also to be noted that as the number of SUs increases, capacity of a user also increases up to a certain number of users, then there is a decrease in SINR of each user due to interference that in turn decrease the user capacity. If the protection area of a PU and its primary cell radius decreases,

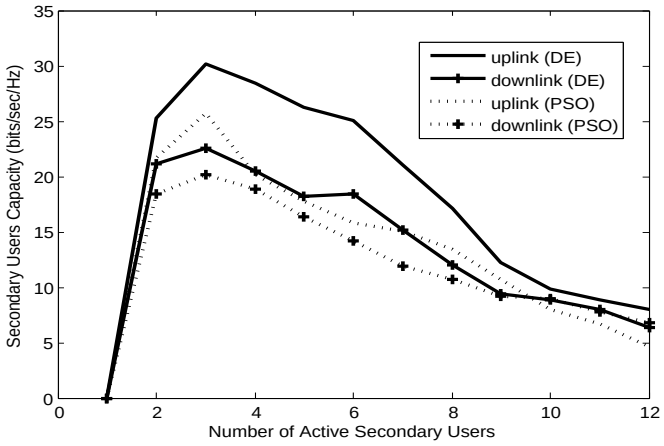

Figure 5.11: Sum of Secondary users capacity per user vs. number of secondary users with different rates for R=0.3 bits/s/Hz

deployment of a large number of SUs transmissions affect the interference caused to PUs and user capacity. Hence, there must be a trade-off between the number of active SUs and average user capacity. From the Figures 5.8 - 5.11, it is observed that DE performs better compared to PSO in terms of quality of solution due to its robustness and quick convergence in finding a solution.

5.2.6 MODE based Joint spectrum and power allocation

The network utility function Max-sum-Reward (MSR) is considered to obtain the relationship between the average capacity of a user and total network utilization. This function maximizes the total network capacity by assigning more number of channels with optimal power. It is given by [8]

$$MSR = \sum_{i=1}^{M} \sum_{j=1}^{N} B_{i,j} * A_{i,j} \tag{5.17}$$

where $A_{i,j}$ is the channel assignment matrix and $B_{i,j}$ is the reward matrix of SUs. In this subsection, the dependency between total network utilization and average sum of user capacity is observed for a fixed number of SUs and PUs. Hence, it is considered that the total network utilization (MSR) as one objective function and average capacity of SU (Cap_{sum}) as another objective function. The proposed joint spectrum and power allocation problem is formulated as a multi-objective optimization problem and the two objective functions (MSR & Cap_{sum}) are opti-

mized simultaneously using MODE and NSGA-II algorithms. The multi-objective optimization of joint spectrum and power allocation problem is formulated as:

$$
\begin{aligned}
&Maximize\ [MSR\ ,\ Cap_{sum}] \\
&\text{subject to:} \\
&\qquad A_{i,j}\epsilon\{0,1\}\ p_{i,j}\epsilon[0,p_{th}], for\ i=1,..,M\ j=1,...,N \\
&\qquad P_{out} = Prob\{Cap_i \leq R_i \mid R_i, w\} \leq w
\end{aligned}
$$

In the proposed MODE-based joint spectrum and power allocation algorithm, the solution is represented as a population that defines channel and power assignment to CR users. The dimension of population is calculated by considering the number of elements in availability matrix L and the transmitted power values $p_{i,j}$ depends on the number of SUs. The proposed MODE-based joint spectrum and power allocation algorithm is described as follows:

1. Initialize the MODE and joint spectrum and power allocation algorithm parameters.

2. Given $L = \{l_{n,m} | l_{n,m} \in \{0,1\}\}_{N\times M}$, $B = \{b_{n,m}\}_{N\times M}$ and $C = \{c_{n,p,m} | c_{n,p,m} \in \{0,1\}\}_{N\times N\times M}$, dimension of the population is set to $D = 2*\sum_{n=1}^{N}\sum_{m=1}^{M} l_{n,m}$ and G is set to 0;

3. Initialize the parent population ($Parent(g,h)$) randomly, $X_g = [x_{1,g},..,x_{D,g}]$ where $x_{d,g} \in 0,1$, and $g \in (1 \ldots XPOP)$

4. Evaluate the channel assignment matrix using L, B and C matrices that satisfies the PU and SU's constraints as described in Algorithm 1.

5. Find $p_{i,j}$ using power allocation algorithm as defined in Algorithm 10.

6. Calculate the fitness of each member of the current population using Equations (5.8) and (5.17).

7. Perform mutation operation to generate mutant vectors, and verify the bounds of the population.

8. Perform crossover and selection operations and update the parent population as defined in the MODE algorithm.

9. Increment the G, if it reaches the maximum predefined iterations or maximum number of function evaluations then the parent population obtained from the step (8) is given to the dominance filter and stop the process else go to step (7) and continue.

10. If stopping criterion is satisfied then the final parent population is given to the dominance filter to get a non-dominated solution, i.e., pareto front.

These two objective functions are optimized simultaneously using MODE and NSGA-II algorithms and pareto front is shown in Figure 5.12. From this figure, it is observed that as the total network capacity (MSR) increases, average capacity of SU is decreasing. It means that increase in total network capacity is due to the allocation of more number of channels to the user, but the Cap_{sum} decreases due to more number of constraints need to be satisfied by the user in the selected channels. It is also observed that MODE performs better compared to NSGA-II algorithm for maximizing total network capacity and average user capacity.

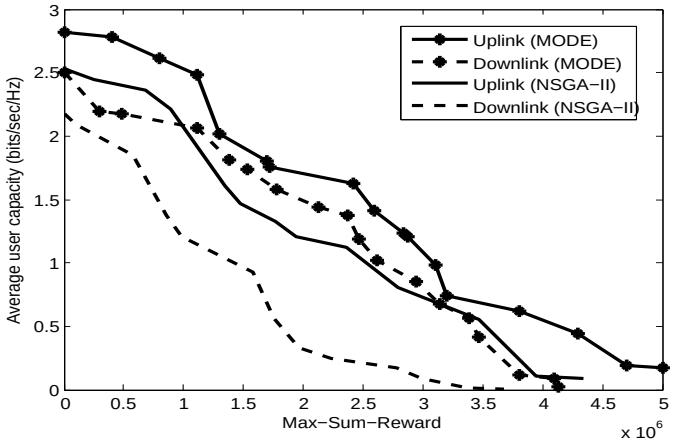

Figure 5.12: Pareto Front between MSR and Average user capacity

Further, forced termination probability is included as one more objective function along with the above two objective functions and optimized simultaneously to find the channel allocation metric. From the resultant pareto front, four instances of the solution are selected that are approximately close to solutions obtained from Figure 5.12. The calculated and optimized forced termination probability are plotted in Figure 5.13. From this figure, it is observed that inclusion of termination probability in joint spectrum and power allocation algorithm gives approximately same termination probability as compared to the termination probability obtained

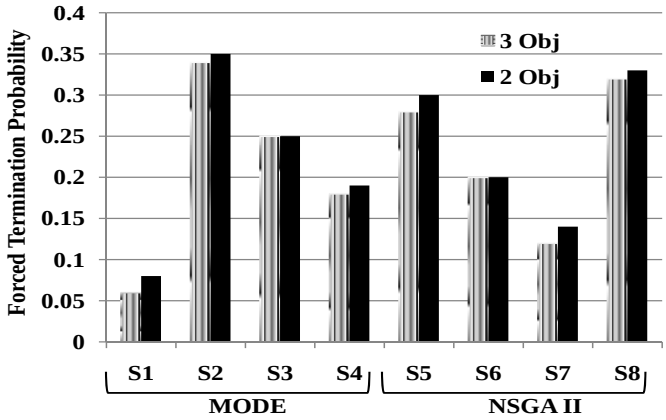

Figure 5.13: Performance of Quality of Service

by optimizing the two objective functions (MSR and Cap_{sum}) simultaneously (in case of S1, S2, S3 and S4 solutions of MODE). This is because the outage probability constraint is included in the proposed joint spectrum and power allocation algorithm. The same conclusion can be drawn for NSGA-II solutions also. In Figure 5.13, the solutions S5, S6, S7 and S8 corresponds four different instances obtained from NSGA-II algorithm.

5.3 FPGA implementation of MODE based Spectrum Allocation technique for Cognitive Radio Networks

In the previous chapter, DE algorithm is implemented in FPGA to solve the single objective SA problem. In this chapter, MODE algorithm is used to solve the SA problem. Hence, this section presents the implementation details of MODE algorithm for solving SA problem on Xilinx Virtex 5 FPGA.

5.3.1 Hardware implementation of MODE algorithm

The hardware architecture of the proposed MODE algorithm is given in Figure 5.14. This architecture consists of nine modules, namely Initialization of population, Mutation, Crossover, Selection, Stopping Criteria, Dominance filter, Random Number Generator, Fitness Evaluation and Control Finite State Machine (FSM). In chapter 4, the initialization, mutation, crossover and selection modules for DE are explained for optimizing a single fitness function. However, here the termi-

nology is slightly different and the architecture is targeted to optimize multiple fitness functions simultaneously.

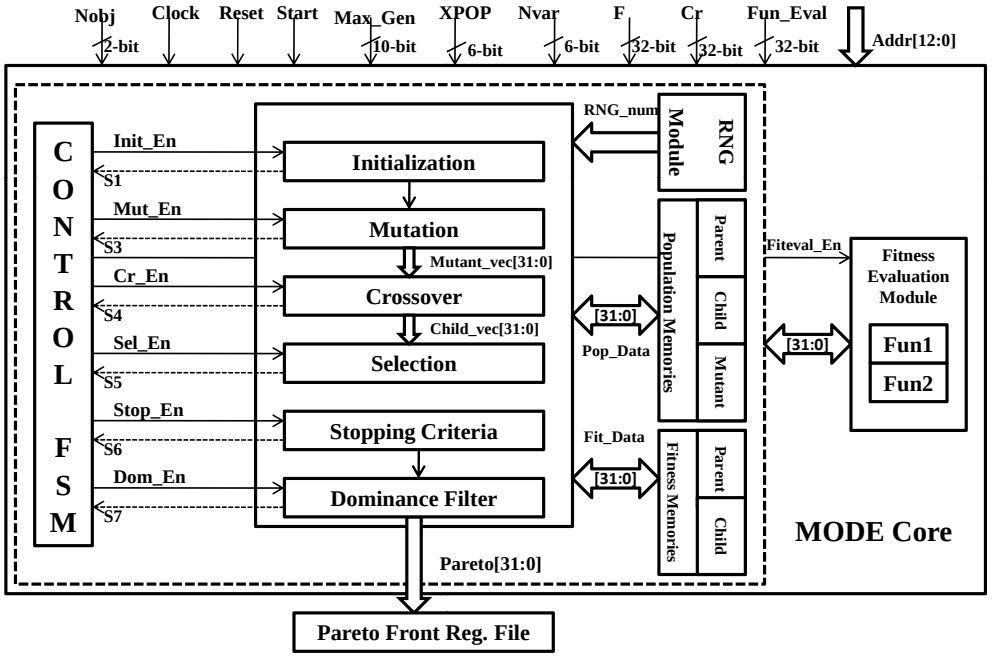

Figure 5.14: Hardware architecture of MODE algorithm

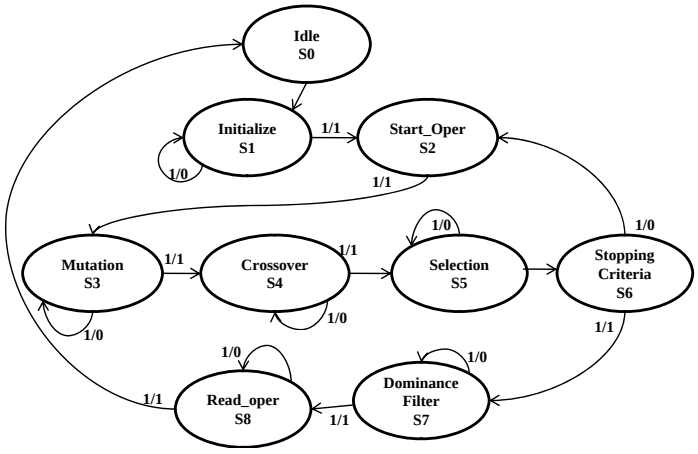

Figure 5.15: FSM diagram of MODE algorithm

The FSM has five states, i.e., *idle*, *initialize*, *operation*, *read* and *wait*. It is used to synchronize the other eight modules of the core as shown in Figure 5.15. Initially (i.e., before execution of the algorithm) all modules are in a reset condition, i.e., *idle* state. The FSM enters into the *initialize* state when the algo-

rithmic parameters are available along with the *start* signal. In this state, parent population and parent fitness memories are initialized. The hardware architecture is scalable in terms of population size ($XPOP$), dimension ($Nvar$), crossover rate (Cr), weight factor (F), maximum number of generation (Max_Gen) and function evaluations (Fun_Eval), so that the user can configure these parameters without redesigning the entire hardware. These inputs are given to the hardware IP through the processor at the beginning of execution. In the *operation* state the three main modules, i.e., mutation, crossover, and selection are executed according to the Algorithm 6. During this state remaining, i.e., initialization and dominance filter modules are in the *wait* state until the maximum number of generations or function evaluations is complete. In *read* state, the population and fitness memories are copied to temporary registers. The final parent population is input to the dominance filter after the completion of the main operation. Finally, the output of dominance filter is stored in pareto front reg file. The fitness functions are integrated into the function evaluation module of the core to avoid bus transaction overhead. Initially, two test bench functions i.e., ZDT1 and ZDT2 are optimized to evaluate the functionality of the IP.

5.3.1.1 Initialization module

This module is used to initialize the parent population for finding the optimal solution of the problem. In this module, parent population is randomly generated using random number generator module within the range of Xmin and Xmax. These values are stored in the parent population memory of 6Kbytes size using a 13-bit address. Each population member consists of $Nvar$ number of variables and each of 32-bit size. Maximum values of population size ($XPOP$) and dimension $Nvar$ is set as 50 and 30 respectively. The population values are given to the fitness evaluation module, in which two functions Fun1 and Fun2 are evaluated and the resultant fitness values (each of size 32 bit value) are stored in the parent fitness memory of size 4Kbits.

5.3.1.2 Mutation module

After initializing the parent population, FSM will be in the operation state, where the three main modules mutation, crossover, and selection are executed. First in mutation module, mutation vector is calculated for each population using mutation

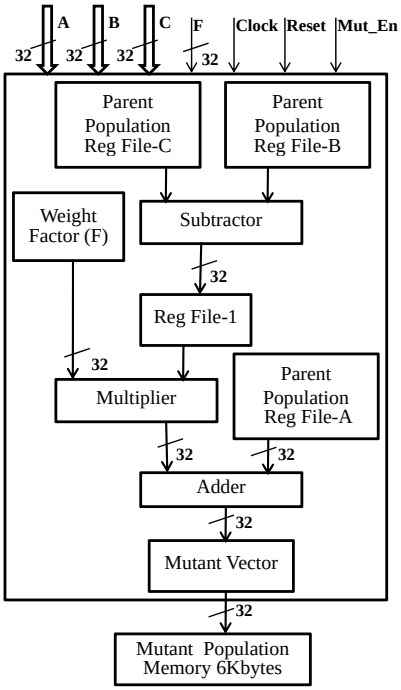

Figure 5.16: Mutation module

operation as shown in Figure 5.16. Here three distinct parent populations A, B and C are randomly selected, and mutation operation is performed with a weight factor (F). The resultant mutant vector is compared with the minimum and maximum values of solution. These vectors are stored in the mutant population memory (6Kbytes).

5.3.1.3 Crossover module

To increase the diversity among the population, crossover module generates the child population. The architecture of crossover module is shown in Figure 5.17. For each population, this module selects the child population member from parent and mutant vector memory by comparing the crossover rate (Cr) with a random number generated using the Random Number Generator (RNG) module. The output of MUX is child population, and it is stored in the child population memory (6Kbytes). Fitness value of each child population is calculated using fitness evaluation module, and the resultant values are stored in the child fitness memory (4Kbits).

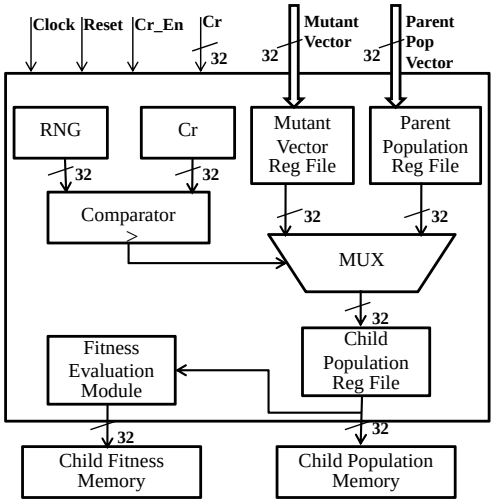

Figure 5.17: Crossover module

5.3.1.4 Selection module

In this module, each parent population and fitness values are updated by comparing with the child population. The architecture of selection module is shown in Figure 5.18. The fitness value of each parent and child population is compared to choose the best solution for next generation. If the fitness value of child population is less than the parent population, then current parent population and its fitness values are replaced with child population and its fitness values. This process is repeated for $XPOP$ times and best population, and corresponding fitness values are updated in parent population and fitness memories.

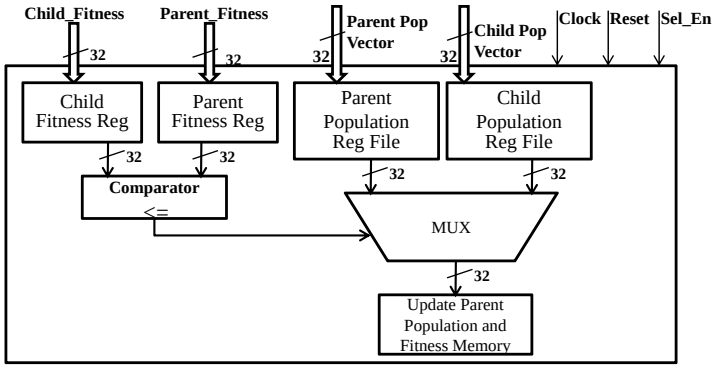

Figure 5.18: Selection module

5.3.1.5 Stopping criteria module

The three operations mutation, crossover and selection steps are performed by corresponding module sequentially for every generation and this process stops either

by reaching the maximum number of generations or maximum number of function evaluations specified by Max_Gen and Fun_Eval parameters respectively.

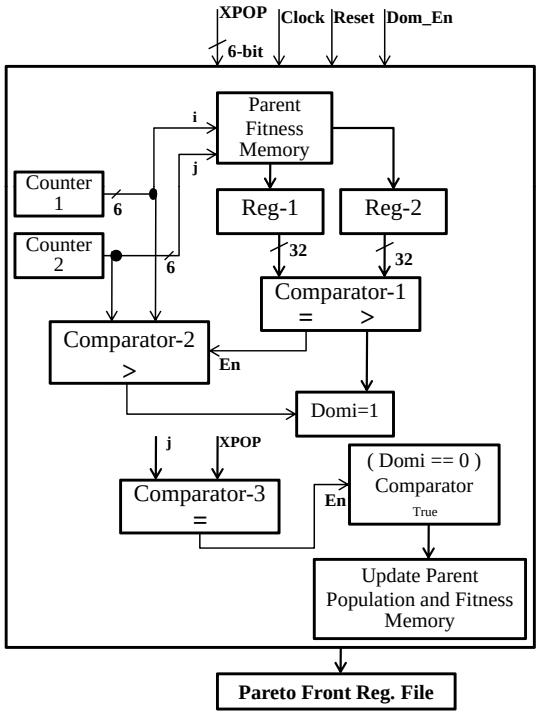

Figure 5.19: Dominance Filter module

5.3.1.6 Dominance filter

In this module, dominance criteria is used to select the non-dominated solutions. The architecture of dominance filter is shown in Figure 5.19. It gives the output as pareto optimal solutions for the functions to be optimized and store the results in a pareto front register file. The final optimal solution is selected by sorting the fitness values from the parent fitness memory according to their dominance. Two counters are used to index the fitness values from the fitness memory and compare each value with the remaining values using the comparator-1 module. The indices of the two counters are compared using the comparator-2 module. If the value of Reg-1 is greater than the value of Reg-2 then the variable $Domi$ updates with a value '1' otherwise updates with a value '0'. This process repeats $XPOP$ times and then if $Domi$=0 then the current parent population is stored in the temporary memory. Subsequently, next parent fitness value is compared with the remaining fitness values and this whole process is repeated for $XPOP$ times. The resultant pareto optimal solution, i.e., non-dominated solutions are stored in a pareto front

Reg file (4Kbits).

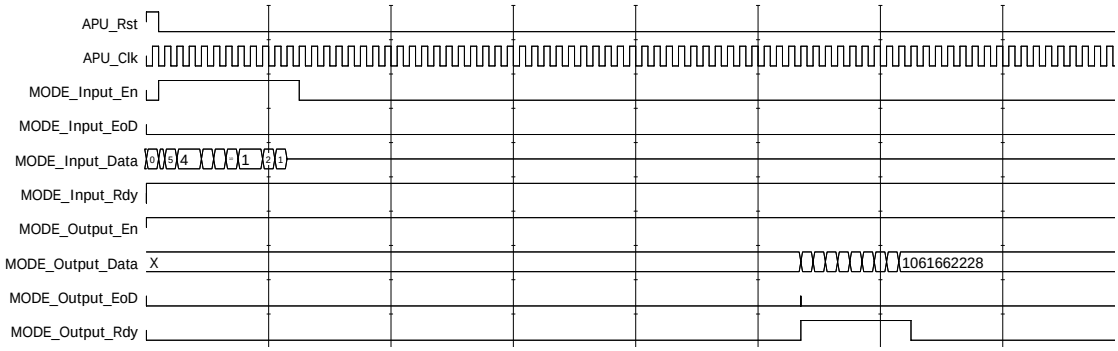

Figure 5.20: Functional simulation of MODE IP Core

5.3.2 Experimental setup

In this work, single precision floating point MODE algorithm is implemented on FPGA. Firstly the entire algorithm is coded in VHDL and functionally verified by simulating the MODE core by optimizing ZDT1 function [150] with Max_Gen=5, $XPOP$=4, $Nvar$=4, Fun_Eval=10, $Nobj$=2, Cr=0.5 and F=0.2. The simulation result is shown in Figure 5.20. From this figure, it is observed that the resultant output is available at MODE_Output_Data port when MODE_Output_Rdy is high. All floating point operations are performed using IEEE-754 supported Xilinx FPU core 5.0 [167]. Next, it is synthesized and implemented on Xilinx Virtex-5 FPGA. Further, an IP core is developed and interfaced to the PPC440 processor via APU controller interface. The execution performance of the proposed core is analyzed by integrating the core in system on chip platform as shown in Figure 5.21. The MODE IP acts as a coprocessor running at 50MHz frequency, and it is accessed using load/store instructions given by PPC440 processor. The timer is used to calculate the execution time to get the pareto optimal solution. The final solution is transferred to the UART from PPC440 processor through Processor Local Bus (PLB) to verify the results. For the software implementation, the same algorithm is coded in single precision floating point C language and executed on the PPC440 processor at 200MHz frequency.

5.3.3 Timing results

MODE algorithm is executed on both software (PPC440) and hardwired logic of Xilinx Virtex-5 FPGA to evaluate its execution time. The algorithmic control

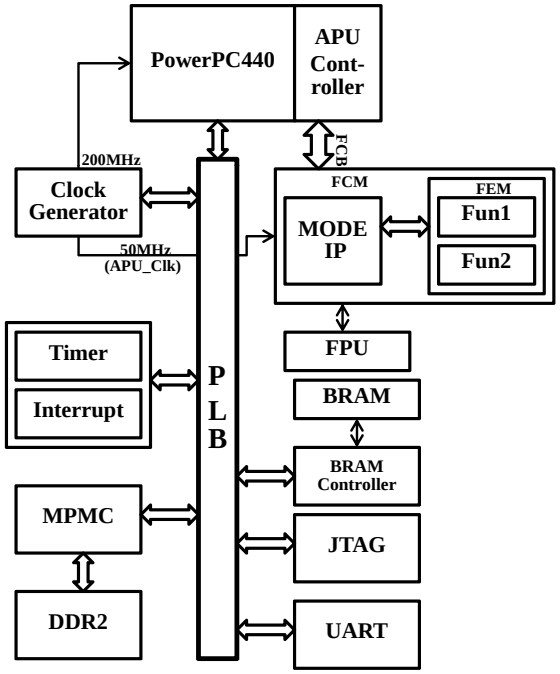

Figure 5.21: System on Chip setup

Table 5.3: Execution time of test bench functions in software (SW) and hardware (HW)

		$XPOP$ =10			$XPOP$=20			$XPOP$=30		
Test Function	Max_Gen	Float SW(ms) (Std%)	Float HW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Float HW(ms) (Std%)	Acceleration factor	Float SW(ms) (Std%)	Float HW(ms) (Std%)	Acceleration factor
$ZDT1$	10	436 (0.4)	6.34 (0.5)	68.77	875 (0.6)	12.79 (0.6)	68.41	1,314 (0.5)	18.98 (0.6)	69.23
	50	2,028 (0.6)	29.14 (0.6)	69.60	3,791 (0.7)	59.91 (0.2)	63.28	4,028 (0.4)	58.89 (0.2)	68.40
	100	3,394 (0.3)	57.94 (0.5)	58.58	4,010 (0.5)	59.91 (0.3)	66.93	4,028 (0.4)	58.89 (0.4)	68.40
	200	3,830 (0.5)	57.94 (0.4)	66.10	4,010 (0.6)	59.91 (0.4)	66.93	4,028 (0.6)	58.89 (0.3)	68.40
	300	3,830 (0.7)	57.94 (0.5)	66.10	4,010 (0.4)	59.91 (0.3)	66.93	4,028 (0.2)	58.89 (0.1)	68.40
$ZDT2$	10	222 (0.3)	3.10 (0.5)	71.61	476 (0.4)	6.21 (0.7)	76.65	714 (0.6)	9.35 (0.4)	76.36
	50	1,056 (0.4)	14.51 (0.4)	72.78	1,975 (0.3)	28.77 (0.5)	68.65	2,219 (0.5)	28.77 (0.8)	77.13
	100	2,007 (0.2)	28.74 (0.3)	69.83	2,233 (0.2)	28.77 (0.3)	77.62	2,219 (0.8)	28.77 (0.7)	77.13
	200	2,110 (0.5)	28.74 (0.5)	73.42	2,233 (0.1)	28.77 (0.5)	77.62	2,219 (0.5)	28.77 (0.5)	77.13
	300	2,110 (0.6)	28.74 (0.7)	73.42	2,233 (0.5)	28.77 (0.6)	77.62	2,219 (0.4)	28.77 (0.3)	77.13

parameters shown in Table 5.1 are input to the core. Timer is used to measure the execution time of the algorithm. Table 5.3 shows the average execution time (in ms) for optimizing test functions for different Max_Gen, $XPOP$ and $Nvar$=30. The tabulated values are the mean value of 20 independent runs. The table also

presents percentage of standard deviation (std%) of execution time for different runs in parenthesis. The execution time of the algorithm using MODE IP and PPC440 is referred as hardware (Float HW) time and software (Float SW) time respectively. It is observed that the MODE IP gives 60x speed up for ZDT1 function and 70x speed up for ZDT2 function over the software implementation of the same algorithm on the hardcore PPC440 processor. The acceleration in execution time of MODE IP is mainly due to its parellelized implementation.

Table 5.4: Timing results of Spectrum Allocation problem in software (SW) and hardware (HW) (MSR & MMR)

Population Size	Max_Gen	SW(ms) (Std%)	HW(ms) (Std%)	Acceleration factor
$XPOP$=20	50	6796.13 (0.6)	130.85 (0.5)	51.94
	100	12249.77 (0.4)	234.67 (0.3)	51.20
	200	27227.98 (0.2)	522.61 (0.5)	52.10
	300	41914.19 (0.5)	784.91 (0.4)	53.40
	400	55713.7 (0.7)	1047.25 (0.6)	53.20
	500	55713.7 (0.6)	1047.25 (0.5)	53.20
$XPOP$=40	50	13678.33 (0.3)	260.87 (0.5)	52.43
	100	24770.45 (0.5)	465.61 (0.4)	53.20
	200	52465.32 (0.7)	984.34 (0.6)	53.30
	300	67705.48 (0.6)	1297.04 (0.8)	52.20
	400	67705.48 (0.4)	1297.04 (0.7)	52.20
	500	67705.48 (0.3)	1297.04 (0.4)	52.20

In this work, MSR, MMR and MPF are optimized by considering two functions at a time using MODE IP core and obtained the pareto solution for the two objectives. Here, three scenarios are considered for optimizing MSR & MMR, MMR & MPF and MSR & MPF objective functions. In each scenario, two functions are integrated with MODE IP and its execution time (HW) is calculated using timers. The execution time is compared with the processor (SW) execution time. For validating the HW, all the network parameters (i.e., primary users, cognitive users, and channels) are chosen as 10. MODE IP is executed to assign the best available channels to SUs. Table 5.4 - 5.6 shows the mean execution time to complete the SA algorithm in three scenarios MSR & MMR, MMR & MPF and MSR & MPF for two different population sizes $XPOP$=20 and 40. From these tables, it is observed that the proposed hardware performs 51-53x speedup

Table 5.5: Timing results of Spectrum Allocation problem in software (SW) and hardware (HW) (MMR & MPF)

Population Size	Max_Gen	SW(ms) (Std%)	HW(ms) (Std%)	Acceleration factor
$XPOP$=20	50	6922.99 (0.4)	130.74 (0.6)	55.75
	100	15634.33 (0.6)	260.26 (0.2)	60.07
	200	30144.47 (0.3)	521.26 (0.3)	57.83
	300	45806.38 (0.6)	782.48 (0.5)	58.54
	400	58837.88 (0.4)	1043.78 (0.4)	56.37
	500	76527.55 (0.5)	1305.04 (0.7)	58.64
$XPOP$=40	50	15079.45 (0.5)	260.66 (0.4)	57.85
	100	29357.14 (0.4)	517.58 (0.6)	56.72
	200	59563.06 (0.5)	1036.42 (0.7)	57.47
	300	75333.13 (0.7)	1295.94 (0.5)	58.13
	400	75333.13 (0.3)	1295.94 (0.6)	58.13
	500	75333.13 (0.4)	1295.94 (0.5)	58.13

Table 5.6: Timing results of Spectrum Allocation problem in software (SW) and hardware (HW) (MMR & MPF)

Population Size	Max_Gen	SW(ms) (Std%)	HW(ms) (Std%)	Acceleration factor
$XPOP$=20	50	8128.43 (0.7)	130.44 (0.4)	62.31
	100	16682.24 (0.7)	258.82 (0.3)	64.45
	200	33126.75 (0.6)	519.88 (0.6)	63.72
	300	48341.51 (0.2)	781.34 (0.3)	61.87
	400	67589.70 (0.3)	1042.89 (0.3)	64.81
	500	83169.18 (0.6)	1304.41 (0.6)	63.76
$XPOP$=40	50	16660.32 (0.4)	260.48 (0.6)	63.96
	100	32479.24 (0.6)	517.68 (0.4)	62.74
	200	65790.23 (0.2)	1037.21 (0.5)	63.43
	300	81486.33 (0.7)	1297.14 (0.7)	62.82
	400	81486.33 (0.4)	1297.14 (0.3)	62.82
	500	81486.33 (0.2)	1297.14 (0.2)	62.82

for optimizing MSR & MMR, 55-60x speedup for optimizing MMR & MPF, whereas 61-64x speedup for optimizing MSR & MPF functions compared to its equivalent software implementation.

Table 5.7: Resource utilization

Test Function	BRAM	DSP48E	Slice Registers	Slice LUTs	Slices	LUT FF pairs	Bonded IOBs	Max Freq (MHz)
ZDT1	105 (70%)	46 (35%)	7,151 (15%)	12,663 (28%)	4,492 (40%)	5,632 (39%)	73 (11%)	96.96
ZDT2	105 (70%)	36 (28%)	5,142 (11%)	9,471 (21%)	3,428 (30%)	4,049 (38%)	73 (11%)	96.72

5.3.4 Synthesis results

The MODE hardware IP is simulated using Xilinx ISE simulator and synthesized using Xilinx Synthesis Tool (XST). Resource utilization of the MODE algorithm core with two test bench functions is shown in Table 5.7. From this table, it is observed that 70% of BRAM is consumed for the two test functions, and maximum operation frequency of the core is observed as 96MHz.

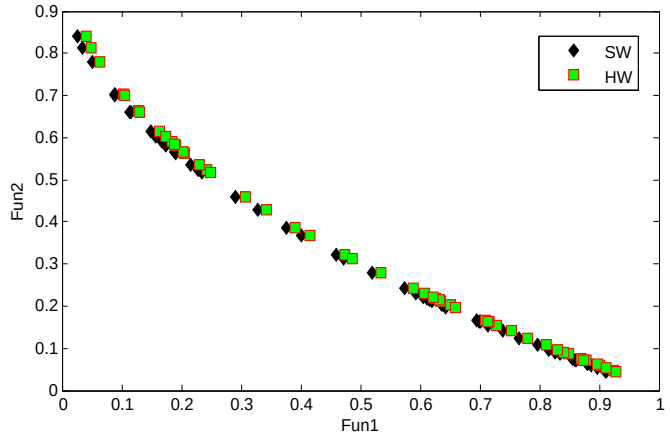

Figure 5.22: Pareto Front of ZDT1 test function

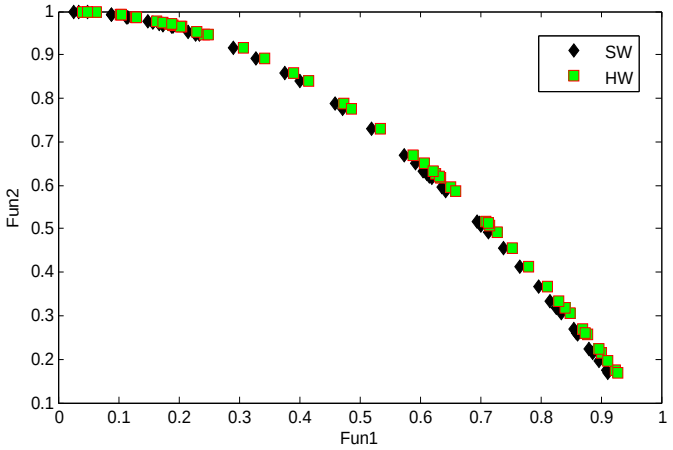

Figure 5.23: Pareto Front of ZDT2 test function

5.3.5 Pareto Front

The MODE algorithm is executed for 20 independent runs with parameters Max_Gen=500, $XPOP$=40 and $Nvar$=30. The pareto front of two test functions ZDT1 and ZDT2 are shown in Figure 5.22 and Figure 5.23 respectively. From these figures, it is observed that the pareto front obtained by both software ((SW) which runs on PPC440 processor) and hardware ((HW) which runs on coprocessor) are almost same. From this result, it is confirmed that the designed MODE coprocessor is functionally behaving correctly with accelerated execution speed.

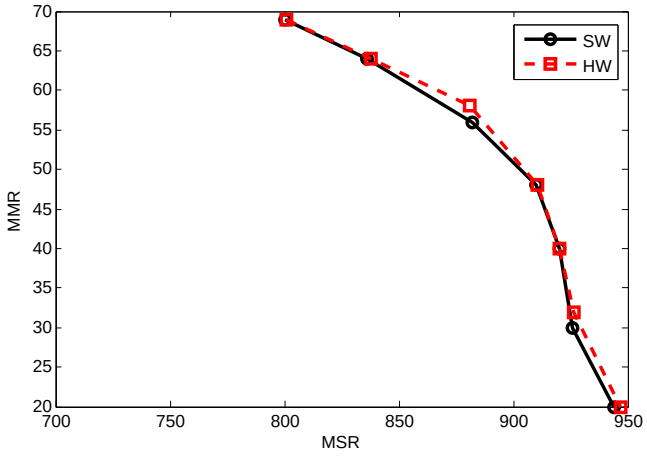

Figure 5.24: Pareto Front between MSR and MMR

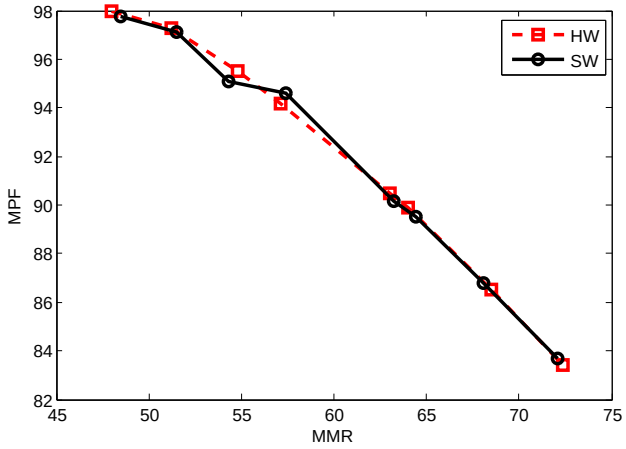

Figure 5.25: Pareto Front between MMR and MPF

The synthesis results of the system on chip implementation of the total system and MODE-SA IP is shown in Table 5.8. This table tabulates the resource utilization of the IP on Virtex-5 FPGA. From the synthesis results of the MODE-SA

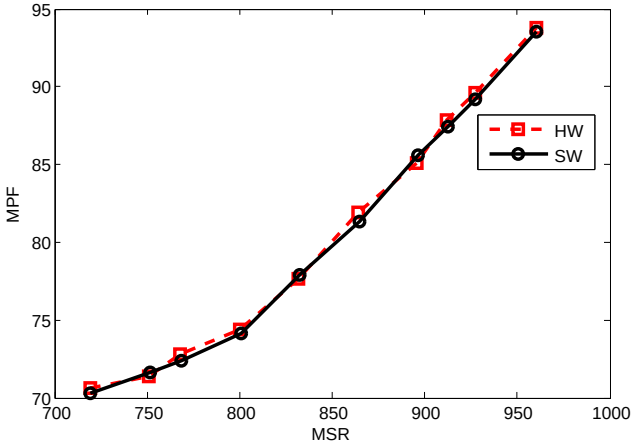

Figure 5.26: Pareto Front between MSR and MPF

Table 5.8: Device utilization of system and MODE-SA Core

Resource	Total SoC	MODE-SA
Slice Registers	10206(22%)	6,344 (14%)
Slice LUTs	16581(37%)	12,385 (27%)
LUT FF-Pairs	19500(43%)	4,724 (33%)
BRAM	80(54%)	111 (75%)
DSP48E	77(60%)	71 (55%)
Bonded IOBs	54(8%)	73 (11%)
BUFG	8(25%)	-
BUFIOs	8(10%)	-
PLL_ADVs	1(16%)	-
External IOBs	145(22%)	-
Max Freq	95.86 MHz	72.51

Table 5.9: Hierarchy power analysis of $MODE - SA$ SoC system

Resource Type	Power(mW)
Total SoC system	94.18
PowerPC440	18.82
Clockgen	39.97
APU	29.42
DDR2	5.39
SysACE_Compact Flash	0.29
RS232_Uart_1	0.13
PLB	0.02
proc_sys_reset_0	0.04
xps_timer_0	0.03
xps_timer_1	0.03
xps_intc_0	0.03
xps_bram_cntlr	0.01
xps_bram	0.00

core, it is observed that 55% of DSP48E slices are consumed, and the maximum operating frequency is 72.5MHz. Figure 5.24, 5.25 and 5.26 shows the variation of pareto fronts in three scenarios using both processor and IP core.

The power consumption of the SoC system is analyzed using XPower Analyzer tool of Xilinx ISE software. The results are tabulated in Table 5.9. It shows the power consumption of each resource of the SoC system. It is observed that MODE-SA coprocessor consumes 29.42mW power, i.e., 31% of the total power consumption of SoC system.

5.4 Conclusions

In this work, firstly SA problem is solved by optimizing Max-Sum-Reward, Max-Min-Reward, Max-Proportional-Fair and Forced termination probability simultaneously using MODE and NSGA-II algorithms. The performance of MODE algorithm is compared with NSGA-II algorithm and observed that MODE algorithm performs better in terms of time complexity and pareto optimal solutions for solving the SA problem. The use of forced termination probability as an objective function gives an improved channel assignment solution with minimum termination probability. Next, a joint spectrum and power allocation algorithm is used to increase the utilization of network resources by selecting suitable channel along with optimally transmitted power. The joint spectrum and power allocation problem is solved using DE and PSO algorithms to maximize the efficiency of spectrum utilization. From the simulation results, it is observed that the DE algorithm outperforms PSO algorithm in terms of quality of the solution, i.e., maximizing the average SU capacity. Further, the work is extended to study the dependency of a user sum capacity over multiple channels to the total network utilization. For this, the network utilization function i.e., Max-Sum-Reward (MSR) and average user sum capacity are optimized simultaneously to maximize the network performance. The above joint channel and power allocation problem is formulated as a multi-objective optimization problem and optimized the two objective functions simultaneously, and the trade-off solutions are obtained under the power constraints and QoS requirements of PUs.

Finally, Multi-Objective Differential Evolution based SA algorithm is implemented on FPGA. Initially, a MODE IP is developed. The IP is scalable in terms of algorithmic parameters such as population size, dimension, crossover rate, weight factor, maximum number of generation and number of function evaluations. The developed IP is synthesized and interfaced to PPC440 processor of Virtex-5 FPGA via APU controller. The hardware implementation results demon-

strated 60x speedup over software implementation while optimizing test functions. Finally, MODE-SA IP is developed by integrating the network utility functions along with the MODE IP and it is connected to PPC440 processor as a coprocessor. The three functions MSR, MMR and MPF are optimized by considering two functions simultaneously in three different scenarios. The hardware solution attained an acceleration of 50-60x compared to equivalent software implementation on PPC440 processor. Thus, it can be concluded that the proposed hardware solution is suitable to solve the SA problem using multi-objective differential evolution algorithm and also it can be used to solve multi-objective optimization problems for different applications in embedded platform.

Chapter 6

Conclusions and Future work

6.1 Conclusions

In the present work, three network utility functions, namely Max-Sum-Reward (MSR), Max-Min-Reward (MMR) and Max-Proportional-Fair (MPF) are optimized for fair allocation of channels to CR users. Three optimization algorithms, namely DE, PSO and Firefly are used to solve the SA problem and compared the performance of these algorithms in terms of quality of solution and time complexity. From the simulation results, it is observed that DE algorithm improved quality of solution (fitness value of MPF) is approximately 19% and 30%, whereas the time complexity is improved by 46% and 242% compared to Firefly and PSO algorithms respectively.

In the distributed network architecture, each SU need to perform the CR operations locally. As each SU implicitly has an embedded computing platform, the SA task need to be performed on a dedicated hardware platform to improve the overall system execution performance. Hence, a DE based SA coprocessor is proposed to accelerate the execution performance. DE hardware IP is integrated with SA objective functions to develop the DE-based SA IP and verified its functionality in a Xilinx Virtex-5 FX70T FPGA based SoC platform. The DE-SA IP is scalable in terms of selecting number of PUs, SUs and available channels. The acceleration factor was evaluated at different network and algorithmic configurations and found to be 5.19-6.91x and 76.79-105x over the fixed and floating point implementation of the SA algorithm on the PPC440 processor respectively.

The spectrum assignment solution must satisfy the multiple objectives simultaneously to provide best channels to the requested SUs. Forced termination probability is a performance metric of assignment solution provided by SA method. Thus in this work, the forced termination probability is included in objective function along with three network utility functions, namely MSR, MMR and MPF related to quality of service and are optimized simultaneously. Multi-Objective

Differential Evolution and Non-dominated Sorting Genetic algorithms are used to find the trade-off solutions between the network utility functions and forced termination probability. It is observed that the MODE algorithm performs better in terms of quality of solution and time complexity. It is necessary to optimize the individual transmitted power of each user during SA to increase efficiency of channel assignment solution. Power allocation also plays an important role in interference management and energy saving in SUs. Hence, a joint spectrum and power allocation model is formulated as an optimization problem and solved using DE and PSO algorithms. From the simulation results, it is observed that the DE algorithm improved the quality of solution (average SU capacity) is approximately 47% compared to PSO algorithm.

Further, the network utility function (MSR) and channel capacity of individual user are optimized simultaneously using MODE algorithm subject to PU outage probability and power constraints posed by SUs. The performance of MODE algorithm is compared with NSGA-II algorithm in terms of quality of solution. From the results, it is observed that the MODE algorithm improved the performance in terms of quality of the solution (average SU capacity and MSR value) approximately 54% compared to NSGA-II algorithm. Further to accelerate the execution speed of the MODE based SA algorithm, it is implemented on Xilinx Virtex 5 FPGA. Initially, a MODE coprocessor is designed and verified its functionality by optimizing two benchmark test functions. The hardware implementation results demonstrated 60x speedup over software implementation of same MODE algorithm on PPC440 processor in Virtex 5 FPGA development board. Finally, MODE based SA IP is developed by integrating the network utility functions along with the MODE IP and connected as a coprocessor to PPC440 processor. The trade-off solutions between the three functions MSR, MMR and MPF are obtained. This hardware solution attained an acceleration of 50-60x compared to equivalent software implementation on PPC440 processor.

6.2 Future work

In this thesis, SA problem is solved using different evolutionary algorithms and implemented on FPGA to accelerate the execution speed of SA algorithm. There are several interesting directions for further research and development based on the work in this thesis. Different variants of DE and MODE algorithms can be

implemented in hardware to improve the quality of solution. These hardware IPs can also be used to solve other real-time optimization problems like parameter adaption in CR and other signal processing applications.

The present work is based on the assumption that SUs and PUs are randomly deployed and based on their positions the proposed algorithm works to solve the SA problem. However, there is a need to integrate the spectrum sensing phase prior to SA task. The spectrum sensing task provides channel characteristics, detection of PU transmissions and its modulation schemes. Based on this information, SA algorithm provides the best channel to the SU without any interference to the PUs. Further, both spectrum sensing and SA algorithms need to be implemented on hardware to accelerate the execution performance such that it maximizes the spectrum utilization.

Appendix A

A.1 Embedded Development Kit design flow for hardware-software co-design

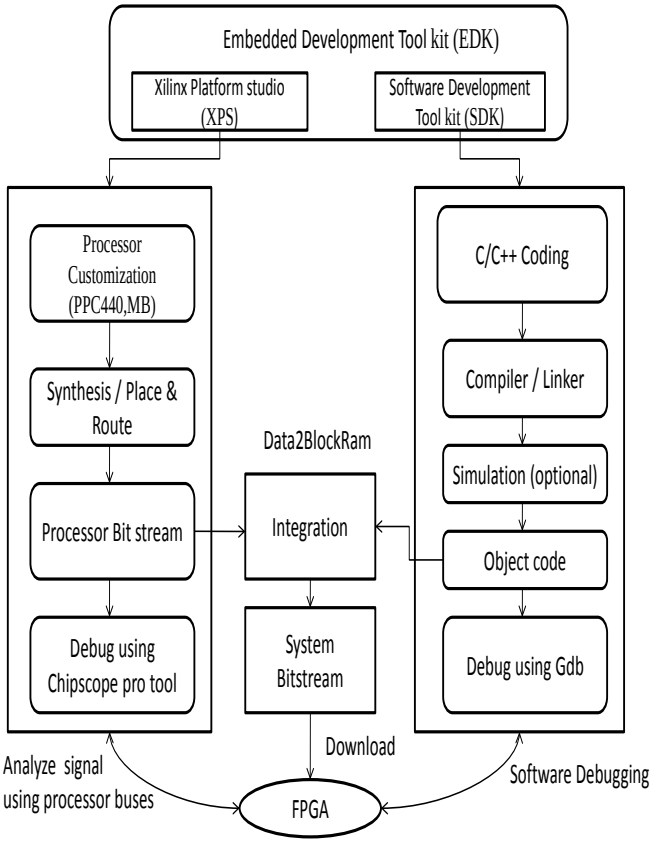

Figure A.1: Hardware software co-design approach using Embedded Development Kit

A.2 Benchmark test functions for single-objective optimization

Function 1 (Two variables): Rosenbrock function:
$f(x) = 100.(x_2 - x_1^2)^2 + (1 - x_1)^2$
Search domain:$-9 < x_j < 11 \;\; j = 1, 2$
One global optimum with $f = 0 \;\; at \;\; (1, 1)$

Function 2 (Two variables): Goldstein function:
$f(x) = [1 + (x_1 + x_2 + 1)^2 \times (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
Search domain:$-2 < x_j < 2, \;\; j = 1, 2.$
One global optimum with $f = 3 \;\; at \; (0, -1)$

Function 3 (Three variables): Sphere function:
$f(x) = x_1^2 + x_2^2 + x_3^2$
Search domain: $-5.12 < x_j < 5.12 \quad j = 1, 2, 3$
One global optimum with $f = 0 \;\; at \;\; (0, 0, 0)$

Function 4 (Four variables): Variably dimensioned function:
$f(x) = \sum_{i=1}^{4}(x_i - 1)^2 + \left[\sum_{i=1}^{4} i(x_i - 1)\right]^2 + \left[\sum_{i=1}^{4} i(x_i - 1)\right]^4$
Search domain:$-9 < x_j < 11, \;\; j = 1, 2, 3, 4$
One global optimum with $f = 0 \; at \; (1, 1, 1, 1)$

Function 5 (32 variables): De Jong function:
$f(x) = \sum_{i=1}^{D} x_i^2 \;\; x = [x_1, x_2, x_3, ..., x_D]$
Search domain:$-100 < x_j < 100 \;\; j = 1, 2, .., 32$
One global optimum with $f(x) = 0 \;\; at \;\; (0, 0, .., 0)$

Function 6 (32 variables): Schwefel's Problem 1.2
$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_i\right)^2$
Search domain:$-100 < x_j < 100, \;\; j = 1, 2, .., 32$
One global optimum with $f = 0 \; at \; (0, 0, .., 0).$

A.3 Benchmark test functions for multi-objective optimization

1. Zitzler-Deb-Thiele 1 (ZDT1) function:

$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) & = x_1 \\ f_2(\boldsymbol{x}) & = g(\boldsymbol{x})\, h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\ g(\boldsymbol{x}) & = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i \\ h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) & = 1 - \sqrt{\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}} \end{cases}$$

2. Zitzler-Deb-Thiele 2 (ZDT2) function:

$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) & = x_1 \\ f_2(\boldsymbol{x}) & = g(\boldsymbol{x})\, h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\ g(\boldsymbol{x}) & = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i \\ h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) & = 1 - \left(\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)^2 \end{cases}$$

References

[1] Mark A McHenry. NSF spectrum occupancy measurements project summary. Shared spectrum company report, 2005.

[2] Ian F Akyildiz, Won-Yeol Lee, Mehmet C Vuran, and Shantidev Mohanty. NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey. Computer Networks, 50(13):2127–2159, 2006.

[3] Won Yeol Lee. Spectrum Management in Cognitive Radio Wireless Networks. PhD thesis, Electrical and Computer Engineering, Georgia Institute of Technology, Germany, April 2009.

[4] E.Z. Tragos, S. Zeadally, A.G. Fragkiadakis, and V.A. Siris. Spectrum Assignment in Cognitive Radio Networks: A Comprehensive Survey. IEEE Communications Surveys Tutorials, 15(3):1108–1135, 2013.

[5] Kyou Woong Kim. Exploiting Cyclostationarity for Radio Environmental Awareness in Cognitive Radios. Technical report, Virginia Polytechnic Institute, 2008.

[6] Spectrum policy task force report, FCC (Federal Communications Commission) 02-155, 2002.

[7] Joseph Mitola III. Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio. PhD thesis, Royal Institute of Technology (KTH), Sweden, May 2000.

[8] Chunyi Peng, Haitao Zheng, and Ben Y Zhao. Utilization and Fairness in Spectrum Assignment for Opportunistic Spectrum Access. Mobile Networks and Applications, 11(4):555–576, 2006.

[9] Zhijin Zhao, Zhen Peng, Shilian Zheng, and Junna Shang. Cognitive Radio Spectrum Allocation using Evolutionary Algorithms. IEEE Transactions on Wireless Communications, 8(9):4421–4425, 2009.

[10] F. Koroupi, S. Talebi, and H. Salehinejad. Cognitive radio networks spectrum allocation: An ACS perspective . Scientia Iranica , 19(3):767 – 773, 2012.

[11] H.Y. Gao and J.L. Cao. Quantum-inspired bee colony optimization algorithm and its application for cognitive radio spectrum allocation. Zhongnan Daxue Xuebao (Ziran Kexue Ban)/Journal of Central South University (Science and Technology), 43(12):4743–4749, 2012.

[12] Q. Liu, H. Niu, W. Xu, and D. Zhang. A service-oriented spectrum allocation algorithm using enhanced PSO for cognitive wireless networks. Computer Networks, 74:81–91, 2014.

[13] Gao, Hong-yuan and Cao, Jin-long. Non-dominated sorting quantum particle swarm optimization and its application in cognitive radio spectrum allocation. Journal of Central South University, 20(7):1878–1888, 2013.

[14] Xiaorong Zhu, Lianfeng Shen, and T.-S.P. Yum. Analysis of Cognitive Radio Spectrum Access with Optimal Channel Reservation. IEEE Communications Letters, 11(4):304–306, April 2007.

[15] J. Martinez-Bauset, V. Pla, and D. Pacheco-Paramo. Comments on "analysis of cognitive radio spectrum access with optimal channel reservation". IEEE Communications Letters, 13(10):739–739, October 2009.

[16] Won-Yeol Lee and I.F. Akyildiz. Joint Spectrum and Power Allocation for Inter-Cell Spectrum Sharing in Cognitive Radio Networks. In Proc. 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), pages 1–12, 2008.

[17] Jelena Vucetic. A hardware implementation of channel allocation algorithms based on a space-bandwidth model of a cellular network. IEEE Transactions on Vehicular Technology, 42(4):444–455, 1993.

[18] Paul Kolodzy. Spectrum policy task force: findings and recommendations. In Proc. International Symposium on Advanced Radio Technologies (ISART), volume 3, 2003.

[19] Joseph Mitola III and Gerald Q Maguire Jr. Cognitive radio: making software radios more personal. IEEE Personal Communications, 6(4):13–18, 1999.

[20] Simon Haykin. Cognitive radio: brain-empowered wireless communications. IEEE Journal on Selected Areas in Communications, 23(2):201–220, 2005.

[21] K.C. Chen, Y.J. Peng, N. Prasad, Y.C. Liang, and S. Sun. Cognitive Radio Network Architecture: Part I - General Structure. In Proc. 2nd International Conference on Ubiquitous Information Management and Communication (ICUIMC), pages 114–119, 2008.

[22] E. Axell, G. Leus, E.G. Larsson, and H.V. Poor. Spectrum Sensing for Cognitive Radio : State-of-the-Art and Recent Advances. IEEE Signal Processing Magazine, 29(3):101–116, May 2012.

[23] T. Yucek and H. Arslan. A survey of spectrum sensing algorithms for cognitive radio applications. IEEE Communications Surveys Tutorials, 11(1):116–130, Jan 2009.

[24] Yonghong Zeng, Ying-Chang Liang, Zhongding Lei, Ser Wah Oh, F. Chin, and Sumei Sun. Worldwide Regulatory and Standardization Activities on Cognitive Radio. In Proc. IEEE Symposium on New Frontiers in Dynamic Spectrum, pages 1–9, April 2010.

[25] Carlos Cordeiro, Kiran Challapali, Dagnachew Birru, and Sai Shankar N. IEEE 802.22: An Introduction to the First Wireless Standard based on Cognitive Radios. Journal of Communications, 1(1):38–47, 2006.

[26] IEEE Standard Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management. IEEE Std 1900.1-2008, pages 1–62, Oct 2008.

[27] J.A. Hoffmeyer. IEEE 1900 and ITU-R Standardization Activities in Advanced Radio Systems and Spectrum Management. In Proc. 4th IEEE Consumer Communications and Networking Conference, pages 1159–1163, Jan 2007.

[28] IEEE 1900.2, "http://grouper.ieee.org/groups/dyspan/2/index.html".

[29] IEEE 1900.3, "http://grouper.ieee.org/groups/dyspan/3/index.html".

[30] S. Buljore, H. Harada, P. Houze, K. Tsagkaris, V. Ivanov, K. Nolte, T. Farnham, O. Holland, and M. Stamatelatos. IEEE P1900.4 Standard: Reconfiguration of multi-radio systems. In Proc. IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering, pages 413–417, July 2008.

[31] IEEE 802.16, "http://www.ieee802.org/16/".

[32] IEEE 802.11, "http://www.ieee802.org/11/".

[33] IEEE 802.19, "http://www.ieee802.org/19/".

[34] ETSI TR 102 838: Reconfigurable Radio Systems (RRS): Summary of feasibility studies and potential standardization topics. Technical report, 2009.

[35] ECMA International Standard. MAC and PHY for Operation in TV White Space . Technical report, December 2012.

[36] SCC, "http://www.scc41.org/".

[37] R. Venkatesha Prasad, P. Pawelczak, J.A. Hoffmeyer, and H.S. Berger. Cognitive functionality in next generation wireless networks: standardization efforts. IEEE Communications Magazine, 46(4):72–78, April 2008.

[38] H. Skalli, S. Ghosh, S.K. Das, L. Lenzini, and M. Conti. Channel Assignment Strategies for Multiradio Wireless Mesh Networks: Issues and Solutions. IEEE Communications Magazine, 45(11):86–95, November 2007.

[39] M.K. Marina and S.R. Das. A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks. In Proc. 2nd International Conference on Broadband Networks, pages 381–390 Vol. 1, Oct 2005.

[40] A.K. Das, H.M.K. AlAzemi, R. Vijayakumar, and S. Roy. Optimization models for fixed channel assignment in wireless mesh networks with multiple radios. In Proc. Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, pages 463–474, Sept 2005.

[41] Nie, Cristina Comaniciu, and Prathima Agrawal. A Game Theoretic Approach to Interference Management in Cognitive Networks. In Wireless Communications, volume 143 of The IMA Volumes in Mathematics and its Applications, pages 199–219. Springer New York, 2007.

[42] N. Nie and Cristina Comaniciu. Adaptive channel allocation spectrum etiquette for cognitive radio networks. In Proc. First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), pages 269–278, Nov 2005.

[43] Zhang Hongshun and Yan Xiao. Advanced Dynamic Spectrum Allocation Algorithm Based on Potential Game for Cognitive Radio. In Proc. 2nd International Symposium on Information Engineering and Electronic Commerce (IEEC), pages 1–3, July 2010.

[44] Gan Liu, Liang Zhou, Kan Xiao, Bo Yu, Guangxi Zhou, Biao Wang, and Guangxi Zhu. Receiver-Centric Channel Assignment Model and Algorithm in Cognitive Radio Network. In Proc. 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), pages 1–4, Oct 2008.

[45] Elias Tragos, George T. Karetsos, Sofoklis A. Kyriazakos, and Kostas Vlahodimitropoulos. Dynamic segmentation of cellular networks for improved handover performance. Wireless Communications and Mobile Computing, 8(7):907–919, 2008.

[46] Anthony Plummer and Subir Biswas. Distributed spectrum assignment for cognitive networks with heterogeneous spectrum opportunities. Wireless Communications and Mobile Computing, 11(9):1239–1253, 2011.

[47] Gan Liu, Liang Zhou, Kan Xiao, Bo Yu, Guangxi Zhou, Biao Wang, and Guangxi Zhu. Receiver-Centric Channel Assignment Model and Algorithm in Cognitive Radio Network. In Proc. 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), pages 1–4, Oct 2008.

[48] Anh Tuan Hoang and Ying-Chang Liang. Downlink Channel Assignment and Power Control for Cognitive Radio Networks. IEEE Transactions on Wireless Communications, 7(8):3106–3117, August 2008.

[49] Anh Tuan Hoang and Ying-Chang Liang. Maximizing Spectrum Utilization of Cognitive Radio Networks Using Channel Allocation and Power Control. In Proc. IEEE 64th Vehicular Technology Conference, pages 1–5, Sept 2006.

[50] Xi Su, Chaowei Yuan, and Shuqun Shen. A New Mechanism of Dynamic Spectrum Allocation in the Cognitive Network. In Proc. 5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCom), pages 1–4, Sept 2009.

[51] Tao Zhang, Bin Wang, and Zhiqiang Wu. Spectrum assignment in infrastructure based cognitive radio networks. In Proc. IEEE 2009 National Aerospace Electronics Conference (NAECON), pages 69–74, July 2009.

[52] Lili Yang, Xianzhong Xie, and Yi Zheng. A Historical-Information-Based Algorithm in Dynamic Spectrum Allocation. In Proc. International Conference on Communication Software and Networks (ICCSN), pages 731–736, Feb 2009.

[53] Prabhjot Kaur, Moin Uddin, and Arun Khosla. Adaptive Bandwidth Allocation Scheme for Cognitive Radios. International J. Advancements in Computing Technology, 2(2):35–41, 2010.

[54] Wenzhu Zhang and Xuchen Liu. Centralized Dynamic Spectrum Allocation in Cognitive Radio Networks Based on Fuzzy Logic and Q-Learning. China communications, 8(1):46–54, 2011.

[55] Huahui Wang, Jian Ren, and Tongtong Li. Resource Allocation with Load Balancing for Cognitive Radio Networks. In Proc. IEEE Global Telecommunications Conference (GLOBECOM), pages 1–5, Dec 2010.

[56] Anh Tuan Hoang and Ying-Chang Liang. Downlink Channel Assignment and Power Control for Cognitive Radio Networks. IEEE Transactions on Wireless Communications, 7(8):3106–3117, August 2008.

[57] Li Yu, Cong Liu, and Wenyu Hu. Spectrum allocation algorithm in cognitive ad-hoc networks with high energy efficiency. In Proc. International Conference on Green Circuits and Systems (ICGCS), pages 349–354, June 2010.

[58] Dong Heon Lee, Wha Sook Jeon, and Dong Geun Jeong. Joint Channel Assignment and Routing in Cognitive Radio-Based Wireless Mesh Networks. In Proc. IEEE 71st Vehicular Technology Conference (VTC), pages 1–5, May 2010.

[59] Mario Bkassiny and SudharmanK. Jayaweera. Optimal Channel and Power Allocation for Secondary Users in Cooperative Cognitive Radio Networks. In Proc. of 2nd International Conference on Mobile Lightweight Wireless Systems, volume 45, pages 180–191. 2010.

[60] Qin Xin and Jie Xiang. Joint QoS-aware admission control, channel assignment, and power allocation for cognitive radio cellular networks. In Proc. IEEE 6th International Conference on Mobile Adhoc and Sensor Systems (MASS), pages 294–303, Oct 2009.

[61] H.B. Salameh, M. Krunz, and O. Younis. Distance- and Traffic-Aware Channel Assignment in Cognitive Radio Networks. In Proc. 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pages 10–18, June 2008.

[62] E. A. Silver. An Overview of Heuristic Solution Methods. The Journal of the Operational Research Society, 55(9):936–956, 2004.

[63] Jorge J Moré and Stefan M Wild. Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization, 20(1):172–191, 2009.

[64] Thomas Back, David B Fogel, and Zbigniew Michalewicz. Handbook of evolutionary computation. IOP Publishing Ltd., 1997.

[65] Fang Ye, Rui Yang, and Yibing Li. Genetic Algorithm Based Spectrum Assignment Model in Cognitive Radio Networks. In Proc. 2nd International Conference on Information Engineering and Computer Science (ICIECS), pages 1–4, Dec 2010.

[66] M.Y. ElNainay, Feng Ge, Ying Wang, A.E. Hilal, Yongsheng Shi, A.B. MacKenzie, and C.W. Bostian. Channel allocation for dynamic spectrum access cognitive networks using Localized island Genetic Algorithm. In Proc. 5th International Conference on Testbeds and Research Infrastructures for

the Development of Networks Communities and Workshops (TridentCom), pages 1–3, April 2009.

[67] M.Y. El Nainay, D.H. Friend, and A.B. MacKenzie. Channel Allocation and Power Control for Dynamic Spectrum Cognitive Networks using a Localized Island Genetic Algorithm. In Proc. 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), pages 1–5, Oct 2008.

[68] M Mahdavi, Mohammad Fesanghary, and E Damangir. An improved harmony search algorithm for solving optimization problems. Applied mathematics and computation, 188(2):1567–1579, 2007.

[69] J. Del Ser, M. Matinmikko, S. Gil-Lopez, and M. Mustonen. A novel Harmony Search based spectrum allocation technique for cognitive radio networks. In Proc. 7th International Symposium on Wireless Communication Systems (ISWCS), pages 233–237, Sept 2010.

[70] M. Dorigo and L.M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1):53–66, April 1997.

[71] H. Salehinejad, S. Talebi, and F. Pouladi. A metaheuristic approach to spectrum assignment for opportunistic spectrum access. In Proc. IEEE 17th International Conference on Telecommunications (ICT), pages 234–238, April 2010.

[72] L. Liu, G. Hu, and Y. Peng. Swarm Intelligence based Distributed Spectrum Allocation for Cognitive Networks. In Proc. International Conference on Future Information Technology, volume 13, pages 357–361, 2011.

[73] Dervis Karaboga and Bahriye Akay. A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems. Applied Soft Computing, 11(3):3021 – 3031, 2011.

[74] Xiaoya Cheng and Mingyan Jiang. Cognitive radio spectrum assignment based on artificial bee colony algorithm. In Proc. IEEE 13th International Conference on Communication Technology (ICCT), pages 161–164, Sept 2011.

[75] G. Kulkarni, S. Adlakha, and M. Srivastava. Subcarrier allocation and bit loading algorithms for OFDMA-based wireless networks. IEEE Transactions on Mobile Computing, 4(6):652–662, Nov 2005.

[76] Anh Tuan Hoang and Ying-Chang Liang. A Two-Phase Channel and Power Allocation Scheme for Cognitive Radio Networks. In Proc. IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications, pages 1–5, Sept 2006.

[77] Jiandong Li, Dong Chen, Weiying Li, and Jing Ma. Multiuser Power and Channel Allocation Algorithm in Cognitive Radio. In Proc. International Conference on Parallel Processing (ICPP), pages 72–72, Sept 2007.

[78] M. Haddad, AM. Hayar, G.E. Øien, and S.G. Kiani. Uplink Distributed Binary Power Allocation for Cognitive Radio Networks. In Proc. 3rd International Conference on Cognitive Radio Oriented Wireless Networks and Communications, pages 1–4, May 2008.

[79] Bassem Zayen, Majed Haddad, Aawatif Hayar, and Geir E.Øien. Binary power allocation for cognitive radio networks with centralized and distributed user selection strategies. Physical Communication, 1(3):183 – 193, 2008.

[80] Walid Farid Abdelfatah, Jacques Georgy, Umar Iqbal, and Aboelmagd Noureldin. FPGA-Based Real-Time Embedded System for RISS/GPS Integrated Navigation. Sensors, 12(1):115–147, 2011.

[81] Mohd Nazrin Md Isa. High Performance Reconfigurable Architectures for Biological Sequence Alignment, March 2013.

[82] "http://www.xilinx.com/products/boards-and-kits/HW-V5-ML507-UNI-G.html" Xilinx ML507 development board.

[83] Youn-Long Steve Lin. Essential Issues in SOC Design Designing Complex Systems-on-Chip. Springer, P.O. Box 17, 3300 AA Dordrecht, Netherlands, 2006.

[84] Don Davis, Srinivas Beeravo, Ranjesh Jaganathan. Hardware/Software Codesign for Platform FPGAs. Xilinx Application Notes, 2005.

[85] J. A. Darringer, R. A. Bergamaschi, S. Bhattacharya, D. Brand, A. Herkersdorf, J. K. Morrell, I. Nair, P. Sagmeister, and Y. Shin. Early analysis tools for system-on-a-chip design. IBM J. Res. Dev., 46(6):691–707, nov 2002.

[86] Aaron Richard Mandle. FPGA Based Hardware Acceleration: A Case Study in Protein Identification, April 2008.

[87] Ron Sass and Andrew G. Schmidt. Embedded Systems Design with Platform FPGAs Principles and Practices. Elsevier Inc, USA, 2010.

[88] Tomáś Brabec. Algorithm Acceleration using FPGAs. Technical report, Czech Technical University, Prague, 2004.

[89] Yi Zou. Coprocessor Acceleration for Domain-Specific Computing. PhD thesis, Department of Computer Science, University of California, Los Angeles, USA, 2012.

[90] Ahmad Ansari, Peter Ryser, and Dan Isaacs. Accelerated System Performance with APU-enhanced processing. Xcell Journal, first quarter, 2005.

[91] Yanhong Liu. System-level modeling and analysis of multimedia-soc platforms. PhD thesis, Institute of Computing Technology, National University of Singapore, 2007.

[92] PowerPC 405 Processor Block Reference Guide. Xilinx User guide, (018), 2010.

[93] Virtex-5 FPGA Embedded Processor Block with PowerPC 440 Processor . Xilinx Data sheet, (621), 2011.

[94] Pei-Yin Chen, Ren-Der Chen, Yu-Pin Chang, Leang san Shieh, and H.A. Malki. Hardware Implementation for a Genetic Algorithm. IEEE Transactions on Instrumentation and Measurement, 57(4):699 –705, Apr 2008.

[95] Xilinx MicroBlaze Processor. http://www.xilinx.com/tools/microblaze.html [Accessed :21-07-2013].

[96] Vincent Andrew Akpan. Development of New Model-Based Adaptive Predictive Control Algorithms and their Implementation on Real-Time Embedded Systems. PhD thesis, Department of Electrical and Computer Engineering, Thessaloniki, Greece, 2011.

[97] Preeti Ranjan Panda, Nikil D Dutt, and Alexandru Nicolau. On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems. ACM Transactions on Design Automation of Electronic Systems (TODAES), 5(3):682–704, 2000.

[98] Doug Amos, Austin Lesea, and René Richter. FPGA-Based Prototyping Methodology Manual Best Practices in Design-for-Prototyping. Synopsys, Inc, Mountain View, CA, USA, 2010.

[99] Harn Hua Ng and Latha Pillai. Accelerated system performance with the APU controller and XtremeDSP slices. Xilinx Application Notes, (717), 2005.

[100] EECG Toronto. "http://www.eecg.toronto.edu/ pc/courses/432/".

[101] Tue University, "http://www.win.tue.nl/ wsinmak/Education/2IN35/lab/".

[102] Xilinx. Reference Guide: Embedded Processor Block in Virtex-5 FPGAs. Xilinx User guide, (200), 2008.

[103] Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs. Xilinx Application Notes, (462), 2006.

[104] Tien-Lung Lee and Neil W Bergmann. Interfacing methodologies for IP re-use in reconfigurable system-on-chip. In Proc. Microelectronics, MEMS, and Nanotechnology, pages 454–463. International Society for Optics and Photonics, 2004.

[105] Rangababu Peesapati, Samrat L. Sabat, Karthik K. P., Narasimhappa M., Giribabu N., and J. Nayak. FPGA-based embedded platform for fiber optic gyroscope signal denoising. International Journal of Circuit Theory and Applications, 42(7):744–757, 2014.

[106] Jelena F. Vucetic and Dragomir D. Dimitrijevic. Adaptive Channel Allocation in Cellular Networks With Multi-User Platforms. In Proc. 3rd WINLAB Workshop Third-Gen Wireless Communications, volume 217 of The Springer International Series in Engineering and Computer Science, pages 243–258. Springer US, 1993.

[107] J.F. Vucetic and D.D. Dimitrijevic. Extended integrated channel manager (EICM)-an architecture for fast adaptive channel allocation in cellular networks with multi-terminal platforms. In Proc. IEEE International Conference on Communications (ICC), pages 1316–1322 vol.3, Jun 1992.

[108] Xilinx. Synthesis and Simulation Design Guide. Xilinx User guide, (626), 2006.

[109] Xilinx. EDK Concepts, Tools, and Technique: A Hands-On Guide to Effective Embedded System Design . Xilinx User guide, (683), 2011.

[110] Tutorial for FPGA based SoC system, "http://www.fpgadeveloper.com/".

[111] J. Abril, F. Comellas, A. Cortes, J. Ozon, and M. Vaquer. A multiagent system for frequency assignment in cellular radio networks. IEEE Transactions on Vehicular Technology, 49(5):1558–1565, Sep 2000.

[112] Rainer Storn and Kenneth Price. Differential Evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, 11(4):341–359, Dec 1997.

[113] S. Das and P.N. Suganthan. Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation, 15(1):4 –31, Feb. 2011.

[114] V Nithish Kumar, Harsha Bhalavi, G Lakshminarayanan, and Mathini Sellathurai. FPGA based decision making engine for cognitive radio using genetic algorithm. In Proc. IEEE International Conference on Industrial and Information Systems (ICIIS), pages 633–636, Dec 2013.

[115] A. Katidiotis, K. Tsagkaris, and P. Demestichas. Performance evaluation of artificial neural network-based learning schemes for cognitive radio systems. Computers and Electrical Engineering , 36(3):518 – 535, 2010.

[116] Lili Cao and Haitao Zheng. Distributed spectrum allocation via local bargaining. In Proc. Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON) , pages 475–486, 2005.

[117] Haitao Zheng and Chunyi Peng. Collaboration and fairness in opportunistic spectrum access. In Proc. IEEE International Conference on Communications (ICC), volume 5, pages 3132–3136, 2005.

[118] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. American Mathematical Monthly, 69(1):9–15, Jan 1962.

[119] Fang Ye, Rui Yang, and Yibing Li. Genetic spectrum assignment model with constraints in cognitive radio networks. International Journal of Computer Network and Information Security (IJCNIS), 3(4):39, 2011.

[120] KiranKumar Anumandla, Shravan Kudikala, Bharadwaj Akella Venkata, and SamratL. Sabat. Spectrum Allocation in Cognitive Radio Networks Using Firefly Algorithm. In Proc. Swarm, Evolutionary, and Memetic Computing, volume 8297 of Lecture Notes in Computer Science, pages 366–376. Springer International Publishing, 2013.

[121] Quan Liu, Wenjuan Lu, and Wenjun Xu. Spectrum allocation optimization for Cognitive Radio networks using Binary Firefly Algorithm. In Proc. International Conference on Innovative Design and Manufacturing (ICIDM), pages 257–262, Aug 2014.

[122] J. Vesterstrom and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In Congress on Evolutionary Computation,, volume 2, pages 1980 – 1987 Vol.2, june 2004.

[123] R. Eberhart and J. Kenedy. Particle swarm optimization. In Proc. IEEE International Conference on Neural Networks, pages 1114–1121, Piscataway, NJ, November 1995.

[124] R. Eberhart and J. Kenedy. A new optimizer using particle swarm theory. In Proc. 6^{th} International Symposium on Micro Machine and Human Science (MHS), pages 39–43, Cape Cod, MA, November 1995.

[125] J.Robinson and Yahya Rahmath Samii. Particle swarm optimization in electromagnetic. IEEE Transactions on Antenna and Propagation, 52(2):397–400, 2004.

[126] Xin-She Yang. Firefly Algorithms for Multimodal Optimization. In Stochastic algorithms: foundations and applications, pages 169–178. Springer, 2009.

[127] Xin-She Yang. Nature-inspired metaheuristic algorithms. Luniver Press, 2008.

[128] Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, YP Chen, Anne Auger, and S Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Report, 2005005, 2005.

[129] Shih-An Li, Chen-Chien Hsu, Ching-Chang Wong, and Chia-Jun Yu. Hardware/software co-design for particle swarm optimization algorithm. Information Sciences, 181(20):4582 – 4596, Oct 2011.

[130] Pradeep R. Fernando, Srinivas Katkoori, Didier Keymeulen, Ricardo Zebulum, and Adrian Stoica. Customizable FPGA IP core Implementation of a General-Purpose Genetic Algorithm Engine. IEEE Transactions on Evolutionary Computation, 14(1):133–149, February 2010.

[131] Ke Tang, Xiaodong Li, Ponnuthurai Nagaratnam Suganthan, Zhenyu Yang, and Thomas Weise. Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization. Technical report, University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL): China, 2010.

[132] Amin Farmahini-Farahani, Shervin Vakili, Sied Mehdi Fakhraie, Saeed Safari, and Caro Lucas. Parallel scalable hardware implementation of asynchronous discrete particle swarm optimization. Engineering Applications of Artificial Intelligence, 23(2):177–187, Mar 2010.

[133] Girma S. Tewolde, Darrin M. Hanna, and Richard E. Haskell. A modular and efficient hardware architecture for particle swarm optimization algorithm. Microprocessors and Microsystems, 36(4):289 – 302, 2012.

[134] Rogrio M. Calazan, Nadia Nedjah, and Luiza M. Mourelle. A hardware accelerator for Particle Swarm Optimization. Applied Soft Computing, 14, Part C:347 – 356, 2014.

[135] Cheng-Jian Lin and Hung-Ming Tsai. FPGA implementation of a wavelet neural network with particle swarm optimization learning. Mathematical and Computer Modelling, 47(910):982 – 996, 2008.

[136] Mehmet Ali Cavuslu, Cihan Karakuzu, and Fuat Karakaya. Neural identification of dynamic systems on FPGA with improved PSO learning. Applied Soft Computing, 12(9):2707 – 2718, 2012.

[137] B. Vasumathi and S. Moorthi. Implementation of hybrid ANN - PSO algorithm on FPGA for harmonic estimation. Engineering Applications of Artificial Intelligence, 25(3):476 – 483, 2012.

[138] D.M. Mu andoz, C.H. Llanos, L. dos S Coelho, and M. Ayala-Rinco andn. Hardware Particle Swarm Optimization Based on the Attractive-Repulsive Scheme for Embedded Applications. In Proc. International Conference on Reconfigurable Computing and FPGAs, pages 55–60, Dec. 2010.

[139] D.M. Munoz, C.H. Llanos, L. dos S Coelho, and M. Ayala-Rincon. Hardware Architecture for Particle Swarm Optimization using Floating-point Arithmetic. In Proc. Ninth International Conference on Intelligent Systems Design and Applications, pages 243 –248, Dec 2009.

[140] D.M. Munoz, C.H. Llanos, L. dos S. Coelho, and M. Ayala-Rincon. Hardware Particle Swarm Optimization with Passive congregation for embedded applications. In Proc. VII Southern Conference on Programmable Logic (SPL), pages 173 –178, April 2011.

[141] KiranKumar Anumandla, Rangababu Peesapati, SamratL. Sabat, and SibaK. Udgata. SoC based floating point implementation of differential evolution algorithm using FPGA. Design Automation for Embedded Systems, 16(4):221–240, 2012.

[142] Xilinx. Reference Guide UG200 - Embedded Processor Block in Virtex-5 FPGAs. Technical Report 10.1.3, Xilinx, San Jose, California, 95124-3400, 2008.

[143] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182–197, April 2002.

[144] Jacqueline Moore and Richard Chapman. Application of particle swarm to multiobjective optimization. Department of Computer Science and Software Engineering, Auburn University, 1999.

[145] Ruhul Sarker and Hussein A. Abbass. Differential evolution for solving multi-objective optimization problems. Asia-Pacific Journal of Operational Research, 21(02):225–240, 2004.

[146] H.A. Abbass, R. Sarker, and C. Newton. PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems. In Proc. Congress on Evolutionary Computation, volume 2, pages 971–978, 2001.

[147] N.K. Madavan. Multiobjective optimization using a Pareto differential evolution approach. In Proc. Congress on Evolutionary Computation, volume 2, pages 1145–1150, 2002.

[148] F. Xue, A.C. Sanderson, and R.J. Graves. Pareto-based multi-objective differential evolution. In Proc. Congress on Evolutionary Computation, volume 2, pages 862–869, 2003.

[149] B. V. Babu and M.M.L. Jehan. Differential evolution for multi-objective optimization. In Proc. Congress on Evolutionary Computation, volume 4, pages 2696–2703, 2003.

[150] Tea Robič and Bogdan Filipič. DEMO: Differential evolution for multiobjective optimization. In Proc. Evolutionary Multi-Criterion Optimization, pages 520–533. Springer, 2005.

[151] K. Deb, A Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182–197, Apr 2002.

[152] Efrén Mezura-Montes, Margarita Reyes-Sierra, and Carlos A Coello Coello. Multi-objective optimization using differential evolution: a survey of the state-of-the-art. In Advances in differential evolution, pages 173–196. Springer, 2008.

[153] M Janga Reddy and D Nagesh Kumar. Multiobjective differential evolution with application to reservoir system optimization. Journal of Computing in Civil Engineering, 21(2):136–146, 2007.

[154] R. Devarajan, S.C. Jha, U. Phuyal, and V.K. Bhargava. Energy-Aware Resource Allocation for Cooperative Cellular Network Using Multi-Objective Optimization Approach. IEEE Transactions on Wireless Communications, 11(5):1797–1807, 2012.

[155] Sang-Seon Byun and I. Balasingham. Approximations of Multiobjective Optimization for Dynamic Spectrum Allocation in Wireless Sensor Networks. In Proc. International Conference on Consumer Electronics (ICCE), pages 427–428, 2010.

[156] Kai Wen, Lingsheng Fu, and Xuesong Li. Genetic Algorithm Based Spectrum Allocation for Cognitive Radio Networks. volume 121 of Lecture Notes in Electrical Engineering, pages 693–700. Springer Berlin Heidelberg, 2012.

[157] Jonathan Kok, Luis F. Gonzalez, Neil A. Kelson, and Jacques Periaux. An FPGA-based approach to multi-objective evolutionary algorithm for multidisciplinary design optimisation. In Proc. Evolutionary and Deterministic Methods for Design, Optimization and Control. CIMNE, October 2011.

[158] Tatsuhiro Tachibana, Yoshihiro Murata, Naoki Shibata, Keiichi Yasumoto, and Minoru Ito. A hardware implementation method of multi-objective genetic algorithms. In Proc. Congress on Evolutionary Computation, pages 3153–3160, 2006.

[159] R. Samano-Robles and A. Gameiro. Joint Spectrum Selection and Radio Resource Management based on Multi-Objective Portfolio Optimization for Cognitive Radio Networks. In Proc. International Conference on Future Generation Communication Technology (FGCT), pages 22–27, 2012.

[160] Yao Wang, Zhongzhao Zhang, Feng Li, and Jiamei Chen. A Novel Spectrum Allocation Algorithm for Cognitive Radio Networks. Procedia Engineering , 29(0):2776 – 2780, 2012.

[161] M. Haddad, A Hayar, and G.E. Øien. Downlink distributed binary power allocation for cognitive radio networks. In Proc. IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, pages 1–5, Sept 2008.

[162] N. Devroye, P. Mitran, and Vahid Tarokh. Achievable rates in cognitive radio channels. IEEE Transactions on Information Theory , 52(5):1813–1827, May 2006.

[163] A. Jovicic and P. Viswanath. Cognitive Radio: An Information-Theoretic Perspective. IEEE Transactions on Information Theory, 55(9):3945–3958, Sept 2009.

[164] Anh Tuan Hoang, Ying-Chang Liang, and M.H. Islam. Power Control and Channel Allocation in Cognitive Radio Networks with Primary Users' Cooperation. IEEE Transactions on Mobile Computing, 9(3):348–360, March 2010.

[165] L.H. Ozarow, S. Shamai, and A.D. Wyner. Information theoretic considerations for cellular mobile radio. IEEE Transactions on Vehicular Technology, 43(2):359–378, May 1994.

[166] Urban Transmission Loss Models for Mobile Radio in the 900 - 1,800 MHz Bands. COST 231: European Cooperation in the Field of Scientific and Technical Research. 1991.

[167] Floating Point Operator v5.0. Technical report, Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124-3400, March 2011.

Studies on Development of DE based Spectrum Allocation algorithms & FPGA implementation for Cognitive Radio Networks

ORIGINALITY REPORT

23%	11%	15%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Submitted to University of North Texas Student Paper	5%
Submitted to Amity University Student Paper	5%
www.softcomputing.net Internet Source	4%
Anumandla, Kiran Kumar, Rangababu Peesapati, Samrat L. Sabat, and Siba K. Udgata. "SoC based floating point implementation of differential evolution algorithm using FPGA", Design Automation for Embedded Systems, 2012. Publication	3%
Tragos, Elias Z., Sherali Zeadally, Alexandros G. Fragkiadakis, and Vasilios A. Siris. "Spectrum Assignment in Cognitive Radio Networks: A Comprehensive Survey", IEEE Communications Surveys & Tutorials, 2013. Publication	<1%
Abraham, Ajith, Rangababu Peesapati, Kiran Kumar Anumandla, Siba K. Udgata, and Samrat L. Sabat. "Field programmable gate	<1%