DESIGN AND FPGA IMPLEMENTATION OF A RECONFIGURABLE CHANNELIZATION ARCHITECTURE FOR SDR APPLICATION

A PROJECT REPORT

Submitted by

MANOJKUMAR M

Register No: 15MAE007

in partial fulfillment for the requirement of award of the degree

of

MASTER OF ENGINEERING

in

APPLIED ELECTRONICS

Department of Electronics and Communication Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

(An autonomous institution affiliated to Anna University, Chennai)

COIMBATORE-641049

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2017

BONAFIDE CERTIFICATE

Certified that this project report titled **"DESIGN AND FPGA IMPLEMENTATION OF A RECONFIGURABLE CHANNELIZATION ARCHITECTURE FOR SDR APPLICATION"** is the bonafide work of **MANOJKUMAR M [Reg. No. 15MAE007]** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Prof. S.GOVINDARAJU

PROJECT SUPERVISOR

Department of ECE

Kumaraguru College of Technology

Coimbatore-641 049

SIGNATURE

Prof. K.RAMPRAKASH

HEAD OF PG PROGRAMME

Department of ECE

Kumaraguru College of Technology

Coimbatore-641 049

The Candidate with **Register No. 15MAE007** was examined by us in the project viva-voce examination held on…….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First, I would like to express my praise and gratitude to the Lord, who has showered his grace and blessings enabling me to complete this project in an excellent manner.

I express my sincere thanks to the management of Kumaraguru College of Technology and Joint Correspondent **Shri Shankar Vanavarayar** for the kind support and for providing necessary facilities to carry out the work.

I would like to express my sincere thanks to our beloved Principal **Dr. R. S. Kumar,** Kumaraguru College of Technology, who encouraged me with his valuable thoughts.

I would like to thank **Prof. K. Ramprakash,** Head of PG programme, Electronics and Communication Engineering, for his kind support and for providing necessary facilities to carry out the project work.

I am greatly privileged to express my heartfelt thanks to my project guide **Prof. S. Govindaraju,** Department of Electronics and Communication Engineering, for his expert counseling and guidance to make this project to a great deal of success.

In particular, I wish to thank with everlasting gratitude to the project coordinator **Mrs. S. Nagarathinam,** Assistant Professor, Department of Electronics and Communication Engineering, throughout the course of this project work. I wish to convey my deep sense of gratitude to all teaching and non-teaching staff of ECE Department for their help and cooperation.

Finally, I thank my parents and my family members for giving me the moral support and abundant blessings in all of my activities and my dear friends who helped me to endure my difficult times with their unfailing support and warm wishes.

ABSTRACT

A novel channelization architecture, which can simultaneously process two channels of complex input data and provide up to 512 independent channels of complex output data has been proposed. The architecture is highly modular and generic, so that parameters of each output channel can be dynamically changed even at runtime in terms of the bandwidth, center frequency, output sampling rate, and so on. It consists of one tunable pipelined frequency transform (TPFT)-based coarse channelization block, one tuning unit, and one resampling filter. Based on the analysis of the data dependence between the subbands, a novel channel splitting scheme is proposed to enable multiple subbands to share the proposed TPFT block. The multiplier block (MB) and subexpression sharing techniques are used to reduce the number of arithmetic units of the TPFT block.Moreover, the proposed Farrow-based resampling filter does not require division operation and dual-port RAMs resulting in significant area saving. Finally, the proposed channelization architecture is implemented on single field-programmable gate array.The experiment results indicate that the design provides the flexibility associated with the existing work, but with greater resource efficiency.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACRONYMS	ABBREVIATIONS
FPGA	Field Programmable Gate Array
DSP	Digital Signal Processors
SDR	Software Defined Radio
VHDL	Very High Description Language
SRC	Sampling Rate Conversion
ADC	Analog to Digital converter
DAC	Digital to Analog converter
TPFT	Tunable Pipelined Frequency Transform
DDC	Digital Down Converter
PE	Processing Element
FFT	Fast Fourier Transform
VFD	Variable Fractional Delay
NCO	Numerical Control Oscillator
RTL	Register Transfer Level
ASIC	Application Specific Integrated Circuit
HDL	Hardware Description Language

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The wireless industry has been experiencing an exponential growth with the emergence of new radio access technologies and standards. All these technologies have been optimized to obtain a good trade-off between data rate, range and mobility to suit specific application needs. Lack of harmony in spectrum allocation globally has also resulted in this growth. However, with the increase in trade relationship between different continents, researchers had to look for a common multi-standard wireless communication platform which can support all these radio technologies and standards. This has resulted in the birth of the software defined radio (SDR) concept. SDR can be regarded as an ultimate communications solution which can ideally cover any cellular communication standard in a wide frequency spectrum with any modulation and bandwidth. The term SDR signifies that the same hardware architecture can be programmed or reconfigured to cope with any radio standard. The major application of SDR will be in mobile communication transceivers, generic cellular base stations and military radio systems. Some of the benefits that will result with the realization of SDR are:

- Easier international roaming, improved and more flexible services, increased personalization and choice for subscribers of mobile services.
- The potential to rapidly develop and introduce new value-added services and revenue streams with increased flexibility of spectrum management and usage for mobile network operators.
- The promise of increased production flexibility, improved and more rapid production evolution for handset and base station manufacturers.
- The prospect of increased spectrum efficiency and better use of scarce resource for regulators.

1.2 OBJECTIVE OF THE STUDY

SOFTWARE-DEFINED RADIO(SDR) has been widely applied to many fields, such as communication, electronic warfare and instrumentation. The core idea of the SDR is to move the analog-to-digital converters (ADCs) and digital-to analog converters (DACs) as close as possible to the antenna, and to process the digitized data by a software technique. The SDR receiver must be capable of extracting channels corresponding to different communication standards which are independent of each other. This means specifically that the receiver must be capable of extracting non-uniform bandwidth channels as bandwidths of different standards are different. The receiver must be capable of extracting narrowband channels from the wideband input signal. In an ultimate SDR, reconfigurability of the receiver must be accomplished by reconfiguring the same filter bank for a new communication standard with minimal reconfiguration overhead, instead of employing separate filter banks for each standard. In the sequel, we call this requirement as ultimate reconfigurability. The development of ADCs, DACs, and digital circuits has greatly promoted the progress of SDR. Thus SDR should be able to support multiple communication standards by dynamically reconfiguring the same hardware platform. Also, SDR should be able to use the same architecture for any number of channels by simply reconfiguring the digital front-end as compared to a conventional radio transceiver whose complexity grows linearly with the number of received channels

A key component in SDR systems is a real-time configurable digital channelization, which is essential for receiving and resampling the radio signal correctly. Compatibility with different communication standards requires that channelization be dynamically reconfigurable. A resource efficient design is another key requirement for the implementation of channelization.

For digital channelization it is difficult to obtain a perfect solution that can balance all the targets, such as performance and resource. The design and the implementation of channelization are affected by many parameters, such as output sampling rate, bandwidth, frequency resolution, and dynamic configuration. Exploring the cost/performance tradeoffs is an important issue for digital channelization.

Digital communication systems generally perform many of the functions for transmitting and receiving digital data in the analog domain. While the use of analog circuitry can significantly reduce the computational requirements placed on the digital signal processors (DSP) of communication systems, it severely limits the portability of these systems to other communication standards. The software radio (SWR) concept has been introduced as a means to reduce or eliminate this portability problem by placing the analog-to-digital converter (ADC) and digital-to-analog converter (DAC) of the radio transceiver as close to the antenna as possible so that most communication functions can be performed in the digital domain. Such an approach gives SWR systems the flexibility to be ported to different communication standards and the ability to serve multiple communication standards simultaneously by software modification. However, SWR systems require powerful DSPs for performing functions such as channelization digitally, and they possibly require functions that may not be required in conventional radio transceivers, such as sample rate conversion (SRC).

Due to variations in the propagation environment experienced by the frequency division multiplexed (FDM) channels and the desired flexibility obtained by using a single wideband ADC and DAC, the signals processed by a SWR transceiver may contain the complete transmission band of a wireless air interface, with bandwidths of 25–100 MHz and dynamic ranges in excess of 90 dB. This requires ADCs with spurious-free dynamic ranges (SFDRs) of 90–100 dB and sampling rates as high as 250 Samples/s. A significant fraction of the computational power of wideband SWR base station receivers may be dedicated to channelizing the received wideband signals digitally and preparing the extracted channels for baseband processing. The computations for synthesizing the channels in wideband SWR base.+ station transmitters may also be high. The high computational cost for channelizing/synthesizing SWR channels is attributed to the long filters that are required for processing wideband high-dynamic-range signals that consist of a large number of channels. Reducing the computations for channelization/synthesis of wideband signals in SWR transceivers is vital for reducing the cost and power consumption of DSPs in SWR systems.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

This chapter presents the literature survey in the area of designing and implementing channelization architecture using coarse channelization, tuning unit and resampling filter. The purpose of this is to optimize the filter design in order to reduce the area and improve the performance.

2.2 CHANNELIZATION

Channelization is the extraction of independent communication channels from a wideband signal, performed in the receiver of a communications device. Channelization is achieved by filtering, to isolate the channels of interest, and down-conversion, to prepare the channels for subsequent baseband processing.

In wireless communications, and more specifically mobile networks, the Radio Frequency interface is formed by two types of devices: base stations and mobile stations. The physical RF channels employed to transmit information from base stations to mobile stations are termed Downlink channels. On the other hand, the physical channels used to transmit information from mobile stations to base stations are termed Uplink channels. For a mobile station, channelization generally means extracting a single information channel from the DL signal. At the base station side, however, channelization normally implies the extraction of multiple channels from the UL signal, where different channels originate from different mobile stations. Consequently, the channelizer (and in general the complete physical layer of the receiver) must be designed in accordance to the single-channel or multi-channel extraction requirements

2.3 SOFTWARE DEFINED RADIO

Software-defined radio (**SDR**) is a radio communication system where components that have been typically implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead

implemented by means of software on a personal computer or embedded system. While the concept of SDR is not new, the rapidly evolving capabilities of digital electronics render practical many processes which used to be only theoretically possible.

2.3.1 CONCEPT

The ideal receiver scheme would be to attach an analog-to-digital converter to an antenna. A digital signal processor would read the converter, and then its software would transform the stream of data from the converter to any other form the application requires.

An ideal transmitter would be similar. A digital signal processor would generate a stream of numbers. These would be sent to a digital-to-analog converter connected to a radio antenna.

The ideal scheme is not completely realizable due to the actual limits of the technology. The main problem in both directions is the difficulty of conversion between the digital and the analog domains at a high enough rate and a high enough accuracy at the same time, and without relying upon physical processes like interference and electromagnetic resonance for assistance.

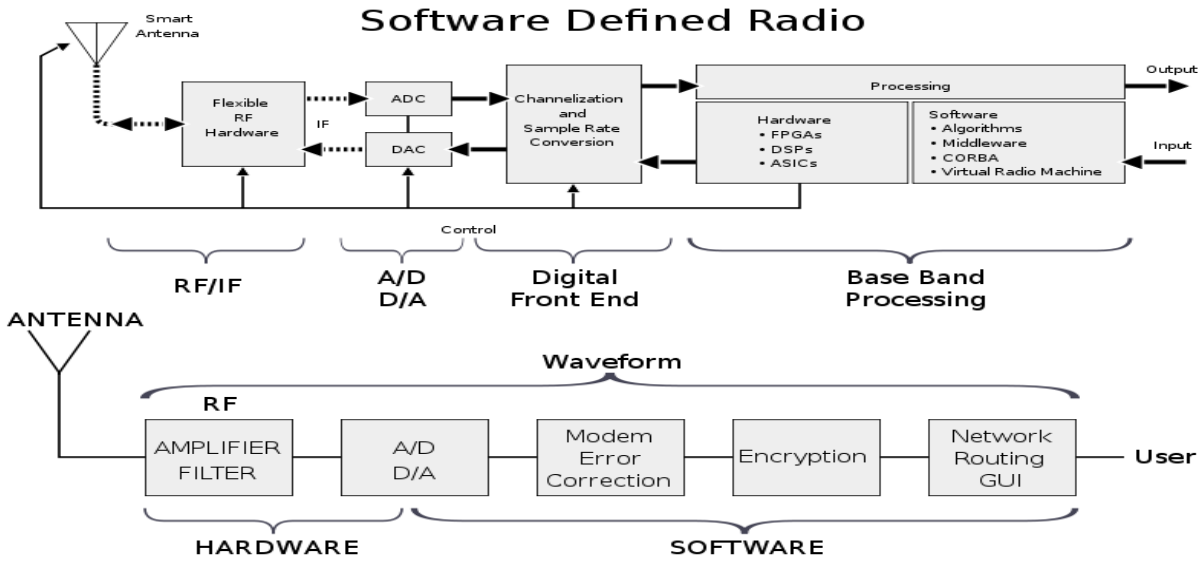


Fig 2.1 Software Defined Radio a) Ideal Receiver b) Ideal Transmitter

2.4 SAMPLING RATE CONVERSION

Sampling Rate Conversion is necessary in software defined radio in order to able to have an autonomous asynchronous analog-digital interface but have a digital signal available at a standard specific rate for a base band processing**.**

Field-Programmable Gate Arrays (FPGAs) are used a wide variety of signal processing applications, including radio communication systems and Software Defined Radio(SDR). In such systems, the initial sampling frequency may not be convenient for a subsequent processing stage, for example, a filter bank in the system requiring a data sampling frequency different from the initial rate. Resampling, or sample rate conversion, SRC, is the operation of changing the sampling rate of a given digital signal. In most cases, the required rate modification is not an integer multiple; this is what we refer to as arbitrary ratio re-sampling.

Although the performance of FPGAs is continually increasing, they still suffer from certain limitations related to intrinsic clock speeds (e.g., maximum clocks speeds of internal digital signal processing blocks; ensuring the clock network feeds correctly every part of the design, etc.); of place-and-route tool performance; and total available chip resources. These concerns limit the range of frequencies and circuit complexity which can practically be exploited in a FPGA architecture, so that certain architectures which may appear viable from a theoretical standpoint may not in fact be implementable on a particular target platform.

Alternatively, if an FPGA does support very high frequencies, the designer's equipment may be unable to generate a clock with a sufficiently high frequency. Spurious Free Dynamic Range, or SFDR, defined as the ratio of a signal to its largest spurious harmonic, is a natural quality criterion for sampling systems. In many situations, spurious harmonics must be handled carefully because they may negatively influence the output of a processing chain. In the case of a SDR system for demodulating Frequency Modulation (FM) radio stations, for example, the amplitude difference between the strongest and weak stations can be 50 dB or more. The designer

must then ensure that the resampling procedure does not introduce spurious harmonics that may mask some of the weaker stations. At the same time, the amount of available resources is limited on an FPGA, which places limitations on achievable SFDR, making design tradeoffs necessary

.

2.5 DIFFERENT CHANNELISATION METHODS.

2.5.1 GENERAL ISSUES.

Before dealing in detail with the various channelisation techniques, a broad overview might be useful. It is not intended to cover some of the more specialist applications such as Chirp-Z transforms or Wavelet filter banks (even though some of the techniques described below may have applicability in these areas). It is those basic techniques which provide multiple channels from a broad band for further processing such as demodulation or signal detection which are of interest here. It is also worth pointing out that the architectures discussed are generally biased towards hardware implementation such as in FPGA's or ASIC's. Firstly, the processing power required in terms of "multiply /accumulate" operations (MACs) is very high, and, in most cases, way in excess of the peak MAC performance of today's programmable DSPs. Secondly, the most difficult aspect to overcome is the memory bandwidth requirements of a wide band, real-time system. For all the cases considered, it is not clear how this could be achieved without using a totally impractical number of DSPs for the high-end specifications.

2.5.2 DIGITAL DOWN-CONVERTERS (DDC'S).

Digital down-converters (DDC's) are a well-established technique and can be found in COTs products from, for example, Analog Devices, Intersil and TI-Graychip. It is also relatively straightforward to implement them in FPGA's using custom or standard cores. In cases where only a few channels (typically 4 to 8) need to be selected from the broad band, then such a solution is quite efficient. It also

provides a very flexible solution in that each channel can be independently configured for center frequency, bandwidth and filter response. For larger numbers of channels, however, the logic and, more particularly, the memory requirements become excessive, as will be demonstrated later.

2.5.3 FAST FOURIER TRANSFORM (FFT).

The Fast Fourier Transform (FFT) and its real-time pipelined implementation are also well established. It provides a very economical solution to the channelisation problem, especially where a large number of channels is required and the channel filter performance is not too critical. It is also generally restricted to cases requiring channels with even frequency spacing and equal filtering.

2.5.4 WOLA AND POLYPHASE DFT FILTER BANKS.

An improvement to the filtering performance can be achieved by the use of polyphase filter banks ahead of the FFT, rather than the use of simple "windowing" of the time data. The technique, generally called the "Weight Overlap and Add" or WOLA or its subset the "Polyphase DFT", is becoming more established and is certainly very efficient where large, high quality filter banks are required. Like the FFT, however, it is generally restricted to cases requiring evenly spaced channels with equal filtering.

2.5.5 PIPELINED FREQUENCY TRANSFORM (PFT).

A novel form of processing, known as the Pipelined Frequency Transform (PFT) uses a different approach. Based on a "tree" structure, successive splitting and filtering of the frequency band is used to achieve a finer and finer resolution of the broad band. Time interleaving of common processes can lead to a very efficient structure. Advantages include the availability of simultaneous outputs from successive stages, which are at different frequency resolutions and also the ability to independently tailor the filters for different frequency bins. Furthermore, if certain frequency bins or blocks of spectrum are not required, it is simple to exclude

them from the processing, leading to greater efficiency.

2.5.6 TUNABLE PIPELINED FREQUENCY TRANSFORM (TPFT).

TPFT allows independent tuning of the center frequency of all bins as well as independent filters for each bin. Because of the availability of different stage outputs, with different frequency resolutions, the end result is equivalent to having the flexibility of the digital down converter approach but with the efficiency of the pipelined frequency transform which is important for a larger number of channels.

2.6 INTERLEAVER

An interleaver is a hardware device commonly used in conjunction with error correcting codes to counteract the effect of burst errors. Interleavers are in widespread use and much is known about them from an engineering standpoint Interleaving is a standard signal processing technique used in a variety of communications systems an interleaver is a hardware device that takes symbols from an fixed alphabet as the input and produces the identical symbols at the output in a different temporal order.

The classical use for interleaving is to disperse sequences of bits in a bit stream so as to minimize the effect of burst errors introduced in transmission Error correcting codes can correct errors successfully as long as there are not too many errors in a single code word. However, errors sometimes tend to be bursty in the sense that there can be a local concentration of many errors too many for typical error correction schemes to handle.

2.7 VHDL

VHDL stands for Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (HDL). It is a language for describing digital electronic systems. It was born out of the United States Government’s VHSIC program in 1980 and was adopted as a standard for describing the structure and function of Integrated Circuits (IC). Soon after it was developed and adopted as a standard by the Institute of Electrical and Electronic Engineers (IEEE) in the US (IEEE-1076-1987) and

in other countries. VHDL continues to evolve. Although new standards have been prepared (VHDL-93) most commercial VHDL tools use 1076-1987 version of VHDL, thus making it the most compatible when using different compilation tools. The 1076-1987 standard has also been used here. VHDL enables the designer to:

- Describe the design in its structure, to specify how it is decomposed into sub-designs, and how these sub designs are interconnected.
- Specify the function of designs using a familiar, C-like programming language form.
- Simulate the design before sending it off for fabrication, so that the designer has a chance to rapidly compare alternative approach and test for correctness without the delay and expense of multiple prototyping.

VHDL is a C-like, general purpose programming language with extensions to model both concurrent and sequential flows of execution, and allowing delayed assignment of values. To a first approximation VHDL can be considered to be a combination of two languages: one describing the structure of the integrated circuit and its interconnections (structural description) and the other one describing its behaviour using algorithmic constructs (behavioural description).

VHDL allows three styles of programming:

1. Structural
2. Register Transfer Level
3. Behavioural

The first one, structural, is the most commonly used as it allows description of the structure of the IC very precisely by the user. This in very many cases gives the best performance over compiler optimised structures, especially for high speed, fixed-point applications. Its behavioural style permits the designer to quickly test concepts, where the designer can specify the high-level function of the design without taking much care how it will be done structurally.

CHAPTER 3

SOFTWARE USED

3.1 MATLAB

MATLAB (**mat**rix **lab**oratory) is a numerical computing environment and fourth-generation programming language. Developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and FORTRAN. Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems. It is a powerful tool use to design any mathematical model very easily and check there result. MATLAB has several specifications but by using MATLAB the analysis of hardware design is not possible it gives the software analysis and results using its calculation in matrix form. To analyze hardware, VHDL codes and VERILOG codes are used by which implementation of that code can be made possible on target device. Now a major question rises is MATLAB code able to convert its software design in any hardware descriptive language and the answer comes itself by using HDL coder in MATLAB. HDL coder is work as a bridge which connects two different software MATLAB and HDL software. HDL coder one can convert .m code to vhdl code or it can change the MATLAB code to VHDL code so that it can implement on FPGA kit. Main advantage of HDL coder is that it gives the elapsed time to execute the program. To convert code, initially it is compulsory to convert floating point design (MATLAB code) to fixed point design in which every variable length is fixed to a particular no of bits for MATLAB added support for 64-bit indexing. Digital signal processing can be divided into two categories fixed and floating points. This refers to format used to store and manipulate no. within devices

3.1.1 FLOATING AND FIXED POINT VARIABLES

Fixed point DSPs are usually representing no. in bit form of different length (16 bit or more). Floating point DSP refers to the values where variable size is not fixed. MATLAB code is a floating point design and HDL code is a fixed point design.

3.1.2 HDL CODER

HDL coder is a coder in MATLAB which is used to covert .m code into VHDL code so that the code can run for FPGA and ASIC design. It is target independent for Xilinx and Altera FPGA's. This coder supports only MATLAB functions it provides an advisor which can generate VHDL codes. HDL coder is open using coder command. Those commands ask to open earlier project or to create new project.

3.2 SIMULINK

Simulink is an environment for simulation and model-based design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that let you design, simulate, implement, and test a variety of time-varying systems, including communications, controls, signal processing, video processing, and image processing.

Simulink offers:

A quick way of develop your model in contrast to text based-programming language such as e.g., C.

Simulink has integrated solvers. In text based-programming language such as e.g., C you need to write your own solver.

Simulink as is true for most of high-level simulation software, does not allow testing certain behaviour patterns that a real target design can exhibit, most of which are available for the VHDL simulator. The most reliable simulation can only be performed after porting the compiled VHDL into the implementation software. Simulink does not:

- Do fixed point arithmetic in the general sense (expected in a future version).

- Have data types compatible with bit logic (bits can only be simulated with floating-point).
- Incorporate propagation delay in its blocks, which is not relevant at this level of abstract, but necessary for the implementation.
- Support reusable symbols (they may have different contents and the same name).

In the structural simulation using bit logic arithmetic it is possible to force Simulink to assign only 0s and 1s, even though they are represented with floating-point variables/signals. Fixed-point arithmetic can be implemented structurally in Simulink using gates. This also simplifies setting propagation delay, as this could be included into the VHDL description of each gate. However, this is not possible in the Simulink model. Summarizing, the structural fixed-point design can be quite easily converted into VHDL directly, without much additional intelligence required from the conversion program.

The model description of the Simulink block is very similar to the representation of the common structure. It contains both the parameters of the simulation, description of each block with parameters for each block and block connections. The problem is that Simulink does not use reusable symbols. These makes the analysis of common blocks much more difficult as these blocks may have slight differences and then qualify as two different ones, even if they have the same name. Therefore, the designer must obey the rule that all blocks having the same symbol must also have the same contents. They may only have different parameters.

3.3 XILINX

Xilinx ISE (Integrated Software Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

It provides synthesis and programming for a limited number of Xilinx

devices. In particular, devices with a large number of I/O pins and large gate matrices are disabled. The low-cost Spartan family of FPGAs is fully supported by this edition, as well as the family of CPLDs.The main design toolkit Xilinx provides engineers is the Vivado Design Suite, and integrated design environment (IDE) with a system-to-IC level tools built on a shared scalable data model and a common debug environment. Vivado includes electronic system level (ESL) design tools for synthesizing and verifying C-based algorithmic IP; standards based packaging of both algorithmic and RTL IP for reuse; standards based IP stitching and systems integration of all types of system building blocks; and the verification of blocks and systems.

3.4 MODELSIM

ModelSim PE, our entry-level simulator, offers VHDL, Verilog, or mixed-language simulation. Coupled with the most popular HDL debugging capabilities in the industry, ModelSim PE is known for delivering high performance, ease of use, and outstanding product support. Model Technology's award-winning Single Kernel Simulation (SKS) technology enables transparent mixing of VHDL and Verilog in one design. ModelSim's architecture allows platform independent compile with the outstanding performance of native compiled code. An easy-to-use graphical user interface enables you to quickly identify and debug problems, aided by dynamically updated windows. For example, selecting a design region in the Structure window automatically updates the Source, Signals, Process, and Variables windows. These cross linked

ModelSim windows create a powerful easy-to-use debug environment. Once a problem is found, you can edit, recompile, and re- simulate without leaving the simulator. ModelSim PE fully supports the VHDL and Verilog language standards. You can simulate behavioral, RTL, and gate-level code separately or simultaneously. ModelSim PE also supports all ASIC and FPGA libraries, ensuring accurate timing simulations. ModelSim PE provides partial support for VHDL 2008

CHAPTER 4

DESIGN METHODOLOGY FOR IMPLEMENTATION ON FPGA

4.1 SPECIFICATION

This is the stage at which we define what are the important parameters of the system/design that you are planning to design. A simple example would be: I want to design a counter; it should be 4 bit wide, should have synchronous reset, with active high enable; when reset is active, counter output should go to "0".

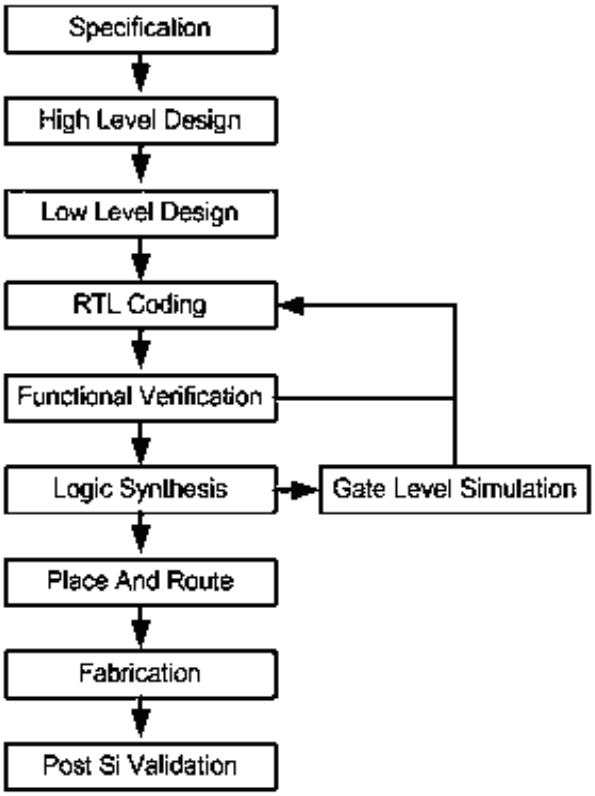


Fig. 4.1 Design flow.

4.2 HIGH LEVEL DESIGN

This is the stage at which you define various blocks in the design and how they communicate. Let's assume that we need to design a microprocessor: high level design means splitting the design into blocks based on their function; in our case the blocks are registers, ALU, Instruction Decode, Memory Interface, etc.

4.3 MICRO DESIGN/LOW LEVEL DESIGN

Low level design or Micro design is the phase in which the designer describes how each block is implemented. It contains details of State machines, counters, Mux, decoders, internal registers. It is always a good idea to draw waveforms at various interfaces. This is the phase where one spends lot of time.

4.4 RTL CODING

In RTL coding, Micro design is converted into Verilog/VHDL code, using synthesizable constructs of the language. Normally we like to lint the code, before starting verification or synthesis

4.5 SIMULATION

Simulation is the process of verifying the functional characteristics of models at any level of abstraction. We use simulators to simulate the Hardware models. To test if the RTL code meets the functional requirements of the specification, we must see if all the RTL blocks are functionally correct. To achieve this we need to write a testbench, which generates clk, reset and the required test vectors. We use the waveform output from the simulator to see if the DUT (Device Under Test) is functionally correct.

4.6 SYNTHESIS

Synthesis is the process in which synthesis tools like design compiler or Synplify take RTL in Verilog or VHDL, target technology, and constrains as input and maps the RTL to target technology primitives. Synthesis tool, after mapping the RTL to gates, also do the minimal amount of timing analysis to see if the mapped design is meeting the timing requirements. (Important thing to note is, synthesis tools are not aware of wire delays, they only know of gate delays).

- Formal Verification: Check if the RTL to gate mapping is correct.
- Scan insertion: Insert the scan chain in the case of A.

4.7 PLACE & ROUTE

The gate level netlist from the synthesis tool is taken and imported into place and route tool in Verilog netlist format. All the gates and flip-flops are placed; clock tree synthesis and reset is routed. After this each block is routed. The P&R tool output is a GDS file, used by foundry for fabricating the ASIC.

4.8 GATE LEVEL SIMULATION (OR) SDF/TIMING SIMULATION

There is another kind of simulation, called timing simulation, which is done after synthesis or after P&R (Place and Route). Here we include the gate delays and wire delays and see if DUT works at rated clock speed.

4.9 POST SILICON VALIDATION

Once the chip (silicon) is back from fab, it needs to put in real environment and tested before it can be released into Market. Since the speed of simulation with RTL is very slow (number clocks per second), there is always possibility to find a bug in Post silicon validation.

CHAPTER 5

CONVERSION FROM SIMULINK MODEL TO HDL CODER

5.1 Design of Simulink Models with Blocks Supported by Simulink HDL Coder

Some MATLAB-Simulink blocks, especially those which contain complex functions like encoders/decoders, modulators/demodulators etc. cannot be converted to VHDL code. To solve this problem, these blocks are redesigned using basic components such that they could be converted to VHDL code. Each branch supports a different type of modulation scheme, while the coding scheme used is convolutional code. The control circuit can be used to decide which transceiver is on and which is off when the input 1 of the control circuit is 0, the lower branch will turn on, while the upper branch will turn off.

The opposite happens when input 1 is decided as logic one. The modulators/demodulators have been designed using embedded MATLAB functions (mfiles) while other blocks are designed by MATLAB-Simulink blocks
supported by the Simulink HDL coder. For example, Fig. 5 shows the implementation of convolutional

5.2 SIMULINK TO VHDL CONVERSION TOOLBOX OVERVIEW

The conversion routine is to be considered as part of a larger design flow and development system. An overview of the Simulink to VHDL conversion process and subsequent model use is shown in Fig 5.1. Here, the main steps from initial modelling through to design data output for design realisation are shown

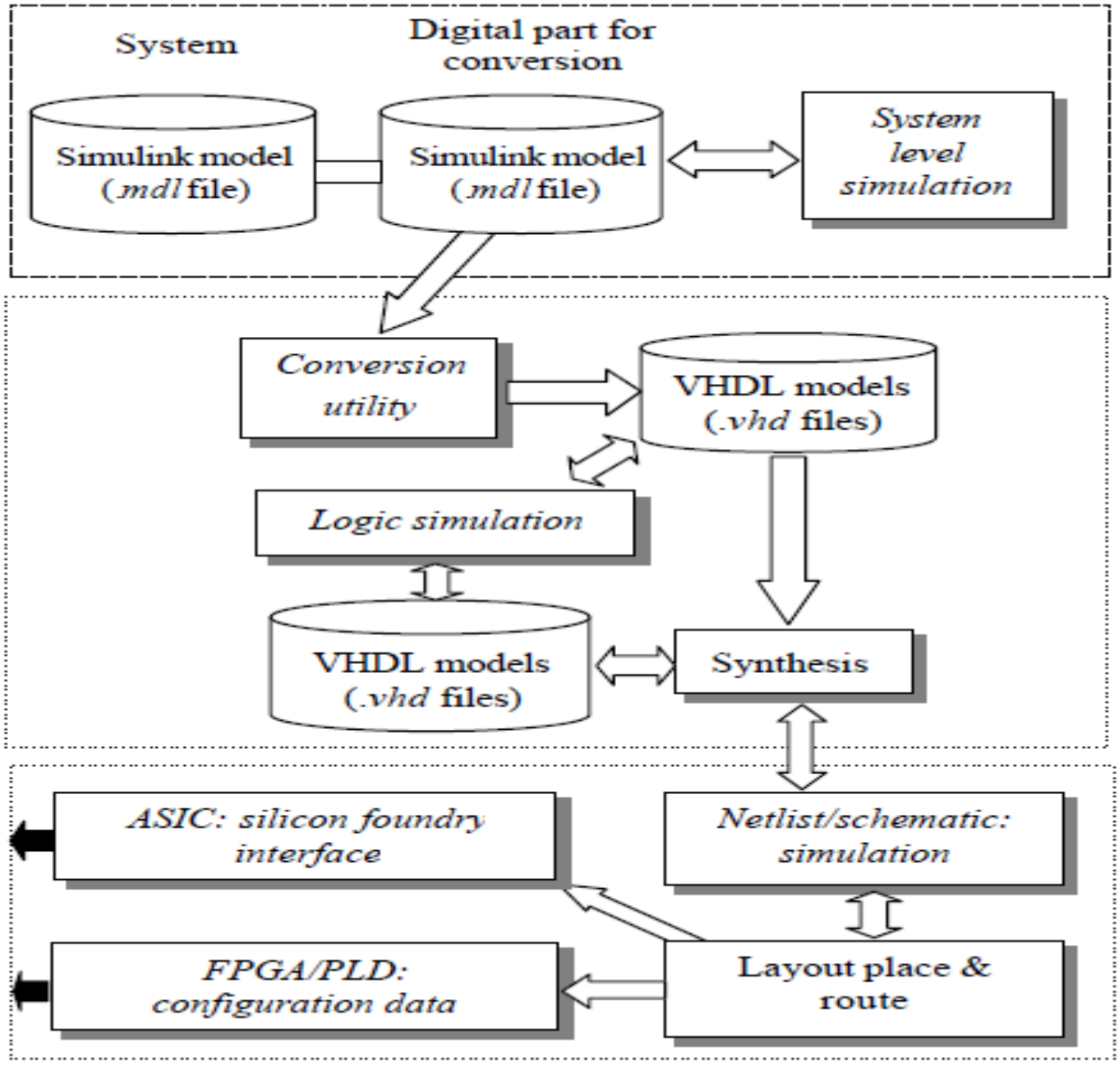


Fig. 5.1 Conversion routine within a design process

The three main steps involved are (i) initial system modeling and simulation, (ii) data conversion and synthesis with technology targeting, and (iii) post synthesis hardware implementation

5.3 Generation of VHDL Code for MATLAB-Simulink Models

The algorithms and designs used to define systems are normally modeled using high level software languages like MATLAB, MATLAB Simulink or C. Their disadvantage is that they normally cannot be used to synthesize real hardware. The Simulink HDL coder is a new tool, which comes with the MATLAB Simulink software package and can be used to generate hardware description language (HDL)

code based on Simulink models and State flow finite-state machines. The coder brings the Model-Based Design approach into the domain of application-specific integrated circuit (ASIC) and field programmable gate array (FPGA) development. Using the coder, system architects and designers can spend more time on fine-tuning algorithms and models through rapid prototyping and experimentation and less time on HDL coding. The Simulink HDL coder compatibility checker utility can be run to examine MATLAB-Simulink model semantics and blocks for HDL code generation compatibility. Then the coder can be invoked, using either the command line or the graphical user interface. The coder generates VHDL or Verilog code that implements the design embodied in the model.

Usually, a corresponding test bench also can be generated. The test bench with HDL simulation tools can be used to drive the generated HDL code and evaluate its behavior. The coder generates scripts that automate the process of compiling and simulating the code in these tools. EDA Simulator Link™ MQ, EDA Simulator Link IN or EDA Simulator Link DS software can be used from the MathWorks™ to cosimulate generated HDL entities within a Simulink model. In this work, the EDA Simulator Link™ MQ is used. Another easy possibility whould be to invoke the ModelSim manually.

The test bench feature increases confidence in the correctness of the generated code and saves time on test bench implementation. The design and test process is fully iterative. At any point, the designer can return to the original model, make modifications, and regenerate code. When the design and test phases of the project have been completed successfully, the generated HDL code can be exported easily to synthesis and layout tools for hardware realization. The coder generates synthesis scripts for the Synplify® family of synthesis tools.

CHAPTER 6

FILTER BANKS FOR SDR CHANNELIZERS

6.1 INTRODUCTION

DDC is generally composed of a numerical control oscillator(NCO) and a sampling rate conversion (SRC) filter. It is mainly used for single-channel channelization, but it is unsuited to multichannel channelization because of its low resource utilization. FFT-based channelization has a simple structure and high resource utilization, but its filtering performance is poor, which can be improved using the windowing function method. A polyphase DFT filter bank has high resource utilization and better filtering performance, but it only realizes a uniform channel division. A Goertzel filter bank can provide a solution to a fixed center frequency problem associated with a polyphase DFT filter bank, but it cannot extract channels with nonuniform bandwidths.

According to the tree-structured filter bank approach, RFEL Ltd. has developed two methods for digital channelization, called pipelined frequency transform (PFT) and tunable PFT (TPFT). The problem of power-of-two channel stacking associated with PFT can be solved by the TPFT technique. TPFT can accurately locate the radio signals, thereby ensuring the correct reception. It consists of a coarse channelization in the PFT stage and a fine channelization using a combination of complex DDC and digital up conversion. However, its implementation complexity is much more than that of PFT. The frequency response masking and coefficient decimation techniques are used to implement digital channelization.

The advantages of these two techniques are low computational complexity and nonuniform channel division. For an analysis/synthesis filter bank approach, it uses the analysis filter bank to implement uniform channel division, and then uses the synthesis filter bank to merge the adjacent channels for realizing nonuniform channel division [9]. However, the channel bandwidths generated by the synthesis filter bank are integer multiples of those generated by the analysis filter bank. Such constraint limits the flexibility of the analysis/synthesis filter bank. The reason is

that the channel bandwidths of different communication standards are not related by an integer factor

6.2. DFT Filter Banks

DFT filter bank is a uniformly modulated filter bank, which has been developed as an efficient substitute for PC approach when the number of channels need to be extracted is more, and the channels are of uniform bandwidth (for example many single-standard communication channels need to be extracted). The main advantage of DFT filter bank is that, it can efficiently utilize the polyphase decomposition of filters. However, DFTFBs have following limitations for multi-standard receiver applications:

- DFTFBs cannot extract channels with different bandwidths. This is because DFTFBs are modulated filter banks with equal bandwidth of all bandpass filters. Therefore, for multi-standard receivers, distinct DFTFBs are required for each standard. Hence the complexity of a DFTFB increases linearly with the number of received standards.

- Due to fixed channel stacking, the channels must be properly located for selecting them with the DFTFB. The channel stacking of a particular standard depends on the sample rate and the DFT size. To use the same DFTFB for another standard, the sample rate at the input of the DFTFB must be adapted accordingly. This requires additional sample rate converters (SRCs), which would increase the complexity and cost of DFTFBs.

- If the channel bandwidth is very small compared to wideband input signal the prototype filter must be highly selective resulting in very high-order filter. As the order of the filter increases, the complexity increases linearly. Ideally, the reconfigurability of the filter bank must be accomplished by reconfiguring the same prototype filter in the filter bank to process the signals of the new communication standard with the least possible overhead, instead of employing separate filter banks for each standard. However, reconfiguration of DFTFB suffers from following overheads:

- The prototype filter needs to be reconfigured. Generally, DFTFB employs the polyphase decomposition. Hence reconfiguration can involve changing the number of polyphase branches which is a tedious and expensive task.
- Down sampling factor needs to be changed. If down sampling is to be done after filtering, then we need separate digital down converters. This will add more cost.
- The DFT needs to be reformulated accordingly which is also expensive.

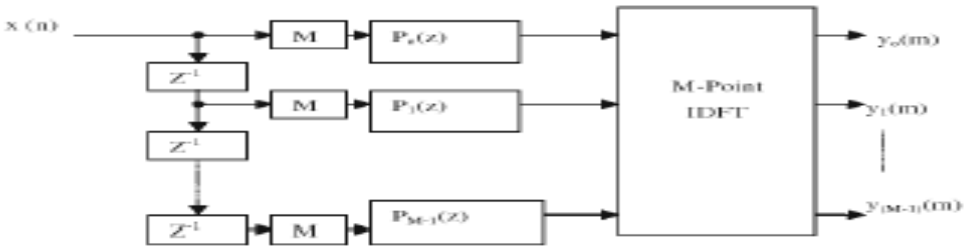


Fig 6.1 DFT Filter Bank

In the case of multiple channel extraction of single standard signal i. e., extraction of many channels of identical bandwidth, the complexity of PC approach is given by N.L.fs, where N is the number of channels extracted, L is the total length of filters employed in all the branches for PC approach and fs is the sampling frequency. The complexity of DFTFB is only L.fs which is N times lower than PC approach. But in the case of SDR, multiple channels of multiple standards need to be extracted.

In that case, the complexity of PC approach and DFTFB are NCSs and NSs respectively, where NC and NS are the number of channels and number of standards respectively. Thus the complexity of these channelizes can be reduced further if the length of filter, L, can be reduced and the same filter bank can be reconfigured to the new standard. Thus there is a need for developing new filter bank architectures for SDR receivers

Goertzel filter bank (GFB)

A Goertzel filter bank (GFB) based on modified Goertzel algorithm was proposed in as a substitute to DFTFB. In GFB, the DFT is replaced by a modified Goertzel algorithm which performs the modulation of the prototype low-pass frequency response to any centre frequency which is not possible using DFT. This will eliminate the limitation of fixed channel stacking associated with DFTFBs. This paper is an attempt towards design of FBs with reduced area complexity and improved reconfigurability when compared with conventional resampling-based FBs The frequency response masking (FRM) technique was originally proposed for designing low complexity sharp transition-band finite impulse response (FIR) filters.

A reconfigurable low complexity channel filter for SDR receivers based on the FRM technique The objective of the work in was to realize a channel filter to extract a single channel (frequency band) from the wideband input signal. New low complexity reconfigurable architecture for the modal filter which can simultaneously extract multiple channels is also presented in this paper.

The modal filter architecture can generate simultaneous frequency responses employing different delay line adders. Coefficient multiplication, which is the most power consuming operation, is done only once for obtaining different frequency responses simultaneously in the proposed modal filter architecture. Consequently, the proposed FB offers low multiplier complexity when compared with other FBs in the literature. The proposed FB has been compared with the conventional PC approach, DFTFB, GFB.

CHAPTER 7

DESIGN OF CHANNELIZATION ARCHITECTURE

7.1 METHODOLOGY

In this architecture the three narrow-band signals A, B, and C is extracted from the input signal. Their parameters, such as center frequency, output sampling rate, and bandwidth, are different. Fig. 7.2 shows the high-level architecture of the channelization solution.

TPFT is used to implement the coarse channelization block. The spectrum of the input signal is separated into multiple bands is shown in Fig. 7.1(a). The bands of each TPFT output stage are half as wide as those of the previous stage. The target signals A, B, and C are contained in band 4_L_L of the third stage, band 2 of the first stage, and band1_L of the second stage is shown in see Fig. 7.1(b).

The tuning unit moves the target signals from their original frequency locations to the zero frequency location is shown in Fig.7.1(c). Complex multiplications and multichannel NCOs are used to construct the tuning unit.

According to the channel parameters, the target signals are further processed by the resampling filter, which is composed of multichannel Farrow-based variable fractional delay (VFD) filters, multichannel 2/4-phase filters, and multichannel gain. The operations performed in the resampling filter include SRC, filtering, and gain control is shown in Fig. 7.1(d).

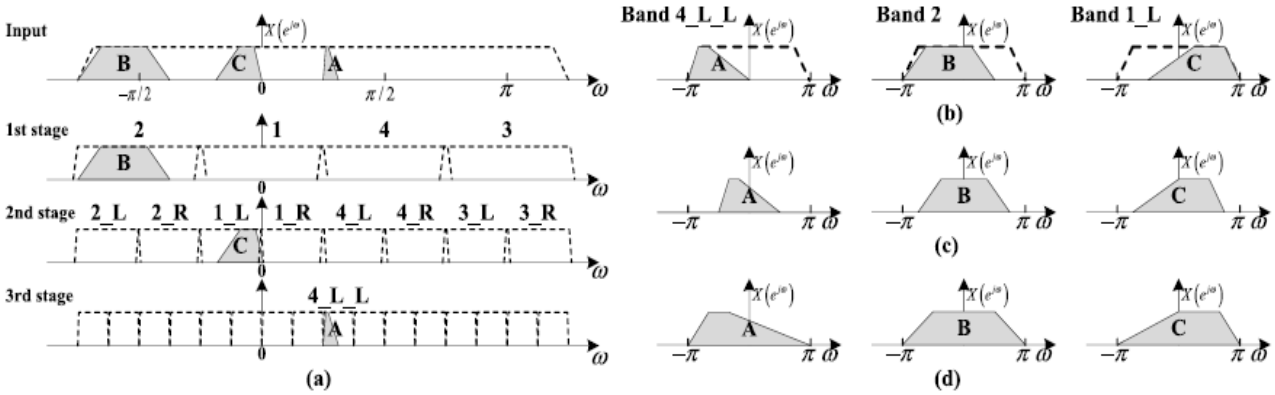


Fig. 7.1 Channel division of wideband complex input signal (a) input signal. (b) coarse channelization. (c)Tuning. (d)Resampling filter.

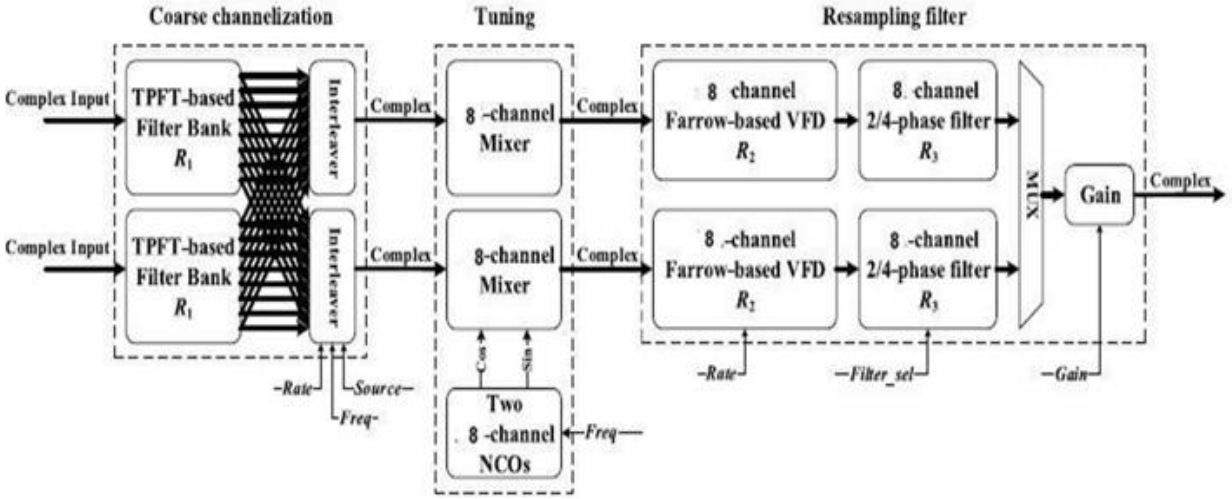


Fig. 7.2 Channelization architecture.

The runtime configurable feature of the tuning unit, the coarse channelization block, and the resampling filter ensures that the architecture has enough flexibility.

Channelization typically works at the ADC sampling clock frequency, while the subsequent baseband processors often operate at the symbol rate. Different communication standards typically have different symbol rates. Thus, SRC is crucial to multistandard channelization.

7.2 SRC FACTORIZATION

A SRC factorization scheme is proposed to assign SRC factors to the coarse channelization block and the resampling filter. In the current design, the SRC factor R is defined as the following equation:

$$R = F_S/F_{OUT} = R1 \times R2 \times R3 \quad (1)$$

where F_{OUT} is the output sampling rate, R1 is the decimation factor of the TPFT-based filter bank, R2 is the decimation factor of the Farrow-based VFD filter and it is in the range (1, 2), and R3 is the decimation factor of the 2/4-phase filter and it is 2 or 4.
Let Rate be the sampling rate control word. Rate is calculated as follows:

$$\text{Rate} = \text{Round}(2^{32} \times F_{OUT}/F_S) = \text{Round}(2^{32}/\text{R1} \times \text{R2} \times \text{R3}) \quad (2)$$

where Round(x) rounds the element x to the nearest integer. According to (2), the resolution of the output sampling rate is F_S / (232) (=0.0596 Hz for F_S = 256 MS/s). When R is in the range [8, 4096), Rate satisfies the following equation:

$$2^{20} < \text{Rate} \leq 2^{29} \quad (3)$$

7.3 COARSE CHANNELIZATION BLOCK

The coarse channelization block consists of two TPFT blocks and two interleavers (see Fig. 7.2). For all successive stages in the TPFT block, the output signal is decimated by two. Thus, each TPFT block is composed of two cascaded processing elements (PEs), and is used to process one channel of complex input data. The coarse channelization block is scalable in terms of the number of input signals, making it easily adapted for multiple applications

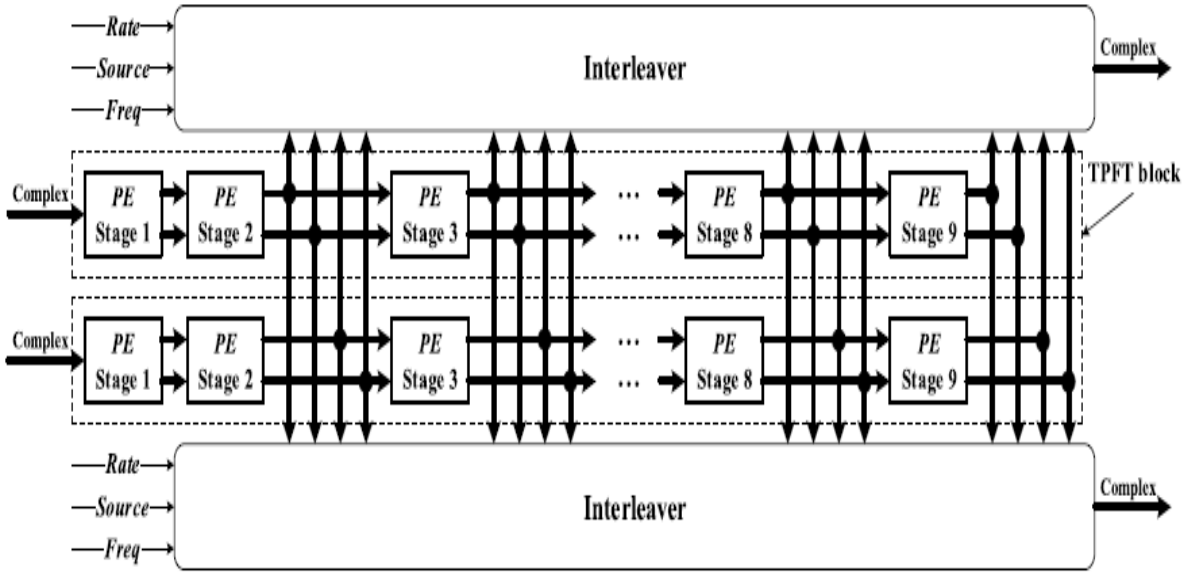


Fig. 7.3. Structure of the TPFT-based channelization block.

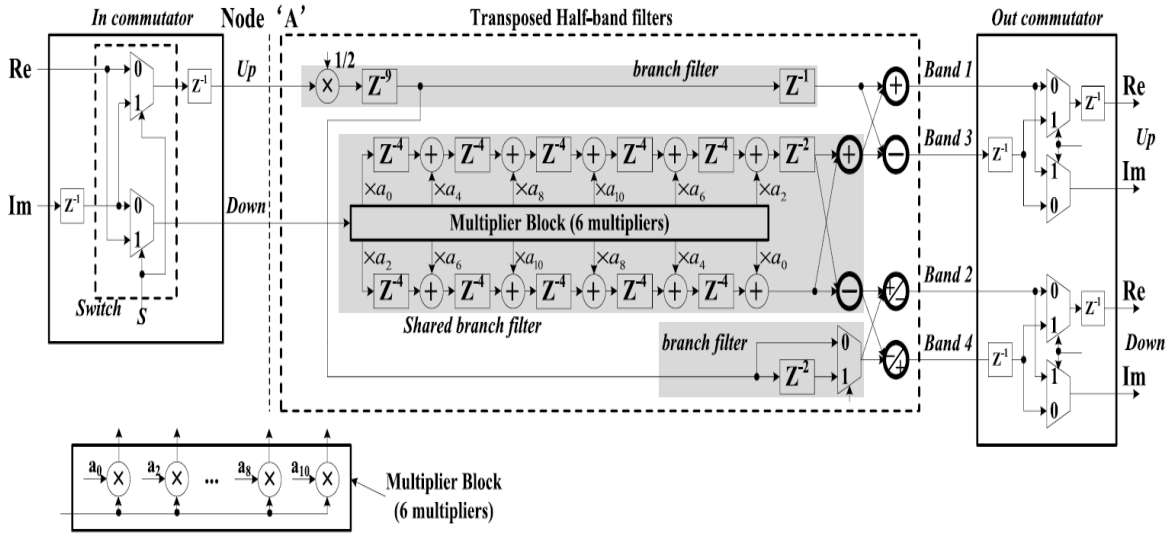


Fig.7.4. Structure of PE 1.

7.4 INTERLEAVER

The output results of PEs are available because of the cascaded architecture. Hence, it is possible to extract frequency bands with different bandwidths and center frequency. Let Frequency be the center frequency control word. Freq is calculated as follows:

$$\text{Freq} = \text{Round}(2^{32} \times FcFs) \quad (1)$$

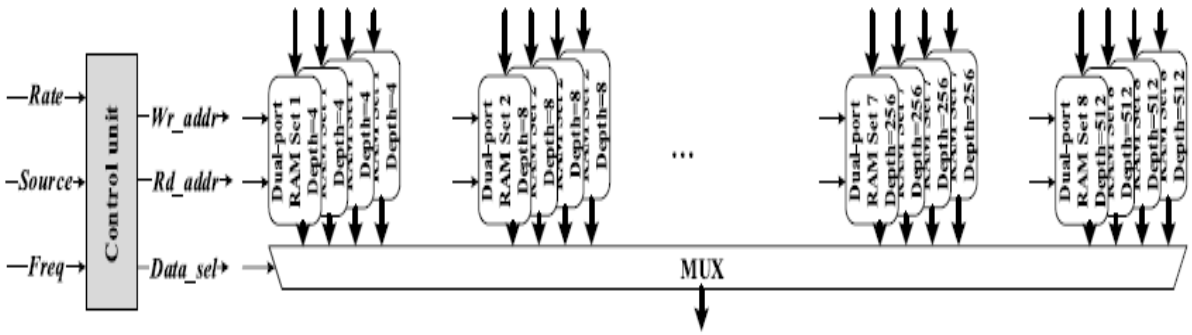


Fig. 7.5. Structure of interleaver

where *Fc* is the center frequency of the target signal. When *Fs* is 256 MS/s, the center frequency resolution is 0.0596 Hz. The coarse channelization block includes two interleavers, each of which can generate up to 8 channels of complex data. The structure of interleaver is shown in Fig. 7.3 It is composed of 32 dual-port RAMs, 1 multiplexer, and 1 control unit. The output results of PEs 2, 3,*4,5,6*... and 9 are written into dual-port RAM Sets 1, 2,*3,4,5*...., and 8, respectively. The depth of each dual-port RAM is equal to one half of the number of the output of a PE stage. For example, PE 2 produces 8 bands, i.e., bands 1_L, 2_L, 3_L, 4_L ,1_R ,2_R, 3_R, and 4_R. They are inserted into two dual-port RAMs, and the depth of dual-port RAM is 4. According to the Source, control word Rate and Freq of each channel, the control unit generates the matching read addresses for the 32 dual port RAMs and performs a selection among them.

7.5 TUNING UNIT

The tuning unit is composed of two parallel 8-channel mixers (see Fig. 2). Each mixer consists of one 8-channel NCO and one complex multiplication.

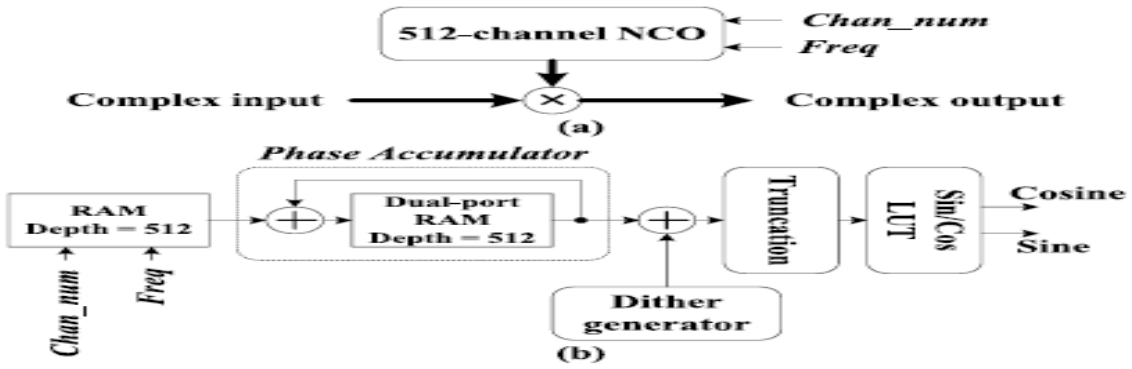


Fig. 7.6 Tuning unit (a) Structure of mixer. (b) Structure of 512-channel phase dithered NCO

The frequency control word Freq is used by 8-channel NCO to generate up to 8 independent tuning signals. Tuning signals must be generated based on the arbiter table of the interleaver. It must be consistent with the output order of a TPFT-based coarse channelization block. The complex multiplication utilizes the tuning signals to move the

target signals from their original frequency location to the zero frequency location.

7.6 RESAMPLING FILTER

Resampling filter composed of two filters VFD filters and 2/4phase filter

7.6.1 VFD FILTER

A fractional delay filter is a device for bandlimited interpolation between samples. It finds applications in numerous fields of signal processing, including communications, array processing, speech processing, and music technology. In this article, we present a comprehensive review of FIR and all pass filter design techniques for bandlimited approximation of a fractional digital delay. Emphasis is on simple and efficient methods that are well suited for fast coefficient update or continuous control of the delay value.

The first stage of the resampling filter consists of two parallel Farrow-based VFD filters (see Fig. 1). A VFD filter is used to implement arbitrary SRC. The SRC factor *R*2 is in the range (1, 2). The maximum input sampling rate of each VFD filter is *Fs* /4 (decimation by *R*1 in the coarse channelization, *R*1 ≥4), and the corresponding antialiasing output bandwidth is *Fs* /20. Thus, the frequency response of a VFD filter is as follows:

$$H\text{VFD}(e^{j\omega}) = \begin{cases} e^{j\omega}(D+\varphi), & 0 \le |\omega| \le 0.2 \\ 0, & 0.8\pi \le |\omega| \le \pi \end{cases} \quad (1)$$

where $D + \varphi$ ($\varphi \in [0, 1]$), 0.2π, and 0.8π are the group delay, the passband and the stopband edges of the VFD filter, respectively. The transfer function of the VFD filter can be expressed by the following equation:

$$HVFD(z, \varphi) = \sum_{l=0}^{L-1} \left(\sum_{n=0}^{N-1} \text{c1(n)} z^{-n} \right) \varphi l \quad (2)$$

where *L* is the order of a VFD filter, *N* is the length of a subfilter, and *cl (n)* is the coefficient of a subfilter. According to the method in [20] and [21], the order of a VFD

filter is 3, and the length of a subfilter is 16. The antialiasing stopband attenuation of the VFD filter can reach to 80 db. The VFD filter can be implemented using the Farrow structure. All the sub filters are implemented in transposed forms, so the input datas are multiplied with all the subfilter coefficients. The SOPOT coefficients and the MB technique can be used to minimize the total number of additions and multiplications. As a result, the number of additions will be reduced to 41 for the SOPOT coefficients. The architecture of a VFD filter is shown in Fig. 7(a). The real or imaginary parts of 512-channel complex input data can share the same VFD filter, so the storage units of subfilters are implemented by RAMs with depths of 512. In the current design, the fractional delay φ is expressed as follows:

$$\varphi n+1\_ = \{\varphi n - 1, \qquad \varphi n \geq 1$$
$$\varphi n + (R2 - 1) = \varphi n - 1 + (2r/\text{Rate}), \qquad \varphi n < 1\} \tag{3}$$

where Rate is in the range $(2r-1, 2r)$ and $\varphi 0$ is 0. When $\varphi n \geq 1$, the filtered result of a VFD filter is discarded.

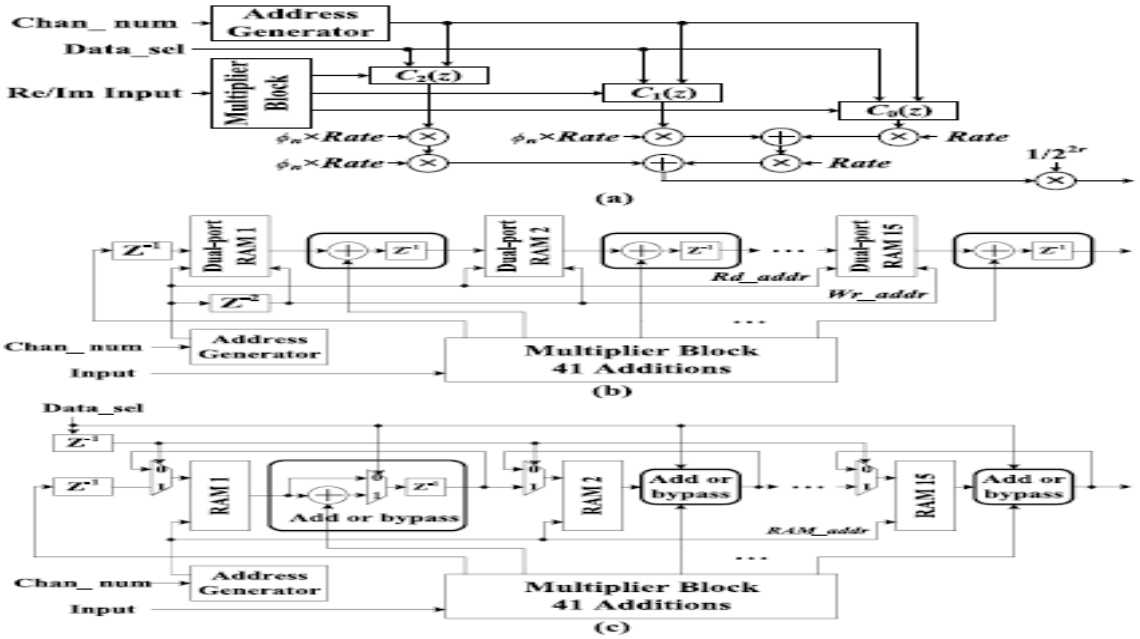


Fig. 7.7 VFD FILTER (a) Structure of VFD filter. (b) First structure of subfilter *C2(z)*. (c) Second structure of subfilter *C2(z)*.

The gain variation can be compensated by the gain module in the resampling filter. The subfilters of VFD filters can be implemented by two architectures [see Fig. 7(b) and (c)], respectively. The first one is operated as follows.

1) When a valid data *xm(n)* of channel *m* comes, the MB generates all the results, which are the products of *xm(n)* multiplied with all the subfilter coefficients *c*0(0), *c*0*(*1*)*, . . ., *c*0*(*15*)*, *c*1*(*0*)*, *c*1*(*1*)*, . . ., *c*1*(*15*)*, *c*2*(*0*)*, *c*2*(*1*)*, . . ., *c*2*(*15*)*.

2) The product *xm(n)* × *c*0*(i)* ($0 \leq i \leq 15$) is first added with the intermediate result read from the address *m* of the buffer *i*, and then, the sum is written into the same address *m* of buffer *i* + 1. The latencies caused by the buffer read operation and the addition, the buffer should be implemented using dual-port RAM.

7.6.2 2/4-PHASE FILTER

The second stage of the resampling filter consists of two parallel 2/4-phase filters. After decimation by (*R*1 × *R*2*)*, the minimum input sampling rate of each 2/4-phase filter is *Fs* /2*R*1 ($1 \leq R2 < 2$), and the maximum antialiasing output bandwidth is limited to 80% of the output sampling rate. The decimation factor *R*3 is 2 or 4. When *R*3 is 2, the passband and stopband edges are set to 0.4π and 0.6π, and the passband ripple and the stopband attenuation are set to 0.1 and 70 dB, respectively. According to FDATool software, the required filter order is 32, so each phase includes 16 filter coefficients. When *R*3 is 4, the passband and stopband edges are set to 0.2$\pi$and 0.3π. According to FDATool software, the required filter order is 64, so each phase still includes 16 coefficients. For *R*3 = 4, the storage units are implemented by RAMs of size 2048. The storage units for filter coefficients are implemented by 16 RAMs of size 4×8, so the proposed design can store eight predefinable filters. The filter for each channel may be independently selected using the filter selection control word *Filter_sel*, and the eight filters are all programmable at runtime. The architecture of a 2/4-phase filter is shown in Fig. 7.8

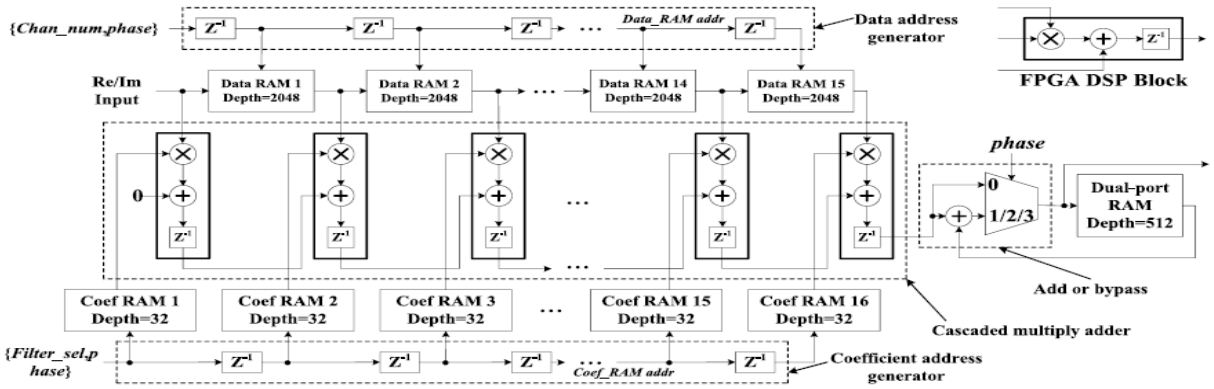


Fig.7.8 Structure of 2/4-phase filter.

It consists of one data address generator, one coefficient address generator, fifteen data RAMs, sixteen coefficient RAMs, one cascaded multiply adder, one add or bypass unit, and one dual-port RAM. To take use of the DSP block in the FPGA, the convolution is implemented using the cascaded multiply adder. The third stage of the resampling filter consists of one multiplexer and one gain module. The gain for each channel is configurable at runtime.

7.7 CHANNEL *RECONFIGURATION*

The parameters of each channel much be programmed. They are organized into register blocks, with 16 addresses for supporting 16 channels. The reconfiguration includes two phases: 1) the update of channel parameters and 2) the generation of arbiter table. Each channel takes one clock cycle to update its channel parameters, so the number of clock cycles updating all the channel parameters is equal to the total number of output channels. Once the update of channel parameters is completed, the control unit in the interleaver is responsible to generate an arbiter table. The control unit repeatedly performs the writing operations until the arbiter table is full or all of the channels are arranged. The size of the arbiter table is equal to the maximum value of $R1$ of all the channels. Therefore, the reconfiguration time is expressed as follows:

$$T = [Nc + \max(R1)]/Fs \tag{1}$$

where Nc is the total number of output channels and max($R1$) is the maximum value of $R1$ of all the channels, which directly determines the size of the arbiter table.

CHAPTER 8
DESIGN OF ARCHITECTURE USING SIMULINK

8.1 COARSE CHANNELIZATION BLOCK

The coarse channelization block consists of two TPFT blocks and two interleavers. For all successive stages in the TPFT block, the output signal is decimated by two. Thus, each TPFT block is composed of two cascaded processing elements (PEs), and is used to process one channel of complex input data. The coarse channelization block is scalable in terms of the number of input signals, making it easily adapted for multiple applications

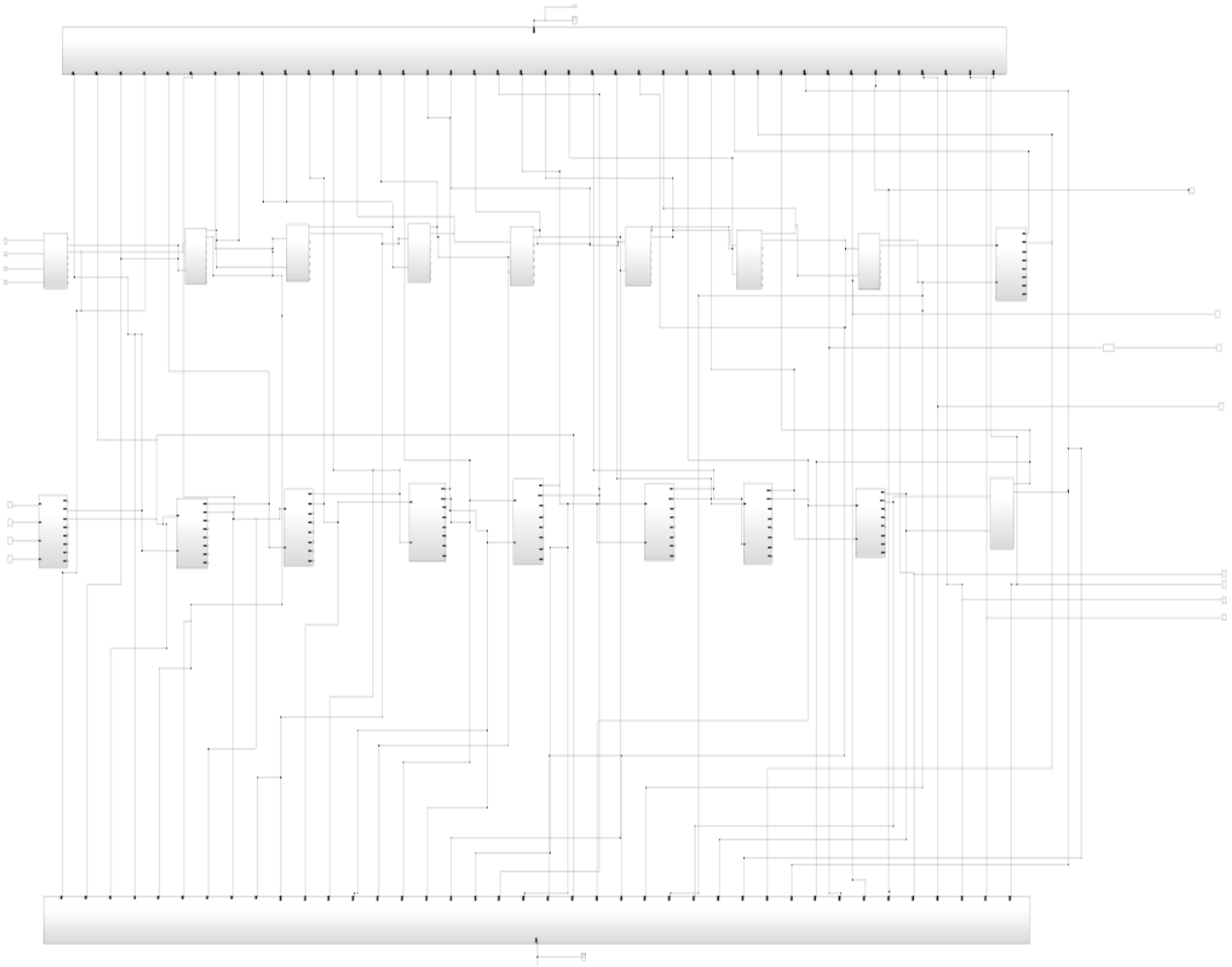


Fig. 8.1 Coarse Channelization Block

8.2 TUNING UNIT

The tuning unit moves the target signals from their original frequency location to the zero frequency location.

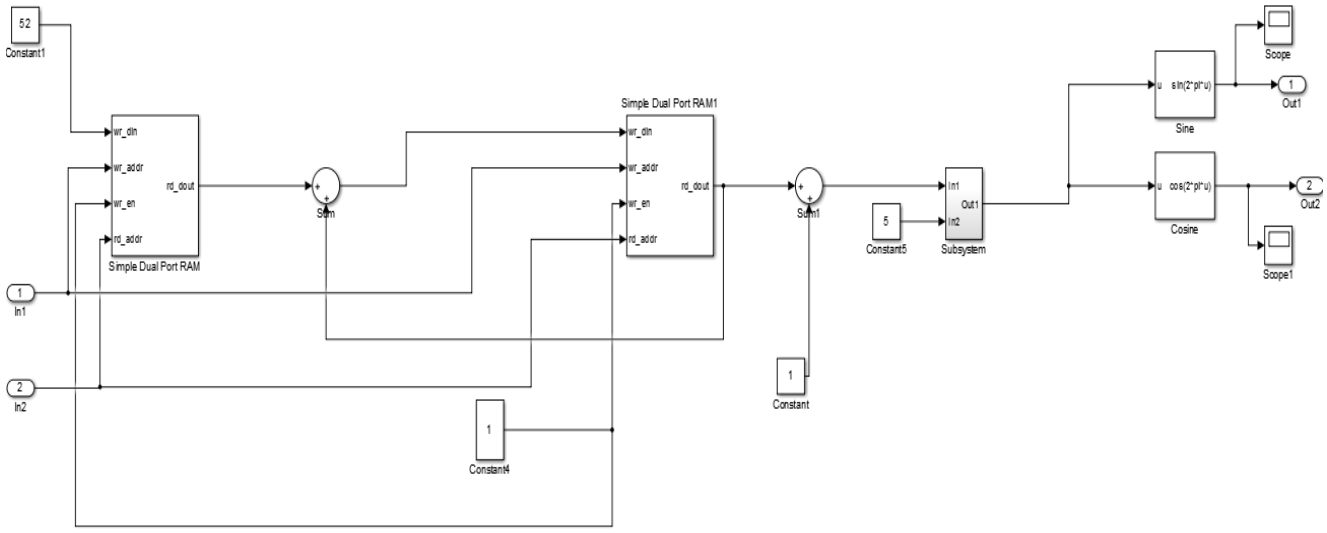


Fig. 8.2 Tuning Unit

8.3 VFD FILTER

A VFD filter is used to implement arbitrary SRC

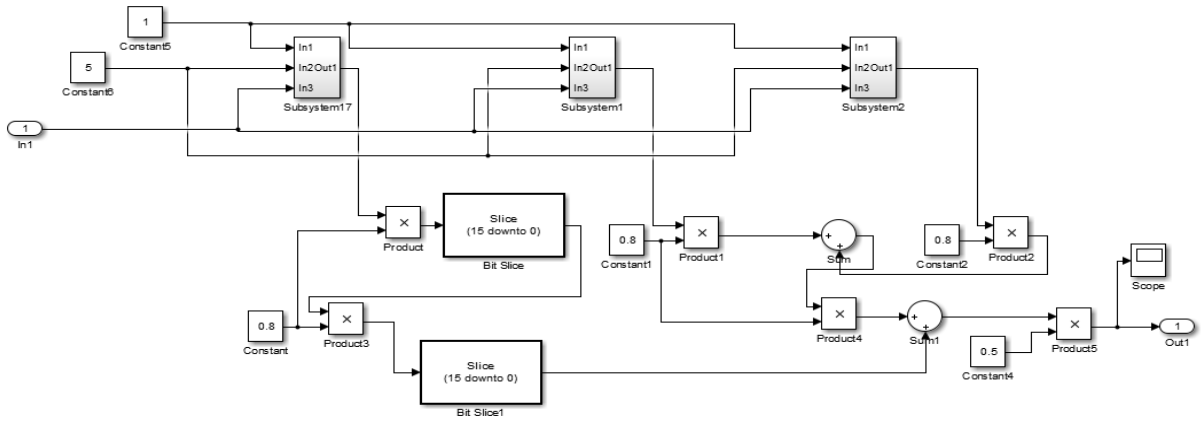


Fig. 8.3 VFD filter

8.4 2/4 PHASE FILTER

2/4 phase filter is used to implement the gain control of the architecture. After decimation by ($R1 \times R2$), the minimum input sampling rate of each 2/4-phase filter is $Fs/2R1$ ($1 \leq R2 < 2$), and the maximum antialiasing output bandwidth is limited to 80% of the output sampling rate. The decimation factor $R3$ is 2 or 4. When $R3$ is 2, the passband and stopband edges are set to 0.4π and 0.6π, and the passband ripple and the stopband attenuation are set to 0.1 and 70 dB, respectively.

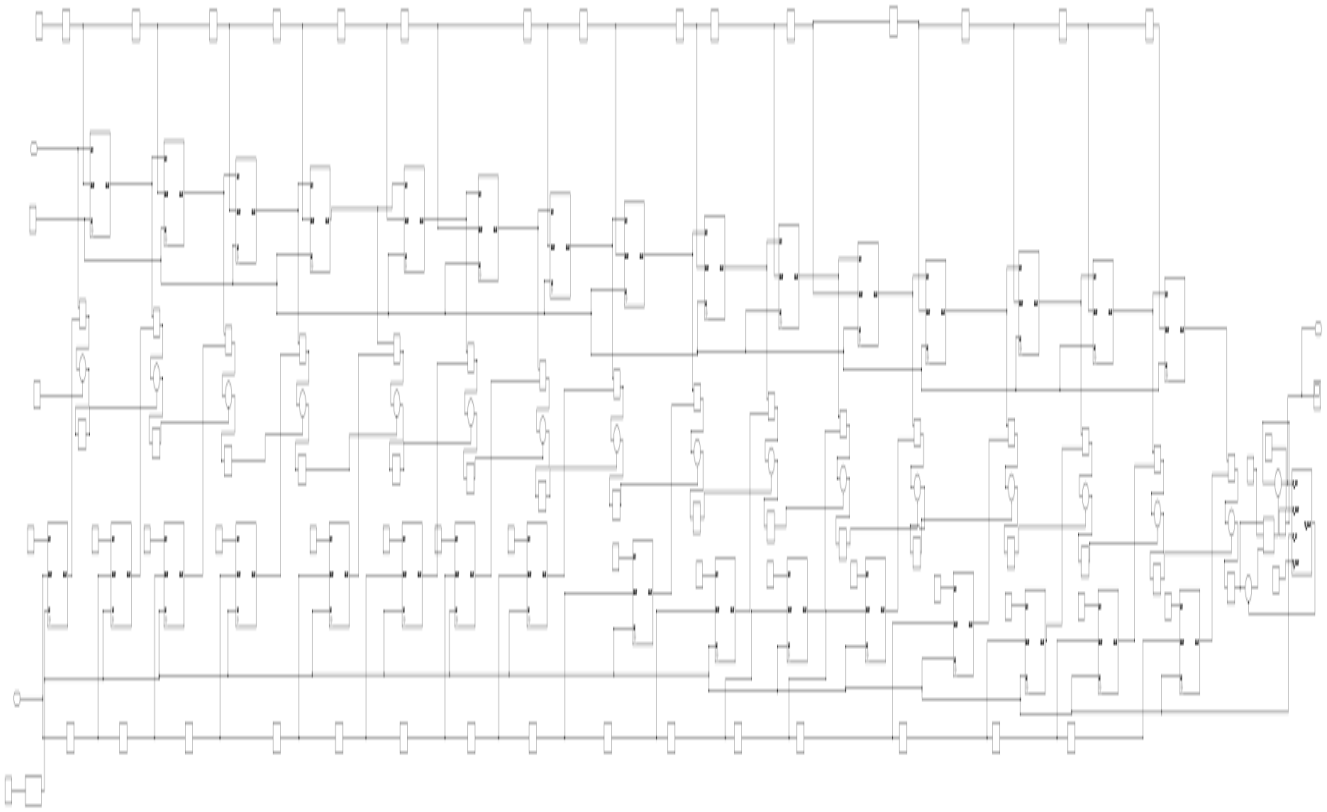


Fig. 8.4 2/4 Phase filter

8.1 THE CHANNELIZATION ARCHITECTURE

The Proposed channelization architecture consist of coarse channelization block, tuning unit and resampling filter.

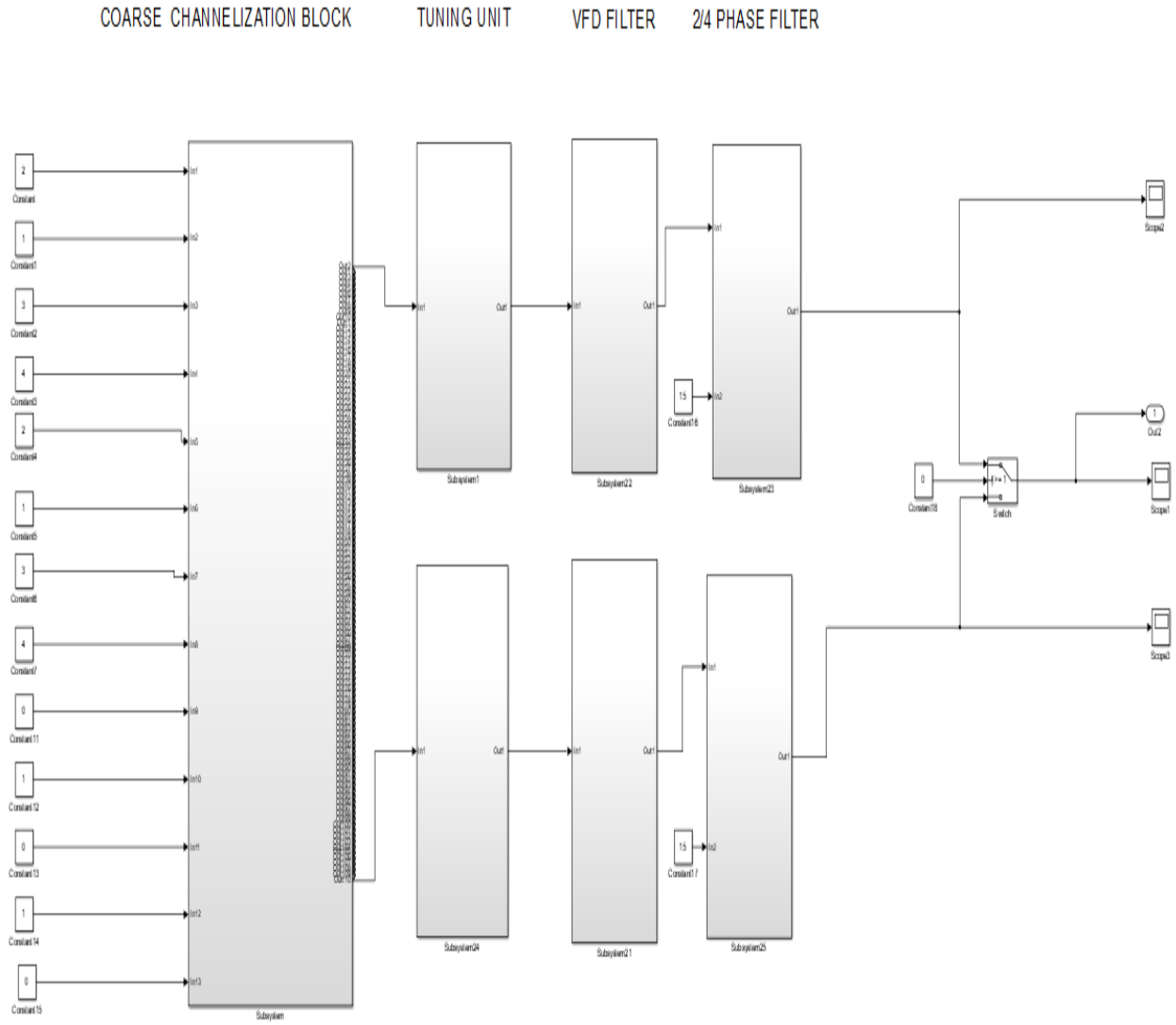


Fig. 8.5 The Channelization Architecture

CHAPTER 9
SIMULATION AND EXPERIMENTAL RESULTS

The proposed algorithm was implemented using VHDL in Model sim and is linked with Xilinx 14.2. A wideband signal is generated by MATLAB. It is then fed into the FPGA implementation. The proposed design can extract all the subbands from the input signal, and perform resampling for these subbands by setting the Freq and Rate parameters. The antialiasing stopband attenuation of all the filters in the datapath is larger than 70 dB. These filters include the transposed half-band filters in the TPFT blocks, the Farrow-based VFD filters, and the 2/4-phase filters in the resampling filter.

9.1 SIMULINK RESULT

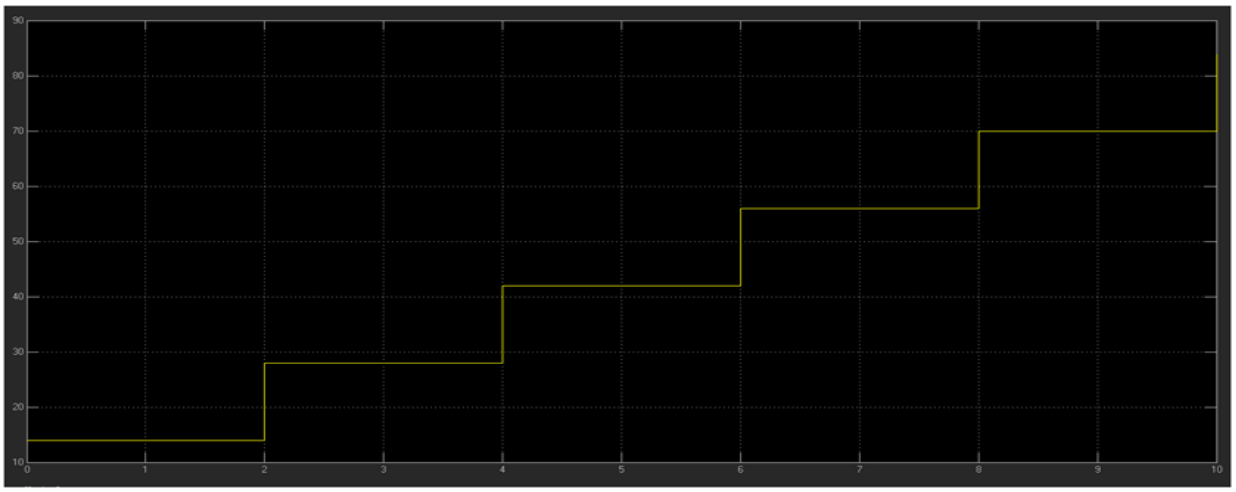


Fig. 9.1 Extracted Channel Output

A wideband signal generated using MATLAB Simulink is shown in Fig 9.1. The proposed design can extract all the subbands from the input signal, and perform resampling for these subbands by setting the Freq and Rate parameters.

9.2 XILINX RESULTS

The proposed channelization architecture can be implement on the Xilinx Virtex-6 FPGA XCV6SX315T-2ff1156, and the corresponding hardware resource is summarized in Table I. Results are obtained from Xilinx ISE 14.2 after place and route analysis. The contribution of each component of the proposed architecture is also shown in Table I.

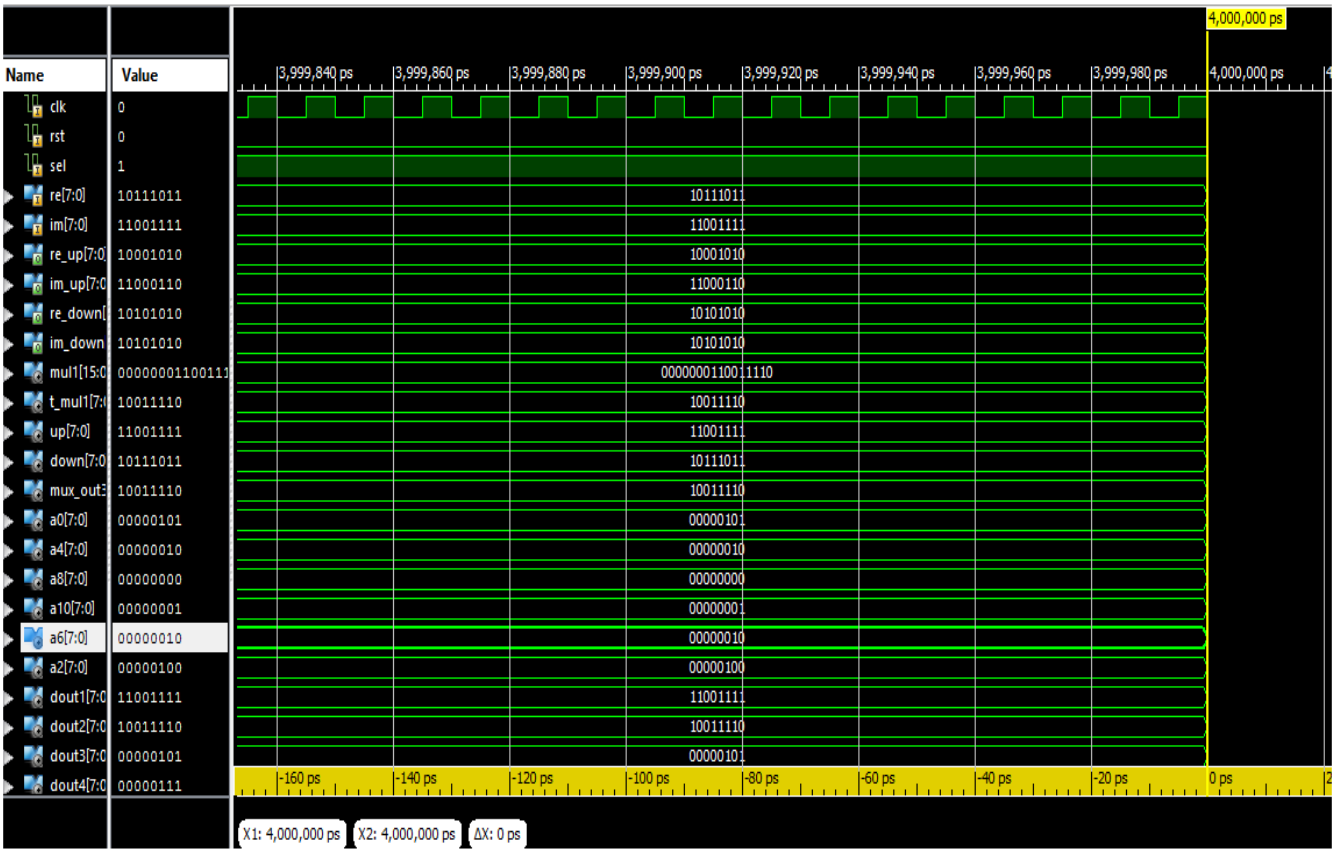
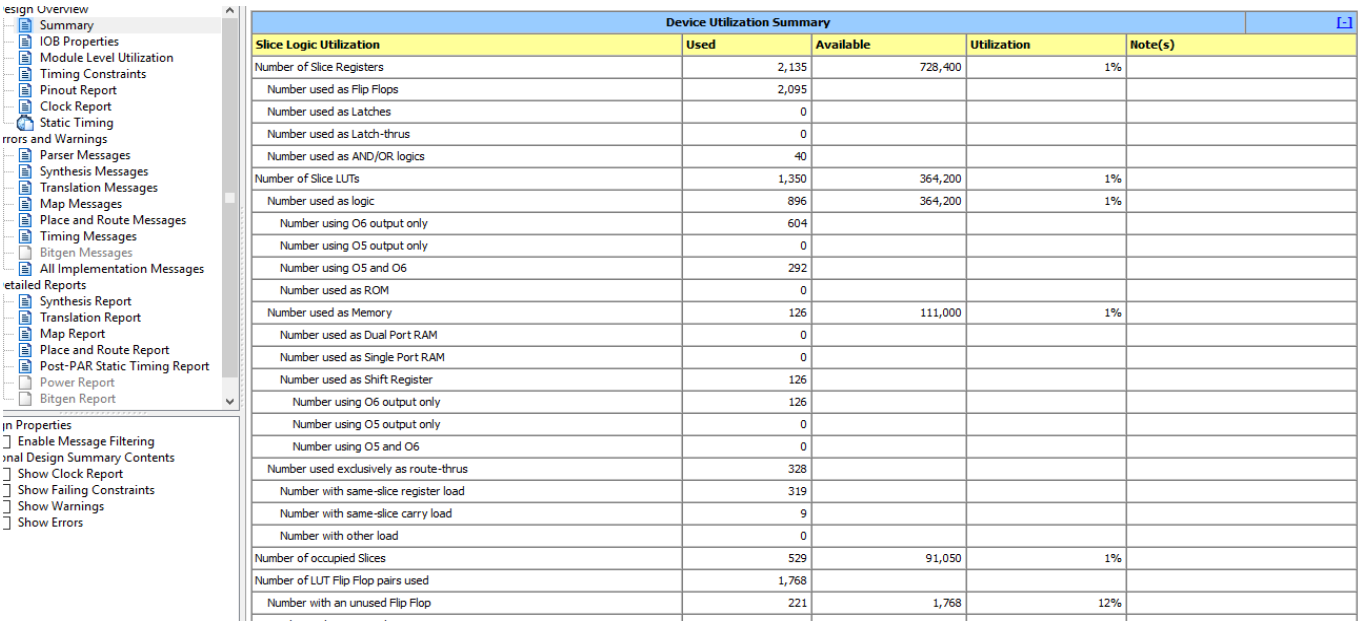


Fig. 9.2 Simulation result

The input bitwidth is 8 bit, and the bitwidths of the TPFT-based filter banks, the tuning unit, the VFD filters, and the 2/4-phase filters are all 8 bit. Simulation result 9.2 shows that input bitwidth is 8 bit applied to benchmark circuit for the architecture and the resulting test generation will also be 8 bit.

9.3 SYNTHESIZE REPORTS

COARSE CHANNELIZATION

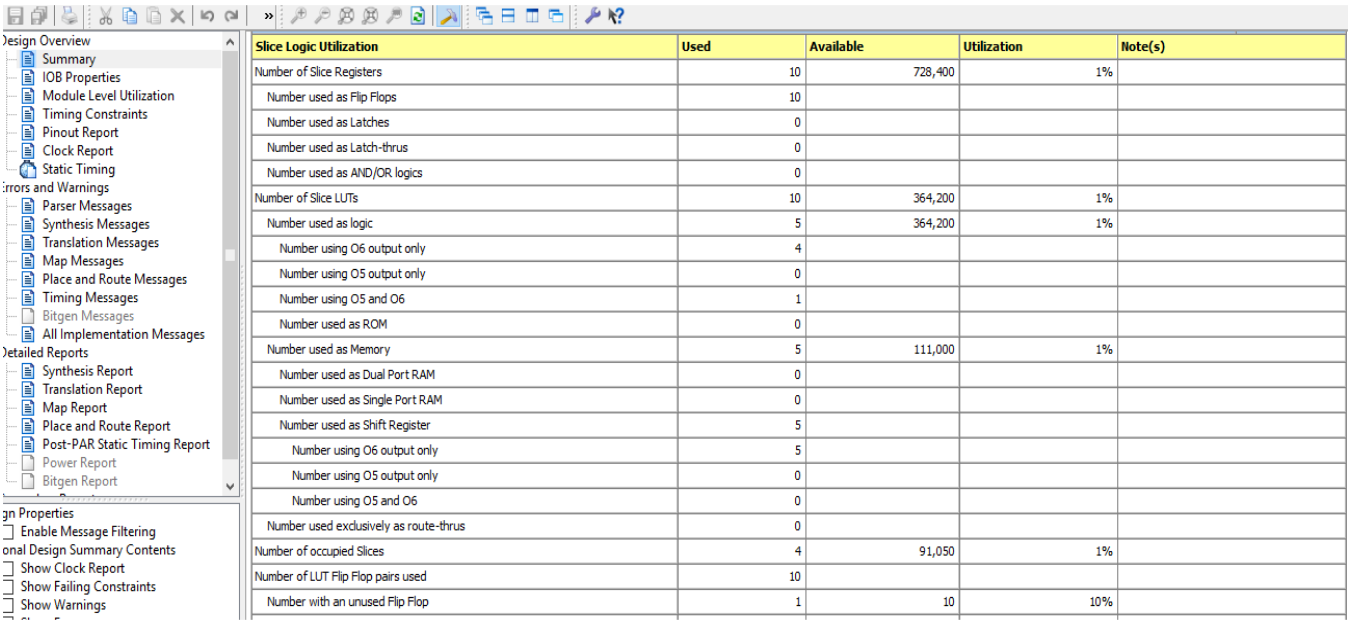


Device Utilization Summary

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	2,135	728,400	1%	
Number used as Flip Flops	2,095			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	40			
Number of Slice LUTs	1,350	364,200	1%	
Number used as logic	896	364,200	1%	
Number using O6 output only	604			
Number using O5 output only	0			
Number using O5 and O6	292			
Number used as ROM	0			
Number used as Memory	126	111,000	1%	
Number used as Dual Port RAM	0			
Number used as Single Port RAM	0			
Number used as Shift Register	126			
Number using O6 output only	126			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	328			
Number with same-slice register load	319			
Number with same-slice carry load	9			
Number with other load	0			
Number of occupied Slices	529	91,050	1%	
Number of LUT Flip Flop pairs used	1,768			
Number with an unused Flip Flop	221	1,768	12%	

The spectrum of the input signal is separated in to multiple subbands using the coarse channelization block. Synthesize report shows the hardware resource for the coarse channelization block

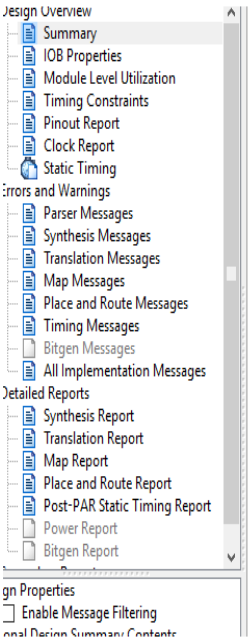
TUNING UNIT



Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	10	728,400	1%	
Number used as Flip Flops	10			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	10	364,200	1%	
Number used as logic	5	364,200	1%	
Number using O6 output only	4			
Number using O5 output only	0			
Number using O5 and O6	1			
Number used as ROM	0			
Number used as Memory	5	111,000	1%	
Number used as Dual Port RAM	0			
Number used as Single Port RAM	0			
Number used as Shift Register	5			
Number using O6 output only	5			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	0			
Number of occupied Slices	4	91,050	1%	
Number of LUT Flip Flop pairs used	10			
Number with an unused Flip Flop	1	10	10%	

Synthesize report shows the hardware resource for the tuning unit which moves the target signal from original frequency location to the zero frequency location.

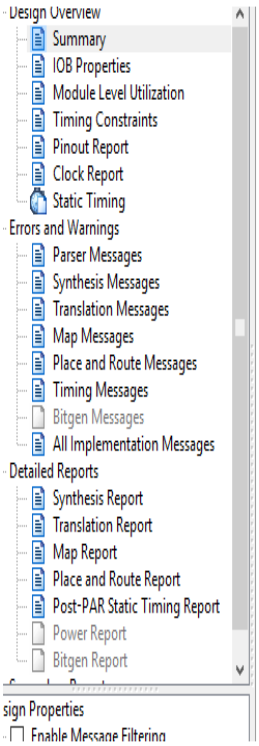
VFD FILTER



Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	833	728,400	1%	
Number used as Flip Flops	833			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	2,242	364,200	1%	
Number used as logic	2,179	364,200	1%	
Number using O6 output only	1,712			
Number using O5 output only	0			
Number using O5 and O6	467			
Number used as ROM	0			
Number used as Memory	0	111,000	0%	
Number used exclusively as route-thrus	63			
Number with same-slice register load	0			
Number with same-slice carry load	63			
Number with other load	0			
Number of occupied Slices	768	91,050	1%	
Number of LUT Flip Flop pairs used	2,249			
Number with an unused Flip Flop	1,482	2,249	65%	

Synthesize report shows the hardware resource for the VFD filter and it is used to implement arbitrary SRC.

2/4 PHASE FILTER



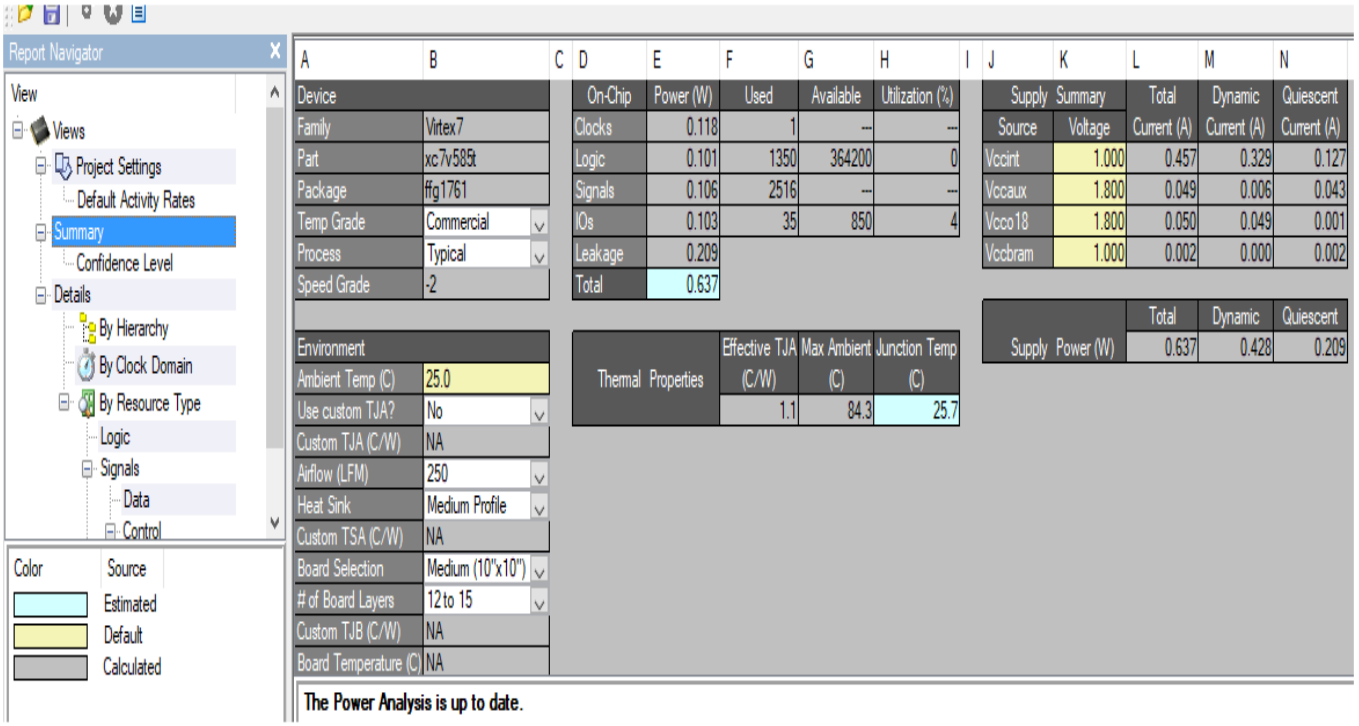
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	525	728,400	1%	
Number used as Flip Flops	512			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	13			
Number of Slice LUTs	98	364,200	1%	
Number used as logic	65	364,200	1%	
Number using O6 output only	51			
Number using O5 output only	0			
Number using O5 and O6	14			
Number used as ROM	0			
Number used as Memory	0	111,000	0%	
Number used exclusively as route-thrus	33			
Number with same-slice register load	33			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	165	91,050	1%	
Number of LUT Flip Flop pairs used	527			
Number with an unused Flip Flop	35	527	6%	

Synthesize report shows the hardware resources for the 2/4 phase filter and it is used to implement the gain control of the architecture.

9.4 POWER REPORT

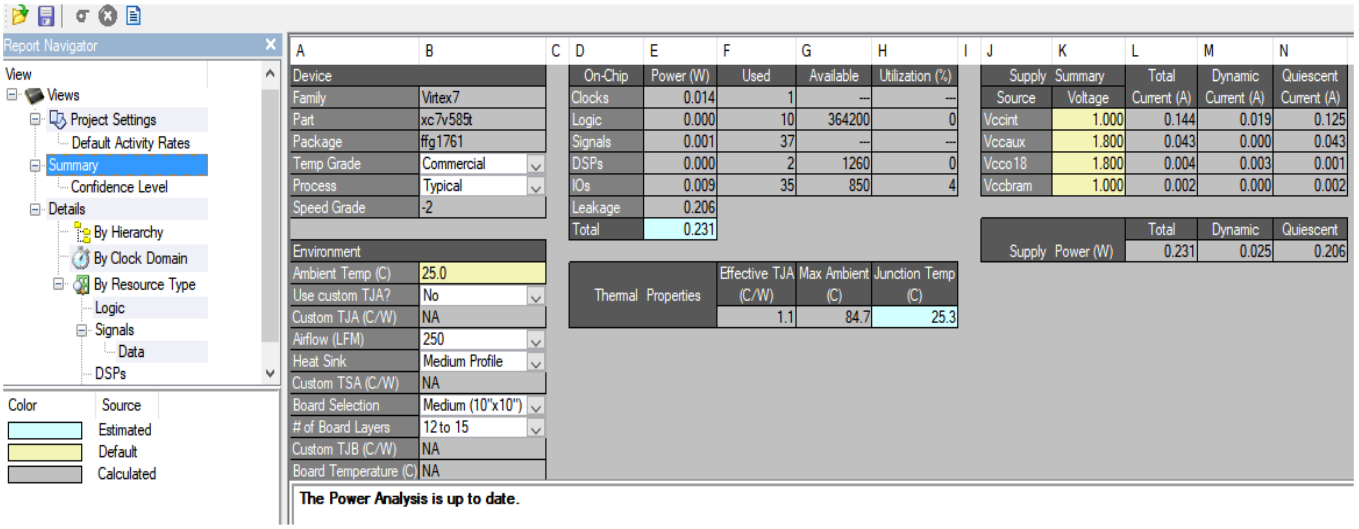
Power plays vital role in the VLSI technology and it is used to handle good and number of hardware to implement in the form of gates. It allows to determine whether the design meets power requirements and it also reports the expected junction temperature, off chip current drawn in to the devices.

Coarse channelization



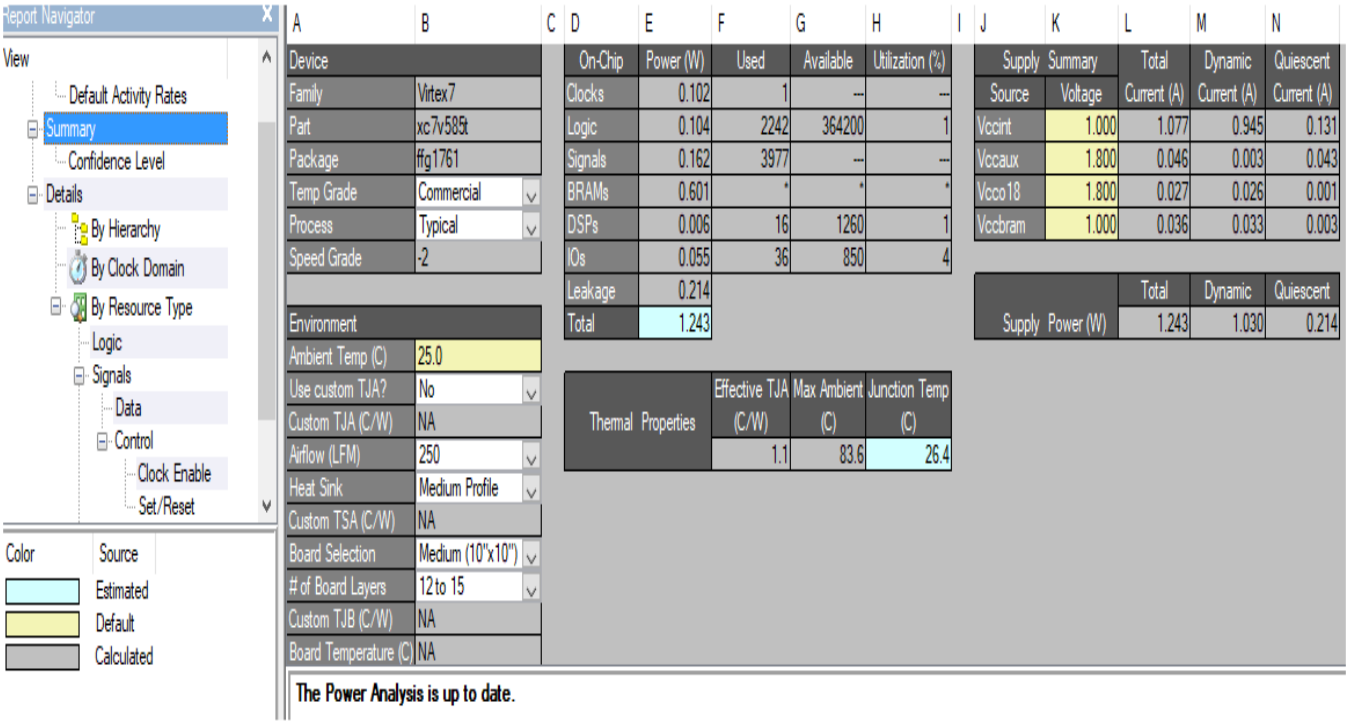
It shows how much power and clock pulses required for the coarse channelization block

TUNING



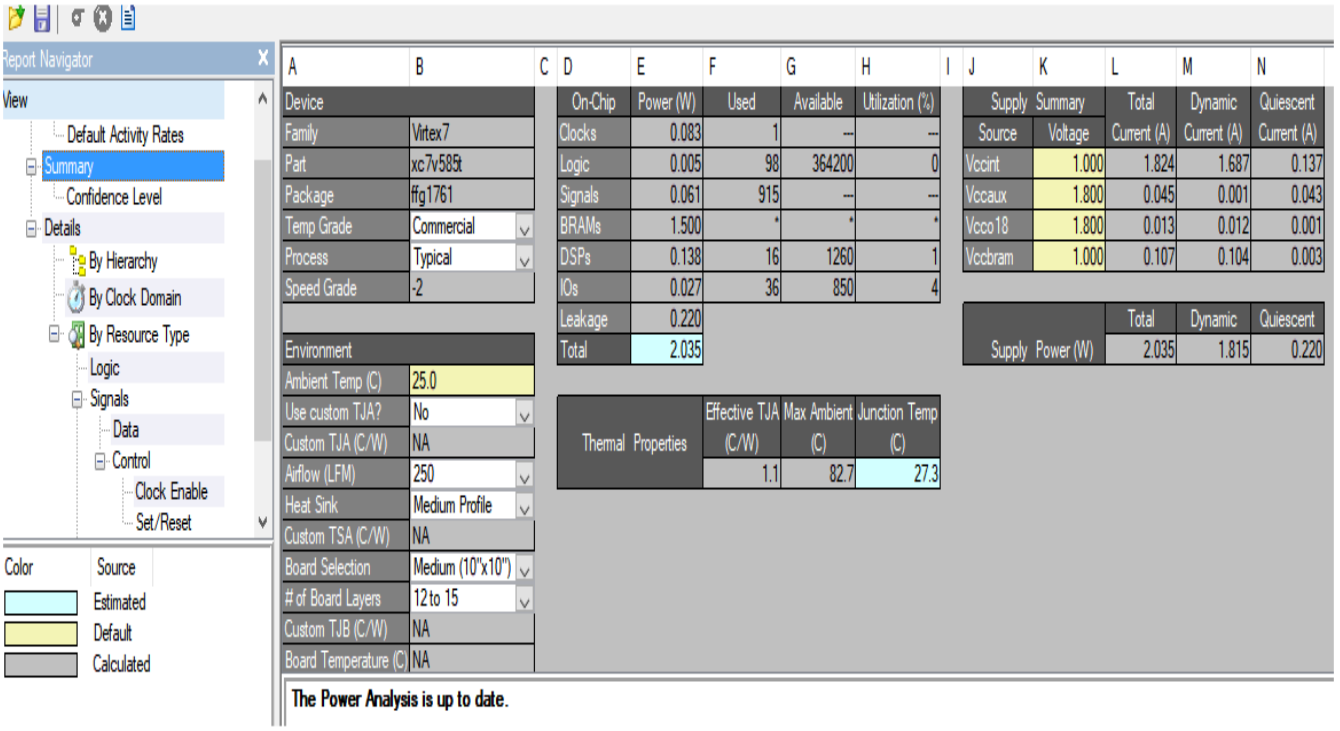
It shows how much power and clock pulses required for the tuning unit.

VFD FILTER



It shows how much power and clock pulses required for the VFD filter

2/4 PHASE FILTER



It shows how much power and clock pulses required for the 2/4 phase filter.

TABLE 9.1: Hardware Resources for the blocks

Hardware resource	Coarse channelization block	Tuning units	Farrow based VFD filters	2/4 phase filters	The whole design
Registers	4270	10x2	833x2	525x2	7006
LUT	3536	10x2	2249x2	527x2	9108
Block RAM	126	5x2	63x2	33x2	328
Latency(ns)	3.456	3.322	3.280	3.663	13.721
Power(mW)	637	231	1243	2035	4146

TABLE 9.2: Comparison Over Other Filter Banks

STANDARD	DFT filter bank[6]	NON uniform filter bank[17]	Proposed solution in Fig 7.2
Registers	11865	10157	7006
Latency	26.549	30.12	13.721

Although our architecture has a lesser latency than the nonuniform filter bank and DFT filter bank, it is a better choice for digital channelization due to the advantages of dynamic configuration and area saving over other existing solutions and for this reason the cascaded implementation of the coarse channelization and the time-multiplexing technique used in the whole design. Table 9.2 shows that our design offers the lesser latency of 2.19% over the nonuniform filter bank in [17]. The spectrum of our design is in the range 0 to *Fs*, which is twice as width as that of the design in [17]

CHAPTER 10
CONCLUSION AND FUTURE WORK

10.1 RESULTS AND DISCUSSION

A reconfigurable SDR channelization architecture that can provide up to 1024 channels of complex down converted output data is proposed. Each channel is independently configurable at runtime and can be programmed to support multistandard digital down conversion.

These configurable parameters include input source, center frequency, output sampling rate, filter response, and gain. This architecture solution can filter individual channels of arbitrary center frequency to a required standard. It can replace multiple Digital Down Converter application-specified integrated circuit devices with a single FPGA, significantly reducing the power, size, weight, and cost. Therefore, the architecture is a better solution for SDR applications, such as instrumentation, electronic warfare, and broad-spectrum surveillance.

10.2 FUTURE SCOPE

Implementation can be extended to 2048 channels and larger size of inputs.

REFERENCES

[1] A. G. Dempster and M. D. Macleod, “Use of minimum-adder multiplier blocks in FIR digital filters,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 9, pp. 569–577, Sep. 1995.

[2] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, “Splitting the unit delay [FIR/all pass filters design],” *IEEE Signal Process. Mag.*,vol. 13, no. 1, pp. 30–60, Jan. 1996.

[3] A. Y. Kwentus, Z. Jiang, and A. N. Willson, Jr., “Application of filter sharpening to cascaded integrator-comb decimation filters,” *IEEE Trans.Signal Process.*, vol. 45, no. 2, pp. 457–467, Feb. 1997.

[4] H. J. Oh, S. Kim, G. Choi, and Y. H. Lee, “On the use of interpolated second-order polynomials for efficient filter design in programmable down conversion,” *IEEE J. Sel. Areas Commun.*, vol. 17, no. 4,pp. 551–560, Apr. 1999.

[5] Y. C. Lim, R. Yang, D. Li, and J. Song, “Signed power-of-two term allocation scheme for the design of digital filters,” *IEEE Trans. CircuitsSyst. II, Analog Digit. Signal Process.*, vol. 46, no. 5, pp. 577–584, May 1999.

[6] T. Hentschel, “Channelization for software defined base-stations,” Ann. Telecommun., vol. 57, nos. 5–6, pp. 386–420, May/Jun. 2002.

[7] C. K. S. Pun, Y. C. Wu, S. C. Chan, and K. L. Ho, “On the design and efficient implementation of the Farrow structure,” *IEEE Signal Process.Lett.*, vol. 10, no. 7, pp. 189–192, Jul. 2003.

[8] J. Lillington, “Flexible channelization architectures for software defined radio front ends using the tunable pipelined frequency transform,” in Proc. IEE Colloq. DSP

Enabled Radio, Sep. 2003, pp. 1–13.

[9] W. A. Abu-Al-Saud and G. L. Stuber, “Efficient wideband channelizer for software radio systems using modulated PR filterbanks,” IEEE Trans. Signal Process., vol. 52, no. 10, pp. 2807–2820, Oct. 2004.

[10] G. López-Risueño, J. Grajal, and A. Sanz-Osorio, “Digital channelized receiver based on time-frequency analysis for signal interception,” IEEE Trans. Aerosp. Electron. Syst., vol. 41, no. 3, pp. 879–898, Jul. 2005.

[11] S. C. Chan, K. M. Tsui, and T. I. Yuk, “Design and complexity optimization of a new digital IF for software radio receivers with prescribed output accuracy,” *IEEE Trans. Circuits Syst. I, Reg. Papers,*vol. 54, no. 2, pp. 351–366, Feb. 2007.

[12] D. Gupta *et al.*, “Digital channelizing radio frequency receiver,” *IEEETrans. Appl. Supercond.*, vol. 17, no. 2, pp. 430–437, Jun. 2007.

[13] N. Lashkarian, E. Hemphill, H. Tarn, H. Parekh, and C. Dick, “Reconfigurable digital front-end hardware for wireless base-station transmitters:Analysis, design and FPGA implementation,” *IEEE Trans. CircuitsSyst. I, Reg. Papers,* vol. 54, no. 8, pp. 1666–1677, Aug. 2007

[14] R. Mahesh, A. P. Vinod, E. M.-K. Lai, and A. Omondi, “Filter bank channelizers for multi-standard software defined radio receivers,” J. Signal Process. Syst., vol. 62, no. 2, pp. 157–171, Feb. 2011

[15] R. Mahesh and A. P. Vinod, “Reconfigurable low area complexity filter bank architecture based on frequency response masking for nonuniform channelization in software radio receivers,” IEEE Trans. Aerosp. Electron. Syst., vol. 47, no. 2, pp. 1241–1255, Apr. 2011.

[16] R. Mahesh and A. P. Vinod, “Low complexity flexible filter banks for uniform and non-uniform channelisation in software radios using coefficient decimation,” IET Circuits Devices Syst., vol. 5, no. 3, pp. 232–242, May 2011.

[17] S. J. Darak, A. P. Vinod, and E. M.-K. Lai, “A low complexity reconfigurable non-uniform filter bank for channelization in multi-standard wireless communication receivers,” J. Signal Process. Syst., vol. 68, no. 1, pp. 95–111, Jul. 2012.

[18] B. H. Tietche, O. Romain, and B. Denby, “A practical FPGA-based architecture for arbitrary-ratio sample rate conversion,” *J. SignalProcess. Syst.*, vol. 78, no. 2, pp. 147154, Feb. 2015.

[19] Z. Wang, X. Liu, B. He, and F. Yu, “A combined SDC-SDF architecture for normal I/O pipelined radix-2 FFT,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 5, pp. 973–977, May 2015.

[20] W.-S. Lu and T.-B. Deng, “An improved weighted least-squares design
for variable fractional delay FIR filters,” *IEEE Trans. Circuits Syst.II, Analog Digit. Signal Process.*, vol. 46, no. 8, pp. 1035–1040,

LIST OF PUBLICATION

CONFERENCE

Presented a paper titled "**DESIGN OF A REAL TIME RECONFIGURABLE CHANNELIZATION ARCHITECTURE FOR SOFTWARE DEFINED RADIO APPLICATION**" in an IEEE Sponsored 4th International Conference on Electronics and Communication Systems(ICIIECS'17) on 17 and 18 march 2017 at Karpagam College of Engineering, Coimbatore