

# FPGA IMPLEMENTATION OF GSM BASEBAND PROCESSING FOR RECONFIGURABLE SOFTWARE DEFINED RADIO

## A DISSERTATION

*Submitted in partial fulfillment of the  
requirements for the award of the degree*

*of*

MASTER OF TECHNOLOGY

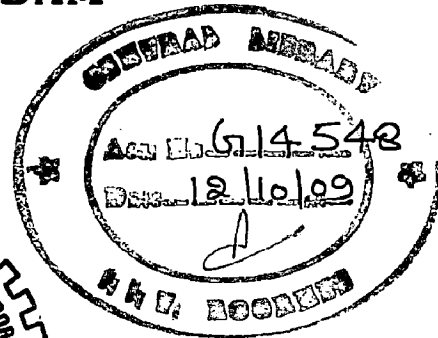
*in*

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Semiconductor Devices and VLSI Technology)

*By*

**SRINIVAS GADDAM**



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE  
ROORKEE - 247 667 (INDIA)

JUNE, 2009

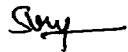
## Candidate's Declaration

I hereby declare that the work being presented in the dissertation report titled "FPGA Implementation of GSM baseband Processing for Reconfigurable Software Defined Radio" in partial fulfillment of the requirement for the award of the degree of Master of Technology in Semiconductor Devices and VLSI technology, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, is an authentic record of my own work carried out under the guidance of Dr. R.C.Joshi, Professor, and Dr.A.K.Saxena, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee.

I have not submitted the matter embodied in this dissertation report for the award of any other degree.

Dated: 29.06.09

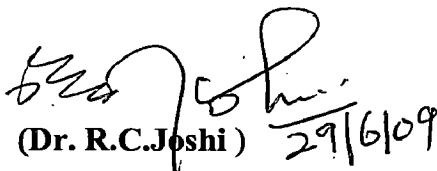
Place: IIT Roorkee.


  
(Srinivas Gaddam)

---

## Certificate

This is to certify that above statements made by the candidate are correct to the best of my knowledge and belief.

  
(Dr. R.C.Joshi) 29/6/09  
Professor

  
(Dr.A.K.Saxena)  
Professor

Department of Electronics & Computer Engineering

IIT Roorkee, Roorkee-247667(India)

Dated: 29.6.09

Place: IIT Roorkee.

## **ACKNOWLEDGEMENTS**

I am thankful to Indian Institute of Technology Roorkee for giving me this opportunity. It is my privilege to express thanks and my profound gratitude to my supervisor Prof.R.C.Joshi, and Prof.A.K.Saxena for their invaluable guidance and constant encouragement throughout the dissertation. I was able to complete this dissertation in this time due to constant motivation and support obtained from Prof.R.C.Joshi.

I am also grateful to the staff of Sponsored Research Laboratory and VLSI Laboratory for their kind cooperation extended by them in the execution of this dissertation. I am also thankful to all my friends who helped me directly and indirectly in completing this dissertation.

I am grateful to Mr. Hari Krishna Boyapati and Mr. Rahul for being excellent peers and creating a congenial environment for work.

Most importantly, I would like to extend my deepest appreciation to my parents, brothers and my best friend Sreelatha, for their love, encouragement and moral support. Finally I thank God for being kind to me and driving me through this journey.

  
(SRINIVAS GADDAM)

## Abstract

*Software defined radio is a feasible solution for reconfigurable radios, which can perform different functions at different times on the same hardware. The partial reconfiguration is the key feature of software defined radio. Partial reconfiguration is the ability of certain Field Programmable Gate Arrays (FPGAs) to reconfigure only selected portions of their programmable hardware while other portions continue to operate undisturbed. A FPGA can be partially reconfigured using a partial bitstream. We can use such a partial bitstream to change the structure of one part of an FPGA design as the rest of the device continues to operate and this reduces the reconfiguration time.*

*The aim of this thesis is to design and implement a software defined radio based wireless communication system (GSM). The baseband section of a wireless communication system is first simulated and then implemented in hardware. The performance of the baseband transmitter is analyzed using constellation and eye diagrams different signal-to noise ratio and different BT(bandwidth, time product) values, while considering an additive white Gaussian noise channel. The performance of the receiver is analyzed by comparing the bit error rates. The performance of the system in real time is also analyzed by implementing the system in hardware using Xilinx Virtex-4 field programmable gate array. A comparison of the simulation results with the results obtained from implementing the system on virtex-4 hardware is presented and discussed.*

*The two different GSM baseband processing versions have been developed i.e., one is area optimized and the other is speed optimized. The total hardware resources occupied by these units have been reduced through time-sharing between modules.*

## List of Figures

Figure 2.1	FPGA Structure	6
Figure 2.2	Compile Time Reconfiguration	7
Figure 2.3	Run Time Reconfiguration	8
Figure 2.4	Global Run-Time Reconfiguration	9
Figure 2.5	Local Run-Time Reconfiguration	9
Figure 2.6	Partially Reconfiguration process	11
Figure 3.1	Basic Architecture of Software Defined Radio (SDR)	21
Figure 3.2	Software Architecture of SDR	22
Figure 3.3	GSM Baseband Processing in SDR	23
Figure 4.1	GSM Transmitters and Receiver	24
Figure 4.2	The block diagram showing three classes of data and data flow at different stages	25
Figure 4.3	CRC generator (or) Parity bits generator	26
Figure 4.4	Simulation waveform of CRC generator used in GSM	27
Figure 4.5	(2, 1, 5) Convolution Encoder	28
Figure 4.6	Simulation waveform of convolutional Encoder	28
Figure 4.7	Interleaving process of 3-frames	30
Figure 4.8	Simulation waveform of interleaver	30
Figure 4.9	The A5/1 stream cipher	32
Figure 4.10	Simulation waveform of A5/1 cipher	32
Figure 4.11	Generation of normal Burst of traffic channel	33
Figure 4.12	Normal burst used in GSM	34
Figure 4.13	GMSK Modulator	35
Figure 4.14	GMSK Demodulator	36
Figure 4.15	BPSK Costas Loop	37
Figure 4.16	Simulation results of GMSK Modulation and Demodulation	38
Figure 4.17	Full rate channel coding	39
Figure 4.18	Read and write cycles of De-interleaving process	41
Figure 4.19	Simulation result of the Block deinterleaver module	41
Figure 4.20	Complete De-Interleaving Process	42

Figure 4.21	Simulation waveform of De-ciphering block	43
Figure 4.22	Trellis diagram for (2,1,4) convolutional encoder	45
Figure 5.1	Matlab Simulink model of GSM transmitter and receiver	47
Figure 5.2	BER vs Eb/No plot for BT=0.3,0.5	48
Figure 5.3	Constellation diagram of GMSK with SNR=15dB	48
Figure 5.4	Eye diagram of GMSK with SNR=15dB	49
Figure 5.5	FPGA Design Flow	50
Figure 5.6	System Generator Model GSM Transmitter & Receiver	52
Figure 5.7	Simulation window of GSM Transmitter & Receiver	53
Figure 5.8	Simulation waveform of Xilinx system generator model	54
Figure 5.9	Xilinx ISE 9.2i Window	55
Figure 6.1	Run-time and Partial reconfigurations of GSM baseband processing versions	58
Figure 6.2	Bus macro design	59
Figure 6.3	PlanAhead window showing two Partially Reconfigurable Regions	60

## **List of Tables**

Table 2.1	GSM specifications.	4
Table 4.1	Conventional radios vs software radios.	16
Table 5.1	Comparison of GSM baseband processing versions	56
Table 6.1	PR Directory Structure.	57

## Abbreviations

<b>CRC</b>	Cyclic Redundancy Check
<b>CTR</b>	Compile Time Reconfiguration
<b>DSP</b>	Digital Signal Processing
<b>DR</b>	Dynamic reconfiguration
<b>EA</b>	Early Access
<b>FPGA</b>	Field Programmable Gate Array
<b>GSM</b>	Global System for Mobile communication
<b>HDL</b>	Hardware Descriptive Language
<b>IEEE</b>	Institute of Electronics and Electrical Engineers
<b>IOB</b>	Input Output Blocks
<b>ISE</b>	Integrated Software Environment
<b>ISDR</b>	Ideal Software Defined Radio
<b>JTAG</b>	Joint Test Action Group
<b>NCD</b>	Native Circuit Description
<b>NGD</b>	Native Generic Database
<b>PC</b>	Personal Computer
<b>PCI</b>	Peripheral Component Interconnect
<b>PR</b>	Partial Reconfiguration
<b>PRR</b>	Partial Reconfiguration Region
<b>RTL</b>	Register Transfer Logic
<b>RTR</b>	Run Time Reconfiguration
<b>SDR</b>	Software Defined Radio
<b>UCF</b>	User Constraints File
<b>VHSIC</b>	Very High Speed Integrated Circuit
<b>VHDL</b>	VHSIC Hardware Description Language
<b>XST</b>	Xilinx Synthesis Technology
<b>XUP</b>	Xilinx University Program



# Table of Contents

<b>Candidate's Declaration &amp; Certificate</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>List of Figures</b>	<b>iv</b>
<b>List of Tables</b>	<b>v</b>
<b>Table of Contents</b>	<b>vi</b>
<b>Chapter 1: Introduction and Statement of the Problem</b>	<b>1</b>
1.1 Introduction	1
1.2 Statement of the problem	2
1.3 Organization of Thesis	3
<b>Chapter 2: Background and Historical Review of SDR</b>	<b>4</b>
2.1 History of GSM	4
2.2 FPGA for Baseband processing	5
2.3 Reconfigurable systems	7
2.3.1 Reconfiguration	7
2.3.2 Types of Reconfiguration	7
2.4 Partial Reconfiguration (PR) Techniques	10
2.4.1 Module Based PR	10
2.4.2 Difference Based PR	11
2.5 Historical review of SDR	12
<b>Chapter 3: Reconfigurable SDR and its Architectural Requirements</b>	<b>15</b>
3.1 Introduction	15
3.2 Concepts of SDR	16
3.2.1 Definition of SDR	16
3.2.2 Conventional radio vs Software Defined Radio	16
3.2.3 Technical Challenges	17
3.3 Architectural Requirements of SDR	18
3.3.1 Architectural characteristics intrinsic to SDR	18
3.3.2 Architectural characteristics import to SDR	19
3.3.3 Architecture of SDR	20

<b>Chapter 4: Design of GSM Baseband Processing for SDR</b>	<b>24</b>
4.1 Overview of digital blocks of GSM Transmitter and Receiver	24
4.2 GSM Transmitter	25
4.2.1 Speech coder	25
4.2.2 Channel Encoder	26
4.2.3 Interleaver	28
4.2.4 Encryption(A5/1 Cipher)	29
4.2.5 Burst Builder	32
4.2.6 GMSK Modulator	34
4.3 GSM Receiver	34
4.3.1 GMSK Demodulator	35
4.3.2 De-interleaver	37
4.3.3 De-cryption	43
4.3.4 Viterbi Decoder	43
4.3.5 CRC remover	44
<b>Chapter 5: Implementation Details of GSM Baseband Processing on FPGA and Results</b>	<b>46</b>
5.1 MATLAB Simulink model	46
5.2 Simulation Results	46
5.2.1 Bit Error Rate (BER)	47
5.2.2 Constellation Diagrams	48
5.2.3 Eye Diagrams	49
5.3 FPGA Design Flow	49
5.4 Simulation using ModelSim 6.0d	51
5.5 Hardware Co Simulation using Xilinx system generator tool	51
5.6 Implementation on virtex-4 FPGA	54
5.7 Comparison of Two GSM Baseband Processing Versions	56
<b>Chapter 6: Partial Reconfiguration of GSM Baseband Processing Versions</b>	<b>57</b>
6.1 Module Creation for Partial Reconfiguration	57
6.1.1 Design of PR Regions	58
6.1.2 Design of Static Module	58
6.1.3 Design of Bus Macro	58
6.2 Implementation Flow using ise 10.1 & PlanAhead 10.1	59

<b>Chapter 7: Conclusion and Scope for Future Work</b>	61
7.1 Conclusion	61
7.2 Scope for Future Work	62
<b>References</b>	63
<b>Authors Publications</b>	67
<b>APPENDIX A IMPLEMENTED DESIGNS</b>	68
<b>APPENDIX B DESIGN STATISTICS</b>	70

# CHAPTER I

## INTRODUCTION AND STATEMENT OF THE PROBLEM

---

### 1.1 Introduction

Mobile communication has grown exponentially ever since its emergence and is still growing phenomenally. In fact in the entire history of telecommunications the rate of growth of mobile communication has been unprecedented. Use of technology for widespread application of information transfer has been the most important factor for the success of mobile communication. The mobile technology, originating with analog mobiles has seen significant changes from 2G to 2.5G to 3G and now 4G standards are being frozen. The success of implementation of higher generation technologies rests on several consequent technologies particularly DSP.

Today a mobile communication system uses many different frequency bands. For the convenience of the users it is important that a single terminal, which can be programmed depending on the service available in a given region, functions for all the multiple accessing techniques and associated technologies. The DSP can provide such a desired flexibility. Considering the various signal processing functions and the multiband and multimode operations required in mobile communications, the software approach is more attractive than the hardware. This concept for radio connectivity has given birth to the nomenclature of software defined radio [1]. Advances in the analog to digital conversion and processor technologies have made it possible to go for the software radios, where in majority of the communication functions of a radio link are performed easily by reconfigurable and possibly down-loadable software. The SDR can provide multi-functionality, global mobility, compactness and power efficiency, ease of manufacture, and ease of upgrades. If one were to consider SDR principles for mobile communication there would be many issues and challenges [2] in the implementation of base stations and mobile stations. Key enabling technologies for software radio particularly for handset terminal implementation are signal digitization, silicon capability, signal processing, SIM cards, downloadable software, and personal Java / Java card. Suitable algorithms need to be developed for implementation of various functions.

Software defined radio (SDR) technology can be used to take advantage of programmable hardware modules to build open system architecture based on software. In this case, a variety of transceiver functions such as automatic gain control, frequency translation, filtering, modulation and demodulation can be integrated on a single hardware platform. This could result in maximizing the number of radio functions for a particular application. Software defined radio offers the flexibility and upgradeability necessary to satisfy these requirements [2].

Consider a typical communication system scenario where the user would like to have access to information through different wireless networks (e.g., CDMA, GSM, wireless local area network (WLAN), Bluetooth, etc.), or a mobile phone user may be traveling between two regions around the globe, where the wireless technologies or standards are different. To utilize the services offered by the broad range of technology alternatives around the world, the user has to carry different devices due to incompatibility of systems and standards. The practical solution to overcome this problem is to use a single device that can adapt to different technologies [4]. This could be possible using software defined radio, since it represents a radio that uses a reprogrammable hardware to create a generic hardware base. On top of the generic hardware platform, flexible software architecture is embedded. The software allows for multiple protocols, fast upgrades, and complete reconfigurations of radio features and functions.

## **1.2 Statement of the problem**

The objective of this thesis is to design and implement GSM baseband processing for reconfigurable software defined radio on FPGA.

The design of GSM baseband processing consists of the design of GSM transmitter and receiver. So to achieve these goals, the problem can be sub divided as follows:

1. To Model the GSM baseband processing in HDL (Hardware Descriptive Language) and simulating using ModelSim6.0d.
2. To Model the GSM baseband using Xilinx system generator tool and performing hardware co-simulation.

3. To synthesize and optimize the area, performance of modules using Xilinx ISE tools and to generate two versions of bitstreams.
4. To implement the GSM baseband processing on FPGA.
5. To perform partial reconfiguration of GSM baseband processing versions.

### **1.3 Organization of Thesis**

This thesis is organized as follows: **Chapter 2** presents the necessary concepts used in design of baseband processing section of SDR. These are, an overview of GSM communication standard, FPGA device structure and its advantage for GSM baseband processing, and partial reconfiguration techniques have been discussed. Historical review of different SDR architectures and baseband processing implementation methodologies developed in this thesis.

**Chapter 3** gives the fundamentals of SDR. Concepts of SDR including comparison of conventional radios with SDR, architectural requirements, technical challenges in implementing SDR and basic architecture of SDR.

**Chapter 4** describes the GSM baseband processing details. In this, GSM transmitter has channel coder, interleaver, burst builder, A5/1 cipher and GMSK modulator have been explained in detail. Similarly, GSM receiver has GMSK demodulator, de-cipher, burst de-builder, de-interleaver and channel decoder.

In **Chapter 5**, implementation details of GSM transmitter and receiver are presented. It includes, simulation results of MATLAB simulink, Modelsim 6.0d, Hardware co-simulation results. The synthesis results of GSM baseband processing versions are also explained. Comparison of two GSM baseband processing versions GSM\_v1, GSM\_v2 is done.

**Chapter 6** presents the partial reconfiguration of two GSM baseband processing versions. It includes, design of static module, controller of bus macros and partial reconfigurable modules.

Finally **chapter 7** provides conclusion and future scope.

## CHAPTER 2

### BACKGROUND AND HISTORICAL REVIEW OF SDR

---

This chapter provides useful concepts for SDR implementation and its previous research details.

#### 2.1 History of GSM

In 1982 the main governing body of the European telecommunication operators, also known as CEPT (Conference European des Postes et Telecommunications), created a committee called Group Special Mobile (GSM) and tasked it with specifying a pan-European cellular radio system to operate within 900 MHz band [7]. GSM is used by the 80% of the global market. GSM differs from its predecessors in that both signaling and speech channels are digital, and thus is considered a *second generation (2G)* mobile phone system.

ETSI published the GSM phase 1 specification in 1990 and commercial service began in 1991. Since then, GSM has grown to cover the world with both terrestrial and satellite networks. GSM was originally designed for operation in the 900 MHz band and has since been adapted to 1800 MHz (DCS1800), 1900 MHz (PCS1900) and is now being offered at 450 MHz. The 1900 MHz band is used in the United States and competes directly with CDMA (IS-95). GSM has steadily evolved with publication of phase 2 and phase 2+ specifications that add improved data services, new speech coding algorithms and other enhancements. Phase 2 add features like call waiting, call hold, conference calling etc., and phase 2+ covers multiple service profiles, private numbering plans, inter working with digital enhanced cordless telecommunication and other business oriented features. Table 2.1 gives the specifications of GSM standards.

In addition to traditional speech services, GSM provides a variety of data services including FAX and Short Messages. The Short Message Service (SMS) includes both broadcast and point-to-point text messaging. GSM started with circuit-switched data at various rates up to 9.6 kbps. Today, the maximum rate is 14.4 kbps to 115.2 kbps and will soon be extended to at least 384 kbps per user.

Table 2.1 GSM Specifications

Multiple Access Technology	FDMA/TDMA
Duplex Technique	FDD
Uplink band frequency	933-960 MHz
Downlink band frequency	890-915 MHz
Channel Spacing	200Khz
Modulation	GMSK
Speech coding	RPE-LTP
Speech channels per RF channel	8
Channel data rate	270.833kbps
Frame duration	4.615 ms

## 2.2 FPGA for Baseband Processing [4]

Throughout its history in the last 50 years, digital electronics technology has improved exponentially over time, doubling in performance roughly every 18 months while device sizes and costs have shrunk correspondingly.

One alternative being considered for the future is based on the technology of field programmable gate arrays (FPGAs). It should be obvious that every application would be best served by custom circuitry targeted specifically for it; and, in fact, application-specific integrated circuits (ASICs) are often made in response to special needs. But no one can afford to turn out a custom chip for every application he wants to run. FPGAs are able to meet the above requirements by their ability to be reconfigured any number of times. All FPGAs contain a regular structure of programmable basic logic cells surrounded by programmable interconnects and all these resources are configurable resources and its structure is shown in Fig 2.1.



FPGAs are usually slower than their application-specific integrated circuit (ASIC) counterparts, cannot handle as complex a design, and draw more power (for any given semiconductor process). But their advantages include a shorter time to market, ability to re-program in the field to fix bugs, and lower non-recurring engineering costs.

To define the behavior of the FPGA the user provides a hardware description language (HDL) or a schematic design. Common HDLs are VHDL and Verilog. Then, using an electronic design automation (EDA) tool, a technology-mapped netlist is generated. The netlist can then be fitted to the actual FPGA architecture using a process called place-and-route, usually performed by the FPGA company's proprietary place-and-route software. The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file generated is used to (re)configure the FPGA. Detailed design flow is explained in section 5.3.

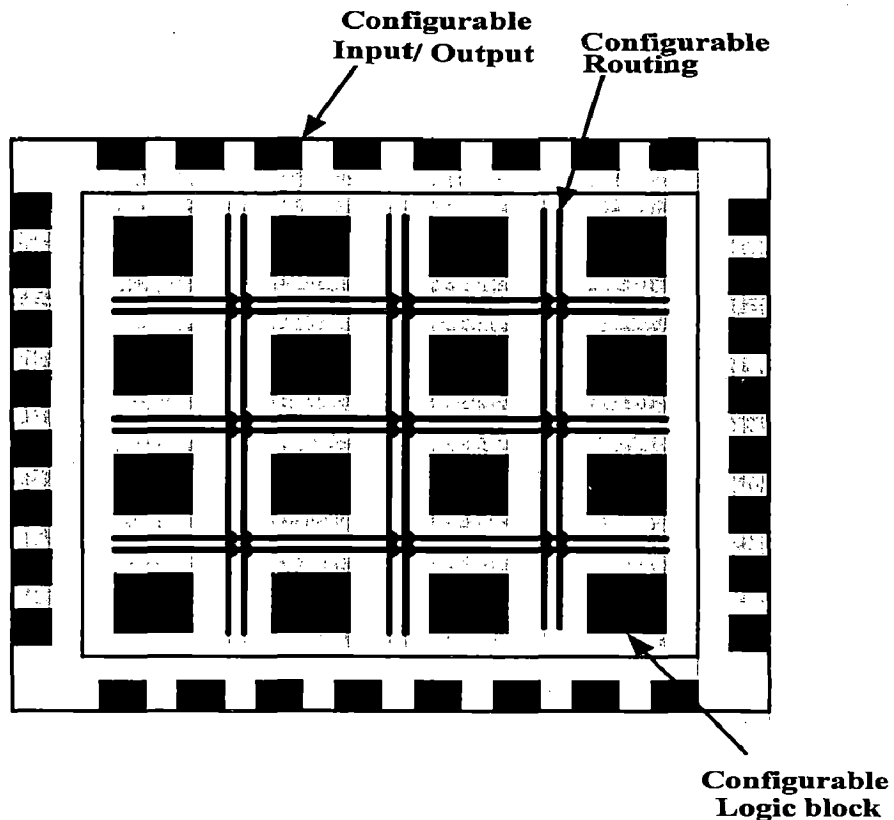


Figure 2.1: FPGA Structure

## 2.3 Reconfigurable systems

### 2.3.1 Reconfiguration

Reconfiguration is a post-fabrication process in which processing elements are programmed spatially and temporally i.e., computation in space and time, using hardware that can adapt at the logic level to solve specific problems [5].

The term “reconfiguration” refers to reprogramming an FPGA after its configuration is complete. Reconfiguration can be initiated by pulsing the full chip reset pin (this method is identical to configuration), or by re-synchronizing the device and sending configuration data. The latter method is only available in Select MAP and JTAG configuration modes. To reconfigure a device in Select MAP mode without pulsing full chip reset pin, the BitGen persist option must be set otherwise, the data pins becomes user I/O after configuration

### 2.3.2 Types of Reconfiguration

There are two types of reconfiguration mechanisms, depending on the use they make of the dynamic nature of the reconfigurable device.

#### a) Compile-Time Reconfiguration

CTR [5] is the simplest and most commonly used approach for implementing applications with reconfigurable logic. The most important feature of CTR applications is that they consist of a single system-wide configuration for all the system (Fig 2.2). The FPGAs are loaded with their respective configurations before the execution of the operation, and once execution of the application starts, they remain in this configuration till the end of execution.

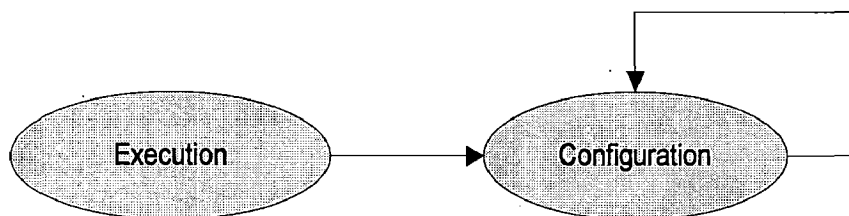


Figure 2.2: Compile Time Reconfiguration

This approach is similar to using an ASIC because the hardware does not change during the execution of the application.

### **b) Run-Time Reconfiguration**

Run-Time reconfigurable applications consist of a set of time-exclusive tasks that can be downloaded into the FPGA (one at each time, or several simultaneously) using a dynamic allocation scheme. In contrast to CTR, the FPGA will probably be reconfigured more than once during the execution of an application (Fig 2.3). Developing dynamic reconfiguration [5] is difficult because of the need for both software and hardware expertise to determine how best to partition a computation into sections to implement in hardware, how to sequence these circuit sections, and how to tie them together to produce an efficient computation. This overhead can be reduced to some extent by using dynamic partial reconfiguration which is described below.

The main advantage of RTR in front of CTR is that it allows reusing the reconfigurable device several times for the same application. To be able to do that it is necessary to partition the application into a set of configurations, but instead of using spatial exclusiveness as a criterion, this method uses time exclusiveness. We can distinguish two classes of run-time reconfiguration schemes Global reconfiguration and Local reconfiguration which are described below.

#### ***i) Global Run-Time Reconfiguration***

Here application is divided into distinct temporal phases where each phase is implemented as a single system wide configuration that occupies all system FPGA resources.

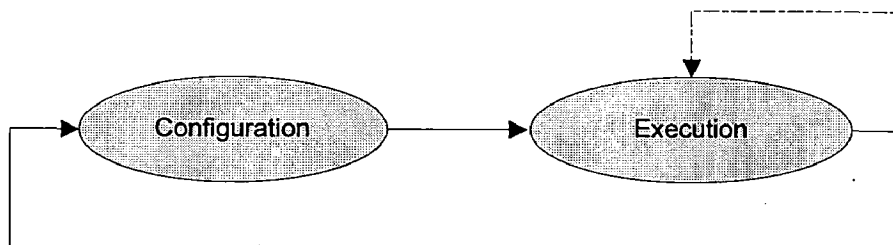


Figure 2.3: Run Time Reconfiguration.

In this case, reconfiguration time is more critical than in a CTR application. In this the reconfiguration of the FPGA is not only performed during the set-up of the system, but several times during the execution of the application. Fig 2.4 shows the execution of a Global RTR application which is mapped into two configurations.

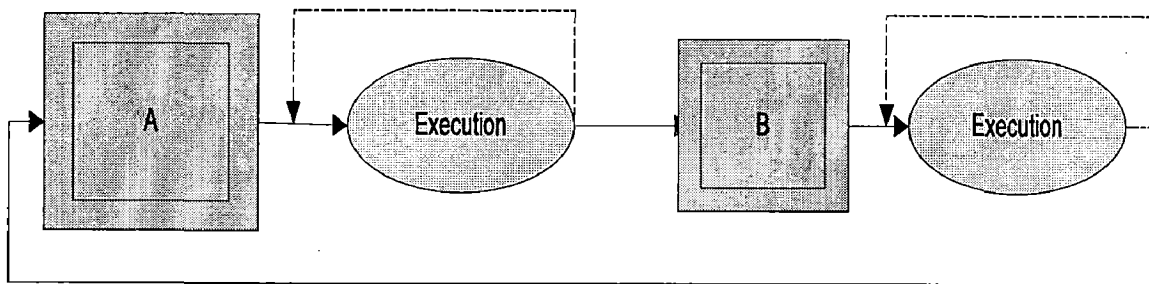


Figure 2.4: Global Run-Time Reconfiguration

**ii) Local Run-Time Reconfiguration**

It is also possible to reconfigure only subsets of the reconfigurable circuit. This approach is called partial reconfiguration or Local RTR. In this case important time-savings are made compared with a complete reconfiguration of the components, as reconfiguration is quite a time-consuming operation and with Local RTR not all the circuitry must be reconfigured to carry out changes.

Fig 2.5 shows an example of Local RTR where the application to implement consists of 4 partitions A, B, C and D. In a first step, partitions "A", "B" and "C" are loaded into the FPGA and then executed. In a second step, partitions "B" and "C" are removed and partition "D" is loaded into the FPGA, which is followed by the execution of the application.

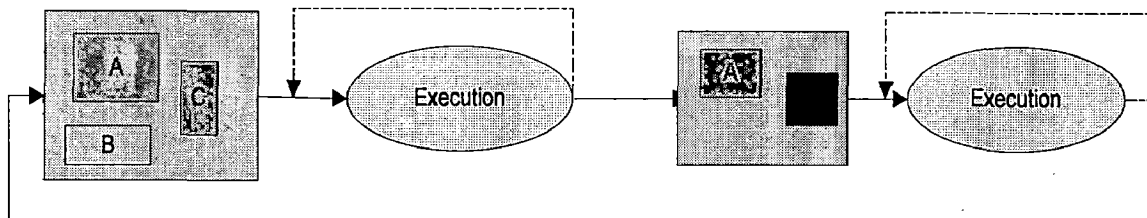


Figure 2.5: Local Run-Time Reconfiguration

## 2.4 Partial Reconfiguration Techniques

Reconfiguring the whole system is complicated costly in terms of overhead and may also be redundant in cases when desired functionality can be implemented by changing only a part of the circuit. The solution is to use partial reconfiguration which proves to be more efficient. Partial reconfiguration involves partitioning the hardware [9] within the platform to reduce the reconfiguration overhead. Partial Reconfiguration is the ability to reconfigure a portion of an FPGA while the remainder of the design is still operational. Certain areas of a device can be reconfigured while other areas remain operational and unaffected by reprogramming. If Partial Reconfiguration is done when the device is active it is called Active Partial Reconfiguration or Dynamic Partial Reconfiguration.

Dynamic Partial reconfiguration is again divided into two types [13]

- 1) Module-Based Partial Reconfiguration
- 2) Difference-Based Partial Reconfiguration.

**2.4.1 Module-Based Partial Reconfiguration:** In this method entire reconfigurable module is modified while leaving base region intact. Modular Design is best used for large designs that can easily be partitioned into self-contained modules. It is also used when communication is needed between modules.

### **Base and Partially Reconfigurable Regions (PRR):**

The base region is the portion of the design that does not change during partial reconfiguration and may include logic that controls the partial reconfiguration process. PRRs contain logic that can be reconfigured independently of the base region and other PRRs. The shape and size of each PRR is defined by the user through a range constraint. Each PRR has at least one, and usually multiple, partially reconfigurable modules (PRM) that can be loaded in to the PRR. Fig 2.6 illustrates a design with a single partial reconfiguration region PRR A. PRR A can be loaded with PRMs A1, A2, or A3. Each of the PRMs contains different logic for processing data passed from the static logic in the base region to the dynamic logic programmed in PRR A. Partial Reconfiguration can be

carried out in two different ways. It is possible to reconfigure part of the circuit while operation of the other parts is interrupted. This kind of reconfiguration is called Passive

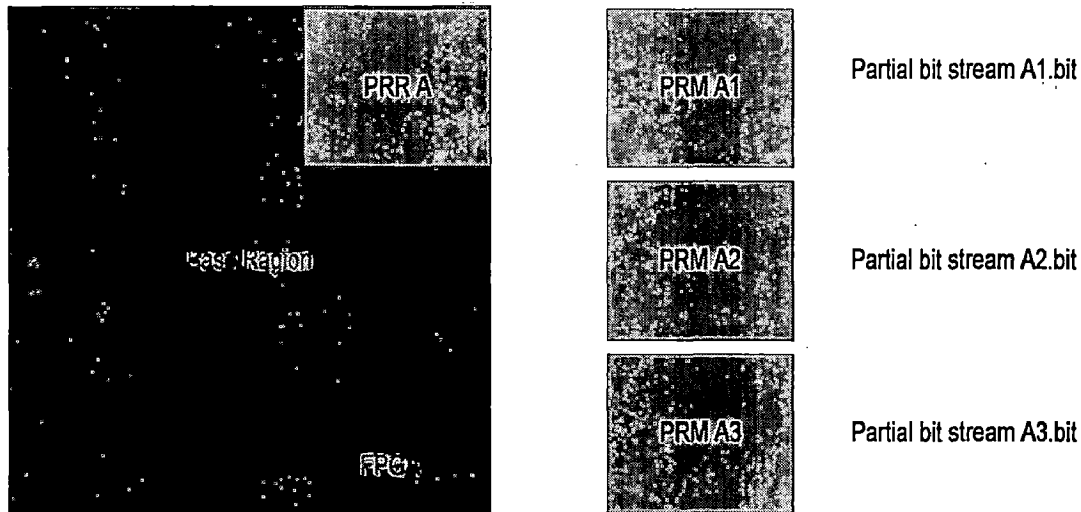


Figure 2.6: Module based partially reconfiguration process

partial reconfiguration. It is also possible in some cases, when partial reconfiguration is applied to leave the non-reconfigured parts of the circuit in operation while other parts are being reconfigured. This method is called Active Partial Reconfiguration. In the case of Passive Partial Reconfiguration time savings are made by lowering the reconfiguration time compared with a complete reconfiguration of the components. If Active Partial Reconfiguration is applied then time-savings are even more important as the execution of the application is not interrupted.

#### 2.4.2 Difference-Based Partial Reconfiguration:

This method of Partial Reconfiguration is accomplished by making a small change to a design (normally done in FPGA\_Editor), and then by generating a bitstream based on only the differences in the two designs. Switching the configuration of a module from one implementation to another is very quick, as the bitstream differences can be extremely smaller than the entire device bitstream. This method is very useful for implementing modules which differ by only little changes in their coding. In this work I have followed module based partial reconfiguration and it is described in detail in chapter 6.

## 2.5 Historical review

The term "Software Defined Radio" was coined in 1991 by Joseph Mitola, who published the first paper on the topic in 1992. One of the first public software radio initiatives was a U.S. military project named *SpeakEasy*. The primary goal of the *SpeakEasy* project was to use programmable processing to emulate more than 10 existing military radios, operating in frequency bands between 2 and 2000 MHz. Further, another design goal was to be able to easily incorporate new coding and modulation standards in the future, so that military communications can keep pace with advances in coding and modulation techniques. From 1992 to 1995, the goal was to produce a radio for the U.S. army that could operate from 2 MHz to 2 GHz, and operate with ground force radios.

In this *SpeakEasy* project, wide range was not supported due to architectural complexities, so *SpeakEasy* phase II was launched to operate in the range 4 MHz to 400 MHz. The goal was to get a more quickly reconfigurable architecture (i.e. several conversations at once), in an *open* software architecture, with cross-channel connectivity (the radio can "bridge" different radio protocols). The secondary goals were to make it smaller, weigh less and cheaper. The project was the first known to use FPGAs (field programmable gate arrays) for digital processing of radio data. The time to reprogram these is an issue limiting application of the radio.

After *SpeakEasy* II, JTRS (Joint Technical Radio System) was started. JTRS is a program of the US military to produce radios that provide flexible and interoperable communications. Examples of radio terminals that require support include hand-held, vehicular, airborne and dismounted radios, as well as base-stations (fixed and maritime). This goal is achieved through the use of SDR systems based on an internationally endorsed open Software Communication Architecture (SCA). The SCA, despite its military origin, is under evaluation by commercial radio vendors for applicability in their domains. In 1999, Joseph Mitola proposed SDR architecture and it's technical challenges [1].

[1] Describes the evolving concepts and architecture of software defined radio. It also presents the technical challenges like clock generation and distribution, power

management and receiver architecture. The main challenge in power management is sleep delay vs paging delay. To solve this drawback, a solution was proposed to use power managed DSP devices. To overcome the efficiency of receiver architecture, wideband SDR was proposed and to overcome the computational efficiency of software, FPGAs, JAVA engines were proposed. The architecture developed in [1] could not handle the power management properly.

In [7] GSM baseband processor was developed using high level language (C language) to implement on DSP kit. The A|RT Library extensions to C greatly simplified the hardware design effort, and A|RT Builder combined with Mistral 2 allowed to design the FPGA and yet maintain a C language environment for verification.

The FPGA implementation occupied approximately 70% of the logic resources of the Xilinx XCV800. It occupied 25 of 28 Block RAMs and consumes power less than 500 mW maximum. The maximum clock speed was 13 MHz and the external SRAM speed was 6.5 MHz. An ASIC implementation was expected to dramatically reduce power consumption and cost. The baseband processing developed in [7] was not concurrent approach.

[8] Presents concurrent software defined approach for common baseband processing. The main focus is on exploring the algorithmic and architectural design spaces of 3G and 4G systems to identify the computational and geometric structures shared by diverse coding schemes, services and hardware platforms, and the efficient and flexible integration of these structures on innovative extensible hardware.

A subset of GSM baseband processing modules involved in the generalized GSM transmitter have been successfully implemented using Linedancer. The algorithms developed for the parallel architecture enable efficient, concurrent multi user processing and contribute to the high speed software reprogrammable implementation. The design used 270 MHz clock rate for all its baseband module processing. The baseband model developed in [8] not used special purpose IP cores, which reduce the turnaround time. The synthesis time was also more.



[9], [10] Presents High-level “frameworks” such as the SCA (Software Communication Architecture) add virtualization layers on heterogeneous DSP-FPGA systems with the promise of code portability. The implementation was done in the general context of the development of an SCA board support package for Lyrtech’s DSP/FPGA development platforms as well as the development of example applications and reference designs running on these platforms.

The SCA GSM implementation was derived from a non-SCA model-based GSM design. To design these modules, special purpose VHDL code wrapped in a system generator block was used. As a result of this design, the DSP/FPGA platform operates 64 mbps and operating frequency of 65 MHz. In [9] area optimization of FPGA was not taken into consideration.

From the above historical review, it can be concluded that, the following major research gaps still exist.

In [7], the code developed has to be synthesized using synthesizer called Mistral 2. So the generated bitstream will occupy more silicon area of FPGA. So it is possible to optimize the FPGA silicon area.

In [8], technique introduced will take more time to synthesize the design and also increases the power consumption of FPGA. So it is possible to optimize the power consumption. This paper also presented the two baseband processing modules, GSM, OFDM. But it has not explained the reconfiguration of two baseband processing models. So it is also possible to perform the partial reconfiguration which is a key feature of software defined radio.

[9] This paper presented how a GSM waveform can be implemented for an SCA environment on a DSP/FPGA. This paper has shown only system generator model development. So it is possible to develop a baseband model using VHDL code to optimize the area.

Thus, this dissertation work is to effectively fill some of above stated research gaps.

## CHAPTER 3

### RECONFIGURABLE SDR AND ITS ARCHITECTURAL REQUIREMENTS

---

#### 3.1 Introduction

With the increase of wireless standards in television, radio, and mobile communications, compatibility issues have emerged in wireless networks. Inconsistency between wireless standards is causing problems to subscribers, wireless network operators, and equipment vendors [14]. Subscribers are forced to change their handsets whenever the latest breed of standards is introduced. Network operators face the dilemma during the upgrade of a network from one generation to another due to the presence of a large number of subscribers using legacy handsets incompatible with newer generations of standards. Equipment vendors face difficulty in airing new technology because of short time-to-market requirements [15]. Inconsistency between wireless standards is inhibiting deployment of global roaming facilities and causing problems in introducing new features and services [16]. Users are expecting more from their mobile terminals in terms of quality of service and multimedia applications. Traditional wireless systems, with their capabilities hard-coded in them, are no longer able to keep step with this brisk growth rate.

Introduction of software into the radio systems has brought the concept of software radio. It is now possible to realize various radio functions using suitable software on the same hardware. Such radios have been referred to as Software-Defined Radio (SDR). The SDRs are programmable and reconfigurable. Programmability / reconfigurability have become necessity of the day, because of the multiple standards, multiple frequency bands, and variety of applications.

The SDRs can provide multi-functionality, global mobility, compactness and power efficiency, ease of manufacture, and ease of upgrades. Design of SDRs required definition of suitable architecture and proper partitioning of the function in a radio system. Suitable algorithms need to be developed for implementation of various functions. The development of digital techniques in communication systems resulted in

additional performance improvement, because of use of source coding, channel coding, encryption, multiplexing, and multi accessing techniques. All these techniques can easily be defined and implemented in SDR.

### 3.2 Concepts of Software Defined Radio

#### 3.2.1 Definition of SDR

Definition of SDR is provided by the SDR forum [3], is that SDR is the radio that accepts fully programmable traffic and control information and supports a broad range of frequencies, air interfaces, and application software.

#### 3.2.2 Conventional Radio vs Software Defined Radio (SDR)

To compare the functionalities of Conventional radios with software radios, the following table is given.

Table 2.1: Difference between conventional radios and software defined radios

Conventional Radios	Software Defined Radios
Radio functionalities are primarily defined in hardware with minimum configurability in software	Radio functionalities are defined in software
Since the design is dominated by hardware, upgrading the design is not possible.	Software based architecture allows for easy upgrade of the design without abandoning the older design
The user has to use different mobile devices due to incompatibility of standards.	Global mobility can be achieved by downloading the appropriate air interface thus overcoming the incompatibility of standards.
Multi-function radios design including separate silicon for each system decreases the efficiency and becomes bulky.	Reprogrammability makes SDR to be efficient and compact
Results in waste of silicon area since each system has to be implemented separately	Silicon area is conserved by using the same chip to perform a function and changing the configurations during runtime to perform another function

### 3.2.3 Technical Challenges

This section discusses the technical issues, which have to be solved before software radio can be commercially available. The important technical issues involved in the development of a software radio system are as follows:

(1) In transceivers, the border between analog and digital domain should be moved closer, as much as possible, towards the RF. This requires ADC and DAC wide band converters placed as near as possible to the antenna. Increasing the border between the analog and digital domain is not exclusively for software defined radio. Much research has been carried out in the wideband transceiver realization [15]. The primary goal of this transceiver was to extend the digital domain at the IF stage and keeping the RF stage analog [15].

(2) The process of replacement of dedicated hardware in communication systems with DSPs or FPGAs should be further developed. In other words, we need to define the radio functionalities as much as possible in software. This opens the way to two possible horizons: software implementation of baseband functions, such as coding, modulation, equalization and pulse shaping; and re-programmability of the system to guarantee multi-standard operation. Though DSP technology has been used in implementing the baseband processing in base stations, it is not possible to categorize it as SDR since not all baseband functionalities are implemented in DSPs. Also, the software is limited and pre-loaded; therefore the system is constrained to a specific radio interface and cannot be reconfigured [1]. Hence, implementing communication functions in software presents a major challenge in practical systems.

(3) Analog-to-digital and digital-to-analog conversions for the ideal software defined radio are difficult to achieve. In practice, the selection requires trading power consumption, dynamic range and bandwidth. Current conversion technology is limited and is often the weak link in the overall system design. There are post digitization techniques based on multirate digital signal processing that can be used to improve the flexibility of the digitization process [2], [16].

(4) Power management is also a major challenge. For example, sleep modes of DSPs or other hardware save power but introduce a probability that the radio will be asleep during a paging message. A possible solution is a structured timing of paging messages, which reduces the probability of a miss, and further conserves battery life [17], [18].

(5) The clock generation and distribution is another challenge in SDR design. Every standard such as GSM or IS-95 has its own clock rate. Using one reference oscillator per standard may increase parts count, increase complexity, and therefore cost. A single master clock may use the least common multiple (LCM) of the required clocks, but this leads to a high clock rate, which is power inefficient. A possible solution is to use normalize standards to avoid clock rates with large LCMs [1].

(6) Receiver complexity is typically four or more times the transmitter complexity [2]. Thus, the receiver architecture has a first order impact on handset cost. The challenge is to develop a simple receiver. With the current technology, the support of many standards leads to complex and power inefficient solutions. Application specific integrated circuits are power efficient but inflexible. Field programmable gate arrays could be a possible solution. Hybrids of platform implementation could be utilized.

(7) The ideal radio frequency stage for SDR should incorporate flexibility in selection of power gain, bandwidth, dynamic range, etc. Achieving strict flexibility is impractical and trade-offs must be made [2]. These are the major challenges that must be addressed before full realization of SDR. Besides these important issues there are other challenges, which have to be solved like software architecture selection, hardware architecture selection etc., which are not discussed in this thesis. More information can be found in [1], [20].

### **3.3 Architectural requirements of SDR**

This section gives the details of the architectural requirements of the SDR.

#### **3.3.1 Architectural characteristics intrinsic to SDR**

To implement SDR system, the system architecture must be designed in the following aspects [21].

**a) Reconfigurability:** Reconfigurability is the ability to accommodate more than one functional unit on the same hardware by doing reconfiguration without changing the system architecture. Thus, it is the ability of change of a radio's personality through reprogramming of both software and hard-ware, in general. In addition, reconfigurability may be extended to change individual algorithms by changing the parameters of individual algorithms.

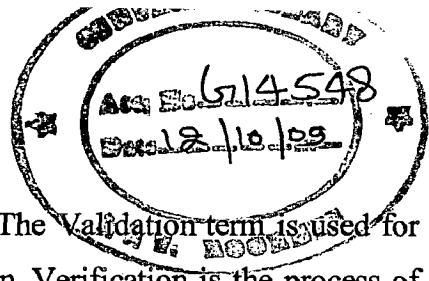
**b) Flexibility:** SDR architecture must exhibit to support more than one standard as well as for incorporation of future standards in a single device, with reprogramability as an essential feature.

**c) Modularity:** Modularity is another basic characteristic of SDR architecture. Modularity involves the encapsulation of each of the various tasks that define a system into individual and separate modules, whether in software or hardware, that can be linked together in a logical manner through their interfaces to form the desired system. In a well-designed modular SDR, the functionality of the system can be incrementally changed through the addition or replacement of individual modules without impacting the design of other modules.

### **3.3.2 Architectural characteristics import to SDR**

**a) Scalability:** One of the most important characteristics of a SDR is scalability, the ability to add new modules, either in hardware or software, to enhance the performance of the radio. *The synergistic effects of reconfigurability and scalability may prevent optimality.* Due to a software radio's many software and potential hardware changes and the complexity of the system, it is often difficult to say with absolute certainty how the radio will perform after a change is made and whether the radio will be able to accommodate these changes.

**b) Validation & Verification:** Validation addresses the functionality verification. Thus, designed architecture and particular HW/SW functionality correctness is important and should be verified before they are integrated as a combination of components in the integrated system. The difference between validation of architecture and validation of an



application is by referring to the latter as verification. The validation term is used for architecture and verification term is used for application. Verification is the process of conforming that a particular HW/SW combination with application, of the potentially limitless number of combinations will perform as expected. Verification is an essentially important issue for the wireless communication systems with multiple standard support like the SDR.

**c) Replicability:** Replicability is the ability to support addition of new channels to the system by simply adding copies of the basic radio. The replicability effectively provides scalability for the entire system, allowing it to expand to handle additional traffic.

**d) Interoperability:** The past decade has seen the introduction of many software radio products that can be drawn upon to reveal the practical characteristics of SDRs. However, with so many different SDR architectures with varying degrees of cross-compatibility being introduced, one may wonder if this multitude of SDR designs will bring the wireless market back, full circle, to interoperability problems similar to the ones that endangered the software radio concept. Fortunately, this potential problem was identified several years ago, and several movements have started the process of creating standardized SDR architecture. In an attempt to provide a measure of standardization, with a hope of promoting interoperability among SDR designs, a number of SDR groups have been formed.

### 3.3.3 Architecture of Software Defined Radio (SDR)

The digital radio system consists of three main functional blocks: RF section, IF section and baseband section [16] as shown in fig 3.1. The RF section consists of essentially analog hardware modules while IF and baseband sections contain digital hardware modules.

#### **RF Section:**

The RF section (also called as RF front-end) is responsible for transmitting/receiving the

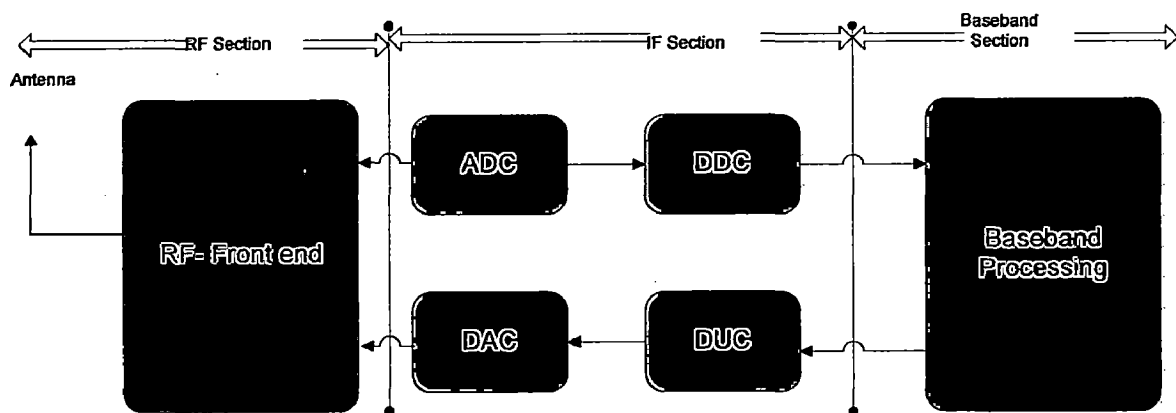


Fig 3.1: Basic Architecture of Software Defined Radio

radio frequency (RF) signal from the antenna via a coupler and converting the RF signal to an intermediate frequency (IF) signal. The RF front-end on the receive path performs RF amplification and analog down conversion from RF to IF. On the transmit path, RF front-end performs analog up conversion and RF power amplification.

#### IF Section:

The ADC/DAC blocks perform analog-to-digital conversion (on receive path) and digital to analog conversion (on transmit path), respectively. ADC/DAC blocks interface between the analog and digital sections of the radio system. DDC/DUC blocks perform digital down conversion (on receive path) and digital-up-conversion (on transmit path), respectively. DUC/DDC blocks essentially perform *modem* operations, i.e., modulation of the signal on transmit path and demodulation (also called digital tuning) of the signal on receive path. The baseband section performs baseband operations (connection setup, equalization, frequency hopping, timing recovery, correlation) and also implements the link layer protocol (layer 2 protocol in OSI protocol model). The DDC/DUC and baseband processing operations require large computing power and these modules are generally implemented using ASICs or stock DSPs. Implementation of the digital sections using ASICs results in fixed-function digital radio systems. If DSPs are used for baseband processing, a programmable digital radio (PDR) system can be realized. In other words, in a PDR system baseband operations and link layer protocols are



implemented in software. The DDC/DUC functionality in a PDR system is implemented using ASICs. The limitation of this system is that any change made to the RF section of the system will impact the DDC/DUC operations and will require non-trivial changes to be made in DDC/DUC ASICs.

A software-defined radio (SDR) system is one in which the baseband processing as well as DDC/DUC modules are programmable. Availability of smart antennas, wideband RF front-end, wideband ADC/DAC technologies and ever increasing processing capacity (MIPS) of DSPs and general-purpose microprocessors have fostered the development of multi-band, multi-standard, multi-mode radio systems using SDR technology. In an SDR system, the link-layer protocols and modulation/demodulation operations are implemented in software. If the programmability is further extended to the RF section (i.e., performing analog-to-digital conversion and vice-versa right at the antenna) an ideal software radio system that support programmable RF bands can be implemented. However, the current state-of-the-art ADC/DAC devices cannot support the digital bandwidth, dynamic range and sampling rate required to implement this in a commercially viable manner. Figure 3.2 illustrates the architecture of software components in a typical SDR system.

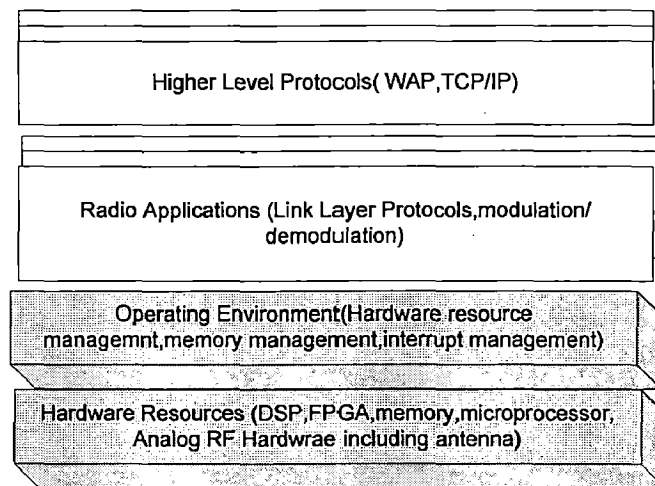


Fig 3.2: Software Architecture of SDR

The system uses a generic hardware platform with programmable modules (DSPs, FPGAs, microprocessors) and analog RF modules. The operating environment performs hardware resource management activities like allocation of hardware resources to different applications, memory management, and interrupts servicing and providing a consistent interface to hardware modules for use by applications. In SDR system, the software modules that implement link layer protocols and modulation/demodulation operations are called radio applications and these applications provide link-layer services to higher layer communication protocols such as WAP and TCP/IP.

**Baseband Section:**

In SDR the digital baseband processing consists of the following modules: speech coder and decoder, channel coder and decoder, interleaver, de-interleaver, burst builder and burst de builder, cipher and decipher, GMSK modulator and demodulator. The fig.3.3 shows the SDR with digital baseband processing modules.

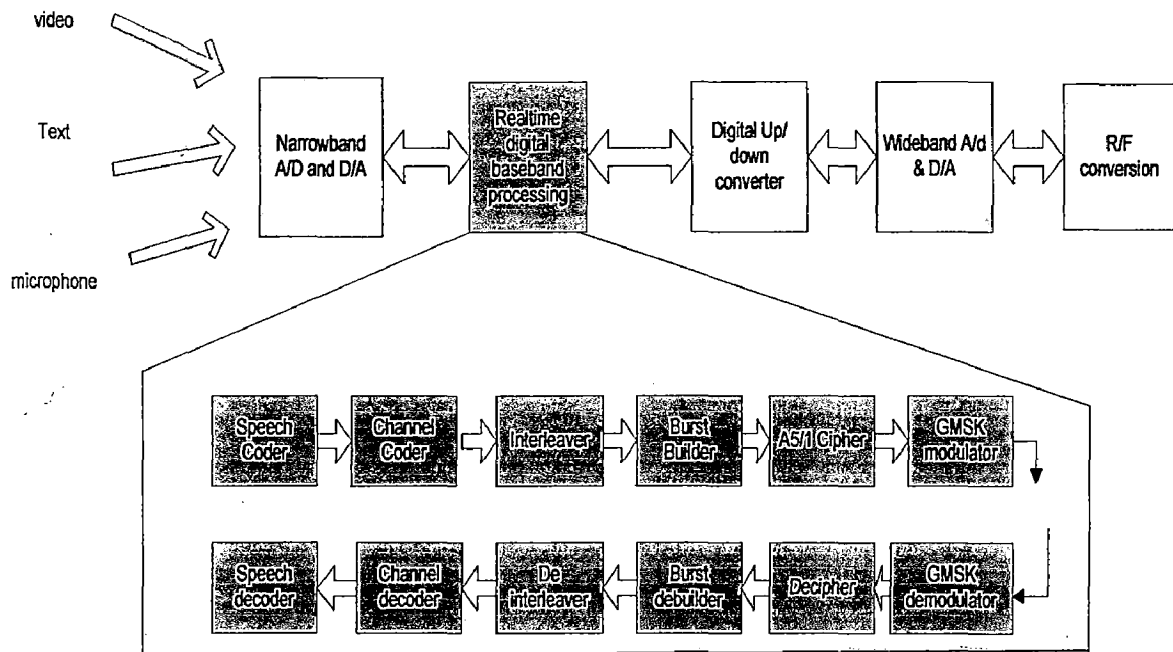


Fig.3.3: GSM Baseband processing in Software Defined Radio

## CHAPTER 4

### DESIGN OF GSM BASEBAND PROCESSING FOR SDR

---

#### 4.1 Overview of Digital blocks of GSM Transmitter and Receiver:

The analog part includes the RF transmitter and receiver and the digital part downlink signal processing consists of speech encoding, channel encoding, interleaving, encryption, burst building and modulation (Figure 4.1(a).) On the uplink, the signal processing consists of receive filtering, demodulation, equalization, decryption and channel decoding (Figure 4.1(b)) [24]. Each of these blocks has been implemented individually in VHDL and they are described below.

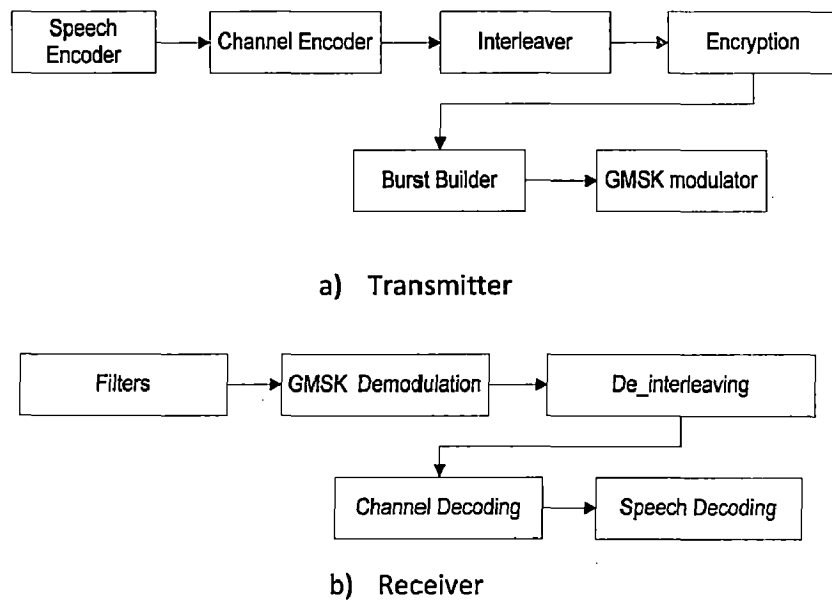


Fig 4.1: GSM Transmitter and Receiver

In this standard the data packets are sent at specific times at specific frequencies. Thus, several conversations take place simultaneously and at the same frequency using different time slots. Systems are also frequency duplex so that the transmit and receive frequencies are different, and both sides of the transmission (Mobile-to-Base and Base-to-Mobile) are concurrent.

## 4.2 GSM Transmitter

This section gives the detailed explanation of the blocks used in GSM transmitter.

### 4.2.1 Speech encoder:

Depending on compression achieved and quality of the resultant data following are three coding rates:

1. Full rate (RTE-LTP)
2. Half rate
3. Enhanced full rate

The speech coding scheme at 13 kbps is called RPE-LTP, which stands for Regular Pulse Excitation-Long Term Prediction is used in standard GSM. The 13 kbps rate is also referred to as "full rate". The full rate speech encoding algorithm processes 20 ms frames of speech and produces 260 bits of data per frame. The 20 ms input frame consists of 160 samples of speech at a sampling rate of 8 KHz. The speech blocks each of 20 ms duration coming out of speech coder are grouped into three classes of sensitivity to errors depending on their importance to the content of the information samples in speech. They are as following:

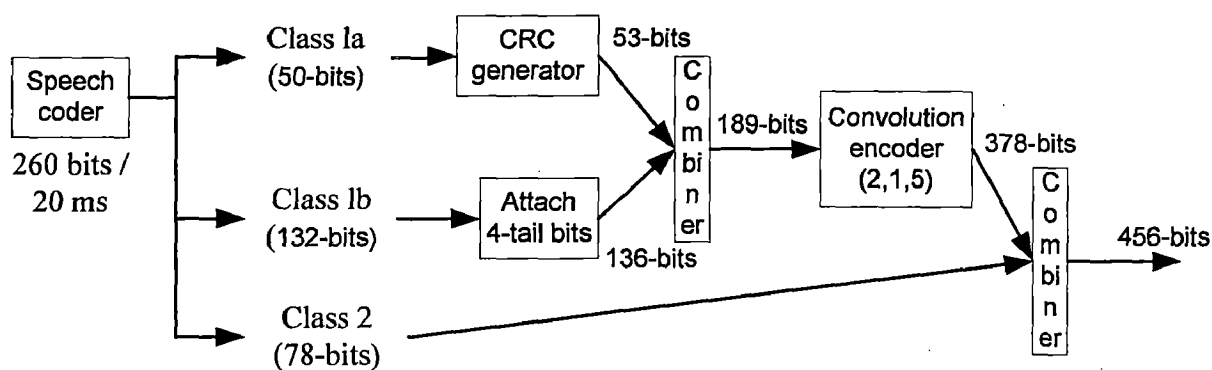


Figure 4.2: The block diagram showing three classes of data and data flow at different stages

1. Class 1a: Three parity bits are derived from the 50 class 1a bits. Transmission errors within these bits are catastrophic to speech intelligibility; therefore, the speech decoder

is able to detect uncorrectable errors within the class 1a bits. If there are class 1a bits in errors, the whole block is usually ignored.

2. Class 1b: The 132 class 1b bits are not parity checked but are fed together with the class 1a and parity bits to a convolutional encoder. Four tail bits are added first and then  $r = 1/2$ , (Constraint length  $K = 5$ ) convolutional code provides an output of 378 bits.

3. Class II: The 78 least sensitive bits are not protected at all.

A splitter has been designed to group the bits coming out from speech coder.

#### 4.2.2 Channel coder

As shown in fig.4.2, channel coder mainly consists of two blocks: CRC generator and convolutional encoder.

##### CRC generator:

Before convolutional coding, three bits of parity are added to class 1a bits. The generating polynomial is Generator Polynomial  $G = D^3 + D + 1$ . The block diagram of the CRC generator is as shown in figure 2. The hardware operation is as follows: three flip-flops with active rising edge are connected in linear feedback shift register manner and a switch in between which separates data input with the parity bits. For the first 50 clocks the switch remains closed and for the last 51 to 53 clocks the switch is kept open.

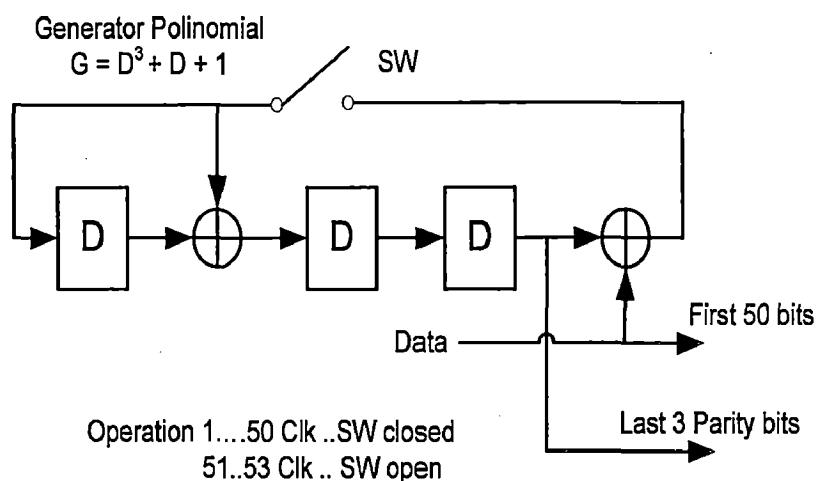


Fig.4.3: CRC generator (or) Parity bits generator

### Simulation result of CRC generator used in GSM:

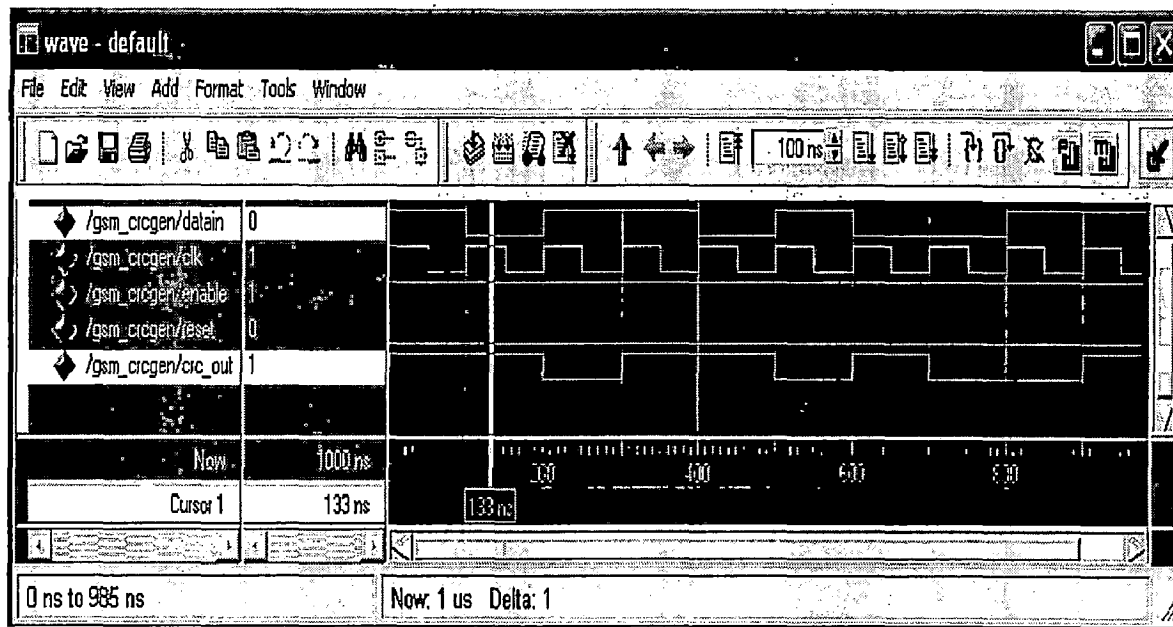


Fig 4.4: Simulation waveform of CRC generator used in GSM

### Convolutional Encoder:

In general the channel coder adds redundancy in a manner the decoder can detect and correct errors in received data stream. Convolutional encoder has been used as channel encoder for GSM standard. Convolution encoder is characterized by its constraint length and its code rate. The constraint length of encoder is defined as one plus the number of memory elements and code rate defined as ratio of input bits to output bits. In GSM communication standard traffic channel, convolution encoder of type (2, 1, 5) has been used. Whose constraint length is 5 and code rate is  $\frac{1}{2}$ . This code is applied to both the class 1b (132 bits + 4 tail bits) and class 1a (50 bits + including 3 parity bits). In order for a code to be able to correct errors, a certain number of additional bits have to be added. The added bits are called redundancy bits. Before the information bits are encoded, four bits are added. These bits are all set to zero and used to reset the convolutional code. These bits are called tail bits. The block diagram of a convolution encoder is shown in fig.4.5. For 189 bits input the output of the convolution encoder is 378 bits for one frame duration (i.e. 20 ms).

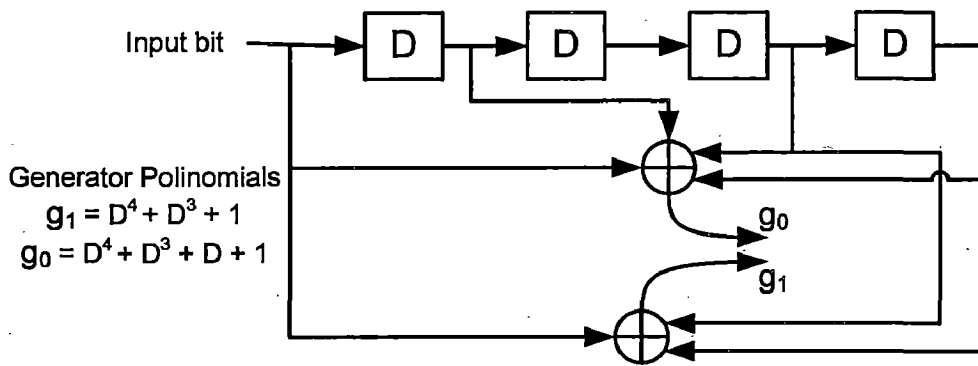


Fig.4.5: (2, 1, 5) Convolution Encoder

**Simulation Result:**

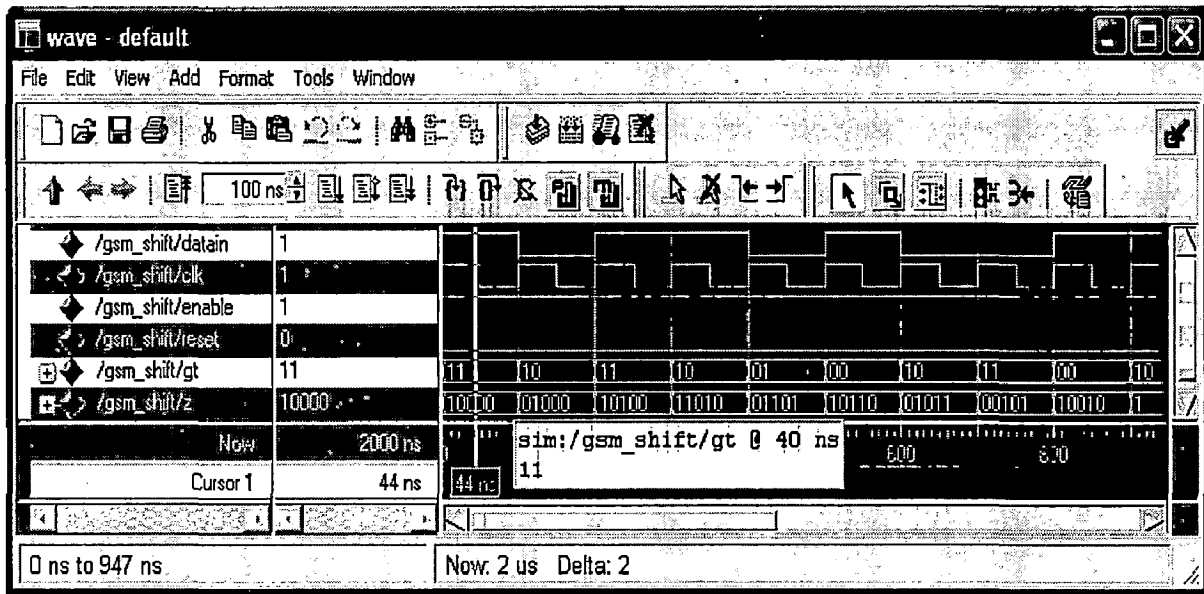


Fig 4.6: Simulation waveform of convolutional Encoder

**4.2.3 Interleaver:**

In order to combat the effects of error due to interference and noise, error correction techniques are used. The redundancy introduced due to error-correcting codes increases the data rate. For example, the raw data rate, due to speech coding, is only 260 bits over a period of 20 ms, as shown in fig 4.8.

However, after channel encoding through convolution encoder, the number of bits are increased to 456 bits, resulting in a data rate of 22.8 Kbps. Most probably the bit errors often occur in bursts. This is due to the fact that long fading dips affect several consecutive bits. To deal with this problem consecutive bits of a message are separated so that these are sent in a nonconsecutive way. This is done by interleaving, which is the process of distributing data bits in a different order in which they are generated. The output of the combiner from figure 2 fed to block interleaver.

The block interleaver divides the 456 bits of one frame into the eight sub blocks in the following way. Bit number 0 goes into sub block 1, bit number 1 goes into sub block 2, and so on until all eight sub blocks are used up. Bit number 8 ends up in sub block number 1 again. The first four sub blocks are put into the even-numbered bits of four consecutive bursts, and the second four sub blocks are put into the odd-numbered bits of the next four consecutive bursts.

First, the 456-bit encoded speech message block is read into an 8-column by 57-row matrix RAM, filling each row in turn. The bits are then read out of the RAM by column, forming eight sub blocks of 57 bits each. Note that adjacent bits in the code word are placed into different sub blocks. As each burst contains 114 traffic-carrying bits, it is in fact shared by two speech blocks. Each block will share four bursts with the block preceding it and four with the block that succeeds it. A burst will then be transmitted in the designated timeslot of eight consecutive TDMA frames, thus providing the interleaving depth of eight. The complete process in vivid manner is as shown in figure 5 above. The training sequences for different timeslots are given in table 4.1.

#### **4.2.4 Encryption:**

A5/1 algorithm has been used as encryption algorithm for this GSM communication standard. The A5 algorithm uses a frame number and key  $K_c$  to produce a stream of 114 bits that are used to encrypt and decrypt a burst of data [28]. At the transmitter, each of the 114 bits in the encryption stream is exclusive-OR'ed with the corresponding bits in the data stream. At the receiver, the A5 algorithm generates the same bit stream using the



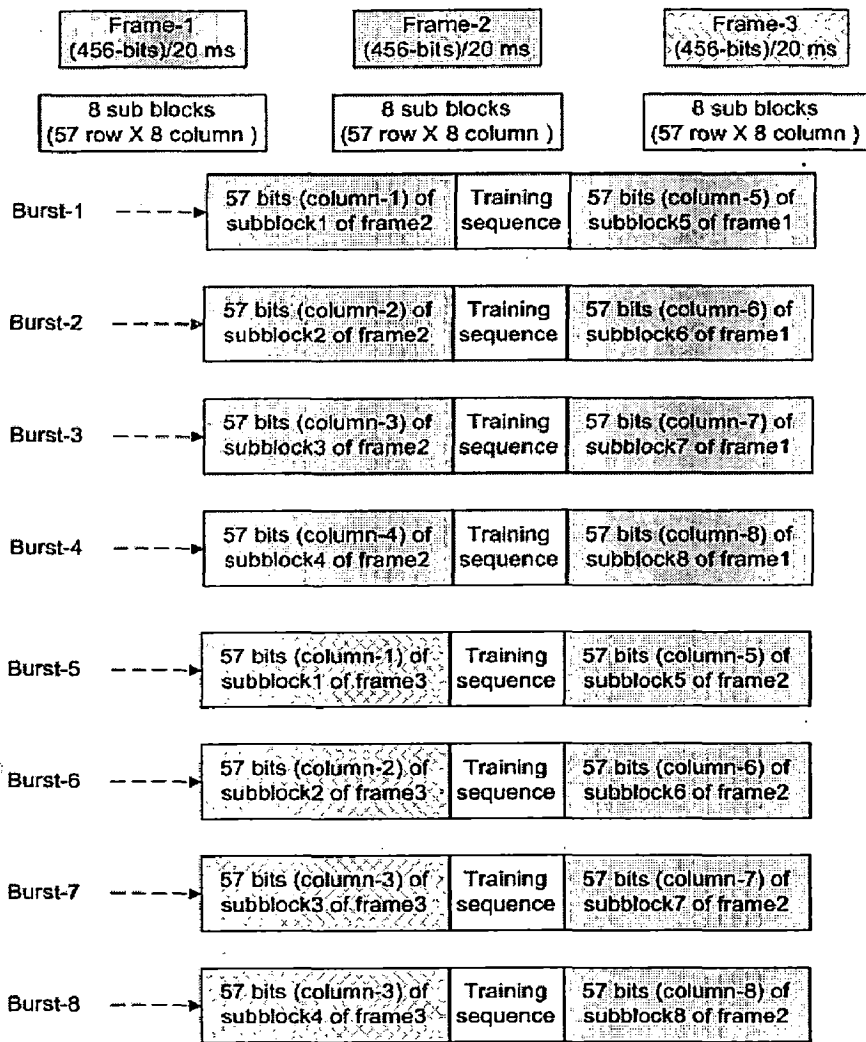


Figure 4.7: interleaving process of 3-frames

Simulation result:

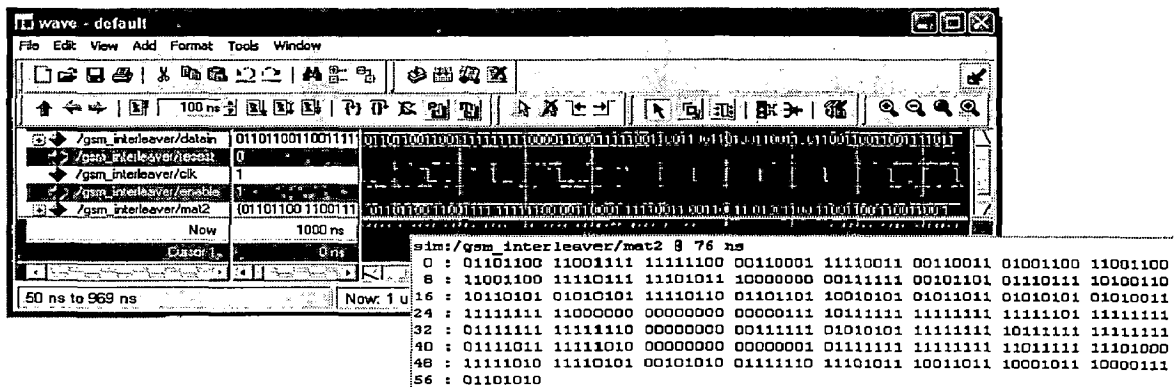


Fig 4.8: Simulation waveform of interleaver

frame number and Kc as at the transmitter and again performs the exclusive OR procedure to reproduce the original data bits. Since the frame number changes on every burst, the encryption stream also changes. Depending on network procedures, the key Kc may change on every call. This makes it very difficult for listeners to break the code. The A5/1 algorithm is as follows:

**A5/1 Stream cipher:** A5/1 is a stream cipher used for encrypting over the air transmissions in the GSM standard. A GSM conversation is transmitted as a sequence of 228-bit frames (114-bit in each direction) every 4.6 millisecond. Each frame is XORed with a 228-bit sequence produced by the A5/1 key stream generator. The initial state of this generator depends on a 64-bit secret key, Kc, which is fixed during the conversation, and on a 22-bit public frame number, Fn.

The A5/1 architecture is composed of three LFSRs,  $R_1$ ,  $R_2$ , and  $R_3$  of lengths 19-, 22-, and 23-bit, respectively. Each LFSR is shifted, using clock cycles that are determined by the Majority Function. This unit uses 3 bits  $C_1(8)$ ,  $C_2(10)$ , and  $C_3(10)$ . If two or more bits of them are zero then the majority is  $m = 0$ . Similarly if two or more of them are equal to 1 then the majority is  $m = 1$ . If  $C_k = m$  then corresponding register  $R_k$  is shifted, where  $k=1, 2, 3$ . The feedback polynomials for  $R_1$ ,  $R_2$ , and  $R_3$  are:  $x^{19}+x^5+x^2+x+1$ ,  $x^{22}+x+1$  and  $x^{23}+x^{15}+x^2+x+1$ , respectively. At each cycle, after the initialization phase, the last bits of each LFSR are XORed to produce one output bit. The fig.4.7 shows the A5/1 cipher algorithm.

A GSM transmission is organized as sequences of bursts. In a typical channel and in one direction, one burst is sent every 4.615 milliseconds and contains 114 bits available for information. A5/1 is used to produce for each burst a 114 bit sequence of key stream which is XORed with the 114 bits prior to modulation. A5/1 is initialized using a 64-bit key together with a publicly-known 22-bit frame number. In fielded GSM implementations 10 of the key bits are fixed at zero, resulting in an effective key length of 54 bits. The A5/1 stream cipher uses three LFSRs. A register is clocked if its clocking bit (orange) agrees with the majority of the clocking bits of all three registers.

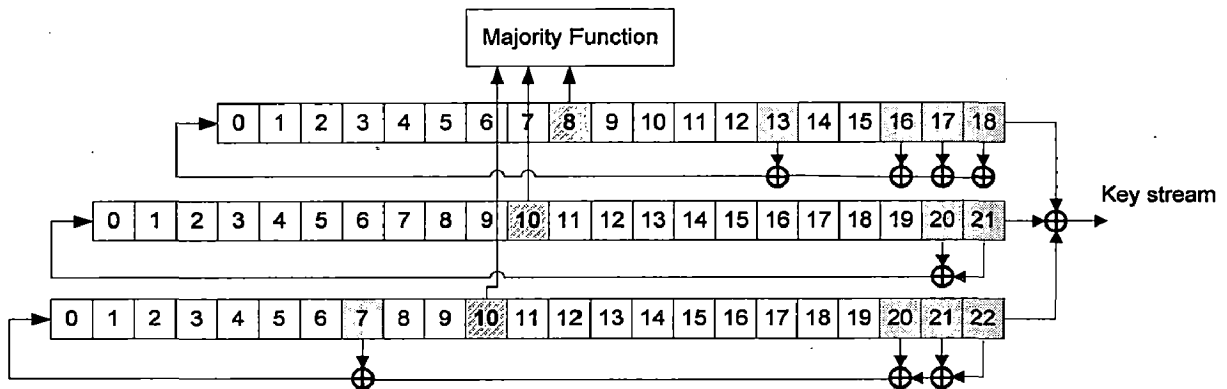


Figure 4.9: The A5/1 stream cipher

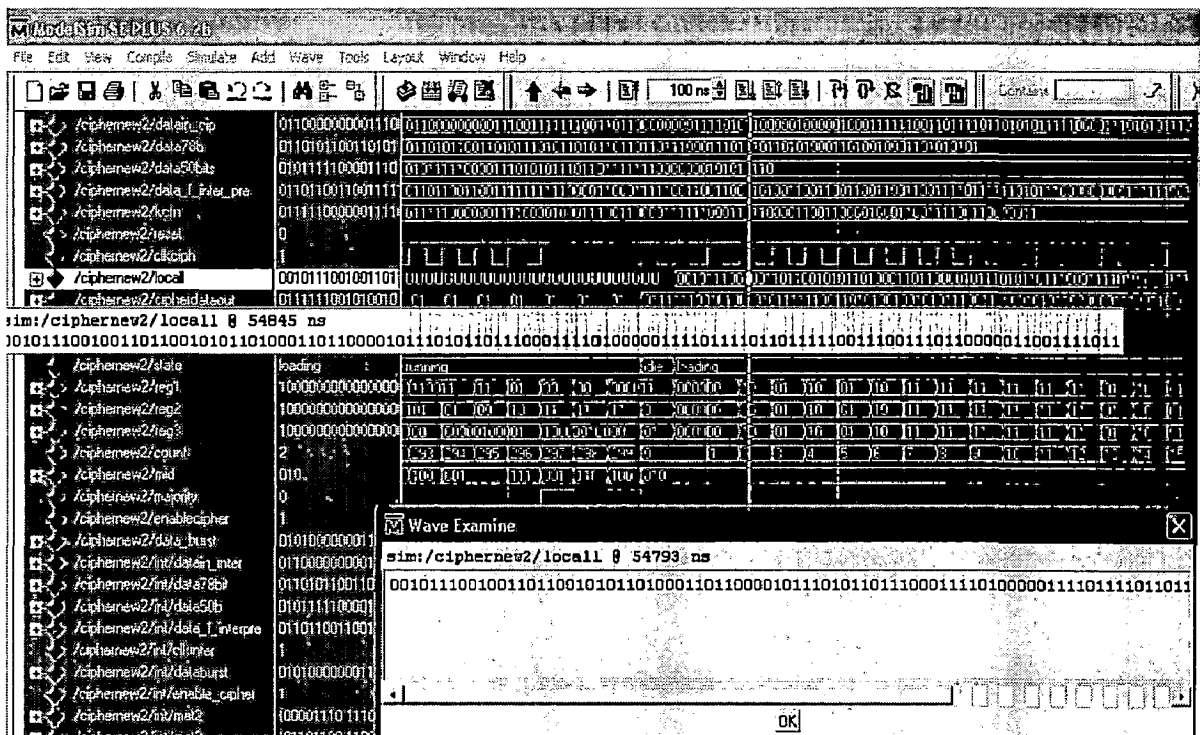


Fig.4.10: Simulation waveform of A5/1 cipher

#### 4.2.5 Burst Builder:

The GSM specifications define 4 different types of bursts; a normal burst has been considered (fig.4.10). The other types of burst (Frequency correction burst and Synchronization burst) are indeed only transmitted from the base station to the mobile station at known arrival time. A normal burst is used to transmit data information.

It lasts 576.9ms (15/26 ms) and is composed of 156.25 bits, which are:

- (a) 2\*3 tail bits used to allow the signal to ramp up and down for a transition.
- (b) 2\*57 data bits (two half burst of 57 bits each, from the interleaver, are introduced in a burst).
- (c) 2\*1 signaling flags, which indicate either the data is signaling traffic or user traffic.
- (d) 26 bits of training sequence.
- (e) 1\*8.25 bit-times of guard period (30.4ms) at the end of the burst to help compensate for multipath echoes.

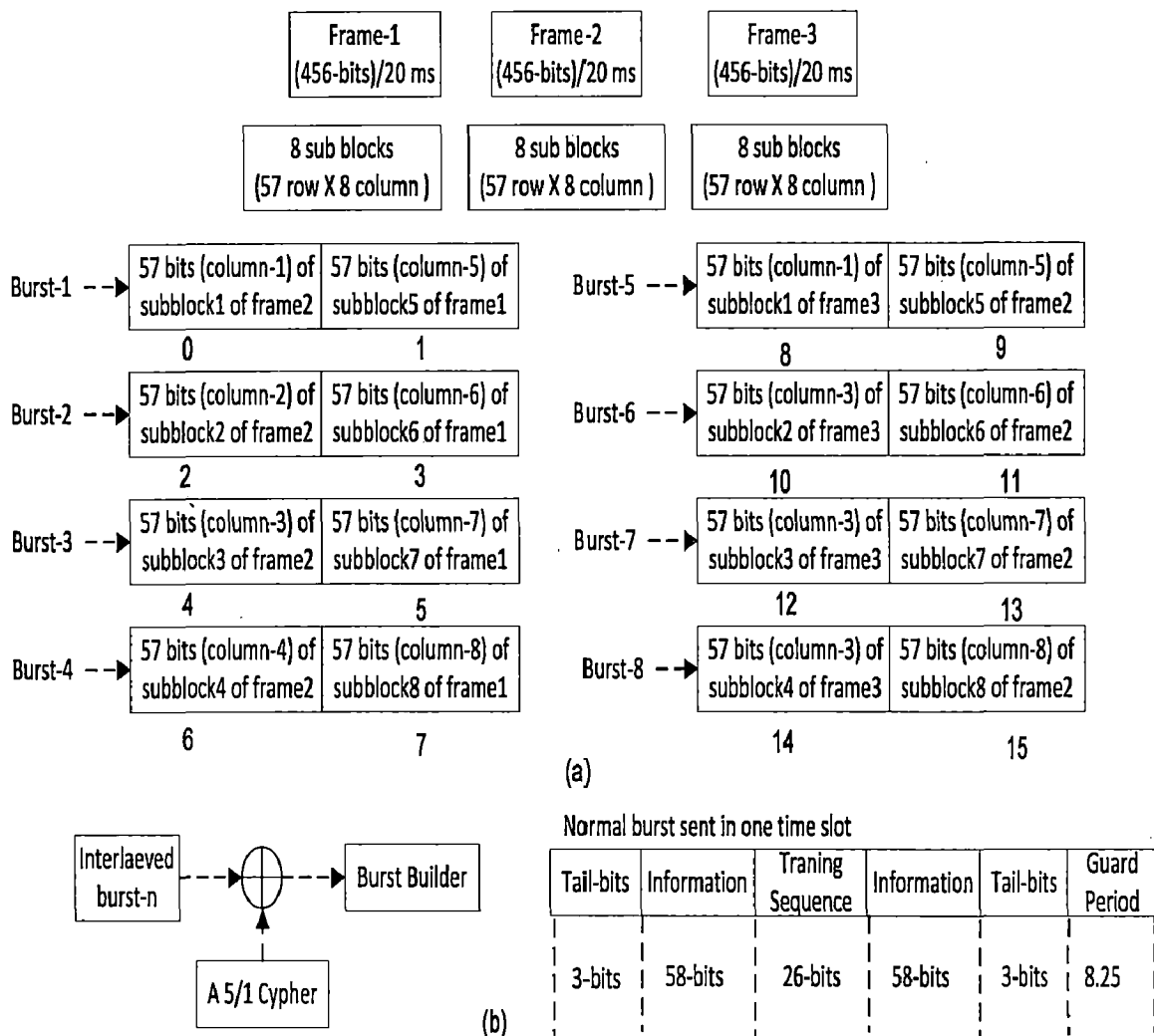


Fig 4.11: Generation of normal Burst of traffic channel

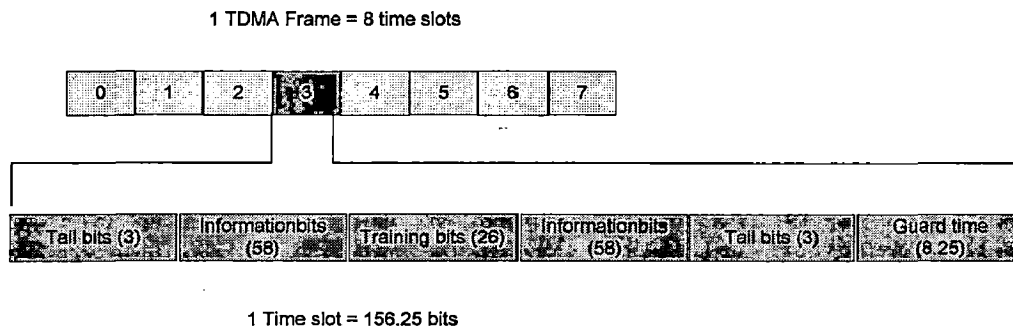


Fig 4.12: Normal burst used in GSM [3]

**4.2.6 GMSK Modulation:**

The GSM system uses a Gaussian Minimum Shift Keying(GMSK) modulation. This is a form of frequency shift keying. Minimum shift keying is named for the fact that the two frequencies used for the 0 and 1 states spaced the minimum distance required to maintain orthogonality for coherent detection. Coherent detection refers to matched filter detection using knowledge of carrier phase. The modulating bit rate is  $1/T = 1624/6$  K bits/sec or approximately 270.833 K bits/sec. The bits are differentially encoded prior to modulation [29]. The Gaussian filter has an impulse response that lasts about three bit periods. This means that the instantaneous transmitted phase is a function of a sequence of bits rather than a single bit.

The data  $d_i(t)$  to Gaussian Minimum Shift Keying (GMSK) modulator is first differentially encoded by performing modulo-2 addition of the current and previous bits, giving  $b_i(t)$  (as in the standard GSM system),

$$b_i(t) = d_i(t) \text{ XOR } d_{i-1}(t) \dots\dots\dots(1)$$

$b_i(t)$  has a value either 1 or 0. As input to the GMSK modulator must either be +1 or -1, we convert 1 to -1 and 0 to +1 using,

$$\hat{b}_i(t) = 1 - 2 b_i(t) \dots\dots\dots (2)$$

The modulating data,  $\hat{b}_i(t)$  are then passed through a Gaussian filter which has the response  $h(t)$

$$h(t) = \frac{1}{\sqrt{2\pi\sigma T}} e^{-\frac{t^2}{2\sigma^2 T^2}} \dots\dots\dots (3)$$

Where  $\sigma = \sqrt{\ln 2} / (2\pi BT)$

T is the bit period and B is the 3-dB Gaussian filter bandwidth. The BT product is the relative bandwidth of Gaussian filter which is used in the GSM system and is set to 0.3. The  $\hat{b}_i(t)$  after passing through the filter is then interlaced into odd  $\hat{b}_{oi}(t)$  and  $\hat{b}_{ei}(t)$  even bits. Modulated signal is then generated by using the following equation (8).

$$V_{GMSK}(t) = A [\hat{b}_{ei}(t) \sin 2\pi (t/4Tb)] \cos \omega_0 t + A [\hat{b}_{oi}(t) \cos 2\pi (t/4Tb)] \sin \omega_0 t \dots\dots (4)$$

here first and second term represents the inphase and quadrature phase components respectively as in fig 4.13.

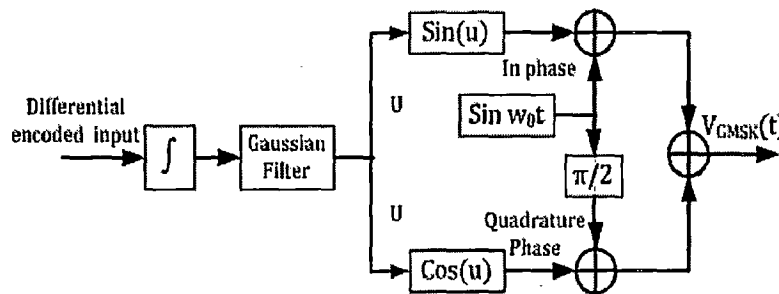


Fig.4.13 : GMSK Modulator

### 4.3 GMSK Receiver

This section gives the detailed explanation of the blocks used in GSM receiver.

#### 4.3.1 GMSK Demodulator

As mentioned above carrier is recovered using a loop of center-frequency locking scheme from a BPSK Costas-loop [20]. Subsequently carriers  $\cos\omega_0t$  and  $\sin\omega_0t$  are extracted from Voltage Controlled Oscillator (VCO) and  $\pi/2$  phase shifter of the Costas loop.

Then  $\cos\omega_0t$  is multiplied with the whole GMSK signal which is received at the receiver and let say an intermediate signal  $x(t)$ , is produced. Similarly  $y(t)$  is produced by using carrier  $\sin\omega_0t$ .  $x(t)$  and  $y(t)$  are given by the following Equations respectively.

$$x(t) = v_{\text{GMSK}}(t) \times \cos\omega_0t$$

$$y(t) = v_{\text{GMSK}}(t) \times \sin\omega_0t$$

When  $x(t)$  is passed through a low pass filter inphase component  $[b^{\wedge}ei(t) \sin 2\pi (t/4Tb)]$  is recovered. Similarly  $y(t)$  gives quadrature phase component  $[b^{\wedge}oi(t) \cos 2\pi (t/4Tb)]$ . Then we generate the components  $\sin 2\pi (t/4Tb)$  and  $\cos 2\pi (t/4Tb)$  using known values of  $T_b$ , and recovered the original odd and even bit sequences  $b^{\wedge}oi(t)$  and  $b^{\wedge}ei(t)$ .

### Carrier Recovery using COSTAS LOOP:

The mechanism of the Costas loop carrier recovery is to iterate its internally generated carrier from the VCO into the correct phase and frequency based on the principle of coherency and orthogonality. The outputs of the both Low Pass Filter (LPF) give the information about the signals which is modulated using BPSK. But the information we have taken only the carriers coming out from VCO and  $\pi/2$  phase shifter (i.e.  $\cos\omega_0t$  and  $\sin\omega_0t$  respectively).

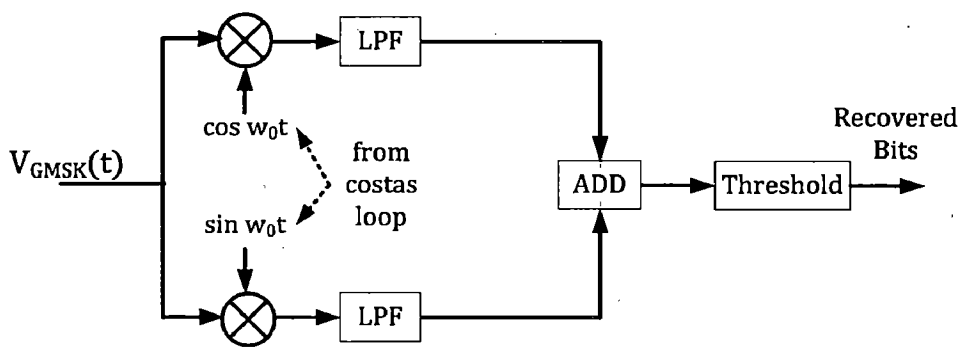


Fig.4.14: GMSK Demodulator

Figure 4.14 shows BPSK Costas loop. Two LPF represents two parallel tracking loops (I and Q) (i.e. Inphase and Quadrature phase), simultaneously a loop filter is used to drive the product of the I and Q components of the signal that drives the VCO. Once the

frequency of the VCO is equal to the suppressed carrier frequency, an error voltage is produced proportionally by the I and Q multiplication, which is passed through the loop filter and then VCO to control the frequency of VCO (i.e. carrier frequency).

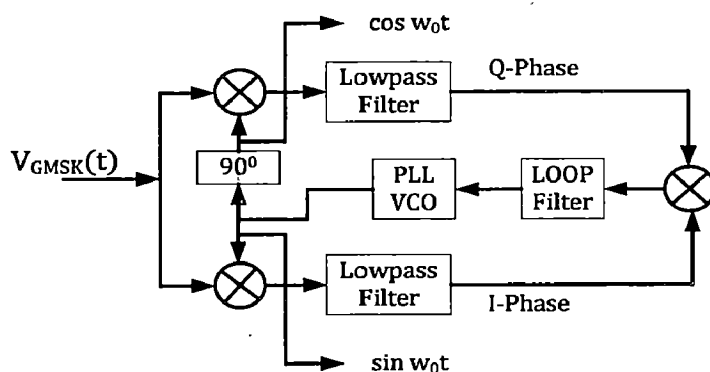


Fig.4.15: BPSK costas loop

Simulation result of GMSK modulator and demodulator is shown in fig.4.16.

#### 4.3.2 Deinterleaver:

The block deinterleaver is used in the receiver side of the GSM system. The function of the deinterleaver is just opposite of the interleaver we used in the transmitter side. This module does have the great functionality in GSM system. As said above that the functionality is just opposite to the interleaver, it reduces the number of bits received in incoming signal after processing. A special type of processing done inside this module which rearranges the bit pattern and reduces in number also to get the original bit sequence started from the transmitter side. Our system is capable of doing parallel processing in every stage from input to output. The fast processing algorithm used in this module allows it to take all the incoming bits at once, process them in parallel and giving output regarding all the bits very fast in very less time using 4 clock pulses only. The main purpose of the de-interleaver is to reverse the functionality of the block interleaver used in the transmitter side and convert the bit sequence, received, in that manner so as to get the meaningful information sent from transmitter. The fig.4.17 below shows the full rate channel decoding scheme used in GSM system.



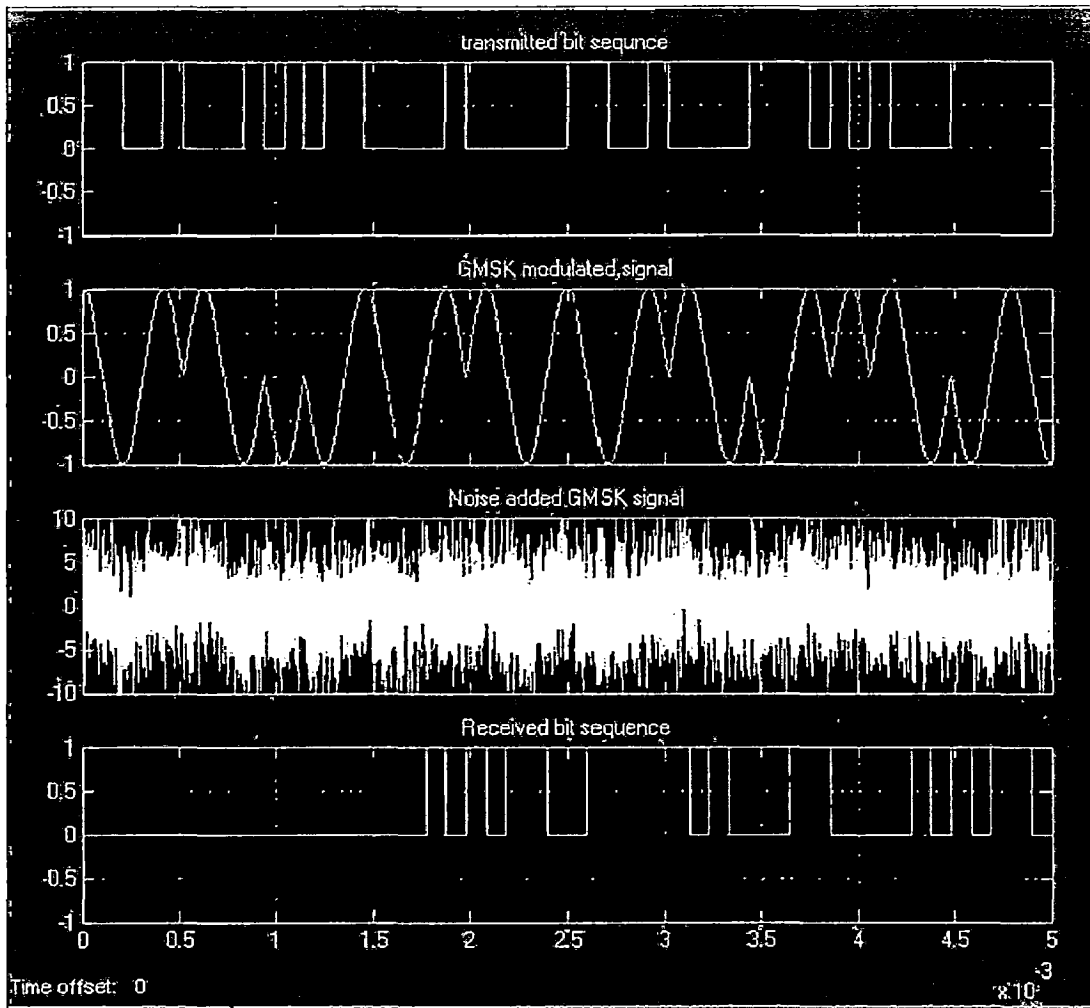


Fig.4.16: Simulation results of GSM Modulation and Demodulation

Block de-interleaver executes reverse process of block interleaver. Received data is stored by inter-column permutation patterns, then de-interleaver outputs data one by one from the top of row to underside row. It acts on the output of the interleaver and puts the symbols back into the original order. The process of deinterleaving is the integral part of the whole interleaving process used in the GSM system.

It reduces the bit rate of the incoming signal, as it removes the redundancy from it and convert it in the signal, we got after block interleaver in the transmitter side. The Fig.4.17

below shows the full rate channel decoding process in the receiver side of the GSM system.

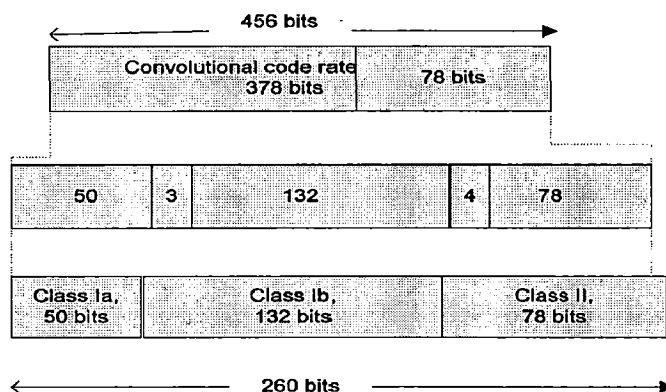


Fig.4.17: Full rate Channel decoding [18]

**Operation:**

The block interleaver/deinterleaver operates in discrete mode with a single -port memory used as a buffer. The symbol transmission consists of an alternating sequence of write and read cycles. Each cycle delay is equal to the buffer size, which is the block length multiplied by the span delay. The total cumulative delay from the transmitter to the receiver can be calculated using the following equation:

$$2 \times \text{number of rows} \times (\text{number of columns} + 4)$$

The block interleaver/deinterleaver uses single-port SRAM memory configured as a matrix of  $n$  rows by  $m$  columns to perform interleaving. During the write cycle, the input symbols are written column by column; during a read cycle, the output symbols are read row by row. The column length is usually equal to the codeword length of the FEC encoder, while the numbers of rows (often called the span) is the interleaver delay. The figures below illustrate block function operation using a 6-symbol codeword.

The channel coding includes not only the classical (forward error correction) block and convolutional codes, but also modulation and any combination of them, such as concatenated coding and modulation.

Most block or convolutional codes are designed to combat random, independent errors usually occurring in a channel without memory. For channels with memory such as mobile channels, burst channel errors are observed due to fading which varies depending on mobile speed, propagation delay spread and frequency. Interleaving (here interleaving means interleaving and de-interleaving both) is deployed to disperse the burst errors when the received signal level fades, and to reduce the concentration of the errors that must or should be corrected by the channel code. The basic de-interleaving process is shown below.

The complete block of 912 bits received, is divided into 8 bursts each of containing 114 bits. These 114 bits of each burst are consecutively placed in 2 different sub blocks of 57 bits each. Same process is repeated for every burst. Then speech message sub block is read into an 8-column by 57-row matrix RAM, filling each row in turn. After that all these bits read column wise and placed in a single array of 456 bits, this gives the original message signal.

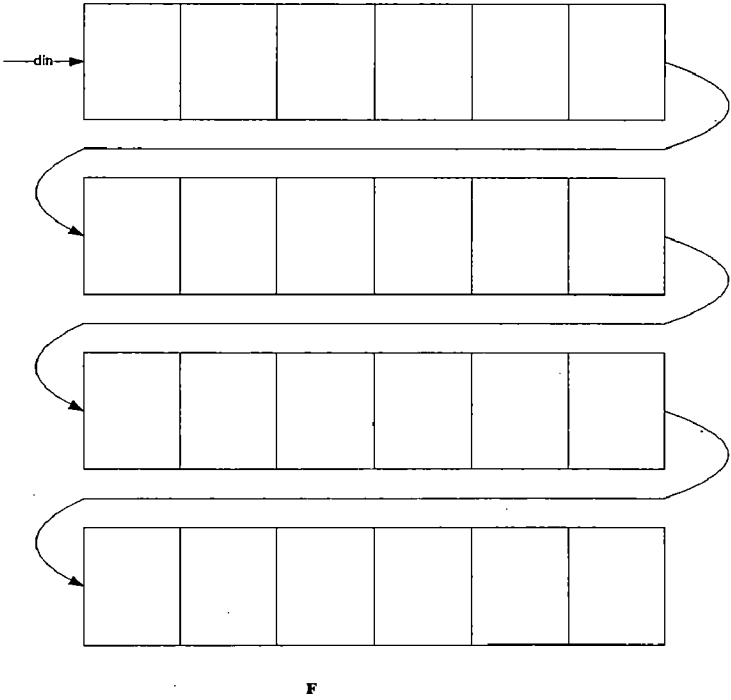


Fig.4.18(a): Read cycle of De-interleaving process

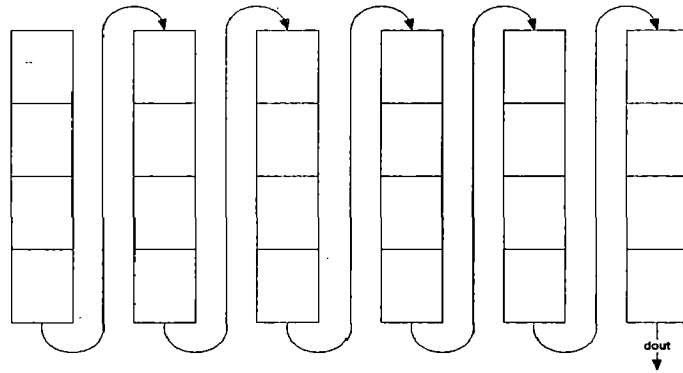


Fig.4.18(b): write cycle of De-intelaving process

### Simulation Results:

The simulation result is taken before the synthesis of the GSM Block De-interleaver module on the Xilinx – ISE software. The pre-synthesis simulation result gives us the solid idea about our module; it is working fine or not. We can check the most of the functionality of our module in the simulation before the synthesis. But there are some drawbacks also of this simulation before the synthesis. This can never tell us about the usage and the availability of the sources on the FPGA. The Figure 4.20 given below displays the presynthesis result of the GSM Block De-interleaver module designed on ModelSim.

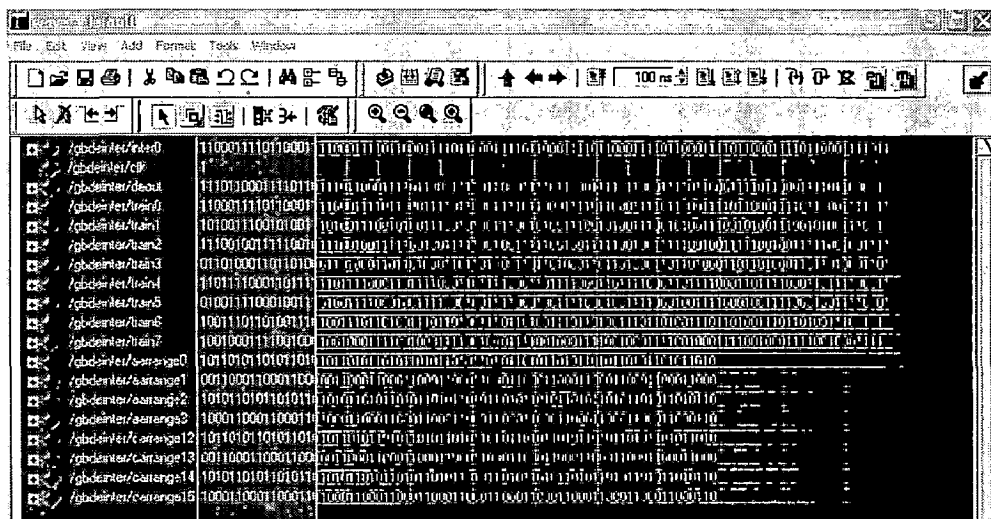


Figure.4.19: simulation result of the Block deinterleaver module

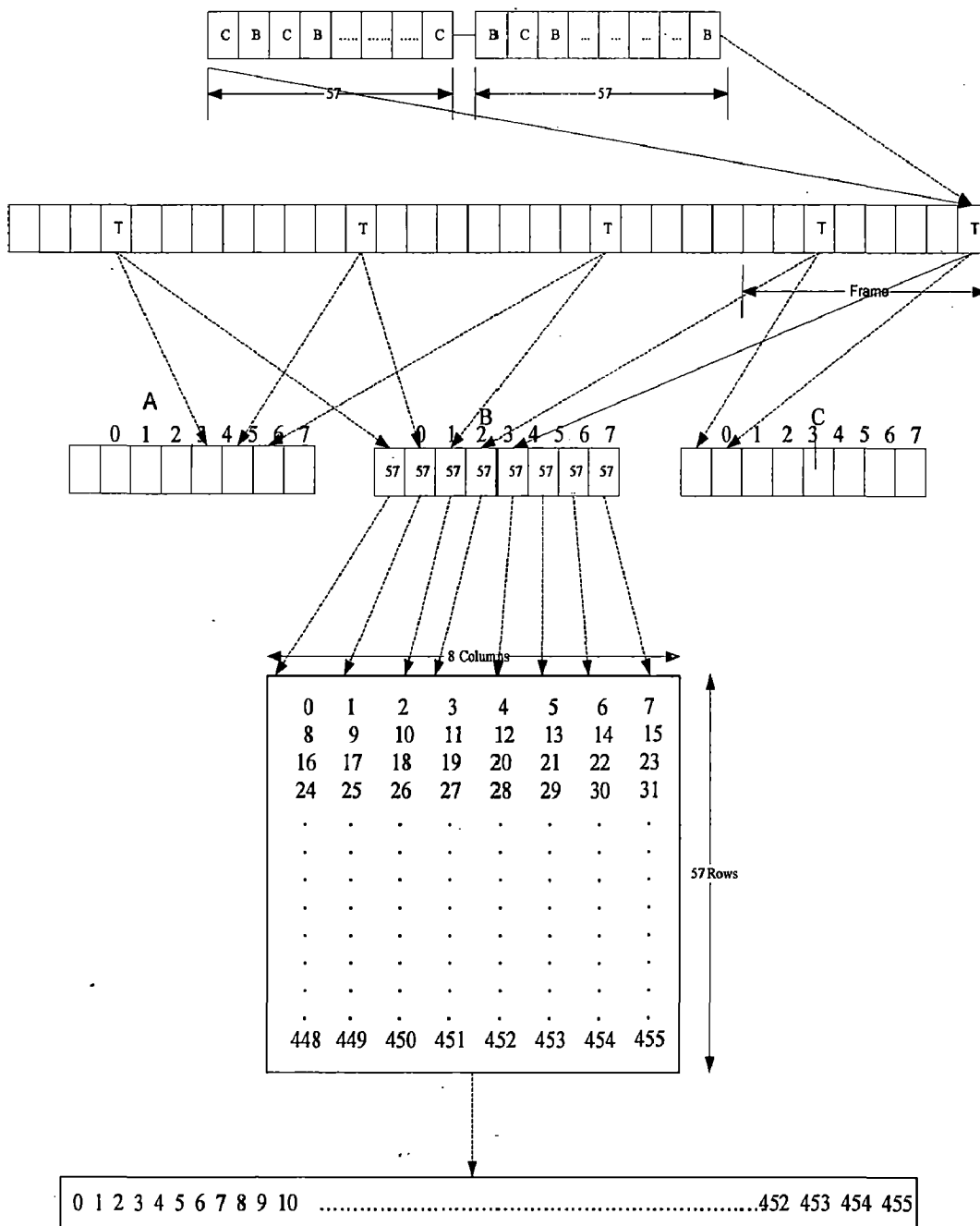


Fig.4.20: Complete De-Interleaving Process

### 4.3.3 Decryption:

The same A5 algorithm is used in receiver to generate the same key stream which is generated by transmitter and the interleaved data is retrieved. This data applied to the deinterleaver gives the required encoded sequence then we go for the decoding phase. Simulation waveform is shown in fig. below. Simulation result is shown in fig.4.21.

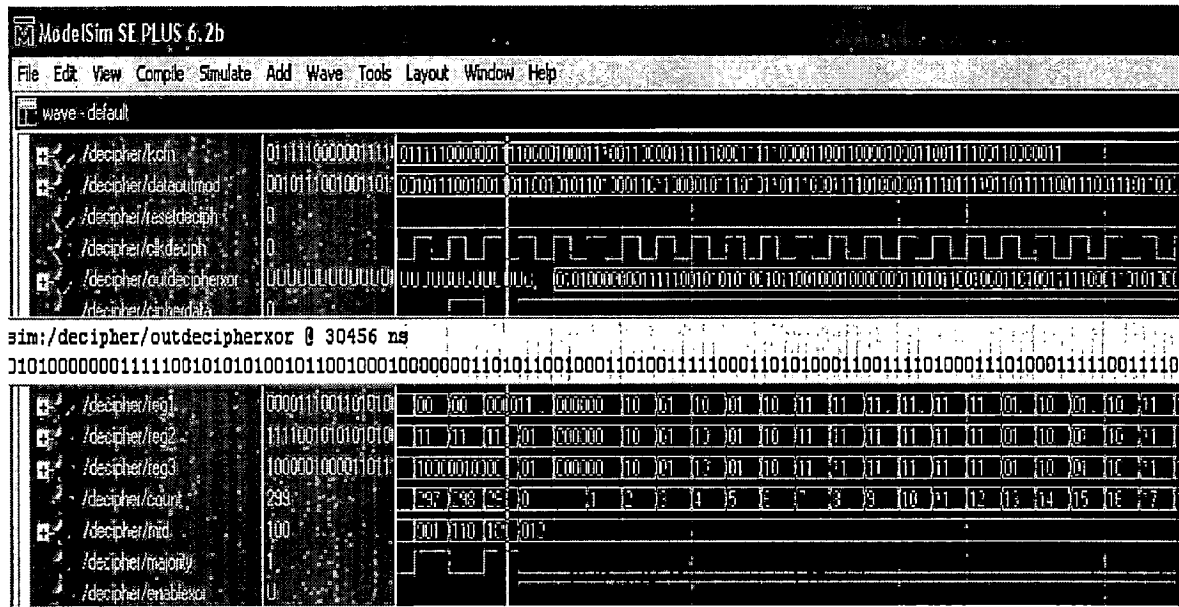


Fig.4.21: Simulation waveform of De-ciphering block

### 4.3.4 Viterbi Decoder:

Viterbi algorithm is commonly used to decode convolutional codes. It is the maximum likelihood decoding technique to decode convolutional codes and involves searching the entire code space for the codeword which most closely resembles the received sequence. This algorithm uses trellis to represent states, output codeword's along with time history of states. We may use tree diagram for the same need .The coded sequence corresponding to an information sequence of length L bits, we have to show  $2^L$  branches.

The number of branches grows exponentially, and, hence, the tree approach must be avoided. Four Steps to get required information sequence:

Step 1: Calculate branch metrics at each of states at different times.

Step 2: Accumulate them through each path ending at different nodes and at different paths to calculate the path metric at times  $t = t_1; t_2; t_3 \dots$  etc.

Step 3: After calculating branch and path metric surviving paths at times  $t = t_1, t_2, t_3 \dots$  etc., which is the path that traversed from chosen state to a particular state at particular time with total Hamming distance is minimum over the paths traversed from the chosen state to that particular state to be calculated.

Step 4: At the final stage, we end up with the initial state and only one surviving path that traversed the states with minimum path metric.

In the event that the path metrics of merging paths are equal, a random choice can be made with no negative impact on the likelihood. The path stored at the right-most node in the trellis diagram is the maximum likelihood path through the trellis diagram and represents the most likely sequence to have been transmitted given the actual received sequence. This algorithm has been implemented from standard logic cores which are provided in Xilinx ISE.

The output the viterbi decoder is sequence of 189 bits. These 189 bits are again divided into two groups i.e, 53 bits to get 50 class Ia bits and 136 bits to get 132 of class Ib bits. These bits will be added with 78 bits of class II to get the transmitted sequence of 260 bits. CRC detector will detect the 50 bits from the applied 53 bits (reverse operation of CRC generator).

#### **4.3.5 CRC Remover:**

The CRC generator will take message string  $M$  as input and divide it by a key word  $k$  that is known to both the transmitter and the receiver. The remainder  $r$  left after dividing  $M$  by  $k$  constitutes the "check word" for the given message. The transmitter sends both the message string  $M$  and the check word  $r$ , and the receiver can then check the data by repeating the calculation, dividing  $M$  by the key word  $k$ , and verifying that the remainder is  $r$ . The CRC detector and generator have been implemented by a LFSR. The basic purpose of this CRC generator is indication of quality of frame. After combining three

classes of data to desired data rate speech decoder will decode the data. So we receive the transmitted data. This completed the transmitting and receiving operations.

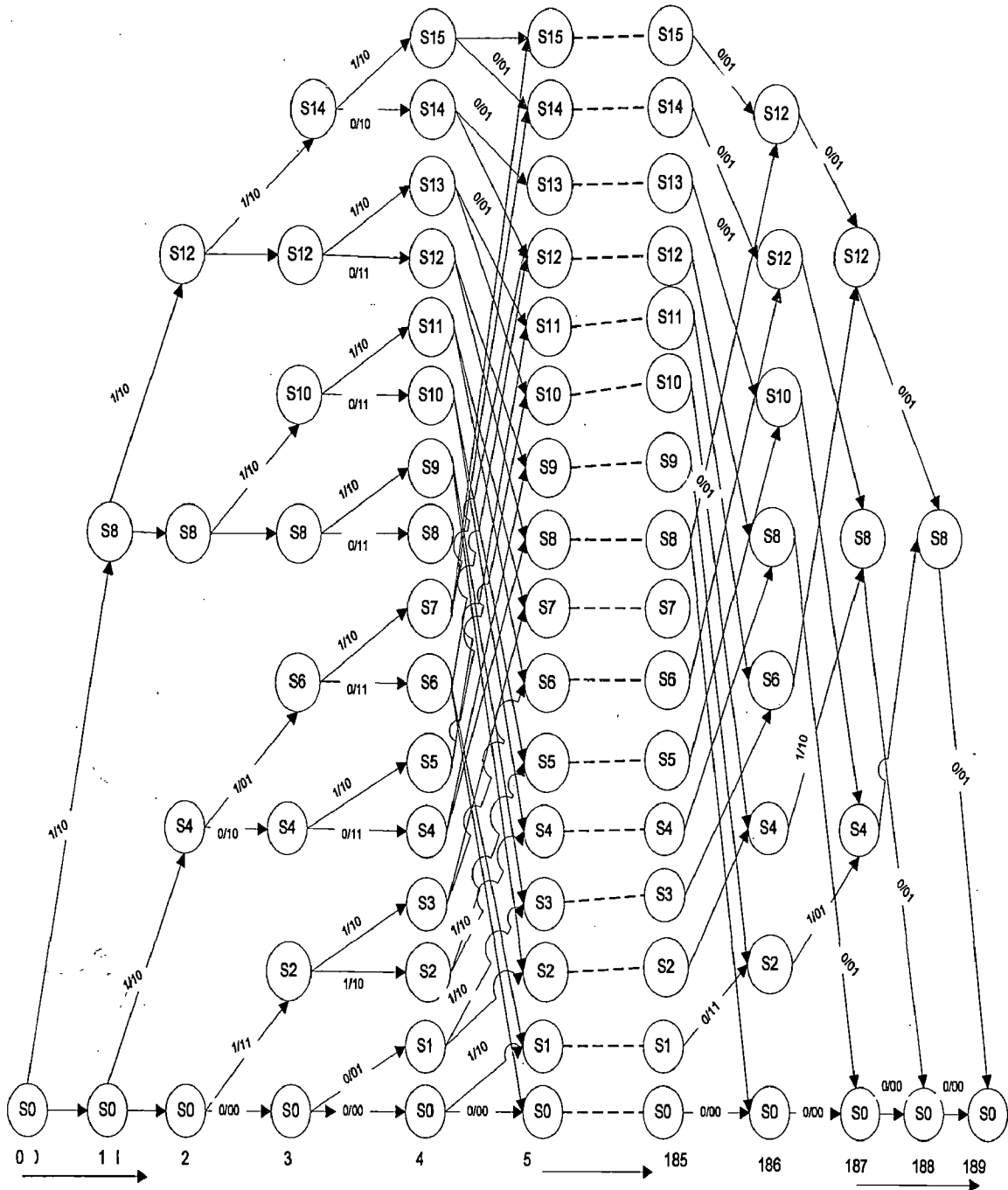


Fig.4.22: Trellis diagram for (2,1,4) convolutional encoder



## CHAPTER 5

### IMPLEMENTATION DETAILS OF GSM BASEBAND PROCESSING ON FPGA

---

In recent years field programmable gate arrays (FPGAs) have become key components in the implementation of high performance digital signal processing (DSP) systems, especially in the areas of digital communications, networking, video and image processing. The logic fabric of today's FPGAs consists not only of look-up tables, registers, multiplexers, distributed and blocks memory, but also dedicated circuitry for fast adders, multipliers and I/O processing.

Initially I have simulated the GSM baseband processing using Matlab Simulink. After finding the correct results in Simulink, I have developed the two versions of GSM transmitter and receiver, one is using ModelSim 6.2d and the other is using Xilinx system generator tool. The aim of developing two versions is to get the area efficient and power,delay efficient models. Depending on the requirement any one of model is utilized using dynamic partial reconfiguration technique.

#### **5.1 Matlab Simulink Model:**

The GSM transmitter and receiver model was designed using blocks available in Matlab Simulink [22]. In this model I have used Bernoulli random generator as a source for data generation with a data rate of 9.6 kbps and frame size of 260 bits like in GSM speech signal .The GSM transmitter block consists of channel coder, convolutional encoder, interleaver, data burst, GMSK modulator blocks. The GSM receiver block consists of GMSK demodulator, burst separation, de-interleaver, viterbi decoder, channel decoder blocks. The fig.5.1 shows the Simulink model of GSM transmitter and receiver.

#### **5.2 Simulation Results**

The results obtained from the simulation is presented and discussed in this section. The bit error rate (BER) ,constellation diagram and eye diagram of the modulated signal from which conclusions about the modulated signal can be drawn is observed at the output of the channel. These diagrams reveal the modulation characteristics of the signal and help to depict the impact of impairments, such as pulse shaping or channel distortions. They

are commonly used to evaluate the overall performance of the digital communication systems. Since the channel used in this thesis is AWGN, the extent to which the noise has affected the modulated signal can be seen from constellation and eye diagrams.

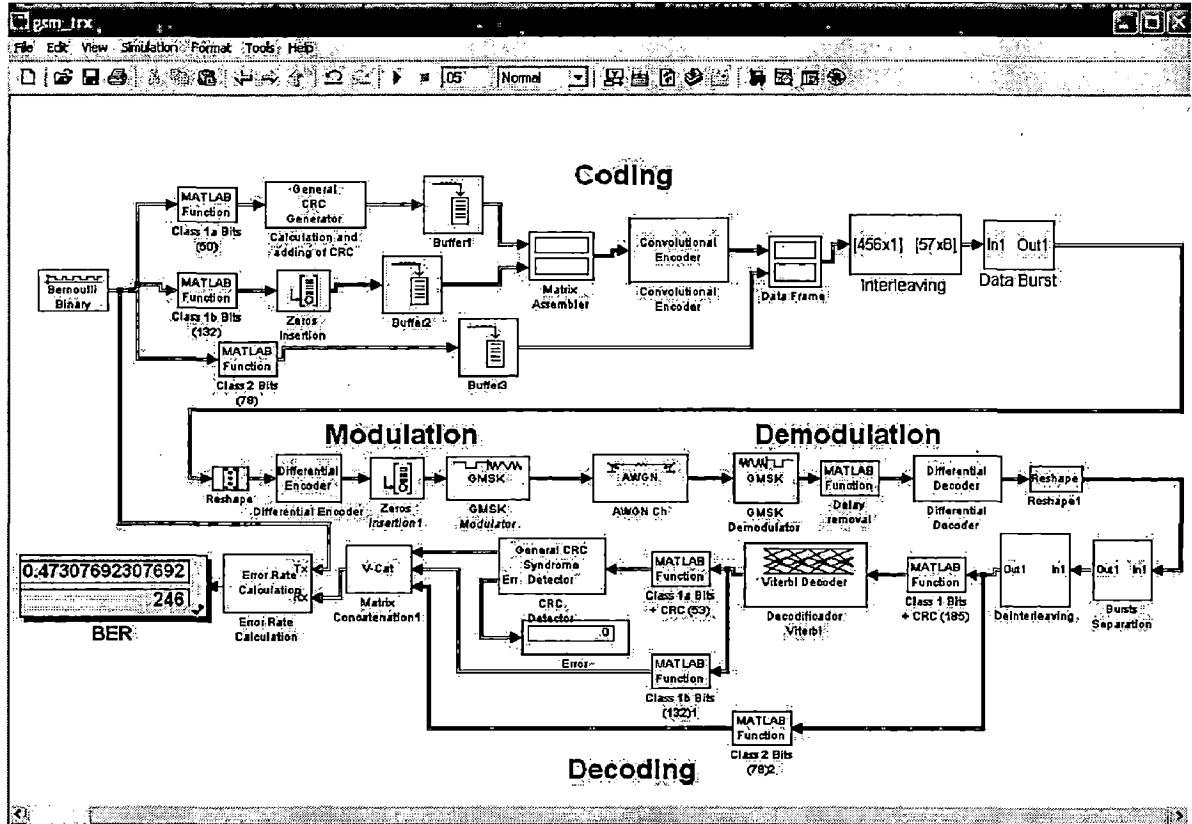


Fig 5.1: Matlab Simulink model of GSM transmitter and receiver

### 5.2.1 BER Performance

GMSK modulator, demodulator performance is measured by calculating BER with BT=0.3,0.5. BER performance is measured using the following equation.

$$E_b/N_0 \text{ dB} = (S/N) \text{ dB} - (10 \log(K)) \text{ dB} + (10 \log(f_s/f_b)) \text{ dB} \dots \dots \dots (4.1)$$

The fig.5.2 shows the plot of BER vs  $E_b/N_0$ .

### 5.2.2 Constellation Diagrams

Figure 4.8 shows the constellation diagram of the modulated signal with signal-to noise ratio (SNR) of 15 dB. Increasing the SNR of the AWGN channel will increase the

performance of the system. The Constellation diagram shows the GMSK signal has constant envelope. The constellation diagram of modulated signal with SNR of 20 dB is shown in fig.5.3.

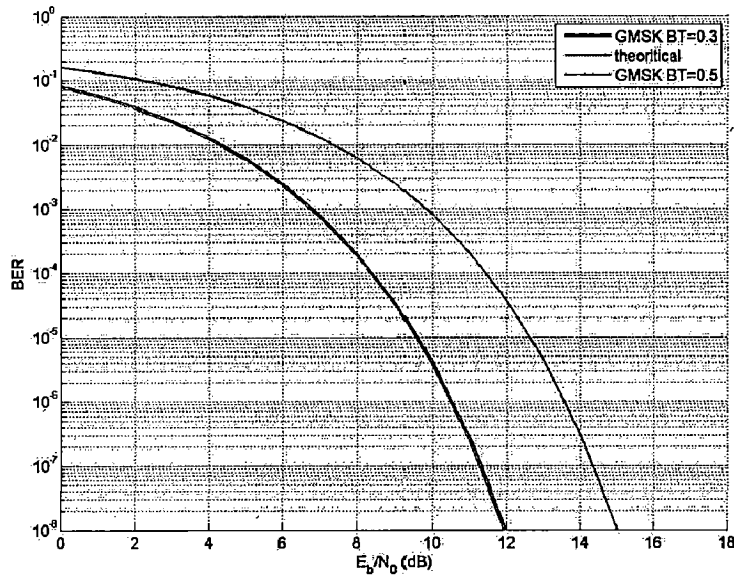


Fig 5.2: BER vs  $E_b/N_0$  plot for BT=0.3,0.5

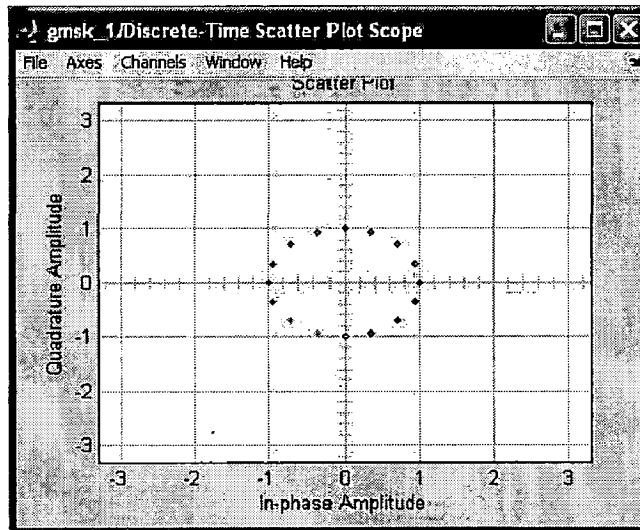


Fig.5.3: Constellation diagram of GMSK with SNR=15dB

### 5.2.3 Eye diagrams:

The width of the eye provides information about tolerance to jitter, and the height of the eye gives information about tolerance to additive noise. Eye closure (inadequate width or

height) is probably due to ISI. The fig.5.4 shows the Eye diagram of GMSK with SNR of 15dB.

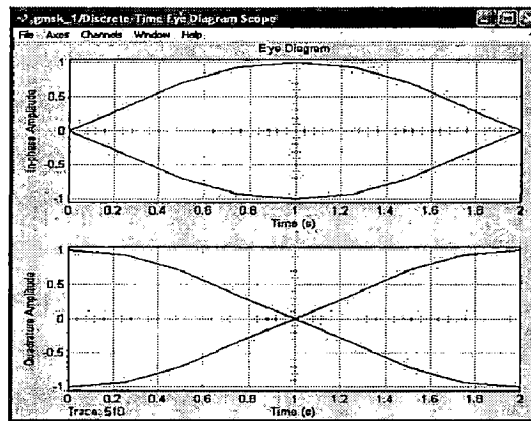


Fig.5.4: Eye diagram of GMSK with SNR=15dB

**5.3 FPGA Design Flow** The following fig.5.5 shows the FPGA design flow using VHDL coding/ Xilinx system generator block set.

#### ***Tools used***

- The following softwares and hardwares were used in this thesis.
- MATLAB Simulink R2006b
- Xilinx System generator
- ModelSim
- Hardware Descriptive Language (HDL)
- C++ Programming
- Xilinx ISE 9.2i
- XUP Virtex-4 [13] FPGA kit

#### ***Synthesis***

Synthesis is the process by which abstract design descriptions are reduced into a lower level circuit representation, such as netlists or equations. HDLs provide the input and output of hardware synthesizers. Floating point arithmetic modules are synthesized and their netlists has been generated.

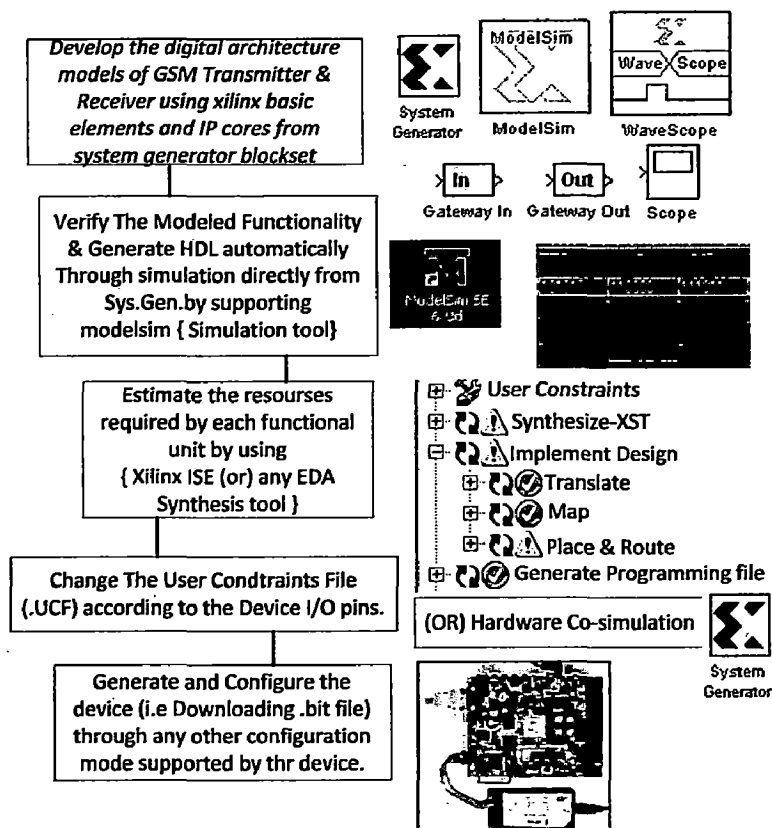


Figure.5.5: FPGA Design Flow

## Implementation

Implementation is the process in which a logical design is converted into a physical file format that can be downloaded to the selected target device. Floating point arithmetic modules are implemented.

### 1. Translate

The Translate process merges all of the input netlists and design constraints and outputs a Xilinx native generic database (NGD) file, which describes the logical design reduced to Xilinx primitives.

### 2. Map

The Map process maps the logic defined by an NGD file into FPGA elements, such as CLBs and IOBs. The output design is a native circuit description (NCD) file that physically represents the design mapped to the components in the Xilinx FPGA.

### 3. Place and Route

The Place and Route process takes a mapped NCD file, places and routes the design, and produces an NCD file that is used as input for bit stream generation.

#### ***Programming File Generation***

The Generate Programming File process produces a bit stream for Xilinx device configuration. After the design is completely routed, you must configure the device so it can execute the desired function. Fig 4.3 shows the Xilinx ISE 9.2i Window on which synthesis process has been carried out.

#### **5.4. Simulation using ModelSim 6.2d tool:**

The design of individual baseband modules has been done using VHDL code. Simulation has been done using Modelsim6.2d. All the individual modules have been integrated to form transmitter and receiver model. The following fig.5.6 is simulation of complete baseband processing model.

#### **5.5. Simulation using Xilinx system generator9.2i tool:**

Xilinx system generator is integrated with Matlab Simulink. So system generator blockset is available as part of matlab Simulink. By choosing blocks available in Xilinx blockset, available IP(Intellectual Property) cores from Xilinx, GSM transmitter & receiver model was designed. The main steps of the Xilinx system generator design are explained below.

#### **System Generator Model of GSM Transmitter & Receiver:**

The following fig. shows the Xilinx system generator model of a GSM transmitter and receiver. The following model was designed by using the Xilinx IP cores available in system generator blockset. The functions which are not available are developed by using the Xilinx black box concept. A piece of code was written in VHDL and those are inserted in black box to get the desired functionality. Fig 5.7 shows system generator model of GSM transmitter and receiver.

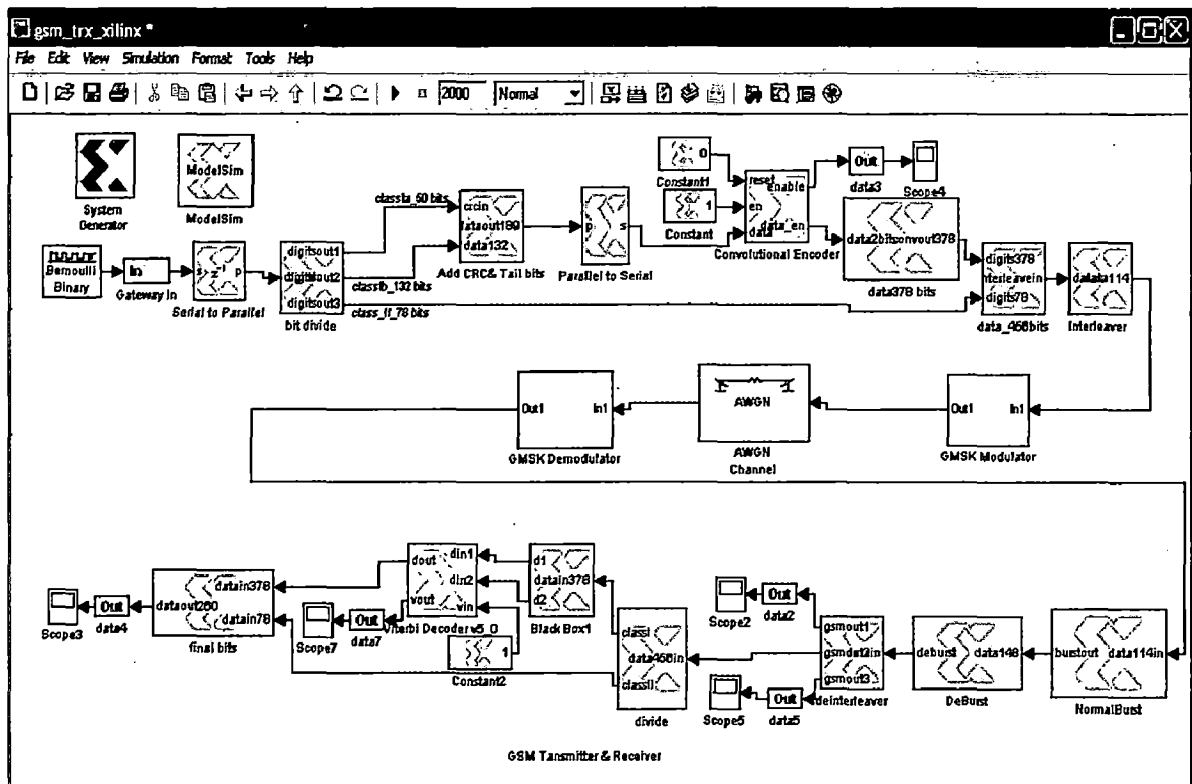


Fig.5.6: System Generator Model GSM Transmitter & Receiver

The ModelSim tool was integrated into the Xilinx system generator. So it is possible to observe the simulation waveforms of the Xilinx system generator in ModelSim. The following waveform shows the simulation waveform of Xilinx system generator model.

### Hardware Co simulation

The System Generator will automatically synthesize, and place and route the design on the target FPGA platform upon selecting the appropriate options, such as compilation type, target FPGA, synthesis tool, and so on. The key steps in the hardware co-simulation process can be summarized as follows [38]:

- (1) The hardware co-simulation platform can be chosen from the System Generator dialog box. When the compilation target is selected, the fields on the System Generator dialog box are automatically configured with settings appropriate for the selected compilation target. The fig shows the appropriate settings for bitstream generation.

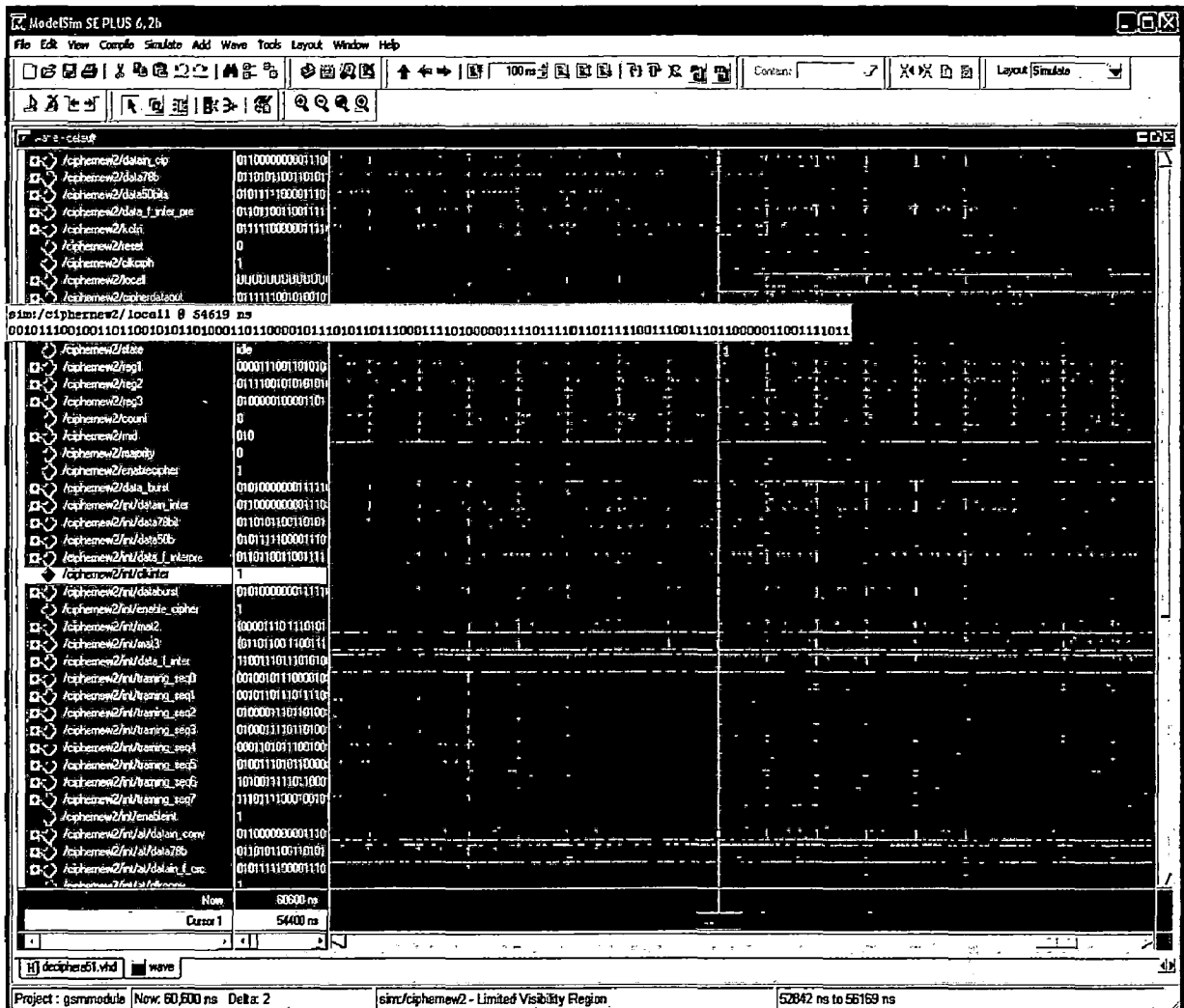


Fig.5.7: Simulation window of GSM Transmitter & Receiver

(2) After initiating the “Generate” button, the code generator is invoked and produces an FPGA configuration bitstream for the design that is suitable for hardware co-simulation. System Generator not only generates the HDL and netlist files for the model during the compilation process, but it also runs the downstream tools necessary to produce an FPGA configuration file.

(3) After the FPGA configuration bitstream is created, a new hardware co-simulation block is created by the System Generator and stored in the MATLAB SIMULINK Library. Hardware co-simulation blocks can be used in the design with other MATLAB



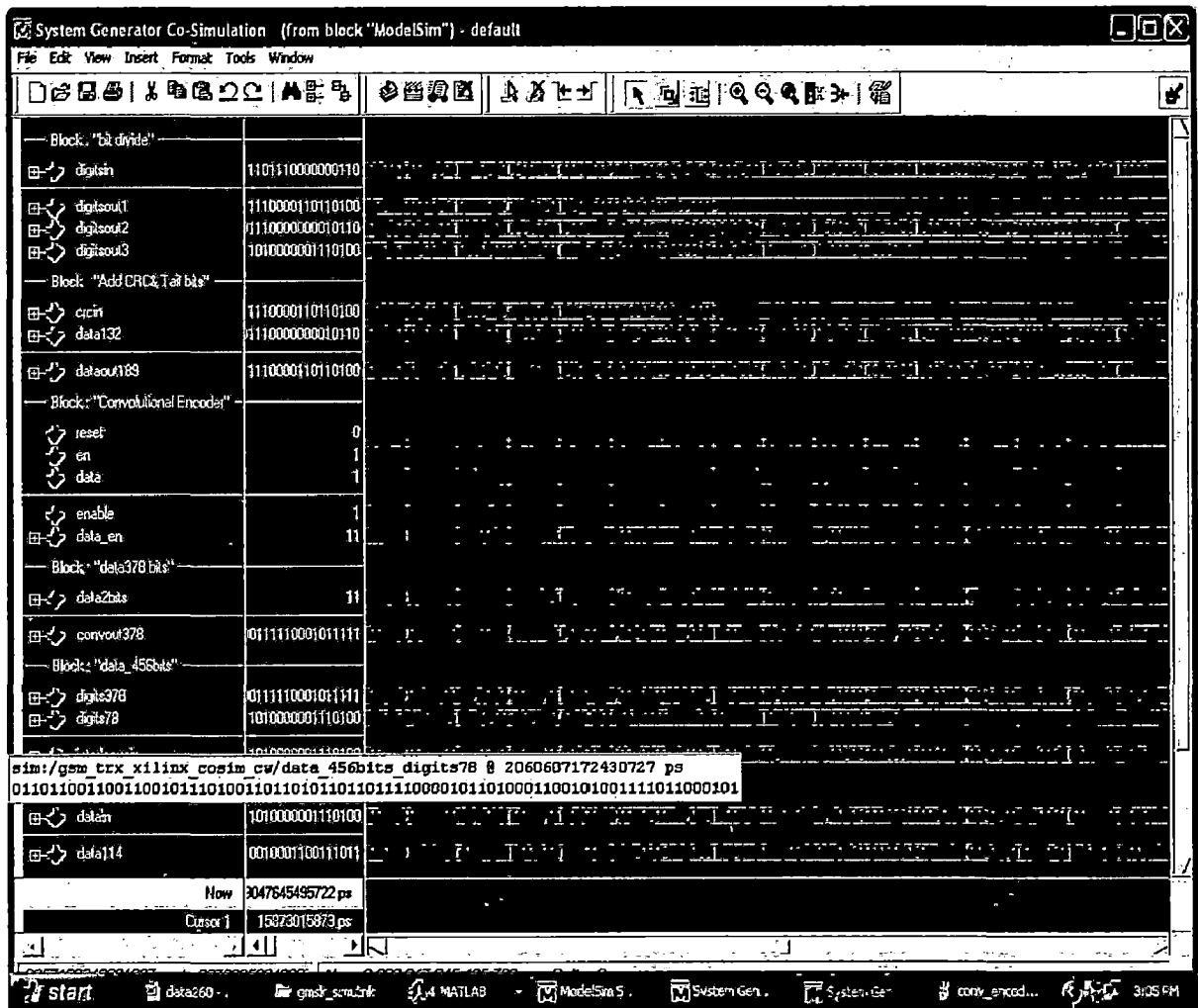


Fig.5.8: Simulation waveform of Xilinx system generator model

Simulink blocks. When the hardware co-simulation block is simulated, it interacts with the underlying FPGA platform and facilitates the design implementation and verification of the desired FPGA. In this thesis, only hardware co-simulation is performed using Virtex-4 FPGA [8].

### 5.6 Implementation of GSM Baseband processing on Xilinx Virtex-4 xc4vfx12ff668 -10 Device

By doing synthesis of baseband processing modules .bit files which can be loaded onto FPGA are obtained. The 260 bit input is applied to GSM Transmitter, and 260 bit output is received at GSM receiver. After completion of synthesis, place& route, bit file has been generated, which when downloaded to the FPGA determines the FPGA's behavior.

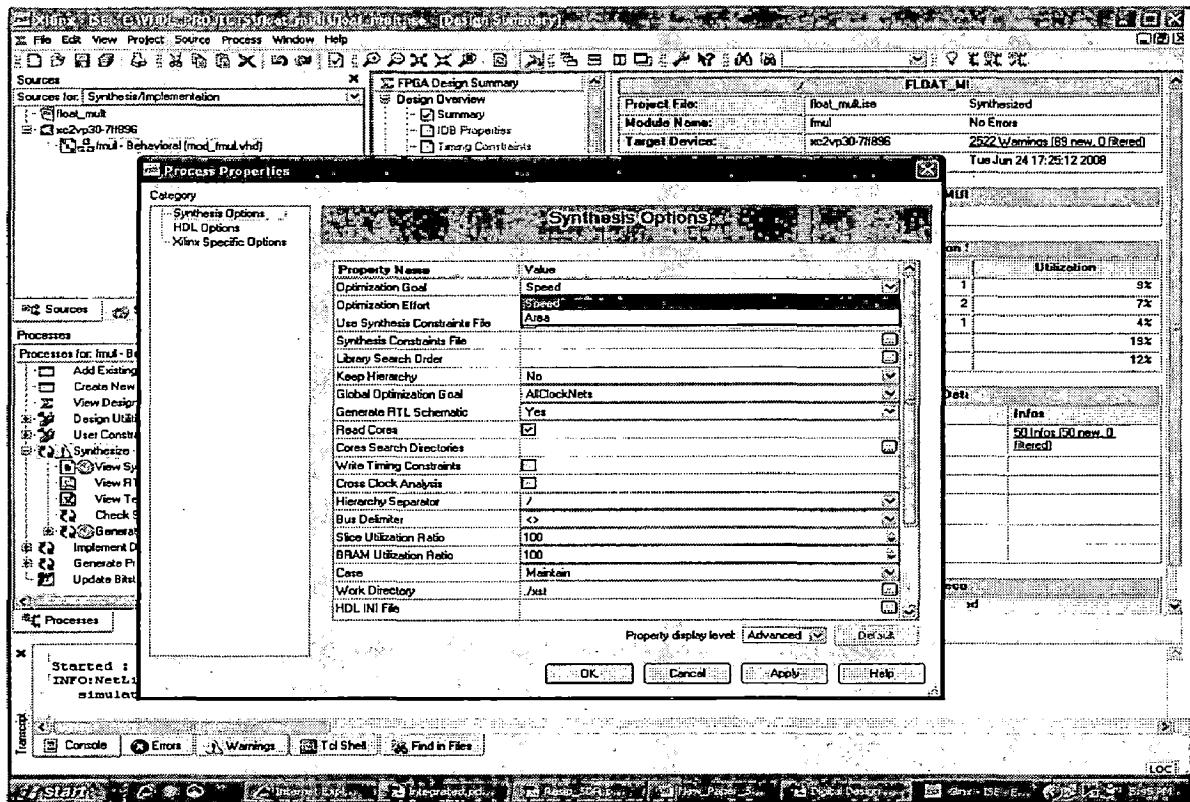


Figure.5.9: Xilinx ISE 9.2i Window

A piece of C++ code is written to control the 260 bit data. Controller will generate an acknowledge signal when the data is 260 bits. The software code downloads the bit file to the FPGA and, using the PCI bus, transfers all the operand data for the application to the card's SRAM banks. Once the data transfer is complete the FPGA design is ready to start. The host computer signals the FPGA, using the status register, to start it running. The status register is a blocking communications system, between the host and the FPGA, that can be used to synchronize the host and FPGA. Immediately before signaling the FPGA the host computer records the current value of the system clock. Once the FPGA has finished running, it signals the host program, and immediately after receiving the signal the host program records the value of the system clock for a second time; the difference in the recorded times is the execution time [6] of the program. After execution of the program we can see 260-bit output stored in the specified location in pc.

### 5.7 Comparison of two GSM Baseband Processing versions.

The following table gives the comparison of performance of two GSM baseband processing versions:

Target Device xc4vfx12ff668 -10	GSM	
	Version1	Version2
Device utilization		
Number of slices(5472)	2298	2845
Number of slice flip flops(10944)	1102	1532
Number of 4 input LUTS(10944)	1203	1870
Number of IOs	262	262
Number of bonded IOs(320)	262	262
Number of BRAMs	17	22
Number of GCLKs(32)	1	1
Delay (ns)	12.032	9.247
Throughput (Gbps)	21.60	28.11

Table 5.1: Comparison of GSM baseband processing versions

#### Analysis of Results:

From the above Table 5.1, it is clear that version1 is taking less silicon area but speed is less, where as in case of version2 silicon area is more but speed is high. The main reason for high speed with more silicon area is parallel processing of FPGA. Since the delay of GSM\_v1 is 12.032 ns, the throughput is 21.60 Gbps and for GSM\_v2, it is 28.11 Gbps because the delay is only 9.247 ns. Detailed device utilization is given in Appendix-A.

## CHAPTER 6

### PARTIALLY RECONFIGURATION OF GSM BASEBAND PROCESSING VERSIONS AND RESULTS

In this chapter dynamic partial reconfiguration implementation details and results are presented. Two versions of GSM baseband processing bit streams (i.e., area efficient code and power efficient code) generation is explained in chapter 5. These two versions are used as partially reconfigurable modules. The detailed dynamic partial reconfiguration flow has been explained in chapter 2.

Partial reconfiguration is useful for systems with multiple functions that can time-share the same FPGA device resources. In such systems, one section of the FPGA continues to operate, while other sections of the FPGA are disabled and partially reconfigured to provide new functionality. Partial Reconfiguration is supported by the devices which can be configured after its manufacture; FPGA is an example of such a device. So we implement partial reconfiguration on FPGA. Further the large gate count of FPGAs made them suitable for design of GSM baseband processing.

#### 6.1 Modules Creation for Partial Reconfiguration

Creating a partial reconfiguration design requires the creation and implementation of the design within a set of specific guidelines.

Table.6.1: PR Directory Structure [12]

	<b>base</b> Implementation directory for the static portion of the base design (i.e. everything except the PRMs).
	<b>merges</b> PRMs are merged with the base design in the merges directories. A separate subdirectory is required for each merge.
	<b>non_pr</b> Non-pr versions of the design are fully implemented for initial system design and test.
	<b>reconfigmodules</b> Each PRM is implemented in a separate directory.
	<b>synth</b> HDL for the top level, the base design, and each PRM is synthesized in the appropriate directory.
	<b>top</b> The top level netlist is translated in a separate directory. The UCF file goes here.

UG208\_02\_01\_191225

The partial reconfiguration flow utilizes a modified form of the Xilinx Modular Design process. All the logic that should be partially reconfigured should be arranged in folders. Recommended directory structure is as shown in figure below. Fig 6.1 shows the folders that should be present in PR design and also the files that should be included in them.

**6.1.1 Design of Partially Reconfigurable GSM baseband processing Modules:** we have designed a partially reconfigurable system which includes two partial reconfigurable regions. In this partial reconfigurable region hardware resources are time-shared between two baseband processing versions. Fig 6.1 shows the run time partial reconfiguration process.

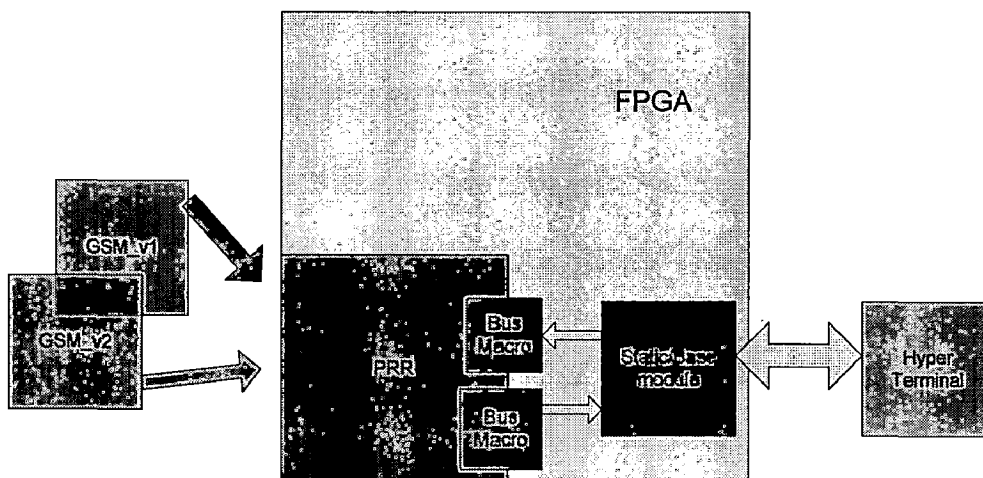


Figure 6.1: Run-time and Partial reconfigurations of GSM baseband processing versions.

### 6.1.2 Design of Static module

Static Region or Base Region which contains the common logic which will not change between the PR modules. In this design the static module is used to display the transmitted and received bit sequence.

### 6.1.3 Design of Bus Macro:

Xilinx provides these bus macros through their web page[ [www.xilinx.com](http://www.xilinx.com) ] and has one bus macro for every family of FPGAs. In this example the bus macro for Virtex-IV is used. An example bus design was given for eight tri state buffers set up in an

arrangement that allows four bits of information to travel either left-to-right or right-to-left, using one TBUF long line per bit, see Figure 6.2.

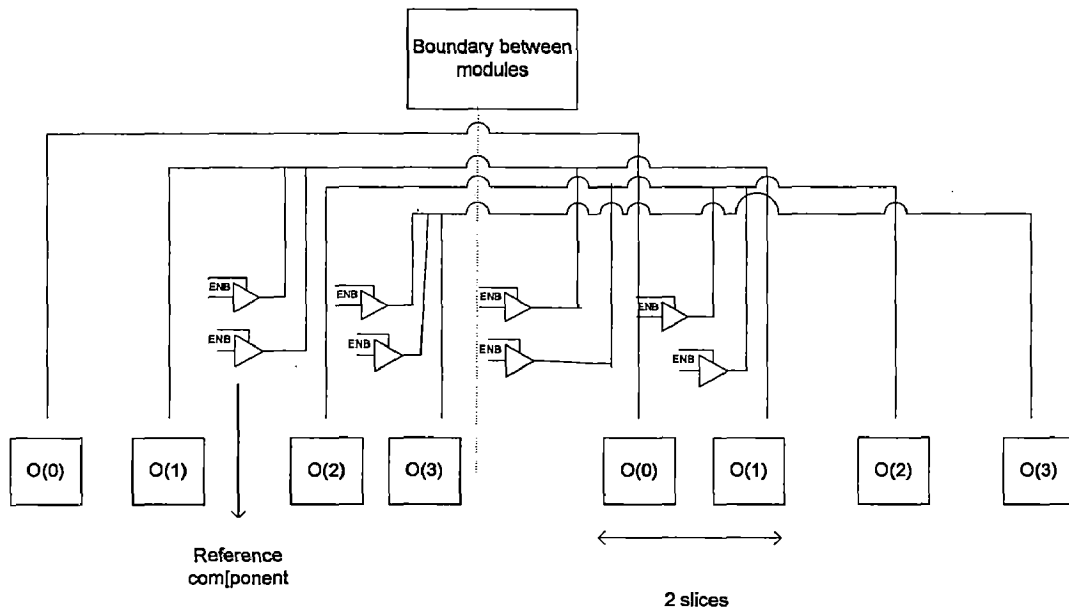


Fig 6.2: Bus macro design

## 6.2 Implementation flow in command line using ISE 10.1i [41]

1. **Built Flat Design:** This design helps us to functionality of the design.
2. **Synthesize all modules including RM:** synthesize all lower-level modules with I/O buffers insertion OFF and synthesize the top-level with I/O buffers insertion selected.
3. **Build Top-Level Design:** This will generate top.ngd file which will be used in next step.
4. **Build Static Design:** This will generate top\_routed.ncd static.used files among other files. The top\_routed.ncd file contains the implemented static design. The static.used file contains routes used by static logic in PRR, the information needed during the RM implementation step.
5. **Build RM Design:** This will generate top\_routed.ncd file among other files.
6. **Assemble Static Design:** In this we will assemble the static modules as well as desired one RM for each PRR into a design that will be loaded when the FPGA is

configured. This will generate static\_full.bit file which contains adder and right shift operations. It will also generate ag\_reconfig\_leds\_blank.bit and ag\_reconfig\_GSM\_blank.bit files which can replace the LEDs and GSM PRR with blank logic.

**7. Generate Partial Bitstreams:** In this step we will generate partial bitstreams. The PRR can be reconfigured instead of the entire FPGA with bit-streams of individual reconfigurable modules i.e GSM\_v1.bit, GSM\_v2.bit files.

**8. Testing:** Use static\_full.bit file to program FPGA and then verify the functionality using partial bit streams. We can notice that programming is very quick reducing the reconfiguration time.

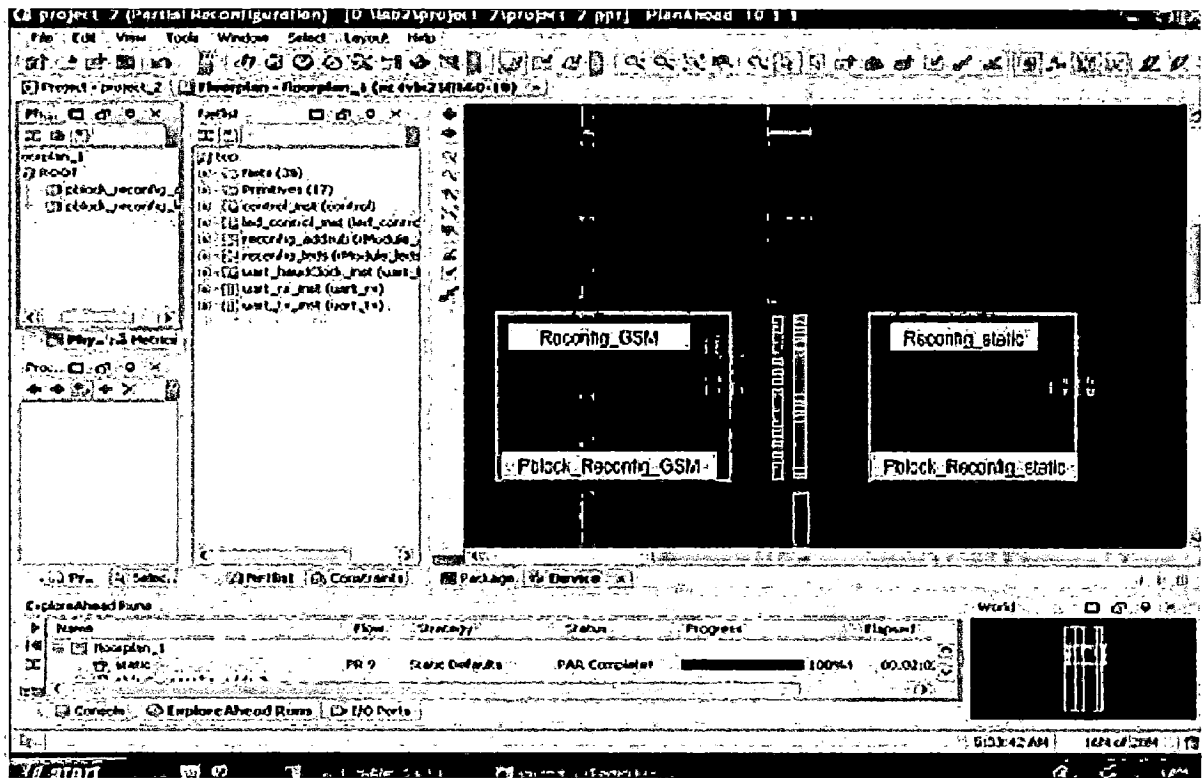


Figure 6.3 PlanAhead window showing two Partially Reconfigurable Regions

Fig 6.3 shows how partially reconfigurable regions are described on FPGA. It also shows the bus macro placement over right corners of partially reconfigurable regions

## CHAPTER 7

### CONCLUSIONS AND FUTURE SCOPE

---

#### 7.1 Conclusions:

In this dissertation report, the design of GSM baseband processing for reconfigurable software defined radio has been presented and implemented on modern virtex-4 FPGA platform.

Initially, the GSM transmitter and receiver are designed using VHDL code and then second version of same was designed using Xilinx system generator tool. The first version is area optimized and the second version is speed optimized. After generating two versions of GSM baseband processing, partial reconfiguration was done on same virtex-4 FPGA platform to save the area of FPGA.

Two GSM baseband processing versions are developed and comparisons are given in Table 5.1. Also the proposed versions are compared with existing baseband designs presented in [7], [8], [9]. The following conclusions can be drawn from this thesis:

- The device utilized the 42%, 52% of the available resources for GSM\_v1, GSM\_v2 versions where as the baseband model developed in [7] occupies 70% of the resources of the virtex – 4 FPGA platform.
- The average delay for slice is 12.032ns and 9.247 ns in case of GSM\_v1 , GSM\_v2 respectively. The average delay of slice delay is 18.64 ns in [7].
- The BRAM blocks used are 25 for the model developed in [7]. These are 17 and 22 in case of GSM\_v1, GSM\_v2.
- The operating frequency of FPGA hardware are 50 MHz and it is only 13 MHz in [7]
- The floor plan view of partial reconfiguration of two baseband processing version was presented which was useful to reduce the required hardware during run time.



## **7.2 Future Scope:**

In this dissertation baseband processing section of software defined radio has been selected for implementation, however, it can be extended to full software defined radio by designing RF section and IF section.

The channel is considered to have additive white Gaussian noise. Inter-symbol Interference, Doppler shift, phase error, multipath fading, etc., can be added to the channel, in order to closely simulate real life systems.

The implemented software defined radio is consists of coding, modulation and spreading. This thesis can be extended by adding more functionalities such as equalization, carrier recovery, phase recovery, etc., needed to counteract the noise and interference problems of a communication system.

Further efficient implementation of baseband processing modules can be done by using difference based partially reconfiguration. But this approach requires more redundancy in baseband processing modules. New standards like OFDM, wi-max can also be implemented for software defined radio.

## REFERENCES

---

- [1] Joseph Mitola II. "Technical challenges in the globalization of software radio" IEEE Communication Magazine, February 1999, pp. 84–89,
- [2] J. Reed, "Software Radio: A Modern Approach to Radio Engineering", Prentice Hall Communications Engineering and Emerging Technologies Series, 2002.
- [3] SDR definition accessed from, [www.sdrforum.org](http://www.sdrforum.org)
- [4] M. Cummings and S. Haruyama, "FPGA in the software radio," IEEE Comm.Mag., vol. 37, no. 2, Feb. 1999, pp. 108-112.
- [5] Antoni.L, Leveugle.R, Feher.M, "Using run-time reconfiguration for fault injection in hardware prototypes", in proceedings of 17th IEEE International Symposium, November 2002, pp. 245–256,
- [6] Early access partial reconfiguration user guide, ug 208, available at [www.xilinx.com](http://www.xilinx.com), march 2006.
- [7] David Coons, "FPGA Implementation of GSM Baseband Processor" available at [www.techonline.com/electronics\\_directory/techpaper/193103486](http://www.techonline.com/electronics_directory/techpaper/193103486)
- [8] H.R.Myler, Shabbir, A. Bagasrawala, Naresh V. Narayana, "A Concurrent Processing Approach for Software Defined Radio Baseband Design" IEEE Region 5 and IEEE Denver Section Technical, Professional and Student Development Workshop,2005,pp.20-24.
- [9] Louis belangar, "Developing an SCA GSM Waveform targeted on DSP/FPGA Architecture" Proceeding of the SDR 04 Technical Conference and Product Exposition, 2004, pp. 73-78.
- [10] Muhammad Imran Anwar, Seppo Virtanen, Jouni Isoaho, "A Software Defined approach for common baseband processing", Journal of Systems Architecture vol.54,2008, pp. 769–786.

- [11] White paper, "Software Defined Radio, A Technology Overview,available" at [www.wipro.com/dsp](http://www.wipro.com/dsp)
- [12] "Two Flows for Partial Reconfiguration:Module Based or Difference Based", XAPP290 (v1.2), Xilinx Inc., Sept 9,2004
- [13] Xiaoyao Liang, Athalye.A, Sangjin Hong, "Dynamic coarse grain dataflow reconfiguration technique for real-time systems design", in proceedings of IEEE International Symposium, Vol. 4, , May 2005, pp. 3511 - 3514
- [14] W.H. Tuttlebee, "Software defined radio: facets of a developing technology", IEEE Personal Communication 6 (2), 1999, pp.38-44.
- [15] E. Buracchini, "The software radio concept," *IEEE Commun. Mag.*, vol. 38, no. 9, Sep. 2000, pp. 138-143.
- [16] A. Haghighat, "A review on essentials and technical challenges of software defined radio," *Proceedings of MILCOM*, vol. 1, Oct 2002, pp. 377-382.
- [17] J. Gunn, K. Baron and W. Ruczzyk, "A low power DSP core-based software radio architecture," *IEEE Journal on Selected Areas in Commun.*, vol. 17, no. 4, Apr. 1999, pp. 574-590.
- [18] A. Truter, E. Wolmarans, "A software defined radio architecture with power control for 3GW-CDMA systems", *IEEE comm.,magazine*, 2000,pp. 25-28.
- [19] J. Noll and E. Buracchini, "Software radio - A key technology for adaptive access," *Wireless Commun. and Mobile Computing*, vol. 2, issue 8, Dec. 2002, pp. 789-798.
- [20] K.Moessnar,D.Bouse,D.Griefendref and J,Stamrien, "Software radio and reconfiguration management", *computer comm.* vol.26, issue 1, Jan.,2003, pp.26-35
- [21] K.Solomon Raju, "System level Architectures and Optimal Mapping for Reconfigurable Computing Systems" Ph.d Thesis, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, May 2008.

- [22] K. Bondalapati, V. Prasanna, "Reconfigurable Computing systems", Proceedings of IEEE, vol. 90, no. 7, July 2002, pp.1201-1217.
- [23] S. Halter, M. Oberg, P. Chau, P. Siegel, "Reconfigurable signal processor for channel coding and decoding in low SNR wireless communications" in: Proceedings of the IEEE Workshop on Signal Processing Systems, Cambridge, MA, USA, 1998, pp. 260–274
- [24] Asha K.Merhotra, "GSM System Engineering" Artech House Publishers, 1997.
- [25] Thierry Turetletti, "Towards the software realization of a GSM base station", IEEE Communication Magazine, vol.17(no.4), April 1999,pp. 265-269
- [26] M.Mouly and M.B. Pautet. "GSM technical specifications list", report, Cell and Sys,2004.
- [27] Caesar S. Wong, "A 3 V GSM Baseband Transmitter" IEEE Journal of Solid-State Circuits, Vol. 34, No. 5, May 1999, pp.121-127.
- [28] G.Kostopoulos,N. Sklavos, M.D. Galanis, O. Koufopavlou, "VLSI Implementation of GSM Security: A5/1 and W7 Ciphers", proceedings of the IEEE Workshop on Wireless Circuits and Systems (IEEE WoWCAS'04), Canada, May 21-22, 2004, pp.321-325
- [29] Santosh Shah and V Sinha. Dsp based implementation of gmsk demodulator using costas loop. National Conference on Communication, IIT Guwahati (India), January 2009, pp.223-227.
- [30] J. G. Proakis. Digital Communications. McGraw Hill, third edition, 1995.
- [31] T. Rappaport, "Wireless Communications - Principles & Practice", 2nd edition. Prentice-Hall, Upper Saddle River, NJ, 1996.
- [32] GSM history accessed from,[www.gsmworld.com](http://www.gsmworld.com)

- [33] R.M. Gunther, M.A. Sessler and E.Vassallo, "GMSK Demodulator implementation for esa Deep Space Missions", In Proceedings of the IEEE ,Contributed Paper, Volume 95, November 2007, pp. 2132-2141.
- [34] XC 4000 series devices and Configuration Guides for Virtex{II Pro, Virtex4 and Virtex-5 available at [www.xilinx.com](http://www.xilinx.com)
- [35] MATLAB simulink implementation details accessed from, [www.mathwork.com](http://www.mathwork.com)
- [36] ModelSim Reference. Retrieved on Oct 18, 2005 from [http://www.xilinx.com/ise/optional\\_prod/mxe.htm](http://www.xilinx.com/ise/optional_prod/mxe.htm).
- [37] Douglas L. Perry "VHDL Programming by Example" Fourth Edition, TataMcGraw-Hill Publication, New York, 2002.
- [38] D. Chin and S. Lam. "Implementing DSP designs with Xilinx System Generator and implementation tools." Retrieved on June 13, 2005 from <http://www.synplicity.com/literature/syndicated/pdf/DSP.pdf>.
- [39] Xilinxwebsite [online]  
[http://www.xilinx.com/publications/xcellonline/xcell\\_55/xc\\_pmethod55.htm](http://www.xilinx.com/publications/xcellonline/xcell_55/xc_pmethod55.htm).
- [40] Partial reconfiguration design flow, [www.xilinx.com/planahead](http://www.xilinx.com/planahead)
- [41] "Partial Reconfiguration on Xilinx Devices." [Online]. Available:  
<http://www.itee.uq.edu.au/listarch/partial-reconfig/>
- [42] Platform Studio User Guide, Xilinx, Inc., 2004, version 3.0.

## AUTHORS PUBLICATIONS

---

- [1] **Srinivas Gaddam**, R.C.Joshi, A.K.Saxena, "FPGA Implementation of QPSK and GMSK Modem for Reconfigurable Software Defined Radio (SDR)", National Conference on Signal processing Communication and VLSI, NCSCV'09, Coimbatore, May 8-9, 2009, pp.704-708.
- [2] Harikrishna.B, Rahul, **Srinivas.G**, R.C.Joshi, M.V.Karthikeyan, "Design and Implementation of Digital Baseband Modules of CDMA IS-95 and GSM for Reconfigurable SDR", 13<sup>th</sup> IEEE VLSI Design and Test Symposium, Bangalore, July 8-10, 2009. (accepted)
- [3] **Srinivas Gaddam**, R.C.Joshi, A.K.Saxena, "Accelerating GSM Baseband Processing for Reconfigurable SDR" INDICON 2009, Ahmedabad, Dec.18-20, 2009, (communicated)

# APPENDIX-A

## IMPLEMENTED DESIGNS

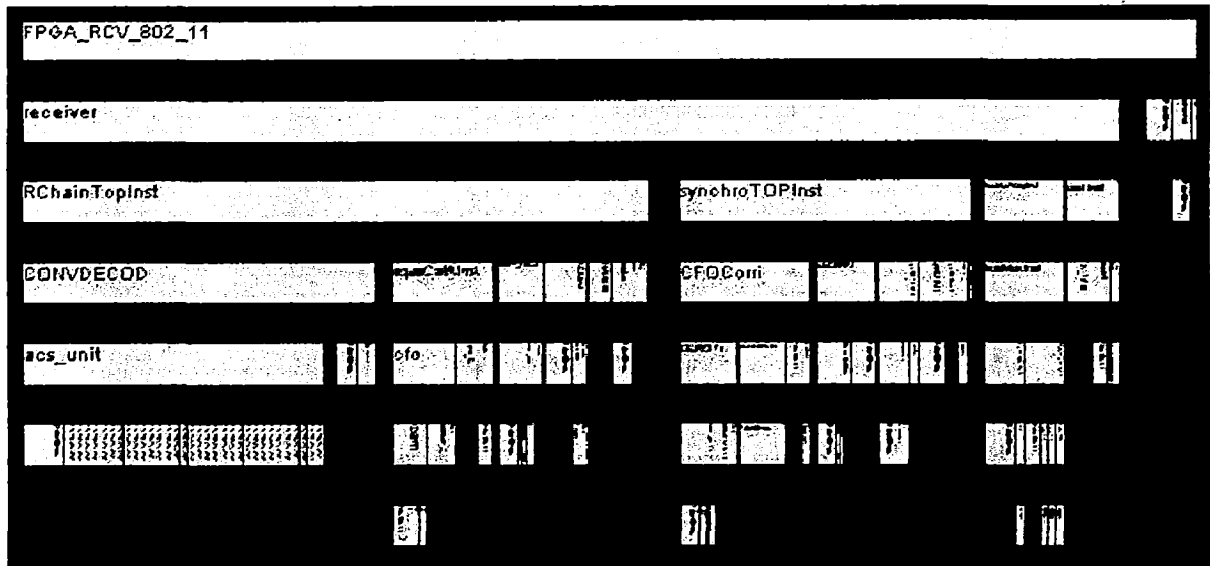


Fig.1: Hierarchical view of GSM design

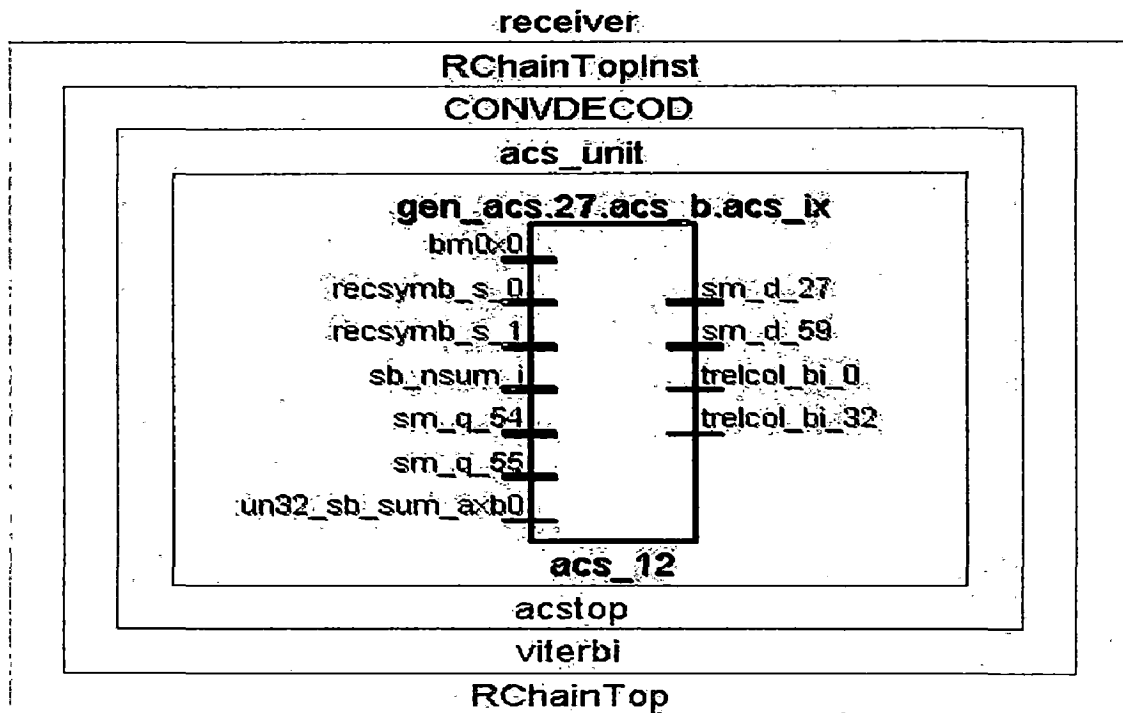


Fig.2: RTL view of Receiver

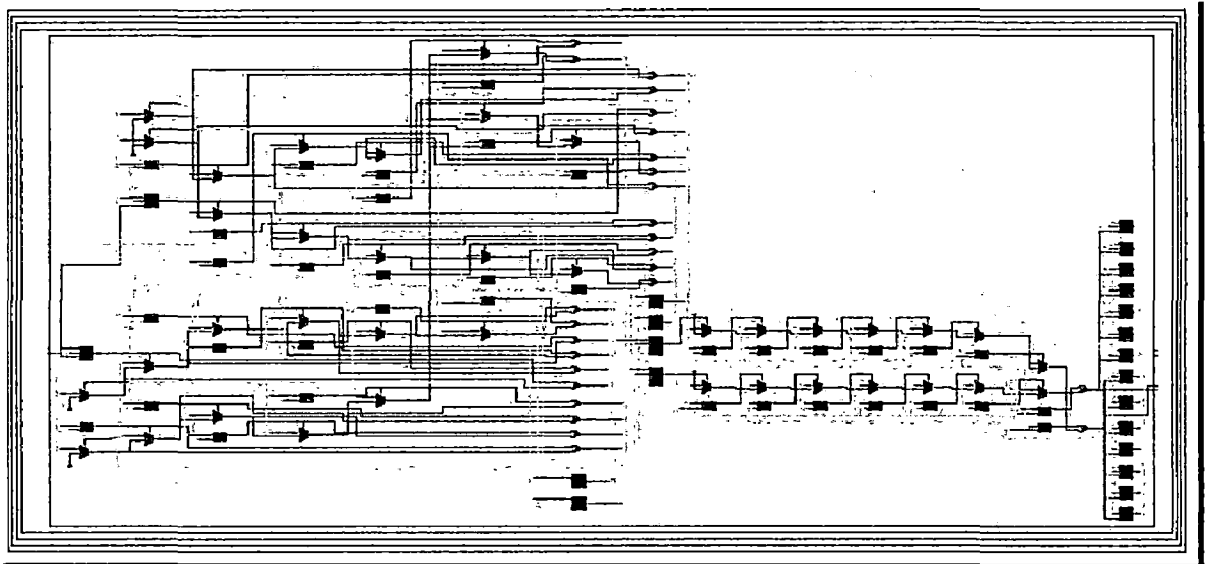


Fig.3: Schematic view of GSM baseband processing

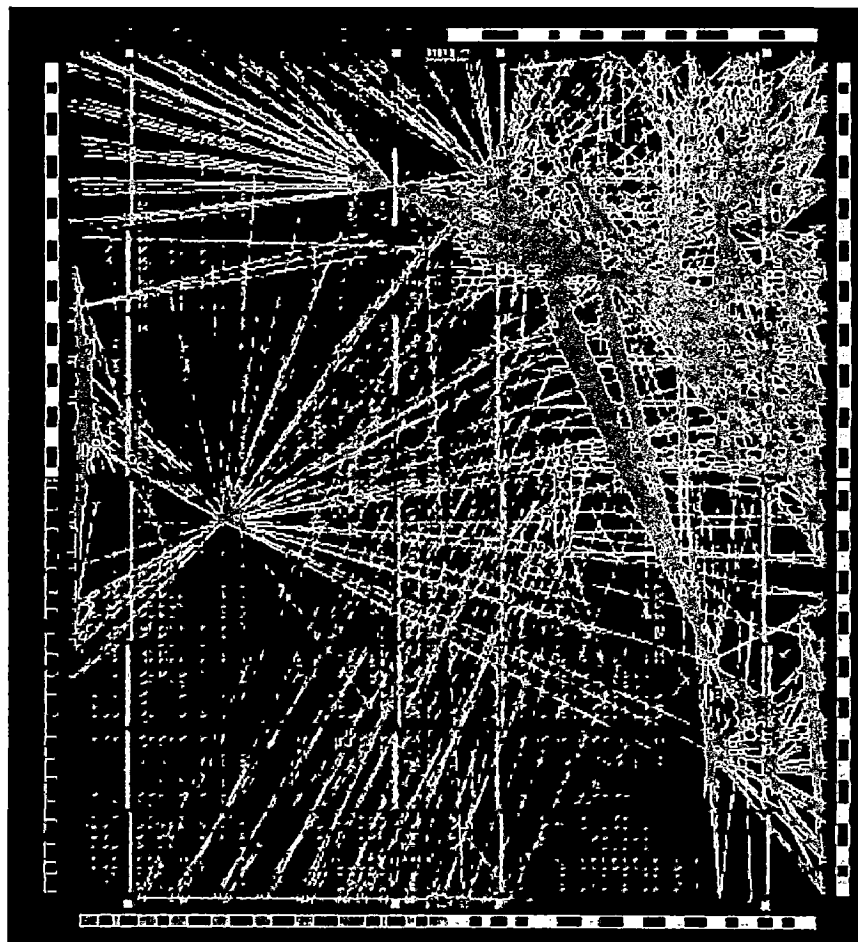


Fig.4: Floor plan view of GSM design



## APPENDIX-B

### DESIGN STATISTICS

**Design statistics of GSM\_v1:**

**Timing summary:**

-----  
 Timing errors: 0 Score: 0  
 Constraints cover 55489067 paths, 0 nets, and 144085 connections

**Design statistics:**

Minimum period: 12.032ns (Maximum frequency: 50.780MHz)  
 Minimum input required time before clock: 12.936ns  
 Minimum output required time after clock: 9.952ns

Analysis completed Monday June 08 03:32:59 2009  
 -----

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	1102	10944	10%
Number of 4 input LUTs	1203	10944	11%
<b>Logic Distribution</b>			
Number of occupied Slices	2298	5472	42%
Number of Slices containing only related logic	2298	5472	42%
Number of Slices containing unrelated logic	0	5472	0%
<b>Total Number of 4 input LUTs</b>	<b>2521</b>	<b>10944</b>	<b>23%</b>
Number used as logic	1203		
Number used as route thru	126		
Number used as shift registers	913		
Number used as bonded IOBs	262	320	81%
Number of FIFO 16/ RAMB 16 s	17	336	5%
<b>Total equivalent gate</b>	<b>1669098</b>		

**Power summary:**

	I (mA)	P (mW)
-----		
Total estimated power consumption:		128.3
---		
Vccint 1.20V:	449	538
Vccaux 2.50V:	234	585
Vcco25 2.50V:	5	13
---		
Clocks:	71	85
Inputs:	3	3
Logic:	30	36
Outputs:		
Vcco25	5	13
Signals:	0	0
---		
Quiescent Vccint 1.20V:	345	408
Quiescent Vccaux 2.50V:	234	584
-----		

**Design statistics of GSM\_v2:****Timing summary:**-----  
Timing errors: 0 Score: 0

Constraints cover 68489067 paths, 0 nets, and 244685 connections

**Design statistics:**

Minimum period: 9.247ns (Maximum frequency: 50.780MHz)

Minimum input required time before clock: 12.936ns

Minimum output required time after clock: 9.952ns

Delay: 12.092ns

**Analysis completed Thursday June 11 11:46:48 2009**  
-----

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	1532	10944	14%
Number of 4 input LUTs	1970	10944	19%
Logic Distribution			
Number of occupied Slices	2845	5472	52%
Number of Slices containing only related logic	2845	5472	52%
Number of Slices containing unrelated logic	0	5472	0%
Total Number of 4 input LUTs	3440	10944	31%
Number used as logic	1970		
Number used as route thru	158		
Number used as shift registers	1028		
Number used as bonded IOBs	262	320	81%
Number of FIFO 16/ RAMB 16 s	22	336	6%
Total equivalent gate	2929278		

**Power summary:**

I (mA)

P (mW)

-----  
Total estimated power consumption:

228

Vccint 1.20V:

449

538

Vccaux 2.50V:

234

585

Vcco25 2.50V:

5

13

-----  
Clocks:

71

85

Inputs:

3

3

Logic:

30

36

Outputs:

5

13

Vcco25

0

0

Signals:

-----  
Quiescent Vccint 1.20V:

345

408

Quiescent Vccaux 2.50V:

234

584  
-----