FPGA Implementation of Multi-standard Wireless Transceiver via Dynamic Partial Reconfiguration

Student Name: Gyan Deep

IIIT-D-MTech-ECE
December 15, 2017

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee

Dr. Sumit Jagdish Darak (Advisor)
Dr. Sneh Saurabh (Internal Reviewer)
Dr. Vivek Bohara (External Reviewer, IIIT-Delhi)

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Electronics & Communication with specialization in
Communication & Signal Processing

Keywords: Partial Reconfiguration, 802.11a, FPGA, Transmitter and Receiver.

Certificate

This is to certify that the thesis titled "**FPGA Implementation of Multi-standard Wireless Transceiver via Dynamic Partial Reconfiguration**" submitted by **Gyan Deep** for the partial fulfillment of the requirements for the degree of *Master of Technology in Electronics and Communication Engineering with specialization in Communication and Signal Processing* is a record of the bonafide work carried out by her under my guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

December, 2017

Dr. Sumit J Darak
Assistant Professor
Department of Electronics and Communication
Indraprastha Institute of Information Technology Delhi
New Delhi, 110020

Abstract

To support wide variety of services ranging from voice, high speed data and multimedia, multi-standard wireless communication transceivers (MWCT) are desired. Such transceivers have the capability to adapt to any desired data rate, bandwidth and center frequency depending on the environmental conditions (for example, wireless channel, distance between transmitter and receiver, multipath fading, jammers etc.). Conventional MWCT employs Velcro approach where multiple communication standards are supported using parallel signal processing chains, one for each standard. This technique is straightforward and fast but there are also some major drawbacks associated with it like more area requirement and overall high power consumption and hence, not suitable for battery operated resource constrained wireless transceivers. Due to the re configurable architecture of Field Programmable Gate Array(FPGA), it has become an attractive platform to implement the wireless testbeds. Also, due to its inherent parallel architecture support, FPGA is ideal for implementing parallel operations like FFT/IFFT computation and digital filter implementation as required in these wireless transceivers.

Excess area and power utilization of velcro method can be reduced by using the dynamic partial reconfiguration technique. In this approach, only the blocks like qpsk modulation/demodulation blocks which need to be changed are swapped with the required blocks like 16-qam modulation/demodulation blocks according to the requirements. This approach leads to reduction of area utilization from sum of all the area required for implementing different versions of a particular block(i.e. parallel implementation) to area which is required by biggest version (in terms of area) of a particular block. Also,in this approach, the rest of design can work uninterruptedly while the reconfiguration is taking place.

The main aim of this thesis is to demonstrate a working hardware testbed which implements a transciver supporting dynamic partial reconfiguration technique. The proposed MWCT can adapt the modulation scheme as well as the transmission bandwidth by partial reconfiguration of modulation and IFFT/FFT blocks of the transceiver. We discuss software controlled as well as hardware controlled approaches by which the partial reconfiguration can be enabled. We demonstrate the proposed approach on the 802.11a transceiver protocol realized on Zynq SoC using Verilog. At the end, we compare the gain in area and power consumption over conventional Velcro approaches. Also, filters are implemented at the end of transmitter and beginning of receiver to demonstrate the implementation of filtered OFDM.

Acknowledgments

It gives me immense pleasure to express my heartily gratitude to everyone who supported and guided me in the completion of my thesis.

Foremost, I would like to express my sincere gratitude to my advisor Dr. Sumit J Darak. Without his excellent guidance, encouragement and support, I would never be able to finish my thesis work. He has been a great source of inspiration and I feel extremely fortunate to work with him.

I would like to thank Dr. Pham Hung Thinh of Nanyang Technological University, Singapore, whose simulated verilog implementation was used in this thesis and whose guidance and feedback regarding the implementation was very helpful for this thesis.

I would like to thank Shanon lab technical staff, Mr. Khagendra Joshi and Mr. Rahul Gupta for providing me quick access to all instruments whenever I needed them.

I would like to thank my friend Miss Sasha Garg for providing me help related to filter designing and filtered OFDM implementation.

I would like to acknowledge my parents and friends for encouraging and supporting me. They have been a source of moral support to me and have extended their helping hands without fail.

Contents

List of Figures

List of Tables

List of abbrevation

ADC	Analog to Digital Converter
AFE	Analog Front End
BPSK	Binary Phase Shift Keying
CRs	Cognitive Radios
DAC	Digital to Analog Converter
DFE	Digital Front End
DPR	Dynamic Partial Reconfiguration
FPGA	Field Programmable Gate Array
FFT	Fast Fourier Transform
ICAP	Internal Configuration Access Port
IFFT	Inverse Fast Fourier Transform
IF	Intermediate Frequency
ILA	Integrated Logic Analyzer
LUT	Look Up Table
MMCM	Mixed Mode Clock Manager
MWC	Modulated Wideband Converter
MWCRs	Multi-Standard Wireless Communication Receivers
PCAP	Processor Configuration Access Port
PRC	Partial Reconfiguration Controller
PLL	Phase Locked Loop
QAM	Quadrature Amplitude Modulation
OFDM	Orthogonal Frequency Division Multiplexing
PL	Programmable Logic
PS	Processing System
PR	Partial Reconfiguration
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
SDR	Software Defined Radio
SPR	Static Partial Reconfiguration
SNR	Signal to Noise Ratio
USRP	Universal Software Radio Peripheral

Chapter 1

Introduction

1.1 Motivation

Software defined radios (SDRs) and cognitive radios (CRs) build an intelligent wireless communication system. Evolution of SDRs and CRs allow mobile devices to reconfigure their transmission parameters such as frequency band, modulation type, symbol rate, IFFT/FFT size etc. to meet the desired quality of service for any given transmission requirements and channel conditions [1, 2]. Hence, mobile devices need multi-standard wireless communication receivers (MWCRs) to receive signals of different communication standards.

Due to re-configurable architecture of FPGA and inherent parallel architecture support, it has become an ideal platform for implementing wireless testbeds. The traditional velcro approach of implementing several communication standards through exclusively dedicated self contained communication components of a given standards is simple and fast to prototype[**?**]. This approach has inherent disadvantage of high resource utilization(resulting in the use of expensive FPGA or FPGAs in parallel), non-scalability and high power consumption[3]. The idea of run time reconfiguration has been there for decades. Earlier, the idea was to load only currently used instructions(utilizing less resources) instead of configuring all instructions in one go. Here implementing such systems required deep knowledge of FPGA architecture and was error prone. Currently, we are dealing with over million look-up-table (LUT) in FPGA. The configuration time required to write several megabytes of bitstreams for initial configuration is too high. Also, if the design is modified slightly, the reconfiguration of whole design is impractical. In order to improve the resource utilization of FPGA slices as well as system flexibility, the use of partial reconfiguration is becoming popular[4]. Here, the a portion of FPGA area is allocated to each changing blocks. When dynamic partial re-configuring is done, only the LUTs dedicated to that area on FPGA are modified by loading the partial bitstreams. Also, in this approach the rest of the design can continue to operate uninterruptedly. A simplified architecture of OFDM Transmitter and Receiver, shown in Fig. 1.1.Transmitter consists of mainly three blocks: 1) Modulation block, 2) IFFT block, and 3) DAC with RF end. Similarly, the main blocks of receiver are 1) RF and ADC, 2) FFT block and 3) Demodulation block. The hardware implementation

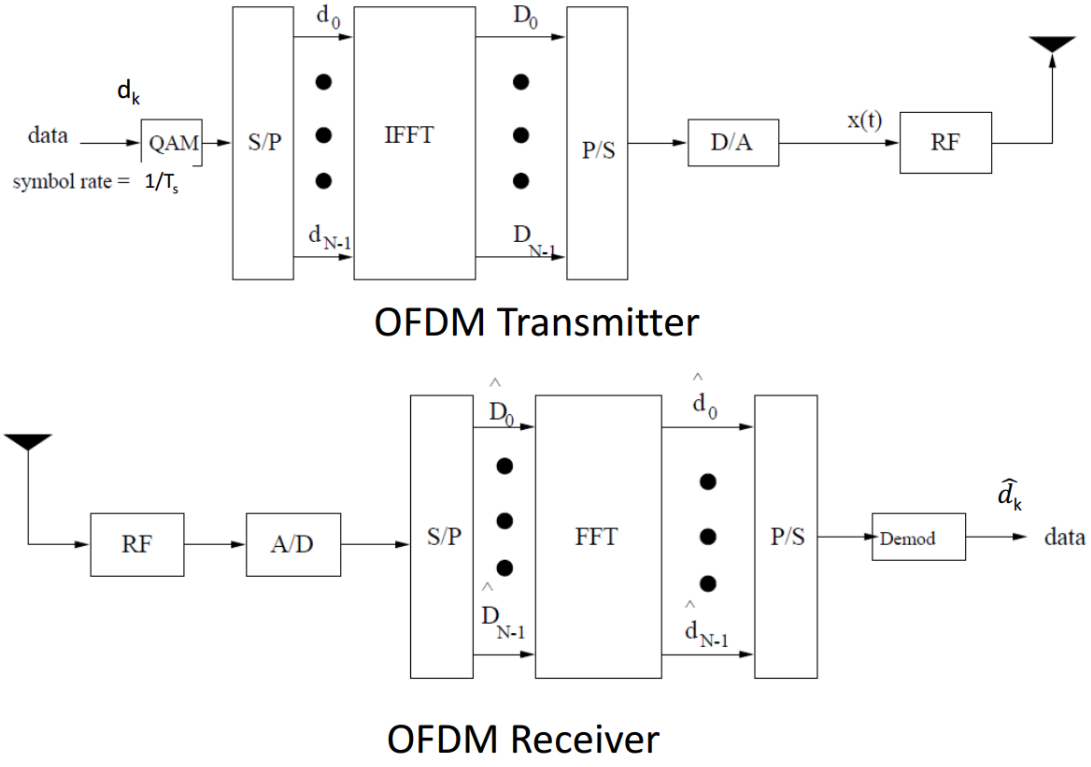

Figure 1.1: Block diagram of OFDM transmitter and receiver

of the basic OFDM transmitter and receiver which can support partial reconfiguration is the focus of the work presented in this thesis. This design will act as a base design which is extended to support filtering operation as required in filtered OFDM.

1.2 Objectives

The objectives of the work presented in thesis are :

1. To implement a basic OFDM transmitter and receiver which can support partial reconfiguration of it's block like that of modulation/demodulation block.
2. To extend this design to support filtered OFDM.

1.3 Organization

The thesis is organized as follows. Chapter 2 presents the detailed introduction of Zynq architecture. In Chapter 3, an introduction of dynamic partial reconfiguration is presented and various techniques to implement it is discussed. Chapter 4 presents the blocks of OFDM transceiver implemented in the design. In Chapter 5, a brief introduction of Filtered OFDM is presented. Finally, Chapter 6 presents different wave-forms that are generated by the test-bench and ILA output to support the validity of the design. Also, it briefly discusses the possible future works.

Chapter 2

Zynq Architecture

The Zynq 7000 family is based on the Xilinx All Programmable SoC architecture. It is the integration of dual/single core ARM Cortx A9 based processing system(PS) and Xilinx programmable logic(PL) on a single die. This family offers flexibility and scalability of FPGA, while providing performance, power, and ease of use typically associated with ASIC and ASSPs. It enables implementation of custom logic in the PL and custom software in the PS. The level of performance offered by integration of PS and PL is above than that of two-chip solutions(e.g. an ASSP with an FPGA) due to their limited I/O bandwidth, latency and power budgets. PS processors always boot first, allowing a software centric approach for PL configuration[6]. The Zynq architecture is shown in Fig. 2.1[6].

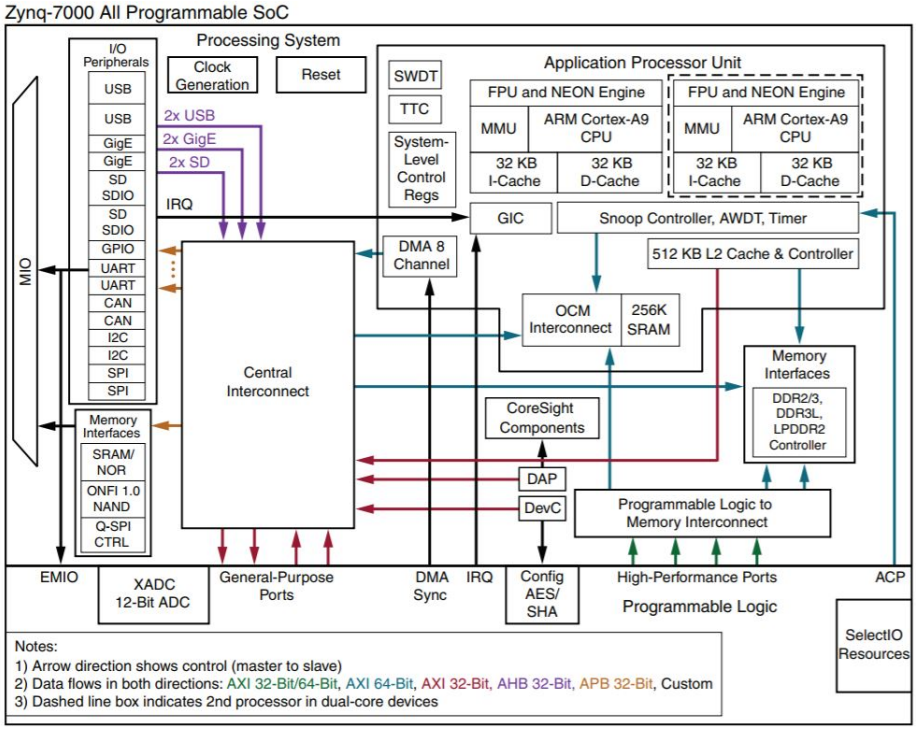

Figure 2.1: Zynq Architecture

2.1 Processing System

As shown in the Fig. 2.1, the PS consist of four major blocks:

1. Application Processor Unit
2. Memory Interfaces
3. I/O Peripherals
4. Interconnect

2.1.1 Application Processor Unit

APU mainly include:

1. Dual-core/single-core ARM Cortex-A9 MPCores
2. Accelerator Coherency Port for coherent accesses from PL to CPU memory space
3. DMA
4. Interrupts and Timers
5. CoreSight debug and trace support for Cortex-A9

2.1.2 Memory Interfaces

It includes a dynaic memory controller and static memory interface modules. The dynamic memory controller supports DDR3, DDR3L, DDR2 and LPDDR2 memories. The static memory controllers support a NAND flash interface, a Quad-SPI flash interface, a parallel data bus, and a parallel NOR flash interface.

2.1.3 I/O Peripherals

It communicate with external devices through a shared pool of up to 54 dedicated multiuse I/O pins. Each peripheral can be assigned one of several pre-defined groups of pins, enabling a flexible assignment of multiple devices simultaneously. Key feature of IOP includes:

1. Two 10/100/1000 tri-mode Ethernet MAC peripherals with IEEE Std 802.3 and IEEE Std revision 2.0 support
2. Two USB 2.0 OTG peripherals
3. DMA
4. Two UARTs

5. Two master and slave I2C interfaces
6. Up to 118 GPIO bits
7. Two full-duplex SPI ports

2.1.4 Interconnect

The memory interface unit, APU and I/O Peripherals are all connected to each other and to the PL through a multilayered ARM AMBA AXI interconnect. The interconnect is non-blocking and supports multiple simultaneous master-slave transactions. The interconnect is designed with latency sensitive masters, such as the ARM CPU, having the shortest paths to memory, and bandwidth critical masters, such as the potential PL masters, having high throughput connections to the slaves with which they need to communicate. Traffic through the interconnect can be regulated through the Quality of Service block in the interconnect.

2.2 Programmable Logic(PL)

PL mainly includes:

1. CLB
2. 36 Kb block RAM
3. DSP slices to perform 18 x 25 signed multiply and 48-bit adder/accumulator.
4. Programmable I/O Blocks.
5. PL configuration module.
6. Two 12-bit analog to digital converters.
7. low power serial transceivers and integrated Endpoint/Root port block for PCI Express in selected devices.

2.2.1 CLBs,Slices, and LUT

The CLB architecture in Zynq consist of a True 6-input LUTs, Memory capability within the LUT and Register and shift register functionality. Here the LUT can be used as single Output with 6 input or as separate output with two 5 input. Also the output of LUT can be registered as flip-flop. Between 25-50 percentage of all slices can be used as distributed 64-bit RAM or as 32-bit shift registers or as two 16-bit shift Resisters.

2.3 I/O Peripherals

Zynq device consist of up to 8 clock management tiles(CMTs),each consisting of one mixed-mode clock manager(MMCM) and one phase-locked loop(PLL). Also, they have support for high-speed buffers and routing for low-skew clock distribution, Low jitter generation and jitter filtering,Frequency synthesis and phase shifting.

2.3.1 Mixed Mode Clock Manager and Phase Locked Loop

Both can be used as frequency synthesizer and jitter filtering for incoming clocks. There are three sets of programmable frequency dividers: D, M,and o. Also they both have 3 input-jitter filter options: Low-bandwidth mode (best jitter attenuator), high-bandwidth mode (best phase offset) and opimized mode.

2.3.2 Clock Distribution

There are six different types of clock lines to address different clocking requirements of high fanout, short propagation delay, and extremely low skew. Clock lines are BUFG, BUFR, BUFIO, BUFH, BUFMR, and the high performance clock.

Global Clock Lines

There are 32 global clock lines which can reach every flip-flop clock, clock enable and set/reset. BUFH drives 12 global clock lines within any clock region. The BUFH can be independently enabled/disabled, allowing for clocks to be turned off within a region. Global clock lines can be driven by global clock buffers,which can also perform glitchless clock multiplexing and clock enable functions.

Regional Clocks

It can drive all clock destinations in their regions. Region is an area of 50 I/O and 50 CLB high and half the device wide. There are between four to fourteen regions.Each regional clock buffer can be driven from either of four clock-capable input pins and its frequency can be divided by any integer from 1 to 8.

I/O Clocks

I/O clocks are especially fast and serve only I/O logic and serializer/deserializer circuits. There is a direct connection from MMCM to the I/O for low-jitter, high-performance interfaces.

2.4 Block RAM

The Zynq-7000 family has up to 755 dual-port block RAMs,each storing 36 Kb with port widths of up to 72 and their are 2 completely independent output ports.

2.4.1 Synchronous Operation

All inputs, data, address, clock enables and write enables are registered. The input address is always clocked, retaining data until the next operation. An optional output data pipeline register allows higher clock rates at the cost of an extra cycle of latency. During a write operation, the data output can reflect either the previously stored data, the newly written data, or can remain unchanged.

2.4.2 Programmable Data Width and Error detection and Correction

Each port can be configured as 32K 1, 16K 2, 8K 4, 4K 9 (or 8), 2K 18 (or 16), 1K 36 (or 32), or 512 72 (or 64).The two ports can have different aspect ratios without any constraints. Each block RAM can be divided into two completely independent 18 Kb block RAMs that can each be configured to any aspect ratio from 16K 1 to 512 36. Everything described previously for the full 36 Kb block RAM also applies to each of the smaller 18 Kb block RAMs. Only in simple dual-port (SDP) mode can data widths of greater than 18 bits (18 Kb RAM) or 36 bits (36 Kb RAM) be accessed. In this mode, one port is dedicated to read operation, the other to write operation. In SDP mode, one side (read or write) can be variable, while the other is fixed to 32/36 or 64/72. Both sides of the dual-port 36 Kb RAM can be of variable width.

Each 64-bit-wide block RAM can generate, store, and utilize eight additional Hamming code bits and perform single-bit error correction and double-bit error detection (ECC) during the read process. The ECC logic can also be used when writing to or reading from external 64- to 72-bit-wide memories.

2.5 DSP Slice

It is dedicated, full custom, low power slices. Each consists of a dedicated 25 x 18 bit two's complement multiplier and a 48 bit accumulator. The multiplier can be dynamically bypassed, and two 48-bit inputs can feed a single-instruction-multiple-data (SIMD) arithmetic unit (dual 24-bit add/subtract/accumulate or quad 12-bit add/subtract/accumulate), or a logic unit that can generate any one of ten different logic functions of the two operands. The DSP includes an additional pre-adder, typically used in symmetrical filters. The DSP also includes a 48-bit-wide Pattern Detector that can be used for convergent or symmetric rounding. The pattern detector is also capable of implementing 96-bit-wide logic functions when used in conjunction with the logic unit.

Chapter 3

Dynamic Partial Reconfiguration

Partial Reconfiguration is the ability to reconfigure select areas of an FPGA any time after its initial configuration[8]. From the functionality point of view there are two types:-

3.0.1 Dynamic Partial Reconfiguration

It is also called active reconfiguration which permits to change the part of device while the rest part are still running as shown in the Fig. 3.1. It is carried out to allow the FPGA to adapt to changing hardware algorithms, improve fault tolerance and resource utilization, to enhance performance or to reduce power consumption. DPR is valuable where devices operate in a mission critical environment that cannot be disrupted while some subsystems are being redefined.

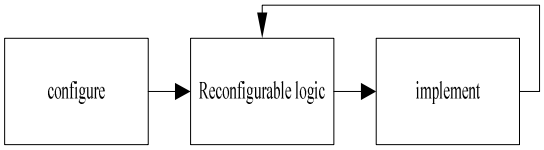

Figure 3.1: Dynamic Partial Reconfiguration

3.0.2 Static Partial Reconfiguration

In Static Reconfiguration the device is not active during the reconfiguration process. When the partial data is sent into the FPGA, the rest of the device is stopped and brought up after the configuration is done, as shown in the Fig. 3.2.

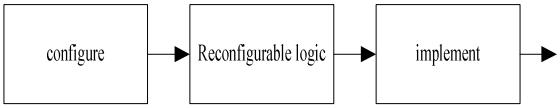

Figure 3.2: Static Partial Reconfiguration

3.0.3 Why Partial Reconfiguration?

The partial reconfiguration is useful in these situations:-

1. Reducing the cost of FPGA device, therefore, reducing the cost and power consumption, as shown in the Fig. 3.3.
2. Providing the flexibility in the choices of algorithms or protocol available to an application.
3. Accelerating configurable computing.
4. Enabling new technique in design security[7].
5. Adaptive systems.
6. Fault tolerant/self-repairing systems.

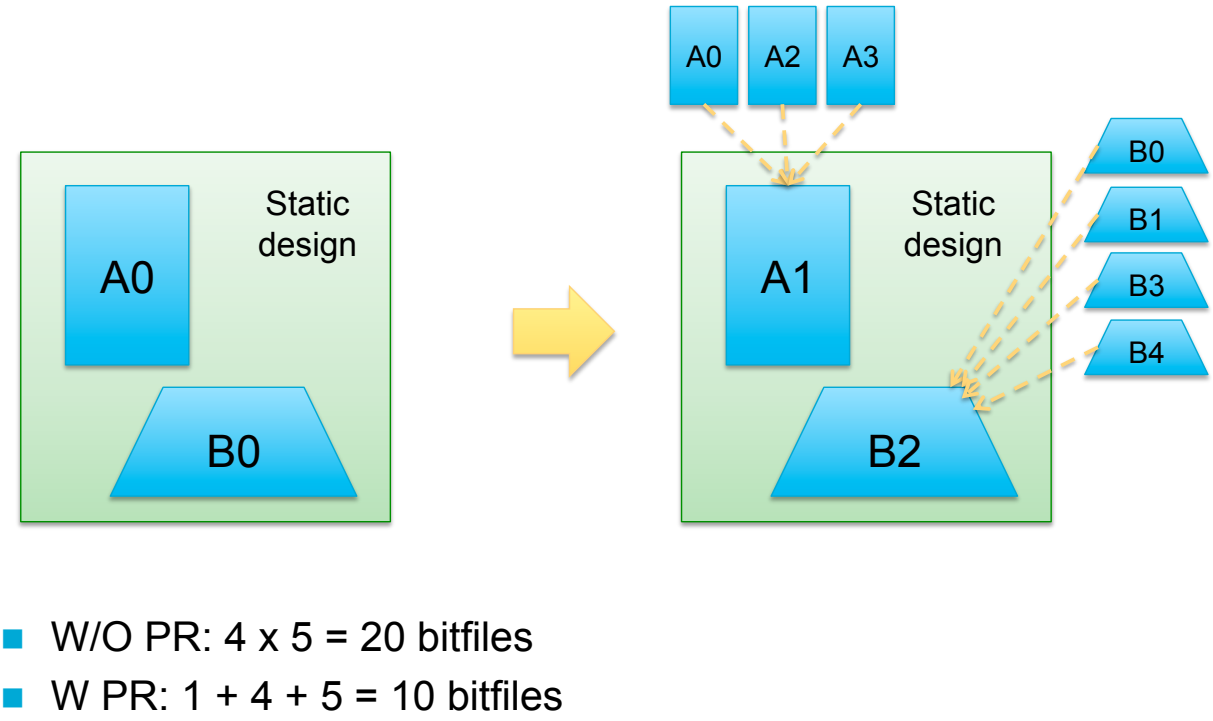

Figure 3.3: Partial Reconfiguration

In Fig. 3.3, the design consists of two partial reconfigurable region i.e. A and B. Region A can have 4 different logics and region B can have 5 different logics. If we have to implement this design without PR then we will need 20 bitfiles which will include both static and reconfigurable parts individually. The PR design require 1 full design bitfile and 4 different bitfiles corresponding to A region and 5 different bitfiles corresponding to B region. Note that the memory requirement is low as number of bitfiles to store is low. Also out of ten only nine are partial bitfiles.

3.1 Different Methods to Implement Partial Reconfiguration in Zynq SoC

1. Reconfiguring using AXI HWICAP IP - Here AXI HWICAP IP is used which in backend uses ICAP as shown in the Fig. 3.4. In software the PCAP is disabled as ICAP is used to reconfigure. The partial reconfiguration is done under software control.

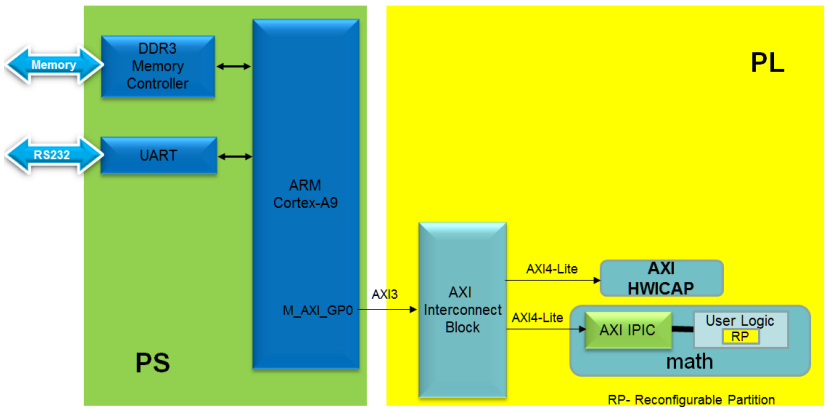

Figure 3.4: Hardware ICAP

2. Reconfiguring using Custom ICAP Processor - The ICAP is accessed through a light-weight custom IP. This custom IP processor requires bitstream length, go and done signals as input as shown in the Fig. 3.5. The partial bitstream is provided by the processor system by reading the partial bitfiles from the SD card,storing them in the DDT memry, and sending the appropriate bitstream to the ICAP processor based on the user's selection.

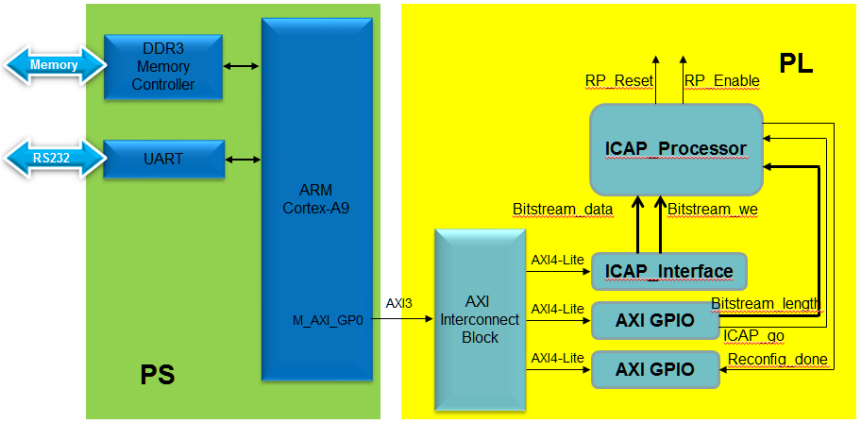

Figure 3.5: Internal Configuration Access Port

3. Using Partial Reconfiguration Controller(PRC) core to reconfigure when a hardware event occurs. Hardware event can be like pressing on-board pushbuttons.

4. Reconfiguring with HW-SW triggers using PRC - The dynamic partial reconfigurable modules are updated either through menu using the PRC's software triggers capability or

through push-buttons using the hardware triggers as shown in the Fig. 3.6.

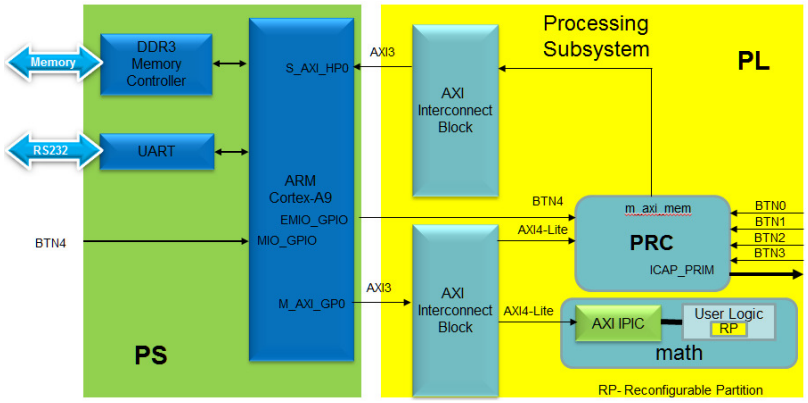

Figure 3.6: Partial Reconfiguration Controller

5. Reconfiguring using PCAP resource and PS sub-system - Here the user can verify the functionality of reconfiguration block using a user application. The dynamic modules are reconfigured using the PCAP resource available through Device Configuration block.

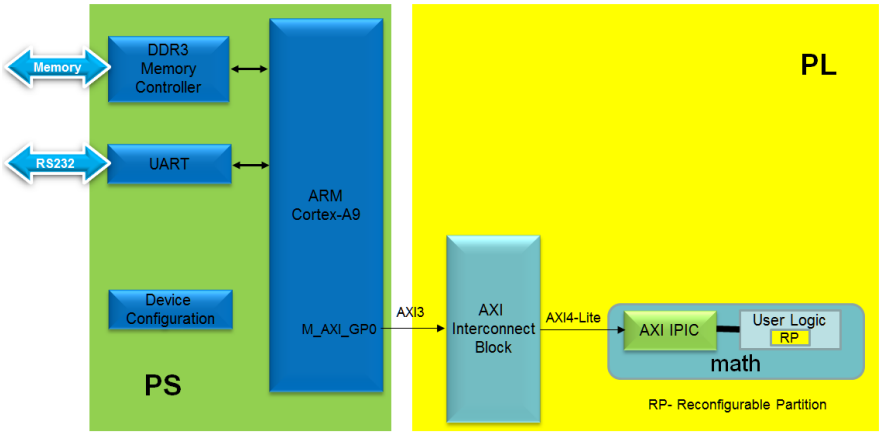

Figure 3.7: Processor Configuration Access Port

3.2 Partial Reconfiguration Software Flow

Implementing partial reconfigurable design is similar to implementing multiple non-PR designs that share the common logic[7]. Partitions are used to ensure that the common logic between the multiple designs is identical as shown in Fig. 3.8.

The top gray box represents the synthesis of HDL source to netlists for each module. The appropriate netlists are implemented in each design to generate the full and partial BIT files for that configuration. The static logic from the first implementation is shared among all subsequent design implementations.

The above software flow can be broken down into 8 basic steps in most of the partial reconfiguration design scenarios. First step consist of generating Design Checkpoints(DCP)from the HDL

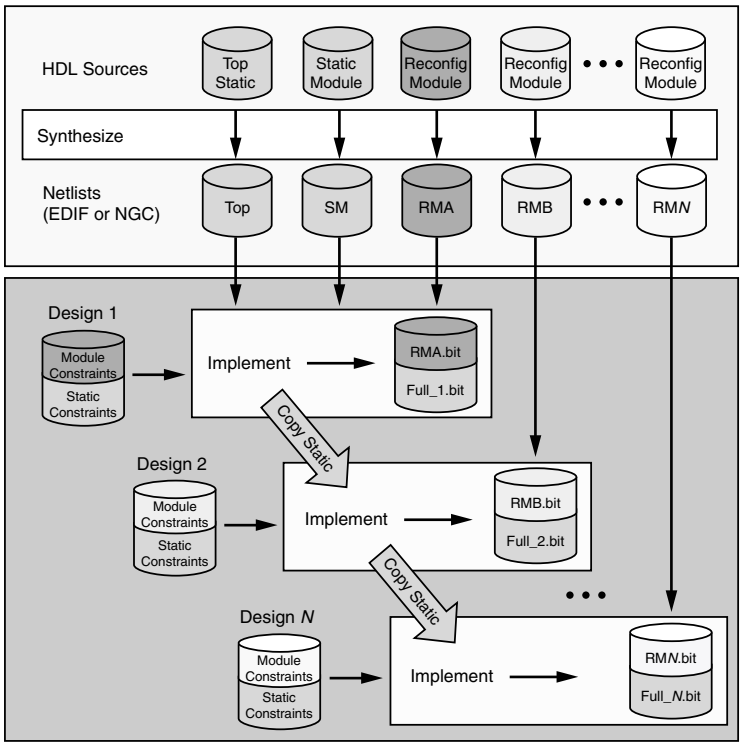

Figure 3.8: Partial Reconfiguration Software Flow

source consisting of Static part and Reconfiguration module parts.It is done after the synthesis of the HDL.Then the Static part is loaded and one reconfiguration module is loaded. A manual floorplan is done by the user to assign the total slices and RAM to the reconfigurable module. Then this whole design consists of static and one reconfigurable module is implemented. Similarly, the reconfigurable module is replaced by other reconfigurable module and again the whole design is implemented. This process is repeated for as much reconfigurable module we have. After generating all the full designs consisting of same static module and different reconfigurable modules, a partial reconfiguration verification is done which simply compares the compatibility of all the full designs. Finally Bit files are generated for each full designs and each partial reconfiguration modules. A software application may be generated if the partial reconfiguration is done through software trigger or some logic is required to be run on the processing system side. Any one full design is loaded which act as the starting state or algorithm for the PL. Then subsequently other PR logic replaces the PR module depending on the subsequent triggers. All the 8 steps are shown in the Fig. 3.10

Some general guideline to follow at this process are:-

1. Decoupling logic is highly recommended to disconnect the reconfigurable region from the static portion of the design during the act of Partial Reconfiguration.

2. A local reset must be issued to reconfigured logic to ensure a known good starting state, since the Global Set/Reset (GSR) command is not issued after a Partial Reconfiguration.

3. A PR design must consider the initiation of Partial Reconfiguration as well as the delivery

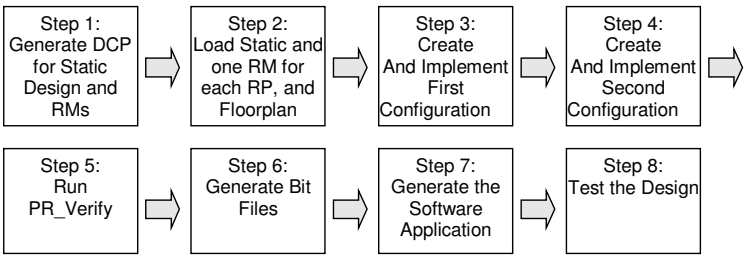

Figure 3.9: General Partial Reconfiguration steps

of partial BIT files, either within the FPGA or as part of the system design

4. A reconfigurable partition must contain a super set of all pins to be used by the varying reconfigurable modules implemented for the partition. It is expected that this will lead to unused inputs or outputs for some module variants, and is designed into the flexibility of the PR solution. The unused inputs will be left dangling inside of the module and will cause the implementation tools to issue messages that you may ignore. In the case of a black box RM (no logic) all partition pin outputs will be driven by a constant Logic 1.

5. Routing challenges may occur if the reconfigurable region is too small or is constructed of non-rectangular shapes.

6. Clocks and Clock Modifying Logic must reside in the static region which includes BUFG, BUFR, MMCM, PLL, DCM.

7. The following components must reside in the static region:

   (a) I/O and I/O related components

   (b) MGT (Multi-Gigabit Transceivers) and related components

   (c) Individual architecture feature components (such as BSCAN, STARTUP, etc.)

8. No bidirectional interfaces are permitted between static and reconfigurable regions, except in the case where there is a dedicated route.

9. IP restrictions may occur due to components used to implement the IP. Examples include:

   (a) ChipScope ICON (BUFG)

   (b) EDK blocks with global buffers

   (c) MIG controller (MMCM)

10. Validation of the integrity of partial BIT files can be checked using an IP core inserted as part of a BIT file delivery mechanism.

3.3 Challenges with Partial Reconfiguration

It requires a detailed knowledge of the FPGA architecture to decide how to partition the FPGA between the static, always-on, parts and reconfigurable regions.First of all, a run-time reconfigurable system must improve a static only solution. This requires that the logic overhead for the communication with the reconfigurable modules is kept low. The designer must also determine which modules should be assigned to which regions and hence the granularity of reconfiguration. For systems incorporating partial reconfiguration, design decisions can impact the cost of using PR. The number of regions used, and the allocation of reconfigurable modules to regions can impact resource requirements considerably. Hence for greater savings, this allocation should be done intelligently [9].

The primary cost associated with partial reconfiguration is reconfiguration time. In a circuit in which all functionality is present on chip, selecting between modes typically takes just a few clock cycles. When using PR, the system must pause, and the PR region must be reconfigured, which takes time. Smaller regions result in shorter reconfiguration times, but can impact area efficiency [9].

Besides the efficiency of the reconfigurable system, it is similarly important to have an efficient way to implement reconfigurable systems. This demands tools and design flows that are able to implement efficient design methods for reconfigurable systems.

Recently, the design flows are based on an incremental design methodology where the static system (the part of the system that will be present at any time, e.g, a CPU or memory controller) will be implemented in a first step. In order to implement the communication to and from the reconfigurable modules, an anchor LUT (called proxy logic by Xilinx) will be added into the reconfigurable regions and connected, as shown in Fig. **??**. The partial modules are then implemented as an increment to the static systems (where the placement and routing will be preserved)[10].

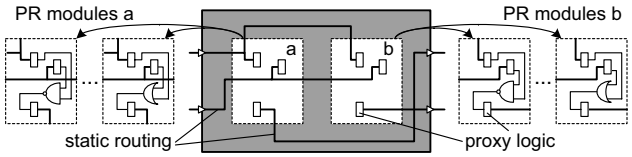

Figure 3.10: Partial module integration using proxy logic[10]

This is nicely supported by the tools but has three major restrictions. 1) As the routing to the anchor LUTs is not strictly constrained, it is in general different for each reconfigurable area, as sketched in Figure 1. Furthermore, there might be static signals crossing a reconfigurable area. This prevents module relocation among different reconfigurable areas, even if the different reconfigurable modules would be identical for both areas. 2) The routing to the anchor LUT will in general change each time the static system is changed. Consequently, all permutations of module instance and placement position have to be reimplemented on each change of the static system. Finally, 3) a reconfigurable area can only host one module exclusively (island

style reconfiguration) and it is not supported to share a reconfigurable area by multiple modules in a flexible manner at the same time[10].

Chapter 4

OFDM and Filtered OFDM

Orthogonal Frequency Division Multiplexing(OFDM) is a dominant technology for broadband multicarrier communications[11]. The basic idea of multicarrier modulation is to divide the transmitted bitstream into many different substreams (primarily orthogonal substreams) and send these over many different subchannels. The number of subchannels are chosen such that each subchannel has a bandwidth less than the coherence bandwidth of the channel, so the subchannels experience relatively flat fading. Subchannels in the multicarrier modulation need not be contiguous, so a large continuous block of spectrum is not needed for high-rate multicarrier communication. The figure of conventional multicarrier receiver and transmitter is shown in Fig. 4.1

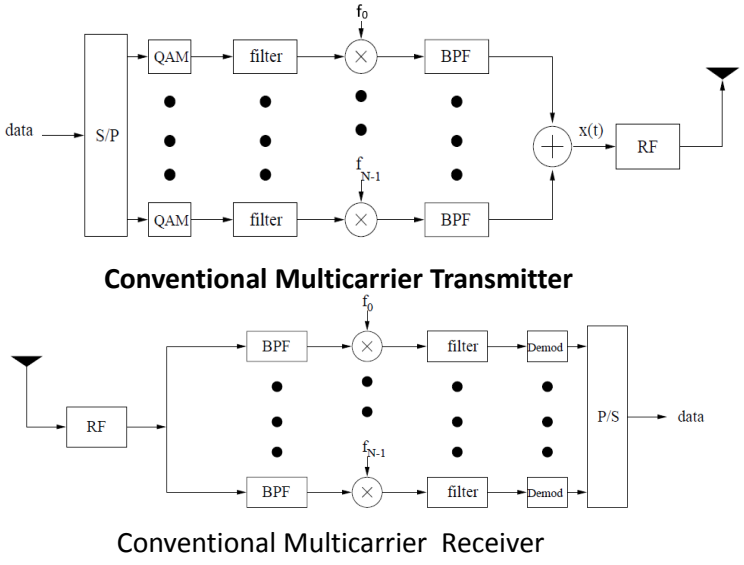

Figure 4.1: Multicarrier Transmitter and Receiver

Problem with conventional multicarrier systems are :-

1. Spectrally inefficient

2. Require a bank of filter with sharp cut-off.

So a need for multicarrier modulation scheme with overlapping subchannels is required.

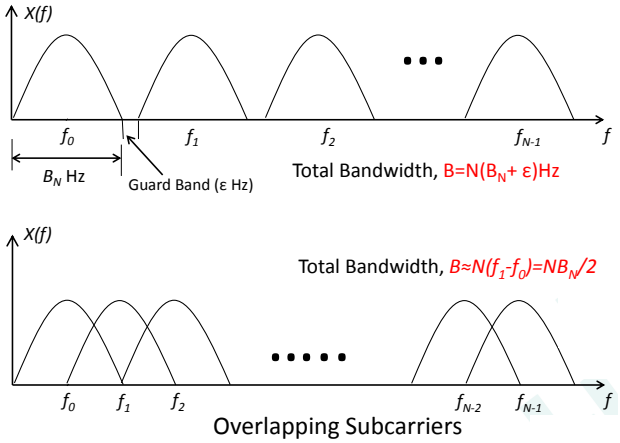

Figure 4.2: Spectral efficiency between overlapping and non overlapping multicarrier modulation

OFDM is a type of Frequency Division Multiplexing where the entire channel is divided into many narrow subchannels/subcarriers. These narrow subchannels are transmitted in parallel which in turn increases the symbol duration and reducing the ISI[12]. Here the sub carriers are overlapped as seen in Fig. 4.3. The sub-carriers occupy the spectral zero crossing positions of other sub-carriers and hence increasing the spectral efficiency. An OFDM modulated signal can be generated by performing Inverse Discrete Fourier Transform (IDFT) on the digitally modulated signal and correspondingly a DFT operation in the demodulator. In OFDM, the bandwidth W is equally divided among all the subcarriers.

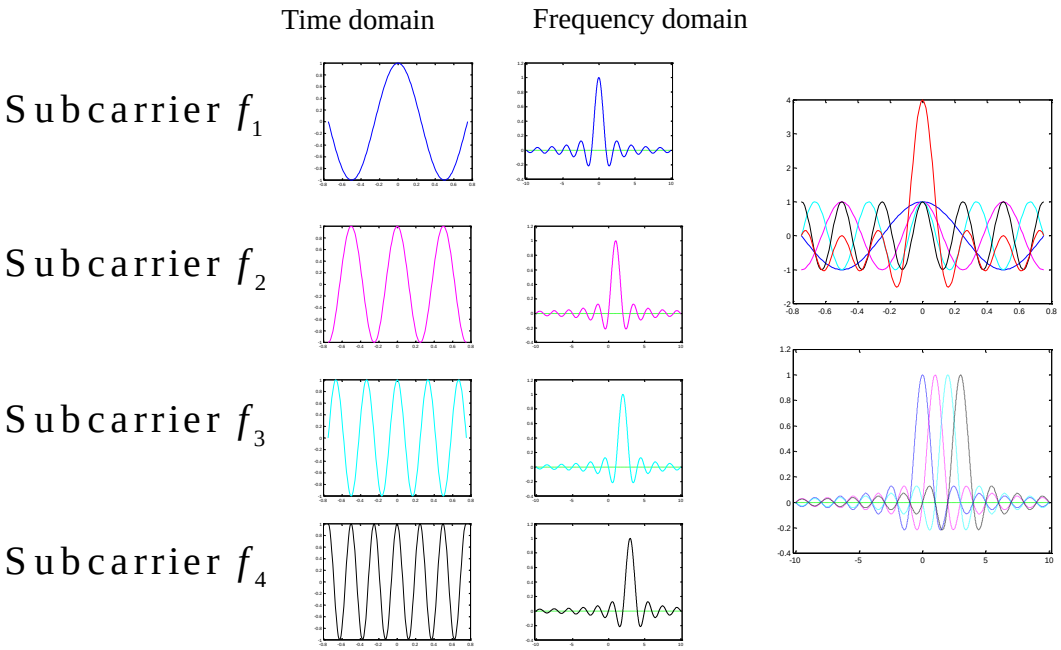

Figure 4.3: Time and frequency domain representation of OFDM symbol

In this thesis,the OFDM transceiver system is based on the IEEE 802.11a standard.

4.0.1 Transmitter

The OFDM based transmitter consists of blocks such as scrambler, convolutional encoder, interleaver, modulator (BPSK,QPSK,QAM16 etc.),null and pilot insertion, fast Fourier transform(FFT),cyclic prefix adder and long and short training sequence adder. The scrambler rearranges the input bits according to a pre-defined scrambling sequence followed by a convolutional encoder and an interleaver block.This is followed by modulation via BPSK with 1 and +1 as the constellation points. Similarly, other modulation schemes such as QPSK, QAM16 etc. can be used. Next steps include serial-to-parallel conversion, 64-point fast Fourier transform (FFT), parallel-to-serial conversion using OFDM modulation block. As per 802.11a specifications, 64 sub-carriers are used out of which 48 are data sub-carriers and the remaining 16 subcarriers consist of 4 pilot and 12 null sub-carriers. The null are mapped as shown in the Fig. 4.4.

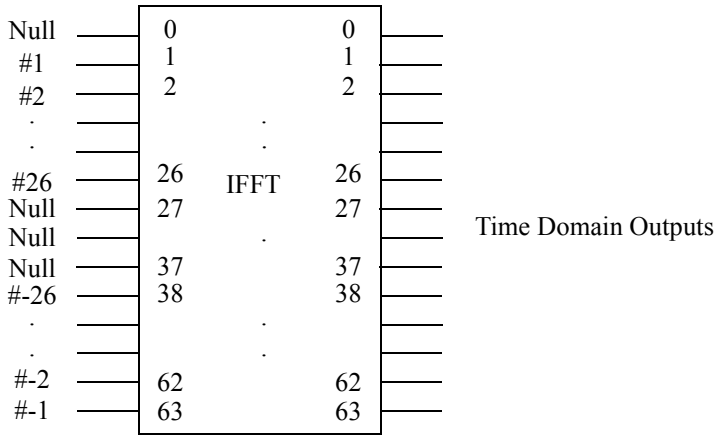

Figure 4.4: Null mapping in IFFT Block in 802.11a [15]

Cyclic prefix of length 16 is appended to the OFDM symbol to avoid inter-block interference. At the end, preamble is added for synchronization. The preamble consists of both short training sequence (STS) for course frequency acquisition, timing acquisition and diversity selection and long training sequence(LTS) for channel estimation and fne frequency acquisition [13] [14]. Long training sequence is repeated twice and short training sequence 10 times for given length of 160 samples. The full diagram for implemented transmitter is shown in the Fig. 4.5.

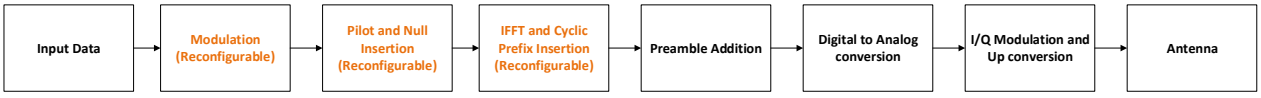

Figure 4.5: The block diagram of implemented 802.11a Transmitter in verilog

4.0.2 Receiver

The receiver is similar to transmitter with all the operations being performed in reverse order as shown in Fig. 4.6.

Figure 4.6: The block diagram of implemented 802.11a Receiver in verilog

The synchronization of receiver with transmitter is done by long and short preamble and there respective use is shown in Fig. 4.7.

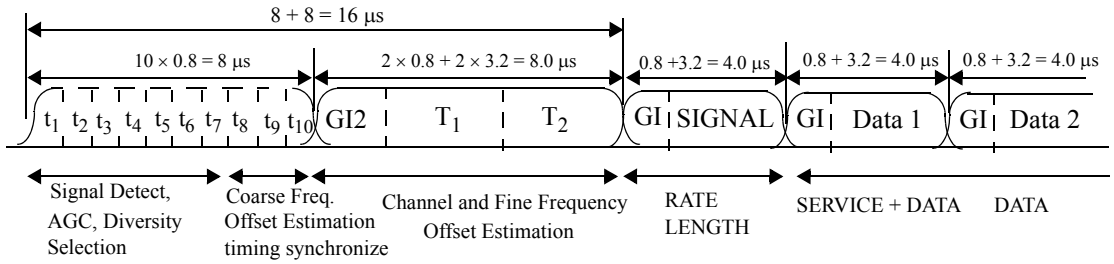

Figure 4.7: Use of long and short training sequence [15]

4.0.3 Filtered OFDM

In F-OFDM, frequency domain filtering is done to achieve improvement in out-of-band attenuation. It uses linear phase finite impulse response filter after OFDM modulation as shown in Fig. 4.8. The linear-phase filter is designed using the Parks-McClellan algorithm. The Parks-McClellan algorithm uses the Remez exchange algorithm and Chebyshev approximation theory to design filters with an optimal fit between the desired and actual frequency responses. The filters are optimal in the sense that the maximum error between the desired frequency response and the actual frequency response is minimized.

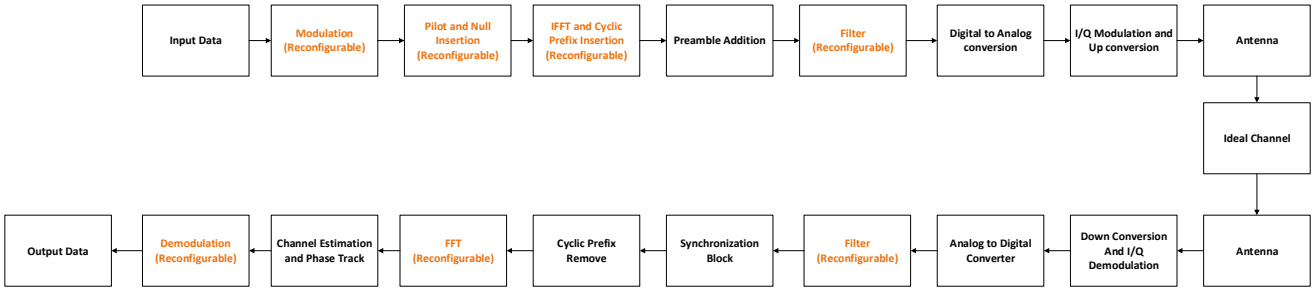

Figure 4.8: Filter OFDM Transceiver block

The use of FOFDM [16][17][18] enables higher transmission bandwidth compared to bandwidth limitation in WOLA-OFDM based system. In addition, sub-band filtering approach of F-OFDM

allows transmission in non-contiguous bands and sharing of adjacent frequency band among multiple asynchronous transceivers. The use of filters however leads to increase in the complexity of the transceiver when compared to WOLA-OFDM and OFDM.

In this project the F-OFDM transceiver uses a linear phase bandpass filter [19][20] of order 50 with a normalized bandwidth of 0.9 and the transition bandwidth of 0.05. A symmetric coeficient filter is used to reduce the area and power utilization. Filtering is convolution in time domain of the input samples and the filter coefficients. Thus, theoretically, number of output samples (N_0) is given by:

$$N_0 = N_i + H - 1 \tag{4.1}$$

is the number of input samples and H is the filter order.

Chapter 5

Implementation Details, Results and Discussion

In this section, we will know about the implementation details of the experiment then we will compare the resource utilization of the various designs. The 802.11a with subcarriers mapping for 650 Khz model with both QPSK and QAM16 is realized first on the zedboard. The inter module communication is taken care by the wishbone interface protocol.

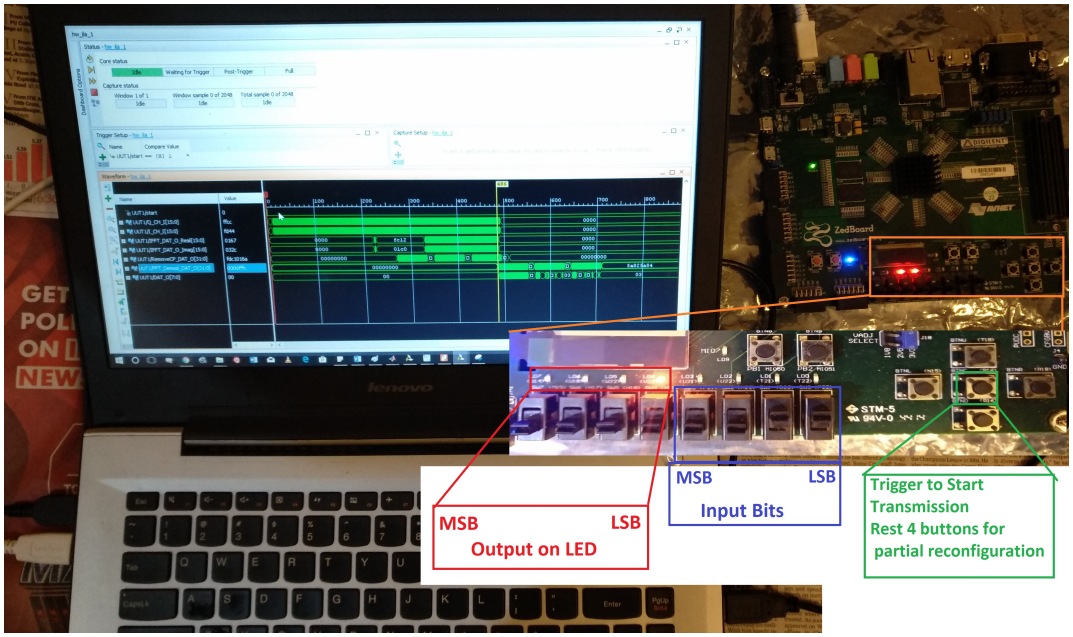

Figure 5.1: Experiment Setup and data from different blocks on FPGA are displayed through ILA

The input is taken from the switch 0 to 3 of zed board as shown in Fig. 5.1. If the modulation scheme is QPSK then the lower two inputs are taken else all the four inputs are taken. The modulation block maps the set of two bits from input to 16 bits binary value where the MSB represents the sign and rest 15 bits are the bits to the right of decimal point. Pilot data are loaded from the file in the ram. The pilot insertion block then maps the pilot and null data at

the specified location in a frame. The data is transferred frame by frame to IFFT block. Here, the IFFT size is fixed to 64 and the cyclic prefix of 16 length from back is appended to the front. The final transmitter block first transmit the preloaded long and short preamble from the RAM then it simply pass out the out of the IFFT block. The output of the transmitter block is then passed through filter as in case of F-OFDM implementation.

The final output of transmitter is directly feed to the receiver. Here the receiver is configured to be always listening. Inside the receiver, the sync block detects the frame start by synchronizing itself with the input sequence using the autocorrelation with long and short preambles.Then the frame data is passed to the cyclic prefix removable block. The length of cyclic prefix to remove is fixed to sixteen in all the variants of the design. The cyclic prefix removed frame data is passed to FFT block. The output of FFT block is directly fed to demodulation block as no channel estimation and phase track is required here as the data have been passed through ideal channel.

To demonstrate the partial reconfiguration capability of the design,initially, in the base design only modulation/demodulation blocks are dynamically reconfigured following the 802.11a specification. Further, for demonstration purpose the reconfiguration of IFFT/FFT blocks of 64 and 128 sizes were also introduced. Note that 128 size versions are not following any standard as it is not operating at twice the base frequency to process twice the data in the same time required in 64 size version and also it's cyclic prefix length is kept sixteen.So a total four combination of transmitter designs are possible as shown in Fig. 5.2. Similarly, receiver design is loaded corresponding to current transmitter design on FPGA.

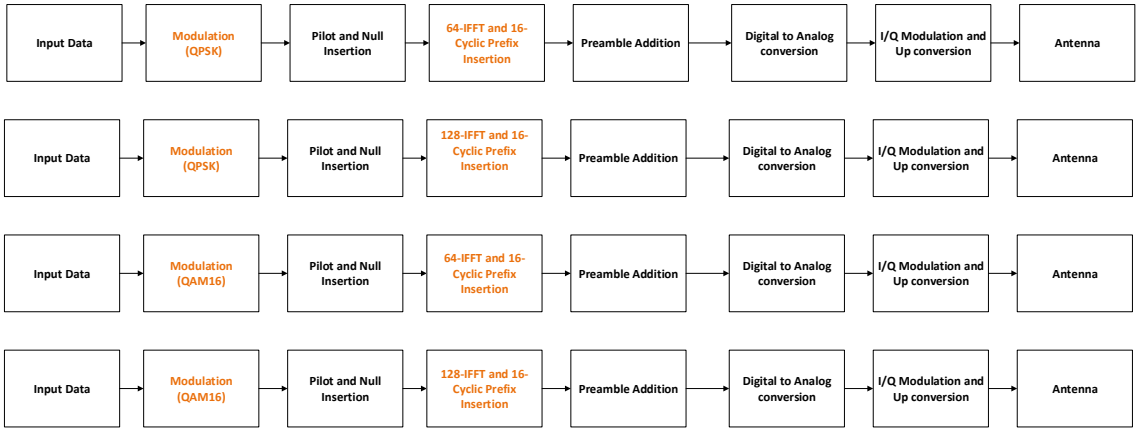

Figure 5.2: Different types of Transmitter possible in current implementation

5.0.1 Integrated Logic Analyzer Output

The customizable Integrated Logic Analyzer (ILA) IP core is a logic analyzer core that can be used to monitor the internal signals of the FPGA design. The triggering of the ILA core can be done through Boolean trigger equations, and edge transition triggers.So the triggering can be programmed on the fly. Because the ILA core is synchronous to the design being monitored, all design clock constraints that are applied to the design are also applied to the components inside

the ILA core. ILA core always runs on the base system clock frequency and it cannot be driven by clock divider or counters.

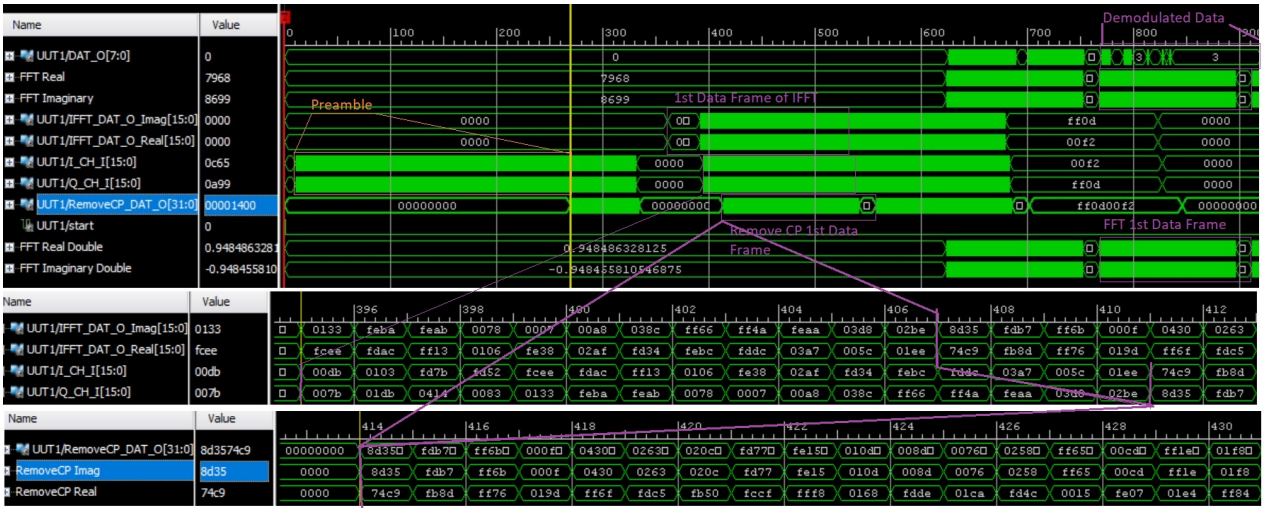

Figure 5.3: ILA output of 16-Qam and 128 length IFFT or FFT size model

A typical output of ILA is shown in Fig. 5.3. Here, the data output of FFT block, IFFT block, receiver input data (I_CH_I and Q_CH_I), remove cyclic prefix block and finally the net output demodulated data (DAT_O) is shown. A trigger represented by start is generated when the middle push button of the zedboard is pushed. All the intermodule data transfer length is of 32 bits (16 bits for imaginary channel data and they are in 16-MSB positions and the rest 16 LSB bits are for real channel data). There is wiggling in output demodulated data because this data contains both pilots,nulls and the actual input data. If we by pass pilot and null insertion then this wiggling is removed and we get only the input data. The input is mapped in modulation block as signed16 1.15 format. The modulation mapping of input of a single channel(inphase or quadrature) is shown in Table 5.1.

Table 5.1: Mapping of input bits of a single channel on signed16 1.15 format

Modulation Scheme	Bit	Mapped Value in Hex	Signed Mapped Value
QPSK	0	A57E	-0.7071
QPSK	1	5A82	0.7071
16-QAM	00	8692	-0.9485
16-QAM	01	287A	0.3162
16-QAM	10	D786	-0.3162
16-QAM	11	796E	0.9487

5.0.2 Results

Table 5.2 contains the resource utilization of all the different version of design as shown earlier in Fig. 5.2. Here, all the four designs are independently implemented. The placing and routing is down automatically by the Xilinx Vivado. No manual floorplanning is done here. If we consider

the automatic routing and placement to be most efficient implementation the design. Then it means that the resource utilization as shown in table 5.2, of the transceiver model which contain 128-FFTIFFT and QAM-16 will be the least requirement for the dynamic partial reconfiguration model which supports all four transceivers. Currently, in dynamic partial reconfigurable design the floorplanning is done manually. Here, we have selected the region dedicated to reconfigurable parts through trial and error. The velcro implementation is bound to exceed this resource utilization number as it will require two versions of each modulation, demodulation, FFT and IFFT blocks. As it can be seen from the table 5.2, velcro model uses upto ten percent more slices, twelve percent more LUTs and 11.6 percent more Slice Registers then the most complex 128-IFFT and 16-Qam model. Here, velcro model means all the reconfigurable blocks are implemented parallely Number of DSP48 units is constant in all 128-IFFT/FFT design because of the same size of IFFT/FFT implementation which require same number of DSP48 to implement , similar is the case for 64-IFFT/FFT implementation. In velcro design, the IFFT/FFT size is implemented IFFT/FFT size is said to equal to 128 because the 64 length IFFT/FFT output is generated simply by down sampling of 128 length IFFT/FFT output by two which is taken care by the tool. We just need to change the configure data to activate this support at the run time.

Table 5.2: Complexity comparison between different Architecture

Parameters	**Velcro Design**	**128-IFFT and 16-QAM**	**128-IFFT and QPSK**	**64-IFFT and 16-QAM**	**64-IFFT and QPSK**
SLICEL	2457	2180	2200	2110	2070
SLICEM	1078	1017	990	960	857
Total Slices	3535	3197	3190	3070	2927
LUT as Logic	7196	6576	6574	6078	6072
LUT as Distributed RAM	317	317	317	317	317
LUT as Shift Register	1818	1498	1498	1247	1247
LUT as Memory	2135	1815	1815	1564	1564
Slice Registers	13882	12440	12414	11421	11394
Block RAM Tile	10.5	10.5	10.5	10.5	10.5
DSP48E1	23	23	23	17	17

The power analysis result of all the models are shown in table 5.3. Here only the dynamic power consumption estimate is shown as the static power consumption in most case almost independent of the design.Static consumption is constant because it does not depend on the implemented logic, it's mostly the power used by clock routing resources and other resources which are necessary to keep FPGA up and running. In current case, the estimated static power consumption value for all the designs is 0.122 Watt. From dynamic power consumption estimate, the dynamic consumption of dynamic partial model is bound to be between 0.031 W to 0.028 W. The velcro design as expected is consuming approximately 26 percent more power than the biggest design i.e. 128-IFFT/FFT and 16 QAM design.

The results of FOFDM is not shown as we were facing problem in synchronization of the filtered

Table 5.3: Dynamic Power Consumption in watt

Velcro design	128-IFFT and 16-QAM	128-IFFT and QPSK	64-IFFT and 16-QAM	64-IFFT and QPSK
0.039 W	0.031 W	0.031 W	0.029 W	0.028 W

OFDM data at the receiver end. We were not able to use the higher order filter (currently we are using the symmetric filter of order 80) due to limited resources available on zedboard. Also, currently the size of the filter coefficients is 16-bits and the format is signed16 1.15. So,the coefficients with the value within 4.5 places to the right of decimal can only be accounted for. As most of the coefficients of this filter are very small,so in hardware most of this value is simply implemented as zero. So, this adds non linearity to the filtered data. In synchronization block, we are able to detect the presence of OFDM signal but the point at which it should ideally start synchronizing is missed, resulting in 10 to 16 samples shifted data feeding into the FFT chain which results in wrong FFT output. Filter order increase, drastically increases the resource utilization which is the main disadvantage of FOFDM. We are trying tackle this situation with the efficient filter implementation.

Chapter 6

Conclusion and Future Work

The main aim of this thesis was to implement a transceiver which supports dynamic partial re-configuration of it's module on the fly. Here the modulation scheme can be changed between QPSK and 16-QAM. Also, support for 128 bit IFFT/FFT size is added to the design. The resource utilization and power utilization values are bounded by implementing all the version in the independent design form. Then, it was proved that the dynamic power consumption and the resources used are indeed saved using this design methodology with respect to the velcro approach. Further this design was modified to support filtering operation required in the F-OFDM. The filter bandwidth is decided by the data transmitted bandwidth. Currently, maximum of 80-tap symmetrical filter is implemented in the F-OFDM.

So, this work result supports the use of partial dynamic reconfiguration in comparison with the existing velcro approach for implementing wireless testbeds. More design comparison is required to further validate this trend which may be a part of future work. Also, currently this is a Programmable Logic dominated work. So, in future extension of this design more serial logic could be ported to processor for more efficient implementation in terms of area and power.

The other extensions of the proposed work are :

- Existing architecture is derived from a simulated design of 802.11a . This architecture (i.e. Verilog code) was not designed taking into account the characteristics of partial reconfiguration capability of FPGA. In future, we would like to optimize the existing architecture to exploit the advantages of partial reconfiguration in better way.
- Existing filtered OFDM architecture has some error in implementation due to limited dynamic range of filter coefficients. Here,the receiver is not able to synchronize with the transmitted data automatically at correct sample in ideal environment. We would like to improve this by efficient filter design.
- We would like to validate the functionality of the proposed architecture in real radio environment.

Bibliography

[1] J. Wang, M. Ghosh and K. Challapali, "Emerging Cognitive Radio Applications: A Survey," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 74-81, March 2011.

[2] J. Palicot, H. Zhang and C. Moy, "On The Road towards Green Radio," *URSI Radio Science Bulletin*, no. 347, pp. 4056, Dec. 2013.

[3] Navin Michael, A. P. Vinod, "Design of Multistandard Channelization Accelerators for Software Defined Radio Handsets," *IEEE Transactions On Signal Processing*, vol. 59,no. 11, 2011.

[4] Dirk Koch, Daniel Ziener, "Partial reconfiguration on FPGAs in practice Tools and applications," *ARCS Workshops (ARCS)*, 2012.

[5] S. J. Darak and A. P. Vinod, "Design of Low Complexity Variable Digital Filters and Recongurable Filter Banks for Multi-Standard Wireless Communication Receivers," PhD Thesis, NTU Singapore, 2013

[6] Product Specification, "Zynq-7000 All Programmable SoC Data Sheet: Overview," *Xilinx Inc.*, 2017

[7] User Guide, "Partial Reconfiguration," *Xilinx Inc.*, 2012

[8] Wang Lie and Wu Feng-yan, "Dynamic partial reconfiguration in FPGAs, " *Third International Symposium on Intelligent Information Technology Application*, 2009

[9] Kizheppatt Vipin and Suhaib A. Fahmy, "Efficient region allocation for adaptive partial reconfiguration, " *International Conference on Field-Programmable Technology*, 2012

[10] Christian Beckhoff, Dirk Koch and Jim Torresen, "GOAHEAD: A Partial Reconfiguration Framework, " *IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*,2012

[11] Behrouz Farhang-Boroujeny, "OFDM Versus Filter Bank Multicarrier, " *IEEE Signal Processing Magazine*, 2011

[12] Ye Li, "OFDM for Wireless Communications: Techniques for Capacity Improvement, " *IEEE Intemational Conference on Communication Technology*, 1998

[13] B. Drozdenko, M. Zimmermann, Tuan Dao, M. Leeser and K. Chowdhury, "High level hardware software codesign of an 802.11a transceiver system using Zynq SoC , " *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 682-683, San Francisco, CA* , 2016

[14] B. Drozdenko, M. Zimmermann, Tuan Dao, M. Leeser and K. Chowdhury, "Hardware Software Codesign of Wireless Transceivers on Zynq Heterogeneous Systems , " *IEEE Transactions on Emerging Topics in Computing, vol. PP, pp. 99* , 2017

[15] , "https://standards.ieee.org/findstds/standard/802.11-2016.html ," , 2016

[16] Abdoli, M. Jia and J. Ma, "Filtered OFDM: A new waveform for future wireless systems, " *16th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 66 to 70,Stockholm, Sweden* , 2015

[17] X. Cheng, Y. He, B. Ge and C. He, "Filtered OFDM Using FIR Filter Based on Window Function Method," *IEEE Vehicular Technology Conference (VTC Spring), pp. 1 to 5, Nanjing, China*, May. 2016.

[18] A. Farhang, M. Molavi Kakhki and B. Farhang-Boroujeny, "Wavelet-OFDM versus filtered-OFDM in power line communication systems," *5th IEEE International Symposium on Telecommunications (IST), pp. 691 to 694, Tehran, Iran* , Dec. 2010.

[19] S J. Darak, A. P. Vinod, E. M-K. Lai, Honggang Zhang and Jacques Palicot, "Linear Phase VDF Design with Unabridged Bandwidth Control over the Nyquist Band," *IEEE Transactions on Circuits and Systems - II (TCAS-II), vol. 61, no. 6, pp. 428-432*, Apr. 2014.

[20] S. J. Darak, A. P. Vinod, and E. M-K. Lai, "Low Complexity Recongurable Non-uniform Filter Bank for Channelization in Multi-standard Wireless Communication Receivers," *Journal of Signal Processing Systems (Springer), vol. 68, no. 1, pp.95-111*, July 2012.