# DSP Based Implementation of Scrambler for 56Kbps Modem

**Davinder Pal Sharma** davinder.sharma@sta.uwi.edu
*Department of Physics*
*University of the West Indies*
*St. Augustine, Tinidad & Tobago*

**Jasvir Singh** j_singh00@rediffmail.com
*Department of Electronics Technology*
*Guru Nanak Dev University*
*Amritsar -143105, India*

**Abstract**

Scrambler is generally employed in data communication systems to add redundancy in the transmitted data stream so that at the receiver end, timing information can be retrieved to aid the synchronization between data terminals. Present paper deals with simulation and implementation of the scrambler for 56Kbps voice-band modem. Scrambler for the transmitter of 56Kbps modem was chosen as a case study. Simulation has been carried out using Simulink of Matlab. An algorithm for the scrambling function has been developed and implemented on Texas Instrument's based TMS320C50PQ57 Digital Signal Processor (DSP). Signalogic DSP software has been used to compare the simulated and practical results.

**Keywords:** Scrambler, 56Kbps Modem, Matlab, Signalogic, Digital Signal Processor.

## 1. INTRODUCTION

Often, it is required by the user to ensure that the transmitted signal should has sufficient randomness or activity so that timing recovery, adaptive equalization and echo cancellation can be reliably performed. Mostly users type characters on PC relatively slow due to which computer terminal transmits null characters most of the time. A long sequence of null characters results in a highly correlated line signal. Such signals can foil timing recovery and other functions, all of which assume that the transmitted symbols are uncorrelated. Scrambling is intended to minimize strong correlations among the information bits so as to make them appear more random. It is a method of achieving dc balance (dc null) and eliminating long sequences of zeros to ensure timing recovery without redundant line coding. Scrambler does not add anything in signal, as it is not based on redundancy. Scrambler performs one-to-one mapping between input data bits and coded data bits. The objective is to map the bit sequences, which are problematic and likely to occur, into a coded sequence that looks more random and less problematic.

Scrambler is a digital device, which maps a data sequence into a channel sequence. If the data sequence is periodic, it converts it into a periodic channel sequence with period, which is many times the data period. A simple scrambler adds a Maximum-Length Shift-Registers (MLSR) sequence to the input bit stream to randomize or whiten the statistics of the data, making it more random. Scrambler consists of linear sequential filters with feedback paths, counters, storage

elements and peripheral logic in their discrete form. The counters, storage elements and peripheral logic, monitor the channel sequence but react infrequently so that the scrambler behaves principally as linear sequential filter [1].

There are many applications of scrambler. This is used for encryption of data in security systems, to remove non-linearity of common carrier systems which causes inter-channel interference and to remove systematic jitter caused by self-retiming circuits in base-band Pulse Code Modulation (PCM) systems. In data communication systems, main purpose of the scrambler is to add redundancy in the transmitted data stream so that at the receiver, timing information can be retrieved from received data i.e. to aid the synchronization between two modems. The present study deals with the implementation of the scrambler used for synchronization purpose.

## 2. THEORETICAL BACKGROUND

In the beginning, scrambler was used to reduce the effect of jitter in the PCM systems. Jitter is a very serious problem as it reduces the Signal-to-Noise Ratio (SNR) and causes more errors to be introduced at the receiver. In addition, jitter also effects the functioning of the repeaters, which has a commutative effect. The proceeding timing circuits generates systematic jitter, which degrades the transmission quality because the r.m.s. value of the jitter increases in proportion to the square root of the number of repeaters. A self-retiming circuit is used in conventional base-band PCM systems to minimize the jitter in which a timing waveform is extracted from an equalized pulse train [2]-[3].

There are certain impairments that vary with the statistics of the digital source in digital transmission systems and these statistics of the data source is related to the problem of timing recovery, equalization and cross talk. One of the methods to isolate the system performance from the source statistics is to use redundant transmission codes. These codes could not provide complete isolation however they generate additional problems by increasing the symbol rate. Alternative method to cope up with this problem is scrambling, which whitens the statistics of digital source. Any source is said to white if it generates statistically independent and equiviprobable symbols using which system impairment can be easily analyzed. All the first order and second order statistics of any binary source can be whiten to any degree at the cost of an arbitrarily small controllable rate [4].

## 3. BASIC SCRAMBLING ACTION

Data transmission through any communication system will be errorless if the timing of each device attached to the system is accurate enough. But it is difficult to ensure the accuracy of the timing in a larger system having many devices or having very large data packets. The solution to this problem of highly accurate clocks can be found by using synchronous communication techniques. In synchronous systems, it is necessary to extract timing information from the received data, which in turn reduces the burden of internal clock circuitry. There are many methods using which timing information from the received data can be retrieved. Line coding techniques like Return to Zero coding and Manchester coding are the few familiar techniques that may be used for this purpose [1]. The problem with the usage of these line-coding techniques is that the timing benefits come at the expense of bandwidth, so these techniques are not used in Public Switched Telephone Network (PSTN), which is severely band-limited network, so one has to choose other options. Scrambling is a technique that can be used in conjunction with simpler line coding algorithms such as simple binary and RS232C protocol to achieve above-mentioned goal without sacrificing the bandwidth requirement.

In digital system, it is common practice to encounter long strings of 1's and 0's within the transmitted data that results in constant output levels. Timing information cannot be retrieved from such outputs because there will be no state-transition during these sequences which may result in transmission errors at the receiver. Scrambler can eliminate this problem by detecting undesirable sequences of bits and inserting state transitions in a pseudo random manner. If there is a long sequence of 1's, 0's will be pseudo randomly inserted in to the stream. It ensures that the probability of receiving a '1' is equal to the probability of receiving '0' and minimizes the probability of periodic or repetitive data transmission, which in turn make clock recovery easier [5]. Simple scrambling action of a scrambler is shown in Fig. (1). While it may not be possible to prevent the occurrence of all undesirable sequences with absolute certainty, at least most of the common replications in the input data stream can be removed by the use of a scrambler. It has been found that the transmission of short repetitive patterns could play havoc with both the equalizer and timing recovery systems [2].

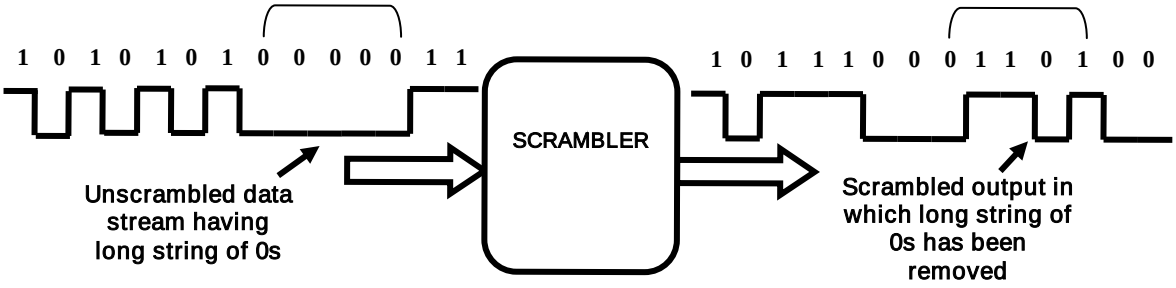



**FIGURE 1:** Scrambling Action

## 4. SYSTEM DESIGN

There are many ways to implement scrambler but all rely on the same basic building blocks of linear feedback shift registers and modulo-2-addition functions. Many researchers have discussed the general theory of implementation of scramblers [2-4], [6-7]. In general, the serial data enters in linear feedback shift register, where each stage in the register delays the signal by one time unit as shown in Fig. (2). The delayed version of the output signal is then fed back and

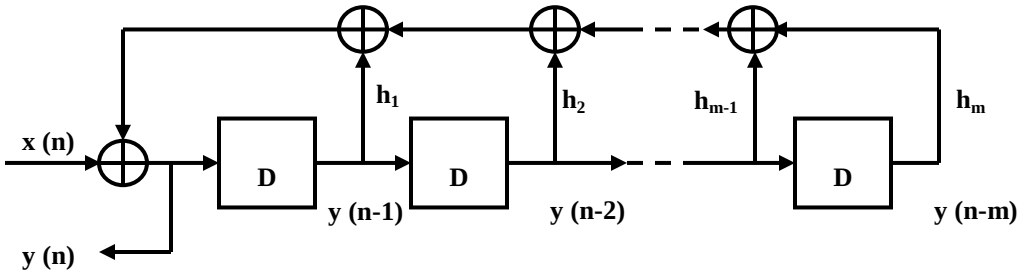



**FIGURE 2:** Basic Structure of a Scrambler

modulo-2-addition is performed with the input signal. The scrambler's input and output relation, in general, is given by [1]-[2] :

$$y(n) = x(n) + \sum_{k=1}^{m} h_k y(n-k) \tag{1}$$

Where y(n) is current output, y (n-k) is the output delayed by k times, x(n) is current input and $h_k$ is system transform function. All the constants and variable in eqn. (1) can only have the values '0' or '1' and all additions performed are modulo-2-additions (exclusive-OR operations). Corresponding state vector s(n) to this eqn. can be written as

$$\begin{aligned} s(n) &= [y(n-1), y(n-2),..., y(n-m)] \\ &= [s_1(n), s_2(n),..., s_m(n)] \end{aligned} \tag{2}$$

In general scrambler is of two types; Frame synchronized scrambler and self-synchronized scrambler. Frame-synchronized scrambler is used for cryptography purposes whereas self-synchronized scrambler is used for clock or carrier recovery purpose and it is this scrambler, which is used in modem design.

Both scramblers are based upon maximum-length shift-register sequence or M-sequences, which are periodic bit sequences with properties that make them appear to be random. These sequences can be generated using a feedback shift register, which is basically a linear sequential filter with feedback paths. Binary m-stage linear feedback shift register is shown in Fig. (3) [8]. Binary sequence of this maximum length is known as M-sequence or pseudo-random binary sequence because it satisfies several statistical tests for randomness. The auto-correlation function of such sequences resembles with the white noise. While implementing such device, the design problem is to select the shift register taps, which generate a M-sequence. The theory behind it is based on finite fields so it involves algebraic polynomials and finite field arithmetic (modulo-2-addition). Polynomials, which generate 'M' sequences, should be primitive. A polynomial y(x) of degree 'm' is primitive if it is irreducible i.e. has no factors except 1 and itself and if it divides $x^k+1$ for $k = 2^m-1$ and does not divide $x^k+1$ for $k< 2^m-1$. Sequence period of such primitive polynomials is $2^m-1$. Characteristic polynomial for M sequence generator is given by [2, 9]:

$$y = 1 + \sum_{k=1}^{m} h_k x_k \tag{3}$$

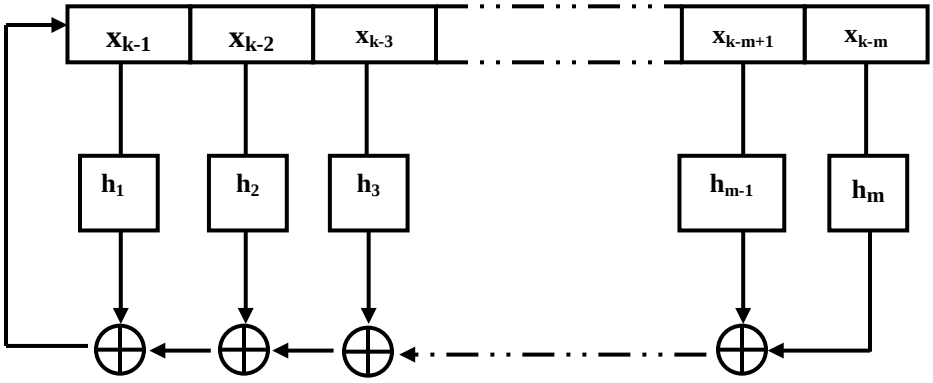



**FIGURE 3:** Binary m-stage Feedback Shift Register

Let x(n) be the binary sequence at the input of scrambler. Taking D-transform (Huffman transform; like z-transform with $D = z^{-1}$ ) of the incoming sequence as

$$X(D) = \sum_{n=0}^{\infty} x(n) D^n \tag{4}$$

Then the connecting polynomial for the scrambler given in Fig. (2) can be written as

$$h(D) = 1 + \sum_{k=1}^{m} h_k D^k \tag{5}$$

Output transform with zero initial state is given by

$$y(n) + \sum_{k=1}^{m} h_k . y(n-k) = x(n) \tag{6}$$

Taking D-transform on both sides of the above equation, we have:

$$Y(D)h(D) = X(D) \tag{7}$$

The output of the scrambler is therefore given by

$$Y(D) = X(D) / h(D) \tag{8}$$

So mathematical operation performed by the scrambler is basically equivalent to dividing the input information sequence by a Generating Polynomial (GP). The polynomial resulting in the fewest feedback connection is often the most attractive for scrambling purpose. Scrambling action of a simple five-tap scrambler specified by the generating polynomial

$$y(x) = 1 + x^{-3} + x^{-5} \tag{9}$$

is shown in Fig. (4).

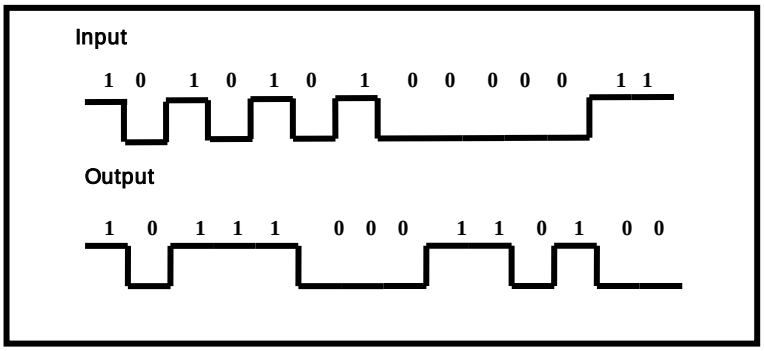



**FIGURE 4:** Input and Output Waveforms of a Five-tap Scrambler

There are many types of characteristic polynomials that can be used in modems. Some of the popular generating polynomials are those which have been used in the CCITT V.22 and CCITT V.27 protocols. The V.22 polynomial is the one in which seventeen-stage operation is recommended [6]. The CCITT V.27 protocol suggests a seven-stage register. Finally, the CCITT V.29 recommends the 23-stage register to implement generating polynomial. Everyone has his own preference regarding which polynomial to use but all rely on the same basic cells; shift registers and modulo-2-adders.

## 5. SCRAMBLER IN 56KBPS MODEM

56Kbps modem uses PCM in the downstream (server to client). Clock recovery circuit of the modem generally uses high-Q resonant tank circuit, which averages the clock phase over several bit periods. Clock phase resulting from sampling a constant signal level will attain a static value and if this bit pattern changes to another pattern, the implementation of clock recovery circuit will force the clock phase to change and attain a new value. The serial PCM bit stream reflects the deterministic parts of the signal and this leads to clock phase jitter in the repeater. Its r.m.s. value increases linearly as square root of the number of cascaded repeaters. Since a clock phase change is equivalent to timing jitter, the data eye will not always be sampled at the optimum time and so there will be an increase in bit error probability. Scrambling is one of the effective methods for the suppression of this jitter [10-11].

The data scrambler specified for the transmitter of 56Kbps digital modem is designed using the following generating polynomial [12]:

$$y(x) = 1 + x^{-18} + x^{-23} \tag{10}$$

This is a self-synchronizing scrambler, which means that it will not only provide clocking information throughout the transmission but also will be used to create the signals for the initial handshaking between the modems on the receiving and transmitting end. If there is an error free transmission then one can use the same generating polynomial with the feed-forward shift-register device for de-scrambling but if there is any error then one have to use different polynomial. Block diagram of 56Kbps digital modem's scrambler is given in Fig. (5).

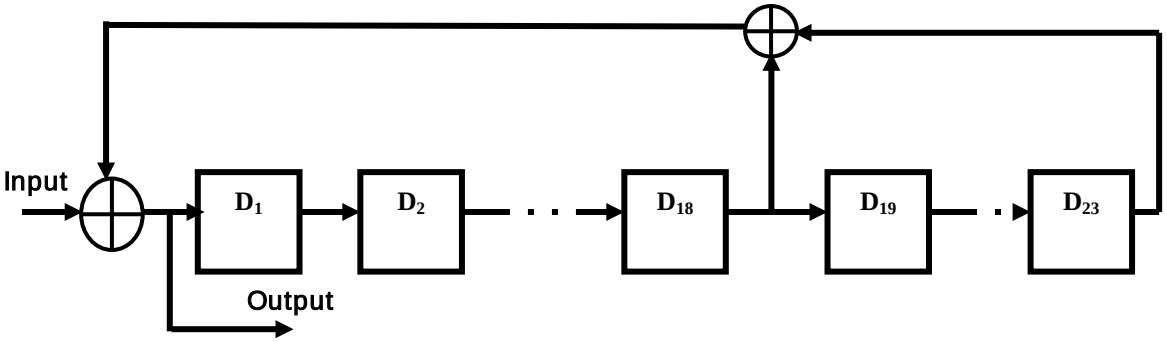



**FIGURE 5:** Scrambler for 56Kbps Digital Modem

## 6. SIMULATION OF SCRAMBLER USING MATLAB

Matlab has been used to study the scrambling action in the transmitter of 56Kbps digital modem which uses generating polynomial given by eqn. (10). Using Simulink toolbox of Matlab, a model of scrambler as shown in Fig. (6), corresponding to the eqn. (10) has been developed and its performance against input binary sequences with different probabilities of occurrence of 0's (**P**) has been evaluated. Model contains a Bernoulli Random Binary Generator (BRBG) block, which generates random number using Bernoulli distribution [13].

The Bernoulli distribution produces zeros with the probability **P** and ones with the probability **1-P**. The Bernoulli distribution has mean value **1-P** and variance **P (1-P)** [14]. The scrambler block scrambles the input signal (the output of BRBG) using scramble polynomial parameter **ρ**, which

defines the feedback connections in the scrambler i.e. feedback connection from 1st, 3rd and 5th delay elements can be represented as $\rho$ = [0,-3,-5] or as $\rho$ = [1 0 0 1 0 1].

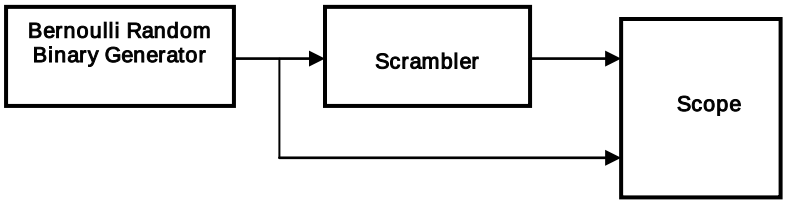



**FIGURE 6:** Matlab Model to Study the Relation between P and R in a Scrambler

Simulation results corresponding to different values of **P** in time domain were obtained and from these results, the number of transitions in input data ($T_{i/p}$) and in output data ($T_{o/p}$) have been calculated. The ratio of $T_{o/p}$ to the $T_{i/p}$ gives the Randomization Parameter **R** which can be considered as a measure of performance of the scrambler, **R** therefore describes how effectively scrambler randomizes the incoming binary signal. Fig. (7) gives the graphical representation of the results obtained and hence it can be concluded that scrambling action of the scrambler is more effective for the input data sequences having high value of probability of occurrence of zeros or ones i.e. the signal having long sequences of 0s or 1s. Scrambler is less effective for the input data sequences having frequent transitions and more over no scrambling is needed for such inputs because these sequences have sufficient number of transitions in it, using which clock information can easily be retrieved.

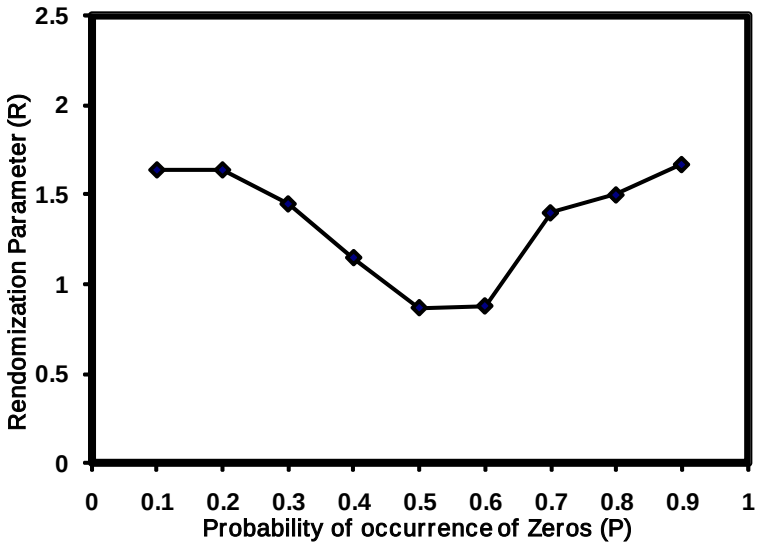



**FIGURE 7:** Variation of Randomization Parameter (R) With the Probability of Occurrence of (P) For Scrambler With GP Given in Eqn. (10).

## 7. IMPLEMENTATION OF SCRAMBLER ON DSP CHIP

Scrambler for 56Kbps server (digital) modem has been implemented on TMS320C50PQ57 DSP chip using MICRO-50EB evaluation module. Experimental setup used for the present implementation is shown in Fig. (8). Evaluation module contains a 16 bit fixed point processor TMS320C50PQ57 along with 48-Kiloword (KW) of monitor EPROM, 16KW of program RAM, 32 KW of data RAM & 32KW of I/O RAM. The highly paralleled architecture and efficient instruction set provide speed and flexibility which make the TMS320C50PQ57 DSP to capable of executing about 57 Million Instructions Per Second **(MIPS)**. TMS320C50PQ57 DSP optimizes speed by

implementing functions in hardware that other processors implement through micro-code or software. This hardware intensive approach provides high processing power previously unavailable on single chip. Its powerful instruction set, inherent flexibility, high-speed number-crunching capabilities, and innovative architecture have made this high-performance, cost effective processor; the ideal solution to many telecommunications, commercial, industrial and military applications [15].

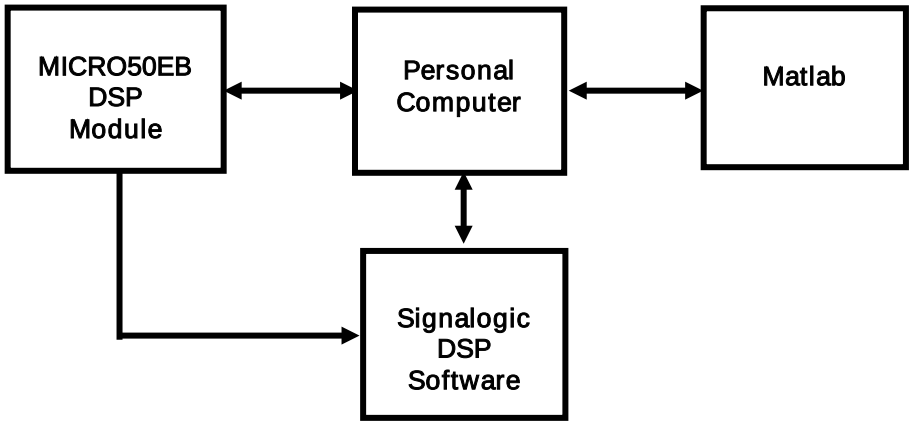



**FIGURE 8:** Experimental Setup for Soft Implementation of Scrambler of 56Kbps Digital Modem on DSP

### 7.1. Algorithm for Soft Implementation of the Scrambler

Scrambler has been implemented using the circular buffers of the TMS320C50PQ57 DSP chip. The chip has special memory mapped registers and associated circuitry for two circular buffers. One of the eight auxiliary registers can be used as a pointer into the circular buffer. Circular buffer was implemented on the DSP on-chip memory block. Two memory-mapped registers are associated with each circular buffer that needs to be initialized with the start and end address. Block diagram of I/P sample circular buffer is given in Fig. (9), where x(n) is the input data at a time n, x(n-1) is input data delayed by single time unit and N is length of linear feedback or feed forward shift register. Initialized circular buffer can be used for producing delay and data can be read from any position pointed by the auxiliary register AR-1. Then XOR operation can be performed on delayed samples to achieve scrambling action. Before the initialization of circular buffer we have to initialize the DSP chip and I/O devices [9].

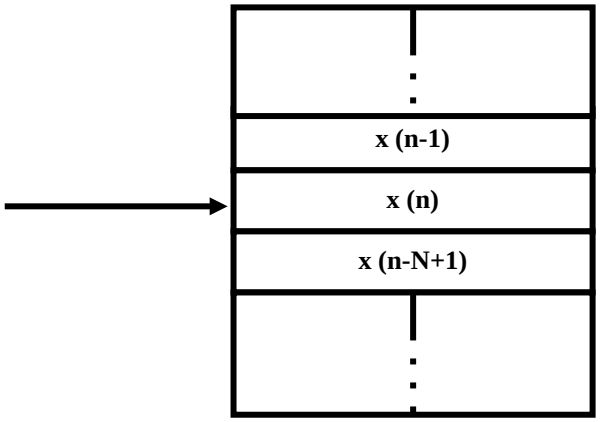



**FIGURE 9:** Structure of Circular Buffer

In present implementation TLC32044 AIC (Analog Interface Circuit) has been used as I/O device. The supporting algorithm for the initialization of DSP module and algorithm for implementation of scrambler of digital modem is given in Appendix (A) along with the algorithm of main scrambling routine using circular buffer.

Assembly level program (source code) for the scrambler of 56Kbps digital modem has been written using algorithm given in Appendix (A). Source code were loaded into the DSP and output data obtained was stored at appropriate data memory location with the slight modification in the program i.e. one more step of data storage in addition with the data transmission has been included in the main program. Stored data was loaded into the Signalogic DSP Software for time and frequency domain analysis of practically obtained results. In the similar fashion output data obtained from the simulation was loaded in to the software package and simulated and practical results were compared in time as well as frequency domain.

Time domain comparison of scrambler's simulated and practical result is shown in Fig. (10). It can easily be concluded from this figure that the present practical result differs very slightly from the simulated results. The transition which was occurring at 3.375ms in the simulation study has been get shifted to 3.875 in actual practice but this shift does not disturb the scrambling action because total number of transitions are same. Similar study in frequency domain has been carried out for simulated and practical results. Spectral analysis i.e. variation of magnitude with frequency of simulated and practical results is presented in Fig. (11). It is clear from this figure that the practical and simulated results are almost same, which confirms the successful implementation of scrambler on DSP.

Present implementation is more efficient than the earlier implementation reported by Steven A.Ttretter, C.J.Buechler and H. Sampath [16]. In the present implementation circular buffer is being used to produce delay and on chip memory of the DSP chip is being used due to which memory requirement as well as execution time have been reduced by using efficient algorithm discussed earlier. Comparison of various implementation parameters like program execution time, program and data memory used for the current and previous study is given in the Table (1).

| Implementation | Program Execution Time (µsec) | Program Memory Used (W) | Data Memory Used (W) |
|---|---|---|---|
| Present | 3.2 | 64 | 23 |
| Previous | 5 | 100 | 15 |

**TABLE 1:** Various Implementation Parameters

## 8. CONCLUSION

Scrambler for 56Kbps digital modem has been simulated using Matlab and implemented using TMS320C50PQ57 DSP chip during the present study. From the simulation it has been found that scrambling action of proposed scrambler is more effective for the input data sequences having high value of probability of occurrence of zeros or ones i.e. the signal having long sequences of 0s or 1s. Scrambler is less effective for the input data sequences having frequent transitions and more over no scrambling is needed for such inputs because it has sufficient number of transitions in it, from which, clock information can easily be retrieved. In present implementation circular buffer is being used to produce delay and on-chip memory of the DSP chip is used due to which program memory requirement as well as execution time has been get reduced using the efficient algorithm as compared to previous study. Simulated and practical results have been compared

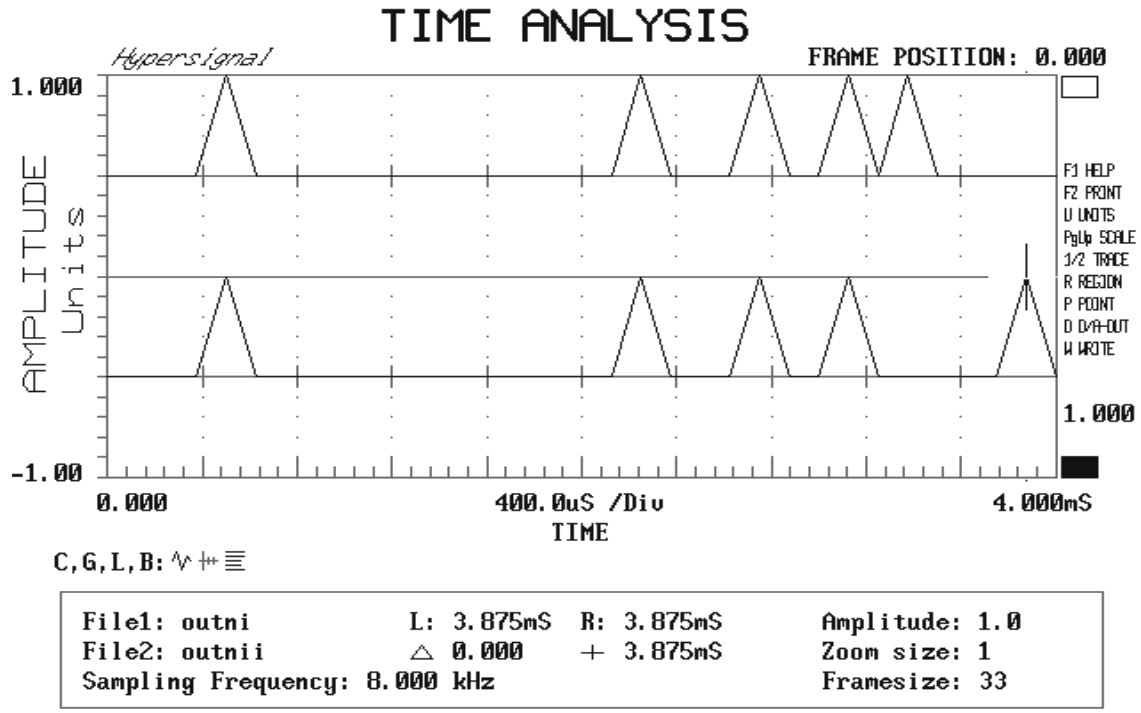



**FIGURE 10:** Comparison of Simulated and Practical Results of Scrambler in Time Domain

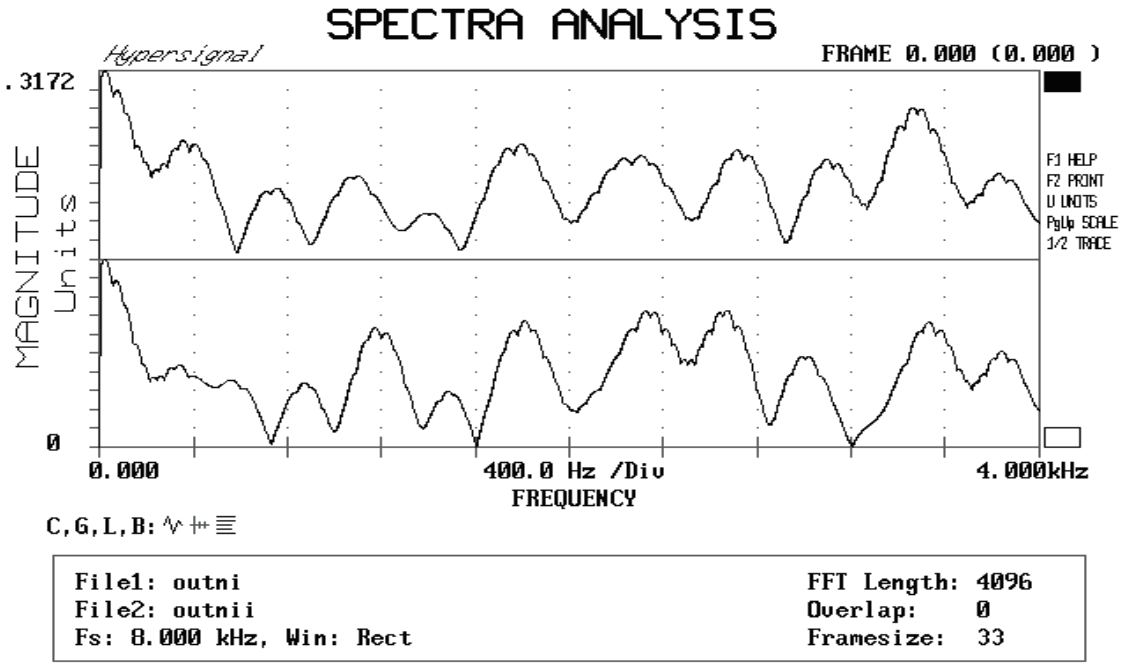



**FIGURE 11:** Comparison of Simulated and Practical Results of Scrambler in Frequency Domain

using Signalogic DSP Software in time as well as frequency domain and have been found same which confirms the successful implementation of scrambler on DSP.

## ACKNOWLEDGEMENTS

One of the authors Davinder Pal Sharma is thankful to Guru Nanak Dev University, Amritsar, for providing DSP Research facilities at Department of Electronics Technology for present research work.

## 9. REFERENCES

1. J. G. Proakis, M. Salehi. "*Digital Communications",* McGraw Hill, (2007).

2. R. D. Gitlin, J. F. Hayes. "*Timing recovery and scramblers in data transmission*", Bell System Technical Journal 54(3): 569 – 593, 1975.

3. H. Kasai, S. Senmoto and M. Matsushita, "*PCM jitter suppression by scrambling*", IEEE Trans. on Communications, 22(8): 1114-1122, 1974.

4. A. Huzii, S. Kondo. "*On the timing information disappearance of digital transmission systems*", IEEE Trans. on Communications**,** 21(4):1072-1074, 1973.

5. R.L. Freeman. "*Practical Data Communications*", John Wiley and Sons Inc., (1995).

6. C. H. Lin et.al. "*Parallel scrambler for high speed applications*", IEEE Trans. on Circuit & Systems-II, 53(7): 558-562, 2006.

7. M. Cluzeau. "*Reconstruction of a linear scrambler*", IEEE Trans. on Computers**,** 56(9): 1283-1291, 2007.

8. "*Data Scrambler / Descrambler with Look Ahead*", IBM Technical Disclosure Bulletin, 28**:** 1063-1064, 1985.

9. S. A. Tretter. "*Communication System Design Using DSP Algorithms: With Laboratory Experiments for the TMS320C6713 DSK*", Springer, 163-171 (2008).

10. ITU-T Recommendation V.92. "*Enhancement to Recommendation V.90",* International Tele communication Union, 2000.

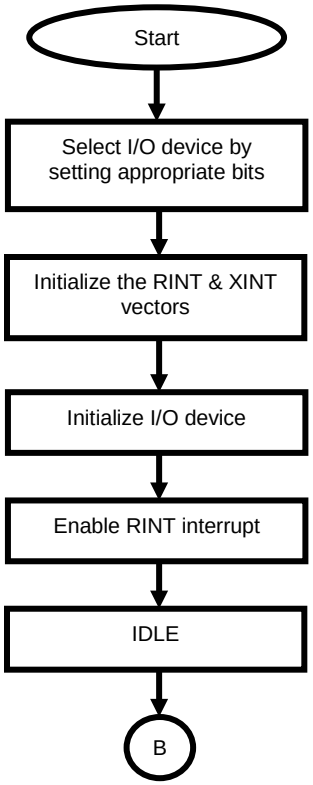
11. D. Y. Kim et. al. "*V.92: The last dial-up modem*", IEEE Trans. On Communications**,** 52(1): 54-61, 2004.

12. ITU-T Recommendation V.90. "*A digital and analog modem pair for use on the PSTN at data signaling rates of up to 56000 bits/s downstream and 33600 bits/s upstream",* International Tele communication Union, 1998.

13. "*Simulink User Guide*", The Mathworks Inc. (2009), http://www.mathworks.com.

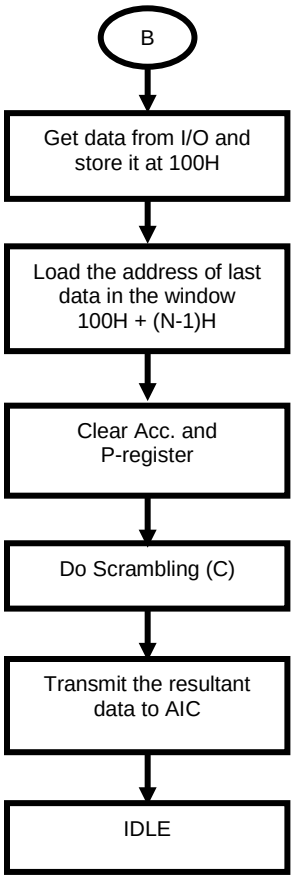
14. G. Zimmermann. "*Probabilities of long sequences of identical bits after data scrambling*". In Proceeding of IEEE International Conference on Communications, 1990, 1521-1525.
15. "*Micro-50 EB user manual*", DSP Series, Vi Microsystems Pvt. Ltd., (2000).
16. S. A. Tretter et. al. " *V.34 Transmitter and receiver implementation on the TMS320C50 DSP*", Digital Signal Processing Solutions, Texas Instruments, USA, 1997.

## APPENDIX (A)

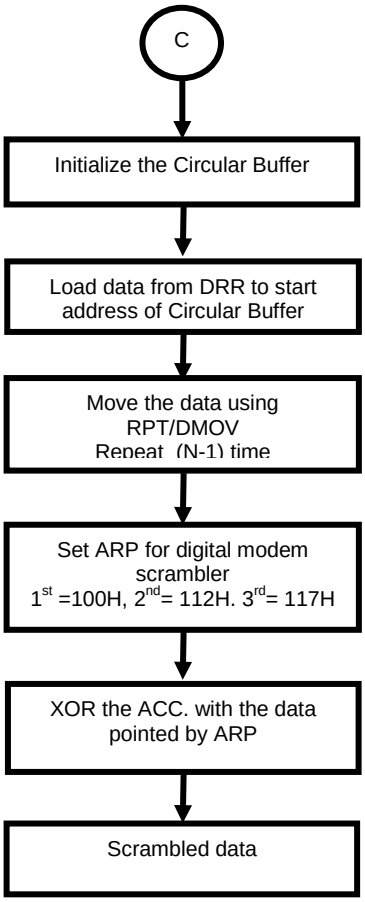



RINT ISR

Algorithm for initialization of DSP Module




Algorithm for initialization of scrambling function of 56Kbps digital modem.




Main scrambling routine