Resource Minimal Architecture Design for Software Defined Radio Front-Ends

Awan, Mehmood-Ur-Rehman

Publication date:
2013

Document Version
Accepted author manuscript, peer reviewed version

Link to publication from Aalborg University

Citation for published version (APA):
Awan, M-U-R. (2013). *Resource Minimal Architecture Design for Software Defined Radio Front-Ends*.

Resource Minimal Architecture Design for Software Defined Radio Front-Ends

PhD Dissertation

Mehmood Ur Rehman Awan

AALBORG UNIVERSITY

Resource Minimal Architecture Design for Software Defined Radio Front-Ends

PhD Dissertation

This dissertation is a revised version of the dissertation
”Resource Optimal Architecture Design for Software Defined Radio Front-Ends”
submitted to Faculty of Engineering, Science and Medicine at Aalborg University,
and defended successfully on March 22, 2013.

ISBN: 978-87-7152-002-6
March 2013

Resource Minimal Architecture Design for Software Defined Radio Front-Ends

PhD Dissertation by:
Mehmood Ur Rehman Awan

Supervised by:

Peter Koch
Associate Professor
Technology Platforms Section, Dept. of Electronic Systems,
Aalborg University, Denmark

fredric j. harris (co-supervisor)
Professor
Dept. of Electrical & Computer Engineering,
San Diego State University, CA, USA

List of Publications

- Mehmood Awan, Yannick Le Moullec, Peter Koch, and fred harris, ”Polyphase Filter Banks for Embedded Sample Rate Changes in Digital Radio Front-Ends”, *Special Issue on advances in Digital Front-End and Software RF Processing: Part II, ZTE Communications*, Vol. 9, No. 4, pp. 3-9, Dec 2011. ISSN 1673-5188.
- Mehmood Awan, Yannick Le Moullec, Peter Koch, and fred harris, ”Hardware Architecture Analysis of Polyphase Filter Banks Performing Embedded Resampling for Software Defined Radio Front-Ends”, *100G and Beyond: Trends in Ultrahigh-Speed Communications (Part I), ZTE Communications*, Vol. 10, No. 1, pp. 54-62, Mar 2012. ISSN 1673-5188
- Mehmood Awan, fred harris, and Peter Koch, ”Time and Power Optimizations in FPGA-Based Architectures for Polyphase Channelizers”, *45th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, Nov. 6-9, 2011, pp. 914-918. ISBN: 978-1-4673-0323-1
- Mehmood Awan, Peter Koch, Chris Dick, and fred harris, ”FPGA Implementation Analysis of Polyphase Channelizer Performing Sample Rate Change Required for both Matched Filtering and Channel Frequency Spacing”, *44th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, Nov. 7-10, 2010, pp. 414-418. ISBN: 978-1-4244-9721-8
- Mehmood Awan, and Peter Koch, ”Combined Matched Filter and Arbitrary Interpolator for Symbol Timing Synchronization in SDR Receivers”, *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems,* Vienna Austria, April 14-16, 2010, pp. 153-156. ISBN (print): 978-1-4244-6610-8/10

- Mehmood Awan, and Peter Koch, ”Polyphase Channelizer as Bandpass Filters in Multi-Standard Software Defined Radios”, *2^{nd} International Workshop on Cognitive Radio and Advanced Spectrum Management (CogART 2009)*, Aalborg, May 18-20, 2009, pp. 59-63. ISBN: 978-1-4244-4066-5

This thesis has been submitted to the Faculty of Engineering and Science at Aalborg University for assessment in partial fulfillment of the PhD degree. The thesis is based on the published scientific papers which are listed above. Parts of the papers are used directly or indirectly in the extended summary of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The thesis is not in its present form acceptable for open publication but only in limited and closed circulation as copyright may not be ensured.

Preface

The work was conducted from October 2008 to November 2012 under the research activities by Center for Software Defined Radio at Aalborg University.

Acknowledgements

I would like to thank my supervisor Assoc. Prof. Peter Koch and honorary co-supervisor Prof. fredric j. harris for their inspiration, contributions, and support during the whole process of the PhD work. I also thank Assoc. Prof. Yannick Le Moullec for his valuable contributions. I also thank Prof. fredric j. harris for inviting me to San Diego State University (SDSU) for my external research stay from February 2010 to July 2010. It was a wonderful time and experience, where I learned and explored the power of the multirate signal processing especially for Software Defined Radio. A special thanks goes to Michael Bayliss for his cooperation and hospitality during my stay in San Diego, CA. I thank Chris Dick from Xilinx Inc. for his cooperation. Furthermore, I would like to thank my colleagues at Aalborg University, Alex Aaen Birklykke, Muhammad Mahtab Alam, Rasmus Abildgren, Andreas Popp, and Jesper Michael Kristensen and my colleagues at SDSU, Elettra Venosa, and Xiaofei Chen for making every workday an enjoyable time. The same goes for the rest of my colleagues in the Technology Platforms Section.

Last but not least I thank my family and friends for their encouragement, company, and support that has always given me the energy and motivation to work.

Aalborg, Denmark, November 2012 Mehmood Ur Rehman Awan

List of Abbreviations

The following abbreviations are used throughout the thesis:

ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
AM	Amplitude Modulation
ASIC	Application Specific Integrated Circuit
CIC	Cascade Integrator Comb
CLB	Configurable Logic Block
CMOS	Complementary Metal-Oxide-Semiconductor
CORDIC	COordinate Rotation DIgital Computer
CR	Cognitive Radio
DAC	Digital-to-Analog Converter
DCM	Digital Clock Manager
DDC	Digital Down Converter
DDS	Direct Digital Synthesis
DFE	Digital Front-End
DFT	Discrete Fourier Transform
DFTFB	Discrete Fourier Transform Filter Bank
DSE	Design Space Exploration
DSP	Digital Signal Processing
DUC	Digital Up Converter
FFT	Fast Fourier Transform
FIFO	First-In-First-Out
FIR	Finite Impulse Response
FM	Frequency Modulation
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
HDTV	High Definition Television

HW	Hardware
IC	Integrated Circuit
IF	Intermediate Frequency
IFFT	Inverse Fast Fourier Transform
IIR	Infinite Impulse Response
IP	Intellectual Property
ISE	(Xilinx) Integrated System Environment
LPA	Load Process Architecture
LUT	Look-Up Table
MAC	Multiply-and-ACcumulate
MPRB	Modulated Perfect Reconstruction Bank
OQAM	Offset Quadrature Amplitude Modulation
PC	Personal Computer
PR	Perfect Reconstruction
QMF	Quadrature Mirror Filter
RA	Runtime Architecture
SDR	Software Defined Radio
SNR	Signal-to-Noise Ratio
SRC	Sample Rate Conversion
SW	Software
TRF	Tuned Radio Frequency
VCO	Voltage Controlled Oscillator
VHDL	Very-High-Speed-Integrated-Circuit (VHSIC) Hardware Description Language
VLSI	Very Large Scale Integration

Contents

Abstract

The high and constantly increasing demand to develop radio transceivers with enhanced functionalities, inter-standard portability, higher throughputs, compact physical size, low cost, and longer battery lifetime impose challenges on the radio designer. A Software Defined Radio is capable of meeting these challenges provided (i) enhanced signal processing algorithms with reduced complexities, (ii) high performance HW/SW platforms, and (iii) the optimized architectural design that finally maps the algorithms to the processing platform.
This dissertation focuses on a resource minimal architecture design for Software Defined Radio (SDR) front-ends targeting a reconfigurable architecture platform using a Field Programmable Gate Array (FPGA). Front-end design techniques ranging from legacy radios to the state-of-the-art radios are presented and analyzed. The results showed that multirate signal processing based techniques provide high performance solutions at a reduced resource complexity, thus making them highly suitable for the SDR front-ends. The outcome is a set of structural guidelines for the designer which can be used to obtain an Area, Time, operating clock Speed, and Power minimized architectural design for SDR front-ends using polyphase filter bank solutions - a multirate signal processing technique. The design space of reconfigurable architectures is explored to suggest different mapping solutions for polyphase filter bank which are analyzed based on their resource usage and corresponding performances. Non-maximally decimated operating scenarios are identified and scheduling schemes are proposed to meet their output time constraints. The scheduling schemes applied are: Load-Process scheduler, Interlaced Load-Process scheduler, and Runtime scheduler. The proposed architectures are based on these schedulers, which in the light of exploration of the reconfigurable target space (Xilinx Virtex-5 FPGA) are presented to compare resource and performance metrics.
The dissertation is composed of four parts: First, the related design challenges and state-of-the-art in SDR front-ends are discussed. The focus is on design solutions with the least computational requirements while operating at the lowest possible clock rate, and having the ability to embed several functions together. Second, a design space exploration of polyphase filter banks is presented and a set of structural design guidelines are suggested to the designer enabling Area, Time, operating clock Speed, and Power optimized solutions for SDR front-end designs. Third, a summary of the contributions is provided which relates to the published papers. Finally, a conclusion and outlook are given.
The main body of the dissertation is composed of six peer-reviewed papers which present the scientific contributions. The results show that the polyphase filter (bank) can embed multiple functions in its operation in a cost effective manner, such as combined arbitrary resampling, match filtering and symbol timing recovery, and rational resampling apart from a maximally decimated scenario. Furthermore, the suggested Runtime Architecture (RA) based on a Runtime scheduling scheme, with (i) distributed RAM organized as a pipelined register bank, (ii) distributed RAM or pipelined block RAM based coefficient bank, and (iii) systolic array of DSP48E (dedicated computational resources on Xilinx

FPGA) slices based Multiply-Accumulate (MAC) is a preferred choice to enhanced performance and reduce latency for SDR digital front-ends.
The lessons learned during the exploration of the solution space for the polyphase filter banks provide a sound basis for the system designer to choose the solution(s) which meet(s) the normally tight resource and performance constraints. Besides, the formulated design guidelines which constitute our most important scientific contribution, can help the designer at any point in the design process, thus potentially leading to a reduced overall cost and time-to-market overhead.

Dansk Resumé

De til stadighed voksende krav, der stilles ifm. udvikling af radio-sendere og -modtagere med avancerede funktioner som eksempelvis understøttelse af flere forskellige kommunikationsstandard, høje datatransmissions-rater, kompakt fysisk størrelse, lav pris og lang batterilevetid, medfører markante faglige udfordringer for den moderne radio-ingeniør. Med udbredelsen af den såkaldte ”software defined radio” (SDR), er der imidlertid tilvejebragt et koncept, som er i stand til at imødekomme mange af nutidens krav og udfordringer, idet netop dette koncept gør brug af (i) højtydende signalbehandlings-algoritmer med lav beregningsmæssig kompleksitet, (ii) avancerede HW/SW platforme, og ligeledes (iii) understøttes muligheden for at designe HW/SW-arkitekturer, som er skræddersyede til de nødvendige signalbehandlings-algoritmer.

Denne phd-afhandling fokuserer på design af resource-minimale FPGA-baserede arkitekturer til SDR modtageres front-ends, dvs. de første signalbehandlingsalgoritmer umiddelbart efter analog-til-digital konverteringen. Vi præsenterer og analyserer design-teknikker for såvel traditionelle som moderne radio front-ends, og vi viser at signalbehandlingsalgoritmer baseret på det såkaldte ”multi-rate” princip med fordel kan benyttes til at frembringe højtydende SDR front-ends med reduceret ressource-forbrug. Vi tilvejebringer retningslinier for struktureret design med henblik på at opnå bedst mulig performance mht. fysisk areal, eksekveringstid, og effektforbrug for SDR front-end arkitekturer. Hertil udnytter vi såkaldte poly-fase filter-banker. Med udgangspunkt i omkonfigurerbare HW-arkitekturer afsøger vi det såkaldte design-løsningsrum med henblik på at fremsætte forslag til design-løsninger for poly-fase filter-banker. Løsningsforslagene evalueres ift. det aktuelle ressource-forbrug og den givne performance. Vi identificerer scenarier baseret på ”non-maximal decimation” og vi foreslår i relation hertil eksekverings-planer, der imødekommer algoritmernes tidsmæssige krav. Planerne omfatter Load-Process, Interlaced Load-Process og Runtime afvikling. Med udgangspunkt heri foreslår vi tilhørende HW-arkitekturer, hvis ressource-forbrug og performance analyseres.

Afhandlingen består af fire hovedelementer; For det første forkuserer vi på de udfordringer der er ved design af moderne SDR front-ends. Vores fokus er rettet mod designløsninger med mindst mulig beregnings-kompleksitet, som kan udføres med den lavest mulige clock-frekvens, og som samtidig understøtter muligheden for at udføre adskillige funktioner i den samme operation. For det andet udforsker vi løsningsrummet for poly-fase filterbanker med henblik på at definere retningslinier for ressource-optimal design af SDR front-ends. For det tredje gennemgår vi de resultater der er tilvejebragt og efterfølgende afrapporteret i form af videnskabelige artikler. Endelig konkluderer vi vores arbejde og funderer over mulige fremtidige aktiviteter. De fundamentale resultater, som er publiceret i seks bedømte artikler, viser, at poly-fase filter-banker kan bringes til at udføre flere samtidige funktioner, som eksempelvis ”arbitrary resampling”, ”matched filtering”, ”symbol timing recovery”, og ”rational resampling” (med undtagelse af maksimal decimering scenariet). Desuden viser vores resultater, at den foretrukne løsning (dvs. bedst performance og korteste delay) er vores Runtime Architecture (RA), som

baseres på Runtime eksekverings-planlægning. Denne løsning kan realiseres ved brug af forskellige metoder til lagring af og multiplikation med filterkoefficienterne, eksempelvis (i) distribueret RAM organiseret som en pipelined register-bank, (ii) blok-RAM, eller (iii) et systolsk array bestående af DSP48E beregningskerner (Xilinx FPGA).
De opnåede resultater ifm. afsøgning af løsningsrummet for poly-fase filter-banken har tilvejebragt et solidt og nyskabende videngrundlag, som system-designeren kan benytte ifm. valg og design af SDR front-end løsninger, der imødekommer strenge ressource- og performancemæssige krav. Den frembragte viden er omsat til strukturerede retningslinier for SDR front-end design, hvilke således udgør det mest fundamentale resultat i vores arbejde. Dette resultat har medført, at der nu kan udvikles løsninger, som er mere konkurrencedygtige mht. pris og udviklingstid end tidligere tiders løsninger.

1 Introduction

Telecommunication has penetrated into our daily life so much that it is difficult to imagine a world without it. From a fundamental point of view, communication is the transmission of information from one point to another. The radio frequency devices transmitting and receiving the information are called transceivers. The constantly increasing demands upon these radio devices are more functionality, higher throughput, compact physical size, and longer battery lifetime, and of course low cost. This requires not only advancements in signal processing algorithms but also advances in processing platforms and architectural implementation. In this chapter we will review the fundamental development of radio technologies, discuss the state-of-the-art, and conclude by formulating a hypothesis which will serve as the overall inspiration for our work.

1.1 History of Radio Transceivers

The history of the radio started with the discovery of electromagnetic waves by Maxwell in 1887. The successive inventions by Hertz, Marconi, and many others concerning propagation of waves and wireless telegraphy gave birth to various wireless technologies [1]. Marconi revealed the first practical radio system (apparently copying an earlier invention of Nicola Tesla) in 1895 which led to transatlantic radio communication in 1901 [2]. The invention of the crystal detector, as well as Fleming's valve and De Forest's audion improved the sensitivity of the receiver. Edwin Armstrong invented the regenerative sets based radio receivers in 1906, which made long distance reception a reality. The development of valves (tubes) as an amplifier and regenerative detector (using positive feedback) tremendously improved radio performance in terms of gain and selectivity. The First World War further drove the development of radio receiver technology. The development of tuned radio frequency receiver (TRF) technology in 1920s, basically a chain of individually tuned amplifiers, represented a major improvement in radio performance. Further improvements include direct conversion and autodyne receivers. Direct conversion used a local oscillator based mixer, producing an audible signal that is further amplified. This requires separate valves for oscillator and mixer. The autodyne receiver used the same valve for both mixer and oscillator making it difficult to optimize both functions [3]. The need for a higher level of selectivity and sensitivity performances led to the development of the superheterodyne receiver. This is a receiver with an amplifier at a fixed intermediate frequency, and a filter. The incoming signal is mixed with a variable frequency oscillator (local oscillator) to obtain the down-conversion at a fixed lower frequency called an

Intermediate frequency (IF). The signal is further passed through a lowpass filter to reject the unwanted (mixing) components, and then finally amplified [2]. This processing improved the selectivity of the receiver, enhanced the gain for valves being used at the lower frequencies (after the frequency conversion), and reduced the circuit oscillation problems [3].

In the late 1940s the transistor (using semiconductor technology) was invented which later replaced the vacuum tubes in radio receivers during the 1950s and 1960s [3] [4] [5]. These transistor devices turn out to be miniature, reliable, long lasting, and consume less power and generate less heat dissipation as compared to their counterpart, as vacuum tubes are fragile, bulky, unreliable, power hungry, and have high heat dissipation. Transistor radios became attractive for their inherent features. The realization of complex electronic circuits using a large number of transistors and their required interconnects led to further developments in semiconductor technology leading to the Integrated Circuit (IC). This enabled radio receiver technology to utilize high performance circuits that can be built at low cost, and in a significantly reduced volume. The technology developments resulted in new techniques, for example, a frequency synthesizer realizing a phase lock loop in the digital domain [4]. This synthesizer generates a precise and stable local oscillator signal for the receiver. With the further advancements in receiver technology, many of the functions performed in the analog domain have been replaced by digital methods using Digital Signal Processing (DSP) technology. The benefit is that the DSP techniques are not affected by temperature and other physical variables, electronic noise, and aging.

DSP based radio transceivers have passed through several generations of architectures. The traditional heterodyne architecture, considered the first generation of digital radio architecture, is shown in Fig. 1.1(a). It consists of a dual-stage down-converter, and only the baseband processing is done in the digital domain [6]. In the first stage, the RF signal is down-converted to a bandlimited Intermediate Frequency (IF). In the second stage, the IF filter output is again down-converted to baseband by a matched-quadrature mixer, and matched baseband filters that perform final bandwidth control. Next, the signal passes into the digital domain where the output of the analog-to-digital converter (ADC) is processed by digital signal processing (DSP) engines. These engines perform the required baseband processing, that is, synchronization, equalization, demodulation, detection, and decoding. The problem with this type of architecture is that gain and phase are imbalanced for the in-phase and quadrature (I/Q) components. This results in cross-talk between the narrowband channels due to aging (time, temperature) of the analog components of the quadrature down-converter. Each imbalance-related spectral image must be lower than the desired spectral term, and this is difficult to sustain over time and at varying temperature.

The need for extreme I/Q balance gave rise to the next generation of digital radios where the second stage (IF) down-conversion and, consequently, the channelization process is digital, as shown in Fig. 1.1(b). Digital conversion at IF provides greater control over the imbalance by manipulating the number of bits involved in the arithmetic operation. The precision of the coefficients used in the filtering process sets an upper limit for spectral artifacts at -5 dB/bit. This means that 12-bit arithmetic can achieve image levels below -60 dB [6]. The DSP based complex down-conversion, however, has two advantages:

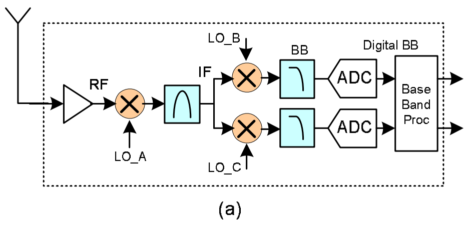


Figure 1.1: (a) First generation of digital radio receiver architecture, and (b) Second generation of digital radio receiver architecture.

(i) the spectral images are controlled so that they are below the quantization noise floor of the ADC involved in the conversion process, and (ii) the digital filters before and after the mixers are designed to have linear phase characteristics [6].

With more functions performed in DSP, the Personal Computer (PC) radio evolved. This has great advantage of being field upgradable both for existing and new functionalities. A Graphical User Interface (GUI) provides unlimited flexibility to the radio and new features can be added simply by a software upgrade instead of adding more hardware components. The next level in radio / software integration is Software Defined Radios (SDR).

1.2 Software Defined Radio

The concept of the SDR originates from the work of Mitola [7] in 1995, where a software adaptable radio architecture was proposed that enable the radio to automatically adjust to several different communication standards. This means that SDR is a radio communication system which can tune to any frequency band and receive any modulation across a large frequency spectrum by means of programmable hardware which is controlled by software. An Ideal Software Radio (ISR) is shown in Fig. 1.2. An SDR demodulating a simple AM broadcast may also be able to decode an HDTV broadcast simply by changing the software for its programmable hardware. The SDR not only provides high flexibility to adapt the radio front-end for any desired modulation, channel bandwidth, or carrier frequency, but also provides a cost effective solution by exploiting digital technology [8].

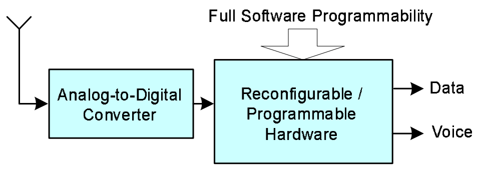

Full Software Programmability
Analog-to-Digital Converter
Reconfigurable / Programmable Hardware
Data
Voice

Figure 1.2: An Ideal Software Defined Radio.

In the early years, SDR was mainly attractive for military applications where there is a need for a single radio which can communicate with the many types of military radios that use different RF bands and different modulation schemes [9]. The emerging standards and the high penetration of mobile communications systems strongly demand software radio based mobile terminals and base stations for mobile communications systems [10] [11]. There are lots of system level issues in the wireless communication industry which embrace the essence and motivation for SDRs, such as;

- The rapid growth of commercial wireless network standards (2G, 2.5G, 3G, and 4G) has caused incompatibility issues for subscribers, wireless network operators, and equipment vendors, due to differences in their link-layer protocol standards.
- Subscribers are obliged to buy new handsets to utilize new services.
- Wireless network operators have to migrate their network from one generation to the next, while supporting both new and legacy handsets, and further they face deployment issues while rolling-out new services.
- Handset vendors face challenges in manufacturing viable multi-mode handsets.
- Equipment vendors face challenges in developing newer generation equipment due to short time-to-market requirements.
- The use of different air interface and link-layer protocols across various geographies has restrained the deployment of global roaming facilities.

SDR technology promises to solve (at least to a certain extent) these problems by implementing the radio functionality as software modules running on a generic hardware platform. Further, multiple software modules implementing different standards can be present in the radio system. The software modules that implement new services/features can be downloaded over-the-air onto the handsets. This kind of flexibility offered by SDR systems helps alleviate problems due to differing standards and issues related to deployment of new services/features. There are a lot of advantages of the full downloadable type of software radio because the system can be changed on demand by changing the software. There are many advantages not only for operators and service providers, but also for government and commercial customers such as global roaming services, bug fixes without the need to recall the product, and new services can be added without changing the terminals [12]. The most promising application of SDR is the application of cognitive radio (CR) [13]. As the radio spectrum becomes more and more sparse, making it a difficult task to allocate new spectrum for new services. A CR is aware of its environment, internal state, its location, and can make a decision about its operating behaviour based on this information.

Challenges

The idea behind SDR is to push as much radio functionality as possible into the digital domain via programmable digital hardware in order to achieve highly configurable designs [7]. This can be achieved by bringing the analog-to-digital conversion process (i.e., Analog-to-Digital Converters (ADCs)) as close to the antenna as possible. An ISR, as shown in Fig. 1.2, samples the signal at RF, just after the antenna and is termed full-band digitization. This straightforward approach is highly dependent on the ADC technology that could provide sufficient bandwidth and dynamic range performance for a SDR application together with low power consumption. The state-of-the-art ADC technology, however, bring the concept of Giga-Hertz sampling to a reality [14, 15].

In a realizable version of the software radio, shown in Fig. 1.3, the signal bandwidth must be reduced which is termed partial-band digitization. This can be realized with conventional receiver techniques, i.e. a heterodyne receiver as described in the second generation of digital radio. The partial-band digitization demands an analog front-end. Although it requires less analog functionality as compared to the conventional analog receivers, the requirements are heavily increased due to the presence of higher bandwidth and the dynamic range of the multi-channel signals for an SDR receiver. After digitization through the ADC, the signal of interest must be extracted from the digitized multi-channel signal and shifted to the baseband before the signal processing can be performed. Furthermore, the sample rate applied to the selected signal must be in accordance with the air interface, which therefore requires some digital signal processing before the baseband processing. Traditionally, this digital signal processing is done through analog techniques in the analog front-end. The shifting of the ADC towards the antenna realized this functionality in the digital domain. This is termed a Digital Front-End (DFE) which includes channelization (i.e., digital down conversion and channel filtering) and sample rate conversion [10].

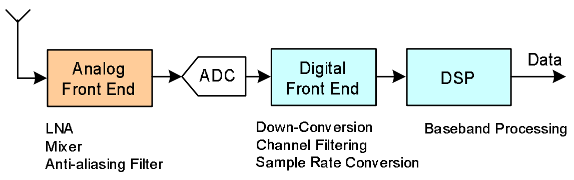


Figure 1.3: A realized version of Software Defined Radio.

The large bandwidth and high dynamic range of the signals to be processed by SDR eventually result in high sample rate and large word lengths. The high sample rates not only increase power consumption but also make the DFE infeasible for implementation on programmable digital signal processor devices. Apart from the other critical and enabling components of SDR (i.e., wideband analog front-end and the ADC) the requirement of reconfigurability and/or programmability for SDR in the presence of very high sample rates turns the DFE into one of the most power- and time-critical functionalities of an SDR [16].

Architectures

The hardware platform is the most prominent and challenging component of SDR as it has to provide massive computational power, flexibility, and at the same time meet strict power- and size limitations [17] [18]. Therefore, the selection of hardware architectures for SDR based applications is not a trivial task. As SDR signal processing algorithms are becoming more and more complex, the computational requirements on the target HW/SW-platforms are becoming equally high, and consequently software solutions make it possible to establish a smooth transition from dedicated, single-purpose hardware (ASICs) to highly versatile general-purpose hardware systems such as Field Programmable Gate Array (FPGA).

In general, algorithmic complexity (sometimes referred to as Shannon's law) is increasing at a rate faster than the processing capabilities of the hardware platforms (known as Moore's law), which introduces a technology gap. Bridging this gap may be done by developing new algorithms with reduced numerical complexity as well as devising alternative hardware architectures with higher performance. The advances in silicon technology are progressing according to Moore's law which however, cannot keep pace with the growing computational complexity. But at the same time Moore's law has also led to the fact that fabrication technologies now enable more transistors to be implemented on a single die than typical state-of-the-art design tools can easily handle. This situation leads to a so-called design-gap [19] [20]. According to the National Technology roadmap for semiconductors, the number of transistors that could be fabricated on a die was increasing at a rate of about 60% a year, whereas the number of transistors that circuit designers could design into new independent circuits was growing only by 20% a year [20]. This trend is observed in the DSP processor community where even the high-end DSP processors do not push the transistor densities as described by Moore's law [21].

To bring value to the state-of-the-art semiconductor products, the transistor budget must be used in a different way. That brings us to the FPGA technology which has improved tremendously over the past five to ten years. An FPGA is the ultimate commercial device for architectural customization, providing a huge solution space to designers for constructing their signal processing systems [21]. FPGAs are getting faster and their internal building blocks are becoming more efficient in terms of clock speed and reduced power consumption. The inherent parallel processing capability of FPGAs has made them the core-processing engine in SDR applications. These devices spend their transistor budget in a fundamentally different way than ISA (Instruction Set Architecture) machines and have enriched structure with CLBs (Configurable Logic Block), high speed I/Os, a variety of memory architectures, embedded hardware multipliers, Digital Clock Managers (DCMs), soft-core processors, and multi-gigabit transceivers (MGT). The most frequently used FPGA vendors are Xilinx [22] and Altera [23]. The internal architecture of a Xilinx Virtex-5 FPGA is shown in Fig.1.4.

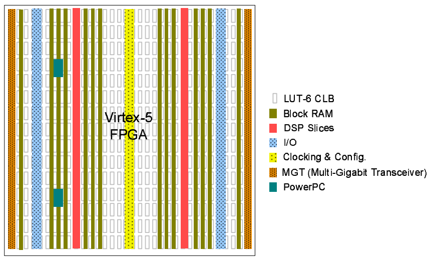


Figure 1.4: Architecture of Xilinx Virtex-5 FPGA.

FPGAs normally consist of two layers, where the first layer contains arrays of logic blocks, dedicated resources and routing between them. The actual configuration of the

FPGA is determined by a second layer of memory, determining the functionality of the logic and routing in the first layer.

The vast number of configurable parameters and customization in FPGA systems poses a great challenge to the designer as they offer a huge number of system, architectural, and logic design choices [24]. This creates a huge design space of possible architectures and mappings which increases the time and effort needed for the final mapping (implementation) of the system [25] [26] [27]. Finding the optimal design is often a matter of trade-offs between costs and performance, thus opening a new era of designing highly efficient FPGA based architectures requires design space exploration tools to help the designer to prune the design space.

The mapping process includes the design challenges of allocation, binding, and scheduling where allocation decides the number of resources (processing units) for performing a task, binding assigns the individual tasks to the processing units, and scheduling defines the relative task operations in time. Design space exploration is the evaluation of possible mapping solutions in the presence of a cost function, as expressed by Eq. 1.1, which may specify some of the parameters (e.g. execution time, area, power consumption, numerical precision etc.) as constrained by the requirements. A cost function can also include other parameters such as price and development time, degree of reconfigurability, maintainability, etc. It is generally accepted that finding an optimal design solution for a reconfigurable system is an NP complete [28] combinatorial optimization problem. We believe that a methodology/ tool can aid the designer during the design space exploration process in order to speed up the product development process and thereby meet the time-to-market requirements. There exist many design methodologies, but there may still be room for further improvements. To our knowledge, very little has been done towards experimenting with the implementation of advanced DSP algorithms for SDR front-ends on highly complex FPGA platforms. Therefore, in order to explore the possibilities and limitations of such designs, we have decided to investigate the advantages and disadvantages of design trade-off's as compared to selected metrics in the cost function. The cost, C, of an implementation typically is expressed by a cost function,

$$C = f\{A, T, P, N, ..\} \tag{1.1}$$

where A is area or resource consumption, T is execution time, P is power consumption, and N is the numerical precision. Each parameter can be assigned a weighting factor that defines its significance, giving a cost function such as $C = f\{\alpha A, \beta T, \gamma P, ..\}$. Thus, a 0.1 factor would mean that the corresponding parameter is 5 times less important than a parameter with a factor of 0.5. This is a somewhat abstract definition in the sense that the individual parameters can not be really compared. These factors (or metrics) lead to the design constraints which need to be optimized, either individually or simultaneously. In this work, we will not consider the actual values of the weight factors.

1.3 Multirate Signal Processing

We have now explained the requirement for high computationally efficient and flexible architectures for SDR. But at the same time the design of high-performance SDR front-end requires advancements in the signal processing algorithms with reduced computational and/or resource complexities. Multirate signal processing specifies alternative ways of performing DSP tasks which are normally not available in traditional DSP designs. It provides not only a reduction in the cost, but also enhances the performance of the implementation.

Multirate signal processing exploits sample rate changes at various stages of a system (hence the name multirate), in the form of decimation and interpolation of discrete time sequences [29]. From a traditional DSP perspective, the sample rate is selected to satisfy the Nyquist criterion. In multirate signal processing, selection and modification of the sample rate are the primary considerations and options in the signal processing chain. This ability to change the sample rate within the processing stream introduces a remarkable list of processing tricks and performance enhancements. It enables the processing task to be performed at the lowest rate matching the signal bandwidth, which is the Nyquist rate of the signal component of interest [30].

In contrast to a common processing task i.e., to reduce the bandwidth of a signal by filtering and then reducing the sample rate to match the reduced bandwidth, a multirate signal processing trick known as the Noble Identity interchanges the order of filtering and sample rate change so that the filter processing is done at the reduced output sample rate rather than at the high input sample rate. The Noble Identity operation is illustrated in Fig. 1.5. The reduction in the sample rate *prior* to bandwidth reduction causes aliasing of the input spectrum. Multirate signal processing permits, and in fact, supports this intentional aliasing, which can be unwrapped by subsequent processing. Most of the tricks and enhancements associated with multirate signal processing are related to spectral aliasing due to the sample rate change. The change in sample rate is also used to intentionally alias a signal at one center frequency to another center frequency. This option includes aliasing a signal from an intermediate center frequency to baseband by reducing the sample rate, as well as aliasing a signal from baseband to an intermediate center frequency by increasing the sample rate [30].

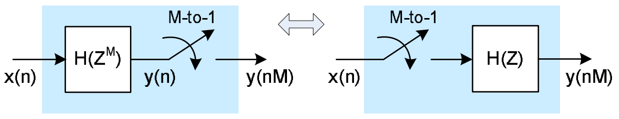


Figure 1.5: Noble Identity: A filter processing every M^{th} input sample followed by an M-to-1 down sampler is equivalent to an input M-to-1 down-sampler followed by a filter processing every input sample.

Polyphase decomposition, which originated from the work by Bellanger [31], plays a fundamental role in many multirate DSP applications. These include efficient real-time implementation of decimation and interpolation filters, fractional sampling rate changing devices, uniform DFT filter banks, and perfect reconstruction analysis/synthesis systems. A multirate polyphase filter can perform the tasks of a multichannel receiver which significantly reduces the amount of system resources required to perform multichannel processing and, consequently, reduces costs [6] [30]. Polyphase filter banks can be used in spectral sensing for cognitive radios [32]. Furthermore, they form a foundation of timing recovery schemes [33]. This allows large systems to be implemented efficiently requiring less area resources and cost, as explained in [34] where a polyphase filter bank based design replaced a huge number of multi-channel analog modulators. The 384 discrete FM modulators were implemented on a single Xilinx Virtex-4 FPGA.

In the next sections, we will explain the DFE building blocks: channelization, sample rate conversion, and synchronization; especially in the light of multirate signal processing and discuss the state-of-the-art.

1.4 Channelization

Channelization is the process of extracting one or more user channels from a wideband signal down to the baseband (at the required output sample rate) for further processing. Traditionally, it is done with a digital down-converter followed by a lowpass (channel) filter and optional sample rate conversion. The channels to be extracted may have equal or unequal bandwidths and may be uniform or non-uniform and continuously or non-continuously distributed over the input frequency band. As the number of down-converted channels is increased, the complexity of the system increases as each channel will require its own separate down-converter. Furthermore, at a high sample rate, standard approaches for channelization are inappropriate for implementation as the underlying technology platforms (e.g. FPGAs) can not handle the required processing loads.

Down-conversion

Down-conversion is the process of shifting the channel of interest centred at IF to the baseband. It is realized by multiplying the IF signal with a rotating complex phasor having a frequency identical to the IF centred frequency. The digital down-conversion process is illustrated in Fig.1.6.

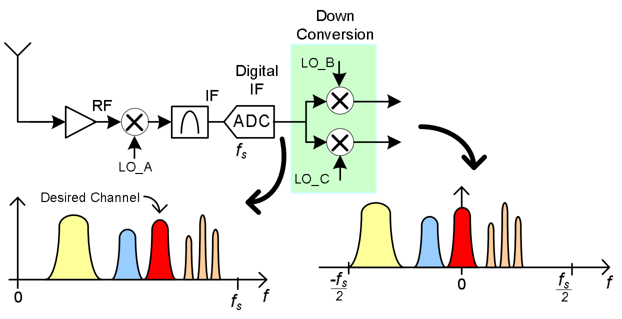


Figure 1.6: Down conversion process where the channel of interest is shifted to baseband.

The complex phasor can be generated in different ways. A straightforward way is to pre-calculate the corresponding sine and cosine values and store them in a memory. Direct Digital Synthesis (DDS) is a method to generate waveforms directly in the digital domain. It is composed of a phase accumulator and a phase-to-amplitude converter [35]. A conventional DDS is based on lookup tables (LUTs) where the amplitude values of sine or cosine waves are stored and are addressed by the phase accumulator. The COordinate Rotation DIgital Computer (CORDIC) algorithm [36] can behave as a quadrature phase-to-amplitude converter that directly generates sine and cosine waveforms [37] [38]. The main advantage of using CORDIC-based DDS in comparison with LUT-based methods is that it can achieve both high phase resolution and high precision with lower hardware cost [39]. A difference between the two methods is that the phase accumulator generates an integer value that addresses a LUT in the LUT-based method, while it generates an angle in the CORDIC-based DDS [40]. Among others, [41] and [42] presented several variants of DDS using CORDIC for very high spurious free dynamic range performances. An alternative to complex multiplication is the CORDIC algorithm operating as a vector rotator [43] which not only provides the complex samples of the rotating phasor, but also performs the multiplication at the same time.

The methods described above are used to generate the required complex phasors to be applied in multiplication for the down-conversion. The corresponding multiplication process and the preceding filtering operation are carried out at the input sample rate. The digital down-converter can be simplified considerably if the centre frequency of the channel of interest is centered at a quarter of the sample rate, thus the required complex phasors for multiplication reduces to a sequence of -1, 0, and 1. This will not only eliminate the DDS but also avoids the use of real number multipliers. Schreier and Snelgrove [44] presented a similar approach for down-conversion of signals centered at 1/8 of the sample rate.

The down conversion techniques mentioned above are replicas of analog legacy solutions. Multirate signal processing specifies alternative ways of performing DSP tasks where a 4-path polyphase filter performs the task of the down conversion from quarter sample rate while simultaneously performing baseband filtering and down sampling by a factor of 4 [30].

Channel Filtering

Channel filtering is required to extract the frequency divided channels from a wideband signal, as shown in Fig. 1.7. The channel filter has to attenuate the adjacent channel interferers according to the requirements of the particular air interface.

SDRs require filters of different bandwidths or a tunable filter to process the many different bandwidth signals. A filter can be implemented on reconfigurable hardware, e.g., FPGA that enables a complete reconfiguration for different air interfaces, or as a common filter that is parameterizable for all the required channel filtering. A reconfigurable filter is an easy solution, obtainable simply by reconfiguring the FPGA with another filter functionality. A parameterizable filter needs sophisticated solutions and the conventional FIR filter designs are not well suited. This is because the filter's length, i.e., number of taps varies inversely with the fractional bandwidth f_{BW}/f_s, and thus the required computational resources (such as data registers, multipliers, and adders) is different for filters with different lengths. Implementation considerations favor filters that are implemented with a fixed number of multipliers rather than with a varying number of multipliers. The

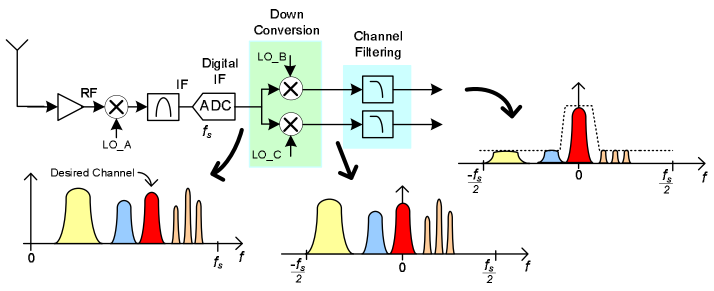


Figure 1.7: Channel filtering extracts the desired channel (channel at baseband) from the wideband signal while attenuating the adjacent channels.

application of multirate filters can provide a parametric filter solution by changing the rate factors [45]. Harris [46] presented a variable bandwidth FIR filter architecture with fixed computational resources. His solution used arbitrary interpolators at the input and output of a fixed bandwidth filter, and by changing the input and output rate by the arbitrary interpolator, the required bandwidth is achieved. The variable bandwidth filters can also be realized by frequency masking filters, tunable IIR filters, low-pass to low-pass transformation tunable FIR filters, and a number of other solutions. Many of these options have a constant computational complexity but none preserves linear phase [46]. Harris [47] also presented a selectable bandwidth filter by using a pair of M-path analysis and synthesis filters. The bandwidth is changed by masking, enabling or disabling, the connections between the analysis and the synthesis filter.

Sample Rate Conversion

In digital receivers, it is often required to have Sample Rate Conversion (SRC) either to interconnect different processing blocks or to deliver the user-defined data rate. A receiver architecture with sample rate conversion is shown in Fig. 1.8. Sample rate conversion is not limited to digital radio, it is widely used in digital audio signal processing and in digital control [48].

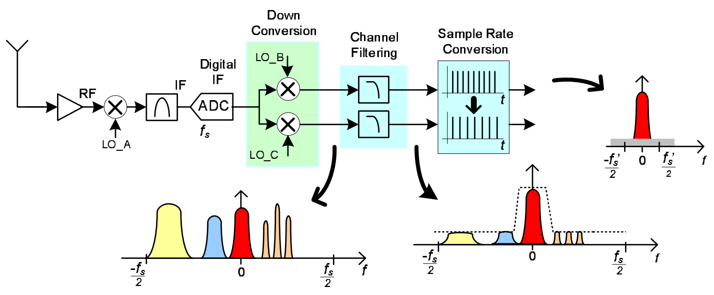


Figure 1.8: Receiver architecture with a sample rate conversion block to achieve the required output sample rate.

One of the very important and cost effective applications of SRC is an interpolation process in a modulator. It raises the input sample rate to allow the translation of the input spectrum to an intermediate frequency (IF), and then outputs the digital IF via a single DAC as compared to standard baseband process which requires matched DACs, matched lowpass filters, matched balanced mixers, and a quadrature oscillator to form the first IF band [30]. The necessity for SRC in communication receivers arises with the need to implement different air interfaces in a software defined radio. This is because of the fact that the baseband processing for any associated air interface is usually carried out at the target sample rate and not at an arbitrary sample rate. The target sample rate can, in principle, be realized by clocking the ADC at the specific rate of the actual air interface which requires a high quality tunable oscillator as a clock for the ADC. To avoid complex analog components, a tunable sample rate of the ADC is not the best choice instead one can use a SRC in the digital domain [10].

Sample rate conversion can be conducted as: (i) integer based up- and down-sampler, (ii) rational P/Q re-sampler, or (iii) more sophisticated arbitrary resampler. Traditionally, up-sampling is achieved by inserting zero-valued samples followed by an anti-imaging lowpass FIR filter, and down-sampling is achieved by discarding the appropriate samples preceded by a band-limited anti-aliasing FIR filter [49]. These up- and down-samplers are shown in Fig. 1.9a and Fig. 1.9b, where the input signal is down-sampled by a factor *M* and up-sampled by a factor *L*, respectively.

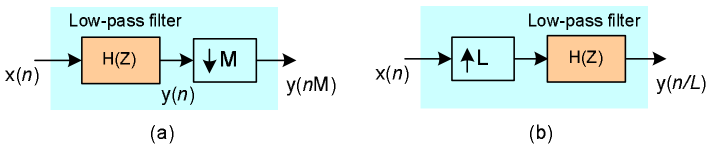


Figure 1.9: (a) Down-sampler by a factor *M* and (b) up-sampler by a factor *L*.

It is computationally inefficient to discard the computed samples in the case of a down-sampler, and to process the zero samples in the case of an up-sampler [30]. Multirate signal processing approaches [30], i.e., a polyphase filter based up- and down-sampler (employing the Noble Identity) or a polyphase filter based arbitrary resampler are efficient and cost effective solutions for the SRC. Furthermore, narrow band FIR filters using conventional DSP pose a serious problem because such filters need to have a very high order to meet their tight frequency response specifications. The use of multirate techniques leads to very efficient implementation by allowing filtering to be performed at a much lower rate, which greatly reduces the filter order [50]. The performance of a multirate system depends critically on the type and quality of the filter used. Either FIR or IIR filters can be used for decimation or interpolation, but FIR is the most popular because of its desirable attributes such as linear phase response and low sensitivity to finite word length, as well as being simple to implement [50].

A polyphase filter based arbitrary resampler needs pre-computed weights of the polyphase filter stages. On the other hand, a Farrow filter, which is a multirate filter structure, can provide a continuously adjustable resample ratio. It uses low order piecewise polynomials to compute the coefficients of the polyphase filter stages on the fly. With an up-sampling ratio greater than 1-to-5, the Farrow filter provides reduced complexity as compared to the standard polyphase form [51]. The polynomial form of the polyphase filter set has a significant advantage of reducing the memory resources required for the stage coefficients.

A class of multiplier-less filters, e.g., Cascade Integrator Comb (CIC) filters, can also be used for up- and down- sampling tasks. Although a CIC filter is not a good filter in terms of achieving desired out-of-band spectral attenuation, a cascade of 3 to 5 stages of CIC filters becomes attractive. Furthermore, it has a non-uniform passband gain which distorts the baseband spectrum. This main lobe gain reduction usually limits the input signal bandwidth to be less than 25% of the main lobe bandwidth. The spectral distortion due to the main lobe is corrected by embedding the inverse response in a FIR filter preceding or following the CIC [30].

Single channel and Multi-channel

There are two possible areas of application for channelization. In a mobile terminal, only one channel is normally selected, while a base station needs to select many channels. Single channel applications require a one-channel channelizer which can be realized by down-conversion and the channel filtering techniques discussed above. A straightforward solution for the multi-channel application is the ”per-channel approach”, shown in Fig. 1.10, which essentially is a parallel realization of M one-channel channelizers, where M is the number of channels [10]. This approach is highly flexible as it has no constraints on the channel bandwidths or their distribution. On the other hand it is rigid with respect to alteration and the resulting wideband channelizer demands high silicon cost and high power consumption. Other approaches are frequency domain filtering, pipelined frequency transform, and polyphase filter bank channelizer.

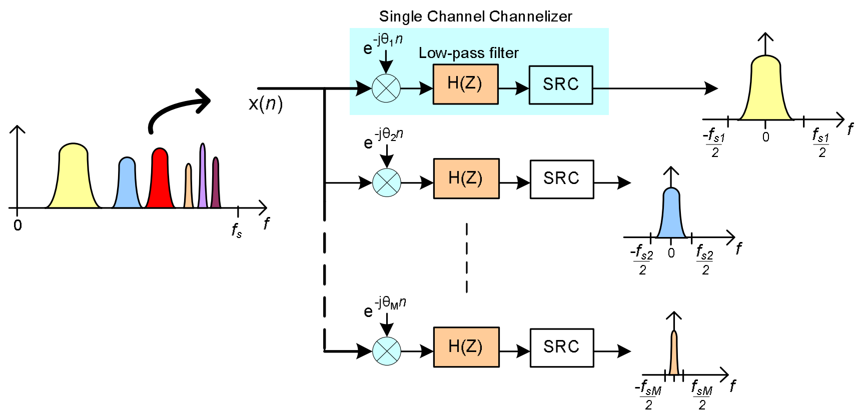


Figure 1.10: Extraction of multiple channels from a wideband input signal using a per-channel approach.

Frequency domain filtering utilizes the properties of the Fast Fourier Transform (FFT) to realize a large bank of narrowband filters. The frequency bins representing each channel of interest are extracted, filtered, and converted back to time domain using an Inverse FFT (IFFT) [52] [53].

The pipelined frequency transform approach creates a binary tree of digital down converters and sample rate converters to split the incoming signal frequency into a low and a high frequency sub-band until the last tree level separates the required channels [54] [55]. The computational complexity can be significantly reduced by exploiting half-band symmetry and sampling rate reduction at each output stage. The resulting structure is similar to Hierarchical Multistage Method (HMM) [56] or Quadrature Mirror Filter (QMF) tree [57]. A binary tree of low and high band frequency converter consisting of a set of Digital Down-Converters (DDC) and Digital Up-Converters (DUC) followed by SRCs is shown in Fig. 1.11.

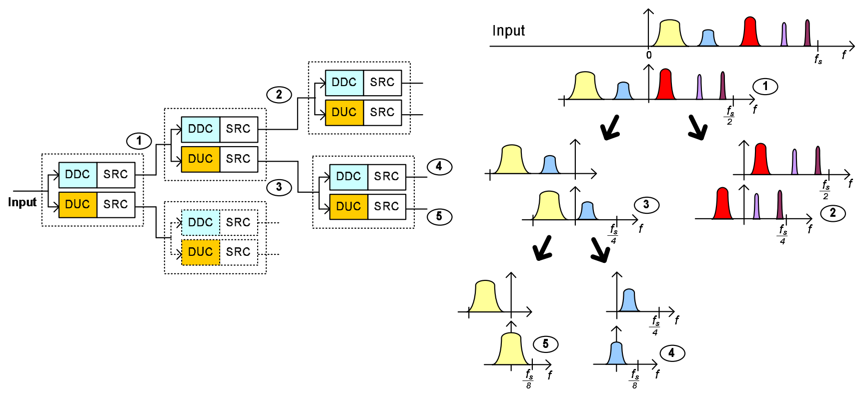


Figure 1.11: Pipelined frequency transform as a binary tree of low (DDC) and high (DUC) band frequency converters followed by SRC.

Computationally efficient algorithms for implementing a filter bank channelizer using FFT have been developed by several authors, e.g., [31] [58] [45] [59] [60] [30]. A polyphase filter bank channelizer improves upon the efficiency of the frequency domain filtering technique by assuming redundancy within the frequency plan of the wideband channel [45]. It employs a polyphase filter which is created through the decomposition of the lowpass filter used to provide channel isolation on a ”per channel” basis. The input rotary switch known as the ”commutator” down-samples and down-converts all the channels to baseband which are then extracted efficiently by using FFT. A polyphase filter bank channelizer is shown in Fig. 1.12. The step-by-step transformation of a single channel channelizer to a polyphase channelizer is presented in [6] [30]. The technique, often called Modulated Filter Banks or Discrete Fourier Transform Filter Bank (DFTFB), requires the channels of extraction to be uniformly distributed, and the sample rate of the incoming signal must be an integer multiple of the channel spacing. It is highly computationally and resource efficient even for a small number of channels [30]. There are several modifications of DFTFB in the literature which provide channelization for non-uniformly bandwidth channels.

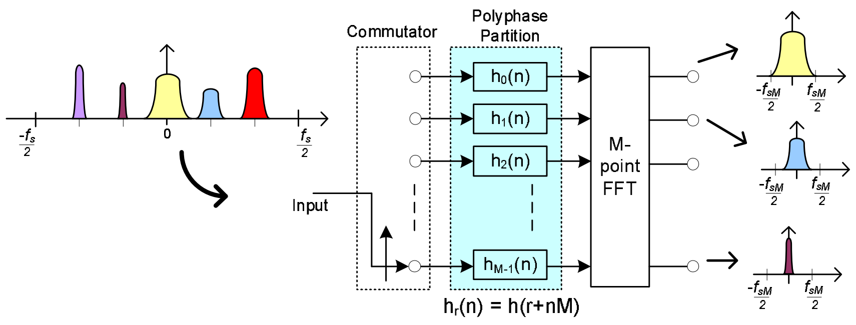


Figure 1.12: Polyphase filter bank channelizer which can efficiently extract all the input channels to the baseband by using a single lowpass filter.

The requirement for evenly distributed channels in DFTFB (referred to as fixed channel stacking) can be overcome by using the modified Goertzel approach [61], but at the expense of increased computational complexity. Abu-Al-Saud and Stuber [62] proposed a channelizer based on a modulated perfect reconstruction bank (MPRB). It consists of an analysis and a synthesis filter, where the analysis filter splits the incoming signal into sub-bands which are added together by the synthesis filter to generate the required wide-band signals. It uses complex exponential modulated perfect reconstruction filter banks to eliminate the aliases overlap at the expense of 2M branches in the filter bank.

In [63] Mahesh and Vinod present a reconfigurable polyphase filter bank with a coefficient decimation approach [64] that is capable of extracting channels of non-uniform bandwidth. However, for each configuration based on the selected coefficient decimation factor, the channel bandwidths are uniform. Harris, et al. [65] present an efficient structure, based on polyphase filter banks to filter and simultaneously down convert multiple signals having arbitrary bandwidths and randomly located center frequencies. Their approach is based on an analysis channelizer, post analysis block, selector, and synthesis channelizers. The analysis channelizer has a Perfect Reconstruction (PR) filter, performs $M/2$-to-1 down sampling while aliasing all M channels to the baseband. The post analysis block extracts the spectra or their fragments belonging to the different signals from the baseband aliased channels. The selector selects the output ports of the post analysis block that contain spectral fragments of the same bandwidths which are subsequently reassembled in the up converter synthesis channelizers.

1.5 Synchronization

Synchronization is one of the complex tasks performed in a high data rate wireless system [66]. If the radio is not synchronized, none of the sub-systems such as matched filters, equalizer, detectors, error correcting codes, decryption, and source decoding can operate [67] [68]. As would be expected, the receiver becomes complicated, requiring more subsystems than the transmitter. These subsystems perform synchronization and signal conditioning tasks which are required to demodulate the input signal. These subsystems consist of control loops that estimate the unknown parameters of the known input signal. These estimates are then used to perform corrective signal processing and signal conditioning operations in order to avoid the degrading due to the unknown parameters on the demodulation process. Fig. 1.13 represents a complete receiver architecture illustrating these subsystems which are [67];

1. an AGC loop to estimate and remove the unknown channel attenuation,

2. a carrier recovery loop to estimate and remove the unknown frequency offset between the input signal's nominal and actual carrier frequency,

3. a timing recovery loop to estimate and remove unknown time offsets between the receiver's sampling clock and the optimal sample positions of the matched filter output series,

4. an equalizer loop to estimate and remove unknown channel distortion which would be responsible for inter-symbol interference,

5. a phase recovery loop to estimate and remove unknown carrier phase offset between the input signal and the local oscillator, and

6. an SNR estimator to provide important information to the other subsystems.

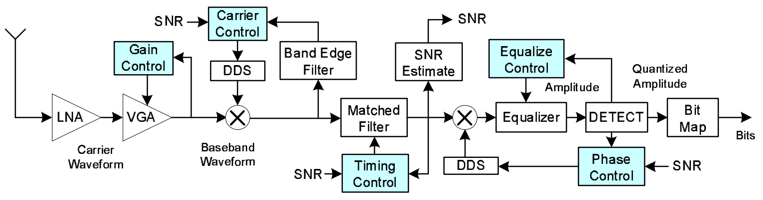


Figure 1.13: Receiver architecture illustrating the control loops for the synchronization.

Synchronization techniques based on DSP implementations are often digital versions of their analog prototypes. Such solutions make compromises appropriate for their time and do not exploit the actual strength of the DSP system [33]. In the 1^{st} generation radio receiver, apart from the gain and phase imbalance between the quadrature paths that limits the fidelity, the digital loop filters are implemented in DSP for the carrier and timing recovery, but analog components are used in the feedback paths in order to control the Voltage Controlled Oscillator (VCO). In the 2^{nd} generation, in order to eliminate the gain and phase imbalance of quadrature paths, as well as the analog components for control loops, the ADC is moved to the IF stage where quadrature down-conversion is performed in the digital domain. The sequence of operations in the 2^{nd} generation is (i) frequency translation, (ii) filtering, and (iii) sample rate conversion. These operations are the digital equivalent to the analog operations performed in a 1^{st} generation receiver. In modern 3^{rd} generation receivers, the processes of translation, filtering, and re-sampling are rearranged in clever ways (explained in the next section) which reduces the computational complexity, while increasing the utility of the operations [33].

Multirate signal processing techniques and CORDIC subsystems deliver efficient solutions for synchronization tasks, i.e., carrier recovery, matched filtering, timing recovery, and phase detection.

1.6 Modern Receivers - state-of-the-art

Many companies designing radio receivers and transmitters are still using analog solutions which they have directly translated into the digital domain. These legacy designs include compromises which were appropriate for their time (e.g., traditional analog filters trading off amplitude- and phase responses for filter order), but it is inefficient to map such compromises directly into the digital domain. Therefore, new design methodologies which can lead to more efficient DSP based architectures for digital front-ends, and which can efficiently utilize promising new hardware technologies, e.g., FPGAs, are highly needed. A digital filter can often perform more than its intended primary filtering task. It can absorb many of the secondary signal processing tasks of a system. There are many ways of folding other functions into the filtering process using multirate signal processing techniques, which will be presented here.

In a 2^{nd} generation radio receiver, the sequence of operations (in a digital replica of the analog operations performed in a 1^{st} generation receiver) is frequency translation, filtering, and sample rate conversion. In 3^{rd} generation of radio receivers [33], as shown in Fig. 1.14, the process of translation and filtering are interchanged according to the Equivalency Theorem [30] which states that down-conversion followed by baseband filtering is the same as the filtering at the carrier followed by down-conversion. This reordering operation is shown in Fig. 1.15.

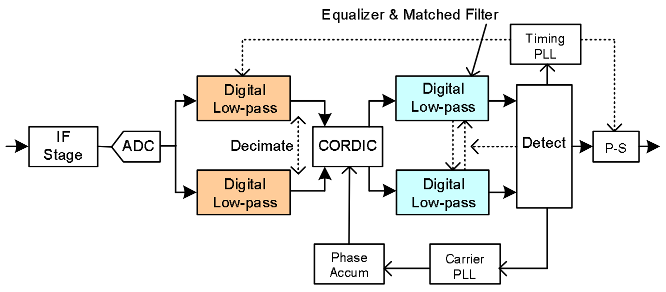


Figure 1.14: 3^{rd} generation of radio receiver architecture.

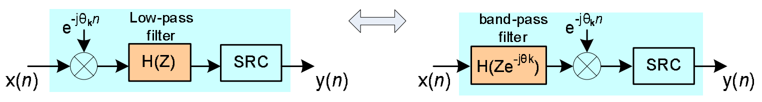


Figure 1.15: The Equivalency Theorem - down-conversion followed by baseband filtering is the same as filtering at the carrier followed by the down-conversion.

So, rather than moving the desired band to baseband with a complex digital heterodyne, the baseband filter is moved to the desired center frequency and filtering is performed to reduce the bandwidth at the digital IF frequency. The resultant reduced bandwidth, complex IF signal is image free and can be down-sampled M-to-1, to its appropriate Nyquist rate f_s/M. The down-sampling results in aliasing of the center frequency to a new spectral position. The down-sampling can alias the IF frequency directly to the baseband provided that the IF frequency is positioned at a multiple of the output sample rate [30]. A signal having an offset from a multiple of the output sample rate is also aliased to baseband plus that offset. This offset can then be removed with a carrier recovery scheme while operating at the reduced sampling rate. The rearrangement of the processes is shown in Fig. 1.16 which results in the advantage that the spectral translation now occurs after the filtering and at the low output rate rather than before the filtering at the high input rate [33].

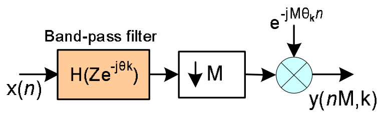


Figure 1.16: Reordering process that moves the down-conversion after the filtering and resampling process.

The M-to-1 down-sampling after the filtering at IF discards the processed samples which wastes the computational power used by the filter. To reduce this workload, the resampling and the filtering operations are reordered so that one output is computed for every M input samples. This is achieved by applying the Noble identity as was shown in Fig. 1.5 which requires the original up-converted filter to be partitioned to M subfilters. The reordering of the M-partitioned subfilters (lowpass filter) and resampler using the Noble Identity results in a polyphase filter, as shown in Fig. 1.17. It has the resampler at the input (a rotary switch which, as previously mentioned, is denoted by the commutator) feeding M subfilters one at a time which operate at the reduced output rate rather than the original input rate. As the subfilters operate sequentially, they can be merged into a single path structure having a subfilter that accesses the coefficients of the successive M subfilters.

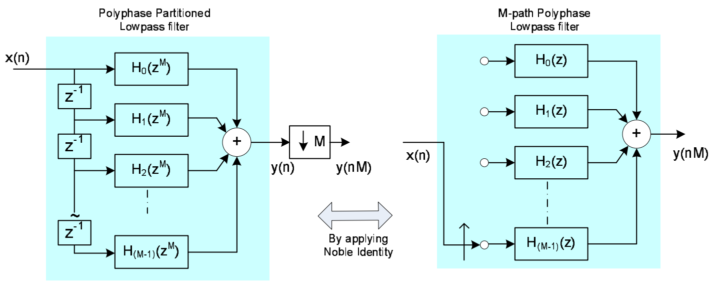


Figure 1.17: Polyphase partitioning of a lowpass filter used to apply the Noble Identity to form an M-path polyphase filter.

The lowpass version of a polyphase filter can be converted to its bandpass version by associating the complex heterodyne term of the modulation of the filter weights with the delay elements storing the filter weights. This is achieved by feeding the complex scalar $e^{jkr(2\Pi/M)}$ to each path of the M-path filter [30], where r is the number of the polyphase path, and k is the number of channel to be extracted at the baseband. This is shown in Fig. 1.18.

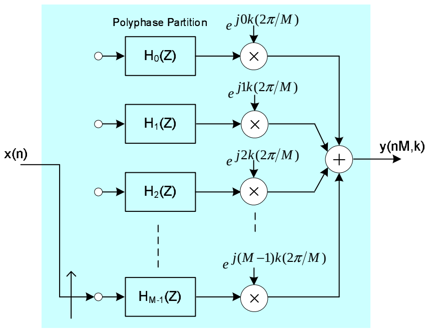


Figure 1.18: Resampling M-path down-converter.

The down-sampling process provides a unique feature of changing the phase of the sample locations in the down sampled time series relative to the epochs in the series. In an M-to-1 down sampling, the output time series can be accessed with M different sample phase offsets. These M possible output series can effectively be used for a timing recovery process. In such a case, the timing recovery loop controls the phase of the re-sampling process in the re-sampling filter rather than the loop modifying the locations of sample points during the ADC process as done in 1^{st} and 2^{nd} generation receivers. The phase of the re-sampling filter is defined by the selection of the phase weights in its single path structure. Timing recovery is achieved by controlling an index pointer in the filter's coefficient space. For a timing recovery process, the number of the polyphase stages (for a polyphase filter) must be increased to satisfy the required timing granularity rather than the down sampling requirement [33].

The re-sampling process in the polyphase filter (effecting both the spectral translation and the timing recovery process), results in a complex baseband signal at a reduced sample rate and with the correct timing. However, it does not have the correct carrier frequency and phase correction, which is subsequently corrected by a complex heterodyne operation before performing further processing by the equalizer and matched filter. The heterodyne on the complex baseband signal requires four multipliers as compared to the heterodyne on the real IF centered signal that requires only two multipliers. The complex multiplier can be embedded in the Direct Digital Synthesizer by using the CORDIC algorithm as a vector rotator [43] which not only provides the complex samples of the rotating phasor but performs the multiplication at the same time.

Furthermore, in an M-path polyphase filter, the output sample rate of the baseband aliased center frequencies is always equal to M. The M-path polyphase filter bank can also be modified to allow resampling by arbitrary ratios while simultaneously performing baseband aliasing from center frequencies at Nyquist zones that are not multiples of the output sample rate. These resampling technique are explained in [69] which embed the resampling process in (i) the polyphase commutator, that is, in the interaction between input data registers and the polyphase coefficients, and (ii) the interaction between the polyphase outputs and the FFT input. This only requires a state machine to schedule the interactions, hence there is no computational overhead.

1.7 Polyphase Filter Bank Mapping - State-of-the-Art

We have explored and presented algorithms/ techniques for performing the DFE processing tasks for modern SDRs. Multirate signal processing based algorithms turned out to be a valuable approach for efficient SDR processing engines. An M-path polyphase filter has M parallel branches, each operating at $1/M$ of the input rate. This M-paths parallelism can be exploited in a number of different ways to meet the desired cost function of area, time, power, etc. The mapping of these parallel structures to highly complex reconfigurable platforms (FPGA) demands a structured methodology to obtain not only a cost-function constrained implementation, but also to exploit enhanced mapping strategies. The state-of-the-art for mapping of polyphase filter banks to reconfigurable platforms is therefore discussed next.

Egg, Harris, and Dick [70] presented a polyphase filter bank based wideband channelizer implemented on a Xilinx Virtex-4 FPGA, to extract narrow band channels and further to reconstruct a wider bandwidth channel using perfect reconstruction filter. Melis and Comoretto [71] considered a 512 MHz polyphase filter bank with overlapping bands for use in a spectrometer. Their design used a Xilinx Virtex-5 FPGA, and the multipliers for the polyphase filter are based on CLB resources. Berner and Leon [72] discussed the FPGA implementation aspects of a polyphase filter bank as analysis and synthesis filter banks. Valdes et al. [73] showed a polyphase filter bank implementation in MP3 decoding by using 32-point DCT-II and windowing functions. Marinova et al. [74] presented a multicarrier modem core on an FPGA which uses Offset Quadrature Amplitude Modulation (OQAM) and polyphase filter based analysis and synthesis filter banks. Han [75] analyzed the front-end design for a wideband SDR. A 32 channel DFE using a polyphase filter bank was implemented on a Xilinx Virtex-5 FPGA. In [76], Ang et al. showed a Virtex FPGA implementation of a polyphase filter for sample rate conversion. They presented optimizations in terms of transformation to reduce the critical path, and constant coefficient multiplier for both area and speed performance improvement. Fahmy and Doyle [77] suggested an FPGA based reconfigurable polyphase filter bank architecture for spectrum sensing which is based on a shared MAC structure. Harris, Vuletic, and Lowdermilk [34] presented a polyphase filter bank based design to replace multi-channel analog modulators, where 384 FM modulators are implemented on a single Xilinx Virtex-4 FPGA. Wang et al. [78] showed an energy-efficient hardware architecture and VLSI implementation of a polyphase channelizer with applications to subband adaptive filtering. They used the Computation Sharing Differential Coefficient (CSDC) method to obtain a low complexity parallel multiplier-less implementation of FIR subfilters.

In most of the above mentioned literature, the mapping of the polyphase filter (banks) to an FPGA is presented as a final result emphasizing reduced resource usage, but often they lack a discussion of how these results are actually achieved on highly complex platforms which provide a huge design space. On the other hand, Intellectual Property (IP) cores are available from different vendors, e.g., [22] [79] [80] [81], these can be customized according to our needs and can eliminate to a certain extend the need for design space exploration. Normally, these IP cores are available for fixed functionalities (i.e., polyphase decimators and polyphase interpolators) which operate in a fixed mode where the sample rate is either decreased or increased by a factor equal to the number of polyphase partitions. The IP cores, are considered to be the future approach to FPGA solutions [82], currently lack advanced multirate signal processing engines that allow several functions to be embedded within a single filter. These functions include embedded sampling rate changes that allow any rational sample rate change while simultaneously performing filtering operations, combined arbitrary resampling, match filtering, and symbol timing recovery, etc. As a natural consequence, we therefore formulated our hypothesis to address this missing work.

1.8 Thesis formulation

So far we have discussed many technical aspects of modern radio receivers, and we have concluded that traditional legacy design can be beneficially replaced by Software Defined Radios. In order for such devices to operate efficiently, both from a numerical robustness prospective and from a computational complexity point of view, we have to map the associated signal processing algorithms onto advanced HW/SW platforms, which is not necessarily a trivial task. As a consequence, we therefore realized the need for giving sound answers to the following two questions:

1. How do we map the polyphase filter banks onto reconfigurable HW/SW platforms, which normally implies a huge solution space exploration, such that the cost function metrics are simultaneously minimized?

2. Is it possible to embed additional signal processing functions into the polyphase filter bank, and if so, how is the mapping process efficiently performed with respect to the cost function?

Based on these questions, the thesis is formulated as:

Is it possible to design polyphase filter bank engine(s) performing multiple functions for a SDR DFE while achieving cost effective mapping to reconfigurable platform in terms of Area, Time, operating clock Speed, and Power consumptions?

In order to investigate this thesis, the following goals have been defined:

1. to explore the design space by taking advantage of the reconfigurable target architecture and

2. to carry out mapping experiments in order to gain sufficient experiences that I can formulate a set of structured design guidelines.

1.9 Outline of the Dissertation

The remainder of this dissertation is organized as follows: First, the methodology for the design space exploration for polyphase filter bank based SDR DFE is described in Chapter 2. In Chapter 3, the summary of our contributions is given. This is followed by the conclusion and outlook in Chapter 4. Chapter 5 gives an overview of the papers A-F which are the contributions to the dissertation. We will refer to these papers by giving the letter in brackets, e.g., [A]. Citations of other references are numbers within brackets, e.g., [1].

2 Design Space Exploration of Polyphase Filter Banks for High Performance SDR Front-Ends

In chapter 1, the enhanced algorithms/ techniques based on polyphase filter banks for designing DFE are presented. A general M-path polyphase filter operating at M times the reduced rate than the input sampling rate, still uses the same number of hardware resources such as registers, and multipliers (the exception being the number of adders). The M paths in an M-path polyphase filter provide parallelism which can be exploited in a number of different ways that will be discussed in this chapter. The FPGA takes advantage of this inherent parallelism in the algorithm. Furthermore, the rich architecture of modern FPGAs provides a large number of choices to map a design, which creates a huge design space. It is therefore of great importance to analyze the building components of system (polyphase filter bank) functionalities with the purpose of determining their possible performance and the cost of applying different processing elements which are present in the FPGA platform. This task is called Design Space Exploration (DSE). The best design mapping fulfils the requirements and constraints while minimizing the cost function which can be either one or a combination of area, time, power, numerical precision, implementation effort, etc. The mapping often requires several iterations which leads to a longer development time. This not only costs more money, but also increases the time to market. This chapter presents the detailed analysis of the polyphase filter (bank) structures and the step-wise refinements in their hardware architectures. The proposed architectures form an essential part of the contribution in this thesis. The outcome of this DSE and its step-wise refinements can lead to the development of Intellectual Property (IP) cores for polyphase filter (banks) that will be beneficial for SDR front-end research and development.

2.1 Polyphase Filter Structures

An M-path polyphase filter consists of M parallel subfilters. The data is fed by a rotary switch called a commutator and the output is taken after the DFT (FFT) operation. Fig. 2.1a shows the M-path polyphase filter where the M-point DFT is replaced by a M-input adder. The commutator is a M-to-1 sampler or a demultiplexer, feeding data to each of the subfilters that operate at M-times the reduced rate as compared to the incoming sampling rate. The DFT block constructs the individual channels from the down-sampled data. The M-path parallelism in a M-path polyphase filter can be exploited to achieve several possible structures such as symmetric structures, and serial polyphase structure with serial and/or parallel MACs.

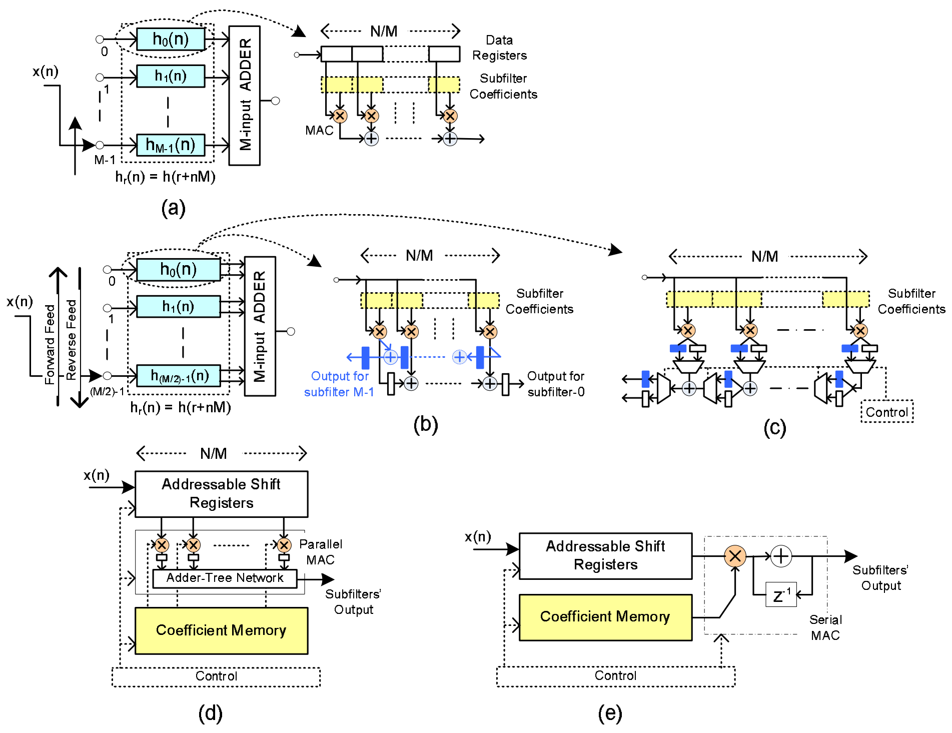


Figure 2.1: Different architectural structures of a polyphase filter: (a) General Polyphase Structure, (b) Symmetric Structure, (c) Adder Shared Symmetric Structure, (d) Serial Polyphase Structure with Parallel MAC, and (e) Serial Polyphase Structure with Serial MAC.

The general polyphase filter structure has M subfilters of length N/M, where N is the length of a non-partitioned low pass filter, each operating at $1/M$ times the input rate as shown in Fig. 2.1a. The symmetric structure utilizes the symmetry of the coefficients in the subfilters, and therefore reduces the number of subfilters and commutator length to $M/2$ [83]. The commutator moves in both the forward and reverse feed directions as shown in Fig. 2.1b. In this structure, the multipliers are shared by the two subfilters. The adders can also be shared by using multiplexers and de-multiplexers to form an adder shared symmetric structure as shown in Fig. 2.1c. The clock requirements for the symmetric and the adder shared symmetric structures are doubled i.e., $(f_s/M) \times 2$, as compared to the general polyphase structure because of the subfilter sharing and reduced commutator length. In a M-path polyphase filter, only one subfilter is processed at a time and the remaining M-1 subfilters are idling. The serial polyphase structure takes advantage of this feature and merges the subfilter's data registers and coefficients to form banks of data registers and coefficients, respectively. These banks are addressed by a control sequence to select the desired subfilter's data registers and coefficients to perform MAC operations. The MAC operations can be performed in parallel or in a serial fashion to form a serial polyphase structure with parallel MAC or with serial MAC [84] as shown in Figures 2.1d and 2.1e, respectively.

2.2 Comparative Analysis

A comparative analysis in terms of resource complexities and clock rates is performed for a N tap prototype filter which is partitioned into a M-path polyphase filter, each path with N/M taps, for the following structures: (i) general polyphase structure, (ii) symmetric structure, (iii) adder shared symmetric structure, (iv) serial polyphase structure having serial MAC, and (v) parallel MAC. Their resource complexities and the respective required operating clock rates are tabulated in Table 2.1. A trade-off is seen between the complexities versus the processing clock rates. The serial polyphase with serial MAC shows the least complexity but demands a high clock rate of $(f_s/M) \times N$, whereas the serial polyphase with parallel MAC has a slightly higher complexity but operates at a input clock rate of f_s. Similar analyses of these complexities for a 30-tap filter (partitioned into 5-path polyphase filters, each path with 6 taps), and a 960-tap filter (partitioned into 48-path polyphase filters, each path with 20 taps) are presented in our papers [B] and [D], respectively.

Table 2.1: Comparative analysis in terms of resource complexities and clock rates for general polyphase structure, symmetric structure, adder shared symmetric structure, serial polyphase structure having serial MAC, and parallel MAC.

	General Structure	Symmetric Structure	Symmetric Structure (Adder Shared)	Serial Polyphase (Serial MAC)	Serial Polyphase (Parallel MAC)
Multipliers	N	N/2	N/2	1	N/M
Adders	((N/M)-1)M	((N/M)-1)M	((N/M)-1)M/2	1	(N/M)-1
Registers	N	N	2N	N+1	N+2(N/M)
Multiplexers	0	0	N-(M/2)	(N/M)+1	N/M
Demultiplexers	0	0	M/2	0	0
Clock Rate	f_s/M	(f_s/M)2	(f_s/M)2	(f_s/M)N	f_s

Any of these structures can be selected depending upon the allowed complexity and clock rate. The resource complexity of a general M-path polyphase filter is the same (except for adders) as for a conventional N-tap FIR filter, but each subfilter in the M-path polyphase filter operates at a reduced clock rate of f_s/M. This reduced clock rate operation leads to reduced dynamic power consumption which for the normally applied CMOS technology is proportional to the operating clock rate (toggle frequency), as described by the following equation [85]:

$$P_{dynamic} = nCV^2 f \tag{2.1}$$

where *n* is the number of nodes being toggled, *C* is the load capacitance per node, *V* is the supply voltage, and *f* is the toggle frequency. *C* and *V* are device parameters whereas *n* and *f* are design parameters. *V* can also be a design parameter for the voltage-scale scenario, but we do not consider it further.

On the other hand, the serial polyphase structure with parallel MAC has M times reduced resource complexity as compared to the general polyphase structure except for the number of registers which is the same for both, but at the same time it requires a M times higher clock rate (i.e., f_s for the subfilter's processing) as compared to f_s/M for the general polyphase structure. It is interesting to note that one parameter of the dynamic power equation i.e., *n* which is related to area is decreased while the other parameter *f* is increased. Since dynamic power is proportional to both of these two parameters, theoretically the general polyphase and the serial polyphase with parallel MAC have almost the same dynamic power consumption. So, the serial polyphase structure with parallel MAC reduces the complexity while operating at an input clock rate of f_s and having almost the same dynamic power consumption as the general polyphase structure.

A general polyphase structure where each subfilter operates at a reduced rate of f_s/M is a highly desirable solution for very high speed digital front applications as explained for an ultra wideband 1.6 GHz channelizer in [70]. The current technology platforms, FPGAs, are not capable of processing at GHz clock rates; however, the parallelism offered by multirate signal processing and the polyphase filter bank offers a filter structure where subfilters can be realized by current technology platforms while operating at a reduced clock rate. The highest clock rate supported by the latest FPGAs such as Virtex-7 is 700 MHz. However, the internal hardware architecture would limit the maximum operating frequency of the design. The unique feature of polyphase filter banks being mapped as a general polyphase structure is used in our paper [F], where a variant of a polyphase channelizer which accommodates the spectral shift of multiples of quarters of the channel spacing [30], is used as bandpass filters for a dual-standard (UMTS & WLAN) front-end. It splits the data into multiple paths, operates the subfilters at a lower rate than the input sample frequency, and eliminates the need for an IQ demodulator. Although the best use of the polyphase channelizer in this case would be to directly extract the individual channels of both standards rather than splitting the bands followed by separate channelizes. The approach uses a standard polyphase filter bank (modulated filter banks) where the (extracted) individual channel bandwidth is an integer multiple of the input sample rate. The advanced polyphase filter bank approaches mentioned by [65] which filter and simultaneously down convert multiple signals having arbitrary bandwidths and randomly located center frequencies, can be used for a highly efficient solution.

2.3 Subfilter Architectures

The basic block of a polyphase filter is a subfilter implemented as a FIR filter, with data and coefficient register banks in the case of a serial polyphase structure. There can be several different algorithms or techniques for implementing a FIR filter such as FIR filtering using parallel multipliers and accumulators, a bit-level systolic array, Distributed Arithmetic, Fast FIR algorithms, Frequency domain filtering, Multiplier-less FIR filter (SOPOT), etc. The most direct approach of performing a FIR filter's operation is by using parallel multipliers and accumulator (MAC). Today's FPGAs have dedicated multipliers (e.g., DSP48E slices in Xilinx FPGAs) which are designed especially for high speed and low power consumption. They are generally highly preferred for the filter's MAC operation instead of a CLB-based MAC.

The parallel MAC structure is illustrated in Fig. 2.2a, and is derived directly from the FIR convolution equation in [49]. In this structure, each tapped delay input is multiplied with the filter tap's coefficient in parallel and then computed outputs are accumulated together to get the filter's output. The structure in Fig. 2.2a has long combinatorial delays through the accumulation chain, so the adder tree network shown in Fig. 2.2b or the transposed form shown in Fig. 2.2c are often preferred for the implementation. Both these forms i.e., adder tree and transposed form can have pipelined accumulation chains to increase the performance. The transposed form allows the MACs to be placed in a linear systolic fashion that minimizes the routes and maximizes the performance of the design. The long routes in the adder stages of the tree network slow down the design's performance, as the number of MAC units are increased. In the transposed form of the filter structure, the input signal drives every MAC unit and requires a large fan-out as the number of MAC unit increases. To reduce this fan-out while utilizing a linear systolic array of MAC units and pipelined accumulation chain, an additional stage of pipelining can be introduced into the direct filter structure in both the inputs and outputs of each MAC, as shown in Fig. 2.2d. In all of these MAC structures the filter's tap-delay registers corresponding to a single FIR filter are used either as data delay registers or as MAC output delay registers. The resource utilization of this parallel MAC approach can be decreased by increasing the number of taps computed per MAC unit. This will decrease the required number of MACs and hence reduced area, but on the other hand this demands a high clock rate. This is the technique used in the custom VLSI chip described in [86, 87].

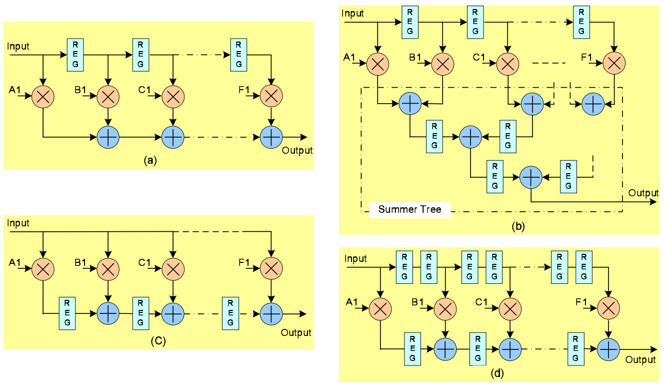


Figure 2.2: FIR filtering structure using parallel multipliers and adders: (a) direct form, (b) adder tree structure, (c) transposed form, and (d) pipelined transposed form.

2.4 Non-Maximally Decimated Systems

An M-path polyphase filter bank can be realized with a modified N-path polyphase filter to achieve the targeted output sample rate other than the maximally decimated mode [69] [6] [30]. Such systems allow resampling by arbitrary ratios while simultaneously performing baseband aliasing from the center frequencies at Nyquist zones that are not multiples of the output sample rate. This versatility of a polyphase filter bank is presented in our paper [A] by utilizing five different resampling factors. These resampling cases correspond to: (i) maximally decimated, (ii) under-decimated, (iii) over-decimated, and combined up- and down-sampled with (iv) single, and (v) multiple stride length scenarios. The resampling technique is based on sliding cyclic data loading interacting with cyclic shifted coefficient memory [69].

A non-maximally decimated polyphase filter bank, where the number of data loads is not equal to the number of M subfilters and processes M subfilters in a time period which is either less or greater than the M data loads. The embedded resampling in the polyphase filter banks requires architectural changes for the structures presented in Fig. 2.1 to meet the required time constraint. These changes mainly apply to structures which share resources such as symmetric structure and serial polyphase structures. A general polyphase structure is a fully parallel solution and is operationally not affected in non-maximally decimated modes because each subfilter is operating independently and M subfilters can be processed even within a single data input time period. However, it does require some changes in its state machine. Symmetric structures are limited to cases with an even number of polyphase partitions to make use of filter symmetry, which is not the case for serial polyphase structures. In our further discussion, we focus on the serial polyphase structure with parallel MAC for non-maximally decimated polyphase filter banks.

In the serial polyphase structure with parallel MAC, as shown in Fig. 2.1d, the data and the coefficients registers are merged to form respective banks and share the same MAC in a parallel configuration. The realization of the data register bank is challenging as it has to shift the new data element to the respective subfilter's tap-delay line and having access to all the taps of that subfilter at the same time. In the case of a maximally decimated system, where the down-sampling factor is equal to the polyphase partition, FIFOs can be used as delay lines to derive an optimal structure, as described in [88], operating at the input data rate. The non-maximally decimated systems (under-decimated, over-decimated, and combined up- and down-sampling with single and multiple stride length of the commutator), require a two dimensional memory based solution where only the targeted subfilter can be loaded with the input data.

2.5 Optimizations for Area Resources

There exist a number of design options depending upon the usage of the FPGA resources. Today's FPGAs have rich architectures with special memory resources such as distributed RAMs and block RAMs, and high performance computational resources such as DSP48E slices in addition to the basic CLBs. The DSP48E slice provides not only improved flexibility and utilization, but also reduced power consumption, increased operating frequency, and reduced set-up and clock-to-out time. The efficient usage of these dedicated resources can result in a high performance system in terms of achieving high operating clock rates and reduced CLB requirements.

A straightforward mapping of the polyphase partitioned data register bank is to use CLB slice registers to form individual subfilter's delay lines which are connected to multiplexers to form a register bank as shown in Fig. 2.3. This solution can be suitable for a small filter because even a small filter results in a relatively high resource utilization of the CLB's slice registers and LUTs. The resource utilization will scale linearly with the filter size.

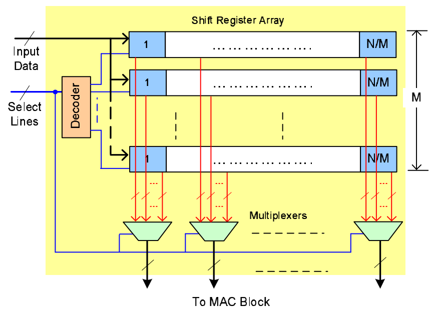


Figure 2.3: Data register bank formed by individual shift register arrays, multiplexers, and decoder.

In our papers [B] and [D] different design options for a register bank mapping on a Virtex-5 FPGA by exploiting its resources such as slice registers, RAM based shift register, distributed RAM, and block RAM are presented. Fig. 2.4 illustrates the mapping options for a 24×40 register bank [D] (40 rows and 24 columns register bank with 32-bits complex data register). Fig. 2.5 presents their resource usage in terms of the CLB's slice registers and LUTs. The straightforward mapping of a register bank which is the structure shown in Fig. 2.3 uses 47% and 21% of the slice registers and LUTs, respectively. This is only for real data and therefore the numbers would be double this for complex data. Next, each data register in the subfilters is replaced by a RAM-based shift register (SRL16 / SRL32 mode of the slice LUTs) and the resource utilization (for real data only) comes out to be 0% and 21% for slice registers and LUTs, respectively. This eliminates the need for the slice registers, but the LUT usage is the same as before. In a Virtex-5 FPGA, each CLB has 64-bit distributed RAM [89], but it is bit-addressable, so for 16-bit data, 16 CLBs are collectively used as a single 16-bit register. The rest of the 63 bits in each CLB remain unused. Based on this analysis, distributed memory is used such that each 64-bit memory contributes 1-bit to a 16-bit data element for 64 subfilters (only 40 are used here) as shown in Fig. 2.4. So the memory which was previously used for only one subfilter is now being used for all the 40 subfilters. The bit-width is further extended to 32-bits to incorporate complex data $(16 + 16$ bits), and the resource utilization dramatically decreases to 2% both for slice registers and LUTs (for complex data). This will also eliminate the need for the decoder and multiplexers to select the desired subfilter's data elements in the case of a shift register based register bank. The same concept is further applied to block RAMs, which completely eliminates utilization of the CLBs. The block RAMs utilization for a 40×24 register bank is 12 block RAMs, which corresponds to 9% utilization for a Virtex-5. A similar analysis for a 5×6 register bank is presented in our paper [B].

The distributed RAMs and block RAMs can efficiently replace each column of the data register bank and their cascade can form a complete register bank as shown in Fig. 2.6. This reduces the resource usage but at the same time it requires loading of the new data

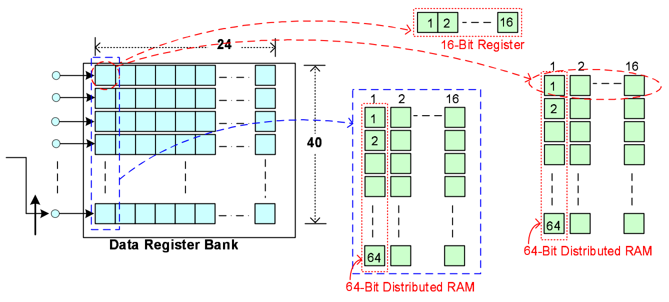


Figure 2.4: Data register bank mapping options in the Virtex-5 FPGA.

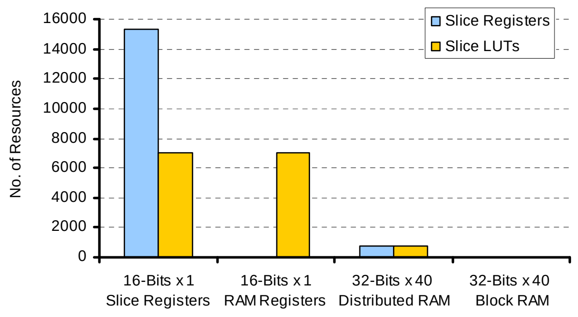


Figure 2.5: 40×24 register bank resource usage by using slice registers and LUT, RAM registers, distributed RAM, and block RAMs.

element to memory and shifting the previous data to the next cascaded memory stage.

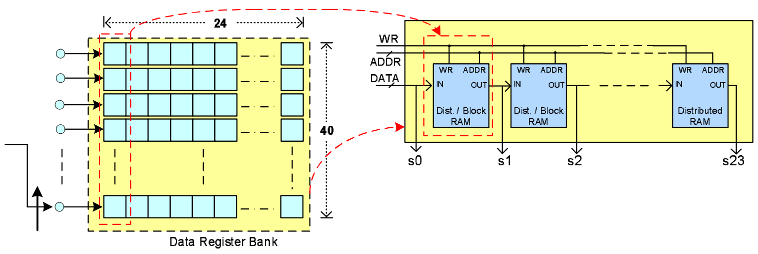


Figure 2.6: Realization of complete register bank by using cascade memories (distributed RAMs or blockRAMs).

A block RAM based register bank which does not require any CLB resources needs an extra clock cycle for each data load and shift, as described in our paper [D]. The reason is that the data shift in the subfilters requires data to be available from the preceding memory which requires one clock cycle to read the data element. The next clock cycle will load the new data element along with shifting of the subfilter. This is due to the fact that block RAMs have synchronous write and read operations. On the other hand, a distributed

RAMs based register bank has no extra clock cycle penalty for each data load and shift (provided that there is no input/output register) as such memories have asynchronous read and synchronous write operation. The filter coefficient bank can also be realized by the resource options presented in Fig. 2.5. However, it does not require the shifting as was necessary for a data register bank.

2.6 Architecture Exploration

Among the embedded resampling scenarios presented in our paper [A], the cases where the number of data loads is less than the number of subfilters (M) which are under-decimated and combined up- and down-sampling with single stride length of commutator, the run-time processing at the input sample rate f_s for serial polyphase structure with parallel MAC is not possible. One solution is to operate the polyphase engine at a higher clock rate and interface to the input data at a lower rate by using a FIFO.

A Load-Process Architecture (LPA) with block RAM based register bank (LPA-BR) for a polyphase filter bank operating in the under-decimated mode is presented in our paper [D]. It has 48- and 40-path polyphase filter banks for up- and down-sampling channelizers, respectively. Fig. 2.7 shows the architecture for the down-sampling channelizer with an output accumulator instead of an FFT, representing the channel at the baseband. It uses a common address bus for all the cascaded block RAMs which corresponds to individual taps of the subfilters, whereas the 48-path up-sampling channelizer requires a distributed address bus (5 distinct address pointers) for its 20 cascaded block RAMs to achieve its required circular data shift [D].

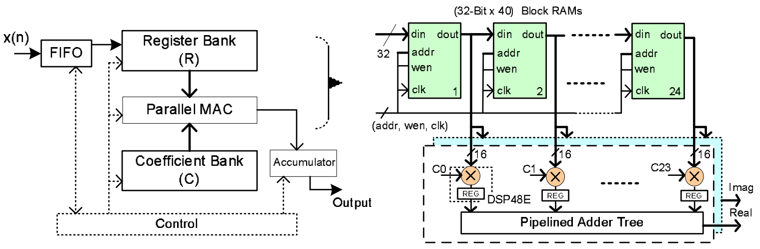


Figure 2.7: Load Process Architecture with block RAM based register bank for a down-sampling polyphase filter operating in under-decimated mode, running at a clock rate higher than the input data rate.

During the load phase, the data elements are fed from the input FIFO and loaded to the subfilters as directed by the data pointer. In the processing phase, coefficient and data pointers having embedded shifts, select the coefficient and data memory elements to perform MAC operations. The multipliers are based on DSP48E slices whereas the additions are performed by using a CLB based adder tree network. The architecture loads the register bank with N data elements from the input FIFO and then process M subfilters while the FIFO is being loaded. As the FIFO is continuously loaded and unloaded only during the process cycles, shown in Fig. 2.8a, the FIFO eventually overflows. In order to overcome the FIFO's overflow and to meet the output time constraints, the polyphase filters need to run at high clock rates, which are 4 times the input clock rates i.e., 4×200 MHz and 4×240 MHz for the up- and down-sample channelizers, respectively. These high clock rate demanding designs are not only difficult to achieve but also the architectures become power hungry, since the dynamic power consumption is proportional to the toggle frequency. The high clock rates for the LPA may limit the use of polyphase filter designs to lower input sample rates.

A reduced sized filter bank channelizer, specifically a 5-path polyphase filter with each path being 6 taps is presented in our paper [A] for different embedded resampling cases and their architectural exploration is performed in our paper [B]. The block RAM based LPA for the embedded resampling cases presented in our paper [B] shows that on the average approximately 60% of the time is used for loading the register bank from the input FIFO, which eventually demands high operating clock rates to process the loaded data within the output time constraints. The high clock rate demands of LPA-BR can be reduced by using distributed RAM based register banks and by applying different schedules such as interlaced load-process and runtime scheduling. The use of distributed RAM based register banks in the LPA, reduces the clock rate by 20 to 25% as compared to their block RAM based counterparts [B][D].

In LPA with distributed RAM based register bank (LPA-DR), the scheduling of the load and process cycles can also be improved to further reduce the required operating clock rates. The idea is to interlace the load process architecture such that the processing of the subfilters (process-cycle) starts along with the data loading from the FIFO (load-cycle), instead of waiting for the complete loading of the data-loaded (targeted) subfilters before processing. In this case, the process-cycle continues for M subfilters, but the data loading is stopped for non data-loaded (non-targeted) subfilters. This is the only time when input data is pushed onto the FIFO without being pulled at the same time. Since the number of non-targeted subfilters is less than the number of targeted subfilters, the FIFO does not grow as rapidly as in the previous case where process-cycle starts after completing the load-cycle as shown in Fig. 2.8a. This interlaced scheme for loading and processing as shown in Fig. 2.8b reduces the overall clock cycles count and therefore reduces the required clock rate.

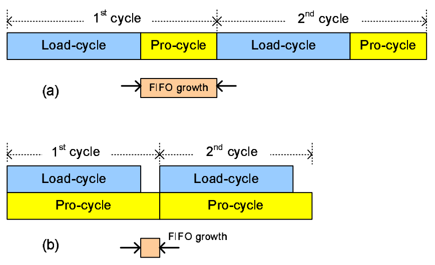


Figure 2.8: Scheduling of load- and process-cycles for (a) LPA and (b) interlaced LPA.

By applying the interlaced scheme to LPA-DR in our paper [C], the required clock rate can be reduced to $2\times$ (twice the input data rate) as compared to $3\times$ and $4\times$ clock for its LPA-DR and LPA-BR counterparts. In an interlaced LPA, most of the process cycles which are overlapped by the load cycles are idling as they are waiting for the next available data from the input FIFO. To make efficient use of these idling cycles of the $2\times$ clock, our paper [B] introduces a Run-time Architecture (RA) where non-targeted subfilter's processing is achieved by idling cycles of the $2\times$ clock instead of processing them after the load cycle. The idea is to operate at twice as high a data clock rate, process the input data while simultaneously loading the subfilters on the first cycle of the $2\times$ clock and the next cycle of the $2\times$ clock is used to process the non-targeted subfilter. By using this scheme, the non-targeted subfilters get processed within the time frame of the targeted subfilters and therefore reduce the overheads as in the case of LPA and interlaced LPA. It also eliminates the need for the input FIFO. The RA scheduling for the embedded resampling cases are presented in our paper [B] and the RA is shown in Fig. 2.9. It uses a distributed RAM based register bank (without input/output registers), a block RAM based coefficient bank, and a MAC based on a systolic array of DSP48E slices.

The parallel data from the data register is delayed by a set of delay elements to align the subfilter's MAC and final accumulation process within the systolic array MAC as shown in Fig. 2.9. The corresponding delay for the coefficient register bank can efficiently be achieved by offset addressing, instead of using a delay element network. The RA with a DSP48E systolic array based MAC has a large latency due to the pipeline and delay registers, but has an enhanced maximum operating clock rate. The multiplexer block prior to the delay elements switches between the targeted subfilter's and non-targeted subfilter's processing, which is used only in under-decimated and combined up and down sampling with single stride length.

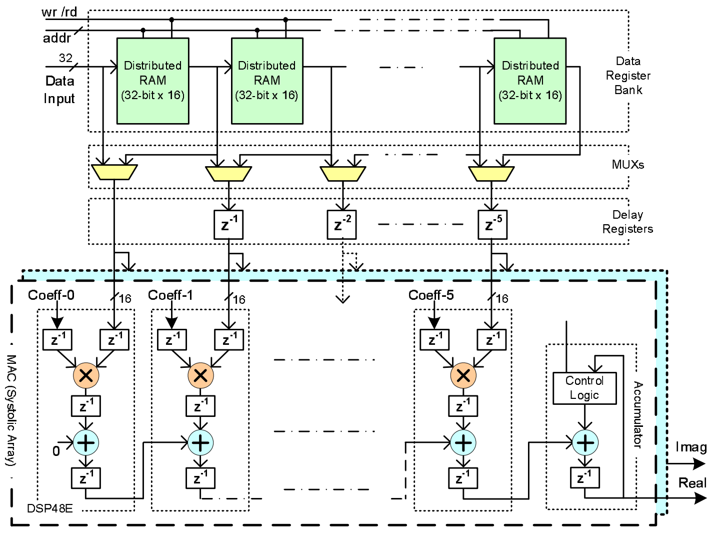


Figure 2.9: Runtime Architecture with a DSP48E systolic array based MAC. The data from the register bank is fed through a set of delay elements that delay the parallel data element by element to time align the subfilter's MAC and final accumulation process.

Among the architectures presented in our paper [B], the LDA-BR, LDA-DR, and RA use

DSP48E slice multipliers and a CLB based adder tree for their MACs, whereas RA with a DSP48E systolic array (RA-DSP48E) uses a DSP48E slice based systolic array MAC. All these cases with different configurations use a block RAM based coefficient bank. The resource usage of FPGA mapping for the five embedded resampling cases for their LPA and RAs are presented in [B], in terms of slice registers, slice LUTs, dedicated resources, required operating clock rates, and dynamic power consumptions. The RAs turn out to be a preferred choice to reduce operating clock rates and reduce dynamic power consumption. Using RA is also a preferred means to reduce area resources with the exception of some cases (under-decimated and combined up- and down-sampling with single stride length of commutator) which use multiplexers and pre-stored indices in the LUTs for addressing their data and coefficient banks. RA slightly increases the area but on the other hand, operates at a reduced clock rate as compared to their variants.

The coefficient banks which are based on block RAMs for these architectures can also be replaced by distributed RAMs. In this case, the block RAM resources which were previously used by coefficient banks are unnecessary and the bits occupy additional CLB resources.

2.7 Optimization for Enhanced Clock Speed Performance

Fig. 2.10 shows an analysis for the best case achievable clock rate for the architectures for the five cases of embedded resampling presented in our paper [B]. These values are generated under balanced performance design goal and strategies in the Xilinx ISE tool [90]. This tool optimizes the design's place and route to meet the specified timing constraint. All the cases presented have different timing constraints except for the RAs operating at $2\times$ clock rate. These targeted timing constraints are further tightened to obtain the maximum possible optimization by the tool to achieve the higher clock rates for these architectures. The red dotted horizontal line on each bar graph in Fig. 2.10 shows the timing constraint of the corresponding case. All the timing constraints are met except the one for case 4, which is the LPA with block RAM based register bank. The LPA-BR, LPA-DR, and RA (in balanced design goal and strategy in Xilinx ISE tool) can execute at up to 200 MHz whereas RA-DSP48E (systolic array) can execute at up to 350 MHz. This speed enhancement results from the systolic array of a DSP48E slice based MAC. All these architectures have a block RAM based coefficient bank. The clock speed can be further enhanced by using either distributed RAM or pipelined block RAM based coefficient banks to 300 MHz and 400 MHz for the adder tree and the systolic array MAC architectures, respectively.

The RA for the up- and down-sample polyphase filters presented in our paper [C], operating at $2\times$ the data clock rate, require processing at 400 and 480 MHz, respectively to meet their output time constraints. The asynchronous path between the cascaded distributed RAMs (without input/output registers) turns out to be a bottleneck with respect to achieving higher clock rates. In our paper [C], it is shown that as the number of cascaded distributed RAM (without input/output register) increases, their maximum operating clock rate decreases. On the other hand, a minimum of 500 MHz performance can be achieved even with up to 24 cascades of distributed RAMs having output registers. The

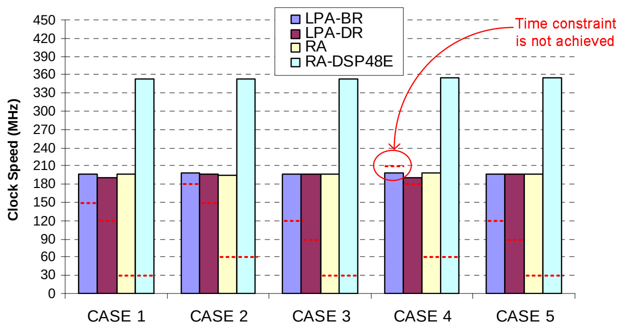


Figure 2.10: Best case achievable clock rate performance for the five embedded resampling cases; maximally decimated system (CASE 1),Under-decimated system (CASE 2), Over-decimated system (CASE 3), Combined up- and down-sampling with single stride length (CASE 4) and with multiple stride length (CASE 5).

insertion of an output register for each cascaded distributed RAM forms a pipelined register bank that allows high operating clock rates, but at the same time the added registers require balancing of the control and data paths and also one tapped-delay shifting for the cascaded distributed RAM (pipelined). These are achieved by inserting the pipeline registers for the control path (i.e., address and read/write signal), and by delaying each memory output path by the total number of cascaded distributed RAMs minus its tap number, as shown in Fig.2.11. The added pipeline register network will add additional latency which is a compromise to achieve high operating clock rates. Furthermore, the real and imaginary register banks are separated and use 16-bit memories instead of a combined (real and imaginary) data register bank using 32-bits memories. The separate memory banks allow the tool's place and route process to meet the timing constraints of 480 MHz. The Run-time Architectures are capable to meet the challenging demands for a modern SDR front-end in terms of minimal area, time (clock frequency), and power requirements.

2.8 Explored Solution Space

Finally, the explored solution space for designing polyphase filter banks operating in non-maximally decimated modes is presented by a complete flow diagram shown in Fig. 2.12. The flow diagram wraps up every thing we have presented previously and helps to give the answer to our initial thesis. The processing engines for the selected serial polyphase filter structure can be realized by Load-Process and Runtime Architectures. Their internal building blocks i.e., data and coefficient register bank can be mapped, in a straightforward manner, to CLB based slice registers or RAM based registers, which results in a high resource utilization of CLB resources even for a small sized polyphase filter. On the other hand, the data and coefficient banks can be mapped in a resource efficient manner to block RAM and/or distributed RAMs. The MAC architecture can be based on summer tree or systolic array consisting of DSP48E slices.

The design flow provides a mean to explore the trade-off between the targeted constraints

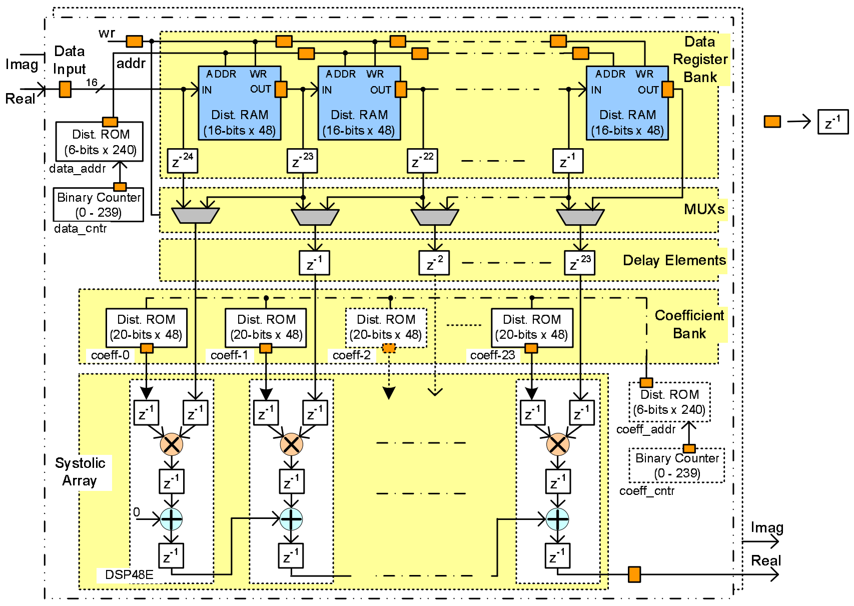


Figure 2.11: The complete architecture for the down-sample polyphase filter with 24 cascaded Dist. RAMs, delay/pipeline registers, multiplexers, systolic array of DSP48E slices, coefficients' distributed ROM, and data and coefficient address distributed ROM.

and the required FPGA resources. The explored architectures for the polyphase filter bank (for its serial polyphase with parallel MAC structure) followed by their step-wise refinements provide customized solutions for the designers to meet their targeted design constraints of area, time, and power. A comparison of area, time and power for the explored architectures and their customized solutions for a polyphase filter bank operating in five different resampling modes is presented in our paper [B]. The LPAs are the most power hungry solutions because of their requirements for high operating clock rates. The interlaced LPAs provide significant improvements to LPA by reducing the clock rates. The RA while operating at maximum of two times the input rate provides the least power consumption solutions. Furthermore, it is seen from Fig. 2.12 that the design customization results in wide range of operating clock rate performance from 200 MHz to 500 MHz. The RA supports the designer's requirements of minimal area, power, and operating clock rates, while at the same time being capable of achieving high clock rate performance.

Furthermore, an analysis is performed for latency and throughput of the presented architectures. The throughput is constrained by the output sample rates so that it is same for all the architectures under the same resampling case; however, they vary in terms of their latencies. A detailed latency analysis is tabulated in Tables 2.2 to 2.6 representing the five different embedded resampling cases for the polyphase filter bank. The previous time calculation for the load process architectures as presented in our paper [B], are carried out on the control signals from their state machines, i.e., controlling the load and process cycles. The loading time was calculated from the non-empty signal from the FIFO, but

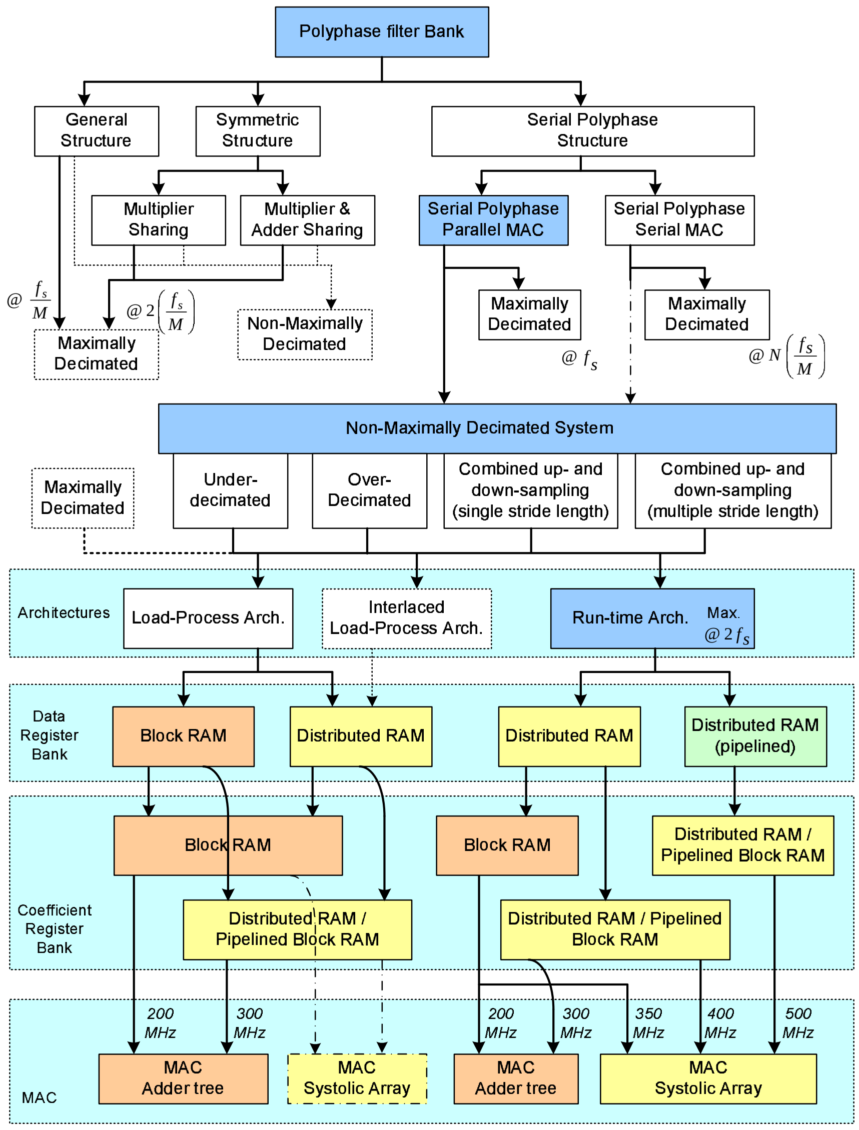

Polyphase filter Bank
General Structure
Symmetric Structure
Serial Polyphase Structure
Multiplier Sharing
Multiplier & Adder Sharing
Serial Polyphase Parallel MAC
Serial Polyphase Serial MAC
$@\frac{f_s}{M}$
Maximally Decimated
$@2\left(\frac{f_s}{M}\right)$
Non-Maximally Decimated
Maximally Decimated
$@f_s$
Maximally Decimated
$@N\left(\frac{f_s}{M}\right)$
Non-Maximally Decimated System
Maximally Decimated
Under-decimated
Over-Decimated
Combined up- and down-sampling (single stride length)
Combined up- and down-sampling (multiple stride length)
Architectures
Load-Process Arch.
Interlaced Load-Process Arch.
Run-time Arch.
Max. $@2f_s$
Data Register Bank
Block RAM
Distributed RAM
Distributed RAM
Distributed RAM (pipelined)
Block RAM
Block RAM
Distributed RAM / Pipelined Block RAM
Coefficient Register Bank
Distributed RAM / Pipelined Block RAM
Distributed RAM / Pipelined Block RAM
200 MHz
300 MHz
200 MHz
300 MHz
350 MHz
400 MHz
500 MHz
MAC
MAC Adder tree
MAC Systolic Array
MAC Adder tree
MAC Systolic Array

Figure 2.12: Flow diagram based on the explored solutions for designing the polyphase filter banks operating in non-maximally decimated modes.

the FIFO loading time is not included. Similarly, the process time is calculated from the state machine's control signal, but it did not incorporate the processing delay through the MAC operations. These two timing parameters do not effect the results presented in our paper [B] in Table 2 and Table 3, but they do have an effect on the latency, which are now taken into consideration.

Table 2.2: Case 1 (Maximally Decimated): Latency and clock cycles distribution for different architectures.

	LPA-BR	LPA-DR	RA	RA (Systolic Array)
Clock Period (ns)	6.66	8.33	33.33	33.33
Latency - Time (ns)	246.7	258.3	333.33	466.7
Latency - Clock Cycles	37	31	10	14
	Clock Cycle Distribution			
FIFO Load/Unload +	24	19	-	-
Data Register Bank Load				-
State Machine's Overhead	2	2	-	-
(load and process)				-
Data Register Bank Read	1	-	-	-
MAC - Adder Tree				
Multiply	1	1	1	-
Adder Tree $(log_2(N/M))$	3	3	3	-
MAC - Systolic Array				
Multiply pipelined delay	-	-	-	3
Delay across Array $(N/M-1)$	-	-	-	5
Subfilter to be processed	5	5	5	5
M-input Accumulation	1	1	1	1
TOTAL	**37**	**31**	**10**	**14**

It can be seen from Table 2.2 to Table 2.6, LPAs have large latency which basically is due to the loading cycle. This eventually results in needing high clock rates to meet the targeted throughputs. The latency of LPA-BR is larger than the latency of LPA-DR because of the fact that block RAM takes one clock cycle to read and will accumulate over the subfilter's length during the loading cycle. Similarly block RAMs' read operation in the process cycle has a one clock cycle delay as compared to the case for distributed RAMs. The RA's simultaneously loading and processing of the inputs, results in reduced latency solutions. The latency for MAC operation is fixed (depending on the subfilter's length) and is the same for all the architectures except the systolic array based MAC which takes 4 extra clock cycles. Furthermore, RAs operating on $2\times$ clock rates take one clock cycle extra to switch between the targeted (data-loading) and non-targeted (non data-loading) subfilters' processing.

The RA with distributed RAM pipelined register bank, distributed RAM or pipelined block RAM based coefficient bank, and systolic array of DSP48E slices based MAC is a preferred choice in terms of enhanced performance and low latency for state-of-the-art SDR digital front-ends.

It was stated in chapter 1 that high performance radio designs (which is also true for

Table 2.3: Case 2 (Under-Decimated): Latency and clock cycles distribution for different architectures.

	LPA-BR	LPA-DR	RA	RA (Systolic Array)
Clock Period (ns)	5.55	6.66	16.66	16.66
Latency - Time (ns)	161.1	160/166.7	183.3	250
Latency - Clock Cycles	29	24/25	11	15
	Clock Cycle Distribution			
FIFO Load/Unload +	17	12/13	-	-
Data Register Bank Load				-
State Machine’s Overhead	2	2	-	-
(load and process)				-
Data Register Bank Read	1	-	-	-
Multiplexer	-	-	1	1
MAC - Adder Tree				
Multiply	1	1	1	-
Adder Tree $(log_2(N/M))$	3	3	3	-
MAC - Systolic Array				
Multiply pipelined delay	-	-	-	3
Delay across Array $(N/M-1)$	-	-	-	5
Subfilter to be processed	5	5	5	5
M-input Accumulation	1	1	1	1
TOTAL	**29**	**24/25**	**11**	**15**

Table 2.4: Case 3 (Over-Decimated): Latency and clock cycles distribution for different architectures.

	LPA-BR	LPA-DR	RA	RA (Systolic Array)
Clock Period (ns)	8.33	11.11	33.33	33.33
Latency - Time (ns)	308.3	333.3/322.1	366.7	500
Latency - Clock Cycles	37	30/29	11	15
	Clock Cycle Distribution			
FIFO Load/Unload +	24	18/17	-	-
Data Register Bank Load				-
State Machine’s Overhead	2	2	-	-
(load and process)				-
Data Register Bank Read	1	-	-	-
MAC - Adder Tree				
Multiply	1	1	1	-
Adder Tree $(log_2(N/M))$	3	3	3	-
MAC - Systolic Array				
Multiply pipelined delay	-	-	-	3
Delay across Array $(N/M-1)$	-	-	-	5
Subfilter to be processed	5	5	6	6
M-input Accumulation	1	1	1	1
TOTAL	**37**	**30/29**	**11**	**15**

Table 2.5: Case 4 (Combined up- and down-sampled with single commutator stride length): Latency and clock cycles distribution for different architectures.

	LPA-BR	LPA-DR	RA	RA (Systolic Array)
Clock Period (ns)	4.76	5.55	16.66	16.66
Latency - Time (ns)	147.5/119	150/122.2	183.3	250
Latency - Clock Cycles	31/25	27/22	11	15
	Clock Cycle Distribution			
FIFO Load/Unload +	18/12	15/10	-	-
Data Register Bank Load				-
State Machine's Overhead	2	2	-	-
(load and process)				-
Data Register Bank Read	1	-	-	-
Multiplexer	-	-	1	1
MAC - Adder Tree				
Multiply	1	1	1	-
Adder Tree $(log_2(N/M))$	3	3	3	-
MAC - Systolic Array				
Multiply pipelined delay	-	-	-	3
Delay across Array $(N/M - 1)$	-	-	-	5
Subfilter to be processed	5	5	5	5
M-input Accumulation	1	1	1	1
TOTAL	**31/25**	**27/22**	**11**	**15**

Table 2.6: Case 5 (Combined up- and down-sampled with multiple commutator stride length): Latency and clock cycles distribution for different architectures.

	LPA-BR	LPA-DR	RA	RA (Systolic Array)
Clock Period (ns)	8.33	11.11	33.33	33.33
Latency - Time (ns)	375/341.7	400/366.6	433.3/400	566.7/533.3
Latency - Clock Cycles	45/41	36/33	13/12	17/16
	Clock Cycle Distribution			
FIFO Load/Unload +	32/28	24/21	-	-
Data Register Bank Load				-
State Machine's Overhead	2	2	-	-
(load and process)				-
Data Register Bank Read	1	-	-	-
MAC - Adder Tree				
Multiply	1	1	1	-
Adder Tree $(log_2(N/M))$	3	3	3	-
MAC - Systolic Array				
Multiply pipelined delay	-	-	-	3
Delay across Array $(N/M - 1)$	-	-	-	5
Subfilter to be processed	5	5	8/7	8/7
M-input Accumulation	1	1	1	1
TOTAL	**45/41**	**36/33**	**13/12**	**17/16**

other systems) need both the smart algorithms and their optimized or near optimized implementation on the processing platforms. Multirate signal processing based algorithms/ techniques for numerous DFE functionalities as explained in Chapter 1 allow low complexity design for DFE tasks which can further be implemented in a cost effective manner. Reducing sampling rate of the signal along with the bandwidth reduction (filtering) and signal processing by effective use of the aliasing process are the distinguishing features of multirate signal processing. Furthermore, instead of a normal FIR filter process, this alternative approach allows distributed processing over a number of subfilters (as a polyphase filter) and even at a reduced processing rate. These techniques should be adopted by the state-of-the-art radio transceiver designs and are highly recommended to the system designer of SDR based DFE designs including:

Single and multiple channel extraction and filtering,

With equal and unequal channel (spacing) distribution,

Embedded (rational) sample rate conversion,

Arbitrary sample rate conversion, and

Combining multiple tasks into a single processing filter.

A polyphase filter structure capable of combining multiple tasks (i.e., arbitrary resampling, match filtering, and symbol timing recovery) is presented in our paper [E].

This dissertation contributes to the DFE designs mainly in terms of HW architectures for polyphase filter banks to reduce area, time, power, and operating clock rates. Today designers often lack a structural methodology or fail to explore the design space and stick to their already known / experienced architecture design which may not necessarily be a minimal or a near optimal solution in comparison to multirate signal processing techniques. This dissertation paves the way towards resource minimized HW architectures for highly efficient SDR techniques based on polyphase filter banks. The design space for the FPGA platform was explored and through several experiments and step wise refinements, various architectural solutions were presented. These experiments and the resulting hardware architectures can serve as structural guidelines for the hardware designers to achieve their targeted constrained solution. Based on the explored design space presented in Fig. 2.12, our findings and recommendations for designers are:

1. A polyphase filter bank has M parallel paths which can be exploited in a number of different ways to meet the design constraints. Changing from a straightforward M parallel path implementation to a shared path with a single MAC approach provides a significant variation in resource usage and the required operating clock rates. The designer constraints of area, processing time, and operating clock rates determine the selection of the solution. The design of bandpass filters for a multi-standard receiver [F] has an input sample rate of 630 MHz. This is higher than the upper bound of the 550 MHz maximum possible clocking rate for Virtex-5 FPGA. This situation (where the input sample rate is higher than the platform's processing capabilities) calls for a fully parallel solution allowing each subfilter to process at a lower rate of $630/7 = 90$ MHz which is now within the operating limits of the target platform, but requires greater area resources.

2. The situation where the input sample rate is significantly lower than the platform's processing capabilities often favors a solution using resource sharing architectures. The resource sharing demands higher clock rates which should be achievable by the design and most importantly by the platform itself. The platform's operating clock rate and the design performance often limit the amount of resource sharing in the architecture. The designer can consider the trade-off between the serial polyphase with serial MAC that gives the least resource complexity but demands high clock rates, and the serial polyphase with parallel MAC that takes the advantages of operating at an input clock rate with a reasonable resource complexity.

3. The polyphase filter bank can embed rational sample rate conversion together with performing down-conversion and channel filtering. This is a very powerful feature of the polyphase filter bank that with no extra processing load, it can efficiently achieve the target output sample rate and eliminate the separate block for sample rate conversion. This approach is highly recommended for the SDR-DFE designers and they should use this embedded function of the polyphase filter bank whenever possible. Four re-sampling scenarios other than the maximally decimated system are presented. These scenarios presents situations where M subfilters are to be processed within a time period of either more than or less than M input samples. These scenarios lead to different schedulers for the subfilters' processing, which were considered for serial polyphase structure with a parallel MAC approach.

4. The resampling scenarios where M subfilters are to be processed in a time period less than the M input samples require higher processing rates to meet the output time constraint. Three different schedulers for subfilters' processing were presented based on stepwise refinements towards reduced operating clock rates. The Load Process scheduler requires more clock cycles to compute one output sample, which eventually requires much higher clock rates to compensate in order to meet the output time constraint. The Interlaced Load Process scheduler reduces the number of clock cycles by starting the subfilters' processing earlier than the complete subfilters' loading. It therefore significantly reduces the operating clock rates even down to 2x clock rates for the scenarios presented. The Runtime scheduler further optimizes the idling cycles of a 2x clock and uses these for non-targeted subfilters. The Interlaced Load Process scheduler and the runtime schedulers are recommended for the designer but not the load process scheduler, which turns out to be an inefficient approach. The Runtime scheduler results in a lower latency solution as compared to the interlaced load process scheduling. The Runtime scheduler has a simple state machine but non-trivial complicated scheduling that interlaces the processing of data loaded and non-data loaded subfilters, whereas the interlaced load process scheduler has a simple scheduler but a non-trivial complicated state machine. Furthermore, interlaced processing of data loaded and non-data loaded subfilters in Runtime scheduling results in reordering of the processed output samples, which is not the case for interlaced load process scheduler.

5. The FIFO based tapped delay line approach [88] for decimator, using the serial polyphase structure with parallel MAC approach, can only be used for a maximally decimated system. To achieve rational resampling other than the maximally decimated system in a serial polyphase structure, a two dimensional addressable

memory (as a data register bank) is required. Different mapping options for a data register bank were explored, these can enable the designer to achieve highly minimized area resource solutions. The designer can choose among solutions based on CLB registers, block RAMs, and distributed RAMs. The CLB based data register bank demands high area resource even for a small sized filter. The block RAM based register bank optimizes the area resources but requires an extra clock cycle to have data shifted among the cascaded block RAMs, which is not the case for a distributed RAMs based data register bank. The runtime scheduler utilizes a distributed RAM based data register bank which offers one clock cycle data shifting among the cascaded memories.

6. The mapping of a coefficient register bank is not as challenging as a data register bank since it does not require coefficient shifting among the taps. It requires segregated memories which can either be block RAMs or distributed RAMs. The coefficients assigned to each tap's processing are usually not so large, thus the distributed RAM based is the preferred solution. The designer, however, can choose between both of these resources, but should take their access (response) time into account. The memory response time can be a bottleneck to achieve a high operating clock rate. The distributed RAM has a faster response time than the block RAM, unless it is pipelined which will then take 2 clock cycles longer to read than distributed RAM read operation. For low latency and high system performance solution, a distributed RAM based coefficient bank is recommended. The extra clock cycles required by block RAM can be compensated for by utilizing an advanced read operation.

7. We have considered the serial polyphase structure with parallel MAC approach because of its reasonable resource complexity while operating at the input clock rate as compared to the other solutions that were explored. The serial polyphase with parallel MAC approach has a MAC unit for N/M inputs. Two different MAC approaches (i.e., adder tree and systolic array approaches) were considered. The adder tree approach uses DSP48E slice based multipliers and CLB based adders, whereas the systolic array approach uses only DSP48E slices. The adder tree based MAC has a latency of $1 + log_2(N/M) + M$ clock cycle, where as systolic array based MAC has a latency of $3 + ((N/M) - 1) + M$ clock cycles. The systolic array based MAC approach has an extra latency of 4 clock cycles, but enhances the system's performance allowing it to achieve a higher operating clock rate. This is, again, a trade-off for the designer between the reduced latency and the achievement of higher clock rates.

8. The delay across the cascaded memory structure for the data register bank is deemed to be a bottleneck with regard to achieving higher operating clock rates. The block RAM based cascaded memory structure has a lower operating clock rate performance because of the large clock-to-data-out time, as compared to a cascaded structure with distributed RAMs. It was further analyzed that the path from the output of one distributed RAM (without an output register) to the input of next distributed RAM is an asynchronous path that also leads to a bottleneck with regard to achieving higher operating clock rates. This path can be made synchronous by inserting an output register between the cascaded distributed RAMs, but this will

effect a one cycle shift operation among the cascaded structure required for the run-time scheduler. The delay caused by the insertion of output registers is balanced by inserting pipeline registers both in the control and output paths as explained in our paper [C]. This adds additional latency to the system but, at the same time, is highly recommended to the designer in order to achieve operating clock rates of up to 500 MHz on the Virtex-5 platform.

9. A pipelined data register bank based on distributed RAM is used for enhanced operating clock rate performance. The designer, however, can use a similar approach for a block RAM based data register bank. In such a case the block RAM should be used with output and pipelined registers. This requires doubling the balancing registers for the control and the output data path and will result in doubled latency as compared to the pipeline register bank solution with distributed RAMs.

10. The FFT algorithms Radix-2 and Radix-4 are commonly used for DFT calculation. The Winograd Fourier Transform offers reduced complexity solution by eliminating twiddle factors and complex multipliers. However, it depends on the transform sizes being split into relative prime numbers. It is highly recommended for designers to use this reduced complexity transform whenever allowed by the transform size.

11. In a Runtime scheduler for cases where M subfilters are to be processed in a time period of more than the M input samples (case-3 and case-5), the twice loaded subfilters are processed twice as well. However, the last M processed outputs are valid outputs of the subfilters. The previous output cycles can be masked so that they are not processed further. This can avoid some unnecessary processing resulting in (dynamic) power saving.

3 Summary of Contributions

This chapter provides a summary of the contributions of this work. The summary is related to the papers A-F.

The body of this dissertation is formed by the papers A-F. The summary of these papers together with authors' contributions are given:

Paper A:

Authors' Contributions:

***Mehmood Awan** contributed by proposing different resampling scenarios within polyphase filter bank. He derived five different cases basically by extending the previous work of fred harris. He performed all the MatLab simulations, and interpreted their results. He wrote the entire draft version of the paper, and revised it according to co-authors' comments.*

***Yannick Le Moullec** provided the suggestions to improve the draft version of the paper, and to link the paper to the other paper presenting the hardware architectures.*

***Peter Koch** contributed by interacting with Mehmood Awan on the scientific aspects of defining the resampling cases. He provided suggestions for layout and presentation of the draft version of the paper.*

***fred harris** provided the basic DSP technique for resampling in polyphase filter banks; which is further extended to five general re-sampling cases by Mehmood Awan. fred harris reviewed the final version of the paper.*

This paper presents efficient processing engines for Software Defined Radio (SDR) front-ends. These engines use a polyphase channelizer for arbitrary sample rate changes, frequency selection, and bandwidth control. This paper presents an M-path polyphase filter bank based on a modified N-path polyphase filter. Such a system allows resampling by arbitrary ratios while performing baseband aliasing from center frequencies at Nyquist zones that are not multiples of the output sample rate. This resampling technique is based on sliding cyclic data load interacting with cyclic-shifted coefficients. A non-maximally-decimated polyphase filter bank (where the number of data loads is not equal to the number of M subfilters) processes M subfilters in a time period that is less or greater than the M data loads. A polyphase filter bank with five different resampling modes is used as a case study for embedded resampling in SDR front-ends. These modes are (i) maximally decimated, (ii) under-decimated, (iii) over-decimated, and combined up- and down-sampling with (iv) single stride length, and (v) multiple stride lengths. These modes can be used to achieve any required rational sampling rate change in an SDR front-end based on a polyphase channelizer. They can also be used for translation to and from arbitrary center frequencies that are unrelated to the output sample rates.

Paper B:

Authors' Contributions:
Mehmood Awan *contributed by proposing different HW architectures for performing resampling within polyphase filter banks. He derived all the presented solutions and performed their architectural analysis and comparison. He performed the Design Space Exploration (DSE) experimentation, their RTL designs and associated simulations, and interpreted the results. He entirely wrote the draft version of the paper, and revised it according to co-authors comments.*
Yannick Le Moullec *contributed with scientific and technical feedback on HW architectural designs and analyses. He suggested grammatical and presentation improvements to the draft version of the paper.*
Peter Koch *provided feedback concerning the presentation and layout, as well as relation and mutual impact among the scientific experiments and results. He suggested technical, organizational, wording, and illustrative refinements to the draft version of the paper.*
fred harris *interacted with Mehmood Awan on multirate signal processing techniques and their benefits for low complexity solution. He reviewed the final version of the paper.*

In this paper, we describe resource-efficient hardware architectures for software defined radio (SDR) front-ends. These architectures are made efficient by using a polyphase channelizer that performs arbitrary sample rate changes, frequency selection, and bandwidth control. We discuss area, time, and power optimization for field programmable gate array (FPGA) based architectures in an M-path polyphase filter bank with modified N-path polyphase filter. Such systems allow resampling by arbitrary ratios while simultaneously performing baseband aliasing from center frequencies at Nyquist zones that are not multiples of the output sample rate. A non-maximally decimated polyphase filter bank, where the number of data loads is not equal to the number of M subfilters, processes M subfilters in a time period that is either less than or greater than the M data-load's time period. We present a load-process architecture (LPA) and a runtime architecture (RA) (based on serial polyphase structure) which have different scheduling. In LPA, N subfilters are loaded, and then M subfilters are processed at a clock rate that is a multiple of the input data rate. This is necessary to meet the output time constraint of the down-sampled data. In RA, M subfilters processes are efficiently scheduled within N data-load time while simultaneously loading N subfilters. This requires reduced clock rates as compared to LPA, and potentially less power is consumed. A polyphase filter bank that uses different resampling factors for maximally decimated, under-decimated, over-decimated, and combined up- and down-sampled scenarios is used as a case study, and an analysis of area, time, and power for their FPGA architectures is given. For resource-optimized SDR front-ends, RA is superior for reducing operating clock rates and dynamic power consumption. RA is also superior for reducing area resources, except when indices are pre-stored in LUTs.

Paper C:

Authors' Contributions:

Mehmood Awan *contributed to defining the overall problem and proposed the solution for high speed processing HW architecture for resampling polyphase filter bank. He derived the solution, and performed all the RTL analysis and simulations. He identified relevant performance metrics and interpreted the experimental results from the Design Space Exploration (DSE) experimentation. He wrote the entire draft version of the paper, and revised it according to co-authors comments.*

fred harris *reviewed the final version of the paper.*

Peter Koch *interacted with Mehmood Awan on design space exploration (DSE) experimentations and provided feedback on the derived results. He suggested general, technical and grammatical improvements for the draft version of the paper. He also actively took part in the process of preparing the oral presentation for the conference.*

This paper presents the time and power optimization considerations for Field Programmable Gate Array (FPGA) based architectures for a polyphase filter bank channelizer with an embedded square root shaping filter in its polyphase engine. This configuration performs two different re-sampling tasks required for spectral shaping and for an M-channel channelizer. In an under-decimated (non-maximally decimated) polyphase filter bank scenario, where the number of data-loads is less than the number of sub-filters, the serial polyphase structure with parallel MAC approach requires a larger processing time than the corresponding data-load time. In order to meet the output time constraint, the serial polyphase structure with parallel MAC has to run at a higher clock rate than the data input rate and hence potentially consumes high power. In contrast to the Load-Process Architecture (LPA), a Run-time Architecture (RA) operating only at twice the input data rate is presented which efficiently schedules the sub-filter's processing within the data-load time. The RA offers time and power efficient structure for the presented up- and down-sample polyphase filters utilizing 9% and 11% slice LUTs and 10% and 13% slice register resources of a Xilinx Virtex-5 FPGA, operating at 400 and 480 MHz, and consuming 1.9 and 2.6 Watts of dynamic power, respectively.

Paper D:

Authors' Contributions:

Mehmood Awan *contributed to defining the overall problem and proposed the solution to its HW architecture. He performed complexity analyses for the structures and the algorithms, design space exploration of FPGA resources and suggested a HW solution. He explored the options for mapping the register bank of polyphase filters to FPGA, and presented different scheduling schemes for FFT algorithm. He performed all the MatLab simulations and RTL analysis and simulations. He interpreted the results and wrote the draft version of the paper, which was revised according to co-authors comments.*

Peter Koch *provided scientific and technical feedback on HW architecture analysis*

and the interpretation of their results. He suggested the writing and expressive improvements for the draft version of the paper. He contributed in finalizing the oral presentation for conference.
***Chris Dick** reviewed the technical aspects of the final version of the paper.*
***fred harris** interacted with Mehmood Awan on core concepts and basic theory of DSP. He gave the idea of polyphase filter bank processing on vector processor, which together with the processing constraints for polyphase sub-filters led to load process architecture (LPA) by Mehmood Awan. He contributed on the FFT algorithms and provided feedback on their complexities. He reviewed the final version of the paper.*

The paper presents the architectural domain analysis for FPGA (Field Programmable Gate Array) implementation of a polyphase filter bank channelizer with an embedded square root shaping filter in its polyphase engine that performs two different re-sampling tasks required for spectral shaping and for M-channel channelizer. In terms of algorithms; Radix-2 FFT, Prime Factor and Winograd Fourier Transform are considered for IFFT, where as the polyphase filter is analyzed in terms of symmetric structure, and serial polyphase structures with serial and parallel MAC approaches. The computational workload for these algorithms and their implementation structures are presented together with their hardware mapping to a Virtex-5 FPGA by exploiting the inherent parallelism. Their resource utilizations and Design Space Exploration in terms of Area-Time is presented along with different optimization techniques.

Paper E:

Authors' Contributions:
***Mehmood Awan** contributed towards the design of a low-complexity synchronizer by proposing the solution using polyphase filter banks techniques. He derived the solution and performed MatLab simulations to verify it. He also performed the RTL analysis and simulations. He interpreted the simulation and experimental results with comments from the co-author. He completely wrote the draft version of the paper, and revised it according to the co-author's comments.*
***Peter Koch** provided general and scientific feedback on low complexity designs, and suggestions to improve the technical writing and readability of the draft version of the paper.*

This paper describes a low complexity multi-rate synchronizer that makes use of a polyphase filter bank to simultaneously perform matched-filtering and arbitrary interpolation for symbol timing synchronization in a sampled-data receiver. Arbitrary Interpolation between available sample points is achieved by selecting the appropriate filter in the bank having the polyphase partitioned matched filter, which provides the optimal sampling time. Two different structures are considered which are modified to perform combined arbitrary resampling. The computational complexity is analyzed to have the resource optimal solution. Simulation results are analyzed and their resource utilization for Virtex-5 FPGA implementation is presented.

Paper F:

Authors' Contributions:

Mehmood Awan *contributed to defining the overall problem and proposed a solution by using polyphase filter banks techniques. He derived the solution and performed the MatLab simulations to verify the filtering process. He interpreted the results and identified the performance criteria verses the complexity of the solution. He entirely wrote the draft version of the paper, and revised it according to the co-author's comments.*

Peter Koch *provided feedback concerning the derived solution, and writing improvement to the draft version of the paper. He provided suggestions towards the oral presentation at conference.*

The aim of this work is to design efficient bandpass filters for multi-standard software defined radios. Software Defined Radio (SDR) based applications which demand high sampling rate to eliminate the most of the analog components require high-performance technology and digital signal processing methods to handle and process the high sample rate data. State of the art technology such as FPGAs can support several hundred MHz of I/O data transfer rate. The internal hardware architecture, however, would limit the maximum operating frequency of the design. An alternative is therefore to use advanced DSP methods to overcome this bottleneck. Polyphase channelizers having spectral shifter to move the filter position are used in a scenario of dual standard (WLAN and UMTS) SDR receiver for the bandpass filtering. It will not only split the data in multiple paths to reduce the per arm data rate but the filter also operates at lower rate than the input sample frequency. Furthermore it can also eliminate the need of IQ demodulator.

4 Conclusion

This chapter gives the conclusion of the work, supporting the thesis. Furthermore, suggestions for future work are provided.

The high and constantly increasing demand for radio transceivers with enhanced functionalities, inter-standard portability, higher throughputs, compact physical size, low cost, and longer battery time open challenges for the radio designer. This requires both enhanced signal processing algorithms with reduced complexities and robust performance, and high performance HW/SW platforms for implementing these algorithms. It is not only the algorithms and the signal processing platforms that are required to meet there challenging demands, but we also need knowledge about optimizing or minimizing the architectural design that finally maps the algorithms to the processing platform. An algorithm can be more efficient in terms of its physical design by having an implementation that requires less resources, less processing time, and reduced power consumption.

SDR technology promises to provide a solution for meeting these demands by digitizing the radio signals as close to the antenna as possible and implementing the radio functionality as software modules running on generic hardware platforms such as an FPGA. The enhancements in FPGA technology and especially its inherent parallel processing capabilities have made these components the core-processing engine in SDR applications. Their rich architecture with customization features offers a huge design space solution which opens another challenge to the designers i.e., to create optimal or near-optimal designs for SDR. By moving the ADC close to the antenna, radio functionalities such as down-conversion and channel filtering which were previously implemented in the analog domain become digital forming a Digital Front-End. The large bandwidth and high dynamic range of the signals result in a high sample rate and large word length. The DFE is one of the most power- and time-critical functionalities of an SDR, as it has to meet the requirement of reconfigurability and/or programmability for an SDR in the presence of very high sample rates. The traditional digital signal processing design methodologies are not practically suited for implementing the SDR DFE. This is because of the fact that the traditional or legacy design solutions were based on compromises appropriate for their times. Furthermore, the straightforward SDR DFE designs that replicate their analog counterparts operating at high input sample rates are not only difficult to map to the current technology platforms, but also make their hardware architectures power hungry.

This dissertation targets resource minimal FPGA based architecture designs for a SDR DFE. These architectures are based on multirate signal processing methods/ techniques which specify new ways of performing DSP tasks that are normally not available using traditional DSP design methods. Multirate signal processing violates the traditional DSP methodology by *intentionally* aliasing the signal, processing at reduced sampling rate, and efficiently extracting the desired output signal from the aliased signals. A state-of-the-art DFE (with down-conversion, channel filtering, sample rate conversion, single and multiple channel extraction) including synchronization was presented which clearly illustrates that multirate signal processing based methods are well suited for the design of a SDR DFE. It not only provides low complexity and high performance solutions, but at the same time it allows several functionalities to be embedded within a single filter, these include: (i) extraction of multiple channels by using a single filter, (ii) embedded sampling rate changes in the polyphase filter banks that allow any rational sample rate change while simultaneously performing filtering operations, and (iii) combined arbitrary resampling, match filtering and symbol timing recovery, etc. The design and the FPGA mapping of a polyphase filter based approach for performing combined arbitrary resampling, match filtering, and symbol timing recovery for a BGAN receiver was presented. A structure with Maximum Likelihood (ML) control of polyphase matched filter was modified to embed arbitrary interpolation, which in terms of computational complexity is more efficient (12% reduction) than the cascaded structure of an arbitrary interpolator and matched filter. The main contribution of the dissertation was the Design Space Exploration (DSE) for polyphase filter bank architectures for the design of SDR DFE, together with the step-wise refinements to enhance their FPGA implementation in terms of reduced area resources, processing time, power consumptions, as well as achieving high operating clock rates. The Design Space Exploration (DSE) is performed for the building blocks of the polyphase filter bank i.e., polyphase filter and FFT.

In the following sections, we provide answer to the questions that lead to the formulated thesis in section 1.8.

A polyphase filter bank was analyzed in terms of its (i) general, (ii) symmetric, (iii) adder-shared symmetric structures, and serial polyphase with (iv) parallel and (v) serial MAC structures. The resource complexity and processing clock rate analyses showed a significant variation in resource usage and required operating clock rates, from a straightforward M parallel path implementation to a shared path with single MAC approach. The situation where the input sample rate is nearly equal or higher than the platform processing capabilities; a fully parallel solution is recommended where each subfilter is processed at a lower rate, and consequently requires larger area resources. On the other hand, the situation where the input sample rate is significantly lower than the platform processing capabilities, a resource sharing architecture is often favored. The resource sharing demands higher clock rates which should be achievable by the design and most importantly by the platform itself. Among all the structures which do not operate at a clock rate higher than the input clock rate, the serial polyphase structure with parallel MAC turned out to be the most resource efficient.

The goals defined in the introduction of this dissertation have been achieved as follows:

1. *to explore the design space by taking advantage of the reconfigurable target architecture*

Different design techniques for mapping register banks, especially the data register bank, were presented, which include: (i) a straightforward mapping based on CLB slice register or RAM registers, and (ii) more advanced solutions using an FPGA's dedicated memory resources, such as distributed and/or block RAMs. CLB based data register bank demands more area resources even for a small sized filter. The block RAM based register bank optimized the area resources but required an extra clock cycle to have data shift among the cascaded block RAMs, which was not the case for a distributed RAM based data register bank. This extra clock cycle resulted in higher operating clock rate demands to meet their output constraint. The pipelined configuration of data register bank can operate at a higher clock rate, but at the expense of increased latency. The inserted pipeline registers in the cascaded path were balanced by extra pipeline registers both in the control and the output data paths for the data register bank. The latency was doubled for the pipelined block RAM based register bank as compared to the pipelined distributed RAM based register bank. The serial polyphase with parallel MAC approach has an N/M inputs MAC unit. Two different MAC approaches i.e., adder tree and systolic array approaches were considered. The systolic array based MAC was modified with a set of pipeline registers to match its operation with the outputs from the data register bank. Both of these approaches used dedicated computational resources i.e., DSP48E slices for multipliers however, the adder tree used CLB based adders, whereas the systolic array approach used the adder built-in the DSP48E slice. The systolic array based MAC had a 4 clock cycles higher latency than the adder tree based MAC, but enhanced the system performance to achieve higher operating clock rates.

2. *to carry out mapping experiments in order to gain sufficient experiences to formulate a set of structured design guidelines*

The experimentation and the step-wise refinements toward achieving area, time, power, and latency optimized solutions were presented as a design flow diagram which actually explored a design space. The design all started from a very basic structure and its straightforward mapping, to more sophisticated structures with schedulers and their optimized mapping. It is always desirable to achieve optimal or near optimal solutions. Sometimes this may not be necessary due to loose constraints, bad design practices, and often short time to market that does not allow the designers to optimize their work. In such a case a design space exploration will definitely help the designer to foresee the performance of a solution with different building blocks. The explored design space shown in Fig. 2.12, is of course a subset of the complete design space, but provides a set of structural guidelines to help the designers choose a solution for their SDR DFE design and architectures within the given constraints of area, time, power, and latency.

The polyphase filter bank can embed rational sample rate conversion together with performing down-conversion and channel filtering. This is a very powerful feature of the polyphase filter bank as with no extra processing load, it can efficiently achieve the target output sample rate and eliminate a the separate block for sample rate conversion. The polyphase filter was analyzed in terms of five different embedded resampling modes:

(i) maximally decimated, (ii) under-decimated, (iii) over-decimated, and combined up- and down-sampling with (iv) single and (v) multiple stride lengths of the commutator that feeds the input data into the filter bank. These scenarios are applicable to any rational sampling rate change in a polyphase channelizer-based SDR front-end. A non-maximally decimated polyphase filter, where the number of data loads was not equal to the number of M subfilters, processed M subfilters in a time period that was either less than or greater than the M data loads' time period. Two different types of architectures i.e., Load Process Architecture and Runtime Architecture, based on load process and runtime scheduling schemes, for the serial polyphase with parallel MAC structure were explored. The LPA required a high clock rate to meet its output time constraint, which can be reduced by using interlaced LPA scheduling. The RA, on the other hand, operated at a maximum of double the input data rate (for the presented resampling scenarios), which enabled scheduling of subfilter processes along with the data loading. A detailed analysis of LPA and RA for the five embedded resampling cases in terms of area, time, and power were carried out and showed that RA was superior to LPA in reducing operating clock rates and dynamic power. RA was also superior in reducing area resources, except when address indices for subfilters were pre-stored in the LUT. RA was further optimized by using a systolic array of DSP48E (dedicated computational resources on Xilinx FPGAs) slices based MAC and a pipelined configuration of cascaded distributed RAM based register bank to achieve the high operating clock rates up to 500 MHz.

An analysis was performed for the latency and throughput of the explored architectures. The throughputs were constraint by the output sample rates which became the same for all architectures operating in same resampling scenario. The RAs turned out to have lower latency as compared to LPAs. This is due to the simultaneous loading and processing of the input samples, rather than first load and process afterward as in the case for LPAs. In the beginning of our experimental process, we stated that a FIFO based tapped delay line approach [88] for a decimator, using a serial polyphase structure with parallel MAC approach, is the most optimal solution. This solution, for example, has a latency of 12 cycles for a 16-tap filter which is partitioned into four subfilters, which results in a 4-to-1 down-sampling. The same analysis for our design based on runtime architecture results in the same latency. However, our architecture can also perform rational resampling for other cases in addition to the maximally decimated system which is only the case for the first approach.

The DFT was analyzed by considering Radix-2 FFT (together with its variant in terms of number of butterflies being processed simultaneously), Prime Factor, and Winograd Fourier Transform (WFT) algorithms. WFT has the lowest computational complexity because of elimination of twiddle factors and complex multipliers. The DSE for a 40-point transform being mapped as a 5x8 WFT was carried out to explore fully parallel, partially parallel, and sequential designs with different scheduling schemes. The corresponding trade-off between the area resources used and the required clock rates was presented.

Based on our work in [A] to [F], we conclude that the polyphase filter bank engines are the most desired choice for SDR DFE and that they can be efficiently implemented on a reconfigurable platform with minimized area, time, operating clock speed, latency, and power parameters. The explored solution space not only presents different architectures based on different scheduling schemes, but also gives a detailed insight into the selection and the performance criteria for their building blocks. The interaction of these building blocks with different configurations and the resulting output performances are highly recommended for the designers which will help them to plan the system resources for a desired constraint and performance implementation, thus saving design time and cost. These design customizations resulted in wide range of operating clock rate performance from 200 MHz to 500 MHz (for the investigated platform i.e., Xilinx Virtex-5). A set of detailed structural guidelines based on our explored solution space were presented in chapter 2. We believe that some, if not all of them, will definitely be helpful for any specific scenario. Furthermore, with the experimentation we performed, we conclude that RA with distributed RAM pipelined register bank, distributed RAM or pipelined block RAM based coefficient bank, and a systolic array of DSP48E slices based MAC is a preferred choice for an enhanced performance and reduced latency solution for SDR digital front-ends.

Although the Runtime Architecture has proven to be a low latency and high performance solution among the presented solutions as it supersedes the LPA in terms of reduced area (except where subfilter indices are pre-stored in LUTs), time, operating clock rate, dynamic power, and of course latency, but it still has some disadvantages:

1. It has a simple state machine but a non-trivial complicated scheduling that interlaces the processing of data loaded and non-data loaded subfilters. It is sometimes difficult to calculate on-the-fly the next indices for data-loaded or non data-loaded subfilter processing. The easy and the recommended solution is to store these indices in LUTs, which requires a bit of extra area resources.

2. Because of interlaced processing of data loaded and non-data loaded subfilters, the processed outputs are reordered. This does not affect the case for single channel extraction because all the subfilters' outputs are simply added by an M-input adder. However, it does have an affect for the case of multi-channel extraction where an M-point FFT is used.

Normally, FFT is performed on an incoming stream of data which is processed in blocks according to the transform size. Two approaches can be used to cancel this reordering process, (i) either wait for all the subfilters' outputs and then deliver them to the FFT processing in their natural order or (ii) modify the FFT process to compensate for this reordering. The first approach has the advantages of using the existing FFT processing cores from various vendors, but it will add delay (latency) for the complete processing. This latency is highly dependent on the number of subfilters or the transform size, which may not be desired for low latency solutions. The second approach of customized FFT processing according to the reordered indices offers the solution with no added latency, but requires extra effort to do so.

On the other hand, the interlaced load process scheduler can reduce the operating clock rates of load process scheduler to the operating clock rates of runtime scheduler, having a simple scheduler but a non-trivial complicated state machine and with a larger latency. However, in this case, the processed outputs are not reordered and can be directly fed to the FFT processing. The low latency achieved by the runtime scheduler may not be true when combined together with FFT processing, unless FFT processing is customized.

Finally, based on the explored solution space, and architectural designs to achieve Area, Time, operating clock Speed, Latency, and Power optimized solutions for polyphase filter banks, we conclude the thesis presented in section 1.8 by stating:

Yes, its possible to design polyphase filter bank engine(s) performing multiple functions for a SDR DFE while achieving cost effective mapping to reconfigurable platform in terms of Area, Time, operating clock Speed, and Power consumptions, utilizing the presented explored design trade-offs.

The architectural designs for a polyphase filter bank presented in this dissertation are not limited to only SDR DFE, but they can also be used anywhere polyphase filter bank are employed.

We have explored the solution space and believe that it will be helpful for designers to obtain resource optimized architectures for SDR digital front-ends; there is always room for further optimizations and exploring the solution space even more, which will be discussed next.

Future Work:

The future work can be of manyfold; either by further optimizations in the already explored solutions or by finding new ways and methods to achieve resource optimized architectures for SDR digital front-ends. Every solution has its own advantages and disadvantages, but sometimes these advantages may not necessarily be beneficial to the overall system design.

1. It is seen from our experiments and findings that designing a data register bank, which is a two dimensional memory structure, is a challenging task. The cascaded structure of memories (either distributed or blocks RAMs) initially showed limited performance because of their intra-connections, which was later optimized by utilizing pipeline structures. A new memory structure is suggested as a future work to explore, where all the memories have a common input so that they are no longer in cascade but now are independent of each other. Because of this independency, each memory (representing a subfilter's tap) has to hold the memory contents of all the preceding memories. So the memory's depth will no longer be the same but will become a multiple of the total number of subfilters. For instance, for a 20-tap filter partitioned into 5 subfilters each with 4 taps, the new memory structure (for a maximally decimated system) has 3 independent memories each having a depth of multiple of five, as shown in Fig. 4.1.
   Furthermore the memories for this structure need ”read-first-mode” for their write operation so that the previous data can be extracted for processing before storing the new data in the same address location. Each of these memory has it own address generator to write and read the memory according to the subfilters' operation. For a maximally decimated system, the addresses for individual memories are modulo of multiples of 5, which are listed in Table 4.1.

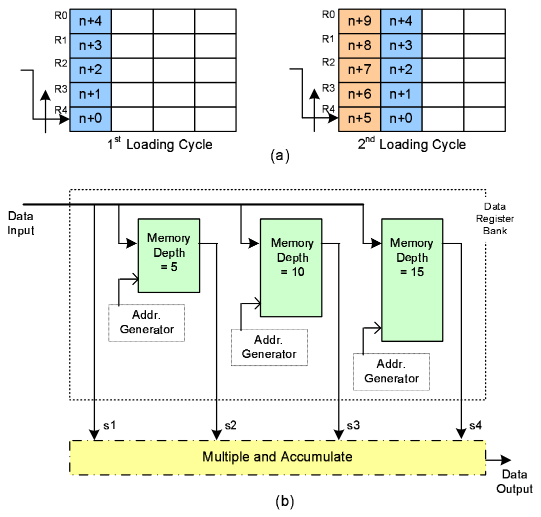


Figure 4.1: a) Two loading sequences for a 5-path maximally decimated system, b) proposed independent memory structure for data register bank.

Table 4.1: Address sequences for 5-path polyphase filter operating as a maximally decimated system, for the first 15 input data samples.

Data Input	**s1**	**s2**	**s3**	**s4**
n+0	**n+0**	@Addr: 0 *Rd:* **0**, *Wr:* **n+0**	@Addr: 0 *Rd:* **0**, *Wr:* **n+0**	@Addr: 0 *Rd:* **0**, *Wr:* **n+0**
⋮	⋮	⋮	⋮	⋮
n+4	**n+4**	@Addr: 4 *Rd:* **0**, *Wr:* **n+4**	@Addr: 4 *Rd:* **0**, *Wr:* **n+4**	@Addr: 4 *Rd:* **0**, *Wr:* **n+4**
n+5	**n+5**	@Addr: 0 *Rd:* **n+0**, *Wr:* **n+5**	@Addr: 5 *Rd:* **0**, *Wr:* **n+5**	@Addr: 5 *Rd:* **0**, *Wr:* **n+5**
⋮	⋮	⋮	⋮	⋮
n+9	**n+9**	@Addr: 4 *Rd:* **n+4**, *Wr:* **n+9**	@Addr: 9 *Rd:* **0**, *Wr:* **n+9**	@Addr: 9 *Rd:* **0**, *Wr:* **n+9**
n+10	**n+10**	@Addr: 0 *Rd:* **n+5**, *Wr:* **n+10**	@Addr: 0 *Rd:* **n+0**, *Wr:* **n+10**	@Addr: 10 *Rd:* **0**, *Wr:* **n+10**
⋮	⋮	⋮	⋮	⋮
n+14	**n+14**	@Addr: 4 *Rd:* **n+9**, *Wr:* **n+14**	@Addr: 4 *Rd:* **n+4**, *Wr:* **n+14**	@Addr: 14 *Rd:* **0**, *Wr:* **n+14**
⋮	⋮	⋮	⋮	⋮

The above structure can be further extended for non-maximally decimated systems. The address generators (for each memory) act as a steering wheel to process the subfilters for the desired embedded resampling factor. The advantages of eliminating the cascaded memory inter-connections, now has the disadvantage of having deep memories and separate address generators which will require more area resources. Furthermore, this new memory scheme may effect the performance with the increased number of individual memories (representing N/M taps) due to the fan-out conditions. This may be overcome by pipeline approaches but at an expense of extra area resources and latency.

2. The explored solution space was targeted for a specific Xilinx platform. This solution can be extended to other reconfigurable platforms (e.g., Altera) to have a comparative analysis of resources and performances. In particular, the performance and resource usage may be affected by the re-mapping of certain operations onto the dedicated internal resources of the FPGA (e.g. DSP48E blocks in Xilinx FPGAs vs. DSP blocks in Altera FPGAs).

3. The architectures presented for the polyphase filter bank (especially the one operating in non-maximally decimated modes) can be used to make new Intellectual Property cores which should be considered in a future approach for FPGA-based solutions.

5 List of Publications

Peer-Reviewed Journals

[A] Mehmood Awan, Yannick Le Moullec, Peter Koch, and fred harris, ”Polyphase Filter Banks for Embedded Sample Rate Changes in Digital Radio Front-Ends”, *Special Issue on advances in Digital Front-End and Software RF Processing: Part II, ZTE Communications,* Vol. 9, No. 4, pp. 3-9, Dec 2011. ISSN 1673-5188.

[B] Mehmood Awan, Yannick Le Moullec, Peter Koch, and fred harris, ”Hardware Architecture Analysis of Polyphase Filter Banks Performing Embedded Resampling for Software Defined Radio Front-Ends”, *100G and Beyond: Trends in Ultrahigh-Speed Communications (Part I), ZTE Communications*, Vol. 10, No. 1, pp. 54-62, Mar 2012. ISSN 1673-5188

Peer-Reviewed Conferences

[C] Mehmood Awan, fred harris, and Peter Koch, ”Time and Power Optimizations in FPGA-Based Architectures for Polyphase Channelizers”, *45th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, Nov. 6-9, 2011, pp. 914-918. ISBN: 978-1-4673-0323-1

[D] Mehmood Awan, Peter Koch, Chris Dick, and fred harris, ”FPGA Implementation Analysis of Polyphase Channelizer Performing Sample Rate Change Required for both Matched Filtering and Channel Frequency Spacing”, *44th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, Nov. 7-10, 2010, pp. 414-418. ISBN: 978-1-4244-9721-8

[E] Mehmood Awan, and Peter Koch, ”Combined Matched Filter and Arbitrary Interpolator for Symbol Timing Synchronization in SDR Receivers”, *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, Vienna Austria, April 14-16, 2010, pp. 153-156. ISBN (print): 978-1-4244-6610-8/10

[F] Mehmood Awan, and Peter Koch, ”Polyphase Channelizer as Bandpass Filters in Multi-Standard Software Defined Radios”, *2nd International Workshop on Cognitive Radio and Advanced Spectrum Management (CogART 2009)*, Aalborg, May 18-20, 2009, pp. 59-63. ISBN: 978-1-4244-4066-5

References

[1] J. Laskar, B. Matinpour, and S. Chakraborty, *Modern Receiver Front-Ends: Systems, Circuits, and Integration*. John Wiley & Sons, Inc., 2004.

[2] C. Drentea, *Modern Communications Receiver Design and Technology*. Artech House, Inc., 2010.

[3] *History of the Radio Receiver*. [Online]. Available: http://www.radio-electronics.com

[4] R. T. Sutton, “Eddystone radio - a short history of receiver developments from 1965-1995,” *International Conference on 100 Years of Radio*, Sept 1995, pp. 134-140.

[5] A. Parssinen, *Direct Conversion Receivers in Wide-Band Systems*. Springer, 2001.

[6] f. harris, C. Dick, M. Rice, “Digital receivers and transmitters using polyphase filter banks for wireless communications,” *IEEE Transactios on Microwave Theory and Techniques*, vol. 51, pp. 1395-1412, April 2003.

[7] J. Mitola, “The software radio architecture,” *IEEE Communications Magazine*, vol. 33, pp. 26-38, May 1995.

[8] P. Cruz, N. Carvalho, K. Remley,“Designing and testing software-defined radios,” *IEEE Microwave Magazine*, vol. 11, no.4, pp. 83-94, June 2010.

[9] R. Lackey, D. Upmal, “Speakeasy: The military software radios,” *IEEE Communications Magazine*, vol. 33, no.5, pp. 56-61, May 1995.

[10] T. Hentschel, *Sample Rate Conversion in Software Configurable Radios*. Artech House, Inc., 2002.

[11] R. Baines, “The DSP bottleneck,” *IEEE Communications Magazine*, vol. 33, no. 5, pp. 46-54, May 1995.

[12] H. Harada, R. Prasad, *Simulation and Software Radio for Mobile Communications*. Artech House, Inc., 2002.

[13] J. Mitola, G. Q. Maguire, “Cognitive radio: making software radios more personal,” *IEEE Personal Communications*, vol. 6, no. 4, pp. 13-18, 1999.

[14] “RF-Sampling and GSPS ADCs Breakthrough ADCs Revolutionize Radio Architectures”, Mar. 2013. [Online]. Available: http://http://www.ti.com/lit/sg/snwt001/snwt001.pdf

[15] “High-Speed Data Converter Portfolio”, Mar. 2013. [Online]. Available: http://www.maximintegrated.com/products/data_converters/high-speed/

[16] T. Hentschel, M. Henker, G. Fettweis, “The digital front-end of software radio terminals,” *IEEE Personal Communications*, vol. 6, pp. 40-46, Aug 1999.

[17] D. Lie et al., “Bridging dream and reality: Programmable baseband processors for software-defined radio,” *IEEE Communications Magazine*, vol. 47, no. 9, pp. 134-140, Sept 2009.

[18] T. Ulversøy, “Software defined radio: Challenges and opportunities,” *IEEE Communications Surveys & Tutorials*, vol. 12, no. 4, pp. 531-550, 2010.

[19] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, *FPGA-based Implementation of Signal Processing Systems*. John Wiley & Sons, Ltd., 2008. ISBN: 978-0-470-03009-7

[20] M. J. Bass and C. M. Christensen, ”The Future of the Microprocessor Business”, IEEE Spectrum, vol. 39, no. 4, pp. 34-39, April 2002.

[21] W. Tuttlebee, *Software Defined Radio Baseband Technology for 3G handsets and Basestations*. John Wiley & Sons Ltd., 2004.

[22] http://www.xilinx.com.

[23] http://www.altera.com.

[24] A. Irturk, “An FPGA design space exploration tool for matrix inversion architectures,” *Symposium on Application Specific Processors (SASP)*, pp. 42-47, June 2008.

[25] K. Skey, J. Bradley, K. Wagner, “A reuse approach for FPGA-based SDR waveforms,” in *Proceedings of IEEE Military Communications Conference (MILCOM)*, Oct 2006, pp. 1-7.

[26] C. L. Sotiropoulou, S. Nikolaidis, “Design space exploration for FPGA-based multiprocessing systems,” in *Proceedings of 17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Dec 2010, pp. 1164-1167.

[27] Zain-ul-Abdin, B. Svensson, “A study of design efficiency with a high-level language for FPGAs,” *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1-7, March 2007.

[28] P. Arato, Z. Adam Mann, A. Orban, “Algorithmic aspects of hardware/software partitioning,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 1, pp. 136-155, Jan 2005.

[29] P. P. Vaidyanathan, V. C. Lie, “Classical sampling theorems in the context of multirate and polyphase digital filter bank structures,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 9, pp. 1480-1495, Sept 1988.

[30] f. harris, *Multirate Signal Processing for Communication Systems*. Prentice Hall, 2006.

[31] M. Bellanger, G. Bonnerot, M. Coudreuse, “Digital filtering by polyphase network: Application to sample-rate alteration and filter banks,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 24, no. 2, pp. 109-114, April 1976.

[32] f. harris, ”On detecting white space spectra for spectral scavenging in cognitive radios”, *Wireless Personal Communications*, 2008, 45:325-342.

[33] f. harris and C. Dick, “On structure and implementation of algorithms for carrier and symbol synchronization in software defined radios,” *EUSIPCO-2000, ”Efficient Algorithms for Hardware Implementation of DSP Systems”*, Sept 2000.

[34] f. harris, D. Vuletic, W. Lowdermilk “How to pack a room of analog FM modulators into a Xilinx FPGA,” *DSP Magazine*, April 2007.

[35] L. Cordness, “Direct digital synthesis: A tool for periodic wave generation (part 1),” *IEEE Signal Processing Magazine*, vol. 21, pp. 50-54, July 2004.

[36] J. E. Volder, “The CORDIC trigonometric computing technique,” *IRE Transactions on Electronic Computers*, vol. EC-8, pp. 330-334, Sept 1959.

[37] J. Vankka, “Methods of mapping from phase to sine amplitude in direct digital synthesis,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 44, pp. 526-534, March 1997.

[38] B. H. Edwards, R. T. Jackson, J. M. Underwood, “How do calculators calculate?,” *in Proceedings of the 9th Annual International Conference on Technology in Collegiate Mathematics*, Nov 1996.

[39] F. Cardells-Tormo and J. Valls, “Area-optimized implementation of quadrature direct digital frequency synthesizer on LUT-based FPGAs,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 50, no. 3, pp. 135-138, March 2003.

[40] J. Valls, et al., “The use of CORDIC in software defined radios: A tutorial,” *IEEE Communications Magazine*, vol. 44, no.9, pp. 46-50, Sept 2006.

[41] M. Jridi, A. Alfalou, “Direct digital frequency synthesizer with cordic algorithm and Taylor series approximation for digital receivers,” *European Journal of Scientific Research*, vol. 30, pp. 542-553, Sept 2009.

[42] f. harris, “Ultra low phase noise DSP oscillator [DSP tips & tricks],” *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 121-124, July 2007.

[43] M. Lohning, T. Hentschel and G. Fettweis, “Digital down conversion in software radio terminals,” in *Proceedings of the 10th European Signal Processing Conference (EUSIPCO)*, Sept. 2000, pp. 1517-1520.

[44] R. Schreier, W. M. Snelgrove, “Decimation for bandpass sigma-delta analog-to-digital conversion,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 3, pp. 1801-1804, May 1990.

[45] R. Crochiere and L. Rabiner, *Multirate Digital Signal Processing*. Prentice Hall, 1983.

[46] f. harris, “Fixed length fir filters with continuously variable bandwidth,” in *Proceedings of the 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, Wireless VITAE*, May 2009, pp. 931-935.

[47] f. harris, “Selectable bandwidth filter formed from perfect reconstruction polyphase filter bank,” in *Proceedings of the 44th Asilomar Conference on Signals, Systems and Computers*, Nov. 2010, pp. 1292-1296.

[48] A. Feuer, G. C. Goodwin, *Sampling in Digital Signal Processing and Control*. Birkhauser Boston, 1996.

[49] A. V. Oppenheim, R. W. Schafer, *Discrete-Time Signal Processing*. 2nd ed., Prentice Hall, 1999.

[50] E. C. Ifeachor, B. W. Jervis, *Digital Signal Processing (A Practical Approach)*. 2nd ed., Pearson Education Ltd., 2002.

[51] f. harris, “Performance and design of Farrrow filter used for arbitrary resampling,” in *Proceedings of the 13th International Conference on Digital Signal Processing (DSP 97)*, July 1997, vol. 2, pp. 595-599.

[52] f. harris, “The Discrete Fourier Transform applied to time domain signal processing,” *IEEE Communications Magazine*, vol. 20, pp. 13-22, May 1982.

[53] L. Pucker, “Channelization techniques for software defined radio,” in *Proceedings of SDR Forum Technical Conference*, 2003.

[54] J. Lillington, “The pipelined frequency transform”, Feb. 2002. [Online]. Available: http://www.rfel.com/whipapdat.asp

[55] G. Savir, “Scalable and reconfigurable digital front-end for SDR wideband channelizer.” MSc. Thesis, TU Delft, 2006.

[56] M.-L. Boucheret, I. Mortensen and H. Favaro, “Fast convolution filter banks for satellite payloads with on-board processing,” *IEEE journal on selected areas in communications*, vol. 17, no. 2, pp. 238-248, Feb. 1999.

[57] A. P. Vinod et al., “A reconfigurable multi-standard channelizer using QMF trees for software radio receivers,” in *14th IEEE proceedings on Personal, Indoor and Mobile Radio Communication*, Sep. 2003, vol. 1, pp. 119-123.

[58] H. Scheuermann and H. Gockler, “A comprehensive survey of digital transmultiplexing methods,” *Proceedings of the IEEE*, vol. 69, no. 11, pp. 1419-1450, Nov. 1981.

[59] P. P. Vaidyanathanr, “Multirate digital filters, filter banks, polyphase networks and applications: A tutorial,” *Proceedings of the IEEE*, vol. 78, no. 1, pp. 56-93, Jan. 1990.

[60] K. Zangi and R. Koilpillai, “Software radio issues in cellular base stations,” *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 561-573, April 1999.

[61] T. Hentschel, “Channelization for software defined base-stations,” *Annales des Telecommunications*, vol. 57, pp. 386-420, May-June 2002.

[62] W. A. Abu-Al-Saud and G. L. Stuber, “Efficient wideband channelizer for software radio systems using modulated PR filter banks,” *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2807-2820, Oct. 2004.

[63] R. Mahesh and A. P. Vinod, “Reconfigurable Discrete Fourier Transform filter banks for multi-standard channelizers,” in *Proceedings of International Conference on Signal Processing and Communications (SPCOM)*, July 2010, pp. 1-5.

[64] R. Mahesh and A. P. Vinod, “Coefficient decimation approach for realizing reconfigurable finite impulse response filters,” in *Proceedings of IEEE International Symposium on Circuits and Systems*, May 2008, pp. 81-84.

[65] f. harris, E. Venosa, X. Chen, B. Rao, “Polyphase analysis filter bank down-converts unequal channel bandwidths with arbitrary center frequencies,” *Analog Integrated Circuits and Signal Processing*, vol. 71, no. 3, pp. 481-494, June 2012.

[66] C. Dick, f. harris, M. Rice, “Synchronization in software radios, carrier and timing recovery using FPGAs,” *IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 195-204, 2000.

[67] R. Prasad, R. Nee, S. Dixit, T. Ojanpera, *Globalization of Mobile and Wireless Communications: Today and in 2020, Signals and Communication Technology*. Springer Science + Business Media, 2011.

[68] E. Venosa, f. harris, F. Palmieri, *Software Radio: Sampling Rate Selection, Design and Synchronization*. Springer Science+Business Media, LLC, 2012. ISBN 978-1-4614-0112-4

[69] f. harris, C. Dick, “Performing simultaneous arbitrary spectral translation and sample rate change, in polyphase interpolating or decimating filters in transmitters and receivers,” in *Proceedings of Software Defined Radio Technical Conference and Product Exposition*, Nov. 2002.

[70] B. Egg, f. harris, C. Dick, “Ultra-wideband 1.6 GHz channelizer: Versatile FPGA implementation,” in *Proceedings of Software Defined Radio Technical Conference and Product Exposition*, 2005.

[71] A. Melis, G. Comoretto, “A 512 MHz polyphase filterbank with overlapping bands.” Arcetri Technical Report, 2011. [Online]. Available: http://www.arcetri.astro.it/pubblicazioni/Reports/11/testi/1_11.pdf.

[72] S. Berner, P. Leon, "FPGA-based filter bank implementation for parallel digital signal processing," 8^{th} *NASA VLSI Symposium*, July 1999.

[73] M. Valdes, M. Moure, J. Dieguez, S. Antelo, "Hardware solution of a polyphase filter bank for MP3 audio processing," *IEEE International Symposium on Industrial Electronics (ISIE)*, pp. 1225-1229, June-July 2008.

[74] G. Marinova, V. Guliashki, D. Ruyet, M. Bellanger, "Multicarrier modem core on FPGA," *IEEE Mediterranean Electrotechnical Conference (MELECON)*, May 2006, pp. 66-69.

[75] Y. Han, "A flexible and compact digital front-end design for wideband software radio receivers," in *Proceedings of* 7^{th} *International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Sept. 2011, pp. 1-4.

[76] C. N. Ang, R. H. Turner, T. Courtney, R. Woods, "Virtex FPGA implementation of a polyphase filter for sample rate conversion," in *Proceedings of* 34^{th} *Asilomar Conference on Signals, Systems and Computers*, Oct-Nov 2000, pp. 365-369.

[77] S. A. Fahmy, L. Doyle, "Reconfigurable polyphase filter bank architecture for spectrum sensing," in *Proceedings of the* 6^{th} *international conference on Reconfigurable Computing: architectures, Tools and Applications*, 2010, pp. 343-350.

[78] Y. Wang et al., "Energy-efficient hardware architecture and VLSI implementation of a polyphase channelizer with applications to subband adaptive filtering," *Journal of Signal Processing Systems*, vol. 58, no. 2, pp. 125-137, Feb 2010.

[79] http://www.4dsp.com/Polyphase_filterbank.php.

[80] http://www.sundance.com/prod_info.php?board=FC108.

[81] http://www.rfel.com/polyphase-dfts-ventrix-range-.aspx.

[82] "Xcell Journal - The Authoritative Journal for Programmable Logic Users," no. 48, 2004. [Online]. Available: http://www.xilinx.com/publications/archives/xcell/Xcell48.pdf.

[83] R. Rao, M. Tisserand, M. Severa, J. Villasenor, "FPGA polyphase filter bank study & implementation," [Online]. Available: http://slaac.east.isi.edu/presentations/retreat_9909/polyphase.pdf.

[84] P. Murphy, "Digital communications using FPGAs & Xilinx system generator," 2006. [Online]. Available: http://cmclab.rice.edu/workshops/materials/2006-10-05_DigitalComm/Rice_Comm_Workshop_Slides.pdf.

[85] P. Abusaidi, M. Klein, B. Philofsky, "Virtex-5 FPGA system power design considerations, Xilinx WP285 (v1.0)," 2008. [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp285.pdf.

[86] T. J. Moeller, D. R. Martinez, "Field programmable gate array based radar front-end digital signal processing," in *Proceedings of* 7^{th} *Annual IEEE Symposium on Field-Programmable Custom Computing Machines, IEEE Computer Society*, pp. 178-187, 1999.

[87] D. R. Martinez, T. J. Moeller, K. Teitelbaum, “Application of reconfigurable computing to a high performance front-end radar signal processor,” *Journal of VLSI Signal Processing Systems*, vol. 28, pp. 63-83, May-June 2001.

[88] “Virtex-5 FPGA XtremeDSP design considerations user guide, Xilinx UG193 (v3.3),” 2009.

[89] “Virtex-5 family overview, Xilinx DS100 (v5.0),” 2009.

[90] “Xilinx ISE 10.1 design suite software manuals and help,” 2008.

Contributions

Paper A

Polyphase Filter Banks for Embedded Sample Rate Changes in Digital Radio Front-Ends

Mehmood Awan, Yannick Le Moullec, Peter Koch, and fred harris

This paper was published in:
Special Issue on advances in Digital Front-End and Software RF Processing:
Part II, ZTE Communications

Abstract

This paper presents efficient processing engines for Software Defined Radio (SDR) front-ends. These engines use a polyphase channelizer for arbitrary sample rate changes, frequency selection, and bandwidth control. This paper presents an $\boldsymbol{M}$-path polyphase filter bank based on a modified $\boldsymbol{N}$-path polyphase filter. Such a system allows resampling by arbitrary ratios while performing baseband aliasing from center frequencies at Nyquist zones that are not multiples of the output sample rate. This resampling technique is based on sliding cyclic data load interacting with cyclic-shifted coefficients. A non-maximally-decimated polyphase filter bank (where the number of data loads is not equal to the number of $\boldsymbol{M}$ subfilters) processes $\boldsymbol{M}$ subfilters in a time period that is less or greater than the $\boldsymbol{M}$ data loads. A polyphase filter bank with five different resampling modes is used as a case study for embedded resampling in SDR front-ends. These modes are i) maximally decimated, ii) under-decimated, iii) over-decimated, and combined up- and down-sampling with iv) single stride length, and v) multiple stride lengths. These modes can be used to achieve any required rational sampling rate change in an SDR front-end based on a polyphase channelizer. They can also be used for translation to and from arbitrary center frequencies that are unrelated to the output sample rates.

1 Motivation

There are several generations of architectures for digital radio transceivers. A base station in a cellular mobile communication system is an example of a multichannel radio receiver that simultaneously down-converts and demodulates narrowband radio frequency (RF) channels [1] [2]. Traditional heterodyne architecture, considered the first generation of digital radio architecture, is shown in Fig. 1a for an N-channel receiver. Each sub-receiver consists of a dual-stage down-converter, and only the baseband processing is done in the digital domain [2]. In the first stage, the RF signal is down-converted to bandlimited intermediate frequency (IF). In the second stage, the IF filter output is again down-converted to baseband by a matched-quadrature mixer, and matched baseband filters that perform final bandwidth control. Next, the signal passes into the digital domain where the output of the analog-to-digital converter (ADC) is processed by digital signal processing (DSP) engines. These engines perform the required baseband processing, that is, synchronization, equalization, demodulation, detection, and decoding. The problem with this type of architecture is that amplitude and phase are imbalanced. This results in cross-talk between the narrowband channels due to aging (time, temperature) of the analog components of the quadrature down-converter. Each imbalance-related spectral image must be lower than the desired spectral term, and this is difficult to sustain over time and at variating temperature.

The need for extreme I/Q balance gave rise to the next generation of radios where second-stage (IF) down-conversion and, consequently, the channelization process are digitized, as shown in Fig. 1b. Digital conversion at IF provides greater control over the imbalance by manipulating the number of bits involved in the arithmetic operation. The precision of the coefficients used in the filtering process sets an upper limit for spectral artifacts at -5 dB/bit. This means that 12-bit arithmetic can achieve image levels below -60 dB [2]. The DSP based complex down-conversion, however, has two advantages: i) the spectral

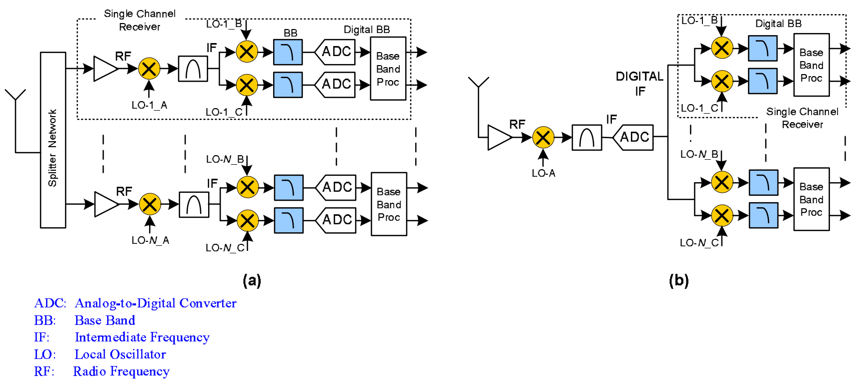


Figure 1: (a) First generation of digital radio architecture for an N-channel receiver, and (b) Second generation of digital radio architecture for an N-channel receiver.

images are controlled so that they are below the quantization noise floor of the ADC involved in the conversion process, and ii) the digital filters before and after the mixers are designed to have linear phase characteristics [2]. The second generation of radio, with digital front-end, is a realizable version of SDR. The range of applications for second-generation architecture shown in Fig. 1b is restricted to those with IF center frequencies of a couple of hundred megahertz. This is due to the limited dynamic range of high-speed ADCs. The dynamic range is often extended by using a hybrid scheme in which the initial complex down-conversion is performed by analog I/Q mixers, and channelization is performed digitally after the ADC. DSP techniques are applied to the digitized I/Q data to balance the gain and phase offsets in the analog ADC [3].

A digital front-end with standard design that includes frequency selection, bandwidth reduction, and sample rate reduction is one of the most power- and time-critical functionalities of an SDR terminal. This is due to the large bandwidth and high dynamic range of the signal to be processed. Consequently, the digital signals may have high sample rates and large word lengths. High sample rates not only increase power consumption but also make the use of time-shared hardware infeasible [4]. On the other hand, multirate signal processing specifies new ways of performing DSP tasks, and these ways are not normally available in traditional DSP designs. A multirate polyphase filter can perform the tasks of a multichannel receiver. In such a multichannel receiver, an input signal composed of many equal-bandwidth, equally spaced frequency-division-multiplexed (FDM) channels which are digitally down-converted to baseband (bandwidth is constrained by digital filters) and subjected to a sample rate reduction commensurate with the bandwidth reduction. This significantly reduces the amount of system resources required to perform multichannel processing and, consequently, costs [2] [3].

The remainder of this paper is organized as follows: In section 2, we briefly introduce a polyphase channelizer and describe how it is formulated from a conventional channelizer. In section 3, we categorize the embedded resampling cases presented in [5] into five different resampling cases: maximally decimated, under-decimated, over-decimated, and combined up- and down-sampled with single and multiple commutator stride lengths. In section 4, we perform MatLab simulations to analyze the performance of polyphase channelizers that are delivering the targeted output sample rates. Section 5 concludes the paper.

2 Introduction

A multirate polyphase filter can perform the tasks of a multichannel receiver. These tasks are equivalent to down-conversion, filtering, and resampling of multiple narrowband signals [5]. The step-by-step conversion of a standard single-channel demodulator into a multichannel polyphase channelizer is described in [2] and [3]. A brief introduction is given here. In the standard single-channel demodulation process, shown in Fig. 2a, the carrier-centered spectrum is translated to baseband (where a filter reduces the bandwidth), and a resampler reduces the sample rate in proportion to the bandwidth reduction. Standard single-channel demodulation is described by Eq. 1 and Eq. 2, where $x(n)$ is the carrier-centered input signal, θ_k is carrier angular frequency for k^{th} channel, $h(n)$ is the baseband filter, and $y(n,k)$ is the output baseband signal for k^{th} channel.

$$y(n,k) = [x(n)e^{-j\theta_k n}] * h(n) \tag{1}$$

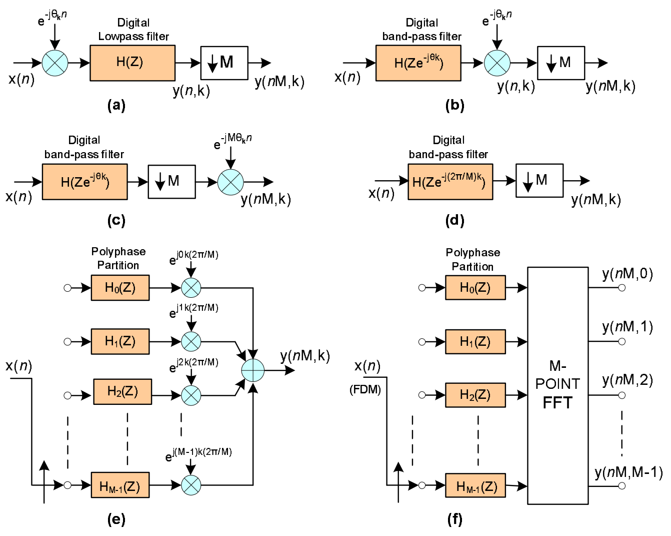
$$y(n,k) = \sum_{r=0}^{N-1} x(n-r)e^{-j\theta_k(n-r)}h(r) \tag{2}$$

According to the Equivalency Theorem [3], down-conversion followed by baseband filtering can be reordered so that filtering at the carrier occurs first, followed by down-conversion. This is the opposite of the traditional channelization process. Fig. 2b shows this reordered operation, which is also described by

$$y(n,k) = e^{-j\theta_k n} \sum_{r=0}^{N-1} x(n-r)h(r)e^{-jr\theta_k} \tag{3}$$

To reduce the work involved in down-converting and then discarding the samples during resampling, the heterodyne and down-sampler are reordered, and only retained samples are down-converted, as shown in Fig. 2c. In this case, the frequency of the heterodyne at the reduced sample rate is $M\theta_k$ rad/sample rather than the original frequency of θ_k. If the center frequency θ_k is a multiple of the output sample rate $2\pi/M$, that is, $k(2\pi/M)$, then the center frequency is aliased to 0 by M-to-1 resampling. Under this condition, the down-sampled heterodyne defaults to unity and can be discarded, as shown in Fig. 2d.

For the computed output for each input, $M-1$ of these computed output samples are discarded by the down-sampler. To reduce this workload, the resampling and the filtering operations are reordered so that one output is computed for every M input sample. This is achieved by applying the Noble identity [3], which states that a filter processing every M_{th} input sample followed by an output M-to-1 down-sampler is equivalent to an input M-to-1 down-sampler followed by a filter processing every input sample. The original up-converted filter is partitioned to M subfilters that operate at the reduced output rate



FDM: Frequency Division Multiplexing
FFT: Fast Fourier Transform

Figure 2: Transformation of a standard single-channel channelizer into a polyphase channelizer with M channels: (a) heterodyning, filtering, and down-sampling for a standard channelizer, (b) reordered channelizer using Equivalency Theorem, (c) reordering of the resampler, (d) down-converter with center frequency at a multiple of output sample rate aliases to baseband by M-to-1 down-sampling, (e) single-channel polyphase channelizer, and (f) polyphase channelizer for M channels.

rather than the original input rate. Eq. 4, 5 and 6 describe the mapping of the filter's Z-transform at the input rate to a sum of Z-transform at the output rate. The phase rotators in each subfilter are constant for that subfilter. Fig. 2e shows the block diagram for Eq. 6. The output resampler is pulled to the input side of each filter stage by applying the Noble identity. The input delay elements and the resampler at each stage are replaced by a rotary switch called a commutator.

$$G(Z) = \sum_{n=0}^{N-1} h(n) e^{j\frac{2\pi}{M}kn} Z^{-n} \tag{4}$$

$$= \sum_{r=0}^{M-1} \sum_{n=0}^{\frac{N}{M}-1} h(r+nM) e^{j\frac{2\pi}{M}k(r+nM)} Z^{-(r+nM)} \tag{5}$$

$$= \sum_{r=0}^{M-1} Z^{-r} e^{j\frac{2\pi}{M}kr} \sum_{n=0}^{\frac{N}{M}-1} h(r+nM) Z^{-nM} \tag{6}$$

In the final step of forming the polyphase filter bank, the sum formed by the phase rotators is one output port of a Discrete Fourier Transform (DFT). The DFT can be implemented as a Fast Fourier Transform (FFT) to extract time samples of each narrowband process located at multiples of the output sample rate (that has been aliased to baseband by the resampler) [5]. This is shown in Fig. 2f and given by

$$y(nM,k) = \sum_{r=0}^{M-1} y_r(nM) e^{j\frac{2\pi}{M}kr} \tag{7}$$

The relationship between the sampling frequency, channel spacing, and number of channels for the polyphase channelizer is

$$f_s = N \cdot \Delta f \tag{8}$$

where f_s is the input sampling frequency, N is number of channels (FFT size), which is the same as M here, and Δf is the inter-channel spacing [3].

The polyphase filter channelizer uses the input M-to-1 resampling to alias the spectral terms residing at multiples of the output sample rate to baseband. This means that for a standard polyphase channelizer processing M input samples at a time, the output sample rate is the same as the channel spacing. When operating in this mode, the system is a maximally decimated filter bank. We experimented with polyphase filter banks using embedded resampling and here present under-decimated, over-decimated, and combined up- and down-sampling (for single and multiple commutator stride lengths) modes.

3 Non-Maximally Decimated Filter Bank

We have briefly presented the polyphase filter bank channelizer in which the output sample rate is the same as the channel spacing. However, in practice, an output sample rate that is different than the channel spacing is often required. To uncouple the output sample rate from the channel spacing, a straightforward approach is to resample each channel with P/Q resamplers [5]. By changing the values of P and Q, the required sample rate can be obtained. An alternative is to embed the resampling process in i) the polyphase commutator, that is, in the interaction between input data registers and the polyphase coefficients, and ii) the interaction between the polyphase outputs and the FFT input. This alternative only requires a state machine to schedule the interactions, and there is no computational cost.

Two schemes [5] for these interactions are i) serpentine shifting the input data that interacts with a fixed set of coefficients and circular buffering the filtered data prior to FFT, and ii) sliding the cyclic data-load that interacts with cyclic-shifted coefficient memory. In the serpentine shift and circular buffering scheme, an input data set (not equal to M) is always fed to the same registers, and the polyphase subfilter coefficients are fixed. Let us consider a single-tapped delay line where all the data is moved further to the right before the next input data set is loaded. The data is moved by an address equal in length to the next input data set. By folding this one-dimensional tapped delay line into the two-dimensional memory of the polyphase filter, the data move is a serpentine shift between the columns. Because this non-equal M input data set is loaded, the data time-origin moves with respect to the FFT time-origin. To keep these two origins aligned, the computed output of the polyphase filter is circular-shifted by the residue address of the data time-origin mod M before the FFT is performed. In the sliding cyclic data-load and cyclic shift of the coefficient memory scheme, the data registers are fixed instead of being cycled, and the coefficient sets are rotated. The input data is fed as sliding cyclic load by the input commutator to a fixed set of registers, and the subfilter coefficients are cyclic-shifted by the same residue address of the data time-origin mod M before FFT is performed. Taking individual subfilters into account, the first scheme seems to require more read and write operations to synthesize the serpentine data shift, which is rather achieved by circular wrapping of block memory (an address control task). In the second scheme, only the loading subfilter gets a data shift.

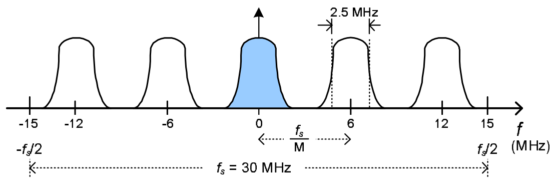


Figure 3: A 30 MHz FDM signal with 5 channels separated by 6 MHz center frequencies. Each channel has a 2.5 MHz symbol rate shaped by a square root Nyquist filter with 20% excess bandwidth.

To demonstrate the embedded resampling, we now describe the example shown in Fig. 3. A system has 5 channels separated by 6 MHz center frequencies. Each channel has a 2.5 MHz symbol rate shaped by a square root Nyquist filter (with 20% excess bandwidth) to form a 30 MHz FDM channel. To satisfy the Nyquist criteria at the output sample rate, the output sample rate must be greater than the occupied channel bandwidth. The occupied channel bandwidth of 3 MHz (symbol rate plus excess bandwidth) is selected to be smaller than the channel bandwidth of 6 MHz to allow down-sampling by large factors within the channelizer. The down-sample channelizer uses a 30-tap prototype low-pass filter with around 60 dB side-lobe attenuation that is partitioned into a 5-path polyphase filter with 6-tap subfilters. Both the data bank and filter coefficient bank are two-dimensional memories of 5 rows and 6 columns, and each row corresponds to a subfilter. According to Eq. 8, the output sample rate for the maximally decimated system becomes 6 MHz, which is the same as the channel spacing. Four other resampling factors are also introduced, and two of these have an embedded up-sampling factor of two. So, in total, these five resampling factors are 5, 3, 6, $5/2$, and $15/2$, delivering output sample rates of 6, 10, 5, 12 and 4 MHz respectively. These correspond to maximally decimated, under-decimated, over-decimated, and combined up- and down-sample cases. Each of these cases will be presented for the sliding cyclic data load and cyclic shift of the coefficient memory scheme.

Case 1: Maximally-Decimated Mode

In a maximally decimated system, data is loaded in stride lengths of 5 mod 5, and a computed output sample has a 5-to-1 down-sampling. Fig. 4a shows the data loading process for the two outputs. The subfilter's data register and coefficients are denoted R and C, respectively. In all the data loads, data loading starts from subfilter $R4$ up to $R0$, and the loaded subfilter's tapped delay line is pushed one tap to the right before a new data

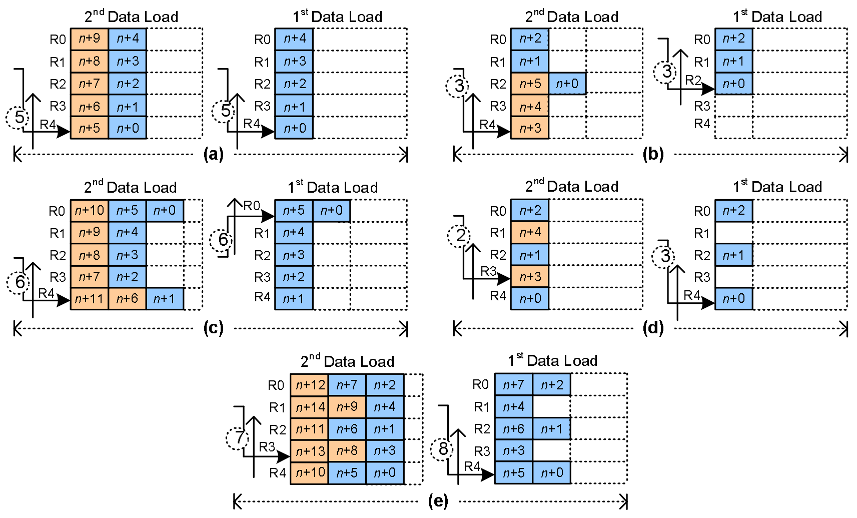


Figure 4: Data loading processes for a 5-path polyphase filter performing (a) 5-to-1 down-sampling (case 1), (b) 3-to-1 down-sampling (case 2), (c) 6-to-1 down-sampling (case 3), (d) 5/2 down-sampling (case 4), and (e) 15/2 down-sampling (case 5).

element is loaded. For every computed output, all the subfilters are fed with input data. Because there is no residue (non-loaded subfilter), there is no offset between the data time-origin and the FFT time-origin. The subfilters' coefficients are therefore fixed from $C0$ to $C4$. There is only one state machine where the 5-point data-loading of the register bank always performs an inner product with the fixed set of subfilter coefficients. Table 1 shows the register loading sequence and corresponding subfilters' coefficients.

State	Loading Sequence	Filter Coefficients
0	$R4\ \ R3\ \ R2\ \ R1\ \ R0$	$C0\ \ C1\ \ C2\ \ C3\ \ C4$

Table 1: Register loading sequence and corresponding subfilters coefficients for a 5-path maximally decimated system

Case 2: Under-Decimated Mode

In a non-maximally decimated (under-decimated) system, data is loaded in stride lengths of 3 mod 5, and a computed output sample has a 3-to-1 down-sampling within a 5-stage polyphase filter. The least common multiple (LCM) of 3 and 5 is 15, which means that the state engine cycles in 15 inputs, and because 3-point data is delivered at a time, there must be 5 distinct states in the state machine. Fig. 4b shows the data-loading processes for the first two states.

In the first state, data loading starts from subfilter $R2$ up to $R0$; and in the second state, data loading starts from $R4$ to $R2$ and so on for the five distinct states. Consequently, there is a residue of 2 for each data-loading operation. To align the data time-origin with the FFT time-origin, the subfilter coefficients are cyclic-shifted by the residue address of the data time-origin mod 5. The time-origin that is being cyclically shifted is also periodic in the LCM of 3 and 5. Thus, the cyclic shift of the polyphase subfilter coefficients has the same period as the data register load and is controlled by the same state machine. The data-loading sequence is always to the next 3 registers that have indexing of mod 5, which means that the next register to accept data when moving from state 0 to state 1 is ($R0$)-1, which is actually $R4$. Similarly, the filter coefficients assigned to perform the inner products with the registers are always offset 3 mod 5 relative to the previous filter set. Table 2 shows the state machine for register-loading and coefficients of each corresponding subfilter for performing 3-to-1 down-sampling in a 5-stage polyphase filter.

State	**Loading Sequence**	**Filter Coefficients**
0	$R2$ $R1$ $R0$	$C0$ $C1$ $C2$ $C3$ $C4$
1	$R4$ $R3$ $R2$	$C3$ $C4$ $C0$ $C1$ $C2$
2	$R1$ $R0$ $R4$	$C1$ $C2$ $C3$ $C4$ $C0$
3	$R3$ $R2$ $R1$	$C4$ $C0$ $C1$ $C2$ $C3$
4	$R0$ $R4$ $R3$	$C2$ $C3$ $C4$ $C0$ $C1$

Table 2: Register loading sequence and the corresponding subfilter coefficients for 3-to-1 sample rate change in a 5-path polyphase filter.

Case 3: Over-Decimated Mode
In a non-maximally decimated (over-decimated) system, data is loaded in stride lengths of 6 mod 5, and a computed output sample has a 6-to-1 down-sampling within a 5-stage polyphase filter. The LCM of 6 and 5 is 30, which means that the state engine cycles in 30 inputs, and because 6-point data is delivered at a time, there must be 5 distinct states in the state machine. Fig. 4c shows the data-loading processes for the first two states.

In the first data load, loading starts from subfilters $R0$, $R4$ up to $R0$, and the second load starts from $R4$ up to $R0$ and $R4$ again and so on for the 5 distinct states. Consequently, there is a residue of 4 for each data-loading operation. To align the data time-origin with the FFT time-origin, the subfilter coefficients are cyclic-shifted by the residue address of the data time-origin mod 5. Table 3 shows the state machine for register-loading and the corresponding coefficients for performing 6-to-1 down-sampling in a 5-stage polyphase filter.

State	**Loading Sequence**	**Filter Coefficients**
0	$R0$ $R4$ $R3$ $R2$ $R1$ $R0$	$C0$ $C1$ $C2$ $C3$ $C4$
1	$R4$ $R3$ $R2$ $R1$ $R0$ $R4$	$C1$ $C2$ $C3$ $C4$ $C0$
2	$R3$ $R2$ $R1$ $R0$ $R4$ $R3$	$C2$ $C3$ $C4$ $C0$ $C1$
3	$R2$ $R1$ $R0$ $R4$ $R3$ $R2$	$C3$ $C4$ $C0$ $C1$ $C2$
4	$R1$ $R0$ $R4$ $R3$ $R2$ $R1$	$C4$ $C0$ $C1$ $C2$ $C3$

Table 3: Register loading sequence and the corresponding subfilter coefficients for 6-to-1 sample rate change in a 5-path polyphase filter.

Case 4: Combined Up- and Down-Sampling Mode (Single Stride)
In the previous three cases, down-sampling was performed by different factors. The polyphase filter is also capable of embedding the up-sampling factor with the down-

sampling so that the sample rate change is rational. In this case we up-sample by a factor of 2 and then down-sample by a factor of 5 to obtain a $5/2$ sample rate change. The up-sampling (performed by zero-packing the input data) is actually achieved by data-load addressing, in which one address is skipped so that 1-to-2 up-sampling can be realized. Down-sampling is performed by cyclic-loading the data (zero-packed) through the filter in stride of length 5. The two data-loading cycles for a $5/2$ sample rate change in a 5-path polyphase filter are shown in Fig. 4d.

In the first data load, 3 data samples are delivered to the 5 register addresses. In the second load, 2 data samples are delivered to the 5 register addresses. The data-loading process is periodic in 2 load cycles, and 2 states are needed to control the process. The data-loading process for the 2 states and the corresponding coefficient sets are listed in Table 4. In the 2 states, 5 inputs are delivered, and 2 outputs from the polyphase engine are taken to realize the desired $5/2$ embedded resampling. The loading scheme has a constant offset of -2 mod 5 within a sequence and also in the transition between sequences. The -2 offset is a result of the 1-to-2 up-sampling, represented by the zero packing.

State	No. of Inputs	Loading Sequence	Filter Coefficients
0	3	$R4\ R2\ R0$	$C0\ C6\ C2\ C8\ C4$
1	2	$R3\ R1$	$C5\ C1\ C7\ C3\ C9$

Table 4: Register loading sequence and the corresponding subfilter coefficients for $5/2$ sample rate change in a 5-path polyphase filter.

There are normally 5 subfilters in the polyphase partition of a 5-stage polyphase filter. Because of the 1-to-2 up-sampling implemented by the zero-packing, only half the coefficients in each stage actually contribute to the subfilter output [5]. Thus, each stage is further partitioned into 2 subsets of coefficients, which results in 10 subfilter coefficient sets. These sets are denoted $C0$, $C1...C9$, where the integer is the starting index from the original non-partitioned prototype filter. The successive filter index increments by 6 mod 10; and between the states, the filter index increments by 5 mod 10. The integer 6 is the offset between two data samples in the zero-packed load in two adjacent rows. Because of up-sampling by a factor of 2, the prototype filter has to be designed to operate at $2 \times f_s$, that is, 60 MHz. Consequently, the filter becomes twice as long as the standard design. However, because only half of it is used per processing cycle, there is no processing penalty [5].

Case 5: Combined Up- and Down-Sampling Mode (Multiple Strides)
The case is similar to case 4 but down-sampled by a factor of 15 to have a $15/2$ sample rate change. Up-sampling is performed by data-load addressing, which skips the next address, and down-sampling is performed by cyclic-loading the data through the filter in stride lengths of 15. The two states of the loading cycle for $15/2$ sample rate change in a 5-path polyphase filter are shown in Fig. 4e.

In the first data load, 8 data samples are delivered to the 5 register addresses. In the second load, 7 data samples are delivered to the 5 register addresses. The data-loading process is periodic in 2 load cycles, and 2 states are needed to control the process. Table 5 lists the data-loading process for the 2 states and the corresponding coefficient sets. In the process, 15-to-2 down-sampling is performed in a 5-path polyphase filter. The filter down-converts the spectral regions from multiples of $f_s/5$ (or $30/5$ = 6 MHz) and maintains a sample rate of $f_s(2/15)$ (or $60/15$ = 4 MHz).

State	No. of Inputs	Loading Sequence	Filter Coefficients
0	8	$R4\ R2\ R0\ R3\ R1\ R4\ R2\ R0$	$C0\ C6\ C2\ C8\ C4$
1	7	$R3\ R1\ R4\ R2\ R0\ R3\ R1$	$C5\ C1\ C7\ C3\ C9$

Table 5: Register loading sequence and the corresponding subfilter coefficients for $15/2$ sample rate change in a 5-path polyphase filter.

4 Simulations

The MatLab simulations show the embedded sample rate changes in a 5-path polyphase filter and DFT operating as a 5-channel channelizer. The FDM input signal has 5 channels which are each 16-QAM modulated, and separated by 6 MHz center frequencies. The sample rate is 30 MHz, and each channel has a 2.5 MHz symbol rate shaped by a square root Nyquist filter with 20% excess bandwidth. Three of the five channels, which are occupied by 3 MHz bandwidth signals, are centered at 0, 6, and 12 MHz. The remaining two channels, centered at -12 and -6 MHz, are intentionally kept empty. The input signal spectrum comprising 5 channels at 30 MHz is shown in Fig. 5a.

In a system operating in maximally decimated mode, the input data is channelized and down-sampled 5-to-1 for an output rate of 6 MHz. Each of the 5 polyphase filter stages are 6 taps long and are anchored to the 5 input registers being fed by the periodic input commutator. Fig. 5b shows the spectra of the 5 output channels with an output rate of 6 MHz. In a system operating in under-decimated mode, the same input data is channelized and down-sampled 3-to-1 for an output rate of 10 MHz. Fig. 5c shows the spectra of the 5 output channels with 10 MHz output rate. Similarly, in a system operating in over-decimated mode, the same input data is channelized and down-sampled 6-to-1 for an output rate of 5 MHz. Figure 5d shows the spectra of the 5 output channels with 5 MHz output rate.

In a system operating in combined up- and down-sampling mode, the input spectrum is channelized, up-sampled by a factor of 2, and down-sampled by 5-to-1 and 15-to-1 for output rates of 12 MHz and 4 MHz, respectively. Because of up-sampling by a factor of 2, there are 10 polyphase filter coefficient stages each with 6 taps. The filters' coefficients are periodically rotated through the 5 input registers (which have a periodic sliding input commutator) according to the state machine described in Table 4 and Table 5. The spectral locations of the channels are reordered as a result of processing the up-sampled data in the polyphase filter [3] [5]. The 5-point FFT processes the polyphase data output frequencies in the order $[0, 2, 4, 1, 3]$, which is seen to be indexing stride of 2 mod 5. These are reordered back to their natural order. Figs. 5e and 5f show the spectra of the 5 output channels at 12 MHz and 4 MHz, which correspond to $5/2$ and $15/2$ sample rate changes, respectively.

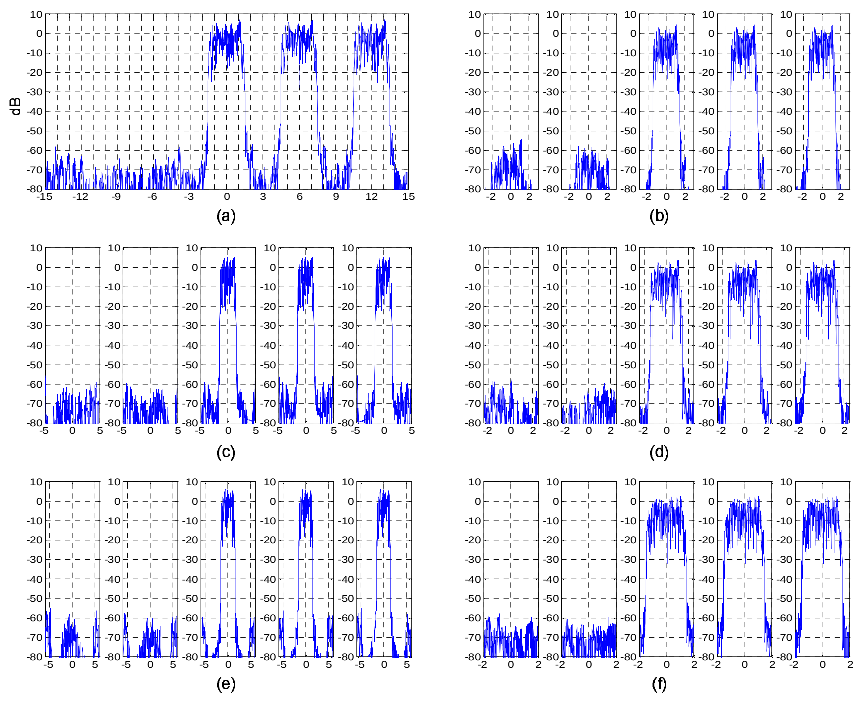


Figure 5: (a) Input signal spectrum with 5 channels of 3 MHz bandwidth and 6 MHz channel spacing at 30 MHz sample rate. The spectra of the 5 output channels are (b) 6 MHz for 5-to-1 down-sampling, (c)10 MHz for 3-to-1 down-sampling, (d) 5 MHz for 6-to-1 down-sampling, (e) 12 MHz for $5/2$ resampling, and (f) 4 MHz for $15/2$ resampling.

The simulations show that embedded sample rate changes can be successfully implemented in a polyphase channelizer. All the output channels have 60 dB of spectral side-lobe attenuation, selected by the prototype low-pass filter. The processing engines used in all the 5 cases are identical except that each has different state machines, register loading schemes, and subfilter coefficient sets.

5 Conclusion

In this paper, we have shown the versatility of a polyphase engine that performs embedded resampling that is uncoupled from frequency selection and bandwidth control. Five embedded resampling modes in polyphase filter banks are presented namely; maximally decimated, under-decimated, over-decimated, and combined up- and down-sampling which correspond to single, short, long, and multiple commutator stride lengths. For various applications, these modes can be used for any required rational sampling-rate change in an SDR front-end using a polyphase channelizer. The suggested modes are highly useful during the process of designing flexible and resource-optimal architectures for advanced software radios. In a subsequent paper ”Hardware Architecture Analysis of Polyphase Filter

Banks Performing Embedded Resampling for Software Defined Radio Front-Ends" [6], we analyze FPGA based hardware architecture of these resampling engines in terms of area, time, and power tradeoffs.

References

[1] A. M. Badda, M. Donati, "The software defined radio technique applied to the RF front-end for cellular mobile systems," *in Software Radio Technologies and Services, E. Del Re, Ed., Berlin, Germany: Springer-Verlag*, 2001.

[2] fred harris, Chris Dick, Micheal Rice, "Digital Receivers and Transmitters using Polyphase Filter Banks for Wireless Communications," *IEEE Transactios on Microwave Theory and Techniques*, vol. 51, pp. 1395–1412, April 2003.

[3] fred j. Harris, *Multirate Signal Processing for Communication Systems*, Prentice Hall, 2006.

[4] T. Hentschel, M. Henker, G. Fettweis, "The Digital Front-End of Software Radio Terminals," *IEEE Personal Communications*, vol. 6, pp. 40–46, Aug 1999.

[5] fred harris, Chris Dick, "Performing Simultaneous Arbitrary Spectral Translation and Sample Rate Change, in Polyphase Interpolating or Decimating Filters in Transmitters and Receivers," *in Proceedings of Software Defined Radio Technical Conference and Product Exposition*, Nov. 2002.

[6] Mehmood Awan, Yannick Le Moullec, Peter Koch, fred harris, "Hardware Architecture Analysis of Polyphase Filter Banks Performing Embedded Resampling for Software Defined Radio Front-Ends," *100G and Beyond: Trends in Ultrahigh-Speed Communications (Part I), ZTE Communications*, vol. 10, pp. 54–62, March 2012.

Paper B

Hardware Architecture Analysis of Polyphase Filter Banks Performing Embedded Resampling for Software Defined Radio Front-Ends

Mehmood Awan, Yannick Le Moullec, Peter Koch, and fred harris

This paper was published in:
100G and Beyond: Trends in Ultrahigh-Speed Communications (Part I), ZTE Communications

Abstract

In this paper, we describe resource-efficient hardware architectures for software defined radio (SDR) front-ends. These architectures are made efficient by using a polyphase channelizer that performs arbitrary sample rate changes, frequency selection, and bandwidth control. We discuss area, time, and power optimization for field programmable gate array (FPGA) based architectures in an M-path polyphase filter bank with modified N-path polyphase filter. Such systems allow resampling by arbitrary ratios while simultaneously performing baseband aliasing from center frequencies at Nyquist zones that are not multiples of the output sample rate. A non-maximally decimated polyphase filter bank, where the number of data loads is not equal to the number of M subfilters, processes M subfilters in a time period that is either less than or greater than the M data-load's time period. We present a load-process architecture (LPA) and a runtime architecture (RA) (based on serial polyphase structure) which have different scheduling. In LPA, N subfilters are loaded, and then M subfilters are processed at a clock rate that is a multiple of the input data rate. This is necessary to meet the output time constraint of the down-sampled data. In RA, M subfilters processes are efficiently scheduled within N data-load time while simultaneously loading N subfilters. This requires reduced clock rates as compared to LPA, and potentially less power is consumed. A polyphase filter bank that uses different resampling factors for maximally decimated, under-decimated, over-decimated, and combined up- and down-sampled scenarios is used as a case study, and an analysis of area, time, and power for their FPGA architectures is given. For resource-optimized SDR front-ends, RA is superior for reducing operating clock rates and dynamic power consumption. RA is also superior for reducing area resources, except when indices are pre-stored in LUTs.

1 Introduction

Polyphase filter banks are versatile engines that perform embedded resampling uncoupled from frequency selection and bandwidth control [1], [2]. In a previous paper "Polyphase Filter Banks for Embedded Sample Rate Changes in Digital Radio Front-Ends" [3], we described the benefits of such a polyphase engine. Five embedded resampling approaches were discussed: 1) maximally decimated, 2) under-decimated, 3) over-decimated, and combined up- and down-sampling with 4) single and 5) multiple stride lengths of the commutator (which feeds input data into the filter bank). These are efficient approaches to rational resampling in polyphase filter banks because there is no computational cost, and only a state machine is required to schedule the interactions. Polyphase engines are promising candidates for realizing software-defined radio (SDR) front-ends where embedded sample rate changes are needed. This paper describes the associated hardware architecture designed to reduce area, time, and power consumption of such a polyphase engines implemented on FPGA.

The hardware platform is the most prominent and challenging component of an SDR because it must provide massive computational power and flexibility at the same time [4], [5]. An important semiconductor technology is the field programmable gate array (FPGA). It consists of a vast array of configurable logic blocks, multipliers, and memories, and it allows a custom data path to be tailored to the application at hand. Parallel processing capabilities in an FPGA has made them the core processing engine in SDR applications. The rich architecture of FPGA with specialized functionalities allows a large design space. In such a large design space, resource-optimized architectures are highly desirable for implementing SDR functions. In this paper, we describe an FPGA-based architecture for polyphase filter banks that is optimized in terms of area, time, and power. A filter bank performs rational sample rate changes together with frequency control and bandwidth reduction.

In section 2, we analyze five structures for the polyphase filter banks in terms of required resources (referred to as resource complexity) and operating clock rates. From these structures, we select the serial polyphase structure with parallel multiply and accumulate (MAC) and explore its implementation options. In section 3, the selected structure is mapped onto the FPGA in two different implementation architectures that are built on different scheduling, denoted load-process architecture (LPA), and runtime architecture (RA). In section 4, we compare resource usages in the LPA and RA variants of the five embedded resampling cases in terms of slice registers, slice lookup tables (LUTs), and dedicated resources as well as the operating clock rates. In section 5, we analyze the dynamic power consumed in the LPA and the RA variants of the five resampling cases. Section 6 concludes the paper.

2 Architecture Design

In this section, we analyze the architectures of the FPGA implementation of a polyphase filter performing embedded sample rate changes. We do not deal with the FFT part and simply replace it with an M-input adder, which represents the output of a channel centered at origin (channel 0). The main building blocks of a polyphase filter are an M-path polyphase lowpass filter and an M-input adder. The structural analysis of the polyphase filter includes general, symmetric, and serial polyphase structures with serial and parallel MACs.

The general M-path polyphase filter structure has M subfilters of length N/M, where N is the length of a non-partitioned lowpass filter. Each of these subfilters operates at $1/M$ times the input rate (Fig. 1a). In the symmetric structure, the symmetry property of the coefficients in the subfilters is exploited, and this reduces the number of subfilters and commutator length to $M/2$. The commutator moves in both the forward and reverse feed directions (Fig. 1b), and the multipliers are shared by the two subfilters [6]. The adders can also be shared by using multiplexers and de-multiplexers to form an adder-shared symmetric structure (Fig. 1c). In an M-path polyphase filter, only one subfilter is processed at a time, and the remaining M-1 subfilters are idle. The serial polyphase structure merges the subfilters' data registers to form a data-register bank and merges the subfilters' coefficients to form a coefficient bank. These data register and coefficient banks are addressed by a control sequence so that the data registers and coefficients of the desired subfilter can be selected to perform MAC operations. The MAC operations can be performed in parallel or in serial such that a serial polyphase structure with parallel MAC (Fig. 1d) or serial MAC (Fig. 1e) is formed [7].

Table 1 shows the resource complexity in an M-path polyphase filter (each path with

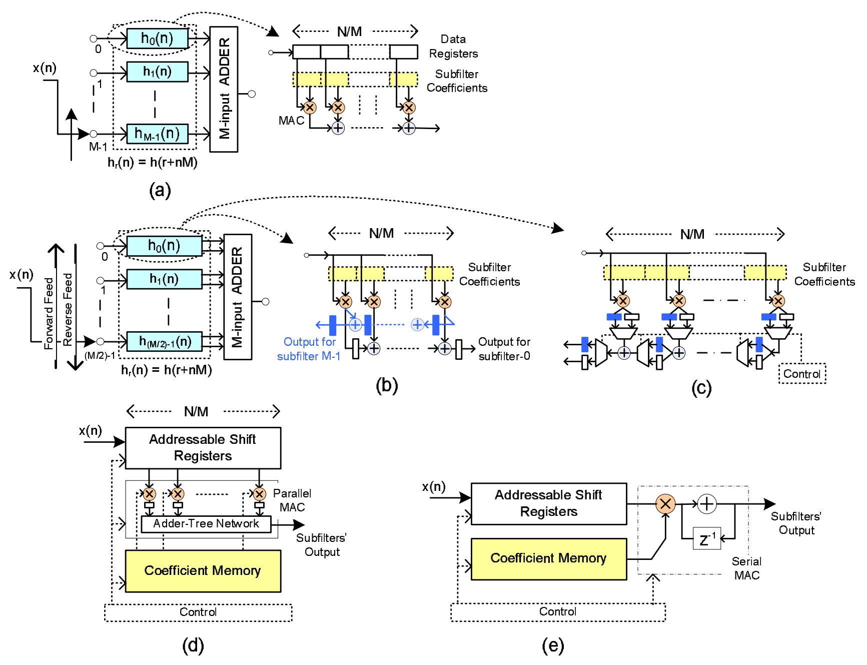


Figure 1: Polyphase filter structures: (a) general, (b) symmetric, (c) adder-shared symmetric, (d) serial polyphase with parallel MAC, and (e) serial polyphase with serial MAC.

N/M taps) for a general polyphase structure (P1), symmetric structure (P2), adder-shared symmetric structure (P3), and serial polyphase structure with serial MAC (P4) and parallel MAC (P5). It also shows the required operating clock rates for these structures (with fs as the input rate). There is a trade-off between complexities and processing clock rates. A serial polyphase filter with serial MAC has the least complexity but demands a high clock rate. A serial polyphase filter with parallel MAC has a slightly higher complexity and operates at a clock rate that corresponds to the input clock rate. Among all the solutions which do not operate at a clock rate higher than the input clock rate, the P5 uses the least resources.

	Mults	**Adds**	**Regs**	**Mux**	**Demux**	**Process Clock Rate**
P1	N	$\{(N/M)-1\}\,M$	N	0	0	f_s/M
P2	$N/2$	$\{(N/M)-1\}\,M$	N	0	0	$2(f_s/M)$
P3	$N/2$	$\{(N/M)-1\}\,M/2$	$2N$	$N-(M/2)$	$M/2$	$2(f_s/M)$
P4	1	1	$N+1$	$(N/M)+1$	0	$N(f_s/M)$
P5	N/M	$(N/M)-1$	$N+2(N/M)$	N/M	0	f_s

Table 1: Resource complexities and required process clock rates for an M-path polyphase filter with P1-P5 polyphase structures (with f_s as the input rate).

A non-maximally decimated polyphase filter bank processes M subfilters in a time period which is either less or greater than the M data-load's time period. The embedded resampling in the polyphase filter banks requires architectural changes for the structures (Fig. 1) to meet the output time constraint. These changes mainly apply to the structures which are sharing resources (P2, P3, P4 and P5). P1 is a fully parallel solution and is not operationally affected in non-maximally decimated modes because each subfilter is operating independently and M subfilters can be processed even for a single data input time period. However, it does require some changes in its state machine. Symmetric structures are limited to cases with an even number of polyphase partitions to make use of filter symmetry which is not the case for serial polyphase structures. Here, we describe the mapping of P5 onto the target platform - a Xilinx Virtex-5 FPGA (xc5vsx50t-3ff1136) [8]. A fixed-point analysis has been used to determine the word lengths that keep quantization errors below 60 dB (commonly required).

There are a number of design options that correspond to the level of FPGA resource exploitation. The targeted polyphase filter used for the embedded resampling cases [3] has five paths each with six taps. A straightforward mapping of the corresponding P5 structure onto an FPGA, where all the filter coefficients and the subfilters' tapped delay lines are implemented as combination logic blocks (CLBs), results in relatively high usage of slice registers and LUTs. The implementation scales linearly with the filter size. Today's FPGAs have rich architectures with special memory resources such as distributed RAM and block RAM, high-performance computational resources such as DSP48E slices in addition to the basic CLBs. The DSP48E slice improves flexibility, utilization, and efficiency of applications. It also reduces overall power consumption, increases maximum frequency, and reduces setup and clock-to-out time [9]. The efficient use of these dedicated resources creates a high-performance system with high operating clock rates and reduced CLB requirements.

The filter coefficient bank can be easily replaced by FPGA block RAMs to eliminate the number of CLB resources; however, the polyphase-partitioned data bank (register bank) is a bit critical because it has to shift the new data element to the respective subfilter's tap-delay line and having access to all the taps of that subfilter at the same time. In a maximally decimated system, where the down-sampling factor is equal to the polyphase partition, a first-in first-out (FIFO) can be used as delay lines to derive the optimal architecture, as described in [9]. For non-maximally decimated systems, a two-dimensional memory solution is required in which only the targeted subfilter can be loaded with the input data. Fig. 2 shows the resource usage when mapping a 5×6 register bank (with 32-bit complex data) according to different design options available on the Virtex-5 FPGA. The 5×6 register bank based on slice registers and LUTs uses 960 slice registers and 389 LUTs. Each data register in the subfilters is replaced with a RAM-based shift register (SRL16 / SRL32 mode of the slice LUTs), and the number of slice registers and LUTs usage becomes 0 and 389, respectively. This eliminates the need for slice registers, but the LUT usage remains unchanged. In Virtex-5, each CLB has 64-bit distributed RAM [8] that is bit-addressable. For a 32-bit data register, 32 CLBs are collectively used as a single 32-bit register. The remaining 63 bits in each CLB are unused. Distributed memory is used so that each 64-bit memory contributes one bit to a 32-bit data register for 64 subfilters (only five are used). In this way, the memory that was previously used for only one data register of a subfilter is now used for one data register for all five subfilters.

The resource usage for a 5×6 register bank based on distributed RAM becomes 192 slice registers and 192 LUTs. This eliminates the need for the decoder and multiplexers to select the desired subfilter's data elements in the case of a shift-register based register bank. The same concept can be applied to block RAM, which completely eliminates the CLB usage. Three block RAMs are used for a 5×6 register bank, which corresponds to 2% utilization of block RAM resources for a Virtex-5.

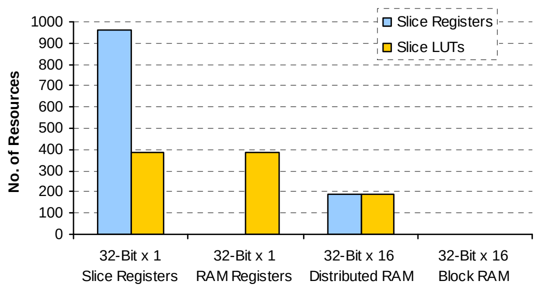


Figure 2: Resource usage for a 5×6 register bank by exploiting FPGA resources.

To use block RAM based register banks, an extra clock cycle is needed for each data load and shift [10] because six block RAMs (32-bit$\times 5$) are cascaded to form a 5×6 register bank, and the data shift in the subfilters requires data to be available from the preceding memory. Therefore, one clock cycle is needed to read the (previous) data elements in the cascaded memories, and the next clock cycle loads the new data element to the subfilter's data memory along with data shifting. It is because block RAMs have synchronous read and write operations. On the other hand, a distributed RAM based register bank has no extra clock cycle penalty for each data load and shift because it has asynchronous read and synchronous write operations.

3 LPA and RA Scheduling

According to the polyphase channelizer configurations for the presented embedded resampling factors [3], five polyphase subfilters need to be processed within the time period of five, three, and six samples and within the time period of zero-packed five and 15 samples. For the resampling cases, where the number of data-loads is less than M (the number of subfilters), it is not possible to process at the input sample rate of fs in a serial polyphase structure with parallel MAC. Fig. 3 shows the time domain view of the under-decimated case where 3-to-1 down-sampling is performed in a 5-path polyphase filter. The five subfilters are to be processed within the time period of three data samples.

In [10], an LPA with block RAM based register bank for a polyphase filter bank operating in under-decimated mode is described. This LPA runs at a higher clock rate and uses a FIFO to interface with the input data at lower rate. Fig. 4 shows this architecture with an output accumulator instead of an FFT to represent the channel at baseband. In the load phase, the data elements are taken from the input FIFO and loaded into the subfilters as directed by the data pointer. In the processing phase, coefficient and data pointers with embedded shifts select the coefficient and data memory elements in order to perform

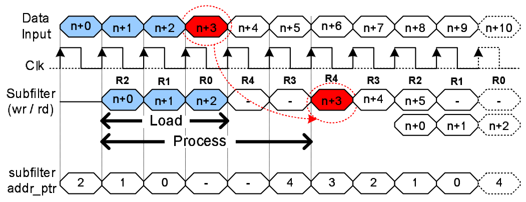


Figure 3: Under-decimated case where 3-to-1 down sampling is performed in a 5-path polyphase filter. The five subfilters are to be processed within the time period of three data samples.

MAC operations. The multipliers are based on DSP48E slices whereas the additions are performed using a CLB-based adder tree network.

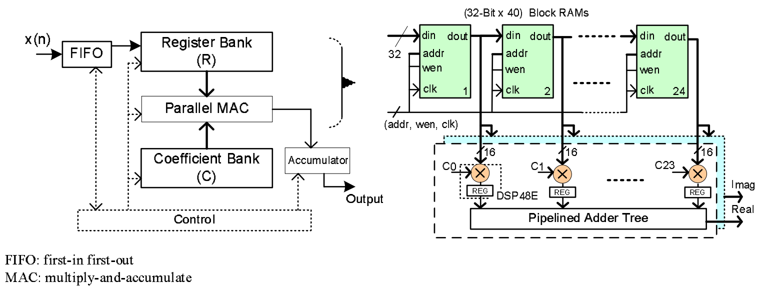


Figure 4: LPA with block RAM based register bank for a polyphase filter operating in under-decimated mode and with a clock rate higher than the input data rate.

The LPAs for the five embedded resampling cases [3] are based on the architecture shown in Fig. 4 according to the number of subfilters (five), number of subfilter taps (six), and corresponding data loading and filter coefficient sequences. The register and coefficient banks are based on block RAMs. Table 2 shows the operating clock rates for the designed LPAs. These clock rates meet the output time constraints of the five different cases, without overflowing the input FIFO. Cases 4 and 5 have two different load times because two different numbers of data inputs are loaded, as shown in Table 4 and Table 5 in [3].

Case	**Output Time Constraint** (ns)	**Required Clock Rate** **Frequency** (MHz)	**Time Period** (ns)	**Load Time** (ns)	**Process Time** (ns)
1	166.6	5×	6.66	113.3	53.3
2	100	6×	5.55	55.6	44.4
3	200	4×	8.33	133.3	67.7
4	83.5	7×	4.76	57.2, 33.4	38
5	250	4×	8.33	199.9, 166.7	66.7

Table 2: Required operating clock rates for the five embedded resampling cases in a polyphase filter with block RAM based register bank.

The loading of LPA data from the input FIFO into the register bank is controlled by the FIFO empty flag, so the loading process may need to wait for the valid data sample in the FIFO. The loading time shown in Table 2 incorporates this waiting time. From Table 2, on average, approximately 60% of the loading time is used for loading the register bank. This eventually requires high operating clock rates in order to process the loaded data within the output time constraints. Although block RAMs do not require CLBs, the synchronous write and read operations require two clock cycles for loading and shifting the input data. Thus, the clock rates must be increased. On the other hand, the distributed RAM based register bank has asynchronous read operation and only requires one clock cycle for loading and shifting. Therefore, the overall clock rate is reduced, but CLB resources are now required.

In the LPA with distributed RAM based register bank, the coefficient bank is based on block RAM which requires the coefficient pointer to be one clock cycle ahead of the data pointer that drives the distributed RAM based register bank. The operating clock rates for the LPA using distributed RAM based register bank are listed in Table 3. These clock rates meet the output time constraints of the five different cases, without overflowing the input FIFO.

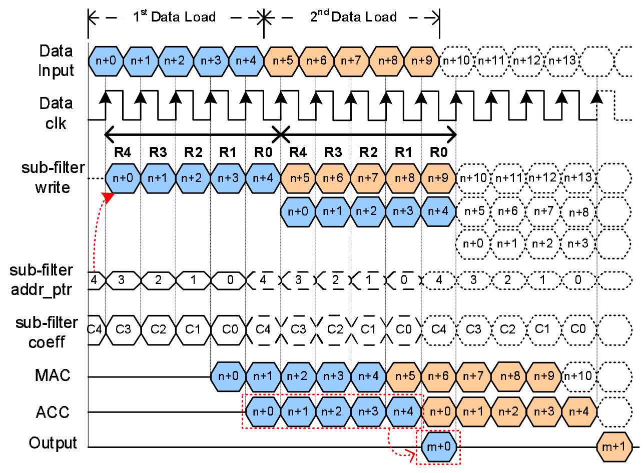
Case	Output Time Constraint (ns)	Required Clock Rate		Load Time (ns)	Process Time (ns)
		Frequency (MHz)	Time Period (ns)		
1	166.6	4×	8.33	100	66.6
2	100	5×	6.66	53.4, 40.1	53.3
3	200	3×	11.11	100, 122.2	88.8
4	83.5	6×	5.55	50, 27.9	44.4
5	250	3×	11.11	177.8, 144.5	88.8

Table 3: Required operating clock rates for the five embedded resampling cases in a polyphase filter with distributed RAM based register bank.

From Table 2 and Table 3, it can be seen that using distributed RAM based register banks, instead of block RAM based register banks, reduces the required operating clock rates for the LPA by approximately 20%. High clock rates for LPA may limit the polyphase filter design to a lower input sample rate because the technology platform has limited ability to achieve high operating clock rates. To overcome the high clock rate demands of the LPA, we introduce the runtime architecture (RA), which runs at a maximum of double the input data rate for the five cases. At double the input data rate, it efficiently schedules the processes of M subfilters within N data load times while simultaneously loading N subfilters. It uses a distributed RAM based register bank and also eliminates the input FIFO for bridging the input data rate with the high processing clock rate. The RA resembles the LPA (Fig. 4) but has a different configuration of distributed RAM based register bank, lacks FIFO, and has a run-time scheduling instead of a load-process scheduling. Here we present the scheduling schemes for the RA for the five embedded resampling cases [3].

Case 1:
Fig. 4a [3] shows the first two loading cycles for the maximally decimated system. The timing diagram for these cycles (Fig. 5) shows the scheduling of the filter operations and

input data loading at the input rate. The parallel multipliers and adder tree take four clock cycles to generate the subfilters' MAC output, which are further accumulated over five MAC outputs to get the output of the polyphase filter (channel 0).

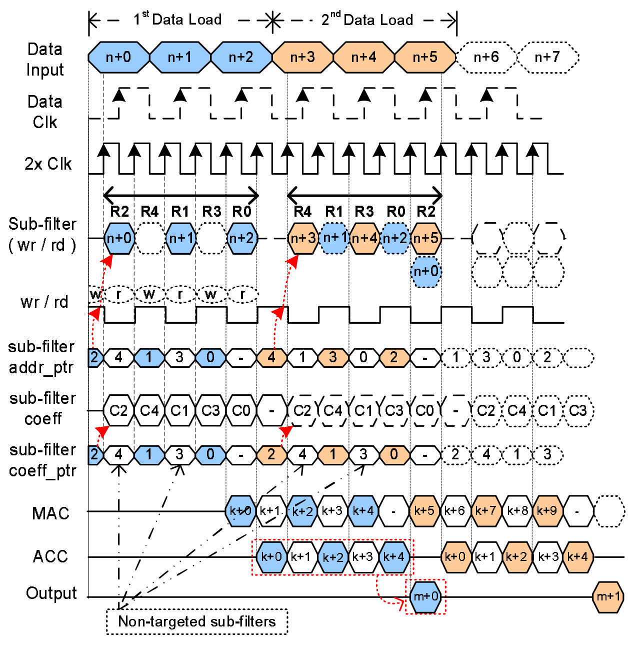


ACC: accumulate
MAC: multiply-and-accumulate

Figure 5: Scheduling of the filter operations (at input clock) and input data loading in a maximally decimated system (case 1).

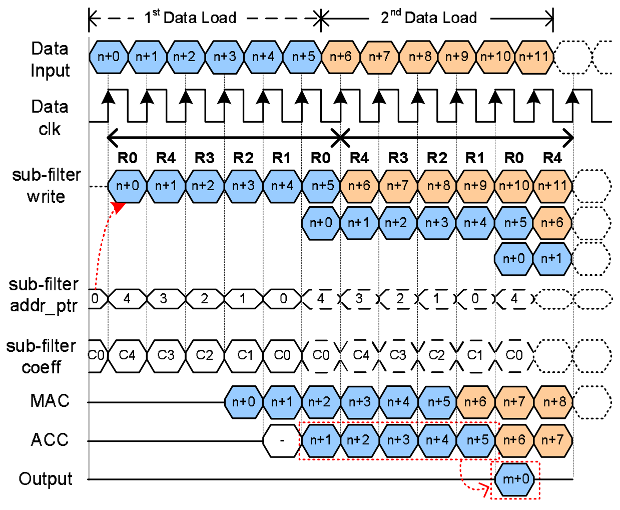
Case 2:
Fig. 4b [3] shows the first two loading cycles for the under-decimated system. The timing diagram for these cycles (Fig. 6) shows the scheduling of the filter operations and input data loading at double the input rate.
The first cycle of the doubled-rate ($2\times$) clock processes the targeted (data-loaded) subfilter while the second cycle of the $2\times$ clock processes the subfilter that is not targeted. The three input data periods have six periods of $2\times$ clock that (for the first load cycle) schedule the processing of the non-targeted subfilters [R4, R3] by interlacing with the processing of the targeted subfilters [R2, R1, R0]. The sixth $2\times$ clock cycle is not used because a new data sample has not yet arrived. The parallel multipliers and adder tree take four cycles of $2\times$ clock to generate the subfilters' MAC output, which is further accumulated over five MAC outputs to obtain the output of the polyphase filter.

Case 3:
Fig. 4c [3] shows the first two loading cycles for the over-decimated system. The timing diagram for these cycles (Fig. 7) shows the scheduling of the filter operations and input data loading at the input rate. The loading and filter processing is the same as that described in case 1, but six input data samples are loaded instead of five, and the accumulation process is modified. The R0 subfilter in the first load cycle accepts two data loads and is processed separately. Only the second processed output for the subfilter (which is loaded twice) contributes to the final accumulation. This second processed output for R0 subfilter also has the contribution from the first loaded data sample, so the MAC outputs from $n + 1$ to $n + 5$ are accumulated for the final output.



ACC: accumulate
MAC: multiply-and-accumulate

Figure 6: Scheduling of filter operations (at 2× clock) and input data loading for an under-decimated system (case 2).



ACC: accumulate
MAC: multiply-and-accumulate

Figure 7: Scheduling of the filter operations (at input clock) and input data loading for an over-decimated system (case3).

Case 4:

Fig. 4d [3] shows the two loading cycles for the system up-sampled by two and down-sampled by five. The timing diagram for these cycles (Fig. 8) shows the scheduling of the filter operations and the input data loading at double the input rate. The data loading cycle is periodic for the two states having three and two data inputs, respectively. These two states correspond to six and four cycles of 2× clock to process five subfilters. In the first state, which has three data inputs, six cycles of the 2× clock can process five subfilters. However, the second state, which has two data inputs with four 2× clock cycles, lacks a clock cycle to process the fifth subfilter. This required processing can be achieved either by using a 3× clock to provide more clock cycles or by efficiently using the non-utilized cycle of the 2× clock in the first state. In the first state, the processing of the targeted sub-filters [R4, R2, R0] is interlaced with the processing of non-targeted subfilters [R3, R1]. In the second state, processing of the non-targeted subfilters [R4, R2, R0] is interlaced with the processing of the targeted subfilters [R3, R1].

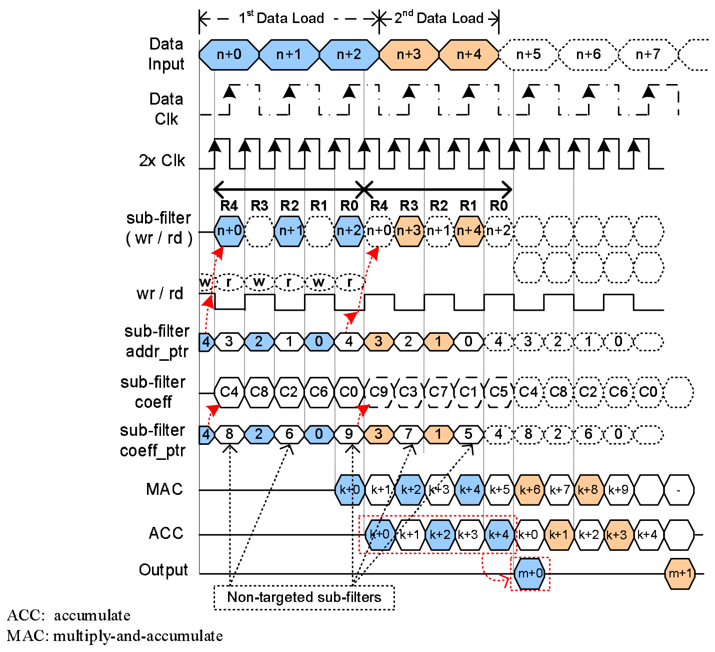
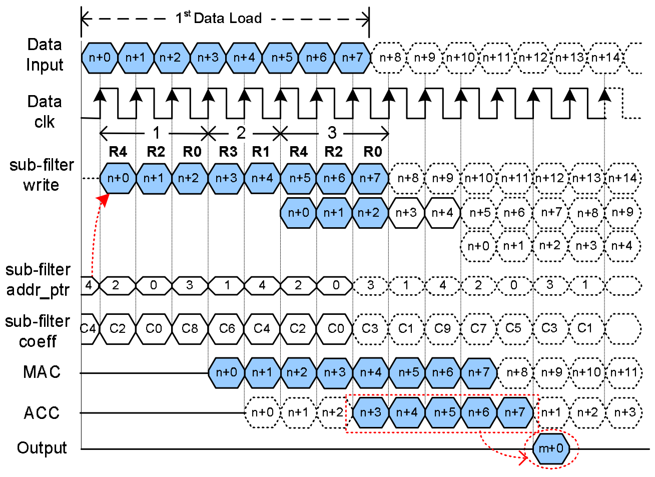


Figure 8: Scheduling of filter operations (at 2× clock) and input data loading for a system up-sampled by two and down-sampled by five (case 4).

Case 5:

Fig. 4e [3] shows the two loading cycles for the system up-sampled by two and down-sampled by 15. The timing diagram for these cycles (Fig. 9) shows the scheduling of filter operations (for one loading cycle) and the input data loading at the input rate. The loading process continues for three stride lengths of the commutator, and the eight input data samples undergo filter processing. The outputs of the subfilter MACs are accumulated from outputs $n + 3$ to $n + 7$ because these outputs contribute to the final accumulation (which includes outputs from the twice-loaded subfilters).



ACC: accumulate
MAC: multiply-and-accumulate

Figure 9: Scheduling of filter operations (at input clock) and input data loading for a system up-sampled by two and down-sampled by 15 (case 5).

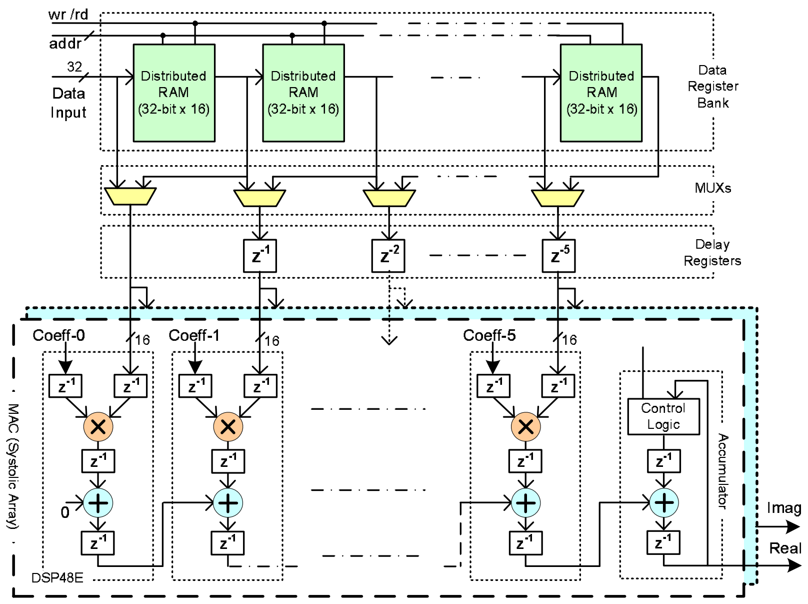
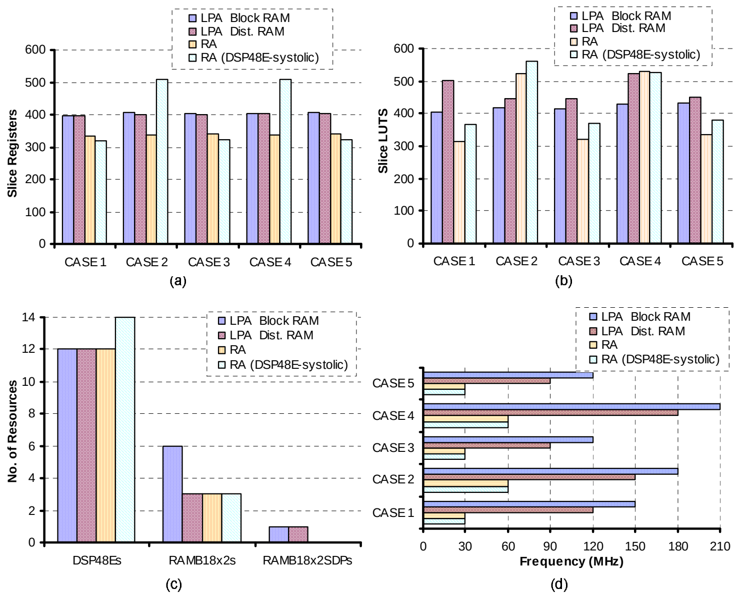


Figure 10: Runtime architecture with DSP48E systolic-array based MAC. The data from the register bank is fed through a set of registers that delay the parallel data, element by element, to align the timing of the subfilter MAC and final accumulation process.

We have so far described the scheduling schemes for the subfilters' processes along with their data loading at the input data rate or double the input data rate. Their architectures have DSP48E slice-based multipliers and CLB-based adder trees that limit the overall operating clock rate to approximately 200 MHz. However, the clock rate can be improved by performing MAC operations in systolic array of DSP48E slices. To perform MAC operations in systolic arrays, the parallel data from the data register bank needs to be time-aligned. The parallel data is fed through a set of registers that delay the data, element by element, in order to align the subfilter's MAC and final accumulation process within the systolic-array based MAC (Fig. 10). The RA with DSP48E systolic-array based MAC has high latency because of the pipeline and delay registers, but it increases the maximum operating clock rate to 350 MHz. The multiplexer block before the delay elements switches between the processing of the targeted subfilter and non-targeted subfilter (which is used only in cases 2 and 4).

4 Resource Usage

We have presented LPA with block RAM and distributed RAM based register banks as well as RA (distributed RAM based register bank) having DSP48E slices based multipliers and CLB based adder tree, and RA with DSP48E slice systolic-array based MAC. These architectures are mapped onto a Virtex-5 FPGA in the form of I/Q components.



DSP48E: dedicated computational resource on Virtex-5 FPGA
RAMB18x2: dedicated memory resource on Virtex-5 FPGA
LPA: load-process architecture
RA: runtime architecture

Figure 11: Resource usage for the five embedded resampling cases under LPA and RA: (a) slice registers, (b) slice LUTs, (c) dedicated resources, and (d) required operating clock rates.

Fig. 11 shows the resource usage in the FPGA for the five embedded resampling cases. The slice registers, slice LUTs, dedicated resources, and required operating clock rates are shown for both LPAs and RAs. In all five cases, the LPA block RAM and LPA distributed RAM (based register banks) have almost the same slice register usage, that is, 397 to 407 slice registers. However, in all five cases, there is a difference in slice LUT usage between LPA block RAM and LPA distributed RAM. The usage in LPA block RAM ranges from 406 to 431 slice LUTs, and the usage in LPA distributed RAM ranges from 447 to 502 slice LUTs. This is due to the fact that distributed RAMs (used for the data register bank) require LUTs. The variation in slice LUT usage within LPAs block RAM and LPAs distributed RAM is due to the different sets of states and control sequences for the polyphase engines with different embedded resampling factors. The RA in all the embedded resampling cases uses almost the same number of slice registers ($335 - 342$) and almost the same number of slice LUTs ($315 - 334$). The exceptions are case 2, which has 524 slice LUTs and case 4, which has 529 slice LUTs. These two systems do not have straightforward indices for accessing their data register and filter coefficient banks. Therefore, the indices for data register and filter coefficient banks are pre-stored in LUTs instead of being generated by sets of counters. The architecture also contains the multiplexer block for switching between targeted and non-targeted subfilter processing that uses slice LUTs and slice registers. Similarly, the RA with DSP48E systolic array based MAC - for case 1, case 3, and case 5 - have almost the same number of slice registers ($318 - 324$) and LUTs ($366 - 381$). The RA with DSP48E systolic-array based MAC - for case 2 and case 4 - use slightly more slice registers (510) and LUTs ($528 - 563$) because of the pre-stored indices for data register and filter coefficient banks, and multiplexer block.

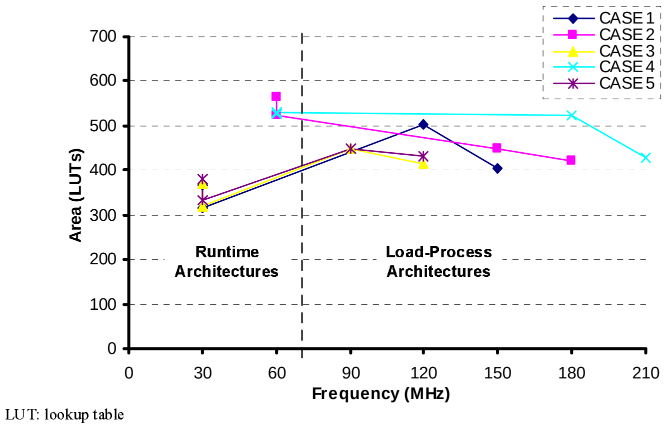


Figure 12: Area versus (operating clock) frequency for the four design solutions for the five embedded resampling cases.

The MAC architecture for LPA and RA has DSP48E slice-based multipliers and CLB-based adder trees. All the cases based on this MAC architecture uses 12 DSP48E slices. However, the RA with DSP48E systolic-array based MAC uses two extra DSP48E slices for the final accumulation process. Similarly, the filter coefficient banks (which are based on block RAMs) in both LPA distributed RAM and RAs uses three dedicated RAMB18x2s [11] resources. However, the LPA block RAM uses three extra resources of RAMB18x2s for its block RAM based register bank. Furthermore, the LPAs use one RAM18x2SDPs [11] for the input FIFO (which is zero for the RAs because they do not use input FIFO).

Fig. 12 shows the number of LUTs versus (operating clock) frequency for the four design solutions for the five embedded resampling cases for the polyphase filter. The RAs for case 1, case 3, and case 5 require less area and lower processing clocks than their LPA counterparts. Similarly, the RAs for case 2 and case 4 require more area but use lower processing clocks than their LPA counterparts. This larger area is due to the pre-stored indices for addressing data register and filter coefficient banks, and also due to the use of multiplexers. Thus, RA is the preferred choice for reduced operating clock rates, and also for reduced area resources with the exception of cases 2 and 4.

5 Power Analysis

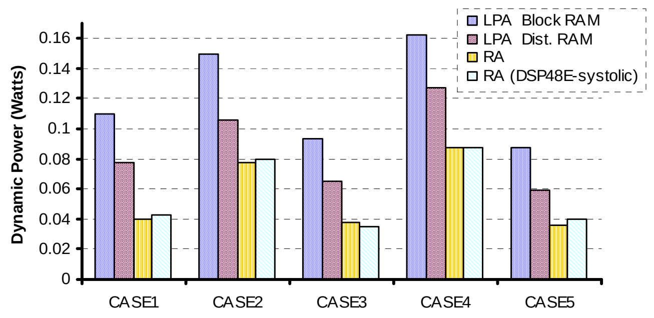
Here we analyze power consumption, focusing on dynamic power (CMOS technology is assumed) for the LPA and RA of the polyphase filter with five different resampling factors. The demand for high clock rates in the LPA is often difficult to satisfy, and the architectures are power hungry because the dynamic power is proportional to the toggle frequency [12]:

$$P_{dynamic} = nCV^2 f \tag{1}$$

where n is the number of nodes being toggled, C is the load capacitance per node, V is the supply voltage, and f is the toggle frequency. C and V are device parameters, whereas n and f are design parameters. By keeping almost the same n and lowering f, power consumption decreases. As reduced clock operation has the same time constraint as high clock operation, energy consumption is reduced as well.

Fig. 13 shows dynamic power consumption for LPA and RA in the five embedded resampling cases. Xilinx XPA [13] tool is used for the analysis. The input vector is the same for all the cases and designs, and the activity rates are calculated using ModelSim [14] post-route simulations with a run length of 500 us. The thermal settings for the power simulations are 25 degrees Celsius-an ambient temperature, and zero airflow.

Fig. 13 shows that the LPA with block RAM based register bank consumes more dynamic power than LPA with distributed RAM based register bank and also more dynamic power than RAs. The maximally decimated system (case 1) based on RAs consumes 64% less dynamic power than its LPA counterpart when block RAM based register banks are used, and it consumes 49% less dynamic power than its LPA counterpart when distributed RAM based register banks are used. The under-decimated system (case 2) based on RAs consumes 48% less dynamic power than its LPA counterpart when block RAM based register banks are used, and it consumes 27% less dynamic power than its LPA counterpart when distributed RAM based register banks are used. The over-decimated system (case 3) based on RAs consumes 60% less dynamic power than its LPA counter-



DSP48E: dedicated computational resource on Virtex-5 FPGA
LPA: load-process architecture
RA: runtime architecture

Figure 13: Dynamic power analysis for all four design solutions for the five embedded resampling cases for the polyphase filter.

part when block RAM based register banks are used, and it consumes 42% less dynamic power than its LPA counterpart when distributed RAM based register banks are used. The up-sampled by two and down-sampled by five system (case 4) based on RAs consumes 46% less dynamic power than its LPA counterpart when block RAM based register banks are used, and it consumes 31% less dynamic power than its LPA counterpart when distributed RAM based register banks are used. The up-sampled by two and down-sampled by 15 system (case 5) based on RAs consumes 59% less dynamic power than its LPA counterpart when block RAM based register banks are used, and it consumes 40% less dynamic power than its LPA counterpart when distributed RAM based register banks are used. Thus, RA is superior to LPA for reducing dynamic power and clock rates. The five cases are representative for polyphase filter banks that perform embedded sample rate changes. The analysis also shows that the dynamic power for RA with DSP48E systolic-array based MAC in cases 1, 2 and 5 is between 3% to 11% higher than their RA with DSP48E slice based multipliers and CLB-based adder tree.

FPGAs provide high parallelism and reprogrammability but at the expense of additional signal routing resources and higher static power consumption due to transistor leakage from parasitic diodes in gate junctions [15], [16]. The static current dissipates power when the device is powered-up and no logic is being clocked. The drive toward smaller transistors in FPGA is necessary for achieving higher chip density and faster dynamic speed. This, in turn is necessary for embedding specialized DSP functionality blocks, but it substantially increases the current leakage. As the size of transistors in FPGA drops to 70 nm, the current leakage becomes more dominant and accounts for more than 50% of power consumption [17]. The clocked (dynamic) power is added to basic static power when logic is active. Both these power contributions substantially increase as the device junction temperature increases. For the targeted Virtex-5 FPGA at 65 nm, static power consumption in all the cases of embedded resampling is around 0.660 W, which is significantly higher than the highest dynamic power consumption of 0.162 W. However, static power consumption is technology-dependent, and only the area and clock rates can be reduced to minimize the overall power consumption in the system.

6 Conclusion

We have described the architecture and FPGA implementation of a polyphase engine that performs embedded resampling. We have analyzed the architectures for five different embedded resampling scenarios in polyphase filter banks: 1) maximally decimated, 2) under-decimated, 3) over-decimated, and combined up- and down-sampling with 4) single and 5) multiple stride lengths of the commutator that feeds the input data into the filter bank. These scenarios are applicable to any required rational sampling rate change in a polyphase channelizer-based SDR front-end. We have described the FPGA-based architectures for a serial polyphase structure with parallel MAC that has load-process and runtime scheduling. The LPA first loads the selected variable data length and then processes the subfilters that require a high clock rate. The RA, on the other hand, operates at a maximum of double the input data rate, which enables scheduling of subfilter processes along with the data loading. We have also described different design techniques for polyphase register bank mapping. A detailed analysis of area, time, and power in the two types of architectures with different resampling factors has been given. This analysis showed that RA is superior to LPA in reducing operating clock rates and dynamic power. RA is also superior in reducing area resources, except where indices are pre-stored in the LUT. Thus, RA is capable of satisfying the need for minimal area, clock frequency, and power consumption in SDR front-ends. From our FPGA implementation analysis, we have derived a set of valuable guidelines that can be used by system designers to create an SDR front-end that is optimized in terms of area, time, and power consumption given certain design specifications. Future work could include a similar analysis for FPGAs from other manufacturers. This would be valuable in generalizing the suggested hardware architectures.

References

[1] f. harris, C. Dick, and M. Rice, ”Digital receivers and transmitters using polyphase filter banks for wireless communications”, *IEEE Trans. Microw. Theory Tech.*, vol. 51, no. 4, pp. 1395-1412, 2003.

[2] f. harris and C. Dick, ”Performing simultaneous arbitrary spectral translation and sample rate change in polyphase interpolating or decimating filters in transmitters and receivers,” in *Proc. Software Defined Radio Tech. Conf. and Product Expo*, San Diego, CA, Nov. 2002.

[3] M. Awan, Y. L. Moullec, P. Koch, and f. harris, ”Polyphase filter banks for embedded sample rate changes in digital radio front-ends,” *Special Issue on advances in Digital Front-End and Software RF Processing: Part II, ZTE Communications*, vol. 9, no. 4, pp. 3-9, Dec. 2011.

[4] D. Liu, A. Nilsson, E. Tell, D. Wu, and J. Eilert, ”Bridging dream and reality: programmable baseband processors for software-defined radio”, *IEEE Commun. Mag.*, vol. 47, no. 9, pp. 134-140, Sept. 2009.

[5] P. Koch and R. Prasad, ”The universal handset”, *IEEE Spectrum*, vol. 46, no. 4, pp. 36-41, 2009.

[6] Raghu Rao et al., ”FPGA Polyphase Filter Bank Study & Implementation”, Presentation, Image Communications/Reconfigurable Computing Lab. Electrical Engineering Dept. UCLA [Online]. Available: `http://slaac.east.isi.edu/presentations/retreat_9909/polyphase.pdf`

[7] Patrick Murphy, ”Digital Communications using FPGAs & Xilinx System Generator”, Presentation, October 5-6, 2006, IIT Delhi, India. [Online]. Available: `http://cmclab.rice.edu/workshops/materials/2006-10-05_DigitalComm/Rice_Comm_Workshop_Slides.pdf`

[8] Xilinx DS100 (v5.0), ”Virtex-5 Family Overview”, Feb. 6, 2009. [Online]. Available: `http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf`

[9] Xilinx UG193, ”Virtex-5 FPGA XtremeDSP Design Considerations User Guide”, Jan. 12, 2009. [Online]. Available: `http://www.xilinx.com/support/documentation/user_guides/ug193.pdf`

[10] M. Awan, P. Koch, C. Dick, and f. harris, ”FPGA implementation analysis of polyphase channelizer performing sample rate change required for both matched filtering and channel frequency spacing”, in *Proc. 44^{th} Asilomar Conference on Signals, Systems, and Computer*, CA, Nov. 7-10, 2010, pp. 414-418.

[11] UG190 (v5.3), ”Virtex-5 FPGA User Guide”, May 17, 2010. [Online]. Available: `http://www.xilinx.com/support/documentation/user_guides/ug190.pdf`

[12] Xilinx WP285 (V1.0), ”Virtex-5 FPGA System Power Design Considerations”, Feb. 14, 2008. [Online]. Available: `http://www.xilinx.com/support/documentation/white_papers/wp285.pdf`

[13] Xilinx ISE 10.1 Design Suite Software Manuals and Help, 2008. [Online]. Available: `http://www.xilinx.com/itp/xilinx10/books/manuals.pdf`

[14] ModelSim SE User’s Manual, Software Version 6.5, Jan. 22, 2009. [Online]. Available:`http://portal.model.com/d/download_request.asp`

[15] A. D. Garcia, L. F. Gonzalez Perez, R. F. Acuna, ”Power consumption management on FPGAs”, in *Proc. 15th International Conf. on Electronics, Communications and Computers*, Puebla, Feb. 2005, pp. 240-245.

[16] Thales, ”Design considerations for size, weight, and power (SWAP) constrained radios”, in *Proc. Software Defined Radio Technical Conf. and Product Expo.*, Nov 14, 2006.

[17] A. Andrei, M. T. Schmitz, P. Eles, Zebo Peng, and B. M. Al Hashimi, ”Quasi-static voltage scaling for energy minimization with time constraints”, in *Proc. Conf. Design, Automation and Test in Europe*, Vol. 1, Mar. 2005, pp. 514-519.

Paper C

Time and Power Optimizations in FPGA-Based Architectures for Polyphase Channelizers

Mehmood Awan, fred harris, and Peter Koch

This paper was published in:
45th Asilomar Conference on Signals, Systems, and Computers

Abstract

This paper presents the time and power optimization considerations for Field Programmable Gate Array (FPGA) based architectures for a polyphase filter bank channelizer with an embedded square root shaping filter in its polyphase engine. This configuration performs two different re-sampling tasks required for spectral shaping and for an M-channel channelizer. In an under-decimated (non-maximally decimated) polyphase filter bank scenario, where the number of data-loads is less than the number of sub-filters, the serial polyphase structure with parallel MAC approach requires a larger processing time than the corresponding data-load time. In order to meet the output time constraint, the serial polyphase structure with parallel MAC has to run at a higher clock rate than the data input rate and hence potentially consumes high power. In contrast to the Load-Process Architecture (LPA), a Run-time Architecture (RA) operating only at twice the input data rate is presented which efficiently schedules the sub-filter's processing within the data-load time. The RA offers time and power efficient structure for the presented up- and down-sample polyphase filters utilizing 9% and 11% slice LUTs and 10% and 13% slice register resources of a Xilinx Virtex-5 FPGA, operating at 400 and 480 MHz, and consuming 1.9 and 2.6 Watts of dynamic power, respectively.

1 Introduction

Traditional polyphase filter banks are formed from three building blocks, an M-port commutator, an M-path polyphase partition of a prototype low-pass filter and an M-point IFFT. These blocks can be arranged to form an M-channel channelizer for modulators and demodulators. The up- and down-converters based on maximally decimated filter banks are dual graphs which by virtue of their linear time varying structures perform the opposite tasks of up-sample and down-sample channelization. The non-maximally decimated version of polyphase filter banks performing dual sample rate change both for up- and down-sample channelization is presented in [1]. Fig. 1 and Fig. 2 show the down- and up-sample channelizers for a system with 32 channels separated by 6 MHz center frequencies and each having a 5 MHz symbol rate shaped by a square-root Nyquist filter with 20% excess bandwidth.

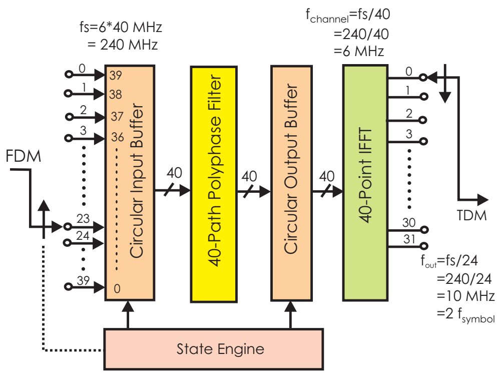


Figure 1: 40-path polyphase FDM-to TDM channelizer with 24-to-1 down-sampling [1].

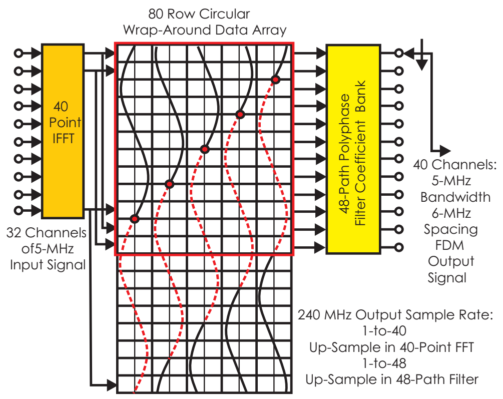

80 Row Circular
Wrap-Around Data Array
40 Point IFFT
48-Path Polyphase Filter Coefficient Bank
32 Channels of5-MHz Input Signal
40 Channels: 5-MHz Bandwidth 6-MHz Spacing FDM Output Signal
240 MHz Output Sample Rate: 1-to-40 Up-Sample in 40-Point FFT 1-to-48 Up-Sample in 48-Path Filter

Figure 2: Two dimension partition of prototype filter with lowest frequency sinusoid of 40 samples extended to 48 samples and shifted in stride of 48 and phase aligned by 8-samples shift [1].

The down-sample channelizer shown in Fig. 1 uses a 40-path polyphase channelizer. To achieve 2 samples/symbol rate, a 24-to-1 down-sampling within a 40-path channelizer is required. This 3/5 fractional rate change is accomplished by adding a cyclic input data buffer and a cyclic output data buffer between the polyphase filter and the IFFT along with a modified commutator operation that feeds 24 samples instead of 40 samples at a time. The Time Division Multiplexing (TDM) to Frequency Division Multiplexing (FDM) counter part i.e., up-sample channelizer shown in Fig. 2, uses the same 40-point IFFT with output spacing of 6 MHz. The shaping filter increases the input symbol rate of 5 MHz to 240 MHz by performing 1-to-48 interpolation. This is accomplished by an extended circular buffer in the interface between the commutator and the polyphase filter which is circularly shifted 8-samples prior to accepting the next 40-point vector from the 40-point IFFT. A second half extension buffer holds the copied first half which makes it easy to have 8-sample shift for 40-sample data.

These under-decimated polyphase filter banks, where the number of data loads (N) is less than the number of sub-filters (M), require the processing of M sub-filters in a time period of N data loads, so run-time processing at the input data rate for the serial polyphase structure (with parallel MAC) is not possible. An LPA for the serial polyphase structure (with parallel MAC) for these under-decimated systems and their FPGA implementation analysis for Xilinx Virtex-5 FPGA is presented in [2]. Fig. 3 shows the LPA with Block-RAM based register bank for a polyphase filter, which runs at a higher clock rate and uses a FIFO to interface with the input data at lower rate. During the load phase, the data elements are fed from the input FIFO and loaded to the sub-filters as directed by the data pointer. In the processing phase, coefficient and data pointers having embedded shifts select the coefficient and data memory elements to perform MAC operations. The multipliers are based on DSP48E slices [3] (dedicated DSP blocks in Xilinx FPGA) whereas the additions are performed by using a CLB based adder-tree network. The architecture loads the register bank with N data elements from the input FIFO and then process M sub-

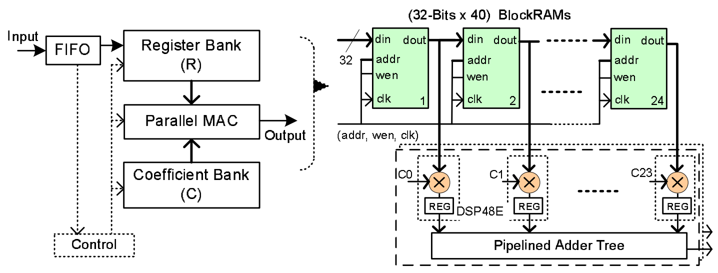


Figure 3: LPA with BlockRAM based register bank for a polyphase filter operating in under-decimated mode, running at a clock rate higher than input data rate.

filters while the FIFO is being loaded. As the FIFO is continuously loaded and unloaded only during the process cycles, the FIFO eventually overflows. In order to overcome the FIFO's overflow and to meet the output time constraints, the polyphase filters for the up- and down-sample channelizers have to run at 4 times the input clock rates i.e., 4×200 MHz and 4×240 MHz, respectively. These high demanding clock rates based designs are not only difficult to achieve but at the same time the architectures becomes power hungry, since the dynamic power is proportional to the toggle frequency, as described by the following equation [4]:

$$P_{dynamic} = nCV^2 f \tag{1}$$

where n is the number of nodes, C is the load capacitance, V is the supply voltage and f is the toggle frequency. C and V are device parameters whereas n and f are design parameters. So, by keeping (almost) the same n (area consumption) and lowering f, the power consumption can be reduced. As the reduced clock operation meets the same time constraint as the high clock operation, the energy consumption will be decreased as well.

The high demanding clock rates for the LPA may limit the polyphase filter design to a lower input sample rate, as the technology platforms e.g., FPGA has certain limitations on achieving the high operating clock rates. This paper is an extension of the work presented in [2], basically in the sense that the LPA is modified to reduce the required clock rates for the up- and down-sample polyphase filters. It is achieved by introducing distributed RAM based pipelined register bank, systolic array [5] of DSP48E slices based MAC, and rescheduling the sub-filter's operations at 2 times the input clock rate. The FFT/IFFT part of the channelizer is not dealt with in this paper.

In the LPA, the data register bank is based on BlockRAMs which completely eliminates the CLB resource but requires an extra clock cycle for each data load and shifting among the cascaded BlockRAMs. This is due to the fact that BlockRAM has synchronous write and synchronous read operations and is thus a source of increased clock rate. The required clock rate for the data register bank can be reduced in the following ways:

1. The distributed RAM has synchronous write but asynchronous read operation. So, a data register bank formed by distributed RAMs without input and/or output registers will require only one clock cycle to perform the required loading and shifting.

It therefore results in an overall reduced clock rate demand, but requires extended CLB resources. The required operating clock rate for the LPA for the up- and down-sample polyphase filters with distributed RAM based register bank reduces to 3x the input data rates.

2. In the LPA with distributed RAM based register bank, the scheduling of the load and process cycles can also be improved to further reduce the required operating clock rates. The idea is to start the processing of the sub-filters (process-cycle) while data is loading from the FIFO (load-cycle), instead of waiting for the complete loading of the data-loaded (targeted) sub-filters before processing. In this case, the process-cycle continues for M sub-filters, but the data loading is stopped for non data-loaded (non-targeted) sub-filters.

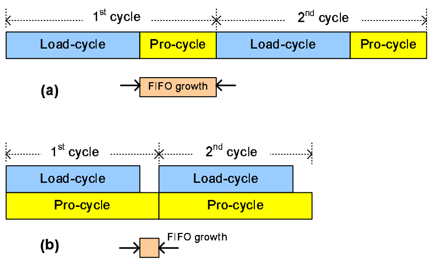


Figure 4: Scheduling of load and process cycles for (a) LPA and (b) interlaced LPA

This is the only time when input data is pushed onto the FIFO without being pulled at the same time. Since the number of non-targeted sub-filters is less than the number of targeted sub-filters, the FIFO does not grow as rapidly as in the previous case where process-cycle starts after completing the load-cycle as shown in Fig. 4a. This interlaced scheme for loading and processing as shown in Fig. 4b reduces the overall clock cycles count and therefore reduces the required clock rate. By operating the sub-filter's processes at 2x clock, the FIFO growth in the (extended) process cycle gets undone in the next couple of sub-filter's processing within the next load cycle. So, the required operating clock rates for up- and down-sample polyphase filter reduces to 2x still meeting the output timing constraints.

3. In interlaced LPA, most of the process cycles which are overlapped by the load cycles are idling as they are waiting for the next available data from the input FIFO. To make efficient use of these idling cycles of 2x clock, an RA is introduced where non-targeted sub-filter's processing is achieved by idling cycles of 2x clock instead of processing them after the load cycle. By doing so, the length of the process cycle becomes equal to the length of the load cycle which will also eliminate the need for the input FIFO.

The RA, operating at twice as high a data clock rate, processes the input data while simultaneously loading to the sub-filters on the first cycle of the 2x clock and the next cycle of the 2x clock get used to process the non-targeted sub-filter. By using this scheme the non-targeted sub-filters get processed within the time frame of the targeted sub-filters and therefore reduce the overheads as in the case of LPA and interlaced LPA.

2 RA Scheduler

Now we present the RA scheduling for the up- and down-sample polyphase filters for the channelizers shown in Fig. 1 and Fig. 2. On the RA for the down-sample polyphase filter, operating at 2x process clock, the 24 input data periods have 48 periods of 2x clocks. It schedules the non-targeted sub-filter's $(R39, R38, ..., R24)$ processing by interlacing it with the targeted sub-filter's $(R23, R22, ..., R0)$ processing during the first state, as shown in Fig. 5. It will be extended accordingly for the 5 periodic states of the polyphase engine. As there are 16 non-targeted sub-filters in each state, the rest of the 8 time slots of the 2x clock allocated for the non-targeted sub-filter's processing remain unused, and we call them invalid sub-filters. The sliding data loading for the 5 periodic states need $48 \times 5 = 240$ distinct addresses for the data address pointer. As the sliding loading of data interacts with the cyclic shifted set of coefficients, the coefficient address pointer needs the same sized distinct addresses as the data address pointer. The unused time slots for the non-targeted sub-filter processing (invalid sub-filter) are addressed by zero-valued data and coefficients at address 40. Because of the interlaced processing of the non-targeted sub-filters with the targeted sub-filters, the final processed outputs need reordering to bring them in their normal order, and delivering them at the required output rate.

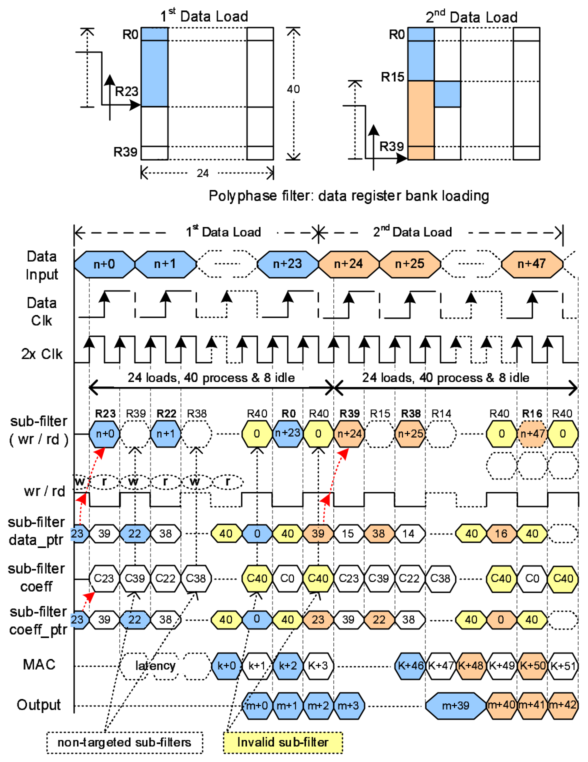


Figure 5: Scheduling of the first two states for RA for the down-sampling polyphase filter. The time-lag for processing non-targeted sub-filters is overcome by using 2x clock and processing non-targeted sub-filters along with targeted sub-filters.

Similarly, for the RA for the up-sample polyphase filter operating at 2x process clock, there will be $40 \times 2 = 80$ time slots for each state to process 40 targeted and 8 non-targeted (extended) sub-filters in a time frame of 40 input samples. The rest of the 32 time slots allocated for the extended sub-filter's processing become unused (invalid sub-filters). In order to incorporate the required circular shifting for the next states, which is periodic over five states, $80 \times 5 = 400$ distinct addresses for the data address pointer are required. In this case the circularly shifted data interacts with the fixed set of coefficients, so the coefficient address pointer is the same for all the 5 distinct states. It therefore requires 80 addresses for the coefficient address pointer, out of which only 48 addresses have valid coefficient indices.

3 Architecture Design

The RA presented in [6] uses a distributed RAM (without input/output register) based register bank and coefficient bank, and a systolic array of DSP48E slices based MAC. The required RA for the up- and down-sample polyphase filters, operating at 2x the data clock rate, has to run at 400 and 480 MHz, respectively. In an effort toward achieving these clock rates, it is noticed that in the RA [6], the data register bank based on distributed RAMs without input/output registers limits the maximum operating clock rates. The bottleneck is the asynchronous path between the cascaded distributed RAMs. This bottleneck can be removed by inserting input /output /pipeline registers for each cascaded distributed RAM. It is seen from Fig. 6a that as the number of cascaded distributed RAM without input/output register increases, their maximum operating clock rate decreases. On the other hand, Fig. 6b shows that a minimum of 500 MHz performance can be achieved even with up to 24 cascades of distributed RAMs with output registers. The results shown in Fig. 6a and 6b are the best case achievable clock rates for the cascaded distributed RAM based register bank together with MAC block for a Xilinx Virtex-5 FPGA [7] which are generated under 1) *timing performance with IOB packing*, 2) *balanced performance*, and 3) *power optimization performance*, design goal and strategies in Xilinx ISE tool [8].

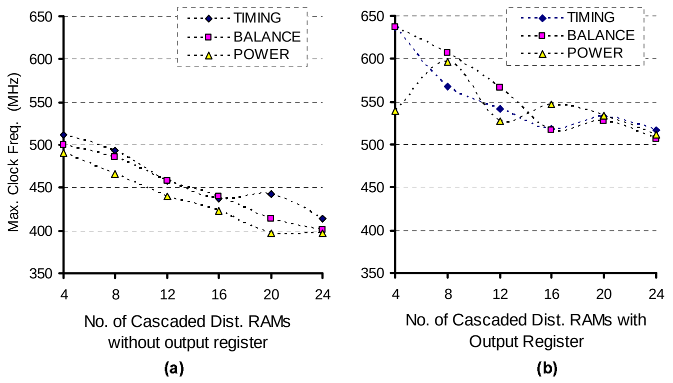


Figure 6: Maximum achievable clock rates for a number of cascaded distributed RAM (a) without input/output registers (b) with output register, and with DSP48E systolic array MAC.

The cascaded distributed RAM without input/output register performs the data load and simultaneously shifts across the cascade in one clock cycle, but it limits the desired achievable clock rates. On the other hand, the distributed RAM with output register meets the targeted clock rates but due to extra register (output register), it requires some changes to balance the control and data paths, and also to achieve data shifts across the cascade in one clock cycle. Fig. 7 shows the data shifting among three cascaded distributed RAMs (a) without and (b) with output registers. This corresponds to a four tap sub-filter for a M-path polyphase filter, where M is equal to the depth of the distributed RAMs. In order to get the one tapped-delay shifting for the cascaded distributed RAM with output register, the pipeline registers are inserted for address and read/write path for each memory. Furthermore, each memory output path is delayed by a total number of cascaded distributed RAMs minus its tap number. This is shown in Fig. 7b where $s0, s1, s2, s3$ represent the tapped outputs from the distributed RAMs which after the corresponding delay elements have $s0', s1', s2', s3$ outputs to form the required tapped delay line. This cascaded distributed RAM pipelined structure for the four-tapped filter is extended to the actual requirements of 20 and 24 cascades for up- and down-sample polyphase filter. The added pipeline register network will add more latency which is a compromise towards achieving high operating clock rates.

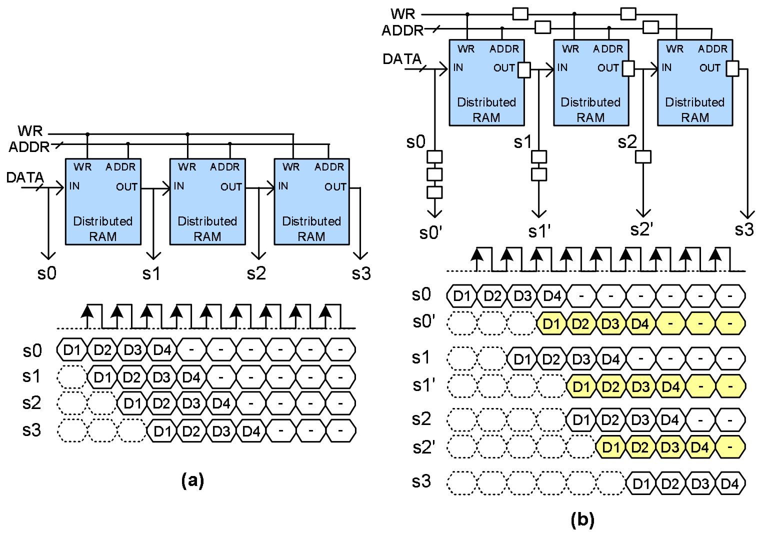


Figure 7: (a) 3 cascaded distributed RAMs without input/output registers, illustrating the data loading and shifting operations for any specific address, (b) 3 cascaded distributed RAMs with output registers and corresponding pipeline registers for control and data paths, illustrating the data loading and shifting operations for any specific address.

Fig. 8 shows the complete architecture for the down-sample polyphase filter with distributed RAM based pipelined register bank, multiplexers, systolic array of DSP48E slices, coefficients' ROMs (distributed RAM), data and coefficient address ROMs (distributed RAM). This architecture needs to be doubled for the complex data processing. The multiplexer at the output of the register bank is used to switch between the targeted and non-targeted sub-filter's processing. The parallel data from the multiplexer block is delayed element-by-element to time align the sub-filter's MAC process in the systolic array of DSP48E slices. The coefficients from the respective coefficient memories being driven by the same address pointer need the same element-by-element delay as the data from the register bank, to time align the coefficients with their corresponding data. This required delay for the coefficients is achieved efficiently by an offset address filling of the coefficient memories instead of using extra delay registers as required by the data path. The coefficient memories are shared by both the real and imaginary data processing unit. The first data input takes 25 clock cycles (24 clock cycles for the tap-0 delays and 1 clock cycle for the multiplexer) to be available for the MAC block which needs to be synchronized to its corresponding coefficient set. This synchronization is achieved by an equivalent advanced count operation in the coefficient address counter. The RA for the up-sample polyphase filter is the same as the down-sample polyphase filter but it has a cascade of 20 distributed RAMs instead of 24, for its register bank, and corresponding coefficient bank and addressing memories.

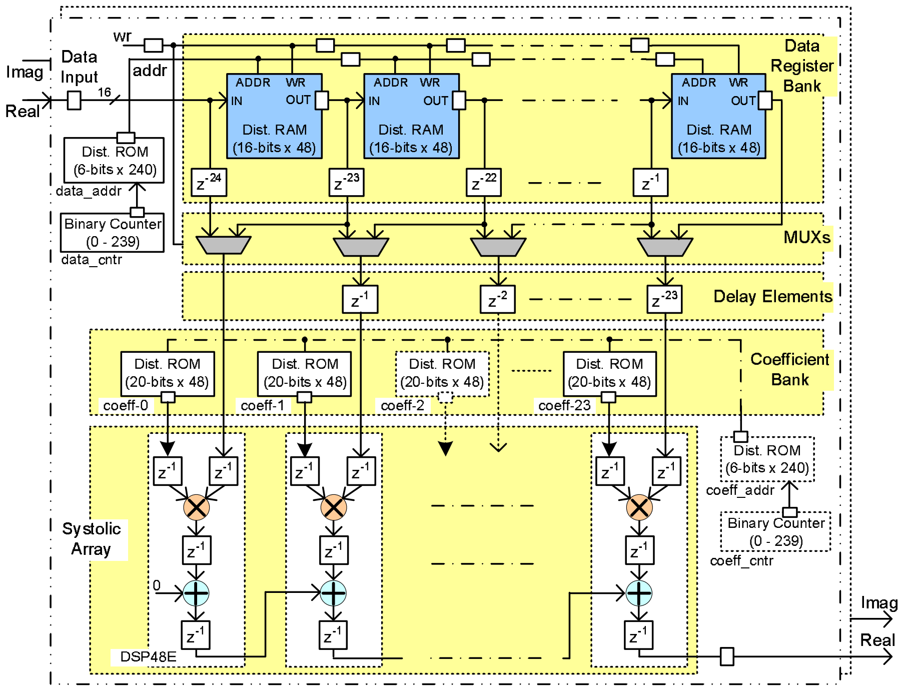


Figure 8: The complete architecture for the down-sample polyphase filter with 24 cascaded distributed RAMs, delay/pipeline registers, multiplexers, systolic array of DSP48E slices, coefficients' distributed-ROMs, data and coefficient address counters, and data address and coefficient address distributed ROMs.

4 Results

The presented RA for the up- and down-sampling polyphase filters are mapped to a Xilinx Virtex-5 (xc5vsx50t-3ff1136) FPGA in the form of I-Q components. The resource usage of the real FPGA implementation for these architectures in terms of (a) slice registers, slice LUTs, and slice LUT flip-flop pairs, and (b) dedicated resources is presented in Fig. 9a and Fig. 9b, respectively. These results are generated under the *timing performance with IOB packing* design goal and strategy, and with *non-resource sharing* and *auto LUT combining* options in Xilinx ISE tool. The resource usage for the slice registers and slice LUTs presented in Fig. 9a is on an average of 3.2 times higher as compared to the usage presented in [2] for their LPA counter parts. This is due to the fact that the RA uses distributed RAM based data and coefficient banks, and added pipeline registers to meet the timing constraints up to 480 MHz. A power consumption analysis focusing on the dynamic power is performed for the RA for the up- and down-sample polyphase filters by using the Xilinx XPA tool [8]. The activity rates based on the input test vectors are calculated by running ModelSim [9] post place and route simulations for a run length of 500 μs. The corresponding thermal settings for these power analyses are 50 degrees Celsius as ambient temperature with 250 LFM air flow and medium profile heat sink. The toggle rates for the flip-flop, I/O and DSP, and the corresponding power consumptions by the up- and down-sample polyphase filters are shown in Fig. 9c and Fig. 9d, respectively. The dynamic power consumptions of 1.9 and 2.6 Watts correspond to the 2x operating clock rates i.e., 400 and 480 MHz for the up- and down-sample polyphase filters, respectively.

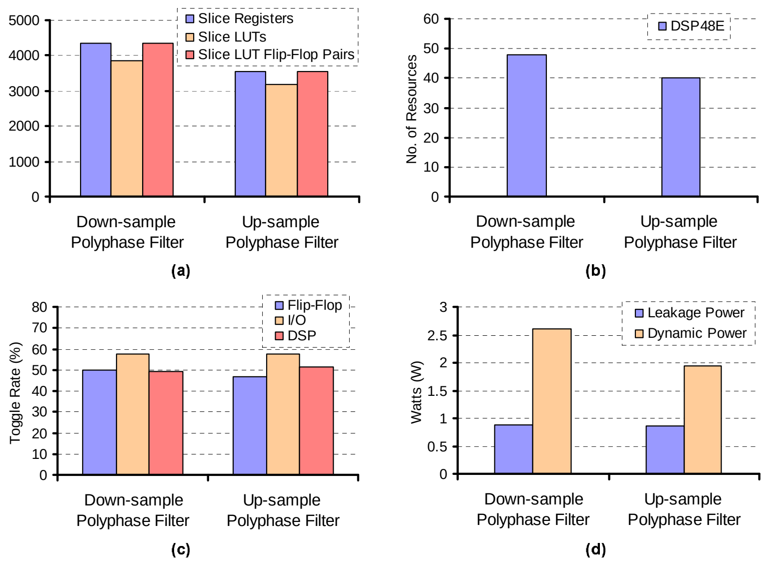


Figure 9: Resource usage in terms of (a) slice registers, slice LUTS, and slice LUT flip-flop pairs (b) dedicated resources for Xilinx Virtex-5 FPGA, (c) toggle rates for the flip-flops, I/Os and DSP, and (d) corresponding power consumption (dynamic and leakage) for the down- and up-sample polyphase filters, respectively.

5 Conclusion

We have presented and demonstrated the RA for the serial polyphase structure with parallel MAC for the up- and down-sampling polyphase filters operating in an under-decimated scenario. In terms of the targeted design goals of Time and Power optimizations, the RA, in contrast to its LPA counterpart, operates at 2x clock rate and efficiently schedules and process the sub-filters within the data load time. The run-time scheduling schemes for embedding the required sliding and cyclic shifting for the up- and down-sampling polyphase filters are presented. The data register bank based on distributed RAMs is modified by inserting pipeline and corresponding balancing registers to achieve the targeted operating clock rates up to 480 MHz on the Xilinx Virtex-5 FPGA. The FPGA resource usage and power consumption analysis is presented. The RA at 2x operating clock rate being the minimum possible rate for the serial polyphase structure with parallel MAC operating in an under-decimated scenario turns out to be a preferable choice for the time and power efficient solution for its FPGA based architectures.

References

[1] fred harris, Chris Dick, ”Polyphase Channelizer Performs Sample Rate Change Required for both Matched Filtering and Channel Frequency Spacing”, in *43rd Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 2009.

[2] Mehmood Awan et al., ”FPGA Implementation Analysis of Polyphase Channelizer Performs Sample Rate Change Required for both Matched Filtering and Channel Frequency Spacing”, in *44th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 2010.

[3] Xilinx UG193 (v3.3), ”Virtex-5 FPGA XtremeDSP Design Considerations User Guide”, Jan 12, 2009.

[4] Abusaidi, Peggy et al., ”Virtex-5 FPGA System Power Design Considerations”, Xilinx WP285 (V1.0), Feb 14, 2008.

[5] H.T. Kung, ”Why Systolic Architectures?”, *IEEE Computer*, vol. 15, no. 1, pp 37-46, Jan, 1982.

[6] Mehmood Awan et al., ”Hardware Architecture Analysis of Polyphase Filter Banks Performing Embedded Resampling for Software Defined Radio Front-Ends”, *100G and Beyond: Trends in Ultrahigh-Speed Communications (Part I), ZTE Communications*, Vol. 10, No. 1, pp. 54-62, Mar 2012. ISSN 1673-5188

[7] UG190 (v5.3), ”Virtex-5 FPGA User Guide”, May 17, 2010.

[8] Xilinx ISE 10.1 Design Suite Software Manuals and Help, 2008.

[9] ModelSim SE User’s Manual, Software Version 6.5, 2009.

Paper D

FPGA Implementation Analysis of Polyphase Channelizer Performing Sample Rate Change Required for both Matched Filtering and Channel Frequency Spacing

Mehmood Awan, Peter Koch, Chris Dick, and fred harris

This paper was published in:
44th Asilomar Conference on Signals, Systems, and Computers

Abstract

The paper presents the architectural domain analysis for FPGA (Field Programmable Gate Array) implementation of a polyphase filter bank channelizer with an embedded square root shaping filter in its polyphase engine that performs two different re-sampling tasks required for spectral shaping and for M-channel channelizer. In terms of algorithms; Radix-2 FFT, Prime Factor and Winograd Fourier Transform are considered for IFFT, where as the polyphase filter is analyzed in terms of symmetric structure, and serial polyphase structures with serial and parallel MAC approaches. The computational workload for these algorithms and their implementation structures are presented together with their hardware mapping to a Virtex-5 FPGA by exploiting the inherent parallelism. Their resource utilizations and Design Space Exploration in terms of Area-Time is presented along with different optimization techniques.

1 Introduction

Traditional polyphase filter bank are formed from three building blocks, an M-port commutator, an M-path polyphase partition of a prototype low-pass filter and an M-point IFFT. These blocks can be arranged to perform M-channel channelizer for modulator (where up to M input time series form an output time series composed of M-equally spaced frequency division multiplexed (FDM) channels) and demodulator (where up to M-equally spaced frequency division multiplexed (FDM) channels are converted to M-interlaced time division multiplexed (TDM) channels). These up-converter and down-converter, based on maximally decimated filter banks, are dual graph which by virtue of their linear time varying structures perform the opposite tasks of up-sample and down-sample channelization [1]. There are a number of different scenarios where we want that the different up-sampling and down-sampling tasks be performed in the channelization process (non-maximally decimated system). Polyphase filter banks performing dual sample rate change both for up-sample and down-sample channelization are presented in [1] where a system has 32 channels separated by 6MHz center frequencies and each having a 5MHz symbol rate shaped by square root Nyquist filter with 20% excess bandwidth. Figures 1 and 2 show the down-sample and up-sample channelizers.

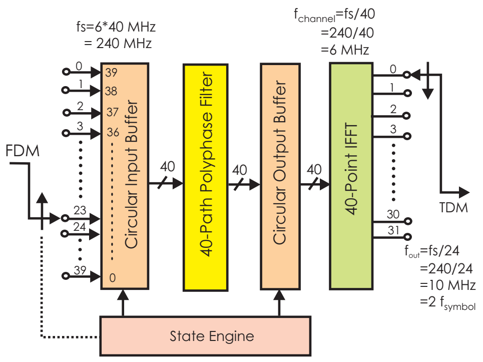


Figure 1: 40-Path Polyphase FDM-to TDM Channelizer with 24-to-1 Down-Sampling [1].

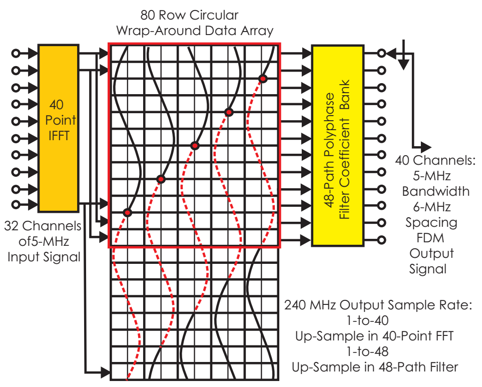


Figure 2: Two Dimension Partition of Prototype Filter with Lowest Frequency Sinusoid of 40 Samples Ex-tended to 48 Samples and Shifted in Stride of 48 and Phase Aligned by 8-Samples Shift [1].

The down-sample channelizer shown in Fig. 1, uses a 40-path polyphase channelizer. To achieve 2 samples/symbol rate, a 24-to-1 down-sampling within the 40-path channelizer is required. This 3/5 fractional rate change is accomplished by addition of a cyclic input data buffer and a cyclic output data buffer between the polyphase filter and IFFT along with modified commutator operation that feeds 24 samples instead of 40 samples at a time [1]. TDM-to-FDM counter part i.e., up-sample channelizer shown in Fig. 2, uses the same 40-point IFFT with output spacing of 6MHz. The shaping filter increases the input symbol rate of 5MHz to 240 MHz by performing 1-to-48 interpolation. This is accomplished by an extended circular buffer in the interface between the commutator and the polyphase filter which is circularly shifted 8-samples prior to accepting the next 40-point vector from 40-point IFFT. The second half extension buffer has the copied first half which makes it easy to have 8-sample shift for 40-sample data [1].

This paper deals with the architectural domain analysis for FPGA implementation of the main building blocks of the structures presented above i.e., the M-point IFFT and the M-path polyphase low-pass filter. The Radix-2 FFT, Prime Factor Algorithm (PFA) and Winograd Fourier Transform Algorithm (WFTA) are considered for IFFT. The polyphase filter is analyzed in term of different implementation structures like symmetric structures, and serial polyphase structure with serial and parallel MAC.

Fig. 3 shows the computational complexities for a 48-path polyphase filter with the following structures; general polyphase structure (P1), symmetric structure (P2), adder shared symmetric structure (P3), serial polyphase structure having serial MAC (P4), and parallel MAC (P5). There is a trade-off between the complexities verses the processing clock speed. Serial polyphase with serial MAC shows the least complexity but demands high clock rate, where as serial polyphase with parallel MAC has slightly higher complexity operating at input clock rate. Fig. 4 shows the computational complexities for Radix-2, PFA, and WFTA algorithms for 40-point (64-point for Radix-2) IFFT. M1, M2, M3, M5 are variants of Radix-2 which indicates the number of butterflies processed at a time [2].

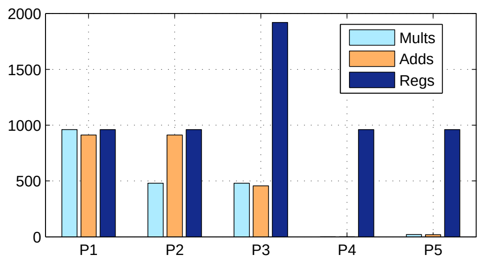


Figure 3: Complexity analysis for 48-path polyphase filter; general polyphase structure (P1), symmetric structure (P2), adder shared symmetric structure (P3), serial polyphase structure having serial MAC (P4), and parallel MAC (P5).

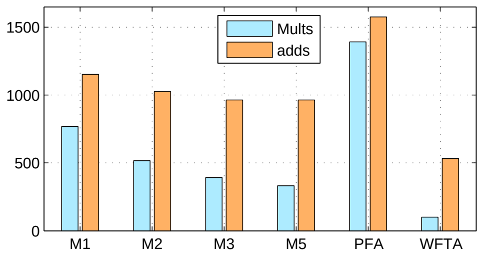


Figure 4: Complexity analysis for 40-point IFFT with Radix-2, Prime factor and Winograd Fourier Transform Algorithms. M1, M2, M3, M5 are variants of Radix-2 FFT.

The WFTA algorithm is the most efficient having only 100 real multiplications for 40-point IFFT.

Based on the computational complexity analysis, the WFTA algorithm and the serial polyphase structures are selected for the mapping to the target platform Virtex-5 (xc5vsx50t-3ff1136). Fixed point analysis is performed to determine the bit widths keeping the quantization errors below 60dB (normally required). The downsample polyphase channelizer has 40 paths filter with 24 taps each. A straight forward mapping of this structure to FPGA where all the filter coefficients and tapped delay lines are CLB slice register based, results in a resource utilization which is more than the slice registers and LUTs on the target platform Virtex-5. It requires various resource optimal techniques to map these structures, both by optimizing the algorithmic structures and by using the dedicated resources within the FPGA. The filter coefficient bank can easily be replaced by FPGA BlockRAMs to optimize the resources, but the critical portion is polyphase partitioned data bank (register bank), especially when we have to perform the embedded resampling within that. In case of a maximally decimated system, where the downsampling factor is equal to polyphase partition, FIFOs can be used as delay lines to have the most optimal structure, as described in [3]. But for non-maximally decimated system which we have in this scenario, a two dimensional RAM based solution is needed. Fig. 5 shows the resource utilization for mapping 24×40 register bank (16-bit) by considering different options available on the Virtex-5 FPGA.

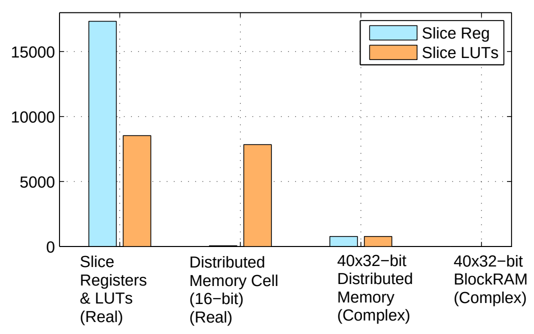


Figure 5: 20×40 Register Bank Resource Utilizations by using Slice Registers and LUTs, Distributed Memory, and BlockRAMs.

The 24×40 register bank based on slice registers and LUTs results in [53, 26]% utilization for slice registers and LUTs respectively. It's only for real data and therefore the numbers would be double for the complex data. Next, each data register in the sub-filters is replaced by a distributed memory based memory cell (16-bit) and the resource utilization (for real data only) comes out to be [0, 24]% for slice registers and LUTs respectively. It saved the slice registers but the LUTs ultilization is almost the same as before. In Virtex-5, each CLB has 64-bit distributed RAM [4], but it is bit-addressable, so for 16-bit data, 16 CLBs are collectively used as a single 16-bit register. The rest of the 63 bits in each CLB remained unused. Based on this analysis, distributed memory is used such that each 64-bit memory contribute 1-bit to 16-bit data element for 64 sub-filters (only 40 are used). So here comes the effectiveness that the memory which was previously used for only one sub-filter is now being used for all the 40 sub-filters. The bit-width is further extended to 32-bit to incorporate complex data (16 + 16 bits) and the resource utilization dramatically decrease to 2% (for complex data). It will also eliminate the need for multiplexers to select the desired sub-filter's data elements for the case of shift register based register bank. The same concept is further applied to BlockRAMs which completely eliminate the CLBs ultilization. The BlockRAMs utilization for (40×32-bit) register bank is 12 BlockRAMs, which corresponds to 9% utilization for Virtex-5.

In order to use BlockRAMs based register banks, we have to pay an extra clock cycle for each data load and shift. The reason is that as 24 memories (40×32-bit) are cascaded to form 24×40 register bank and the data shift in the sub-filters requires data to be available from the preceding memory. So it requires one clock cycle to read the data element and then the next clock will load the new data element along with shifting of the sub-filter, as shown in Fig. 6. According to the polyphase channelizer configurations in [1], the downsampling polyphase filter loads 24 data elements and process 40 sub-filters, whereas the upsampling polyphase filter loads 40 data elements and process 48 sub-filters, so the run time processing at the input sample rate f_s for serial polyphase structure with parallel MAC and N/M times the f_s for serial polyphase structure with serial MAC is not possible. Therefore, the polyphase engine is run at higher clock speeds and a FIFO is used to interface with the input data at lower rate. So a "load and process"

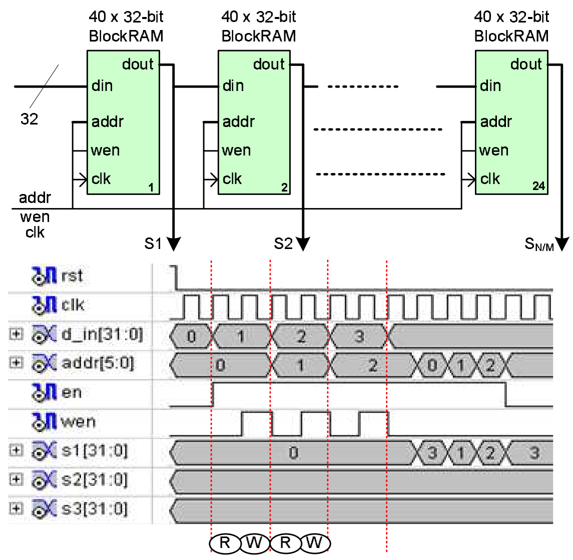


Figure 6: 40×32-bit memory cascade to form 24×40 register bank. One clock cycle to load the date form the memories and another clock cycle to shift the data through memories.

architecture is required where during load, data elements are fed from the input FIFO to sub-filters as directed by the data-pointer to embed the required shift. In the process phase, coefficient-pointer and data-pointer having embedded shifts select the coefficient and data memory elements to perform MAC operations. The load and process sequences for the downsampling polyphase filter which has 4 distinct states, both for its parallel and serial MAC architecture, is presented in Table 1:

Load	**Process**
Load 24 Data Elements	Process 40 Sub-filters
Data Loading Address Sequence: [23, 39, 15, 31, 7]	Data Address Sequence: [0] Coefficient Address Sequence: [0, 24, 32, 16] (Parallel MAC) [0, 24, 32, 16]×24 (Serial MAC)

Table 1: Load & Process sequences for downsampling polyphase filter

In the upsampling polyphase filter, the 40-point IFFT output is extended to 80-points and loaded to a 20×80 register bank, and after circular shifting of 8 samples in stride of 48, only a 20×48 register bank is used for filter processing. This apparently seems to be quite complicated and resource demanding as we are extending the register bank. But by employing a clever address scheme, we can do with only a 20×40 register bank to perform the required circular shift. It has been noticed that each column in the register bank which is (40×32-bit) BlockRAM has a circular shift of multiples of 8 modulo 40. So instead of the common address bus, if each memory has a separate address bus with its own data-pointer having the offset of multiples of 8 modulo 40 from the adjacent memories, we can achieve the required circular shift. The last 8 sub-filters have the same data (or more precisely same data-pointer) as the first 8 sub-filter have, only coefficient pointers are different. It is further noticed that data pointers having offset of multiples of 8 modulo 40 results in 5 distinct pointers which are repeated over for 20 columns. The register bank based on (40×32-bit) BlockRAMs for the upsampling polyphase filter with separated data pointers is shown in Fig. 7. The load and process sequences for the up-

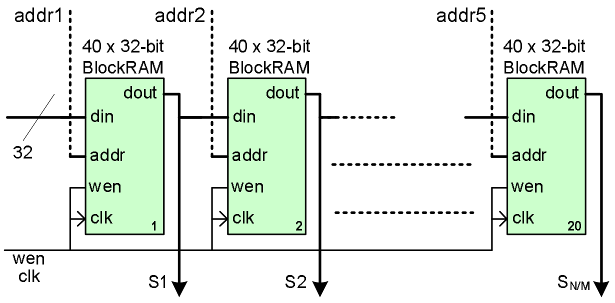


Figure 7: 40×32-bit memory cascade to form 48×20 register bank. It uses separate address buses for each memory block to embed the required circular shift.

sampling polyphase filter, both for its parallel and serial MAC architecture is presented in Table 2.

Load	Process
Load 40 Data Elements Data Loading Address: [0]	Process 48 Sub-filters Data-pointer is modulo-40, so the last 8 sub-filters take the same data as first 8 sub-filters 5 distinct Data-pointer: [0, 8, 16, 24, 32] Modulo 40 Coefficient-pointer: [0] Parallel & Serial MACs

Table 2: Load & Process sequences for upsampling polyphase filter

The filter coefficient bank is mapped on to ROMs by using BlockRAMs resources, both for parallel and serial MAC architecture. For the parallel structure, each column in the coefficient bank corresponds to a separate ROM, whereas the serial MAC structure uses a single large ROM. The final resource utilization for the downsampling and upsampling polyphase filters based on the serial polyphase structure with parallel and serial MAC is

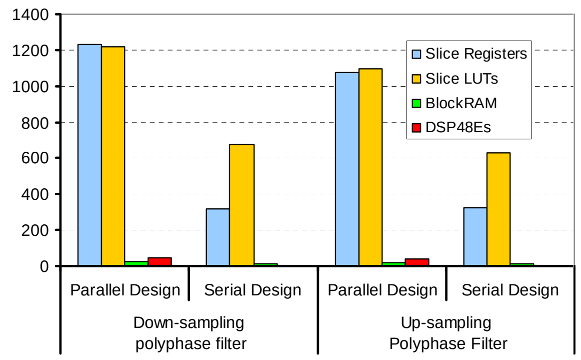


Figure 8: Resource Utilization (in terms of Slice Registers, Slice LUTs, BlockRAMs and DSP48e) for the upsampling and downsampling polyphase filters for their serial polyphase architectures with parallel and serial MACs.

presented in Fig. 8. The downsampling polyphase filter utilizes 3% of CLBs, 18% of BlockRAMs and 16% of DSP48e slices for its parallel structure where as the serial structure utilizes 1% of CLBs, 9% of BlockRAMs and 0% of DSP48e slices. The resource utilization for the upsampling polyphase filter is almost the same as for downsampling polyphase filter. It uses the same architecture with different data and coefficient pointer sequences. These resource utilizations can also be analyzed in comparison to their processing time in term of clock cycles. Fig. 9 shows the DSE (Design Space Exploration) graph in terms of Area and Time. The smaller the processing Time, the larger is the required Area (parallel design) and the smaller the Area, the larger is the processing Time (serial design). The downsampling polyphase filter with parallel structure requires 99 clock cycles for 40 outputs from 24 data inputs, whereas the serial structure requires 1015 clock cycles for the same outputs. Similarly, the upsampling polyphase filter with parallel structure requires 138 clock cycles for 48 outputs from 40 data inputs, whereas the serial structure requires 1044 clock cycles for the same outputs. All the designs can operate at a maximum clock speed of 225 MHz.

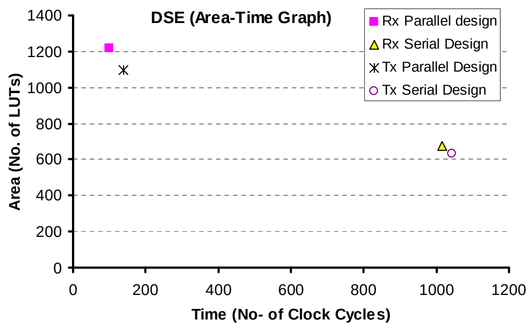
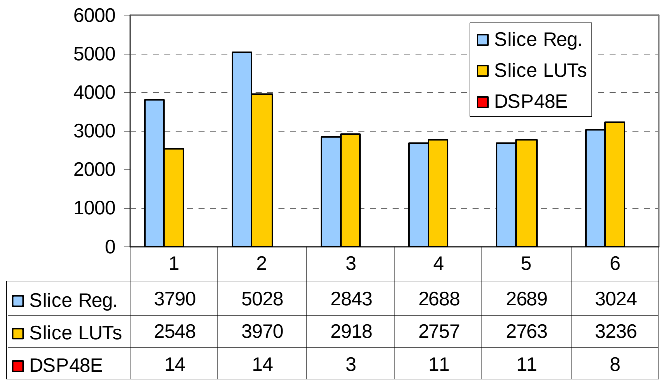


Figure 9: Design Space Exploration (Area-Time) for the downsampling and upsampling polyphase filter for their serial polyphase architectures with parallel and serial MACs.

According to the complexity analysis for 40-point IFFT presented in Fig. 4, Winograd Fourier Transform Algorithm has the lowest computational complexity because of elimination of twiddle factors and complex multipliers. The 40-point Winograd transform is carried out as 5x8 transform (5 and 8 are relative prime numbers). The 40-point vector is loaded to a 5x8 matrix by Chinese Remainder Theorem. The 5-point transform is carried out 8 times and then on intermediate outputs, 8-point transform is carried out 5 times and finally data is unloaded by Ruritanian mapping. So there are two building blocks, i.e., a 5-point transform and a 8-point transform. The 5-point Winograd transform requires 10 multipliers and 34 adders, whereas the 8-point transform requires 4 multipliers and 52 adders. These algorithms can be scheduled in a number of different ways depending on allocated processing resources in terms of adders and multipliers.

For instance, in design no. 1, by allocating 10 multipliers and 8 adders for the 5-point transform, it takes 6 clock cycles to process. Similarly, by allocating 4 multipliers and 16 adders for the 8-point transform, it takes 5 clock cycles to process. So, for the 40-point transform, 6 clock cycles x 8 times and 5 clock cycles x 5 times; total $48 + 25 = 73$ clock cycles are required. The throughput can be increased by operating each clock stage in parallel and by using pipeline registers. It becomes a fully parallel design requiring all multipliers and adders, and plus extra pipeline registers for 5-point and 8-point transforms. By doing so, now 8 times 5-point transform is processed in only 14 cycles and 5 times 8-point transform is processed in 10 cycles. In design no. 2, allocating 4 multipliers and 4 adders for 5-point transform, it takes 11 clock cycles to process. Similarly by allocating 2 multipliers and 4 adders for 8-point transform, it takes 16 clock cycles to process. So, for 40-point transform, 11 clock cycles x 8 times and 16 clock cycles x 5 times; total $88 + 80 = 168$ clock cycles are required. For the pipelined designs which are partial parallel in this case, 8 times 5-point transform is processed in 19 cycles, and 5 times 8-point transform requires 21 cycles. This approach uses the same computational resources as in design no. 1 but due to different scheduling, it results in more pipeline registers as presented in Fig. 10, and hence more clock cycles as shown in the Design Space Exploration presented in Fig. 11.



	1	2	3	4	5	6
🟦 Slice Reg.	3790	5028	2843	2688	2689	3024
🟨 Slice LUTs	2548	3970	2918	2757	2763	3236
🟥 DSP48E	14	14	3	11	11	8

Figure 10: Resource Utilization for 40-point IFFT, as 5x8 Winograd Fourier Transform, by considering different design options based on scheduling and resource allocation.

The sequential approach, design no. 3, scheduling the 5-point transform by allocating 2 multipliers and 4 adders requires 9 clock cycles. Similarly scheduling the 8-point transform by allocating 1 multiplier and 4 adders requires 13 clock cycles. So, for the 40-point transform, 9 clock cycles x 8 times and 13 clock cycles x 5 times; total $72 + 65 = 137$ clock cycles are required. In this design, multipliers are DSP48e based and adders are based on CLBs. The design no. 4 is identical to design no. 3 but adders are also DSP48e based. There is a little reduction in CLB utilization but DSP48e utilization has increased from 3 to 11. DSP48e slice has pipeline registers and it is recommended to use them to increase the performance in terms of clock speed. In design no. 5 (pipelined DSP48e), by using one register on each port-A, port-B and port-C of DSP48e slices, the same resource utilization is achieved as for design no. 4, but it takes longer due to wait stages for data dependencies in the pipeline. In the last design no. 6, the 5-point and 8-point algorithms are rescheduled to avoid the data dependencies on the preceding stages, which are in pipeline. It utilizes a little more CLBs resources and the number of clock cycles are reduced to 217 from 269 in design no. 5. The DSP48e slices are shared to perform both multiplication and addition and therefore, the number of required DSP48e slices is reduced to 8.

Fig. 10 shows the resource utilization for the 40-point IFFT, as 5x8 Winograd transform for different design options mentioned above i.e., 1) Fully parallel (CLB-Adders, DSP48e-Mults) 2) Partial parallel (CLB-Adders, DSP48e-Mults), 3) Sequential Design (CLB-Adders, DSP48e-Mults), 4) Sequential Design, (DSP48e-Adders&Mults), 5) Sequential Design (Pipelined DSP48e-Adders&Mults) 6) Sequential design rescheduled (Pipelined DSP48e-Adders&Mults). The resource utilization is more or less the same for all of the sequential design. All designs use 40 registers (CLB based) to store the intermediate outputs from the 5-point FFTs. The overall resource utilization can be reduced by using distributed memory based registers. Fig. 11 shows the Design Space Exploration in terms of Area-Time plot for the different design options for the 40-point IFFT. The sequential designs which showed almost the same resource utilization now shows differences for their processing clock cycles. All the designs can operate at a maximum clock speed of 400 MHz.

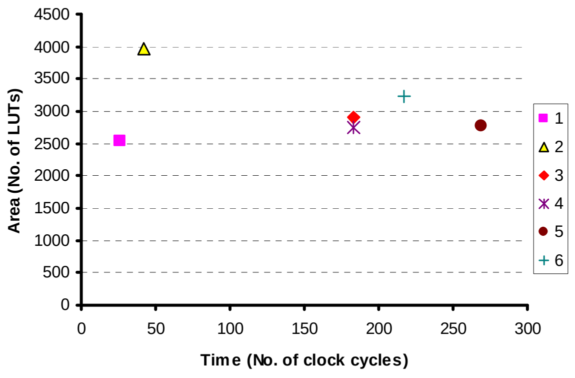


Figure 11: Design Space Exploration (Area-Time) for the different design options for the 40-point IFFT.

2 Conclusions

We have analyzed different implementation structures for polyphase filter bank, and FFT based concepts are analyzed by different algorithms and different design structures based on scheduling and resource allocation. Different design techniques for polyphase register bank mapping, data and coefficient pointers based circular shifting for downsampling and upsampling polyphase filter bank are presented. Based on the results achieved by our FPGA implementation analysis, we have provided a set of guidelines which can be used by the system designers to approach a given solution in terms of Area-Time trade-offs given certain design specifications.

References

[1] fred harris, Chris Dick, ”Polyphase Channelizer Performs Sample Rate Change Required for both Matched Filtering and Channel Frequency Spacing”,in *43rd Asilomar Conference on Signals, Systems and Computers*, 2009.

[2] C.S. S. Burrus, Thomas W. Parks, *DFT/FFT and Convolution Algorithms: Theory and Implementation* John Wiley & Sons, Inc. 1991

[3] *Virtex-5 FPGA XtremeDSP Design Considerations User Guide UG193 (v3.3)* January 12, 2009

[4] *Xilinx DS100 (v1.1) Advance Product Specification*, May 12, 2006

Paper E

Combined Matched Filter and Arbitrary Interpolator for Symbol Timing Synchronization in SDR Receivers

Mehmood Awan and Peter Koch

This paper was published in:
13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems

Abstract

This paper describes a low complexity multi-rate synchronizer that makes use of a polyphase filter bank to simultaneously perform matched-filtering and arbitrary interpolation for symbol timing synchronization in a sampled-data receiver. Arbitrary Interpolation between available sample points is achieved by selecting the appropriate filter in the bank having the polyphase partitioned matched filter, which provides the optimal sampling time. Two different structures are considered which are modified to perform combined arbitrary resampling. The computational complexity is analyzed to have the resource optimal solution. Simulation results are analyzed and their resource utilization for Virtex-5 FPGA implementation is presented.

1 Introduction

Software Defined Radios (SDR) are highly configurable hardware platforms that provide the technology for realizing the rapidly expanding future generation of digital wireless communication infrastructure. Many sophisticated signal processing tasks are performed in a SDR, including advanced compression algorithms, power control, channel estimation, equalization, forward error control and protocol management. In recent years, the FPGA technology has undergone revolutionary changes which has enabled implementation of SDR systems ever further. The gate densities and clock speeds of recent FPGA generations provide the communication system architect with a highly configurable logic fabric that can be used for realizing sophisticated real-time signal processing functions [1]. Amongst the more complex tasks performed in a high data rate wireless system is synchronization. A large amount of time is spent executing this task, and normally significant amount of hardware and software in a SDR is dedicated to synchronization [2]. Physical layer synchronization of symbol timing is required when samples of the received signal are misaligned with the data symbols generated by the transmitter. Synchronization can be done by the use of polyphase filter banks which allow greater flexibility and efficiency in the receiver by computing only those multiplications necessary for matched filtering while simultaneously interpolating to achieve a sample point sufficiently close to the optimum [3].

The development of the polyphase filterbank and its application to perform the interpolations required for symbol timing synchronization is presented in [4]. The authors of [3] extended the work presented in [4] by adding an additional control loop for carrier synchronization after matched filtering. In our work we combine the polyphase structures for performing the arbitrary interpolation and the matched filtering to have a single structure for symbol timing synchronization.

2 Symbol Timing Recovery

There exist two standard DSP approaches to obtain timing recovery in modern QAM receivers [5]. The first approach uses a polyphase interpolator to calculate the samples at the desired locations from the offset samples provided by the free running ADC. These position corrected samples are processed in the receiver matched filter whose output, through a detector, forms a timing error signal to guide the interpolating filter re-sampling process [5]. The second approach folds the interpolation process into a polyphase matched

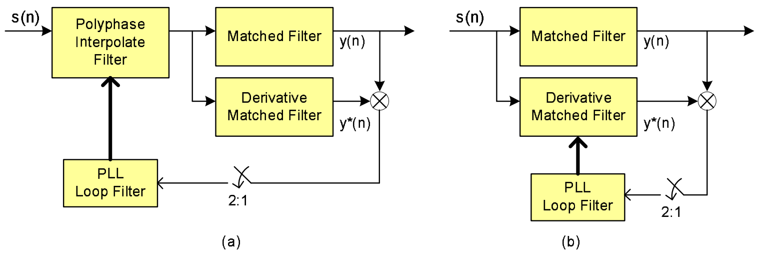


Figure 1: Maximum Likelihood (ML) control of (a) Polyphase Interpolating Filter and of (b) Polyphase Matched Filter [5].

filter. The separate paths of this polyphase filter represent a collection of filters matched to different time offsets between input sample positions and the output sample peak correlation value position. The timing recovery process simply has to determine which filter that matches the unknown time offset between input and output samples. Either process uses a phase locked loop (PLL) to direct the pointer to the appropriate phase leg of the polyphase filter [5].

Figure 1 presents the structure of the two timing recovery schemes based on the Maximum Likelihood (ML) error term formed from two matched filters. Fig. 1a shows the control of the polyphase interpolator while the Fig. 1b shows the control of the matched filters. The loop in Fig. 1a exhibits a larger transport delay through the cascade of two filters than does the loop in Fig. 1b through the delay of a single filter. Due to the additional delay, the polyphase interpolating filter must have a slower (or lower bandwidth) loop filter than does the polyphase matched filter [5].

3 System Design

As a case study we will now investigate an SDR-based design of a BGAN (Broadband Global Area Network) satellite receiver as shown in Figure 2. The ADC data sampled at 57.85 MHz is down-sampled to 723.125 kHz by a cascaded structure of 16:1 and 5:1 re-samplers. It is further required to match the received signal with the transmitted pulse shape using matched filtering along with the timing recovery, thus delivering the output at 151.2 k-symbols/sec. Now the two different approaches for timing recovery, mentioned earlier are employed and compared in order to achieve an optimal solution, in terms of computational complexity.

In the first approach having separate interpolation and match filter, the timing error signal guides the interpolation filter re-sampling operation to achieve the timing synchronization. According to the system requirements, a sample rate of 723.125 kHz is converted to 151.2 kHz. The rate conversion factor is M (number of polyphase partitions or phases) times $723.125/151.2 = 4.782$, which therefore requires an arbitrary interpolator. The structural diagram for the first approach i.e., separate polyphase interpolating filter as

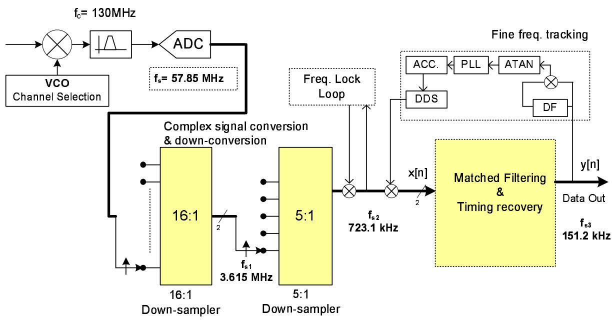


Figure 2: SDR design for a BGAN satellite receiver. Data is sampled at 57.85 MHz and down-sampled to 723.125 kHz by a cascaded structure of 16:1 and 5:1 re-samplers followed by arbitrary interpolator and match filter for symbol timing recovery.

shown in Figure 1a, replaced with arbitrary interpolator is illustrated in Figure 3.

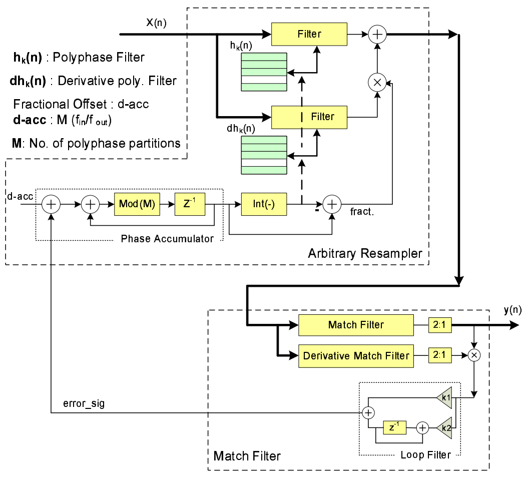


Figure 3: Cascaded structure of arbitrary re-sampler and matched filter with ML control of polyphase interpolating filter.

Figure 3 shows a cascaded structure of arbitrary re-sampler [6] and matched filter. Arbitrary re-sampler is based on linear interpolation to a position between available output points in an M-path interpolator. Interpolation is achieved by polyphase lowpass filter, and the polyphase derivative filter is used for local slope correction. The polyphase lowpass and the polyphase derivative filters' coefficient pointer is addressed by the integer part of the phase accumulator. The phase accumulator is a modulo-M (number of polyphase partitions or phases) addition of the fractional offset factor d_acc and the error signal $error\_sig$ generated by the loop filter. The fractional part of the phase accumulator is multiplied with the derivative filter's output and added to the lowpass filter's output for the slope correction.

In this case, the matched filter operates at twice the symbol rate i.e., 2$\times$151.2 kHz = 302.4 kHz, but the loop filter still operates at 1 sample per symbol. So the required fractional offset factor d_acc for the arbitrary re-sampler comes out to be M times 723.125/(2$\times$151.2) = M$\times$2.39. The polyphase interpolator filter is designed at upsampled frequency 16$\times$723.125 kHz (having 16 phases or subfilters) with transition band of 100-200 kHz and 60 dB side lobe attenuation. The resulting 320 coefficients are partitioned into M=16 polyphase sub-filters (phases) each having 20 coefficients. The fractional offset factor d_acc then becomes 38.26. The matched filter is designed with a roll-off factor of 0.25, filter length of $\pm$ 4 symbols, and operating at 2 samples per symbol, which results in 33 coefficients.

A computational complexity analysis is required and in this work we define the complexity in terms of multiplications per second, similar to the definition applied in [7]:

$$R_m = N \times f_s \tag{1}$$

R_m is multiplications/sec, N is the number of non-zero coefficients, and f_s is the input sampling frequency.

In multirate filters, R_m can be reduced by a sampling rate conversion factor. For polyphase decimator:

$$\begin{aligned} Rm_{DEC} &= (N \times f_{si})/M \\ Rm_{DEC} &= N \times f_{so} \end{aligned} \tag{2}$$

where $f_{si}/M = f_{so}$, f_{si} is sample rate at input, and f_{so} is sample rate at output of a polyphase filter.

For polyphase interpolator:

$$\begin{aligned} Rm_{INT} &= (N \times f_{so})/L \\ Rm_{INT} &= N \times f_{si} \end{aligned} \tag{3}$$

(where $f_{so}/L = f_{si}$)

The computational complexity for the arbitrary resampler and the matched filter turns out to be 2$\times$(96.77)x10^6 and 2$\times$(9.98)x10^6 multiplications per second (M-Mult/sec), respectively. The factor 2 accounts for the complex operations.

In the second approach, the interpolation process is folded into the polyphase matched filter which is realized by filter bank index selection, and therefore the separate interpolating filter following the matched filter is not required. Different loop control structures for this kind of design are possible. An M-stage polyphase filter bank with input data sampled at approximately N samples/symbol can be used in a loop that operates at (a) MN samples/symbol, (b) N samples/symbol, or (c) 1 sample/symbol [4]. It has been investigated from simulations that these loop architectures work only if the input data rate is some integer multiple of the desired output data rate, but not for the case involving arbitrary re-sampling operation. So it requires some modifications to embed the arbitrary re-sampling process.

In order to have the arbitrary re-sampling process embedded in the combined interpolation and match filtering operation, the cascaded approach presented in Figure 3 is modified by the following steps and the modified design is presented in Figure 4.

1. Replace the arbitrary polyphase interpolator low-pass (and polyphase derivative) filters with polyphase matched (and polyphase derivative matched) filters.

2. Discard the previous matched and derivative matched filters.

3. Reconnect the timing error and loop blocks to the polyphase matched and the polyphase derivative matched filters as they were connected before to the matched and derivative matched filters.

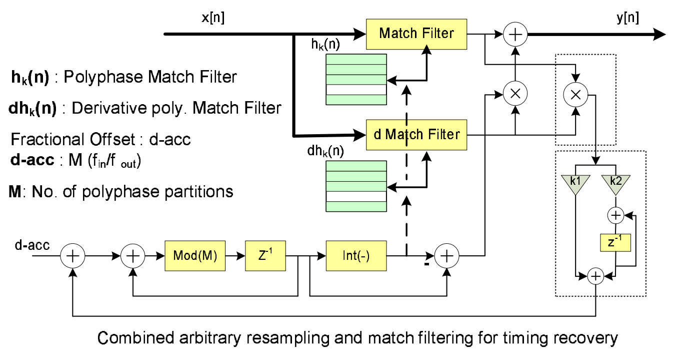

x[n]
Match Filter
y[n]
$h_k(n)$
$h_k(n)$: Polyphase Match Filter
$dh_k(n)$: Derivative poly. Match Filter
Fractional Offset : d-acc
d-acc : M (f_{in}/f_{out})
M: No. of polyphase partitions
d Match Filter
$dh_k(n)$
k1
k2
z^{-1}
d-acc
Mod(M)
Z^{-1}
Int(-)
Combined arbitrary resampling and match filtering for timing recovery

Figure 4: Combined structure of arbitrary re-sampler and matched filter with ML control of polyphase matched filter.

To our knowledge, this modified design has not been published previously, and therefore we consider the concept as a new and innovative idea. The matched filter is designed with the following specifications; roll-off factor of 0.25 with filter length of ±4 symbols, and the loop operates at 1 sample per symbol, which results in 624 coefficients. Upsample L and downsample M factors are 16 and 76.52, respectively. The corresponding computational complexity becomes $2\times(94.35)$ M-Mult/sec. Based on the complexity analysis for the two designs (interpolating filter design and the modified polyphase matched filter design) as shown in Table 1, it can be concluded that our modified polyphase matched filter design leads to a reduced computational complexity by more than 10%. Besides that in the first design, loop exhibits a larger transport delay through the cascade of two filters than does the modified design's loop through the delay of a single filter.

Separate Arbitrary interpolation and Matched filtering	(Arbitrary Resampler) 2×96.77 (Matched Filter) 2×9.98
Combined Arbitrary interpolation and Matched filtering	2×94.35

Table 1: Computational Complexity (M-Mult/sec)

4 Simulations

Closed-loop simulations are performed for the modified polyphase matched filter design in order to demonstrate the features of polyphase filterbank for performing combined arbitrary interpolation and matched filtering for timing synchronization. It uses ML timing error detector which can be easily incorporated into the polyphase filter bank [4]. Simulated input test signal for the BGAN receiver is a 4-QAM signal having one desired channel with no noise and interference. The matched filter is designed at an up-sampled frequency of 32$\times$723.125 kHz (having 32 phases or subfilters) with a roll-off factor of 0.25 and filter length of $\pm$4 symbols. The resulting 1248 coefficients are partitioned into M=32 polyphase sub-filters (phases) each having 39 coefficients. The fractional offset factor d_acc becomes 153.042. The proportional K_p and integral K_i gain values are set to 1.6617 and 0.0033 respectively. Data samples at 1 sample/symbol are processed by the polyphase matched filter and the polyphase derivative matched filter filterbanks. The product of the two filterbank outputs form the timing error which is updated once per symbol. The timing error signal is filtered by a proportional-plus-integrator loop filter which is required for a second-order loop to track out the symbol clock frequency offset [8]. The loop filter output together with fractional offset is used to control the increment in the modulo-32 counter (NCO) which becomes constant when the loop has achieved lock.

Figure 5 shows the simulation results as eye and constellation diagrams, both plotted for 30 to 4000 and 3000 to 4000 symbols. It is seen that the loop filter converges the signal to the desired constellation. Figure 6 demonstrates the loop filter response and the incrementation of the timing accumulator. The loop filter's response converges and the NCO control (timing accumulator) settles to a constant steady-state value (153 in this case).

5 FPGA Implementation

The design is implemented on a Virtex5-vsx50tff1136-3 FPGA. Block RAMs are used for storing filter coefficients and DPRAM (Dual Port RAM) is used for loading of the input data samples as a delayed line required for filtering operation. The device utilization summary is tabulated in Table 2. The design can operate at a maximum frequency of 380.625 MHz.

Number of Slice Registers	520 out of 32640 (1%)
Number of Slice LUTs	533 out of 32640 (1%)
Number of Block RAM/FIFO	3 out of 132 (2%)
Number of DSP48Es	6 out of 288 (2%)

Table 2: Device Utilization (Virtex5-vsx50tff1136-3)

6 Conclusions

In this paper a combined matched filter and arbitrary interpolator for symbol timing synchronization in SDR receivers was presented. Two different structures were considered

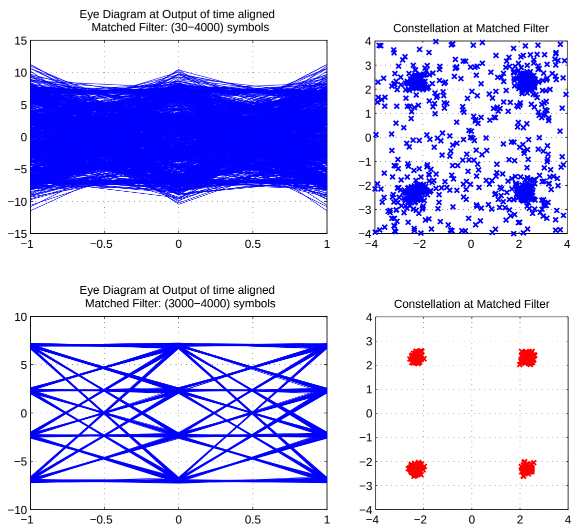


Figure 5: Eye and Constellation diagram for 30 to 4000 and 3000 to 4000 symbols. The plots for 30 to 4000 symbols give a messy picture due to non-aligned symbols before the synchronization has achieved.

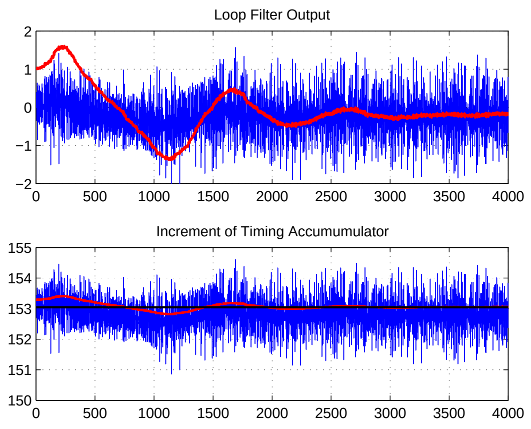


Figure 6: Loop filter output and timing increment of Accumulator

for this task, which were next applied for SDR implementation of a BGAN satellite receiver. A structure with ML control of polyphase matched filter was modified to embed arbitrary interpolation, which in terms of computational complexity is more efficient than the cascaded structure of arbitrary interpolator and matched filter. We find a 10% reduction. Simulations for the modified structure with the loop operating at 1 sample/symbol were presented to illustrate the response of loop filter and timing accumulator. The loop filter's convergence and constant timing accumulator reliably indicate the achievement of symbol timing synchronization. The structure was finally implemented on a Virtex-5 FPGA having device utilization of 1% and maximum operating clock speed of 380 MHz.

Acknowledgment

The research described in this paper is carried out in Center for Software Defined Radio at Aalborg University together with the industrial partner Thrane & Thrane A/S, Denmark. A special thanks goes to Jan Harding Thomsen and Bo Dyssegard from Thrane & Thrane A/S for their valuable discussions and suggestions throughout the system design.

References

[1] Chris Dick, fred harris, Michael Rice,"Synchronization in Software Radios - Carrier and Timing Recovery Using FPGAs",in *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2000.

[2] H. Meyr, M. Moeneclaey and S. A. Fechtal, "Digital Communication Receivers", John Wiley & Sons Inc., New York, 1998.

[3] Joseph Gaeddert et al.,"Multi-rate Synchronization of Digital Receivers in Software-Defined Radios" in *Proceeding of the SDR 07 Technical Conference and Product Exposition*, 2007.

[4] f. harris and M. Rice, "Multirate Digital Filters for Symbol Timing Synchronization in Software Defined Radios", *IEEE Journal on Selected Areas of Communications*, vol. 19, no. 12, December 2001.

[5] Chris Dick, Benjamin Egg, fred harris,"Architecture and Simulation of Timing Synchronization Circuits for the FPGA Implementation of Narrowband Waveforms" in *Proceeding of the SDR 06 Technical Conference and Product Exposition*,2006.

[6] fredric j. Harris, *Multirate Signal Processing for Communication Systems*, Prentice Hall, 2006.

[7] Ljiljana Milic, *Multirate Filtering for Digital Signal Processing: MatLab Applications*, Information Science Reference (December 26, 2008) ISBN-13: 978-1605661780.

[8] F. M. Gardner, "Phaselock Techniques", New York: Wiley, 1979.

Paper F

Polyphase Channelizer as Bandpass Filters in Multi-Standard Software Defined Radios

Mehmood Awan and Peter Koch

This paper was published in:
2nd International Workshop on Cognitive Radio and Advanced Spectrum Management (CogART 2009)

Abstract

The aim of this work is to design efficient bandpass filters for multistandard software defined radios. Software Defined Radio (SDR) based applications which demand high sampling rate to eliminate the most of the analog components require high-performance technology and digital signal processing methods to handle and process the high sample rate data. State of the art technology such as FPGAs can support several hundred MHz of I/O data transfer rate. The internal hardware architecture, however, would limit the maximum operating frequency of the design. An alternative is therefore to use advanced DSP methods to overcome this bottleneck. Polyphase channelizers having spectral shifter to move the filter position are used in a scenario of dual standard (WLAN and UMTS) SDR receiver for the bandpass filtering. It will not only split the data in multiple paths to reduce the per arm data rate but the filter also operates at lower rate than the input sample frequency. Furthermore it can also eliminate the need of IQ demodulator.

1 Introduction

A Software-Defined Radio (SDR) system is a wireless communication system which ideally can tune to any frequency band and receive any modulation scheme by means of various functionalities implemented in software and/or programmable hardware. SDR is an enabling technology for future radio transceivers, allowing the realisation of multi-mode, multi-band, and reconfigurable base stations and terminals. However, considerable research efforts and breakthroughs in technology are required before the ideal software radio can be realised. An ideal software radio (ISR) samples the signal at Radio Frequency (RF), just after the antenna, whereas the realizable version of the software radio is the one where the analog to digital conversion takes place after the first intermediate frequency (IF).

Recent developments and increasing trends toward a single device integrating several features and capabilities encourage the companies and research centers to develop portable multi-standard multi-mode ”all-in-one” front-ends. High level of integration and small size are precedence objectives in these types of mobile applications. In order to achieve those objectives it is feasible to move most of the data processing to the digital domain through shifting the analog to digital converter (ADC) as close to the antenna as possible [1]. This imposes more stringent performance requirements on the analog-to-digital (A/D) conversion, where a high dynamic range must be combined with a high sampling rate [2].

A scenario of a general multi-standard is shown in Fig. 1. This scenario in our work has been scaled down to the UMTS and WLAN standards [1] which actually fits to the mobile application. The RF spectral location for UMTS and WLAN standards are shown in Fig. 2. UMTS has a bandwidth of 60MHz for downlink with 12 channels and WLAN has 84.5MHz of bandwidth with 3 non-overlapped channels. It is required to down-sample and down-convert these channels to baseband.

Bandpass sampling and direct conversion are the two receiver architectures that are suitable for the software radios [6]. The sampling of bandpass signals can be carried out at

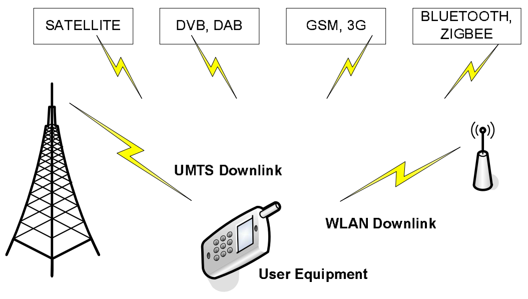


Figure 1: A scenario of multi-standard ”all-in-one” front-ends user equipment. It highlights the user equipment capable of receiving two standards i.e.UMTS and WLAN.

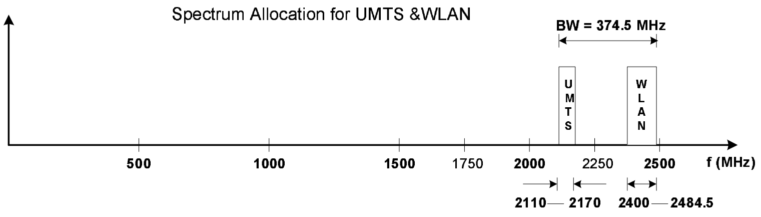


Figure 2: Spectrum allocation for UMTS and WLAN standards. UMTS has a bandwidth of 60MHz for downlink having 12 channels and WLAN has a 84.5MHz of bandwidth having 3 non-overlapped channels.

rates significantly lower than the conventional lowpass Nyquist sampling, causing intentional aliasing of the signal. Bandpass sampling can allow received signals to be digitized closer to the antenna using manageable sampling rates and hence could be favourable for down-conversion in the software radios.

2 System Design

The combined band of WLAN and UMTS is under-sampled at 630MHz which results in overlapped aliases, but the individual bands of UMTS and WLAN are non-overlapped. WLAN band aliases to 36-120MHz and UMTS aliases to 220-280MHz in the Nyquist zone. WLAN band is spectrally inverted. Bandpass sampling aliases are shown in Fig 3. In order to extract the individual channels of UMTS and WLAN standards to baseband with the desired output sample rate, channelizers are required. There are different types of channelizers being per-channel approaches, pipeline frequency transforms and polyphase channelizers [4]. Polyphase channelizers are the most efficient interms of computations and required hardware resources as compared to the other channelizers [4]. Based on the unique features of the polyphase channelizer, we have chosen this concept to implement the system design. The relation between the sampling frequency, channel spacing and number of channels for the polyphase channelizer is [7]:

$$f_s = N \cdot \Delta f \qquad (1)$$

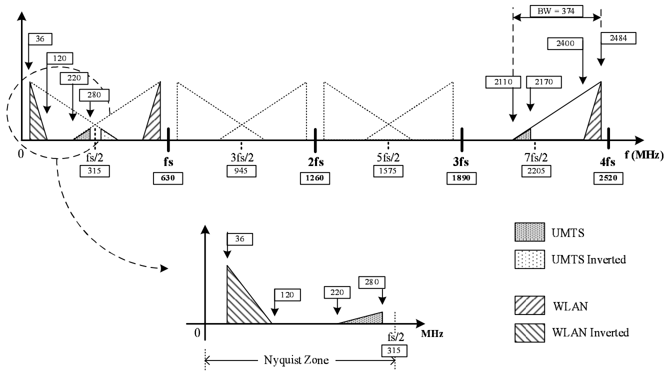


Figure 3: The combined band of UMTS and WLAN is undersampled at 630MHz, and the aliased signals are shown in the Nyquist zones. The aliases are overlapped but individual UMTS and WLAN bands are non-overlapped. WLAN alias is spectrally inverted in the Nyquist zone.

where f_s is the input sampling frequency, N is the number of channels (i.e., transform size) and Δf is the inter-channel spacing. There are two constraints that have to be met; one is that N should be an integer, and the second is that the channels to be down-sampled and down-converted to baseband should be centered onto the multiples of the channel spacing.

The block diagram of the system design [5] having a sampling frequency of 630MHz is shown in Fig. 4. The selection of the sampling frequency is an iterative process that requires the conditions of non-overlap aliases and polyphase channelizer constraints to be fulfilled. The block diagram shows a dual-standard system, sampling at 630MHz and using bandpass filters to separate the two standards before channelization. In order to reduce the computational workload of the polyphase filters, the incoming complex signals from the bandpass filters are down-sampled by possible large factors such that the sampling frequency of the re-sampled signals is above the signal's bandwidth.

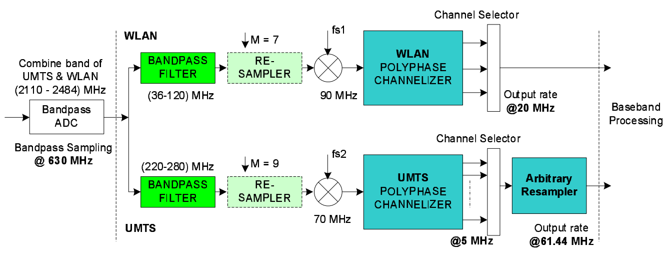


Figure 4: System block diagram having spectrum translation prior to UMTS and WLAN channelizers. UMTS channelizer is followed by Arbitrary Resampler to achieve the target rate of 61.44MHz [5].

In an SDR receiver chain, after the ADC we need to have IQ demodulators to have quadrature signals needed to perform the filtering task on complex data. The well known methods for IQ demodulations are quadrature mixer which is known as Digital Down Converter (DDC), complex filtering and Hilbert transform methods [10]. Irrespective of their performance characteristics, in all of these cases multipliers operate and deliver the output at the same rate as the input. SDR based applications demand high sampling rates to eliminate most of the analog components. It will require not only the technology advancements but especially the advanced signal processing methods to handle and efficiently process the high sample rate data with the current technology.

This paper will focus on the bandpass filters used in a multi-standard system design [5]. High sampling rate at the input leads to the problems of complicated ADC-FPGA interfaces, and the design of bandpass filters which should operate at high rate. These problems can be solved by splitting the data stream in multiple paths and feeding the data to the polyphase channelizer. The split data drives the commutator (a rotatary switch distributing the data among the various arm (sub-filter) in the polyphase filter) of the polyphase channelizer [9] and reduces the per-arm data rate. The next task is to design the polyphase prototype filter to extract the band of channels (standards) rather than single channels. It will be helpful in two folds. First, we do not need to have an IQ demodulator at the input which is the high frequency end. Second, the bandpass filter operates at a low frequency due to the down-sampling at the commutator which further can be configured to deliver the required output sample rate by using embedded resampling (an efficient technique to resample the data in the polyphase arms rather than using separate P/Q resamplers).

A channelizer described in [8] gives a proof-of-concept implementation, operating at a 1.6GHz input signal. The input signal is split in parallel paths, lowering the per-arm frequency and then fed to the polyphase channelizer. The high speed demands of 1.6GHz channelizer are efficiently satisfied using the Xilinx Virtex-4 FPGA by using the DSP48 blocks capable of operating at 500MHz [11]. The input signal is quickly fanned-out to 16 synchronized and parallel paths, effectively reducing the individual path rates. In this case the polyphase channelizer is efficiently used at the high input sampling rate, extracting the individual channels. We will extend its use to act as bandpass filter having the same characteristics in terms of the low operating frequency and split data paths. Referring to the multi-standard system design whose block diagram is shown in Fig. 4. The system uses the undersampling technique and aliases down the combined spectrum of UMTS and WLAN, resulting in overlapped aliases but the individual bands of UMTS and WLAN are still non-overlapped. The channel bandwidth of both standards is different thus requiring separate channelizers. Therefore it requires bandpass filters to separate the standards before the channelizers as shown in Fig. 4.

For a polyphase channelizer to act as a bandpass filter, we follow the same procedure, i.e., split the frequency spectrum in equal spectral bins and try to choose a factor N that meets the $fs = N \cdot \Delta f$ requirements in terms of the channel spectral allocation and non-overlapped channels. Furthermore, the polyphase channelizer can be used in its variant mode where the channels' bin can be shifted by multiples of quarter of the channel spacing [9] in order to have a better channel filtering operation. The filter requirements for UMTS and WLAN standards are shown in Fig. 5.

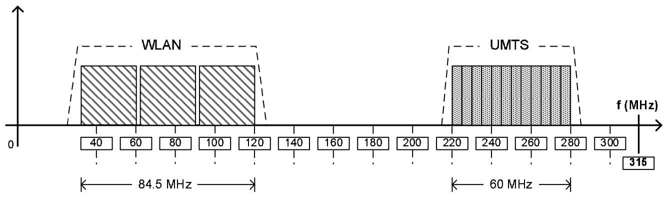


Figure 5: Bands of WLAN and UMTS having 3 and 12 channels respectively. WLAN and UMTS bands are (36-120)MHz and (220-260)MHz respectively.

Designing the prototype filter for the polyphase channelizer to act as a bandpass filter is an iterative task which needs to satisfy both the standard's channel widths. Case A: In the standard polyphase channelizer the prototype baseband filter can not comply to the requirements because no value of N delivers the appropriate spectral bins for UMTS and WLAN channel bandwidths. Case B: The variant of the polyphase channelizer [9] which accommodates the spectral shift of multiples of quarter of the channel spacing, may result in a useful solution. The variant polyphase channelizer can also accommodate the spectral shifts other than the multiples of quarter of the channel spacing, but it results in complex multiplier operations in the sub-filters and therefore demands more resources. The prototype filter designed for WLAN can also cover the specifications of UMTS because the bandwidth of 84MHz can accommodate the bandwidth of 60MHz. Now, aiming at meeting the constraint of Eq. 1, Δf is chosen as 90MHz which will cover both 84MHz and 60MHz bandwidths and satisfy the N which turns out to be 7. The filter spectral locations are shown in Fig. 6

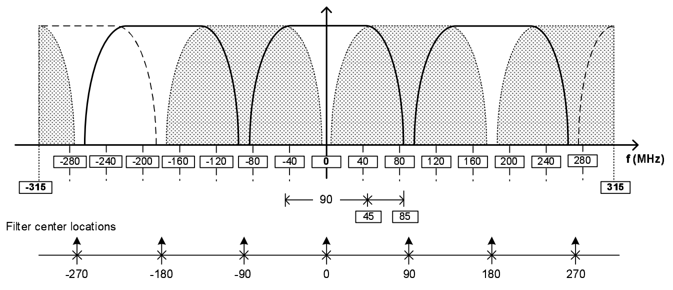


Figure 6: Prototype baseband filter for the polyphase channelizer illustrating its spectral location in other channels' bin.

The prototype filter is designed with the 'firls' method (in MatLab) having the transition widths as [0 45 85 315] as shown in Fig. 6. The passband ripples and stopband attenuation are 0.1 dB and 60 dB respectively. It results in a 42 taps prototype filter which are partitioned into 7 sub-filters (6 taps per sub-filter). The filter is designed for the larger transition band to have overlapped filter response to minimize the filter coefficients. The spectral position of the filter bins are shown in Fig. 7.

It can be seen by comparing the spectral location of the filter in Fig. 6 or Fig. 7 with

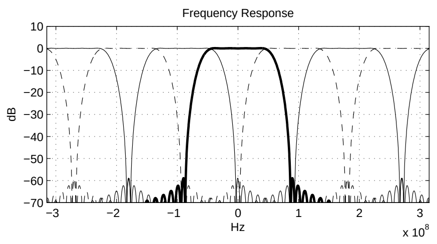


Figure 7: Spectral locations of prototype filter in different channels' bin (N=7)

the required channels in Fig. 5 that the filter's spectral location does not cover the channels bandwidth. Next, we therefore experiment the variant polyphase channelizer having the spectral shifts of quarter of the channel spacing in order to achieve a working solution. The shifted versions in terms of multiples of quarter of the channel spacing (90MHz) are shown in Fig. 8, 9, 10. The first case (case 0) where the shift is zero (s=0) is the same as shown in Fig. 7. In these figures, different cases for the filter spectral position are illustrated having the spectral shift of multiples of quarter of the channel spacing. Now, again by comparing the spectral location of the filters in Fig. 7 to Fig. 10 with the required channels in Fig. 5, it is seen that the filter's spectral location in Fig. 10 provides a solution satisfying the requirements for both of the standards. It will be more clearly shown in the simulation section.

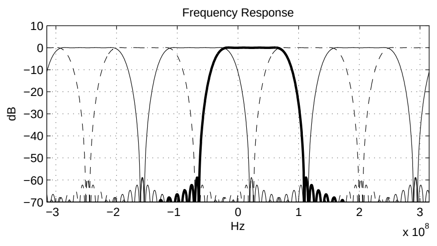


Figure 8: Case 1: Spectral locations of prototype filter in different channels' bin (N=7) with a spectral shift s=1 ('s' is the multiple of the quarter of the channel spacing).

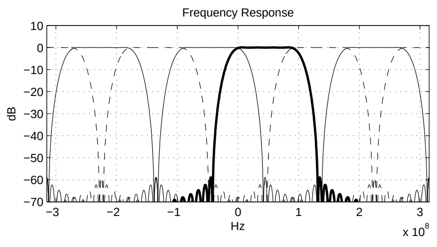


Figure 9: Case 2: Spectral locations of prototype filter in different channels' bin (N=7) with a spectral shift s=2 ('s' is the multiple of the quarter of the channel spacing).

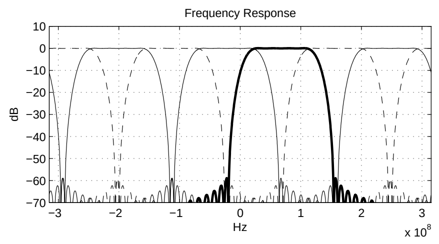


Figure 10: Case 3: Spectral locations of prototype filter in different channels' bin (N=7) with a spectral shift s=3 ('s' is the multiple of the quarter of the channel spacing).

If even by these two solutions (case A & B), the filter requirements of both standards are not met, then variant polyphase channelizer having spectral shifts other than multiples of the quarter of the channel spacing can be employed. As explain earlier, that it will increase to the required resource utilization but the design will have characteristics of low operating frequency and multiple path input data which is the most important while working with the high frequency input data.

3 Simulations

In order to simulate the working of the design, we have generated a composite test signal at the spectral locations of WLAN and UMTS standards which is shown in Fig. 11.

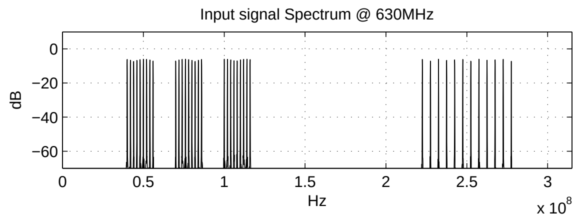


Figure 11: A composite test signal generated at the spectral locations of WLAN and UMTS standards. WLAN has 3 non-overlapped channels whereas UMTS has 12 channels.

This test signal is applied to the variant polyphase channelizer having $s = 3$ as shown above, which delivers the required UMTS and WLAN bands at the output. The input signal's spectrum overlaid with the filter's spectral positions for the variant polyphase channelizer with $s = 3$ which gives a solution in this case as shown in Fig. 12. It can be seen that UMTS and WLAN bands are well occupied by the filter's passband only in channel 1 and 3.

All 7 channel outputs of the channelizers are shown in Fig. 13. It can be seen that only outputs from channel 1 and 3 delivers the required outputs as bandpass filters for UMTS and WLAN standards. The remaining outputs are garbage being mixtures of both the

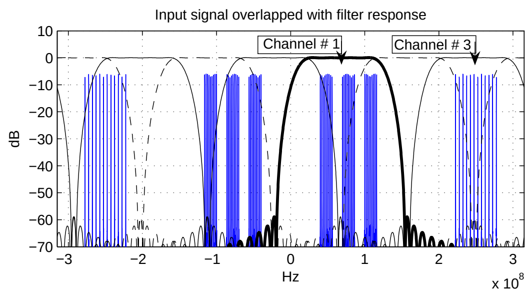


Figure 12: The input signal spectrum is overlaid with the filter's spectral positions for variant polyphase channelizer with $s = 3$.

standards. The outputs have sample rate of 90MHz which is due to down-sampling by 7 in the commutator. Furthermore the outputs from the variant polyphase channelizer are

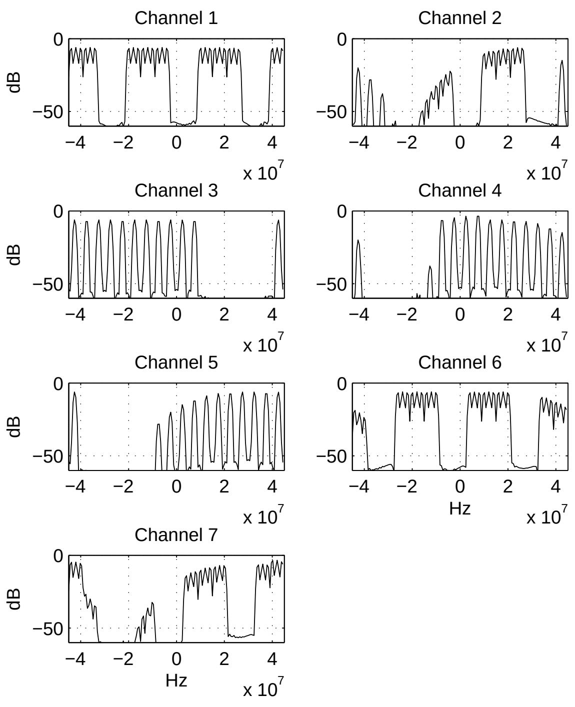


Figure 13: All 7 Channel outputs of the channelizers. It can be seen that only outputs from the channel 1 and 3 deliver the required outputs as bandpass filters for the UMTS and WLAN standards.

complex which therefore eliminate the need of IQ demodulators in this case. WLAN and UMTS bandpass signals are spectrally shifted by Π and $\Pi/2$ rotators respectively to have a clear view of bandpass outputs as shown in Fig. 14. These shifting will not require any real multiplication operation as Π and $\Pi/2$ rotators result only in sign change operation. So we have efficiently bandpass filtered UMTS and WLAN standards at input sampling rate of 630MHz and delivering the outputs at 90MHz.

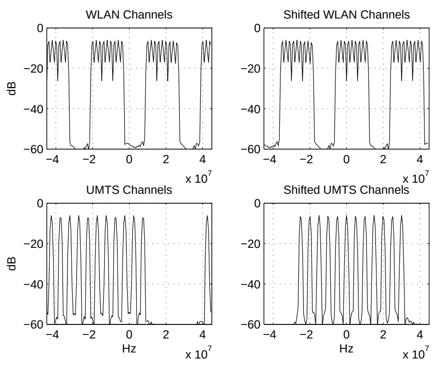


Figure 14: WLAN and UMTS bandpass signals are spectrally shifted by Π and $\Pi/2$ rotators to have a more clear view of the bandpass signals. These shifting will not require any multiplication operation.

4 Conclusion

A dual-standard (UMTS & WLAN) software radio receiver architecture is presented with the focus on high performance bandpass filters. It is required to have bandpass filters to separate out the standards before channelization such that they can handle the high input data stream and can operate on the current technology platforms as well. It is seen that the polyphase channelizer or its variant can be used with their unique and extraordinary features of multiple paths and spectral translation, to act as a high performance bandpass filter which can also eliminate the need for IQ demodulators. The final system block diagram is shown in Fig. 15 where the previous bandpass filters for UMTS and WLAN standards in Fig. 4 are removed by a single polyphase channelizer, making the architecture more compact and resource efficient. The initial estimates for resource utilization result in approx. 32% of reduction compared to the solution with separate bandpass filters implemented in polyphase fashion as well.

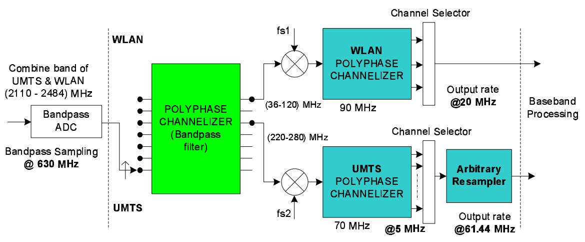


Figure 15: Final system block diagram where the previous bandpass filters for UMTS and WLAN standards are removed by a single polyphase channelizer.

References

[1] Mehmood-ur-Rehman Awan, Muhammad Mahtab Alam, ”Design & Implementation of FPGA-based Multi-standard Software Radio Receiver”, in *Proc. IEEE Norchip Conference on microelectronic*, Nov. 2007

[2] B. Razavi, ”Challenges and trends in RF design”, in *Proc. IEEE ASIC Conf. and Exhibit*, 1996, pp. 81-86

[3] Stream radio goes digital. http://www.csdr.dk.

[4] Mehmood-ur-Rehman Awan, Muhammad Mahtab Alam, ”Design & Implementation of FPGA-based Multi-standard Software Radio Receiver”, Master of Engineering Thesis, Aalborg University Denmark, June 2007

[5] Mehmood-ur-Rehman Awan, Muhammad Mahtab Alam, ”Area efficient implementation of polyphase channelizer for Multi-standard Software Radio”, 5th Karlsruhe Workshop on Software Radios, 2008.

[6] Patel, M. and Lane, P., ”Comparison of downconversion techniques for software radio”, Department of Electronics and Electrical Engineering, University College London. www.ee.ucl.ac.uk/lcs/papers2000/lcs050.pdf.

[7] Fredric J. Harris, Chris Dick and Micheal Rice, ”Digital receivers and Transmitters using Polyphase Filter Banks for Wireless Communications”, in *IEEE Transaction on microwave theory and techniques*, Vol. 51, No. 4 April 2003

[8] Benjamin Egg, Fredric J. Harris, Chris Dick, ”Ultra-Wideband 1.6GHz Channelizer: Versatile FPGA Implementation”, in *Proceedings of the SDR 05 Technical Conference and Product Exposition*, SDR Forum, 2005

[9] fredric j. harris, *Multirate Signal Processing for Communication Systems*, Prentice Hall, 2006

[10] Richard G. Lyons, *Understanding Digital Signal Processing*, Prentice Hall, Second Edition, 2004 ISBN-10: 0-13-108989-7

[11] http://www.xilinx.com