18551 Final Report

Spring 2002

Software Defined GSM Receiver

Group #1
Changshi Xiao (cxiao@andrew.cmu.edu)
Ahmet G. Cepni (acepni@andrew.cmu.edu)
Darren W. Karns (dk5x@andrew.cmu.edu)

I. Introduction:

In this project, we implemented a software-defined GSM receiver on the TI TMS320C67 DSP EVM. First, we implemented the whole GSM transmitter and receiver processes on Matlab, and then we translated part of the receiver process onto the EVM board. As for the demo, we tried to process speech signal and demodulate it on EVM board, and get the speech back from a speaker. The speech encoding was done on Matlab, the encoded bit stream was transferred to Agilent Digital Signal Generator to generate standard GSM signals, then the analog GSM signal was captured by Vector Signal Digitizer, the sampled baseband I&Q channel signals were stored into a file, and then block by block sent to EVM board for demodulation. We also modeled a wireless channel, which had Additive White Gaussian Noise (AWGN) and small-scale Rayleigh fading, and compared the performance of demodulation under different Signal-to-Noise (SNR) values and different fading conditions. This report is organized as flowing: in section II, we'll give some overview of GSM standard and the concept of Software Defined Radio (SDR), which is a hot research area in the last ten years. In section III, we'll introduce more about our system architecture. In section IV, we'll give some details of the algorithm we implemented. In Section V, we'll give some simulation results with various channel models.

II. GSM specifications and Software Defined Radio (SDR)

GSM (Global System of Mobile communications) is the first digital wireless communication standard originated in Europe in 1982, but soon became a global standard because of its popularity. The first GSM is operating at 900 MHz, and then upgraded twice to 1800 MHz and 1900 MHz separately. Here in this project, we only introduce GSM900; the technical specification is shown below.

GSM-900 Specification

Downlink	935-960 MHz
Uplink	890-915 MHz
Channel Sapcing	200 kHz
Modulation	GMSK
Typical Mobile Transmit Power	3.7 mW to 1 W
Maximum BaseStation Transmit Power	320 W
Maximum Distance	35 km
Speech Encoding	LPC (13kbit)
Bit rate	270 kpbs

The basic system flow diagram is showed in Fig. 1. After converting the analog speech signals into digits, the raw source bits are coded according to the model of human speech to reduce the bits needed to be transferred. Depending on the relative importance of the bits, the channel coding function arranges the bits into blocks and adds some error correction codes. The encoded bits are then applied to interleaver, which shuffles the bit positions in the blocks to provide some time-diversity in fading channels. The interleaved bit blocks are assembled with some training bits, and control bits to form normal speech GSM bursts, which are modulated by GMSK modulation technique before sending to RF amplifier and antenna. The process in the receiver side is the reverse of the transmitter. Plus, since the channel is not perfect, there always exists degradation of signal by noise, fading, doppler shifting, etc., so some sort of channel equalization and noise filtering is needed to recover the original signals.

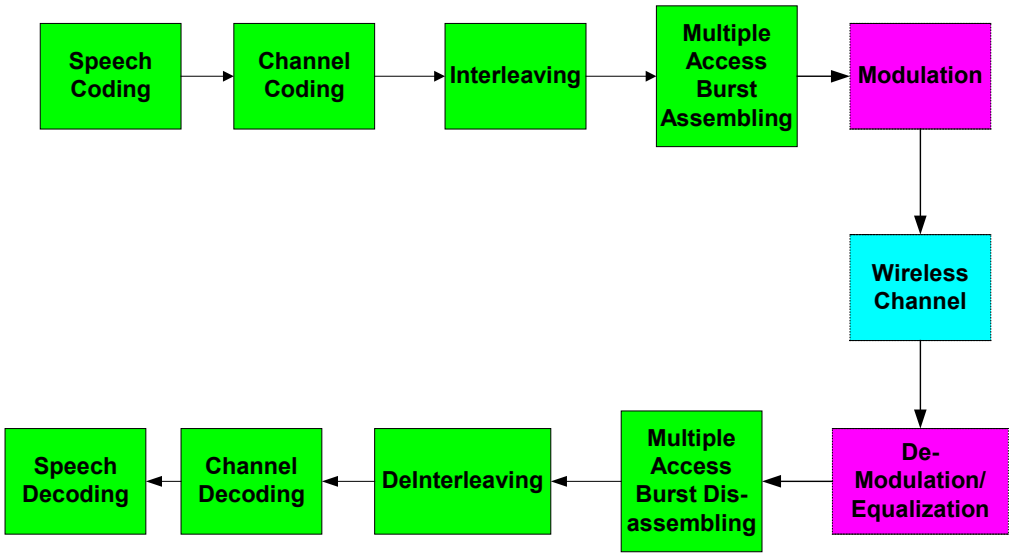


Figure 1: Complete GSM system[1]

GSM frame structure:

There are two frequency bands of 25 MHz each that have been allocated for the use of GSM. The band 890 - 915 MHz is used for the uplink direction (from the mobile station to the base station). The band 935 - 960 MHz is used for the downlink direction (from the base station to the mobile station). By FDMA, the frequency band is divided into 200 kHz subbands, and by TDMA, each subband is divided into 120ms multiframes, and each multiframe is divided into 26 frames, the first two frames are used for controlling channel, and the rest are for traffic channel, and again, each frame is divided into 8 time bursts, each of which is approximately 0.577 ms. In GSM, there are 4 different types of bursts. A normal

[1] GSM technical specs:
Theodore Rappaport, *Wireless Communications, Principles& Practice*, Prentice Hall, 1996.

burst is used to carry speech and data information. The structure of the normal burst is shown below.
Each burst consists of 3 tail bits at each end, 2 data sequences of 57-bits, a 26-bit training sequence for equalization, and 8.25 guard bits.

There are 2 stealing bits (1 for each data sequence) that are used by Fast Access Control Channels. The frequency correction burst and synchronous burst have the same length as normal burst. They have different internal structures to differentiate them from normal bursts. The frequency correction burst is used in Frequency Correction Channels (FCCH) and the synchronous burst is used in Synchronization Channels (SCH). The random access burst is shorter than a normal burst, and is only used on Random Access Channels (RACH).

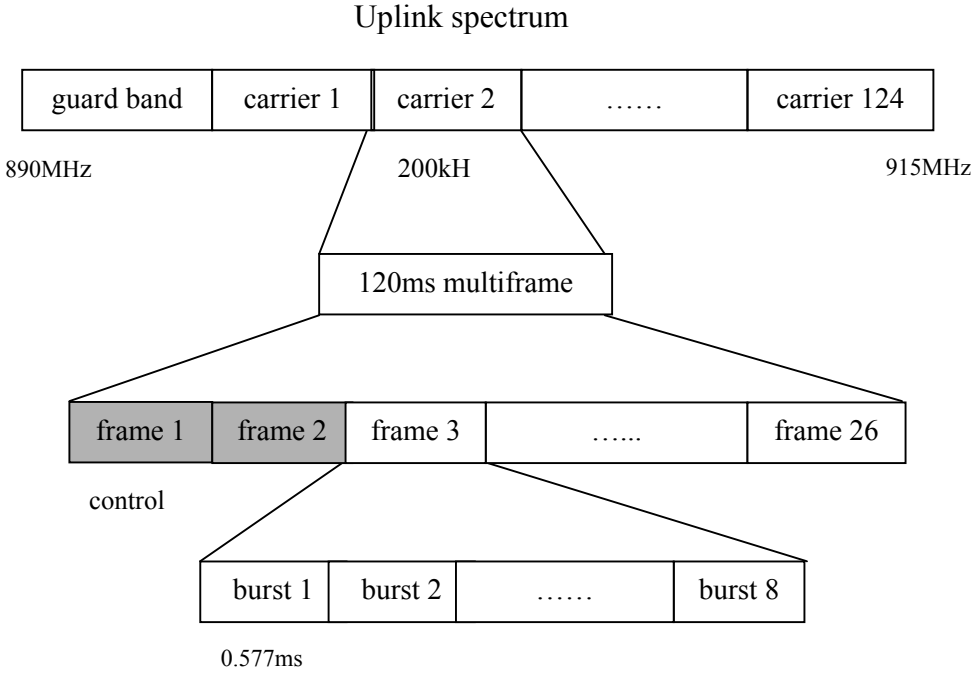


Figure 2. GSM frame structure, uplink

Source Encoding/Decoding

RELP Motivation

In mobile communications systems, bandwidth is a precious commodity. The goal of all speech coding systems is to transmit speech with the highest possible quality using the least possible channel bandwidth. The lower the bit rate at which the coder can deliver quality speech, the more speech channels can be implemented within a given bandwidth. Residual

Excited Linear Predictive Coding, RELP, offers low bit rate speech coding that can be used to meet this challenge[2].

Speech coders are broadly classified into two categories: Vocoders and Waveform Coders. Vocoders are a class of speech coding systems that analyze the voice signal at the transmitter, transmit parameters (spectral coefficients) derived from the analysis, and then synthesis the voice at the receiver using those parameters. Vocoders are much more complex than Waveform Coders, but they are less robust and their performance tends to be talker dependent. The most popular vocoding system is Linear Predictive Coding, LPC.

RELP Transmission

In the RELP approach, standard LPC analysis yields the spectral coefficients. The number of spectral coefficients, L, is predetermined. These spectral coefficients are then used to synthesize the original speech[3].

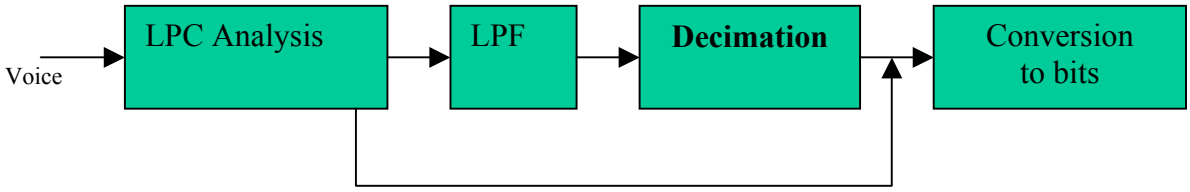


Figure 3: Speech coding

The synthesized speech is then subtracted from the original speech to form a residual signal. The residual signal is then low pass filtered to extract a baseband signal from the residual signal. The baseband residual sequence is then decimated. The decimation factor controls the amount of compression in the encoding process. The baseband residual sequence, the LPC coefficients and the first L original speech values are then coded to bits and transmitted.

At the receiver, the received signal is first interpolated back to its original sampling rate. This interpolated signal is full-band reconstructed through nonlinear distortion. Only the high frequencies need to be reconstructed since the baseband signal is sent intact. To accomplish this high frequency regeneration, the interpolated signal is full wave rectified followed by a double differencing operation. The resulting signal is high pass filtered using the same cutoff frequency as the low pass filter used in transmission. Now the baseband interpolated signal is added with the high frequency signal to produce the residual sequence.

---

[2] Lots of tutorial information and matlab/C code for voice coding http://www.eas.asu.edu/~spanias/index.html

[3] Lots of tutorial information for LPC voice coding http://www.kt.tu-cottbus.de/speech-analysis/

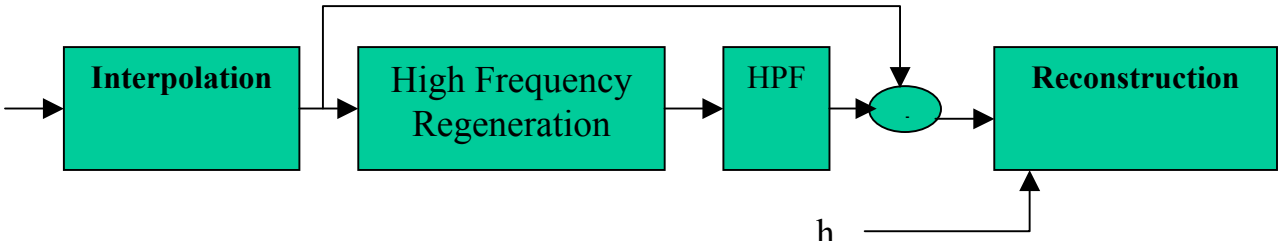


Figure 4: Speech decoding

Using the spectral coefficients that were transmitted, the L original values and the residual sequence the original speech is reconstructed[4].

Channel Encoding/ Decoding

A major feature of digital data transmission is the myriad techniques used to protect data or speech through coding. Coding adds additional bits to the original payload to provide a means of protecting the original information. This gives the data more security, since it is possible to identify and even correct (within certain limits) data corrupted in the RF path[5].

The purpose of the channel encoder is to provide the GSM receiver with the ability to detect transmission errors and eventually correct some of these. This is to improve the transmission quality from a bit error point of view. Various encoding standards are used in GSM depending on the mode of transmission. The encoding implemented on Matlab was burst type TCH/FS (*Traffic Channel Full rate Speech*), which is a normal speech, burst. After the speech coding, the bits are grouped in blocks of size 260. Out of 260 bits, the first 50 bits are called Class Ia bits, the next 132 bits are Class Ib bits and the last 78 bits are Class II bits (see Fig. 5). This ordering was done based on the importance of the bits. For instance, any transmission errors in the class Ia bits effect the overall speech quality more severely than errors in class II bits. Due to this variation in bit importance the different classes of bits are encoded accordingly[6].

[4] Source code for GSM 06.10 RPE-LTP algorithm
http://www.ddj.com/documents/s=1012/ddj9412b/

[5] A detailed discussion on channel encoding/interleaving in GSM systems:
Siegmund Redl , *An introduction to GSM* , Artech House ,Inc. 1995.

[6] Matlab codes for various channel encoding/decoding schemes:
J. Proakis, M. Salehi, *Contemporary Communication Systems using Matlab,* BookWare Companion Series.

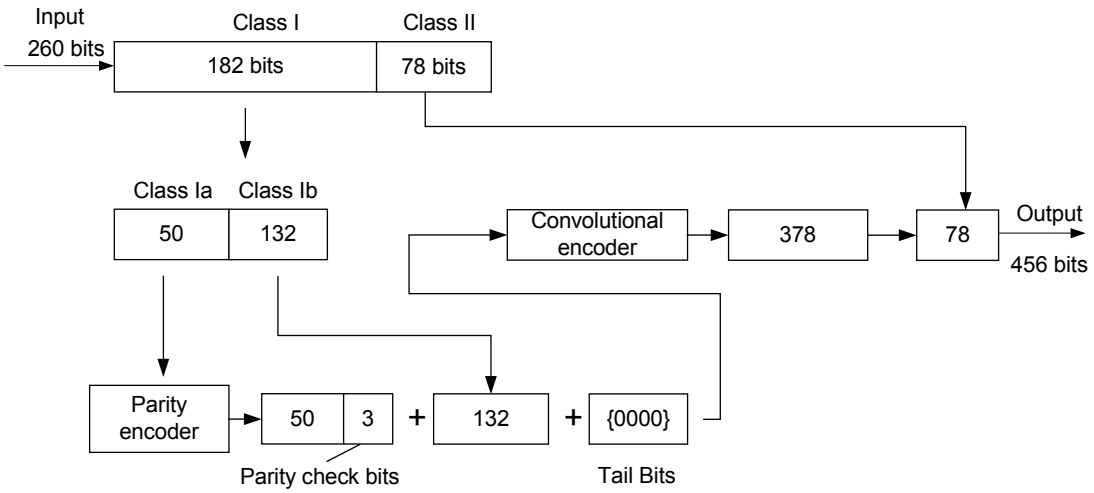


Figure 5: Channel Encoding in GSM

The 50 most significant bits (Class Ia bits) are parity encoded. The parity encoder used in GSM is a systematic cyclic encoder based on three check bits. Systematic means that the parity bits are added to the original class Ia bit sequence. This way the class Ia bits are left unchanged 3 extra parity check bits are added to the end of the Class Ia bits.

The generator polynomial used in the parity encoder has a length of 4 bits and is given as $G(x)=x^3+x+1$. After parity encoding of Class Ia bits, the resulting 53 bits are combined with Class Ib bits, and a tail sequence of four zeros are added to the block. The total number of bits is 189, and this block is sent to convolutional encoder. The convolution encoder takes a block of k bits as input and returns a block of n bits as output. The rate of the encoder, defined as the ratio k/n, is in the GSM system specified to be 1/2. In the convolutional encoding scheme each output bit, c_n is depending not only on the input bit presently being encoded, b_k, but also on some of the previous input bits. The number of input bits required in the processing of encoded output bit is called the constraint length of the encoder. GSM specifies a constraint length of 5 in its encoding scheme defined as:

$$C_{2k}=b_k \oplus b_{k-3} \oplus b_{k-4}$$
$$C_{2k+1}=b_k \oplus b_{k-1} \oplus b_{k-3} \oplus b_{k-4}$$
$\oplus$ =Mod 2 addition
$k \in \{0,1,2,3,\ldots..189\}$ and $b_k=0$ for $-\infty<k<0$

As the convolution encoder is defined as a rate ½ encoder two output bits are generated for every input bit, hence the two expressions. After convolutional encoding, 378 bits are generated out of 189 input bits, and they are combined with Class II bits to produce the output of the channel encoding. The input to the channel encoder has a block length of 260 bits, and the output of the encoder is 456 bits long[7].

[7] "Digital cellular telecommunications system, Channel coding" GSM Technical Specification. GSM 05.03 version 5.2.0, August 1996. Publication of the ETSI.

Convolutional decoding can be performed using a Viterbi algorithm. A Viterbi decoder logically explores in parallel every possible user data in sequence. It encodes and compare each one against the received sequence and picks up the closest match: it is a maximum likelihood decoder. To reduce the complexity (the number of possible data sequence double with each additional data bit), the decoder recognizes at each point that certain sequences cannot belong to the maximum likelihood path and it discards them. The encoder memory is limited to K bits (5 in GSM channel decoder); a Viterbi decoder in steady-state operation keeps only 2^{K-1} paths. Its complexity increases exponentially with the constraint length K. The channel decoder implemented in Matlab reverses the operation done in the channel encoder[6].

Interleaving

The interleaver shuffles the bits contained in the data blocks that are coming from the channel encoder, and distributes them over a number of bursts. The purpose of this procedure is to ensure that the errors that appear in a received data block are uncorrelated. The motivation for reducing the correlation between bit errors is that the convolution code used to protect the class I bits has better performance when errors are not correlated. Correlation between bit errors can occur in for example fading conditions. The block of size 456 bits coming out of channel encoder is spread onto eight bursts in subblocks of 57 bits each. A subblock is defined as either the odd or even-numbered bits of the coded data within one burst. The data are not put an ordered row into these subblocks, but are re-ordered before they are mapped onto the GSM bursts. This further decreases the possibility of a whole group of consecutive bits being destroyed in the radio channel. Convolutional codes are much better at repairing individual bit errors than they are at repairing burst errors. The 456 bits are subdivided onto the eight subblocks in the following way. Bit number 0 goes into subblock 1, bit number 1 goes into subblock 2, and so on until all eight subblocks are used up. Bit number 8 ends up in subblock number 1 again. The first four subblocks are put into the even-numbered bits of four consecutive bursts, and the second four blocks are put into the odd-numbered bits of the next consecutive four bursts. Upon receipt of the next speech block, a burst then holds contributions from two successive speech blocks[5].

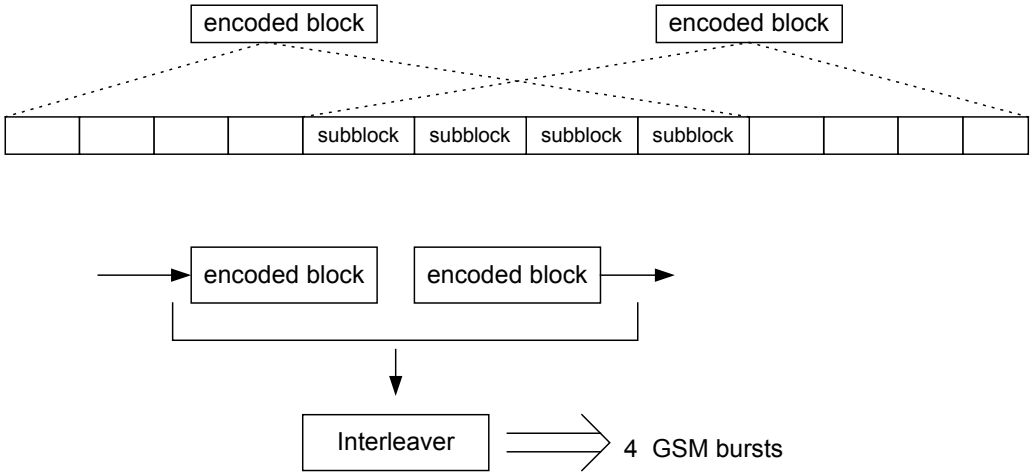


Figure 6: Interleaving block diagrams

It can be realized that the interleaver can be implemented so that it operates at two code blocks at a time. For each interleaving pass four sets of encoded blocks are returned. These data are further processed for the burst structure. Since two instances of encoded blocks contain two times 456 bits, and four set of output burst contain 456 bits, it is evident that all the bits contained in the input to the interleaver are not represented in the output. This is solved by passing each code block to the interleaver two times. In practice this is done by implementing a queue of code blocks.

The de-interleaver reconstructs the received encoded data from the received data. The operation is the inverse of the interleaver, and may thus be considered as an reordering of the shuffled bits [5].

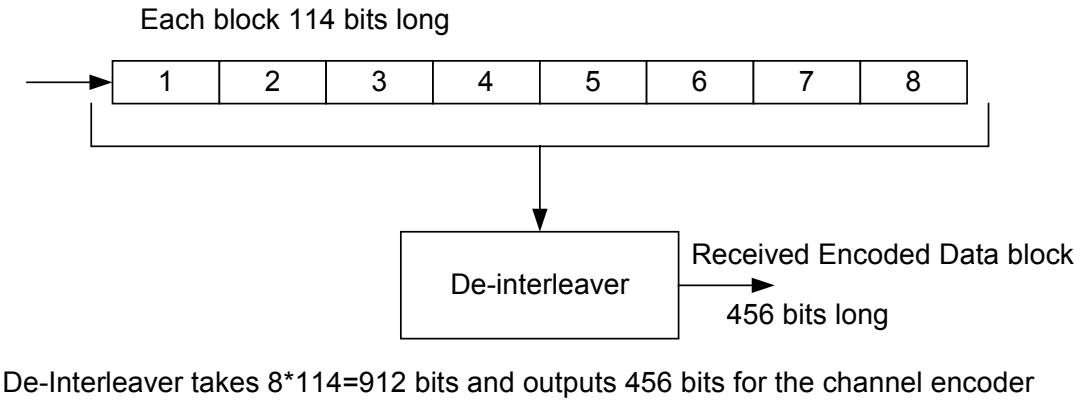


Figure 7: Operation principles of de-interleaving.

Burst Structure

The interleaver takes 2 of 456 encoded data bits and returns 4 blocks of 114 bit blocks, each one of them forms a GSM burst when combined with equalization bits and control bits. The implemented burst, referred to as a GSM normal burst, has the structure given in Fig. 8.

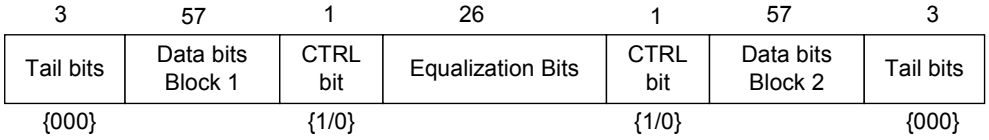


Figure 8: GSM normal burst structure

In one GSM burst, there are 3+57+1+26+1+57+3=148 bits. Of these 148 bits, 57*2=114 are data bits. The equalization bit sequence can be any of eight prescribed ones. The bit sequence selected in this project was:

EQUALIZATION_BITS = [0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1 1];

The tail bits are all zeros, control bits are selected as 1's. This is the structure that is modulated; on the receiver side, the 114 data bits are extracted from each burst and applied to de-interleaver[8].

Differential Encoding

The output from the GSM Burst is a binary {0, 1} bit sequence. This sequence is first mapped from the RTZ (*Return To Zero*) signal representation to a NRZ representation before being input to the GMSK-modulator. This task is accomplished by the differential encoding function[9].

GSM makes use of the following combined differential encoding and level shifting scheme, where

$$d \{0,1\},\ldots\ a\{-1,1\}$$
$$d'[n]=d[n] \oplus d[n-1]$$
$$a[n]=1-2*d'[n]$$

To avoid the start condition problem the GSM-recommendation prescribes that an infinite length sequence of all ones are assumed to precede the burst to be processed. Hence, when calculating a[0] and thereby also d'[0] ,it may be assumed d[-1]=1.

[8] Some algorithms and Matlab codes for GMSK modulation/ GSM specifications:
Vijay K. Garg , *Principles & Applications of GSM*, Prentice Hall Communications Engineering and Emerging Tech. Series

[9] A brief Overview of the GSM Radio Interface Technical Memorandum, MIT, TM-547, March 1996. Thierry Turletti *http://www.sds.lcs.mit.edu/~turletti/gsm-overview/gsm-overview.html*

GMSK-Modulation/Demodulation

Modulation

After the differential encoding of the GSM burst, the signal is GMSK-modulated. The modulation specified for GSM is GMSK (Gaussian Minimum Shift Keying) with BT=0.3 and rate 270.833 kb/s. GMSK is a type of constant-envelope FSK, where the frequency modulation is a result of a carefully adjusted phase modulation. The most important feature of GMSK is that it's a constant-envelope variety of modulation. This means that there is a distinct lack of AM in the carrier with a consequent limiting of the occupied bandwidth [5,8].

GMSK is a modulation form derived from the very similar MSK (Minimum Shift Keying). Mathematically the generation of a MSK-signal may be described as

$$s(t,a) = \sqrt{\frac{2E_c}{T_b}} \cos(2\pi f_c t + \theta(t,a))$$

Where E_c is the bit-energy, f_c the carrier frequency, and $\theta(t,a)$ the information-carrying phase of the MSK signal. If we remove the carrier frequency, the baseband MSK signal expression becomes:

$$\tilde{s}(t,a) = \cos(\theta(t,a)) + j.\sin(\theta(t,a))$$

It's clear that by taking the Cosine and Sine of $\theta(t,a)$ two low-pass baseband signals result, the in-phase I and quadrature Q signals. The discrete time representation of $\theta(t,a)$ is given as:

$$\theta[n] = \frac{\pi}{2} \sum_{k=0}^{n-1} a[k]$$

where a[k] is differential encoded NRZ input data signals that have values {1, -1}. Thus $\theta[n]$ has four possible values $\{0, \pi/2, \pi, 3\pi/2\}$.

To further simplify the MSK-baseband description, a complex sequence I is introduced where the complex data symbols are given as:

$$I[n] = \cos(\theta[n]) + j.\sin(\theta[n]).$$

I[n] thus assumes real and imaginary values in an alternating fashion. This leads to the following recursive MSK-mapping definition.

I[n]=j.I[n-1].a[n-1]

where

I[n] $\in \{1,-1,j,-j\}$

a[n] $\in \{1,-1\}$

The complete MSK-mapping block diagram is shown in Fig.9

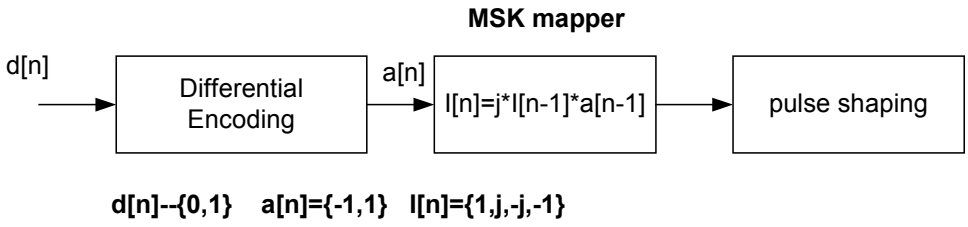


Figure 9: MSK mapping block diagram

In GMSK the phase shift are made smoother than MSK. This results in a more narrow frequency spectrum. The intersymbol-interference (ISI) increases, which can increase the bit error rate. A GMSK-signal can be generated using different approaches, the one showed in Fig.10 is implemented on Matlab for simulation purposes.

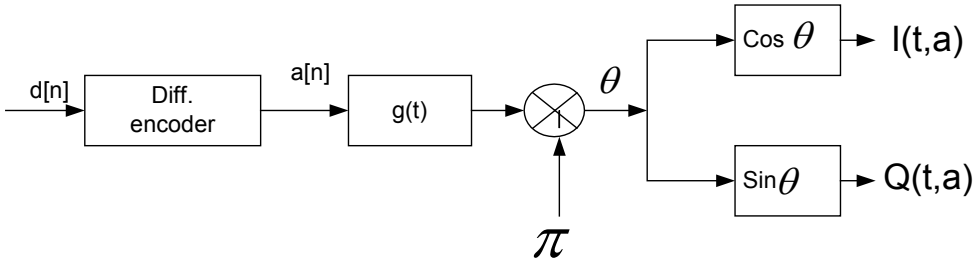


Figure 10: GMSK mapping block diagram

where d[n] is the incoming bit stream {0/1}, a[n] is the differential encoded NRZ data {-1/1}. g(t) is the pulse-shaping function defined as a convolution of a rectangular pulse and a Gaussian function

$$g(t) = v(t) * h_g(t)$$

$$v(t) = 1/(2T_b), \quad \text{for } 0<|t|<T_b/2$$
$$= 0, \quad \text{otherwise}$$

$$h_g(t) = \frac{1}{\sqrt{2\pi}\sigma T_b} \exp\left[\frac{-t^2}{2\sigma^2 T_b^2}\right]$$

$$\text{where} \quad \sigma = \frac{\sqrt{\ln 2}}{2\pi B T_b}$$

The 3 dB bandwidth B , of the Gaussian function is specified by the normalized bandwidth. BT_b is specified to 0.3 for GSM [8,10].

[10] B. Ramamurthi, K. Giridhar,"DSP-based digital FM demodulation for GMSK signals."

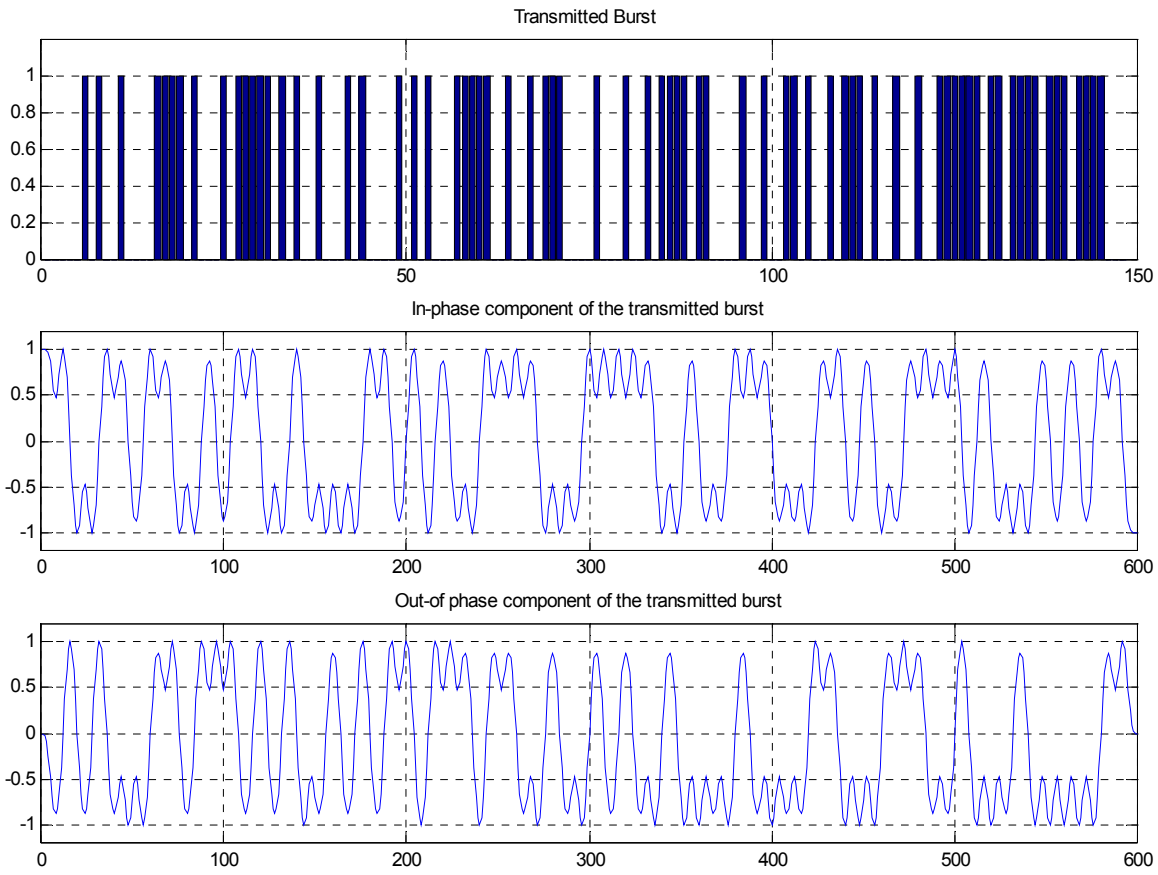


Figure 11: Normal GSM burst (148 bits), the GMSK modulated in-phase and quadrature components.

Fig. 11 shows one GSM burst and the GMSK modulated waveforms of GSM burst. First 260 input bits are applied to GSM channel encoder, channel encoder produces 456 bits, which are grouped into four blocks each of them 114 bits long. In each block, equalization bits and control bits are included, and each one of them forms a GSM burst having 148 bits.

The concept of Software Defined Radio:

Traditionally in digital wireless communication, the digital signal processing is relevant only after the demodulator, that is, after a decision of 1 or 0 has been done, the demodulation process is still in analog world. And the digital signal processing is often implemented in hardware device such as ASIC, the benefits about this solution is the speed and power consumption. However, wireless communication is a fast changing field, there are various standards in different contry and new generations of standards are proposed. A pure hardware solution to wireless communication (or other field) means different set of IC chips has to be designed and manufactured for different standards and different generation of mobile system, which is quite expensive, and since the hardware is fixed, there is no way to communicate if

the standard of the mobile phone is different from that of the base station, so the true global mobility is impossible[11].
The recent development of fast, power-saving general purpose DSP and ADC/DAC chips makes a software solution to this problem possible. In an ideal software defined radio system, the analog to digital conversion is right after the antenna, all signal processing is in digital world. This solution gives maximum flexibility and mobility, because, by simply downloading different software through air link or other method, a mobile system can be easily reconfigured to satisfy different requirements of communication or other user applications[12].

III. Our System Architecture

Our system to implement the whole GSM transmission process is the integration of different subsystems, and each subsystem is implemented in different ways (see Fig. 12). At the transmitter side, the speech signal was recorded with the help of a microphone, and processed on a Matlab program, which performs the speech source encoding. As an example, one second of speech signal takes up 13,200 bits. With the help of a C code we developed to control Agilent Digital Signal Generator (DSG), the encoded bits are transferred to DSG. DSG is capable of generating standard GSM waveforms with the input bit pattern that we provided.

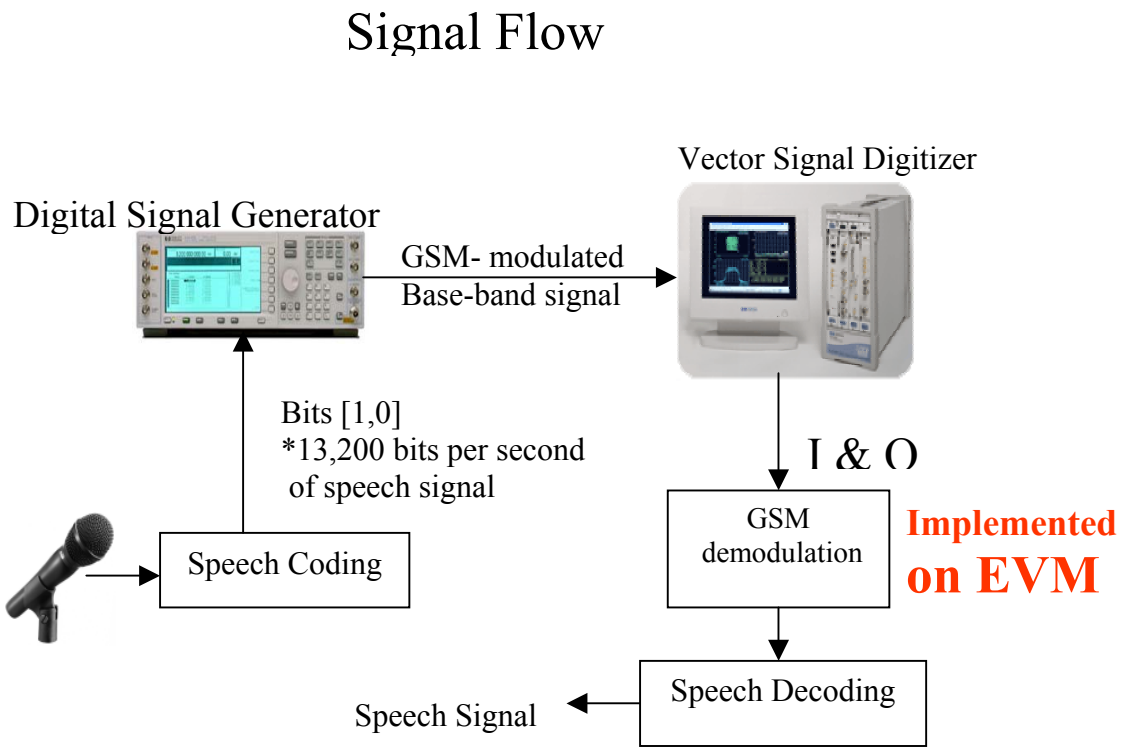
The analog GSM signals were generated with a carrier frequency of 1 MHz (baseband GSM), and transmitter power of –20 dBm. The GSM signal was captured and sampled by a Agilent Vector Signal Digitizer. The quantized GSM data which consists of in-phase (I) and quadrature (Q) components was stored in a file on PC. In our project, because of file size constraint and speed concerns, we stored 0.7 seconds of GSM signal in each file.
How much memory do we need to store 0.7 sec of GSM signal in a text file?

One GSM burst, consisting of 148 bits, takes up 0.577 msec. In one frame, there are 8 GSM bursts (see Fig. 2) consuming 0.577x8= 4.6 msec. So in 0.7 sec. there are 700/4.6=152 frames. How much samples do we have in one frame? There are 5 samples per bit. In one frame there are 148*8 bits, and 148*8*5=5920 samples. In 152 frames, we have 899840 samples for both I and Q components. In a text file, each character is represented by an ASCII number, for one pair of I and Q samples, we have 25 characters, meaning 25 bytes.
So for 0.7 sec. of GSM signal, we need 899840*25=22.5 Mbytes of file to store it as a text file. After storing all these data, we can apply TDMA; we are interested in only one burst at which we modulate our speech bits, so when we remove the 7 of GSM bursts, the file size becomes 22.5/8=2.8 Mbytes. This is the file that we need to process on EVM and demodulate the speech signal. The I and Q data was transferred to EVM board through PCI bus. The program implemented on EVM performs GMSK demodulation, differential decoding, separating user data from equalization bits and control bits. The demodulated bits were then sent back to PC. Another program running on PC collected the decoded bits, performed channel decoding, deinterleaving and speech decoding, the final decoded speech

[11] Philip Mackenzie, “Software Radio on General-Purpose Processors”,

[12] Michel Corneloup**,** “Open Architecture for Software Defined Radio Systems”,

data were written into a .WAV file, and played back through the speaker. Also between the transmitter and receiver, we simulated a wireless channel by modifying the raw sampled data, adding AWGN noise and Rayleigh fading.



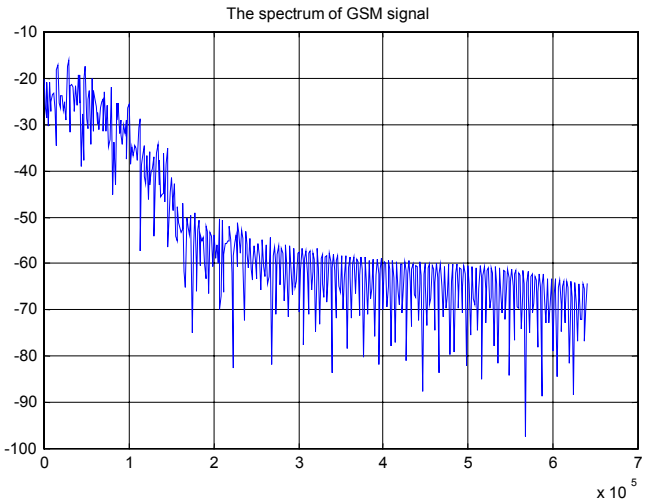


Figure 12: Signal Flow diagram, and the spectrum of GSM signal obtained from Digital Signal Generator

IV. Algorithm Details Implemented in this project

In this project, we implemented some key functions of the GSM receiver and a wireless channel model.

Channel Model:

In mobile radio channels, the Rayleigh distribution is commonly used to describe the statistical time varying nature of the received envelope of a flat fading signal, or the envelope of an individual multipath component. It's well known that the envelope of the sum of two quadrature Gaussian noise signals obeys a Rayleigh distribution. The Rayleigh distribution has a probability density function given by:

$$P(r)=\frac{r}{\sigma^2}\exp(-\frac{r^2}{2\sigma^2}) \qquad (0<r<\infty)$$

where σ is the rms value of the received voltage signal before envelope detection, and σ^2 is the time-average power of the received signal before envelope detection.

We implemented two-ray Rayleigh fading model for the project. In modern mobile communication systems with high data rates, it has become necessary to model the effects of multipath delay spread as well as fading. We will use the following 2-ray Rayleigh fading model in our project.

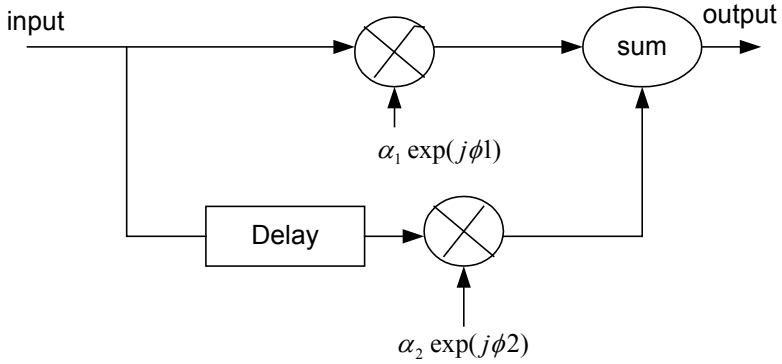


Figure 13: Two-ray Rayleigh fading model.

The impulse response of the model is represented as:

$$h_b(t)=\alpha_1\exp(j\phi_1)(t)+\alpha_2\exp(j\phi_2)(t-\tau)$$

where α_1 and α_2 are independent Rayleigh distributed , ϕ_1 and ϕ_2 are independent and uniformly distributed over $[0,2\pi]$, and τ is the time delay between two rays [1]. τ (time delay) will be 16μ sec as part of GSM specification.

The Rayleigh distributed numbers were generated by using the following formula:

$$R_n=\sqrt{-2\sigma^2\ln(1-A_n)}$$

where A_n is uniformly distributed in the interval [0,1], and σ is the rms value of the received voltage signal before envelope detection[13].

When we are done with Rayleigh fading model, we will include AWGN noise and convolve the incoming signal with our channel model. So the final channel model will be as follows:

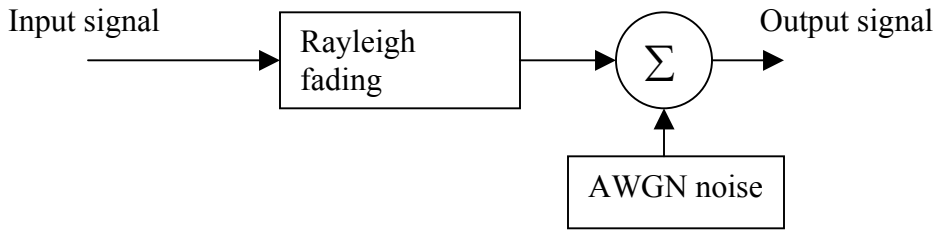


Figure 14: Channel Model

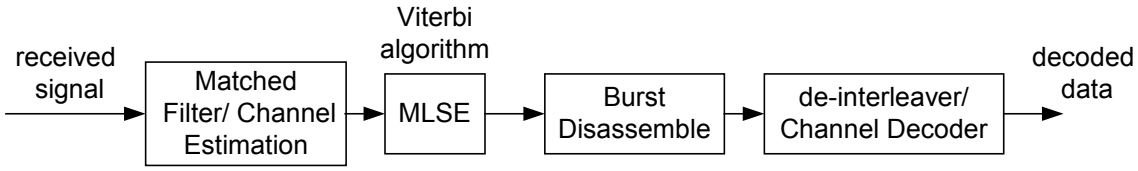


Figure 15: GSM receiver structure

The received signal is first fed to a matched filter[14]. The purpose of matched filter is to estimate the channel and downsample the received signal so that there would be one sample per bit as input to MLSE block. In each GSM burst, we inserted equalization bits. The methode used for obtaining synchronization is based on the mathematical properties of the equalization bit sequence[15].

EQUALIZATION_BITS = [0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1 1];
for which the MSK-mapped equivalent is found as
EQ_1= [1, j, 1, -j, 1, -j, -1, j, -1, -j, -1, -j, 1, j, 1, -j, 1, j, 1, -j, 1, -j, -1, j, -1, -j];

Now, from EQ_1, the central 16 MSK-symbols are picked and zero-padded on both sides (EQ_2). When we correlate these two sets, we get the following plot.

[13] Noise& Fast Correlations 18-551 course notes

[14] GSM-Systems - Channel Estimation, Equalization and Decoding, Volker Franz ,PhD thesis, Technischen Universität München, 19.5.2000

[15] G. Alebachew,"Implementation of feedback methods for improved equalization in GSM," Technical report EX018/2001. http://www.s2.chalmers.se/research/comsys/master_theses/

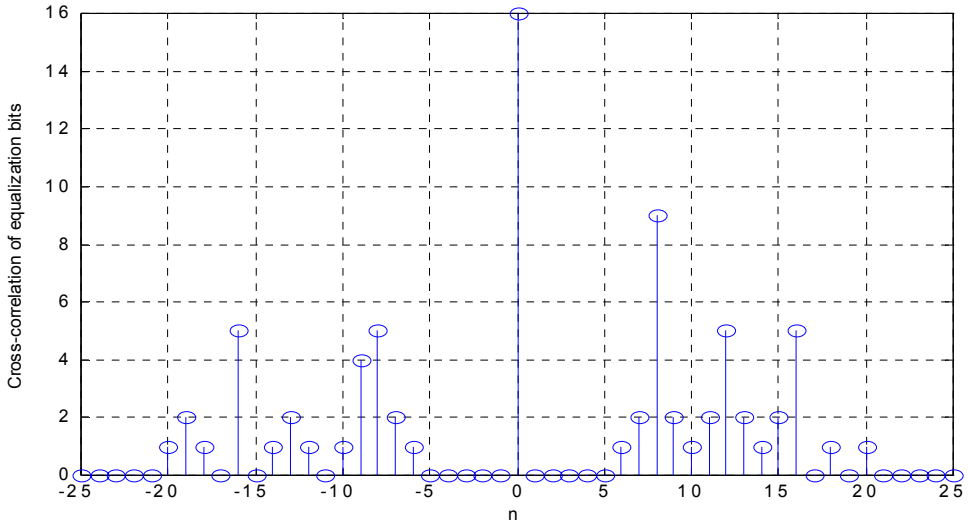


Figure 16: Cross correlation of equalization MSK symbols

where $R_{EQ1,EQ2}[n]=EQ_1*EQ_2[-]*$

$$R_{EQ1,EQ2} = \begin{cases} 16.....n=0 \\ 0....n=\pm\{1,2,3,4,5\}. \\ ?.....otherwise \end{cases}$$

This property is useful since the received signal corresponding to the transmission of the training sequence, may be written as:

$$r_{EQ1}= (EQ_1)*h+w$$

where h is the channel impulse response, w is unknown AWGN noise;
If we convolve this with the zero-padded MSK-mapped Equalization set (EQ_2)

$$r_{EQ1*}\ EQ_2[-]*=((EQ_1)*h+w)*EQ_2[-]*=h*EQ_1*EQ_2[-]*+w*EQ_2[-]*$$

$$r_{EQ1} * EQ_2[-]^* = \begin{cases} 16h + w * EQ_2[-]^*.....n=0 \\ w * EQ_2[-]^*.....n=\pm\{1,2,3,4,5\} \end{cases}$$

Thus, if an entire burst containing EQ_1 is correlated with EQ_2, an estimate of the channel impulse response is present in the result, called S. The estimate of the impulse response that

is contained in S is likely to be more powerful than the neighboring contents of S. This is due to the factor sixteen and the zero samples.

$$S=r*EQ_2[-]*$$

The channel estimation and synchronization is obtained by sliding window technique. Energy of the signal is found as:

$$P[n]=S[n]^2$$

The window energy is then calculated as

$$W[n] = \sum_{k=n}^{n+L} P[k]$$

where L is the length of channel in terms of symbol time. The maximum length of L is set to 4 (that makes the channel a multipath channel with delay spread value of $4*T_s$=16 μs.)

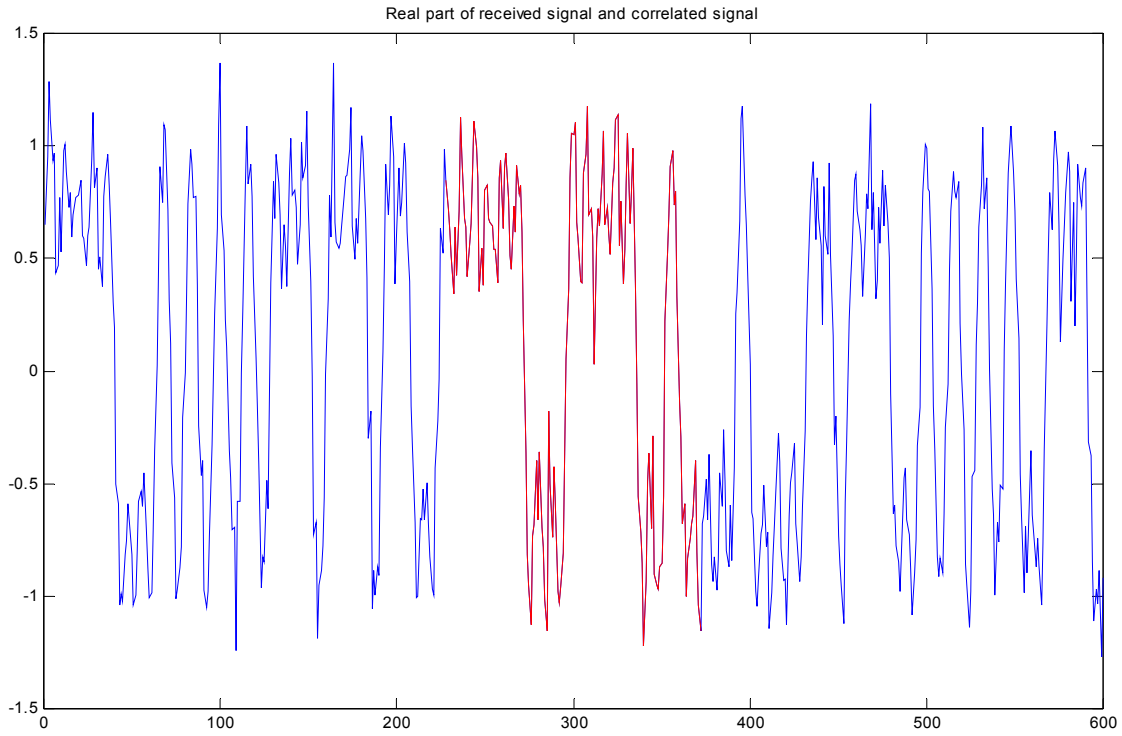


Figure 17: Received GSM burst and red points show the points for correlation

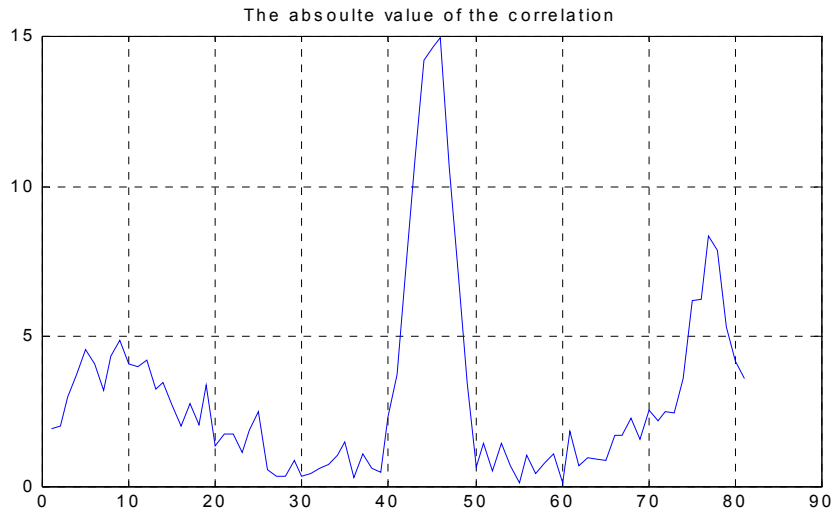


Figure 18: The cross-correlation between the received signal and the MSK-mapped equalization symbols. The peak indicates the training bits are detected in the burst, the amplitude determines the channel response.

Having obtained the synchronization, and the estimate of the channel response, the matched filtering can be done as Y=r*h*[-]. Along with matched filtering of the received signal, down-sampling should also be performed as well. Y must contain one sample per MSK symbol in the received burst. The soft-decision values are inputted to the Viterbi MLSE block.

Maximum Likelihood Sequence Estimation (MLSE): Viterbi Algorithm[16]

The input to the MLSE[17] is the matched filtered and downsampled signal, referred to as Y. It contains one sample for each transmitted symbol. The output of Viterbi is 148 bits which forms one normal GSM burst[18].

GMSK demodulation with timing error correction:

In order to get the bits out from the sampled GMSK signals, a robust, low error rate GMSK demodulator is needed. Because the sampling process in receiver and the encoding and transmission process in transmitter are separated and not synchronized, so recover the timing

[16] Graphical tutorial on convolutional codes and viterbi detection http://www.sonic.net/%7Eejr/viterbi/viterbi.htm

[17] C code for convolutional codes and viterbi detection http://people.qualcomm.com/karn/code/fec/

[18] Viterbi GSM, Viterbi Equalizer source codes on TEXAS INSTRUMENTS ,INC. web site

signal from the sampled signal is critical to correctly demodulate the Gaussian wave received. The difficulty is that the ratio of sampling rate and the bit rate are not integer or rational number, so the sampling time in a bit period will never repeat. There are various techniques to address this problem. Here, we implemented a simple algorithm, the idea is following: since the sampling rate and bit rate is incommensurate, each sample will introduce a timing error, and the timing error will accumulate, so, at certain point of the demodulation process we need to correct the timing error, the key is to decide when to perform this correction. We know that the bits are transmitted in Gaussian shaped bipolar waveform, so each time when we detected a sign change of samples we know we are at the start of next bit, so we can reset the timing error to be zero. We also know that the number of samples must be either N or N+1, where N is the maximum integer less than the ratio of sampling rate over bit rate or mathematically,

R = sampling rate/bit rate

$N = \lfloor R \rfloor$

So, the timing error introduced by each sample is (R-N)/N, the decision is based on the summation of all samples in the bit period instead of only one sample, if positive, then a bit '1' is decided, otherwise the bit is set to be '0'.

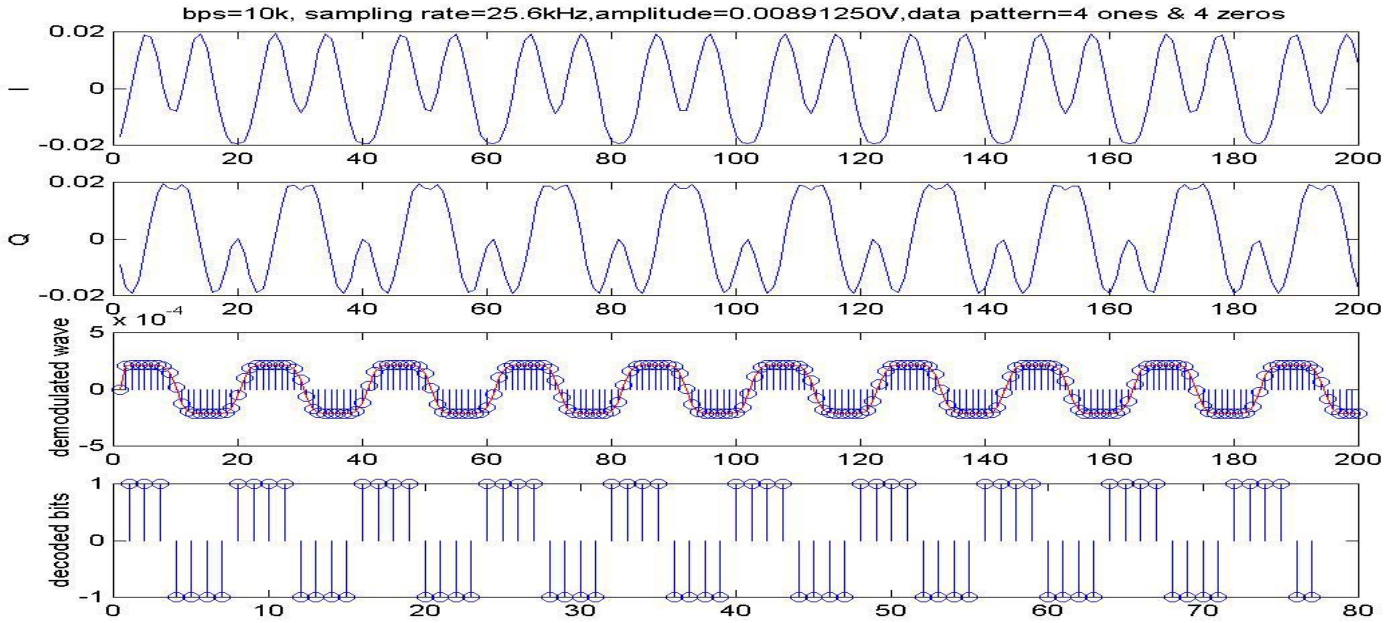
the timing correction algorithm is described by the following psudocode:

```
AccumulatedError =: 0;
DeltaError =: (R-N)/N;
For (each sample come into the demodulator){
If (the sign of current sample if different from the previous sign)
AccumulatedError =: 0;
        Else{
        AccumulatedError =: AccumulatedError + DeltaError;
                If(AccumulatedError>=1){
                AccumulatedError =: AccumulatedError – 1;
                }
        Else{}
        }
        //decision process… }
```

In order to test our algorithm, we generated the GMSK signal of predefined bit pattern by the signal generator, and captured the sampled I&Q data by the vector signal digitizer[19]. We fed those raw sampled data into the demodulator and plotted the demodulated bits, see figure

[19] Floyd M. Gardner, "A BPSK/QPSK Timing-Error Detector for Sampled Receivers", *IEEE Trans. Commun., vol. COM-34,pp. 423-429, May, 1986*

19, the bit pattern is alterative four 1s and four 0s, transmitted at the speed of 10k bps, the sampling frequency of the signal digitizer is 25.6kHz, roughly 2 samples per bit[20] [21].



V. Simulation Results

In this project, we also made some simulations to study how the channel condition affects the receiver performance, especially the bit error rate under different levels of AWGN and Rayleigh fading. The results are plotted as BER vs. SNR. Fig.20. shows the performance under the Gaussian white noise channel model

[20] Roel Schiphorst, Sabih H. Gerez and Cornelis H. Slump, "The Exploration of the Software-Defined Radio Concept by Prototyping Transmitter and Receiver Functions on a Digital Signal Processor", *Progress 2000 Workshop on Embedded Systems, Utrecht, The Netherlands, October 2000*

[21] Richard Paul Lambert, "A Real-Time DSP GMSK Modem with All-Digital Symbol Synchronization", *M.S. Thesis, Texas A&M University, May 1998*

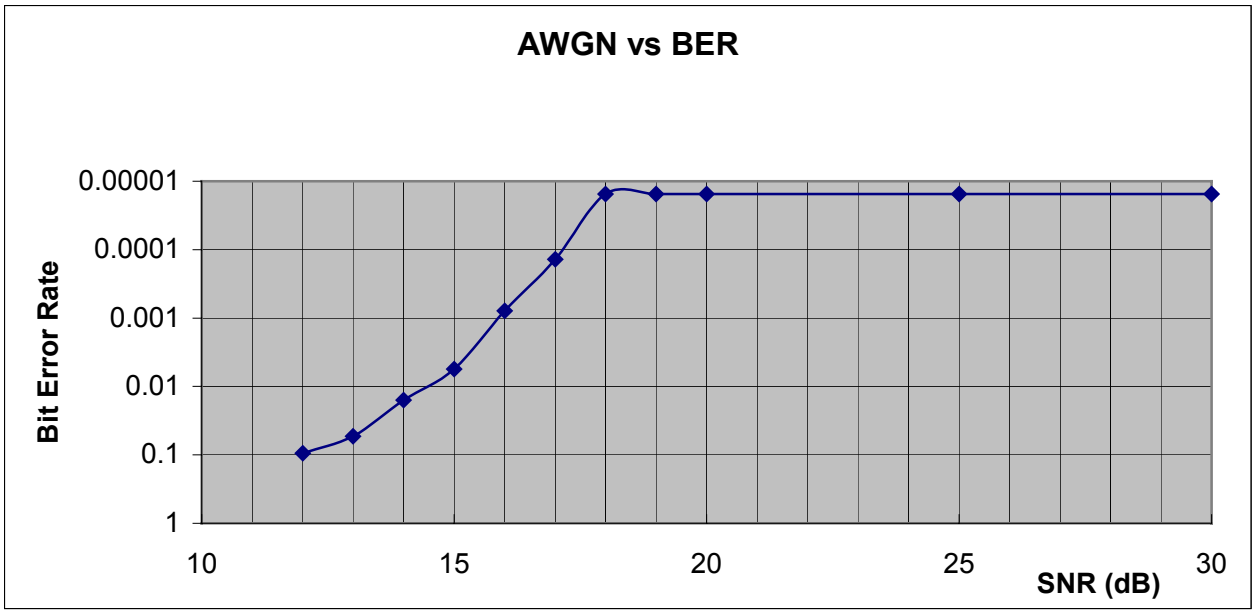


Figure 20:AWGN vs BER performance

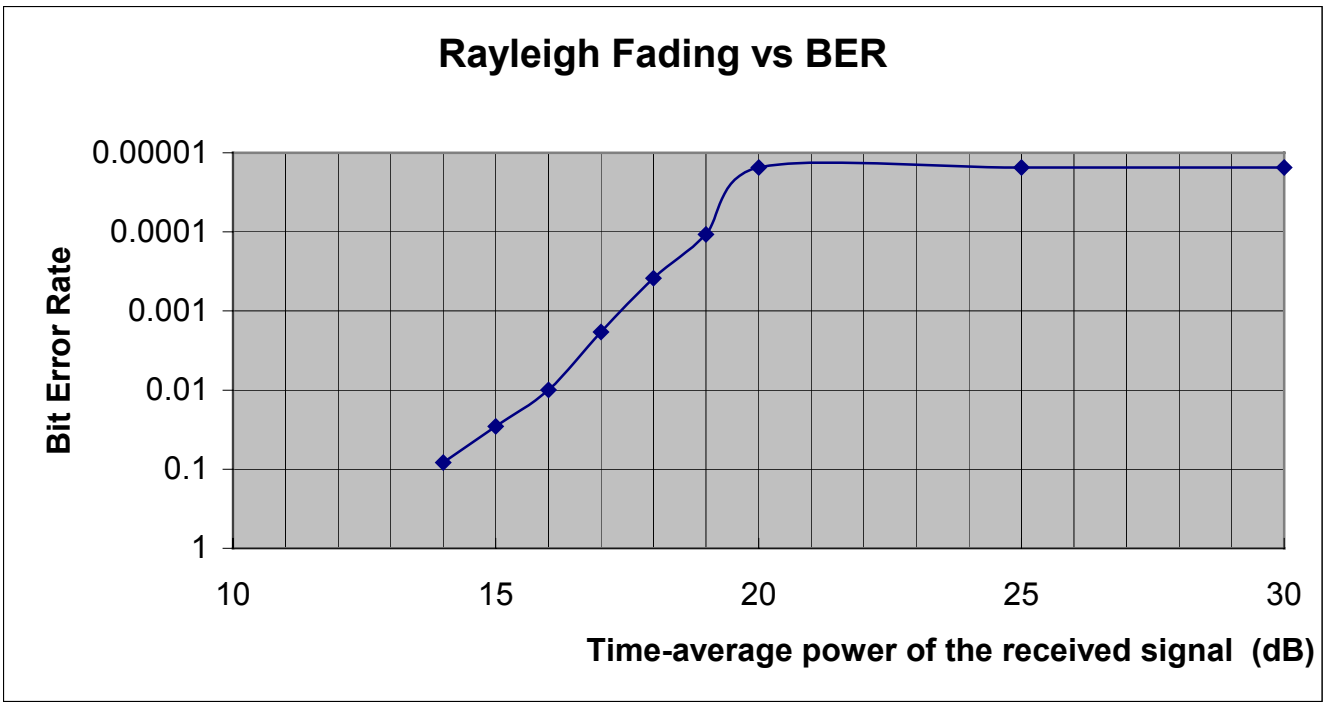


Figure 21: BER as a function of Rayleigh Fading

Table I. The performance of GSM signal with/without channel coding and interleaving. 200 GSM bursts (29,600 bits) were transmitted and demodulated.

	With interleaving/channel coding	Without interleaving/channel coding
SNR	8dB	8dB
Total errors	4300	4500
SNR	12dB	12dB
Total errors	3490	3560
SNR	18dB	18dB
Total errors	41	195
SNR	20dB	20dB
Total errors	0	0

The above figures show the performance of GSM system under different channel models, as it is shown, the receiver needs around 20 dB of SNR to get low BER values. The GSM system was also simulated with/without channel coding and interleaving. The number of errors does not change with interleaving, but maximum number of errors per burst can change with interleaving. Channel coding provides some error correction, e.g. for 18 dB of SNR, without channel coding, we have 195 bit errors, with channel coding , the number of erroneous bits is 41.

Resource used in EVM. Memory allocation, execution speed optimization

Data transfer between PC and EVM is by DMA through PCI bus, in EVM, the data processing is done in a pipelining fashion, there are two buffers allocated in the in chip memory, one is receiving buffer and one is working buffer, while the program is processing the data in working buffer, next block of data is transferred into receiving buffer in the mean time, then the two buffers are swapped, so processing can go one continuouslyl.

The total size of the program running in EVM is about 110k, the key demodulation functions are small enough to fit in the on chip memory, which will improve the speed of the program very much. Following tables give the performance before we switch on code optimiz
ation and that when the optimization is off.

Code optimization off:

Function name	Code size	Average execution cycles
Gmsk_dm_async	1216	24775
GSMDiffDecoder	208	3141
GSMDeburst	392	2040

Code optimization on:

Function name	Code size	Average execution cycles
Gmsk_dm_async	1232	14716
GSMDiffDecoder	140	1274
GSMDeburst	188	974

After optimization, the total cycles need to be decoded in EVM (GMSK demodulation, differential decoding and striping data bits from the burst structure) is about 16964 cycles, for C67, each instruction cycle is 6ns, that makes the time for decoding one burst is 16964*6 = 101784 ns, approximately 0.1 ms. Compared with the GSM specification, which specified the time for one burst is 0.577 ms, we can see that we still have enough time to do other processing, such as deinterleaving and channel decoding, though we didn't implement in EVM in this project. So, we can see, C67 is powerful enough to perform a real time GSM decoding task.

Acknowledgment:

We'd like to thanks Prof. Cassasent for his valuable suggestions and discussions and our TAs, David Wang, Matt. Juhasz, Parag Patel, their patience and support are great help. We also appreciate very much the help from the members of ECE wireless communication lab, especially Prof. Stancil, without his support, this project can't be finished.