"SMT-8036 based implementation of secured adaptive Software Defined Radio system for LDPC coded modulation techniques"

Thesis submitted towards the partial fulfillment of requirement for the award of degree of

Master of Engineering
In
Electronics and Communication

Submitted by
Arvinder Pal Singh Kalsi
Roll no: 801061006
(ECED)

Under the guidance of

Dr. Rajesh Khanna
Professor & Head
ECED

Mr. Sanjay Kumar
Assistant Professor
ECED

ELECTRONICS & COMMUNICATION ENGINEERING DEPARTMENT
THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

PATIALA-147004 (PUNJAB)

CERTIFICATE

I, hereby certify that the work which is being presented in this thesis entitled **"SMT-8036 based implementation of secured adaptive Software Defined Radio system for LDPC coded modulation techniques."** in partial fulfillment of requirements for the award of degree of the Master of Engineering in Electronics and Communication from Thapar University, Patiala, is an authentic record of my own work carried under the supervision of **Mr. Sanjay Kumar,** Assistant Professor, ECED **& Dr. Rajesh Khanna**, Professor & Head, ECED.

The matter presented in this thesis has not been submitted in any other University/Institute for the award of Master of Engineering.

Arvinder Pal Singh

(Arvinder Pal Singh Kalsi)

Signature of the student

Dated: 2/7/2012

This is certified that the above statement made by the candidate is correct to the best of my knowledge.

(Dr. Rajesh Khanna)

Professor & Head, ECED

Date: 2/7/2012

(Mr. Sanjay Kumar)

Assistant Professor, ECED

Date: 2/7/2012

(Dr. S.K. Mohapatra)

Dean, Academic Affairs

Thapar University, Patiala

Date:

ACKNOWLEDGEMENT

I would like to give special thanks to my guides, **Dr. Rajesh Khanna,** Professor & Head, ECED and **Mr. Sanjay Kumar,** Assistant Professor, **ECED**, Thapar University, Patiala, for their advice, kind assistance, and invaluable guidance. It has been my pleasure to work under their guidance.

I would like to express my deepest gratitude to **Dr. Rajesh Khanna**, Professor & Head, Electronics and Communication Engineering Department, Thapar University, Patiala, for his advice, motivation, guidance, moral support, efforts and the attitude with which he solved all of my queries in making this report possible. It has been a great honor to work under his supervision.

I am thankful to him and Electronics and communication Engineering Department, for providing us with adequate infrastructure for carrying out our work.

I also thank **Dr. Kulbir Singh**, P.G. Coordinator, Electronics and communication Engineering Department for the motivation and inspiration that triggered me for the report work.

I would like to thank all the faculty members of ECED for their intellectual support and a special thanks to my family and my friends who constantly encouraged me to carry out this work.

At last but not the least, I am thankful to the authors, whose work I have consulted and quoted in my work.

Arvinder Pal Singh

Arvinder Pal Singh Kalsi

Roll no. 801061006

ABSTRACT

With the increasing computing power of modern microprocessors it becomes feasible to process radio signals completely in software reducing the complexity of the hardware. Using embedded microprocessors in an FPGA and combining them with other advanced features of the fabric allows for a powerful solution on a single, reprogrammable chip [3].

Software-Defined Radio (SDR) is a rapidly evolving technology that is receiving enormous recognition and generating widespread interest in the telecommunication industry. Over the last few years, analog radio systems are being replaced by digital radio systems for various radio applications in military, civilian and commercial spaces. In addition to this, programmable hardware modules are increasingly being used in digital radio systems at different functional levels. SDR technology aims to take advantage of these programmable hardware modules to build an open-architecture based radio system software.

SDR technology facilitates implementation of some of the functional modules in a radio system such as modulation/demodulation, signal generation, coding and link-layer protocols in software. This helps in building reconfigurable software radio systems where dynamic selection of parameters for each of the above-mentioned functional modules is possible. A complete hardware based radio system has limited utility since parameters for each of the functional modules are fixed. A radio system built using SDR technology extends the utility of the system for a wide range of applications that use different link-layer protocols and modulation/demodulation techniques.

Commercial wireless communication industry is currently facing problems due to constant evolution of link-layer protocol standards (2.5G, 3G, and 4G), existence of incompatible wireless network technologies in different countries inhibiting deployment of global roaming facilities and problems in rolling-out new services/features due to wide-spread presence of legacy subscriber handsets. SDR technology promises to solve 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 system can take up different personalities depending on the software module being used. Also, 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 in dealing with problems due to differing standards and issues related to deployment of new services/features.

Summarizing, SDR is a promising technology that facilitates development of multi-band, multi-service, multi-standard, multi-feature consumer handsets and future-proof network infrastructure equipment [1].

Wireless communications systems and standards have been developed around the world without any global plan. Recently hardware technology evolved significantly. Some of the key milestones in this progress are transition from analog hardware to digital hardware and then introduction of sophisticated processors.

This is followed by the development of Software Defined Radio (SDR) structures and virtual hardware that are under development currently. Software Defined Radio (SDR) system is a useful and adaptable future-proof solution to cover both existing and emerging standards, it provides elements with reconfigurability, intelligence and software programmable hardware. Moreover, it has capability of providing global seamless connectivity and solving the interoperability issue.

Here, we have practically implemented a receiver system, in which receiver can switch (or adapt) its demodulator circuit in accordance with the Bit Error Rate of the signal received. For this, I am using DSP processor to take decision that which modulated signal is received and FPGA processor for various demodulator implementations so that later on it can select the receiver dynamically according to the BER of the received signal.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

With the shooting up of the ways and means by which people communicate i.e. data communications, voice communications, video communications, broadcast messaging, command and control communications, emergency response communications, etc., modifying radio devices easily and cost-effectively has become critical. Software defined radio (SDR) technology brings the flexibility, cost efficiency and power to drive communications forward, with wide-reaching benefits realized by service providers and product developers through to end users.

While traditional hardware is still in common use and will be for quite some time to come Software Define Radio (SDR) systems look very promising for the future. The flexibility of software based systems with regards to various use cases and adaptability to a variable environment will make them mainstream for many different applications. They will be capable of replacing traditional hardware systems like FM radio or TV broadcast as well as cell based phone systems.

Software-Defined Radio (SDR) Forum [www.sdrforum.org] defines SDR technology as "radios that provide software control of a variety of modulation techniques, wide-band or narrow-band operation, communications security functions (such as hopping), and waveform requirements of current & evolving standards over a broad frequency range." In a nutshell, Software-Defined Radio (SDR) refers to the technology wherein software modules running on a generic hardware platform consisting of DSPs and general purpose microprocessors are used to implement radio functions such as generation of transmitted signal (modulation) at transmitter and tuning/detection of received radio signal (demodulation) at receiver.

SDR technology can be used to implement military, commercial and civilian radio applications. A wide range of radio applications like Bluetooth, WLAN, GPS, Radar, WCDMA, GPRS, etc. can be implemented using SDR technology [1].

Software-defined radio (SDR) employs digital modulation and demodulation techniques to solve traditionally analogue communications problems. To achieve this, the SDR must be capable of synthesizing and analyzing high-frequency analogue signals. However, sampling frequencies are limited by the capabilities of digital-to analogue and analogue-to-digital converter technology.

SDR is a radio communication system where components that have typically been implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented using software. By this definition, the CEVA, Octasic and Imagination platforms all qualify as SDR. All three of these platforms do things in software that were previously done in hardware.

A definition of an SDR is a radio in which some or all of the physical layer functions are software–defined. The physical layer function is the layer within the wireless protocol in which processing of RF, IF, or baseband signals (including channel coding) occurs. Many of today's SDRs have part of the signal processing implemented in software [2].

1.2 Development of SDR

Wireless communication devices are composed of three main entities; signaling, physical hardware, and its functionalities. The primitive communications devices had very simple signaling, analog hardware, and limited functionality. In time, each of these entities evolved significantly. Some of the key milestones in this progress are transition from analog hardware to digital hardware and then introduction of sophisticated processors. We believe that FPGAs will not only be a big help in that transition but actually will become commodity components in such systems. This can be stated because of their flexibility to partition hardware and software. With that the transition will not need to occur at once or in a single monolithic step but can happen slowly in a controlled manner. The system designer moves components piece by piece and is capable of choosing the optimal way to implement either in soft-hardware through the FPGA fabric or software altogether with embedded processors. Such an approach requires strong building blocks that are available in modern FPGAs [3].

This is followed by the development of Software Defined Radio (SDR) structures and virtual hardware that are under development currently. SDR is envisioned initially to be a promising solution for interoperability, global seamless connectivity, multi-standard, and multi-mode issues.

Software Defined Radio (SDR) system is a useful and adaptable future-proof solution to cover both existing and emerging standards, it provides elements with reconfigurability, intelligence and software programmable hardware. In addition, the emerging user requirements on reconfigurable mobile systems and networks are paving the way for the introduction of re-configurability in future mobile systems.

A software-defined radio system, or SDR, is a radio communication system where components that have been typically implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded computing devices. Basically Software Defined Radio (SDR) is based on software defined wireless communication protocols instead of hardwired implementations. In other words, frequency band, air interface protocol and functionality can be upgraded with software download and update instead of a complete hardware replacement. SDR provides an efficient and secure solution to the problem of building multi-mode, multi-band and multifunctional wireless communication devices. An SDR is capable of being re-programmed or reconfigured to operate with different waveforms and protocols through dynamic loading of new waveforms and protocols. These waveforms and protocols can contain a number of different parts, including modulation techniques, security and performance characteristics defined in software as part of the waveform itself [4].

1.3 Definitions

1. RADIO

Any kind of device that wirelessly transmits or receives signals in the radio frequency (RF) part of the electromagnetic spectrum to facilitate the transfer of information. Traditional hardware based radio devices limit cross-functionality and can only be

modified through physical intervention. This results in higher production costs and minimal flexibility in supporting multiple waveform standards. This problem is solved by software defined radios.

2. Soft ware Controlled Radio

Radio in which some or all of the physical layer functions are software controlled. In other words, this type of radio only uses software to provide control of the various functions that are fixed within the radio.

3. Software Defined Radio (SDR)

It is defined as the "Radio in which some or the entire physical layer functions are software defined".

A radio in which some or all of the physical layer functions are Software Defined is said to be SDR. In simpler words, the software is used to determine the specification of the radio and what it does. If the software within the radio is changed, its performance and function may change.

SDR defines a collection of hardware and software technologies where some or all of the radio's operating functions (also referred to as physical layer processing) are implemented through modifiable software or firmware operating on programmable processing technologies. These devices include field programmable gate arrays (FPGA), digital signal processors (DSP), general purpose processors (GPP), programmable System on Chip (SoC) or other application specific programmable processors [4].

4. Adaptive Radio

Adaptive radio is radio in which communication systems have a means of monitoring their own performance and modifying their operating parameters to improve this performance. The use of SDR technologies in an adaptive radio system enables greater degrees of freedom in adaptation, and thus higher levels of performance and better quality of service in a communications link.

5. Cognitive Radio

Cognitive radio is radio in which communication systems are aware of their internal state and environment, such as location and utilization on RF frequency spectrum at that location. They can make decisions about their radio operating behavior by mapping that information against predefined objectives. Cognitive radio systems automatically adjust to frequencies and power levels.

Cognitive radio is further defined by many to utilize Software Defined Radio, Adaptive Radio, and other technologies to automatically adjust its behavior or operations to achieve desired objectives. The utilization of these elements is critical in allowing end-users to make optimal use of available frequency spectrum and wireless networks with a common set of radio hardware. This will reduce cost to the end-user while allowing him or her to communicate with whomever they need whenever they need to and in whatever manner is appropriate.

6. Intelligent Radio

Intelligent radio is cognitive radio that is capable of machine learning. This allows the cognitive radio to improve the ways in which it adapts to changes in performance and environment to better serve the needs of the end user.

These types of radios, i.e. adaptive radio, cognitive radio and intelligent radio, do not necessarily define a single piece of equipment, but may instead incorporate components that are spread across an entire network.

7. Radio Awareness

Radio awareness is the functionality with which a radio maintains internal information about its location, spectrum environment, or internal state, and is able to detect changes in that information. Radio awareness is required for supporting the cognitive control mechanism.

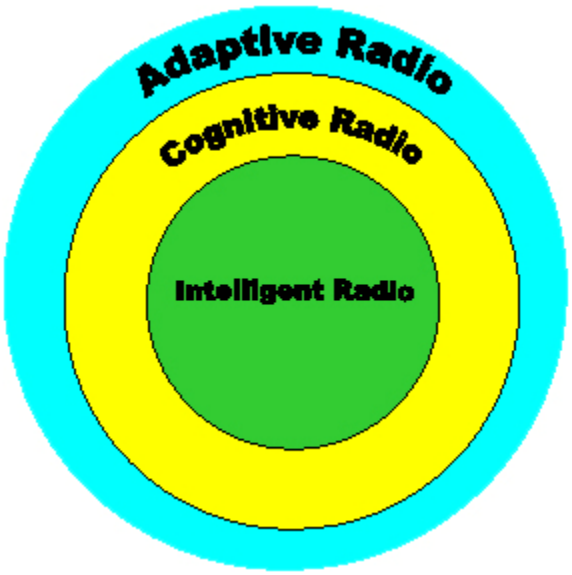


Fig 1.1 Venn diagram illustrating relationship between associated advanced wireless technologies

1.4 Software Defined Radio - Related Technologies

SDR can act as a key enabling technology for a variety of other reconfigurable radio equipments commonly discussed in the advanced wireless market.3 While SDR is not required to implement any of these radio types, SDR technologies can provide the above types of radio with the flexibility necessary for them to achieve their full potential, the benefits of which can help to reduce cost and increase system efficiencies.

1.5 The Need for SDR

SDR has generated tremendous interest in the wireless communication industry for the wide ranging economic and deployment benefits it offers. Following are some of the problems faced by the wireless communication industry due to implementation of wireless networking infrastructure equipment and terminals completely in hardware:

1. Commercial wireless network standards are continuously evolving from 2G to 2.5G/3G and then further onto 4G. Each generation of networks differ significantly in link-layer protocol standards causing problems to subscribers, wireless network operators and equipment vendors. Subscribers are forced to buy new handsets whenever a new generation

of network standards is deployed. Wireless network operators face problems during migration of the network from one generation to next due to presence of large number of subscribers using legacy handsets that may be incompatible with newer generation network. The network operators also need to incur high equipment costs when migrating from one generation to next. Equipment vendors face problems in rolling out newer generation equipment due to short time-to-market requirements.

2. The air interface and link-layer protocols differ across various geographies (for e.g., European wireless networks are predominantly GSM/TDMA based while in USA the wireless networks are predominantly IS94/CDMA based). This problem has inhibited the deployment of global roaming facilities causing great inconvenience to subscribers who travel frequently from one continent to another. Handset vendors face problems in building viable multi-mode handsets due to high cost and bulky nature of such handsets.

3. Wireless network operators face deployment issues while rolling-out new services/features to realize new revenue-streams since this may require large-scale customizations on subscribers' handsets.

SDR technology enables implementation of radio functions in networking infrastructure equipment and subscriber terminals as software modules running on a generic hardware platform. This significantly eases migration of networks from one generation to another since the migration would involve only a software upgrade. Further, since the radio functions are implemented as software modules, multiple software modules that implement different standards can co-exist in the equipment and handsets. An appropriate software module can be chosen to run (either explicitly by the user or implicitly by the network) depending on the network requirements. This helps in building multi-mode handsets and equipment resulting in ubiquitous connectivity irrespective of underlying network technology used [1].

1.6 Benefits of SDR

1. **For Radio Equipment Manufacturers and System Integrators**, SDR Enables:

- A family of radio "products" to be implemented using a common platform architecture, allowing new products to be more quickly introduced into the market.

- Software to be reused across radio "products", reducing development costs dramatically.
- Over-the-air or other remote reprogramming, allowing "bug fixes" to occur while a radio is in service, thus reducing the time and costs associated with operation and maintenance.

2. **For Radio Service Providers**, SDR Enables:

- New features and capabilities to be added to existing infrastructure without requiring major new capital expenditures, allowing service providers to quasi-future proof their networks.
- The use of a common radio platform for multiple markets, significantly reducing logistical support and operating expenditures.
- Remote software downloads, through which capacity can be increased, capability upgrades can be activated and new revenue generating features can be inserted.

3. **For End Users** - from business travelers to soldiers on the battlefield, SDR technology aims to:

- Reduce costs in providing end-users with access to ubiquitous wireless communications enabling them to communicate with whomever they need, whenever they need to and in whatever manner is appropriate.

1.7 Advantages of SDR over traditional radio communications:

1. A single SDR device can perform multiple functions simply by changing software modules.
2. System updates can be implemented in software to be downloaded via the transmission network. These include updates to both the software application and to any soft configurable hardware.
3. Highly configurable signal processing systems can be developed with modifications and upgrades made far simpler to implement than more traditional DSP systems.
4. More flexible communications protocols can be developed that adapt to their environment transparently to the system user (e.g. searching for and operating in

locally available bands). This project aims to create a hardware platform that can facilitate simple narrow-band SDR signal reception applications [5].

1.8 Objective of Thesis

- Study of Software Defined Radio architectures, its benefits in practical usage.
- Study of SMT-8036 kit.
- Study of Digital modulation Techniques.
- Implementation of Digital modulation Techniques on SMT-8036.
- Study of LDPC codes.
- Implementation of LDPC codes and their implementation with different digital modulation techniques on SMT-8036.
- SMT-8036 based implementation of secured Software Defined Radio system for adaptive modulation technique with adaptive receiver.

1.9 Outline of Thesis

The thesis is organized as follows:

In chapter 1, Introduction of Software Defined Radios, definitions related to SDR, its development, need and benefits over earlier technologies are discussed.

In chapter 2, Literature survey of Software Defined Radios is given.

In chapter 3, System architectures of Software Defined Radios and design issues of an SDR transmitter and receiver are briefly presented.

In chapter 4, Details and properties of LDPC codes, along with their coding and decoding procedures are included. Digital modulation Schemes, which are implemented practically by me on SMT 8036 along with LDPC coding are also discussed.

In chapter 5, Details of work done, results and conclusions are included along with future scope.

CHAPTER 2

LITERATURE REVIEW

A. P. Vinod & Edmund M-K. Lai & Amos Omondi [6] gives a special issue which gives the receiver architecture as given in fig 2.1. This issue tells us that the ultimate objective of Software Defined Radio (SDR) is to replace the entire analog signal processing in the wireless transceivers with digital signal processing. This will provide the flexibility, through reconfiguration or reprogramming, to enable the transceivers to work with different air-interfaces standards using a single generic hardware platform. Most of the research in SDR to date has been focused on base stations, which do not have as tight constraints in size and power consumption as handsets. However, in order to realize the promise of integrated services and global roaming capabilities, the design and development of a multi-standard SDR handset with dynamic reconfigurability is crucial. The size and power constraints imposed by handsets with limited resources pose a major design challenge. This special issue brings together 11 papers that address various aspects of this challenge from the perspectives of wireless systems, computing architectures, embedded systems and signal processing.

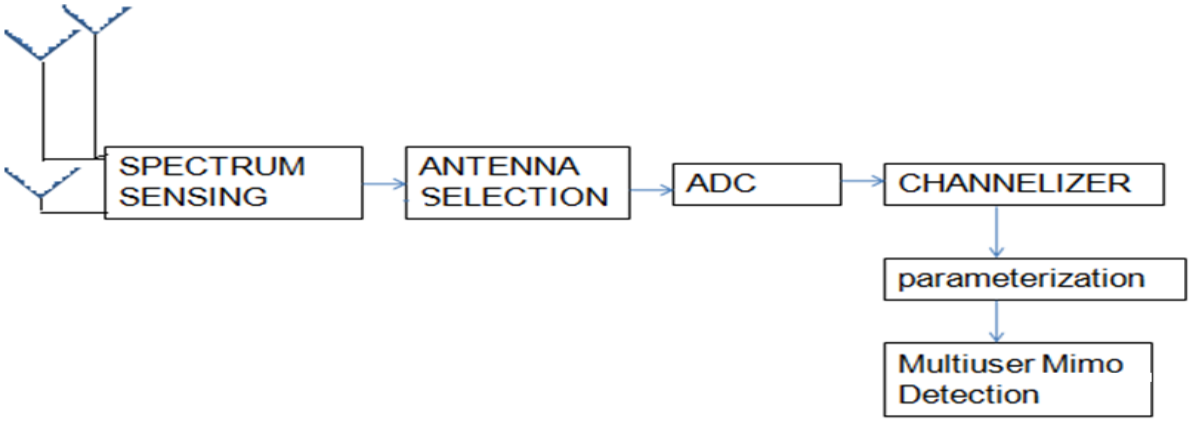


Fig 2.1 Summarized receiver structure of issue [6]

In figure 2.1, after ADC, the digitized received signal will need to be separated into signals within the various bands specified by the communication standard. This is performed by the channelizer which has the highest computational requirements of the whole SDR system as shown in figure 2.2.

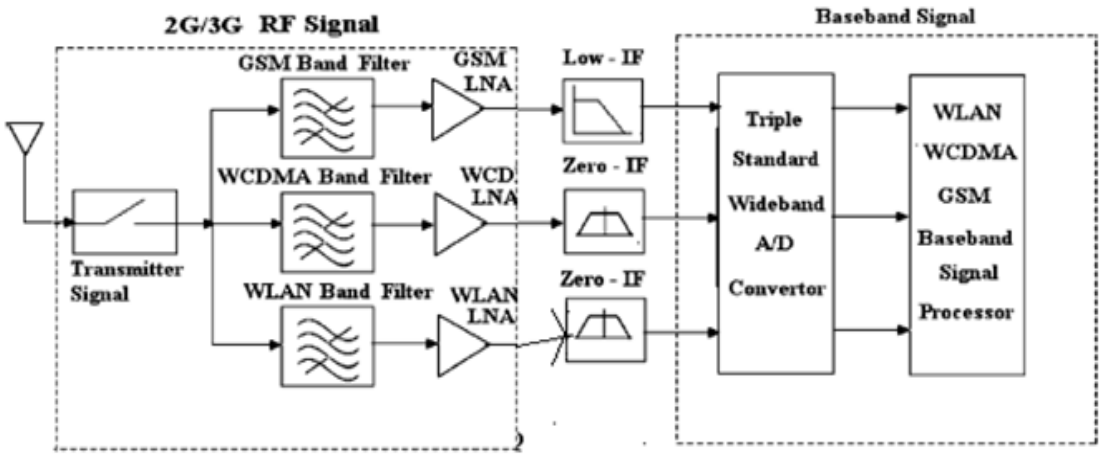


Fig 2.2 Channelizer [6]

M. Saravanan, Dr. S. Ravi [7] presents that Software defined radio development aims at wideband RF access and software partitioning for plug-and-play type of use. The development is facilitated by progress in silicon capabilities, silicon capabilities, signal processing power of new and future processors and reconfiguration methods (software download, smartcards, etc.). The future wireless and multimedia systems differ from old systems and from each other in terms of access methods, modulation, coding, etc. and they will also provide new type of services. This results in need for many product platforms unless we aren't able to design a common platform that could be programmed to different standards. The software radio is a good solution to this problem since it is reconfigurable to suit different standards at different continents. For this IS-95 CDMA technique is modified as a method of Spread Spectrum Multiple Access technique called Frequency Hopping/Multiple Carrier Direct Sequence-Code Division Multiple Access (FH/MC DS-CDMA). In this work, software defined radio (SDR) based FH/MC DS-CDMA transreceiver is proposed for future generation as well as existing generation Networks, which can carry multimedia applications at higher speed. Multiple carriers can be allotted as per user requirements with high secrecy and multiple encryptions.

David Murotake, Ph.D. and Antonio Martin [8], the authors provided details of wireless network threats discovered during software defined radio (SDR) threat analysis study that exposed a potentially serious flaw in the security architecture of SDR. The reconfigurable

radio terminal, and the host to which it is attached, is potentially vulnerable both to exploitation and malicious reconfiguration as a result of "proximity wireless" and Internet based network attacks. The best defensive approach to a blended attack is a "multi-layered" defense, or defense in depth. That is, a combination of methods, implemented in both hardware and software, is implemented throughout the end-to-end communication path if possible.

Author also suggests Virtex II/Pro embedded PowerPC processors employed prototypes of MILS certifiable separation kernel technology to run the security layer applications which included certified IPv4 and IPv6 compliant protocol stacks and firewall applications.

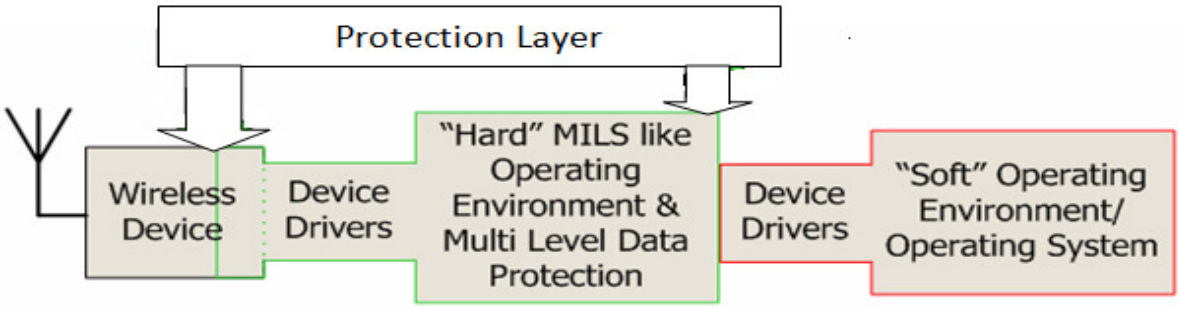


Fig 2.3 Prototypes of MILS certifiable separation kernel technology [8]

For more secure SDR system, we need the ability of the computing platform to be maliciously re-programmed or reconfigured, by-passing encryption, secure browsers and Private Networks through use of root kits/key logger Trojan horses and similar malicious software, may be easy to overlook.

Goran T. Djordjevic, Ivan B. Djordjevic, and Predrag N. Ivanis [9], In this paper, authors tell us about the influence of receiver imperfections on bit error rate (BER) degradations in detecting low-density parity-check coded multilevel phase-shift keying signals transmitted over a Rician fading channel. Based on the analytical system model which we previously developed using Monte Carlo simulations, we determine the BER degradations caused by the simultaneous influences of stochastic phase error, quadrature error, in-phase-quadrature mismatch, and the fading severity.

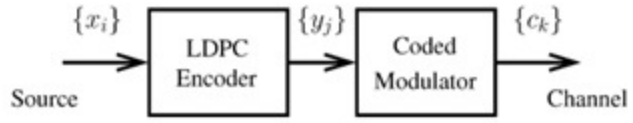
To achieve high spectral efficiency in contemporary wireless communication systems, multilevel phase-shift keying (MPSK) is used. As the number of phase levels increases, the system performance is more sensitive to receiver imperfections. The bit error rate (BER) is generally influenced by several receiver imperfections, including phase error, quadrature error, and in-phase-quadrature (I-Q) mismatch. Since the reference carrier signal in the receiver is reconstructed from a noise-corrupted signal transmitted over a fading channel, there is a difference between the extracted reference signal phase and the received signal phase. Quadrature error results from the phase shifts other than 90 degrees between the I and Q receiver branches. The I-Q gain mismatches are generated by imperfect mixers and filters at the receiver branches.

The BER degradations caused by receiver imperfections when low-density parity-check (LDPC) coded MPSK signals are transmitted over the Rician fading channel are determined by computer simulations. Unlike others, where the static or uniformly distributed phase error was observed, we consider the phase error as a stochastic process with the Tikhonov probability density function (PDF), which is a more realistic model. While other studies treated the problem of the influence of receiver imperfections on BER performance, where uncoded MPSK signal transmission was observed, we use LDPC codes for signal encoding.

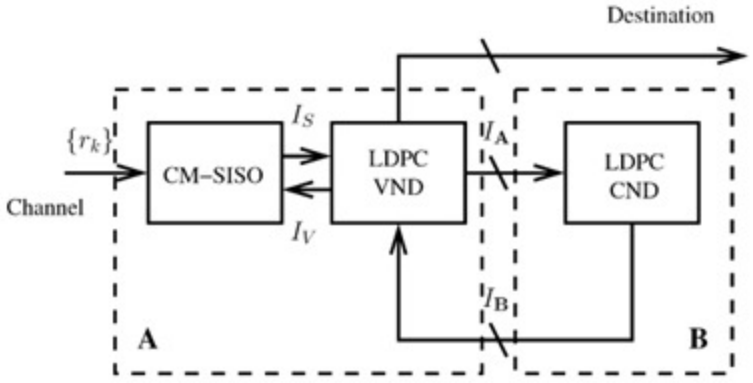
Michele Franceschini, Gianluigi Ferrari, Riccardo Raheli and Aldo Curtoni [10] In this paper, authors discuss about new channel coding techniques: turbo codes and low-density parity-check (LDPC) codes. These codes are well suited for communication over the additive white Gaussian noise (AWGN) channel, allowing near-capacity error-free transmission. Also, it is shown how to partition the set of all LDPC codes in equivalence classes characterized by the same degree distributions.

In the literature, the design of schemes given by the concatenation of an encoder (either convolutional or block) with a high-order modulator has been discussed. In bit-interleaved coded modulation (BICM) schemes, consisting of the concatenation of a binary encoder, a bit interleaver and a high-order memory less mapper are proposed and analyzed. At the decoder

side, a soft demapper generates reliability values for the bits embedded in each modulated symbol, and these values feed a decoder corresponding to the binary encoder used at the transmitter side. An extension of BICM schemes, denoted as BICM with iterative decoding (BICM-ID), is proposed: iterative information exchange between the soft demapper and the decoder is considered and performance advantages are observed. It is shown how to analyze, using extrinsic information transfer (EXIT) charts, the performance of LDPC codes transmitted over a multi-input–multi-output (MIMO) channel through a MIMO modulator, with a corresponding soft-input–soft-output (SISO) module at the receiver side. This can be interpreted as a special instance of a BICM-ID scheme, where the interleaver is not required because of the random nature of the LDPC code. It is a way of designing LDPC codes suited to memory less channels and approaching the capacity limit. Efficient differential modulation schemes suitable to iterative detection/decoding are proposed, where differentially encoded (DE) phase-shift keying (PSK) is considered.



System model: transmitter side.



System model: receiver side.

Fig 2.4 System Models [10]

In this paper, we show how to design good LDPC codes for DE modulations. We consider the concatenation of an LDPC code with a differential modulator for both PSK and QAM. At the receiver side, we follow the approach which enables an accurate EXIT chart-based system performance evaluation.

In particular, this technique can take into account channel impairments, which are usually neglected for the sake of feasibility by other analysis methods. They compare the performance of codes optimized for DE modulations with the performance of standard LDPC codes, i.e., optimized for transmission over a memory less channel. They show that LDPC codes optimized for DE modulations significantly outperform standard LDPC codes. Vice versa, the obtained optimized codes are shown to be tailored specifically for the particular DE modulation format and the proposed receiver scheme: in other words, while they perform well if used jointly with DE, they perform poorly with other modulation schemes. This will be shown to depend on the presence of a large fraction of degree-2 variable nodes.

Mansour Ahmadian, Zhila (Jila) Nazari,Nory Nakhaee and Zoran Kostic [11] presents a paper focuses on the major phases present in the development of an SDR system: design; simulation; code generation and verification. The paper will illustrate the use of Model Based Design methodology and tools to integrate the major development phases into one continuous design cycle. Advanced system design concepts including, simulation, code generation, hardware in the loop testing is presented in it.

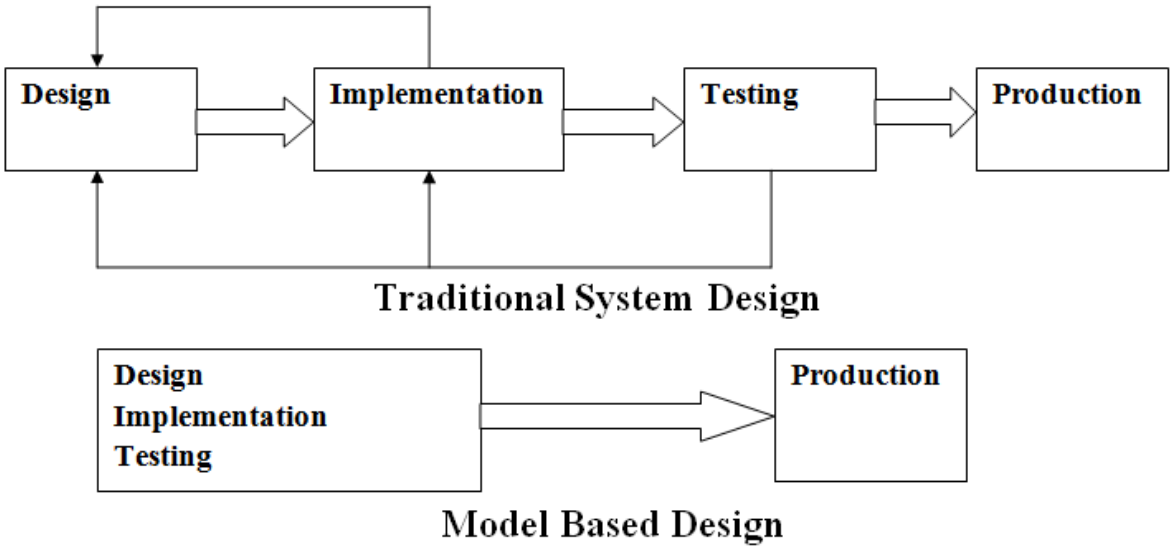


Fig 2.5 Traditional System Design vs. Model Based Design [11]

Complicated systems can be created by using mathematical models, representing system components and their interactions with their surrounding environment on model based design

tool for example MATLAB Simulink. After that code is generated by using code-generation technology. Then we can implement that code on hardware provided**.**

Adrian Tarniceriu, Bogdan lordache, Silvian Spiridon[12] This paper analyses the typical digital modulation techniques used in today's wireless communications. The paper presents the characteristics of the modulation techniques and determines the figure of merit for each particular modulation: Bit Error Rate (BER) vs. Signal-to- Noise Ration (SNR). The analysis emphasizes the importance of such figure of merit in the context of Software Defined Radios (SDR). This paper presented an analysis of the modern modulation techniques. These are used in the latest wireless standards, such as IEEE - 802.16, also known as WiMAX. The SDR systems that can support multiple modulation types can choose with the help of Reconfigurable digital radio part to the appropriate modulation type according to the expected channel quality.

Raúl Dopico-López , José M. Camas-Albar, ed. al. [13] this paper shows that demand in communication characteristics has caused the need of advancements in SDR terminals in order to efficiently deal with high data-rate processing and innovative capabilities. Consequently, different SDR architectures have been proposed by providers and developers in order to palliate communication performance problems, mainly providing lightweight software implementations or multi-core hardware replacements. However, those solutions are neither flexible enough to be integrated with conventional SDR platforms nor scalable enough to have their inner software adapted to forthcoming hardware advancements.

Some multi-core based solutions have been recently adopted to counteract SDR performance problems. However, current applications are not able to fully exploit such parallel architectures and hence obtain considerable improvements in the SDR field.

Therefore, an adaptable multi-core architecture is presented in this paper as shown in figure 2.4 for a suitable solution to cover nowadays SDR performance needs, characterized by supporting the heaviest SDR processing and being able to be fully exploited by means of advanced parallelism techniques.

Physical connections must exist among MPSoC coprocessor and all GPP and DSP type processors allocated in the SDR comprising “parallelization candidate” functions in order to reduce the number of hops to the MPSoC, and hence the slowdown produced by the communications time. The availability of a high number of processors is not a sufficient ingredient to get high parallelism level.

Applications should be adapted to this multiprocessing environment by using advanced parallel programming methods and techniques. So we use libraries for making reliable connections between different processors.

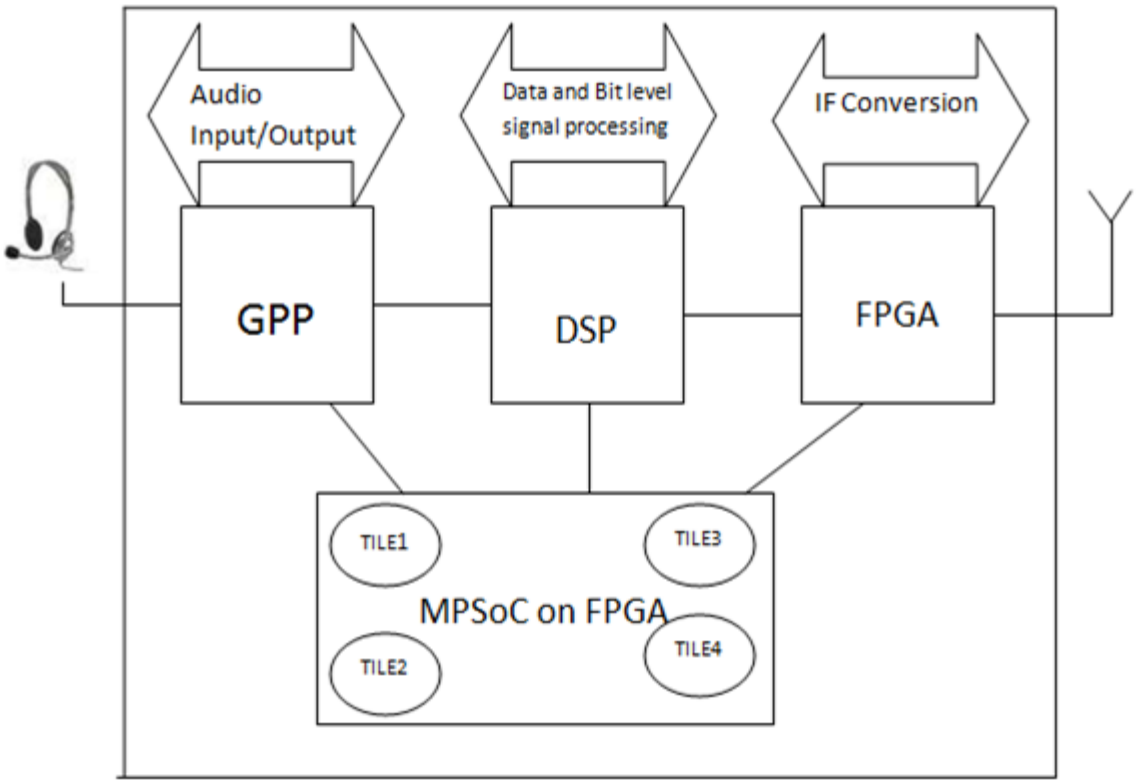


Fig 2.6 Multi-core architecture for recent SDR [13]

Feng Ge and Charles W. Bostian [14] This paper addresses two practical issues governing Software Defined Radio (SDR) performance that are often overlooked: RF front end nonlinearity and dynamic computing resource allocation. Current SDR performance still depends heavily on analog radio frequency (RF) technologies. Inter-modulation and other

nonlinear effects in these devices make it very challenging to create an RF front-end that is applicable to a variety of signals with widely differing center frequencies, modulation bandwidths, and power levels. Unanticipated inter-modulation products can seriously degrade receiver performance. Digital signal processing in wireless communication is fundamentally a real-time task. Achieving real-time performance in SDRs puts stringent requirements on dynamic computing resource allocation; these requirements may be much higher than those in conventional digital radios.

Digital communication systems are fundamentally real-time because data are transmitted as segments at the PHY layer; to receive all the data, the time to process each data segment is limited since the radio must accept and process each incoming data segment before the next one arrives. Such requirements are stringent in the digital domain when signal processing functions are complicated and there is limited battery life to support sufficient computing resources. It is possible to achieve both real-time performance and low power consumption for a single digital waveform because (1) the RF radio signals can be down-converted to the baseband with a necessary minimum data rate, thus minimizing required computing resources; (2) the required digital signal processing functions can be fully optimized and executed on Application-Specific Integrated Circuits (ASICs).

However, it is quite a different story for SDR. First, moving digital signal processing functions closer to the radio antenna requires a high data rate and more functional components in the software domain; this implies additional computing resources such as computing cycles in signal processing and memory space to hold more signal samples. Second, SDR usually needs to reconfigure itself to support more than one application; this makes it very challenging to highly optimize computing resource allocation for dynamically configured radio functions.

Among the three most popular computing systems for SDR development – field-programmable gate arrays (FPGAs), general purpose processors (GPPs), and digital signal processors (DSPs) – FPGAs are difficult to reconfigure and consume significant power, but they can support real-time performance. GPPs are capable of reconfiguration but are power

demanding and also suffer from execution latency. DSPs consume less power and can reconfigure for simple real-time tasks but are not able to support computationally intensive tasks.

We concluded that SDR may require much more CPU resource and battery power than conventional digital radios. Very importantly, SDR performance may deteriorate abruptly if not enough computing resources are available because of the real-time constraint.

Mohamed Ratni, Dragan Krupezevic, et. al. **,[15]** worked on Direct conversion receiver (DCR) which appeals broadband because it requires only one major integrated circuit for the RF portion. DCR can be implemented on monolithic integration easier than heterodyne receiver and DCR suffers less from bulky expensive filters needed for image rejection. The Six-Port receiver is used in high frequency microwave analyzers as well as in Direct Conversion Receiver. Where a wide bandwidth is required, an accurate -90 degree phase shifter is needed at a specific frequency.

The output RF signal is fed to the transmit antenna after a RF switch. The purpose of the RF switch is to switch between different standards. This includes filtering, RF amplification, down conversion and base-band amplification. On the other hand, digital signal processing needs to be reconfigurable and reprogrammable by software downloading. A power divider constitutes with a resistive structure and a phase shifter. The DC signal is then proportional to the RF signal amplitude. In order to have a multi-band and multi-channel receiver, a channel selection can he achieved by simple tuning of the local oscillator signal. The direct down converter technique consists of down converting the signal from RF to baseband without any mixer and with no intermediate frequency.

T. Nesimoglu, M. A. Beach, J. R. MacLeod and P. A. Warr [16] worked on receivers and mixers that are used for down-converting the received radio frequency (RF) signal to baseband or to an intermediate frequency (IF) for further processing and in transmitters, for up-converting. The non-linearity of mixers creates undesired effects like harmonic (HD) and inter-modulation distortion (IMD), spreading the spectrum to a wider bandwidth. The HD

can be filtered out since it appears at one octave higher frequency than the fundamentals. IMD creates ACI to other nearby channels as well as co-channel interference within the same channel. The SDR receiver frontend receives the wanted channel and a number of nearby signals. Non-linear mixers will down-convert all of these received channels to IF. During this frequency translation process, in-band interference will be added to the wanted channel, making it more difficult for the receiver to correctly detect the information.

Mixer linearization schemes used are:

A. Feed-forward Mixer: The signal used as a reference is only an approximation to the required reference signal. The output of the main mixer, which includes IMD is coupled and added in anti-phase to the output of the secondary mixer, thus cancelling the fundamental signals.

B. Single-Loop Feed-forward Mixer: The secondary mixer is driven with a much higher RF signal than the main mixer to provide a high level of IMD which is an approximation to the required error signal, also providing a high SNR. This error signal is amplitude and phase adjusted before being added to the final IF output for suppressing the IMD.

The distorted output of the down-converting mixer at IF is coupled, amplified, frequency retranslated back to RF and filtered. The reference signal at the receiver frontend is also coupled and added in anti-phase to the frequency retranslated sample of the IF output with amplitude correction. This process cancels the fundamental signals and produces an error signal including only the IMD products. This error signal is then combined with the received RF input signal with correct amplitude and phase relation to pre distort the saturated down-converting mixer.

Ajay Kr. Singh, Ankita Taneja, et. al. [17] In this paper it is stated that the SDR allows multiple waveforms to be rapidly ported. It is a communications device whose functionality is defined in software. Defining the architecture also enhances scalability and provides plug-and-play behavior for the components of a radio. SDR controls Transmit and receive frequencies, selects the desired bandwidth, performs the transmit gain control (TGC) and automatic gain control (AGC) functions, sets and monitors the output power level, provides system fault handling and provides the man-machine interface.

The encryption software process configures and monitors security with a recent trend toward performing the entire encryption process in software. Networking aspects of the data traffic are either completely or partially implemented in software. Finally, the man machine interface is completely defined within a software process. The goal is to pass the signal entering the analog to digital converter (ADC) directly to the digital to analog converter (DAC), without actual processing being performed. This process is called talk-through.

S. Glass, V. Muthukkumarasamy, M. Portmann [18] In this paper author analyze traffic from public safety communications and uses APCO Project 25 (P25) standard. (P25 is the digital communications standard. It is used as emergency first-responder). It provides low-level access to the actual message traffic using the Wire Shark packet sniffer. A P25 radio system consists of both fixed and mobile equipment. P25 systems encode all voice traffic using the IMBE vocoder. Using an SDR approach enables a single station to simultaneously process many analog and digital signals because processing is partitioned into blocks with well-defined interfaces. The demodulation stage can be easily replaced to allow for reception of different modulation schemes whilst sharing the common code for packet assembly and decoding. Higher bandwidth requirements are met by Cognitive radio by opportunistic use of bands that are underutilized by their primary users. SDR approaches can ensure interoperability and backward-compatibility with existing equipment.

Daughter boards provide for frequency translation, amplification and filtering to enable receive and transmit access to the VHF and UHF bands used for public safety communications. If a radio needs a signal-processing block that isn't present then it can be written (often using an existing block as a starting point) and added to the framework. The receiver produces digital audio as its output and sends the decoded P25 frames to the WireShark network protocol analyzer where they can be analyzed in detail. The WireShark network protocol analyzer is used to recognize, filter and dissect P25 network traffic.

Shinichiro Haruyama, Robert H. Morelos-Zaragoza **[19]** defines SOPRANO (Software Programmable and Hardware Reconfigurable Architecture for Network) is a software defined radio platform with multiport-based direct conversion. The main features of SOPRANO are a high-level design methodology for digital circuits, a new mixer-less direct conversion

method, and software algorithms for multi-band and multi-mode operation. It is able to receive PSK and QAM signals with two different carrier frequencies at 2.45GHz and 5.25GHz by changing signal processing software.

SDR receiver should be able to receive a wide range of frequencies and bandwidths. In SOPRANO direct conversion is utilized, i.e., RF signals are down converted to baseband signals in just one stage of mixing. Its advantages are wide band reception capability, low power consumption, immunity to image frequency and adjacent channel interferences, relaxing the requirements of the channel selection filter, tolerance to variations of the input power level, making the dynamic range to depend mainly on ADC resolution, wideband operation and digital calibration. Problems such as flicker noise and DC offset can be removed with a spectrum shaping method by line coding.

The algorithms implemented in SOPRANO can be broadly categorized as
(1) Multiport specific (IQ computation and digital calibration) and
(2) Direct-conversion specific (DC offset and IQ imbalance compensation).

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN ISSUES

The basic concept of SDR is to position the digital-to-analog separation as close as possible to the antenna. This is accomplished by implementing many functions traditionally performed by analog circuits using reconfigurable digital circuits. Successfully implemented, this can deliver an obsolescence-proof radio product which can support many existing and future air-interfaces and modulation formats [20].

3.1 SDR Transmitter

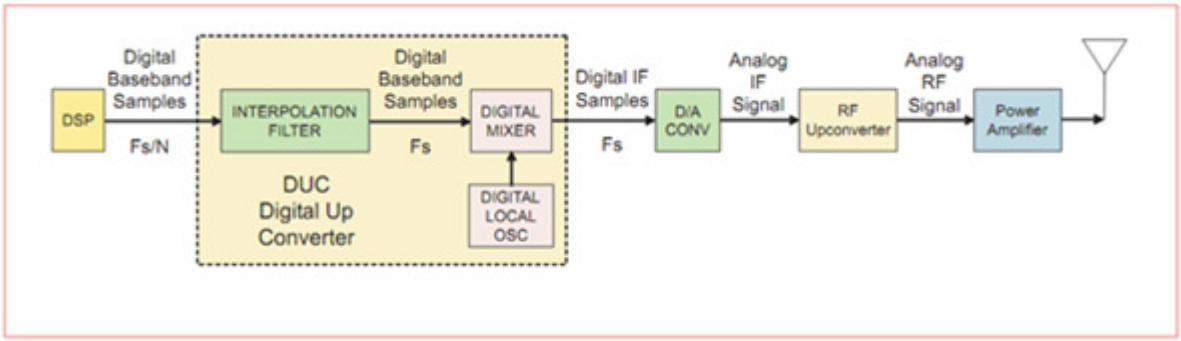


Fig 3.1 SDR Transmitter Block Diagram [21]

The input to the transmit side of an SDR system is a digital baseband signal, typically generated by a DSP stage as shown above. The digital hardware block in the dotted lines is a DUC (Digital Up Converter) that translates the baseband signal to the IF frequency. The D/A converter converts the digital IF samples into the analog IF signal. Next, the RF up converter converts the analog IF signal to RF frequencies. Finally, the power amplifier boosts signal energy to the antenna.

3.2 SDR Receiver

The RF tuner converts analog RF signals to analog IF frequencies, the same as the first three stages of the analog receiver. The A/D converter that follows digitizes the IF signal thereby converting it into digital samples. These samples are fed to the next stage which is the Digital Down Converter (DDC) shown within the dotted lines. The digital down converter is typically a single monolithic chip or FPGA IP, and it is a key part of the SDR system.

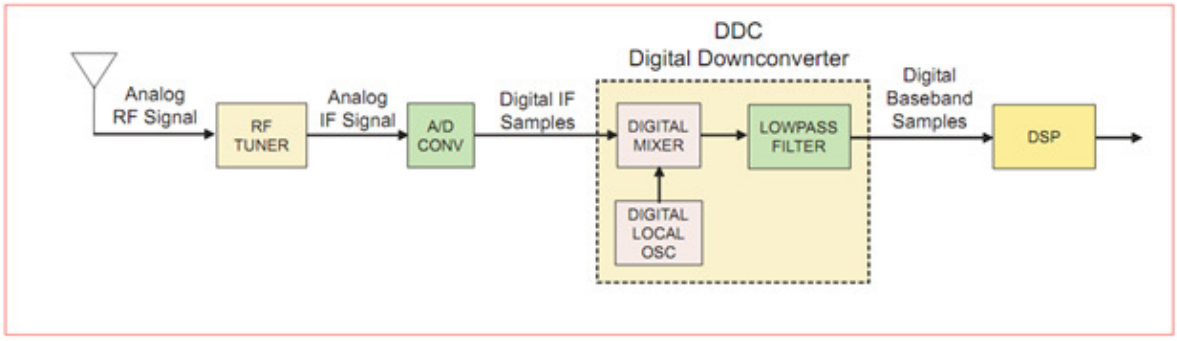


Fig 3.2 SDR Receiver Block Diagram [21]

The digital mixer and local oscillator translate the digital IF samples down to baseband. The FIR low pass filter limits the signal bandwidth and acts as a decimating low pass filter. The digital down converter includes a lot of hardware multipliers, adders and shift register memories to get the job done. The digital baseband samples are then fed to a block labeled DSP which performs tasks such as demodulation, decoding and other processing tasks. Traditionally, these needs have been handled with dedicated application specific ICs (ASICs), and programmable DSPs [21].

3.3 Ideal SDR Architecture

Ideal SDR architecture consists of three main units, which are reconfigurable digital radio, software tunable radio along with embedded impedance synthesizer, and software tunable antenna systems. This structure is illustrated in Figure 3.3. The main responsibilities of reconfigurable digital radio are performing digital radio functionalities such as different waveform generation, optimization algorithms for software tunable radio and antenna units, and controlling of these units.

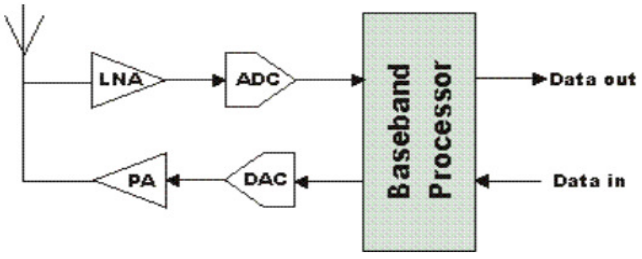


Fig 3.3 Block diagram of an 'Ideal' Software Defined Radio [4]

The key components considered of SDR are the ADC, DAC, DSP, and FPGA devices [4].

Software tunable analog front-end system is limited to the components such as RF filters, combiners/splitters, Power Amplifier (PA), Low Noise Amplifiers (LNA), and data converters. This unit has a impedance synthesizer subsystem, which is used to optimize the performance of software tunable antenna systems for an arbitrary frequency plan specified by cognitive engine. So the basic block diagram of ideal software defined radio Architecture can be constructed as given below.

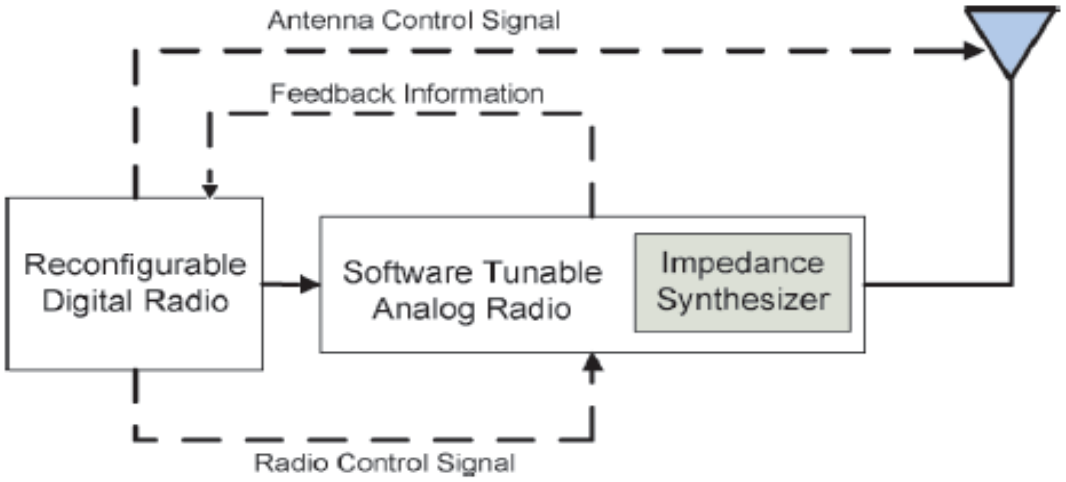


Fig 3.4 An ideal SDR architecture [22]

Reconfigurable digital radio system monitors and controls the software tunable radio system continuously (or periodically depending on system specifications). A basic relationship between the main units of SDR is described as follows. The cognitive engine sends radio configuration parameters to the reconfigurable digital radio so that it can reconfigure the entire radio according to the parameters. These parameters can be waveform type that needs to be generated (e.g. OFDM, CDMA, UWB), frequency plan (e.g. bandwidth, operating center frequency), and power spectrum specifications. Moreover, cognitive engine can request from reconfigurable digital radio to measure or calculate some parameters from environments such as location information of a particular user. Reconfigurable digital radio configures itself along with software tunable radio components and antenna systems. In order to optimize the performance of these two units, reconfigurable digital radio utilizes the feedback information from software tunable radio, especially from impedance synthesizer. Based on this information, it adjusts the parameters of software tunable radio and antenna

units through radio and antenna control signals, respectively. Finally, reconfigurable digital radio acknowledges cognitive engine that the specified configuration is performed [22].

3.4 The SMT8036 system

The SMT8036 is a development kit for SDR applications consisting of a DSP coupled with an FPGA connected to two ADCs and a DAC. It is a PCI system based on 3 main modules: C64xx-based module (SMT365-8-2) combined with a dual high-speed ADC/DAC module (SMT370), both plugged on a PCI carrier board (SMT310Q).

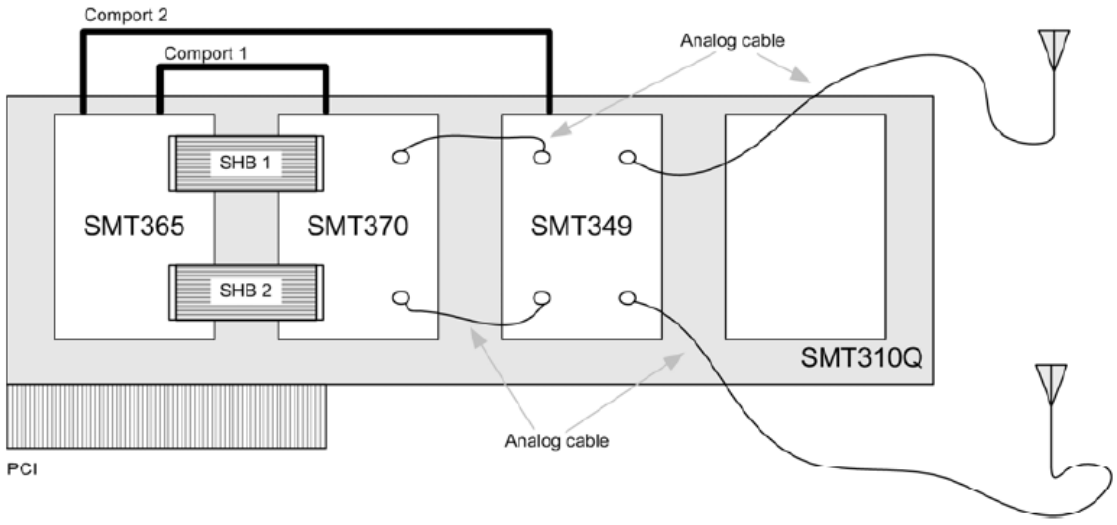


Fig 3.5 SMT 8036 [23]

The SMT8036 comprises a SMT365 connected to a SMT370. Both modules are held on a SMT310Q carrier board.

The SMT365 module provides the FPGA and DSP that are used for the processing. It has the following features:

- One 600 MHz Texas Instrument C6416 DSP
- One Xilinx Virtex II 2000 FPGA
- 8 MB ZBTRAM connected to the DSP
- Two 400 Mbps high-speed bus interfaces (SHBs)
- Six 20 Mbps comports
- 8 MB flash

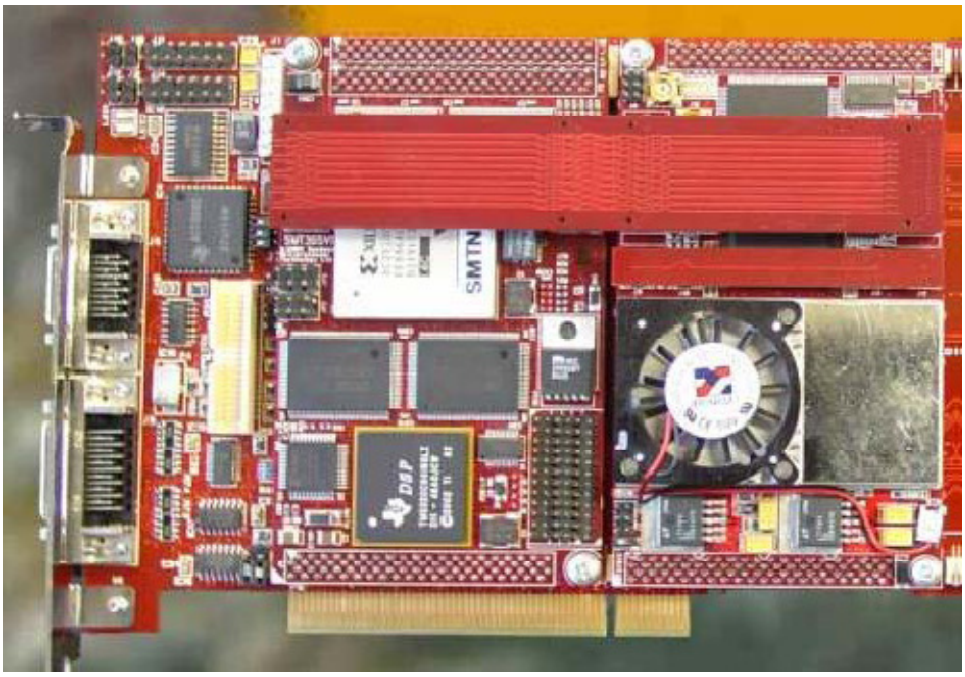


Fig 3.6 The SMT8036 system [23]

The SMT370 has the following features:

- Two 14-bits ADCs at 105 MHz
- One 16-bits dual DAC at 400 MHz (interpolation)
- 4 MB of NTSRAM at 160 MHz
- Two SHBs
- Two comports
- Xilinx Virtex II 1000 FPGA
- Xilinx PROM to configure the FPGA

The ADC of the SMT370 is connected to the FPGA of the SMT365 via its SHBA. The SMT365 is connected to the DAC of the SMT370 via its SHBB. The SMT365 sends control words to the SMT370 via its comport 0 [23].

3.4.1 RF sub module

The RF sub module provides the necessary pre-conditioning for amplifying and filtering the RF signal at 2.4-2.5GHz. The module contains the RF power amplifier and the low noise amplifier. The amplifier provides the necessary RF output power.

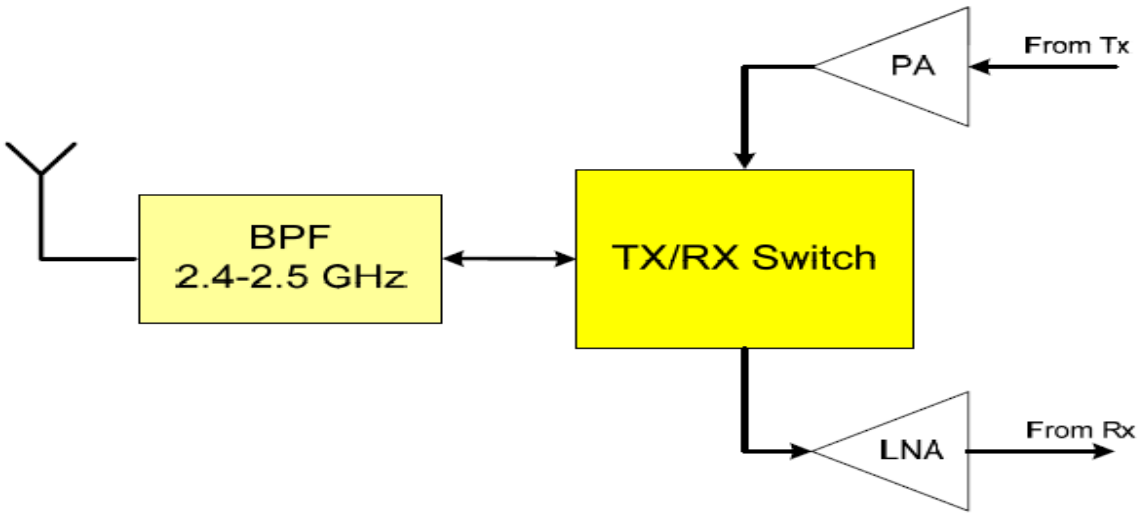


Fig 3.7 A 2.4 GHz RF sub module [24]

3.5 Overview of an SDR application

System Generator is a DSP design tool from Xilinx that enables the use of the Mathworks model-based design environment Simulink for FPGA design. Previous experience with Xilinx FPGAs or RTL design methodologies is not required when using System Generator. Designs are captured in the DSP friendly Simulink modeling environment using a Xilinx specific blockset. All of the downstream FPGA implementation steps including synthesis and place and route are automatically performed to generate an FPGA programming file.

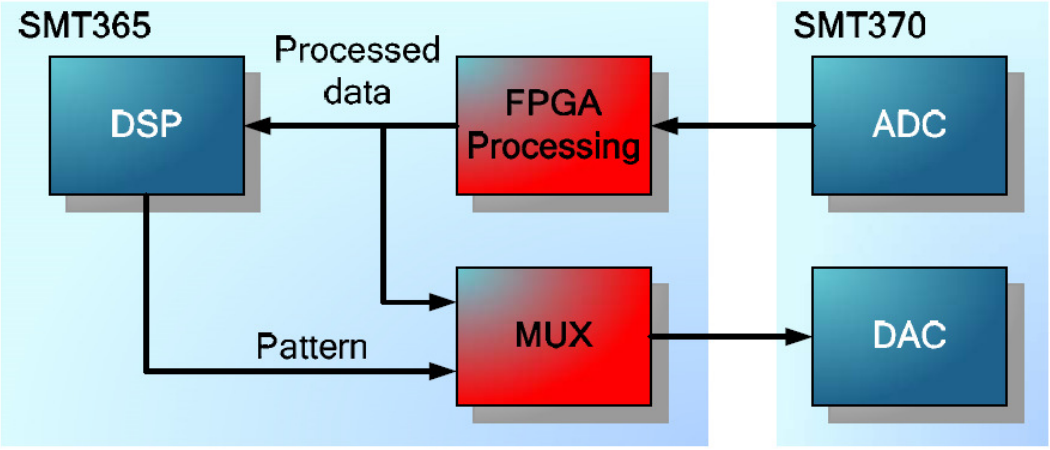


Fig 3.8 Overview of the SDR application [24]

The picture gives an overview of the SDR application. Samples from the ADC are sent to the FPGA on the SMT365 where the processing is implemented. The DAC is either used to output the result of the processing or to output a pattern specified by the DSP.

The operational mode is selected by a multiplexer. In open loop mode, the data processed by the FPGA is not provided to the DAC. In this configuration users typically connect the analog output of the DAC to the analog input of the ADC. In closed loop mode, the MUX provides the data processed by the FPGA to the DAC. This assumes a user provided ADC signal. In both modes, a copy of the processed data is presented to the DSP for further processing or display. A dialog running on the host machine allows configuring the operating mode of the application. This dialog allows the user to change settings of the SMT370.

The following diagram shows how the application is made up from a number of tasks connecting by control channels (dashed lines) and data channels (black lines). Tasks in blue execute on the DSP; some of them send graphical output to the host PC for display. Tasks in red execute on the SMT365 FPGA.

The extract task splits 32-bit values received from the ADC into two streams of 16-bit sample values. (One stream per ADC channel) The user processing task takes these streams, transforms them, and passes the results on to the combine task which reconstitutes a single 32-bit stream. This stream is sent to the duplicate tasks. The duplicate task makes a copy of the data stream and provides the original to the mux task and the copy to the storage task.

The mux task selects the data flow based on the operational mode, open or closed loop, as set by the DAQControl task. In addition, the DAQControl task configures the SMT370 by sending control data from comport 0 of the SMT365.

The storage task is used to capture a contiguous data sample. This is implemented in the FPGA to ensure that the capturing can be done even if the data stream is very fast. In your applications you might choose to omit this task, but then you will need to ensure that the data flow between the FPGA and DSP is slow enough so that data does not get lost [24].

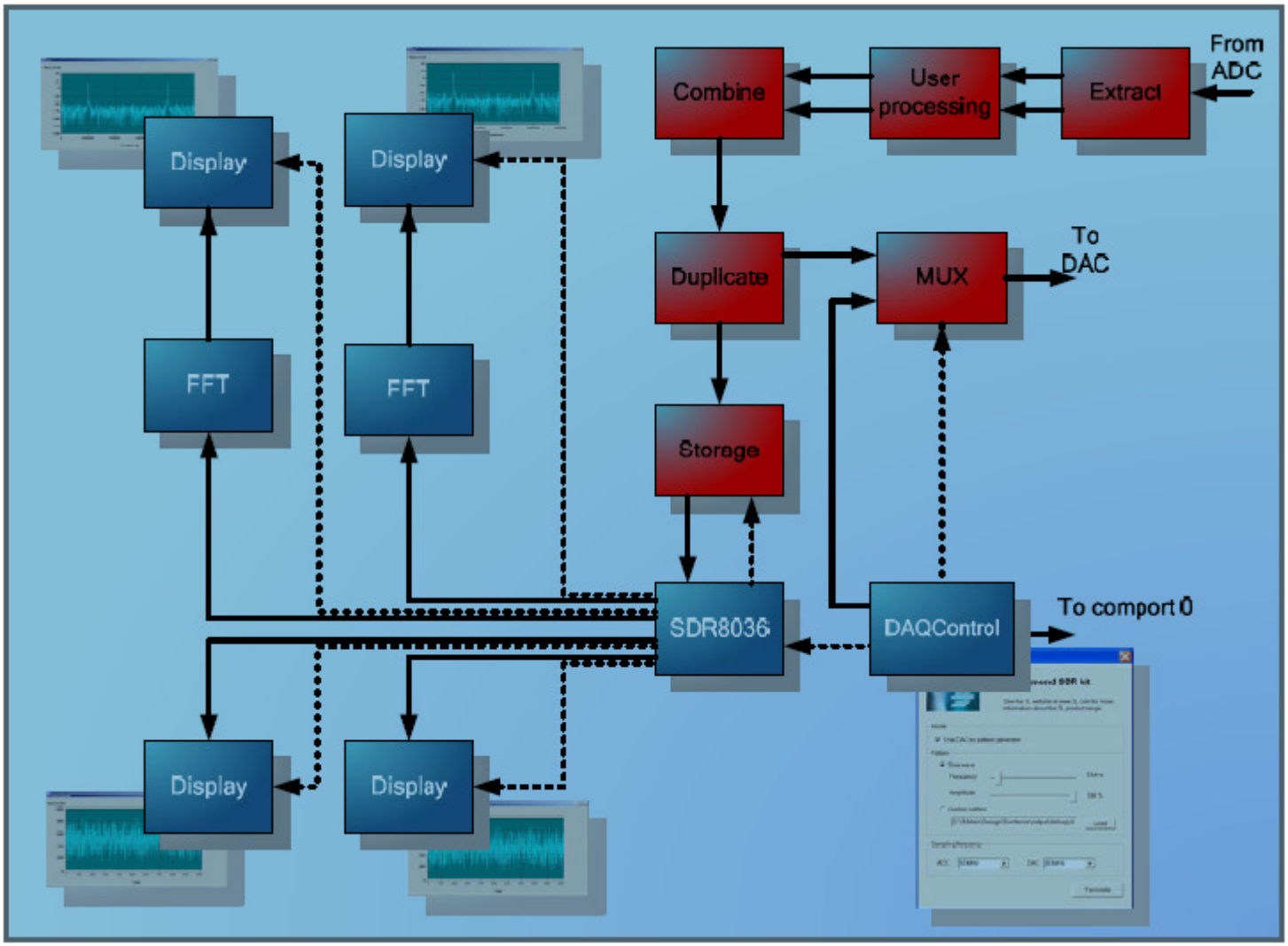


Fig 3.9 Block diagram of the SDR application [24]

The SDR8036 task receives the processed data and sends them to the FFT tasks for processing. The results, along with the original processed data, are then displayed. The SDR8036 task is also responsible for configuring the axis and display properties of the graphs using control channels.

3.5.1 Programming used

Basically, if we want to design any hardware component like counters, multiplexers, demultiplexers, then we use VHDL language and implement it on FPGA processor where as data we want to use, we compute it in c language on DSP processor.

3.5.2 Programming for parallel processing

SMT 8036 is a very powerful platform. We can do many independent tasks by parallel processing on this kit. For parallel processing, we use concept of threads. Diamond library provides a header file called thread.h. We include this file in our program if we want to include threads in our program.

Example:-

```
//header files
.
.
.
#include<thread.h>
// instructions
.
.

void update_thread(void *unused)// function with thread to calculated
{                       // output signal for bpsk modulation
int freq= 10,n,as;
while(1){
scanf("%d",&as);
if(as>=1)
{
for (n=0; n<WORDS; n++)
                        {
                        sine[n]= 100 * sin((2*n*3.14*freq)/WORDS);
                        printf("\n %f \n", sine[n]);
                        }
}
else
{
```

```
for (n=0; n<WORDS; n++)
                        {
                        sine[n]= -100 * sin((2*n*3.14*freq)/WORDS);
                        printf("\n %f \n", sine[n]);
                        }
}
}
}
//instructions
.
.
.
.
void main()
{
//instructions
.
.
.
.
THREAD_HANDLE Updater;//object for thread
Updater = Startup(update_thread);//for sending data at output port
                        while(1)
                {
                chan_out_message(BYTES, sine, &DATA1);
                }
}
```

So, in this example we are doing 2 tasks. One is to calculate data for input for bpsk and other is to send that information to output channel. We are doing this by the use of threads.

3.5.3 Designing the application code

When you use 3L Diamond to create a multiprocessor application, you follow two simple steps:

1. Divide your problem into a number of independent tasks that can communicate with each other as shown in figure 3.10, and then specify the channels that will be used to carry data between your tasks.
2. Choose the actual DSP and FPGA processors you want in your system from the list of supported modules, and then describe the hardware links they will use to intercommunicate.

3.5.4 Pipelining Implementation by Different Tasks

As described earlier also we are working on 3L diamond platform, which compiles our program and burns it on different processors. We divide our problem into a number of independent tasks that can communicate with each other as shown in figure 3.10

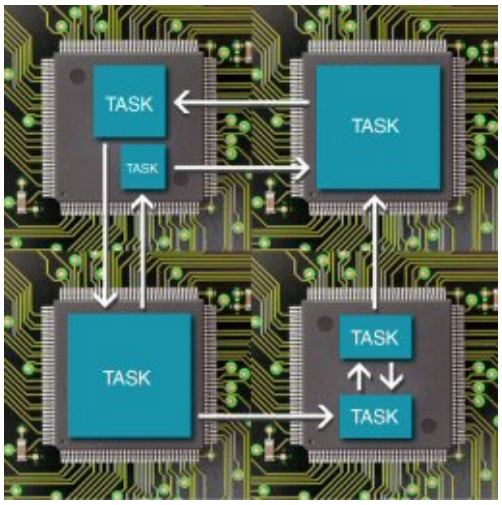


Fig 3.10 Task division [25]

and then specify the channels that will be used to carry data between our tasks. This is shown here by a simple example which takes -ve and +ve number as input and returns -5 and 5 for +ve and –ve number respectively as described in figure 3.11.

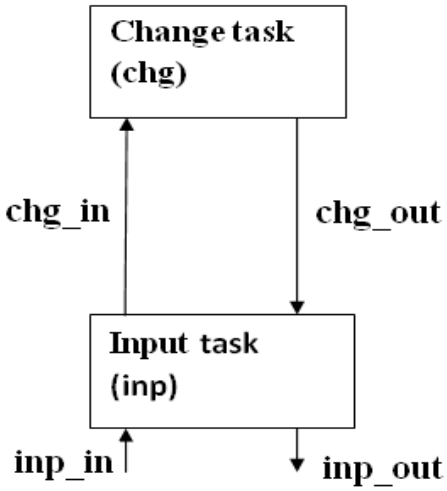


Fig 3.11 Block diagram showing Communication between tasks [5]

This is one way to achieve pipelining in DSP because the moment our input task will accept next data value (+ve or –ve number), previous value will be processed to change task this is made more clear as follows.

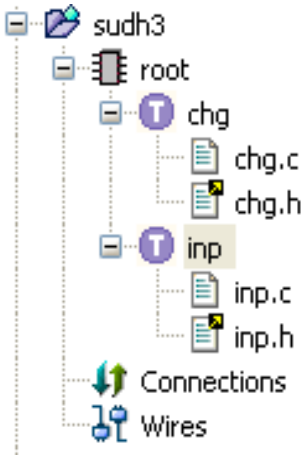


Fig 3.12 Some task names [5]

Figure 3.12 is a screen shot which shows two tasks one is “inp” which takes input and gives us final output other task is “chg” i:e change which contains the logic to change –ve number to -5 and +ve number to 5.

Code for taking “inp” is as shown in figure 3.13.

```
 int a,i;


  printf("this program is for converting +ve value to 5 and -ve value to -5 \n");
  printf("enter 5 values  \n");
  for(i=0;i<5;i++)
  {
scanf("%d",&a);
  chan_out_word(a,&inp_out);
  chan_in_word(&a,&inp_in);


printf("%d   ",a);

  }
```

Fig 3.13 Code for taking input and displaying output [5]

Code for "chg" is given in figure 3.14**.**

```
 int a;
do
{
 chan_in_word(&a,&chg_in);
  if(a>0)
     {
     a=5;
     }
   else
   {
   a=-5;
   }
 chan_out_word(a,&chg_out);
}while(a != 10) ;
```

Fig 3.14 Code for changing the values of variables [5]

Now codes for the two tasks are ready. Next step is the connection of these tasks. For this firstly we define input and output ports of tasks and then make their connections as per our requirement. For the above described example connections will be as shown in figure 3.15**.**

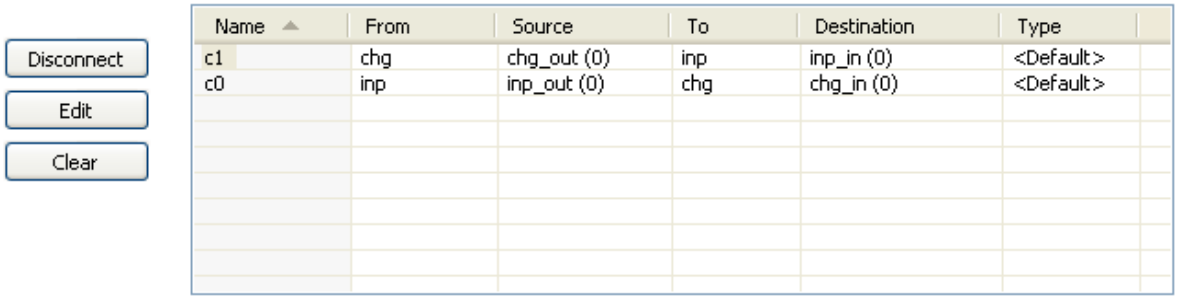


Fig 3.15 Connection between tasks [5]

Finally output for the code is given in figure 3.16

```
this program is for converting +ve value to 5 and -ve value to -5
enter 5 values
2
3
-5
6
-8
5   5   -5   5   -5
```

Fig 3.16 Output of the pipelined program [5]

3.6 DESIGN PHILOSOPHIES

Three basic design philosophies are used for programming today: linear programming (LP), OOPs and CBP.

3.6.1 Linear Programming (LP)

LP is a methodology in which the developer follows a linear thought process for the development of the code. The process follows a logical flow, so this type of programming is dominated by conditional flow control (such as "if-then" constructs) and loops. Compartmentalized functionality is maintained in functions, where execution of a function involves swapping out the stack, essentially changing the context of operation, performing the function's work, and returning results to the calling function, which requires an additional stack swap. An analogy of LP is creating a big box for all items on your desktop, such as the phone, keyboard, mouse, screen, headphone, can of soda, and picture of your attractive spouse, with no separation between these items. Accessing any one item's functionality, such

as drinking a sip of soda, requires a process to identify the soda can, isolate the soda can from the other interfering items, remove it from the box, sip it, and then place it back into the box and put the other items back where they were. C is the most popular LP language today, with assembly development reserved for a few brave souls who require truly high speed without the overhead incurred by a compiler [26].

3.6.2 Object-Oriented Programming (OOPs)

OOPs is a striking shift from LP. Whereas LP has data structures, essentially variables that contain an arbitrary composition of native types such as float or integer. OOPs extends the data structure concept to describe a whole object. An object is a collection of member variables (such as in a data structure) and functions that can operate on those member variables. From a terminology standpoint, a class is an object's type, and an object is a specific instance of a particular class. There are several rules governing the semantics of classes, but they generally allow the developer to create arbitrary levels of openness (or visibility), different scopes, different contexts, and different implementations for function calls that have the same name. OOPs has several complex dimensions; additional information can be found elsewhere.

The differences inherent in OOPs have dramatic implications for the development of software. Extending the analogy from the previous example, it is now possible to break up every item on your desktop into a separate object. Each object has some properties, such as the temperature of your soda, and each object also has some functions that you can access to perform a task on that object, such as drinking some of your soda. There are several languages today that are OOPs languages. The two most popular ones are Java and C++, although several other languages today are also OOPs languages [26].

3.6.3 Component-Based Programming (CBP)

CBP is a subtle extension of the OOPs concept. In CBP, the concept of an object is constrained; instead of allowing any arbitrary structure for the object, under CBP the basic unit is now a component. This component comprises one or more classes, and is completely defined by its interfaces and its functionality. Again extending the previous example, the contents on the desktop can now be organized into components. A component could be a

computer, where the computer component is defined as the collection of the keyboard, mouse, display, and the actual computer case. This particular computer component has two input interfaces, the keyboard and the mouse, and one output interface, the display. In future generations of this component, there could be additional interfaces, such as a set of headphones as an output interface, but the component's legacy interfaces are not affected by this new capability. Using CBP, the nature of the computer is irrelevant to the user as long as the interfaces and functionality remain the same. It is now possible to change individual objects within the component, such as the keyboard, or the whole component altogether, but the user is still able to use the computer component the same as always. The primary goal of CBP is to create stand-alone components that can be easily interchanged between implementations.

Note that CBP is a coding style, and there are no mainstream languages that are designed explicitly for CBP. Even though CBP relies on a well-defined set of interfaces and functionality, these aspects are insufficient to guarantee that the code is reusable or portable from platform to platform. The problem arises not from the concept, but from the implementation of the code. To see the problem, it is important now to consider writing the code describing the different aspects of the desktop components that we described before, in this case a computer.

Conceptually, we have a component that contains an instance of a display, keyboard, mouse, headphone, and computer. If one were to write software emulating each of these items, not only would the interfaces and actual functional specifications need to be written, but also a wide variety of housekeeping functions, including, for example, notification of failure. If any one piece of the component fails, it needs to inform the other pieces that it failed, and the other pieces need to take appropriate action to prevent further malfunctions. Such a notification is an inherent part of the whole component, and implementing changes in the messaging structure for this notification on any one piece requires the update of all other pieces that are informed of changes in state. These types of somewhat hidden relationships create a significant problem for code reuse and portability because relationships that are sometimes complex need to be verified every time that code is changed [26].

3.5.4 Limitations of conventional programming techniques

Conventional programming techniques make our system complex. So, it is very hard for the programmer to create all the codes for each and every part of software defined radio. Moreover, we have to make code in different languages e.g. C code for DSP processor and VHDL code for FPGA processor. So, whole process becomes time consuming. So, we do model based design for some of our parts of software defined radio.

CHAPTER 4

LDPC CODES & DIGITAL MODULATION TECHNIQUES

4.1 Introduction

Low Density Parity Check (LDPC) codes are one of the most promising error correction codes that are being adopted by many wireless standards.

Low Density Parity-Check (LDPC) codes are a class of linear block LDPC codes. The name comes from the characteristic of their parity-check matrix which contains only a few 1's in comparison to the amount of 0's.

Their main advantage is that they provide a performance which is very close to the capacity for a lot of different channels and linear time complex algorithms for decoding. Furthermore, they are suited for implementations that make heavy use of parallelism.

4.2 LDPC Basics

A LDPC code is a class of linear block codes whose code- words satisfy a set of linear parity-check constraints. These constraints are typically defined by an m-by-n parity check matrix H, whose m rows specify each of the m constraints (the number of parity checks), and n represents the length of a codeword. H is also characterized by W_r and W_c, which represent the number of 1's in the rows and columns, respectively.

A LDPC code can be represented by a bipartite graph, which consists of two types of nodes, Variable Nodes (VN) and Check Nodes (CN). Check node i is connected to variable node j whenever h_{ij} of H is non-zero. Fig. 1 describes the matrix H and the corresponding bipartite graph of a simple LDPC code.

In information theory, a low-density parity-check (LDPC) code is a linear error correcting code, a method of transmitting a message over a noisy transmission channel, and is constructed using a sparse bipartite graph.

The noise threshold defines an upper bound for the channel noise, up to which the probability of lost information can be made as small as desired. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length.

4.3 History of LDPC Codes

LDPC codes are also known as **Gallager codes**, in honor of Robert G. Gallager, who developed the LDPC concept in his doctoral dissertation at MIT in 1960. They were first introduced by Gallager in his PhD thesis in 1960. But due to the computational complexity in implementing encoder for such codes and the introduction of Reed-Solomon codes, they were mostly ignored until about ten years ago.

The LDPC codes were impractical to implement when first developed by Gallager in 1963. Hence, LDPC codes were forgotten, but they were rediscovered in 1996. Turbo codes, another class of capacity-approaching codes discovered in 1993, became the coding scheme of choice in the late 1990s, used for applications such as deep space satellite communications. However, in the last few years, the advances in low-density parity-check codes have seen them surpass turbo codes in terms of error floor and performance in the higher code rate range, leaving turbo codes better suited for the lower code rates.

The concatenated RS and convolution codes were considered perfectly suitable for error control coding. These were rediscovered by MacKay (1999) and Richardson / Urbanke (1998).

4.4 Applications

LDPC codes are finding increasing use in applications where reliable and highly efficient information transfer over bandwidth or return-channel constrained links in the presence of data-corrupting noise is desired. Although implementation of LDPC codes has lagged behind that of other codes, notably turbo codes, the absence of encumbering software patents has made LDPC attractive to some.

Low density parity check (LPDC) codes have excellent error correction performance that approaches the Shannon capacity limit. As a result, they have been adopted in many current and next generation wireless protocols such as DVB- S2 and the IEEE 802.16e standard (WiMAX).

In 2003, an LDPC code beat six turbo codes to become the error correcting code in the new DVB-S2 standard for the satellite transmission of digital television. In 2008, LDPC beat convolution turbo codes as the FEC scheme for the ITU-T G.hn standard. G.hn chose LDPC over turbo codes because of its lower decoding complexity (especially when operating at data rates close to 1 Gbit/s) and because the proposed turbo codes exhibited a significant error floor at the desired range of operation. LDPC is also used for 10GBase-T Ethernet, which sends data at 10 gigabits per second over twisted-pair cables. As of 2009, LDPC codes are also part of the Wi-Fi 802.11 standard as an optional part of 802.11n, in the High Throughput (HT) PHY specification.

4.5 Representations for LDPC codes [27]

Basically there are two different possibilities to represent LDPC codes. Like all linear block codes they can be described via matrices. The second possibility is a graphical representation.

4.5.1 Matrix Representation

Here is an example for a low-density parity-check matrix first. The matrix defined in equation (1) is a parity check matrix with dimension $n \times m$ for a (8, 4) code. We can now define two numbers describing this matrix. w_r for the number of 1's in each row and w_c for the columns. For a matrix to be called low-density, the two conditions

i) $w_c << n$ and

ii) $w_r << m$

must be satisfied. In order to do this, the parity check matrix should usually be very large, so the example matrix can't be really called low-density.

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

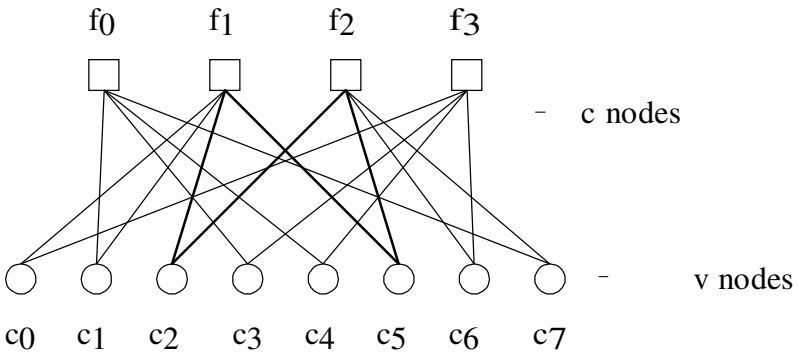


Fig 4.1 Tanner graph corresponding to parity check matrix in above equation [27]

The marked path $c_2 \rightarrow f_1 \rightarrow c_5 \rightarrow f_2 \rightarrow c_2$ is an example for a short cycle. Those should usually be avoided since they are bad for decoding performance.

4.5.2 Graphical Representation

Tanner introduced an effective graphical representation for LDPC codes. Not only provide these graphs a complete representation of the code, they also help to describe the decoding algorithm as explained later on in this tutorial.

Tanner graphs are bipartite graphs. That means that the nodes of the graph are separated into two distinctive sets and edges are only connecting nodes of two different types. The two types of nodes in a Tanner graph are called variable nodes (v-nodes) and check nodes (c-nodes). Figure 1.1 is an example for such a Tanner graph and represents the same code as the matrix in 1. The creation of such a graph is rather straight forward. It consists of m check nodes (the number of parity bits) and n variable nodes (the number of bits in a codeword). Check node f_i is connected to variable node c_j if the element h_{ij} of H is a 1.

4.6 Regular and irregular LDPC codes

A LDPC code is called regular if w_c is constant for every column and $w_r = w_c \cdot (n/m)$ is also constant for every row. The example matrix from equation (1) is regular with $w_c = 2$ and $w_r = 4$. It's also possible to see the regularity of this code while looking at the graphical representation. There is the same number of incoming edges for every v-node and also for all the c-nodes. If H is low density but the numbers of 1's in each row or column aren't constant the code is called an irregular LDPC code.

4.7 Constructing LDPC codes

Several different algorithms exist to construct suitable LDPC codes. Gallager himself introduced one. Furthermore MacKay proposed one to semi-randomly generate sparse parity check matrices. This is quite interesting since it indicates that constructing good performing LDPC codes is not a hard problem. In fact, completely randomly chosen codes are good with a high probability. The problem that will arise is that the encoding complexity of such codes is usually rather high.

4.8 Fundamentals of Linear Block Codes

• The structure of a code is completely described by the generator matrix G or the parity check matrix H.

• The capacity of correcting symbol errors in a codeword is determined by the minimum distance (d_{min}).

– d_{min} is the least weight of the rows in G.

– d_{min} is the least number of columns in H that sum up to 0.

– Example: (7, 4) Hamming code

$$G = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{matrix}$$

$$H = \begin{matrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{matrix}$$

4.9 Properties of LDPC Codes

• H is sparse.

– Very few 1's in each row and column.

– Expected large minimum distance.

• Regular LDPC codes

– H contains exactly Wc 1's per column and exactly $W_r = W_c (n/m)$ 1's per row, where $W_c << m$.

– The above definition implies that $W_r << n$.

– Wc ≥ 3 is necessary for good codes.

• If the number of 1's per column or row is not constant, the code is an irregular LDPC code.

– Usually irregular LDPC codes outperform regular LDPC codes.

4.10 A Sample LDPC Code

$W_c=3$

Any two columns have an overlap of at most one 1. The sparse property allows us to avoid overlapping.

In the part of H shown above, there does not exist a set of columns that add up to 0. The above facts make the d_{min} large.

G is found by Gaussian elimination, i.e.

-H can be put in the form $H = [P^T: I]$

-The generator matrix $G = [I: P]$

4.10.1 Encoding of LDPC Codes

• General encoding of systematic linear block codes

$$c = xG = [x: xP]$$

4.10.2 Issues with LDPC codes

– The size of G is very large.

– G is not generally sparse.

– Example: A (10000, 5000) LDPC code. $\mathbf{P}$ is 5000×5000.

We may assume that the density of 1's in $\mathbf{P}$ is 0.5 and there are 12.5×10^6 1's in $\mathbf{P}$.

12.5×10^6 addition (XOR) operations are required to encode one codeword.

• An alternative approach to simplified encoding is to design the LDPC code via algebraic or geometric methods.

– Such "structured" codes can be encoded with shift register circuits.

4.10.3 Iterative Decoding of LDPC codes

• General decoding of linear block codes

– Only if c is a valid codeword, we have $cH^T = 0$

– For binary symmetric channel (BSC), the received codeword is c added with an error vector e.

– The decoder needs to find out e and flip the corresponding bits.

– The decoding algorithm is based on linear algebra.

• Graph-based algorithms

– Sum-product algorithm for general graph-based codes;

– MAP (BCJR) algorithm for trellis graph-based codes;

– Message passing algorithm for bipartite graph-based codes.

4.10.4 Decoding LDPC codes

The algorithm used to decode LDPC codes was discovered independently several times and as a matter of fact comes under different names. The most common ones are the belief propagation algorithm, the message passing algorithm and the sum-product algorithm. The

algorithm will be explained on the basis of the example code already introduced in equation 1 and figure 1.1. An error free received codeword would be e.g.

$$c = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 1].$$

Let's suppose that the received codeword is having one error bit c_1 flipped to 1.

1. In the first step all v-nodes c_i send a "message" to their (always 2 in our example) c-nodes f_j containing the bit they believe to be the correct one for them. At this stage the only information a v-node c_i has, is the corresponding received i-th bit of c, y_i. That means for example, that c_0 sends a message containing 1 to f_1 and f_3, node c_1 sends messages containing y_1 (1) to f_0 and f_1, and so on.

c-node	received/sent			
f_0	received:	$c_1 \rightarrow 1$	$c_3 \rightarrow 1$	$c_4 \rightarrow 0$ $c_7 \rightarrow 1$
	sent:	$0 \rightarrow c_1$	$0 \rightarrow c_3$	$1 \rightarrow c_4$ $0 \rightarrow c_7$
f_1	received:	$c_0 \rightarrow 1$	$c_1 \rightarrow 1$	$c_2 \rightarrow 0$ $c_5 \rightarrow 1$
	sent:	$0 \rightarrow c_0$	$0 \rightarrow c_1$	$1 \rightarrow c_2$ $0 \rightarrow c_5$
f_2	received:	$c_2 \rightarrow 0$	$c_5 \rightarrow 1$	$c_6 \rightarrow 0$ $c_7 \rightarrow 1$
	sent:	$0 \rightarrow c_2$	$1 \rightarrow c_5$	$0 \rightarrow c_6$ $1 \rightarrow c_7$
f_3	received:	$c_0 \rightarrow 1$	$c_3 \rightarrow 1$	$c_4 \rightarrow 0$ $c_6 \rightarrow 0$
	sent:	$1 \rightarrow c_0$	$1 \rightarrow c_3$	$0 \rightarrow c_4$ $0 \rightarrow c_6$

Table 4.1 Overview over messages received and sent by the c-nodes in step 2 of the message passing algorithm

2. In the second step every check nodes f_j calculate a response to every connected variable node. The response message contains the bit that f_j believes to be the correct one for this v-node c_i assuming that the other v-nodes connected to f_j are correct. In other words: If you look at the example, every c-node f_j is connected to 4 v-nodes. So a c-node f_j looks at the message received from three v-nodes and calculates the bit that the fourth v-node should have in order to fulfill the parity check equation. Table 2 gives an overview about this step.

Important is, that this might also be the point at which the de- coding algorithm

terminates. This will be the case if all check equations are fulfilled. We will later see that the whole algorithm contains a loop, so another possibility to stop would be a threshold for the amount of loops.

3. Next phase: the v-nodes receive the messages from the check nodes and use this additional information to decide if their originally received bit is OK. A simple way to do this is a majority vote. When coming back to our example that means, that each v-node has three sources of information concerning its bit. The original bit received and two suggestions from the check nodes. Table 3 illustrates this step. Now the v-nodes can send another message with their (hard) decision for the correct value to the check nodes.

4. Go to step 2.

v-node	y_i received	messages from check nodes		decision
c_0	1	$f_1 \rightarrow 0$	$f_3 \rightarrow 1$	1
c_1	1	$f_0 \rightarrow 0$	$f_1 \rightarrow 0$	0
c_2	0	$f_1 \rightarrow 1$	$f_2 \rightarrow 0$	0
c_3	1	$f_0 \rightarrow 0$	$f_3 \rightarrow 1$	1
c_4	0	$f_0 \rightarrow 1$	$f_3 \rightarrow 0$	0
c_5	1	$f_1 \rightarrow 0$	$f_2 \rightarrow 1$	1
c_6	0	$f_2 \rightarrow 0$	$f_3 \rightarrow 0$	0
c_7	1	$f_0 \rightarrow 1$	$f_2 \rightarrow 1$	1

Table 4.2 Step 3 of the described decoding algorithm. The v-nodes use the answer messages from the c-nodes to perform a majority vote on the bit value.

In our example, the second execution of step 2 would terminate the decoding process since c_1 has voted for 0 in the last step. This corrects the transmission error and all check equations are now satisfied [27].

4.11 DIGITAL MODULATION SCHEMES

Modulation is a process by which a carrier signal is altered according to instantaneous amplitude of information signal and it is the modulated signal that is transmitted. The receiver then recovers the original signal through a process called demodulation. The carrier frequency, denoted F_c, is the frequency of the carrier signal. The sampling rate, F_s, is the rate at which the message signal is sampled during the simulation.

The frequency of the carrier signal is usually much greater than the highest frequency of the input message signal. The Nyquist sampling theorem requires that the simulation sampling rate F_s be greater than two times the sum of the carrier frequency and the highest frequency of the modulated signal, in order for the demodulator to recover the message correctly. Following are the basic Digital Modulation Techniques

a) Amplitude shift keying
b) Frequency shift keying
c) Phase shift keying [5]

4.11.1 Amplitude Shift Key (ASK) Modulation

In this method the amplitude of the carrier assumes one of the two amplitudes dependent on the logic states of the input bit stream. A typical output waveform of an ASK modulator is shown in Fig. 4.2.

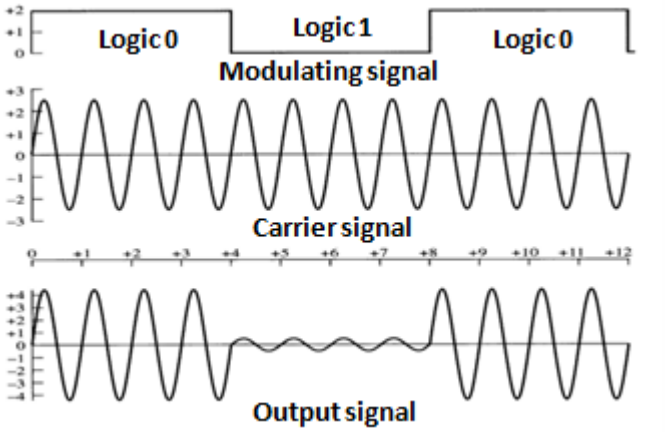


Fig 4.2 Output waveform of an ASK modulator [28]

A binary amplitude-shift keying (BASK) signal can be defined by

$s(t) = A\, m(t) \cos 2\pi f_c t, \qquad 0 < t < T$

where A is a constant,

m(t) = 1 or 0,

f_c is the carrier frequency, and

T is the bit duration.

The effect of multiplication by the carrier signal $A\cos 2\pi f_c t$ is simply to shift the spectrum of the modulating signal m(t) to f_c. As mentioned in the ASK equation there are two constellation points for the ASK. These constellation points are shown in figure 4.3

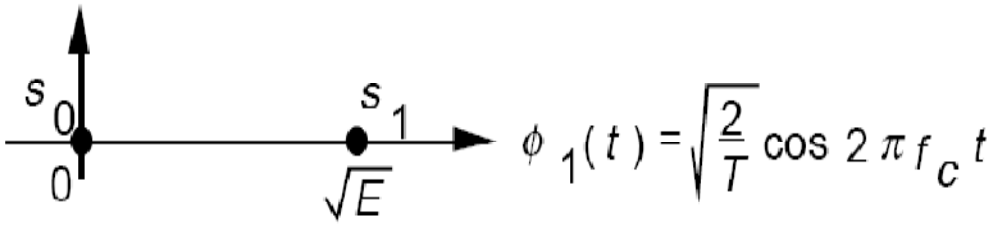


Fig 4.3 Constellation points for ASK [28]

ASK has simple modulator circuit it simply multiplies the binary information signal with carrier signal and generates the output signal. At the receiver, the received signal is multiplied by a local generated carrier and output is passed from a filter having center frequency same as we are using to operate the communication device. At the output we got a DC component based upon it we take decision weather received signal is 0 or 1 [28].

4.11.2 Phase Shift Keying (PSK) and M-ary PSK

Phase shift keying (PSK) is a large class of digital modulation schemes. PSK is widely used in the communication industry. In this chapter we study each PSK modulation scheme where modulator/demodulator block diagrams are included. This chapter also involves M-ARY PSK. The motivation behind MPSK is to increase the bandwidth efficiency of the PSK modulation schemes. In BPSK, a data bit is represented by a symbol. In MPSK, $n = \log_2 M$ data bits are represented by a symbol, thus the bandwidth efficiency is increased to n times.

There are several different types of M-ARY Phase Shift Key (PSK) modulators. These are:

• Two-phase (2-PSK)

• Four-phase (4-PSK)

• Eight-phase (8-PSK)

• Sixteen-phase (16-PSK)

First we present binary PSK (BPSK). Then we discuss M-ary PSK (MPSK) i:e quadrature PSK (QPSK) and 8-PSK .

4.11.3 Binary Phase Shift Keying (BPSK) Modulation

Binary data are represented by two signals with different phases in BPSK. Typically these two phases are 0 and π, the signals are

$s_1(t) = A \cos 2\pi f_c t, \quad 0 \le t \le T$, for 1

$s_2(t) = -A \cos 2\pi f_c t, \quad 0 \le t \le T$, for 0

The reason that they are chosen is that they have a correlation coefficient of −1, which leads to the minimum error probability for the same E_b/N_o, as we will see shortly. These two signals have the same frequency and energy.

BPSK signals can be graphically represented by a signal constellation diagram as shown in fig 4.4.

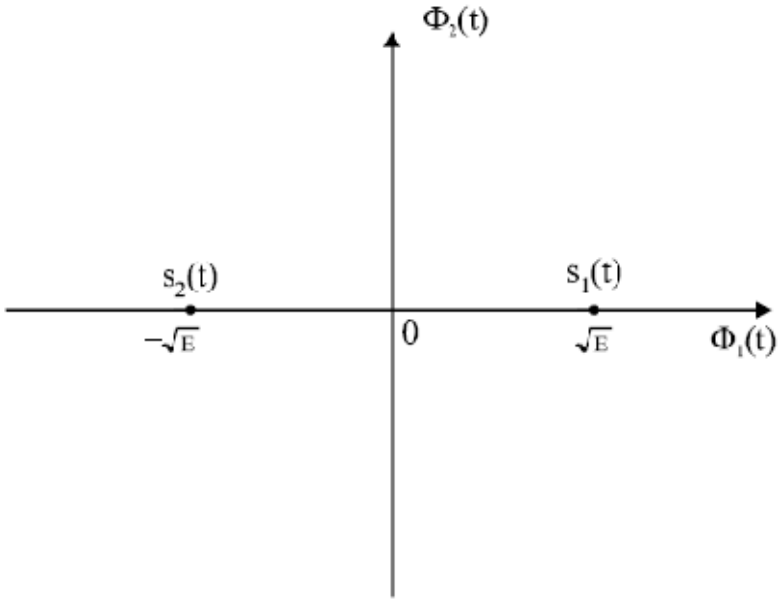


Fig 4.4 BPSK signal constellation [28]

A two-dimensional coordinate system with

$\Phi_1(t) = \sqrt{2/T}\cos(2\pi f_c t)$, $0 \le t \le T$

and

$\Phi_2(t) = -\sqrt{2/T}\sin(2\pi f_c t)$, $0 \le t \le T$

as its horizontal and vertical axis, respectively. The waveform of a BPSK signal is shown in Figure 4.5 and figure 4.6.

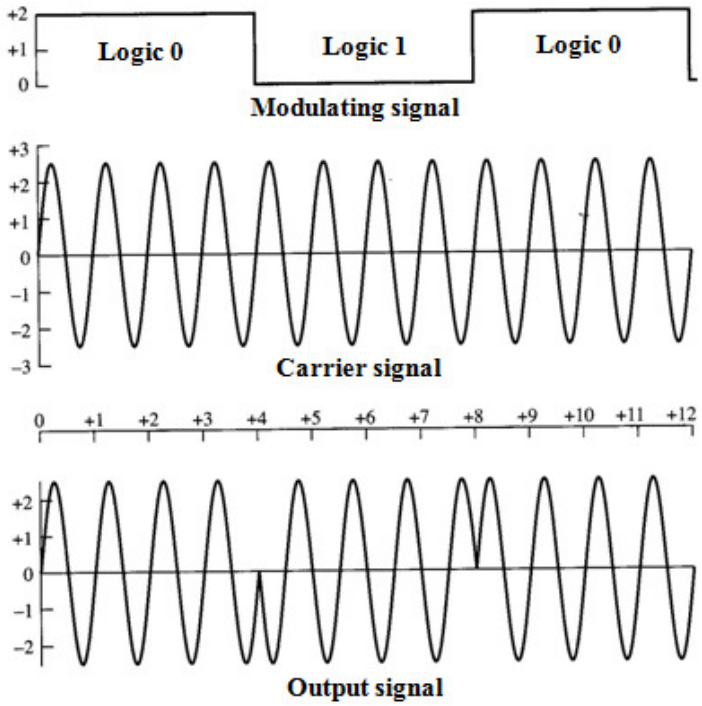


Fig 4.5 BPSK output waveform if $f_c = m\,R_b = m/T$ [28]

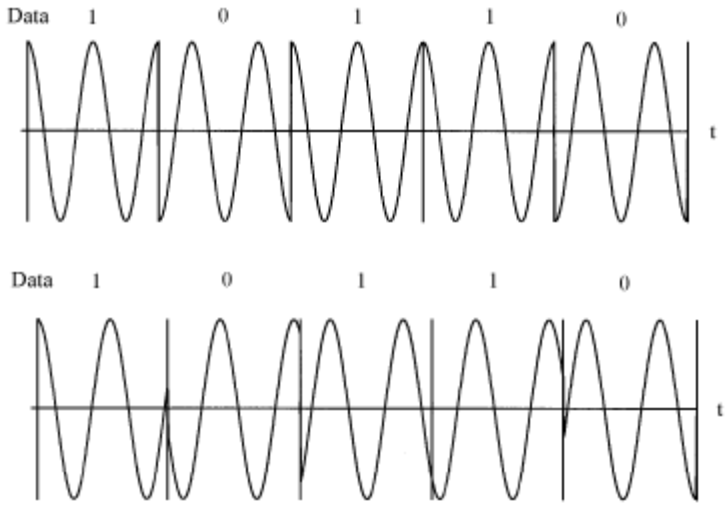


Fig 4.6 BPSK output waveform if $f_c \neq m\,R_b \neq m/T$ [28]

If the $f_c = m R_b = m/T$, where m is an integer and R_b is the data bit rate, and the bit timing is synchronous with the carrier, then the initial phase at a bit boundary is either 0 or π (Figure 4.5), corresponding to data bit 1 or 0.

However, if the f_c is not an integer multiple of R_b, the initial phase at a bit boundary is neither 0 nor π (Figure 4.6)**.** In other words, the modulated signals are not the ones given in BPSK equations. [28]

4.11.4 Quaternary Phase-Shift Keying (QPSK) Modulation

Among all MPSK schemes, QPSK is the most often used scheme since it does not suffer from BER degradation while the bandwidth efficiency is increased. Other MPSK schemes increase bandwidth efficiency at the expenses of BER performance.

Quaternary Phase-Shift Keying is sometime called another form of angle modulated, constant amplitude digital modulation .QPSK is an M-ary encoding scheme where n=2 and M=4 (hence the meaning Quaternary meaning 4). With QPSK four output phases are possible for a single carrier frequency. Because there are four output phases there must be four input conditions. Because the digital input to the QPSK modulator is binary signal, to produce four different input conditions the modulator requires more than a single bit to determine the output condition. With two input bits there are four possible combinations i:e 00,01,10,11. Therefore in QPSK the binary input data is combined in group of two bits called dibits. In modulator each dibit code generates one of the four possible output phases ($+45^o$, $+135^o$, -45^o, -135^o). Therefore, for each two bit dibit clocked into the modulator, a single output change occurs,and the rate of change of output is half the rate of change of bits at input.

In Figure 4.8 it can be seen that with QPSK each of the four possible output phasors has exactly the same amplitude. Therefore, the binary information must be encoded entirely in the phase of output signal. This constant amplitude characteristic is most important characteristic of PSK. The angular separation between any two adjacent phasor in QPSK is 90^o. The truth table 4.3 and waveforms supporting above is presented here in fig 4.8

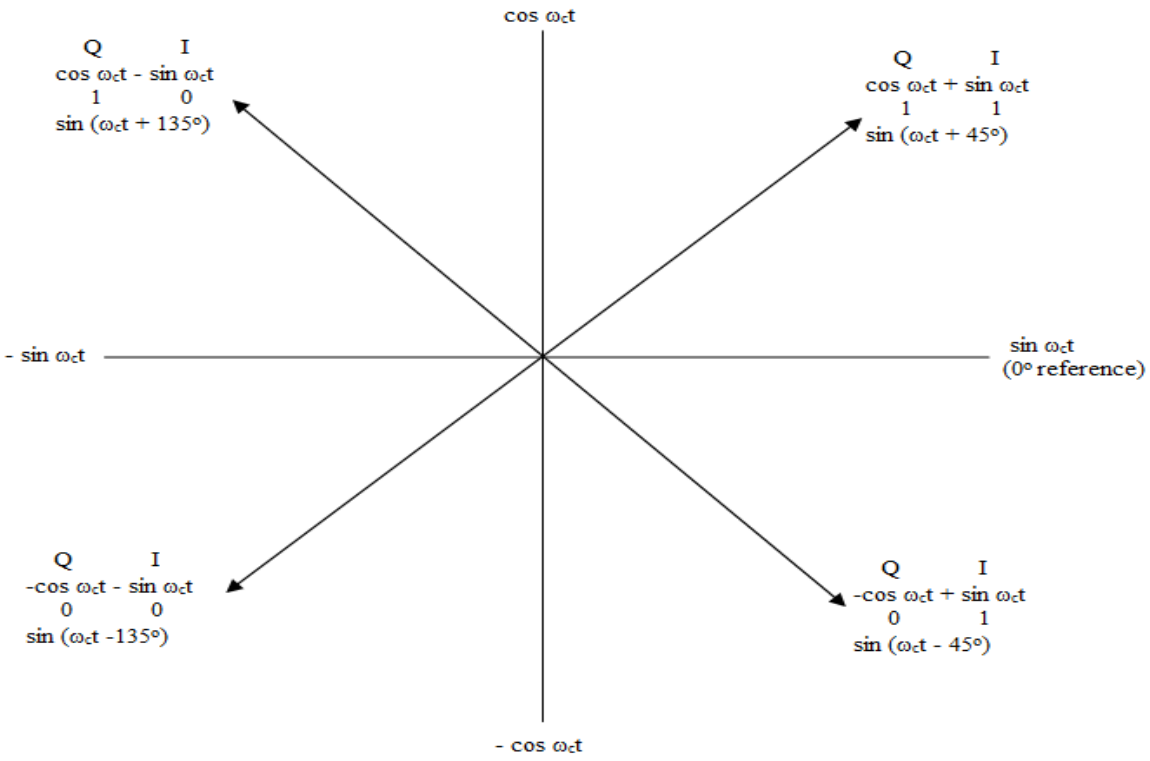


Fig 4.7 Constellation diagram for QPSK [29]

Binary Input		QPSK Output phase (degree)
Q	I	
0	0	-135^o
0	1	-45^o
1	0	135^o
1	1	45^o

Table 4.3 Binary Input vs QPSK Output phase

Thus a QPSK signal can undergo almost a $+45^o$ or -45^o shift in phase during transmission and still retain the correct encoded information when demodulated at the receiver.

The received QPSK signal ($-\sin w_c t + \cos w_c t$) is one of the inputs to the product detector. The output is recovered carrier ($\sin w_c t$). The output of I product detector is

$I = (-\sin w_c t + \cos w_c t)(\sin w_c t)$

$= (-\sin w_c t)(\sin w_c t) + (\cos w_c t)(\sin w_c t)$

$= -(\sin w_c t)^2 + (\cos w_c t)(\sin w_c t)$

$=-1/2(1-\cos 2w_ct) +1/2 \sin(w_c-w_c)t+1/2 \sin(w_c+w_c)t$

$=-1/2 + 1/2 \cos 2w_ct + 1/2 \sin 2w_ct +1/2 \sin 0$

$=-1/2$ V(logic 0)

In Q product detector received signal is multiplied with cos w_ct so

$Q= (-\sin w_ct +\cos w_ct)(\sin w_ct)$

$= (\cos w_ct)^2 + (\sin w_ct) (\cos w_ct)$

$= 1/2(1+ \cos 2w_ct) -1/2 \sin(w_c+w_c)t - 1/2 \sin(w_c-w_c)t$

$= 1/2 + 1/2 \cos 2w_ct - 1/2 \sin 2w_ct -1/2 \sin(0)$

$= 1/2$ V(logic 1)

So, demodulated I and Q bits were 0 and 1 respectively [29].

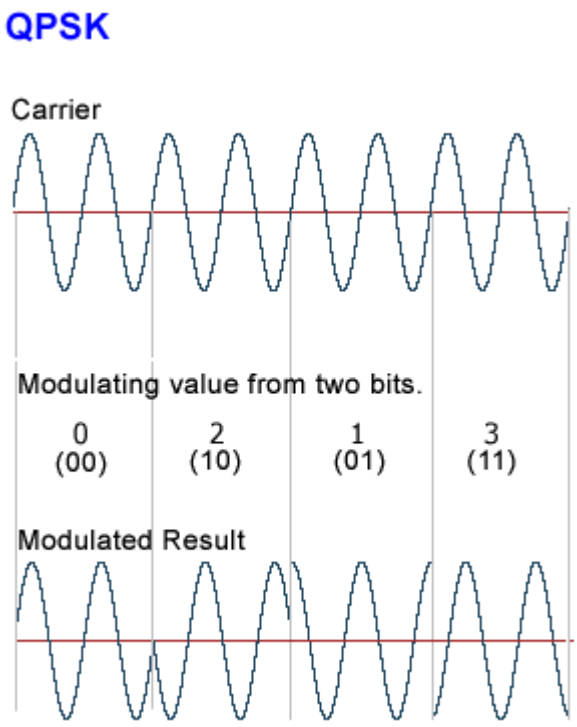


Fig 4.8 QPSK output waveforms [29]

4.11.5 8-PSK

With 8-PSK, Three bits are encoded, forming tribit and producing eight different output phases. 8-PSK, n=3, M=8 and there are 8 different output phases. To encode 8 different phases incoming bits are encoded in group of three, called tribit (2^3=8).

From constellation diagram shown in figure 4.9 it is clear that angular separation between any two adjacent phasors is 45°, half what is with QPSK.

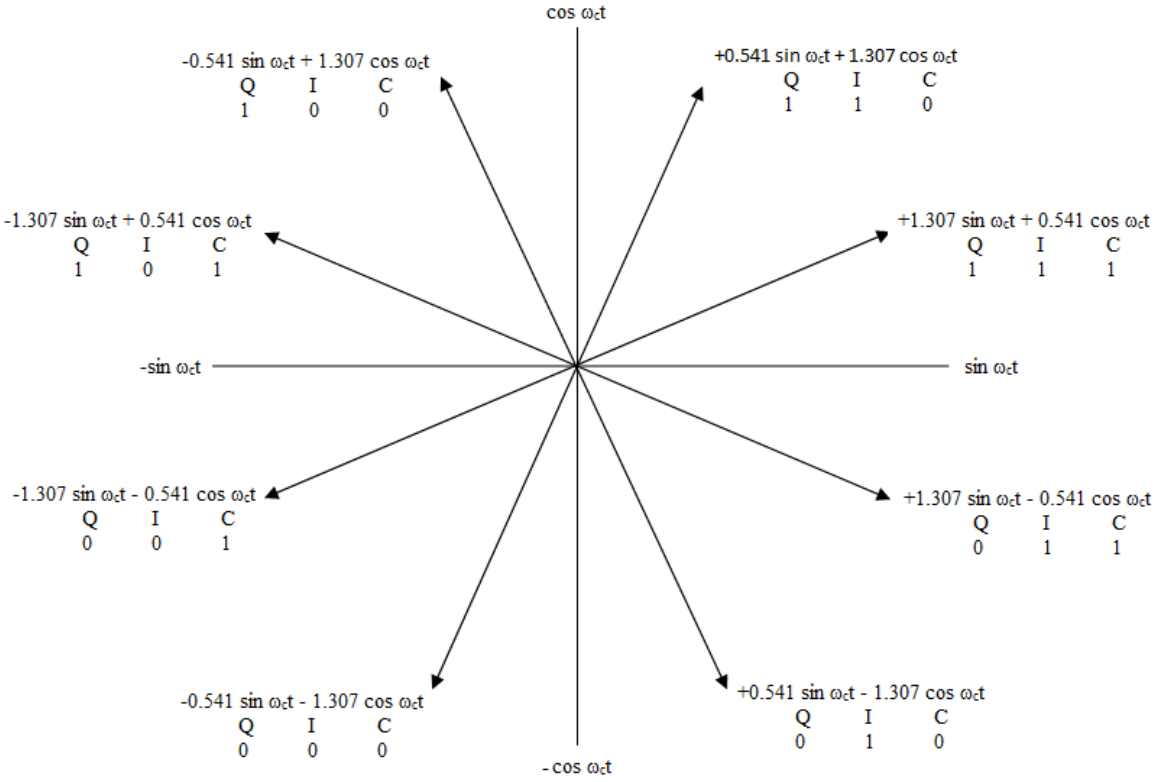


Fig 4.9 Constellation diagram for 8-PSK [29]

Therefore, an 8-PSK signal can undergo almost 22.5° phase shift in positive or negative direction during transmission and still retain its integrity. 8-PSK output phases for every Tribit input is shown in figure 4.10.

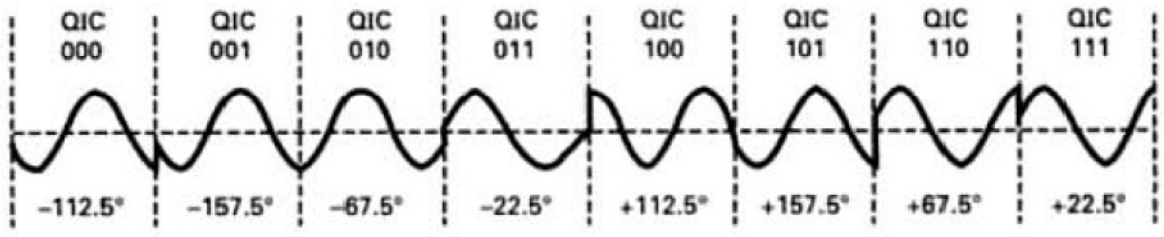


Fig. 4.10: Output phases for 8-PSK [29]

Also each phasor is of equal magnitude. The truth table showing values of output phases is showing in table 4.4. It should also be noted that tribit code between any two adjacent phases changes by only 1 bit. This type of code is gray code or sometimes called maximum distance code. This code is used to reduce the number of transmission errors. If a signal were to undergo a phase shift during transmission, it would most likely to be shifted to an adjacent phasor. Using the gray code results in only a single bit being received in error [29].

Binary Input			8-psk output phase
Q	I	C	
0	0	0	-112.5°
0	0	1	-157.5°
0	1	0	-67.5°
0	1	1	-22.5°
1	0	0	112.5°
1	0	1	157.5°
1	1	0	67.5°
1	1	1	22.5°

Table 4.4 Binary Input vs 8-psk output phase

4.11.6 Quadrature Amplitude Modulation (QAM) Modulation

A single attribute of the carrier is used in ASK (amplitude) and PSK (phase) to convey the information. Naturally, the next step is to consider using both amplitude and phase modulations in a scheme. This leads to the concept of QAM. Thus QAM is a modulation technique that employs both phase modulation (PM) and amplitude modulation (AM).

It is widely used to transmit digital signals such as digital cable TV and cable Internet service, QAM is also used as the modulation technique in orthogonal frequency division multiplexing. The "quadrature" comes from the fact that the phase modulation states are 90 degrees apart from each other.

Analog QAM

Analog QAM uses two carriers 90 degrees out of phase with each other. Each carrier is modulated by an analog signal, and the resulting modulated waves are combined.

Digital QAM

In digital QAM, the input stream is divided into groups of bits based on the number of modulation states used. For example, in 8QAM, each three bits of input, which provides eight values (0-7) alters the phase and amplitude of the carrier to derive eight unique modulation states. In 64QAM, each six bits generates 64 modulation states; in 128QAM, each seven bits generates 128 states, and so on [30].

Amplitude (distance of point from origin) and phase (angle of line from point to origin) modulation.

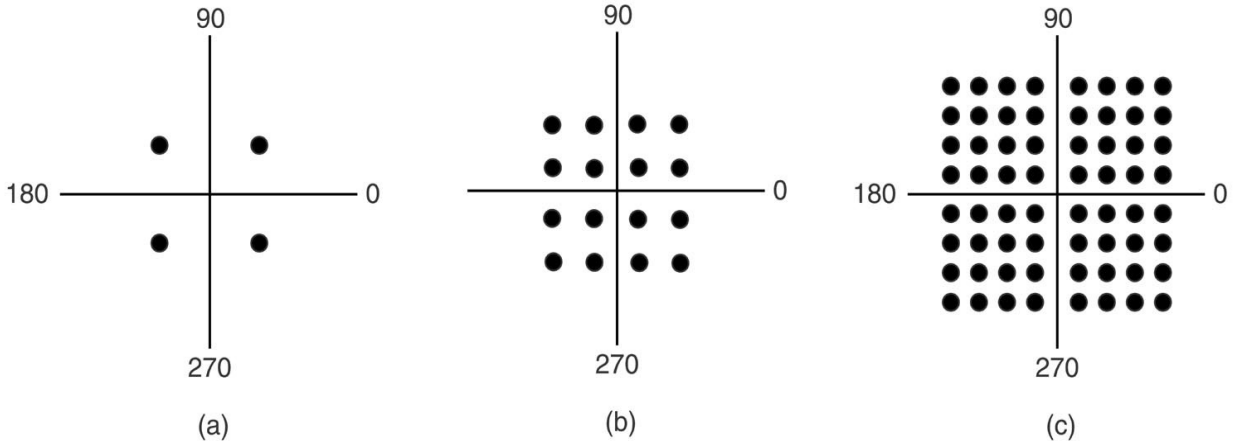


Fig 4.11 Constellation Diagrams for QPSK, QAM-16 and QAM-64 [30]

(a) QPSK - 4 phase shifts

- 4 symbols = 2^2
- bps = 2 .baud
- 4800 bps on 2400 baud

(b) QAM-16.

- 16 symbols = 2^4
- bps = 4 .baud
- 9600 bps on 2400 baud

(c) QAM-64.

- 64 symbols = 2^6
- bps = 6 .baud
- 14.4 kbps on 2400 baud

Analog QAM modulates two carriers 90 degrees out of phase with each from two analog input streams. The modulated carriers are combined and transmitted.

A particular QAM signal can be written as

$S_i(t)=A_i\cos(2\pi f_c t+\theta_i)$, i=1 , 2, ...M

where A_i is the amplitude and θ_i is the phase of the ith signal in the M-ary signal set.

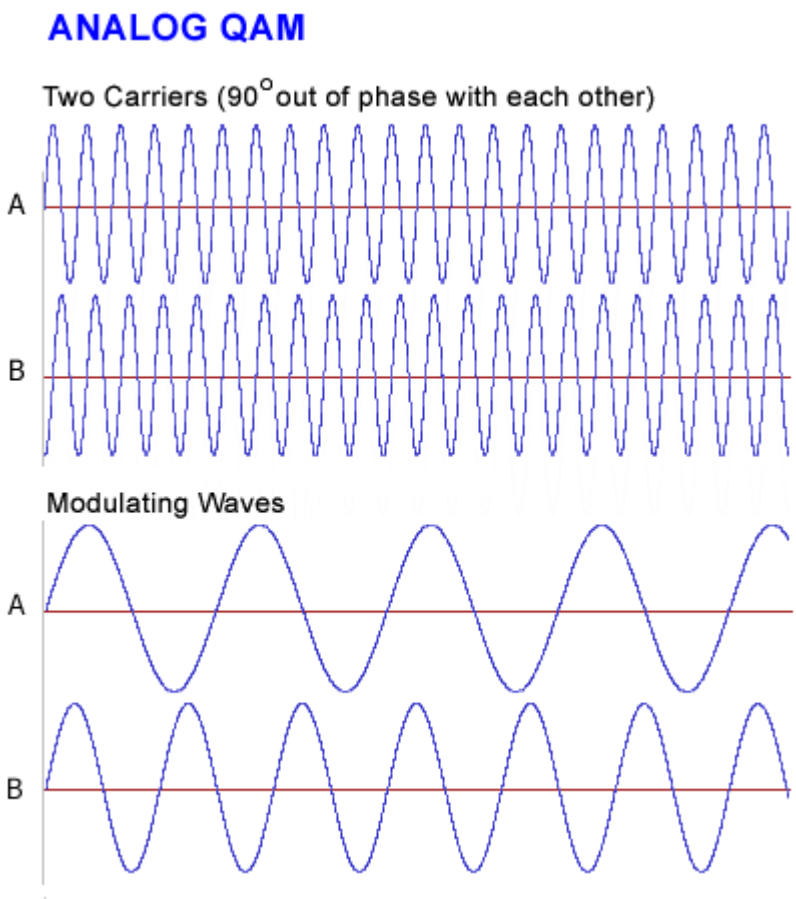


Fig 4.12 Analog QAM [30]

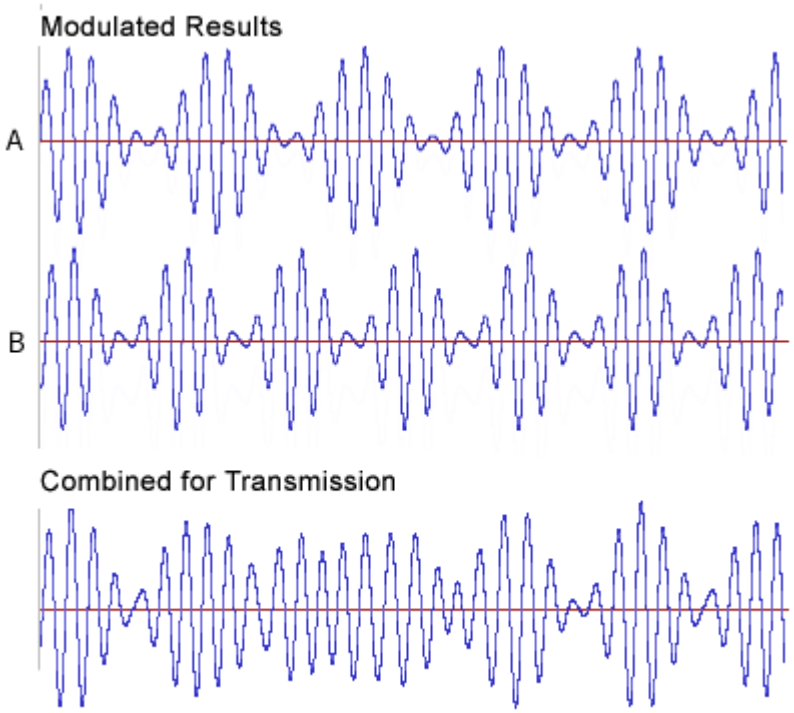


Fig 4.12 Analog QAM (contd.) [30]

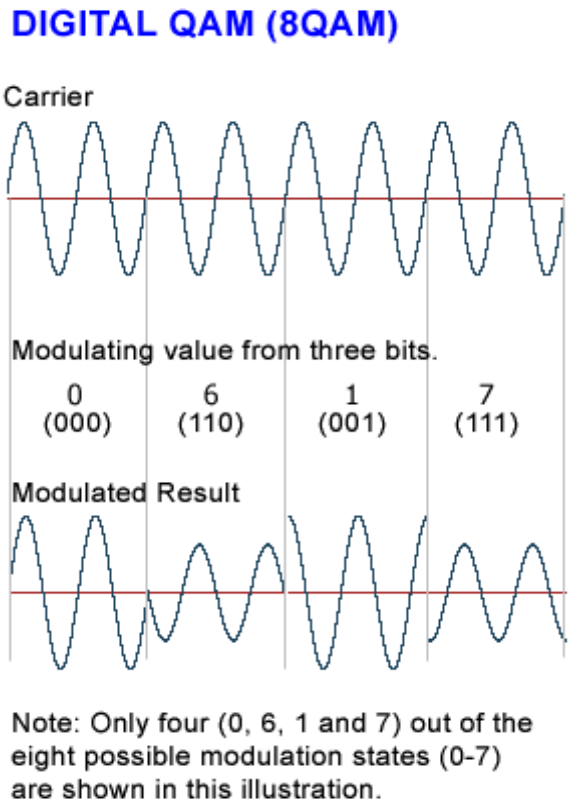


Fig 4.13 Digital QAM [30]

In this 8-QAM example, three bits of input generate eight different modulation states (0-7) using four phase angles on 90 degree boundaries and two amplitudes: one at 50% modulation; the other at 100% (4 phases X 2 amplitudes = 8 modulation states). QAM examples with more modulation states become extremely difficult to visualize [30].

Sample for transmitting a signal for a bit stream:

QAM is simply a combination of amplitude modulation and phase shift keying. We' ll use a signal that is transmitting at 3600 bps, or 3 bits per baud. This means that we can represent 8 binary combinations.

We' ll use 2 measures of amplitude, i.e. 1 and 2. We'll also use 4 possible phase shifts. Thus, combining these two, we have 8 possible waves that we can send.

First step is to generate a table to show us which waves correspond to which binary combination. This can basically be done at random.

Bit value	Amplitude	Phase shift
000	1	None
001	2	None
010	1	1/4
011	2	1/4
100	1	1/2
101	2	1/2
110	1	3/4
111	2	3/4

Table 4.5 Binary Input vs QAM output phase shift

Let's encode a big bit stream:

001010100

011101000

011110

First, we break it up into 3-bit triads:

001-010-100-

011-101-000-

011-110

Now, we shift each wave relative to the wave before it.

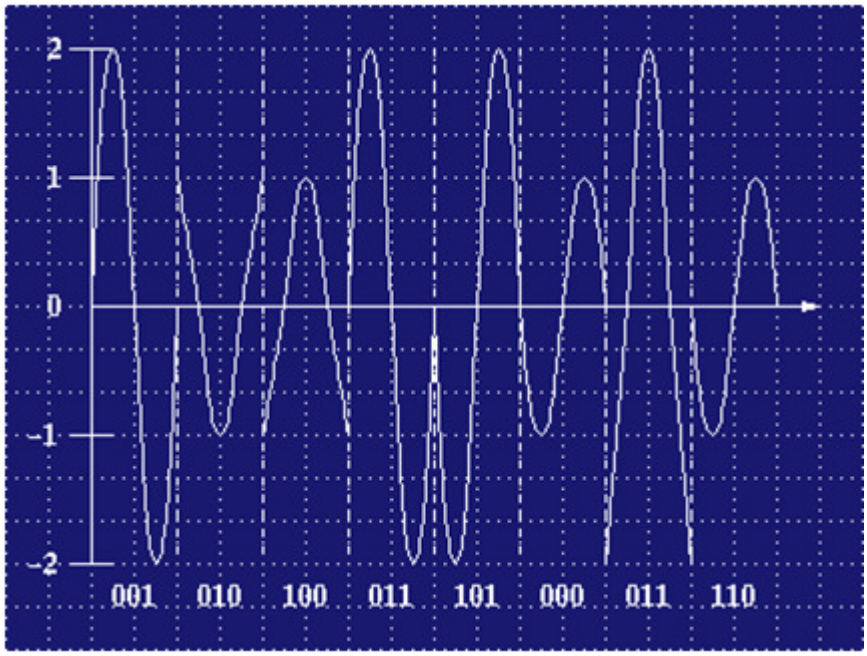


Fig 4.14 QAM modulated signal in accordance with the bit stream

Thus, we have this signal to be transmitted after the QAM modulation of the bit stream.

Chapter 5

WOK DONE AND RESULTS

5.1 Work Done

5.1.1 Implementation of BPSK on SMT-8036

In software defined radio we make baseband signals with coding (either in C or in VHDL). For BPSK, we send a sine signal for logic 1 and a cosine signal for logic 0. Logic for this is as given in Figure 5.1

```
if(doWhat == 1)
    {
    for (i=0; i<PatternCount; i++)
        {
        if (data == 1)
            {
            sineWave = 2*cos(2 * PI * N * ( ( (float)i ) / PatternCount ));
            }
        else
            {
            sineWave = 3*cos(2 * PI * N * ( ( (float)i ) / PatternCount )-(PI/4));
            }
        a = (unsigned short) ( fA * 1023 * sineWave );
        b = a;
        Pattern[i] = (a << 16) | (b);
        }
    }
```

Fig 5.1 Logic for BPSK at Transmitter

Values detected after digitalizing the signal is as shown in figure 5.2

```
Data = 1

  8190.000000  8191.000000  8193.000000  8197.000000  8195.000000  8189.000000  8191.000000  8194.000000  8197.000000  819

Data = 0

  8187.000000  8182.000000  8184.000000  8186.000000  8187.000000  8188.000000  8183.000000  8185.000000  8187.000000  818

Data = 1

  8192.000000  8195.000000  8195.000000  8194.000000  8194.000000  8191.000000  8195.000000  8198.000000  8193.000000  819

Data = 0

  8190.000000  8189.000000  8186.000000  8185.000000  8188.000000  8192.000000  8188.000000  8187.000000  8184.000000  818

Data = 1

  8191.000000  8194.000000  8198.000000  8198.000000  8189.000000  8192.000000  8193.000000  8197.000000  8195.000000  818

Data = 0

  8184.000000  8187.000000  8186.000000  8188.000000  8190.000000  8184.000000  8185.000000  8188.000000  8187.000000  819

Data = 1

  8186.000000  8189.000000  8192.000000  8194.000000  8197.000000  8193.000000  8193.000000  8192.000000  8194.000000  819

Data = 0

  8187.000000  8189.000000  8188.000000  8187.000000  8186.000000  8188.000000  8189.000000  8190.000000  8187.000000  818
```

Fig 5.2 Values for BPSK at receiver

And at receiver side, logic used is as shown in fig 5.3

```
int DetectBPSK()
{
    int f;
    printf("  stream2 is ");
    for ( f=0;f<200;f++)
        {
        printf("  %f",stream2[f]);
        }
    printf("  \n stream2 ends ");
    if (stream2[1]>8195)
        {
        return 0;
        }
    else
        {
        return 1;
        }
}
```

Fig 5.3 Logic for detecting BPSK at receiver

5.1.2 Implementation of QPSK on SMT-8036

In QPSK we send unique cosine signals with a phase difference of π/4 or as the case may be. This is shown in Fig 5.4

```
if(doWhat == 1)
    {
    for (i=0; i<PatternCount; i++)
        {
        if (data < 1)
        {
        sineWave = 2*cos(2 * PI * N * ( ( (float)i ) / PatternCount ));
        }
        else if (data < 2)
        {

        sineWave = cos(2 * PI * N * ( ( (float)i ) / PatternCount )+(PI/4));
        }
        else if (data < 3)
        {

        sineWave = cos(2 * PI * N * ( ( (float)i ) / PatternCount ) )-10;
        }
        else if (data >= 3)
        {
        sineWave = 3*cos(2 * PI * N * ( ( (float)i ) / PatternCount )-(PI/4));
        }


        a = (unsigned short) ( fA * 1023 * sineWave );
        b = a;
        Pattern[i] = (a << 16) | (b);
        }
    }
```

Fig 5.4 Logic for QPSK at the Transmitter

At the receiver, code for detecting these values for QPSK is as shown as under:

```
if (p%200==0)
        {
        for(i=0;i<100;i++)
                {
                if (bigstream2[2*i]==1)
```

```
                {
                if (bigstream2[2*i+1]==1)
                        {
                        code2[i]=1;
                        }
                else
                        {
                        code2[i]=2;
                        }
                }
        else
                {
                if (bigstream2[2*i+1]==0)
                        {
                        code2[i]=3;
                        }
                else
                        {
                        code2[i]=4;
                        }
                }
        }
for(j=0;j<100;j++)
        {
        if (code2[j]==bigstream[j])
                {
                correct=correct+1;
                }
        }
berfinal=(p-correct)/p;
printf("BER for qpsk is = %f\n",berfinal);
```

```
printf("\n");
nmbr=0;
}
```

5.1.3 Implementation of QAM on SMT-8036

In QAM, we send unique cosine signals with different amplitudes as shown in Fig 5.5

```
if(doWhat == 1)
    {
    for (i=0; i<PatternCount; i++)
        {
        if (data < 1)
        {
        sineWave = 2*cos(2 * PI * N * ( ( (float)i ) / PatternCount ));
        }
        else if (data < 2)
        {

        sineWave = cos(2 * PI * N * ( ( (float)i ) / PatternCount ));
        }
        else if (data < 3)
        {

        sineWave = cos(2 * PI * N * ( ( (float)i ) / PatternCount ) )-10;
        }
        else if (data >= 3)
        {
        sineWave = 3*cos(2 * PI * N * ( ( (float)i ) / PatternCount ))-9;
        }


        a = (unsigned short) ( fA * 1023 * sineWave );
        b = a;
        Pattern[i] = (a << 16) | (b);
        }
    }
```

Fig 5.5 Logic for QAM at the Transmitter

At the receiver, code for detecting these values for QAM is shown in Figure 5.6 given below:

```
int f;
if (stream2[1]<8184)
    {
    return 1;
    }
else if (stream2[1]<8194)
    {
    return 2;
    }
else if (stream2[1]<8202)
    {
    return 3;
    }
else if (stream2[1]>=8202)
    {
    return 4;
    }
```

Fig 5.6 Code for detecting QAM at the Receiver

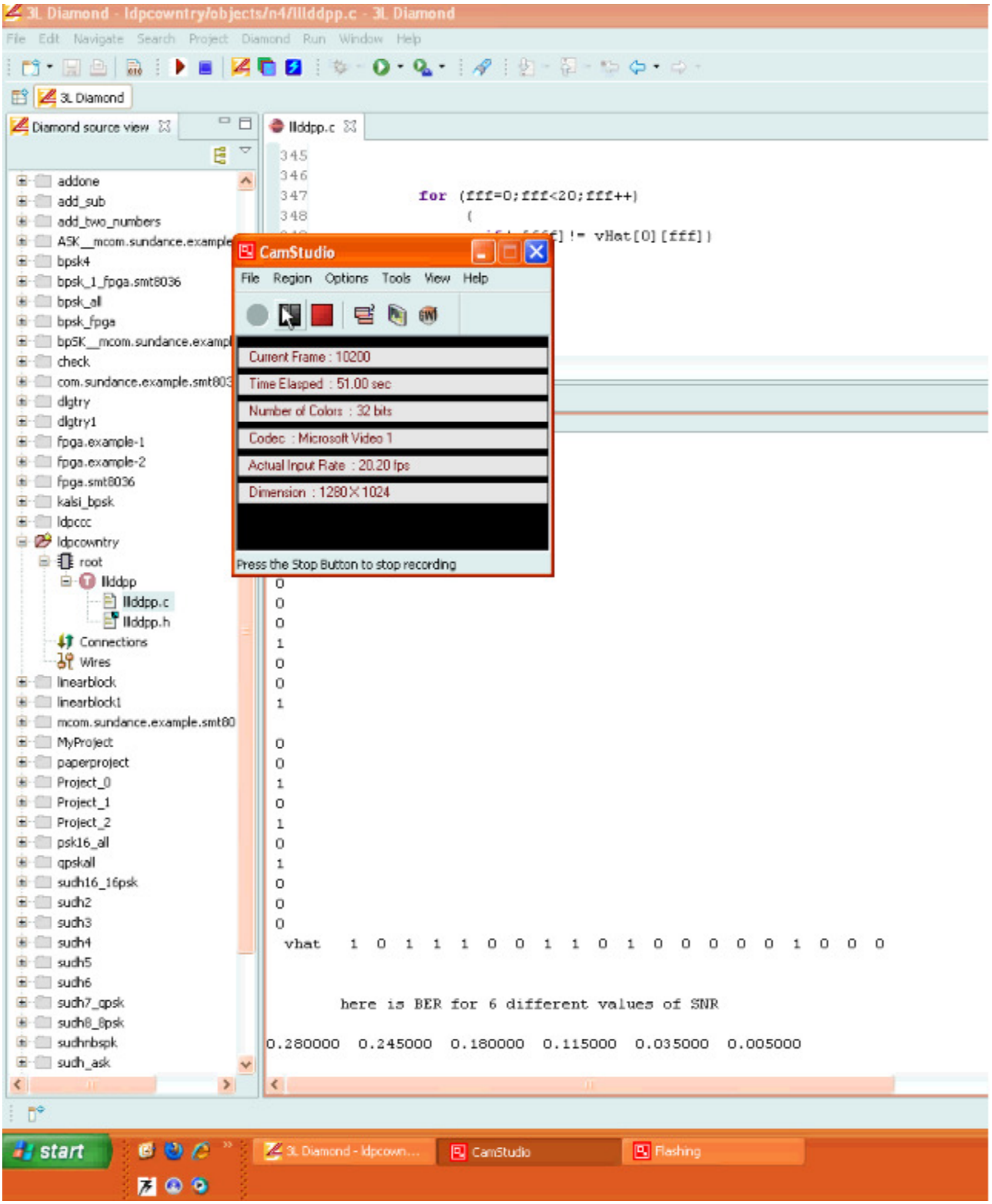


Fig. 5.7 The implementation of LDPC code on SDR8036 kit.

The procedure of the implementation of the project is discussed in above chapters in detail.

5.2 RESULTS

I have implemented the LDPC coding technique with three modulation schemes over a bit stream of 200 bits and successfully completed its repetitive wireless transmission and reception over SDR8036 kit so as to calculate the BER for different modulation schemes over different SNR values.

The results so obtained from these calculations are:

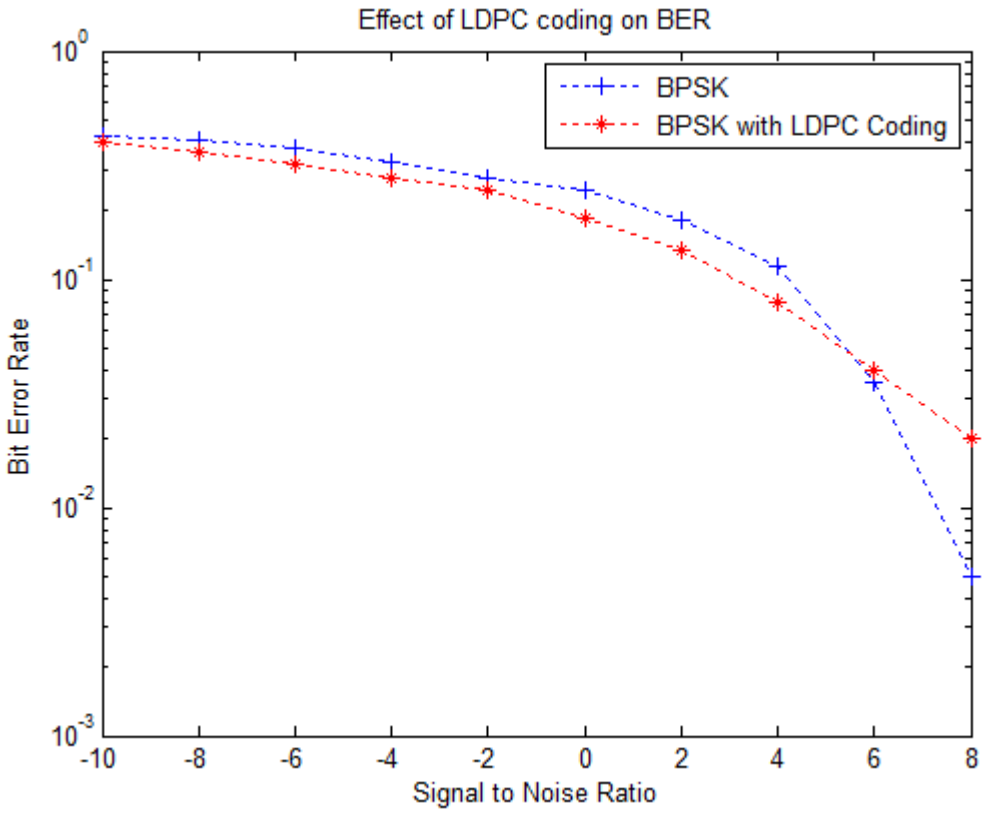


Fig 5.8 Comparison of BPSK modulation with LDPC and without LDPC for different values of SNR.

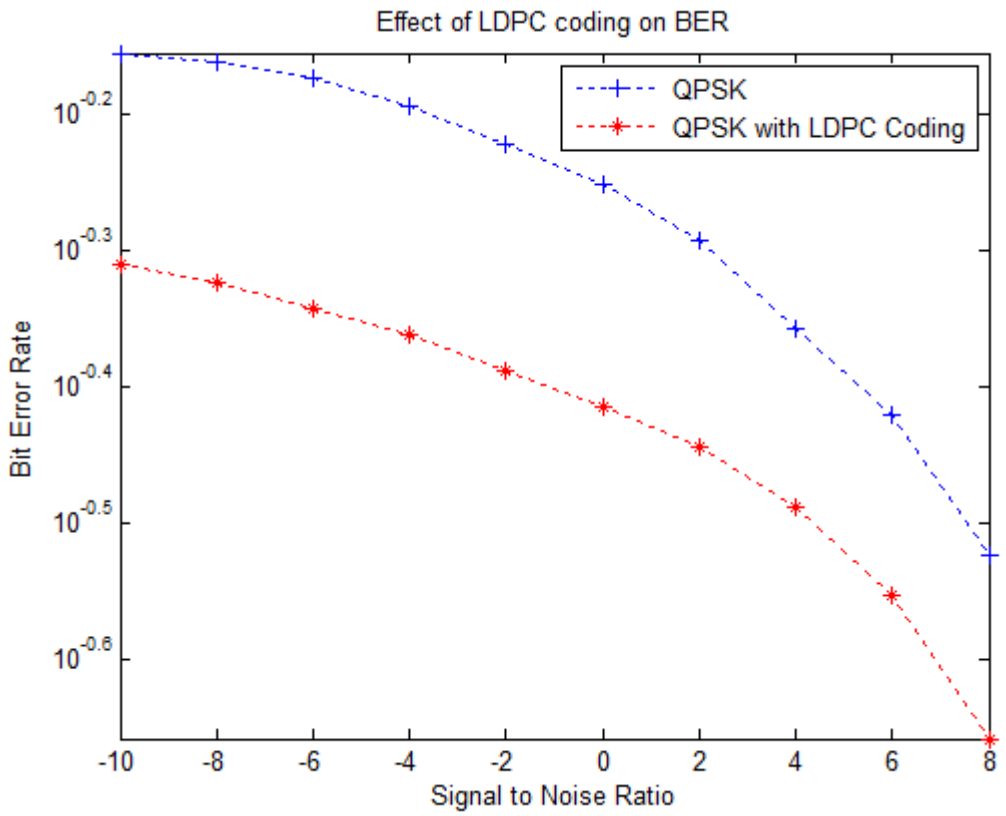


Fig 5.9 Comparison of QPSK modulation with LDPC and without LDPC for different values of SNR.

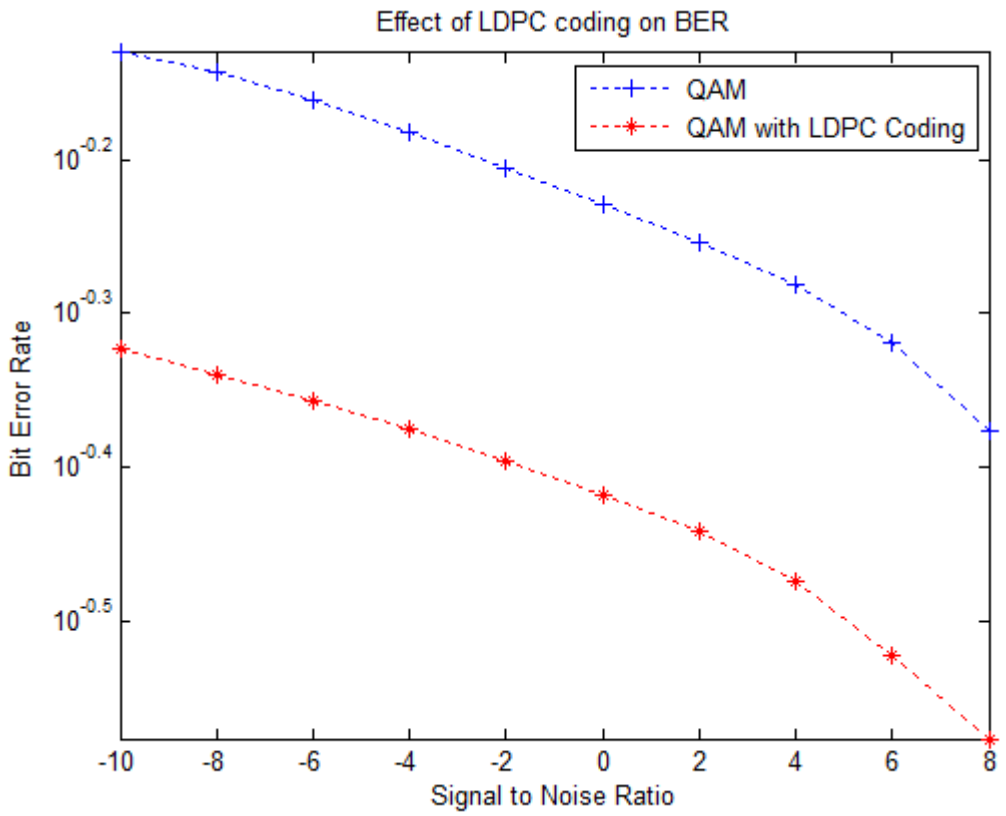


Fig 5.10 Comparison of QAM modulation with LDPC and without LDPC for different values of SNR.

5.3 CONCLUSION

As we are in the age of rapid development of wireless technology, Software defined radio is the base of next generation wireless technology. We can modify our system by just changing the software used in the hardware and hence saving the cost required to change the hardware at each and every place. This also saves the time required to implement the new technology system in practice because we can directly download the software from internet and install on our preexisting hardware.

Basically Software Defined Radio (SDR) system is a useful and adaptable future-proof solution to cover both existing and emerging standards; it provides elements with reconfigurability, intelligence and software programmable hardware. In addition, the

emerging user requirements on reconfigurable mobile systems and networks are paving the way for the introduction of reconfigurability in future mobile systems.

Also in parallel, the functionality of wireless devices is increased by SDR and they become more and more sophisticated. SDR provides an efficient and secure solution to the problem of building multi-mode, multi-band and multifunctional wireless communication devices. We can use tools to integrate the major development phases like simulation, code generation, even including hardware in the loop testing into one continuous design cycle.

We can implement SDR based application on the SMT8036 which is a development kit for SDR applications consisting of a DSP coupled with an FPGA connected to two ADCs and a DAC. It is a PCI system based on 3 main modules: C64xx-based module (SMT365-8-2) combined with a dual high-speed ADC/DAC module (SMT370), both plugged on a PCI carrier board (SMT310Q). We also use Xilinx system generator and Matlab with Simulink and various code generation tools for the implementation.

5.4 FUTURE SCOPE

I want to implement a receiver system, in which receiver can switch its demodulator circuit in accordance with the signal received, i.e. if ASK signal is transmitted then receiver will demodulate signal by ASK demodulator. If FSK signal is transmitted then FSK demodulator will be selected and so on. For this, I can use DSP processor to take decision i.e. which modulated signal (ASK, FSK, PSK (based upon its frequency received, its priority or BER values)) is received and FPGA processor for various demodulator implementations. To check this Adaptive receiver, we have to design a transmitter which can transmit the signal with different modulations in different frequency bands according to the prevailing channel conditions (e.g. SNR values).

REFERENCES

[1] Wipro Technologies, "SDR, A Technical Overview", White Paper, August 2002, pp 1-3.

[2] Joel Krishnan, "Optimize SDR performance", AWR Corporation, December 2011, pp1-2.

[3] Peter Ryser, "SDR with reconfigurable hardware and software", proceedings of Embedded System Conference, 2005, pp 1-2.

[4] "Software Defined Radio, SDR, Tutorial", *www.radio-electronics.com / info / rf – technology – design / sdr / software – defined – radios - tutorial.php,* last accessed on January, 2012.

[5] Sudhanshu Mehta, Rajesh Khanna, Surbhi Sharma, "SMT-8036 based implementation of secured Software Defined Radio system for adaptive modulation technique", *http:// dspace. thapar. edu:8080 / dspace / bitstream / 10266 / 1435 / 1 / sudhanshu + mehta + 800961024.pdf,* last accessed on May, 2012.

[6] A. P. Vinod, Edmund M-K. Lai, Amos Omondi (Ed.) "Special Issue On Signal Processing For Software Defined Radio Handsets", *Journal of Signal Processing System*, vol. 42, no.8, 2009, pp 2.

[7] M.Saravanan, Dr.S.Ravi, "SDR Based Multiple Access Technique for Multimedia Applications", *proceedings of Communications and Information Technologies*, 2007, pp 491-494.

[8] David Murotake and Antonio Martin, "A High Assurance Wireless Computing System (Hawcs™) For Software Defined Radio", *Proceeding of the SDR 06 Technical Conference and Product Exposition*, 2006, pp-116-118.

[9] Goran T. Djordjevic, Ivan B. Djordjevic, Predrag N. Ivanis, "Effects of LDPC Code on the BER Performance of MPSK System with Imperfect Receiver Components over Rician Channels", ETRI Journal, Volume 31, Number 5, October 2009 , pp 1-2.

[10] Michele Franceschini, Gianluigi Ferrari, Riccardo Raheli, and Aldo Curtoni, "Serial Concatenation of LDPC Codes and Differential Modulations", IEEE journal on selected areas in communications, vol. 23, no. 9, September 2005, pp 1-2.

[11] Mansour Ahmadian, Zhila (Jila) Nazari, "Model Based Design and SDR" proceedings *of 2nd IEE/EURASIP Conference*, 2005, pp 8.

[12] Adrian Tarniceriu, Bogdan Lordache, Silvian Spiridon, "An Analysis on Digital Modulation Techniques for Software Defined Radio Applications", *proceedings of Semiconductor Conference*, 2007, pp 571 - 574.

[13] Raúl Dopico-López , José M. Camas-Albar, Miguel A. Melchor,David Castells-Rufas, "How To Obtain More Powerful SDRs Using Multicore Architectures", *proceedings of the SDR'09 Technical Conference and Product Exposition*, 2009, pp 430-434.

[14] Feng Ge and Charles W. Bostian, "SDR Implementation Issues: Rf Front End Nonlinearity And Dynamic Computing Resource Allocation", *proceedings of the SDR'09 Technical Conference and Product Exposition*, 2009, pp 312-315.

[15] Mohamed Ratni, Dragan Krupezevic, "Broadband Digital Direct Down Conversion Receiver Suitable for Software Defined Radio", *The International symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, Vol-1, 2002, pp 100-104.

[16] T. Nesimoglu, M. A. Beach, J. R. MacLeod and P. A. Warr, "Mixer Linearisation for Software Defined Radio Applications" *Proceedings of Vehicular Technology Conference,* IEEE, Vol-1, 2002, pp 534-538.

[17] Ajay Kr. Singh, Ankita Taneja, et. al.," Software Defined Radio - Talk Through Application", *Proceedings of International Conference on Microwave – 08*, IEEE, 2008, pp 821-825.

[18] S. Glass, V. Muthukkumarasamy and M. Portmann, "A Software-Defined Radio Receiver for APCO Project 25 Signals", http:// portal . acm. org/ citation. cfm? id= 1582395, last accessed on March, 2012.

[19] Shinichiro Haruyama and Robert H. Morelos-Zaragoza, "A Software Defined Radio Platform with Direct Conversion: SOPRANO", *Kluwer journal of wireless Personal Communication*, Vol 2, 2001, pp 212 -217.

[20] Altera Corporation, "Architecture and Component selection for SDR applications", White Paper, Version 1.0, June 2007, pp 1-2.

[21] Rodger H. Hosking, "Software Defined Radio Handbook", Ninth Edition, Pentek Inc., pp 7-9.

[22] Hüseyin Arslan, "Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems", *Springer publication*, Second edition, 2007.

[23] Álvaro Álvarez, David Blanco, Rafael Palacio, "WHERE - adapted Platforms Description", http://www.kn-s.dlr.de/where/documents/Deliverable51.pdf, last accessed on May, 2012.

[24] "SMT-349 user manual", http://www.sundance.com/docs/SMT349UserManual.pdf, last accessed on May, 2012.

[25] http://www.3l.com/how-it-works, last accessed on September 2011.

[26] Bruce A. Fette, "Cognitive Radio Technology", *Newnes publications*, first edition, 2006.

[27] Bernhard M.J. Leiner, "LDPC - A brief tutorial", April 8, 2005.

[28] Fuqin Xiong, "Digital Modulation Techniques", *Artech House Publications*, second edition, 2006.

[29] Wayne Tomasi, "Electronic Communications Systems", *Dorling Kindersley (INDIA) Pvt. Ltd.*, Fifth edition, 2009.

[30] The McGraw-Hill Companies, Inc., "Quadrature Amplitude Modulation", McGraw-Hill Dictionary of Scientific & Technical Terms, 6E, www. encyclopedia2. thefreedictionary. Com / QAM, last accessed on May, 2012.

[31] Dr. Mark Humphrys, "Phone Lines", www. computing. dcu.ie /~humphrys / Notes / Networks / physical.phone.html, last accessed on May, 2012.