Analysis of Verilog Modelling and Simulation of Developed 16-bit and 23-bit LFSR Based Data Scrambler and Descrambler for 2.5GT/s, 5GT/s and 8GT/s Data Rate with Minimum Glitch

Monirul Islam, Rajibul Alam, Abdullah Al Hadi

Abstract— Verilog structured and simulation of data scrambler and descrambler for noise free and secure data communication has been presented here. This paper analyses the data scrambler and descrambler considering 2.5 GT/s, 5 GT/s and 8 GT/s data rate. Transmitted bits have some certain characteristics which have effects on the error rate and the achievable bandwidth. This characteristic includes (1) ratio's of 0's and 1'swithin a data byte and (2) maximum no' of clock periods between bit transitions (i.e. 1 - 0 and vice versa). These characteristics are maintained and developed using an efficient scrambler. Scrambler increase the ability of a receiver performing clock recovery or deriving improved bit synchronization, helps distinguished data bit sequences, allows simple detection of byte and word boundaries. The performance results of the simulation have been found without any delay and are in conformity with theoretical observations.

Index Terms— Data Rate, Descramble, EMI, Glitch, LFSR, PHY, Scrambler.

1 Introduction

THE development of new technologies bring the evolution of new and sensitive communication capabilities in the form of Internet, electronic banking, electronic mail, pay channel television, cable television, mobile telephone, satellite phone, broad band services, e-journals, tele-medicine and above all several forms of electronic communications. However, any forms of electronic communications are vulnerable to interference [1, 2].

In telecommunications and recording, a scrambler is a device that manipulates a data stream before transmitting [1,2]. The manipulations are reversed by a descrambler at the receiving side. Scrambling is widely used in satellite, radio relay communications and PSTN modems. A scrambler can be placed just before a FEC coder, or it can be placed after the FEC, just before the modulation or line code. The pseudo-noise (PN) key generation is of paramount importance for any secure communication system [2]. PN sequences based on Linear Feedback Shift Registers (LFSR) and non linear combination based implementations are simplest to give moderate level of security.

The purpose of the scrambler is to eliminate a repetitive pattern on the data stream. A repetitive pattern on a 2.5 Gbps data stream (such as 10101010) can generate significant EMI noise. By scrambling the data stream repetitive patterns are eliminated and thus spread the EMI energy over a broader range in the spectrum [8]. This scrambling technique is often referred to as spread spectrum and is an effective way of whitening the noise.

The 16-bit Scrambler/De-scrambler allows scrambling and de-scrambling of two 8-bit symbols in parallel. It is intended to be used with PHY chips using 16-bit PIPE interface.

- *Monirul Islam is currently working as an ASIC Design Engineer in Fastrack Anontex Limited in Dhaka, Bangladesh, PH-+8801918226190. E-mail: monirul@fastrack-design.com*
- *Rajibul Alam is currently working as an ASIC Design Engineer in Fastrack Anontex Limited in Dhaka, Bangladesh, PH-+8801715345075. E-mail: rajibul.alam@fastrack-design.com*
- *Abdullah Al Hadi is currently pursuing masters degree program in energy engineering and management in Instituto Superior Technico, IST, Lisbon, Portugal, PH-+351969642762. E-mail: abdullah.al.hadi.bd@gmail.com*

2 Theoritical Analysis

Electromagnetic interference or (EMI) is disturbance that affects an electrical circuit due to either electromagnetic induction or electromagnetic radiation emitted from an external source [6]. The disturbance may interrupt, obstruct, or otherwise degrade or limit the effective performance of the circuit. These effects can range from a simple degradation of data to a total loss of data [7].

Integrated circuits are often a source of EMI. On Digital integrated circuits, important means of reducing EMI are scrambling [8]. At higher frequencies, usually above 500 MHz, traces get electrically longer and higher above the plane.

Scrambler and descrambler are very commonly used in PHY chips like Universal Serial Bus (USB), PCI Express and in different protocols.

The scrambling function can be implemented with one or many Linear Feedback Shift Registers (LFSRs). A linear feedback shift register (LFSR) is a delay line which feeds back a logical combination of its stages to the input. On the Transmit side, scrambling is applied to characters prior to the encoding. On the Receive side, de-scrambling is applied to characters after decoding [8].

16-bit LFSR –

For 2.5 GT/s and 5 GT/s data rates The LFSR implements the following polynomial based on the encoder and decoder scheme:

$$G(X) = X^{16} + X^5 + X^4 + X^3 + 1 \quad (1)$$

Scrambling or unscrambling is performed by serially XORing the 8-bit (D0-D7) character with the 16-bit (D0-D15) output of the LFSR.

If the lane number increases due to higher data rates the seed values will be changed on the basis of the LFSR. For 130 bit data stream the above mentioned LFSR produces secure 1's and 0's trail.

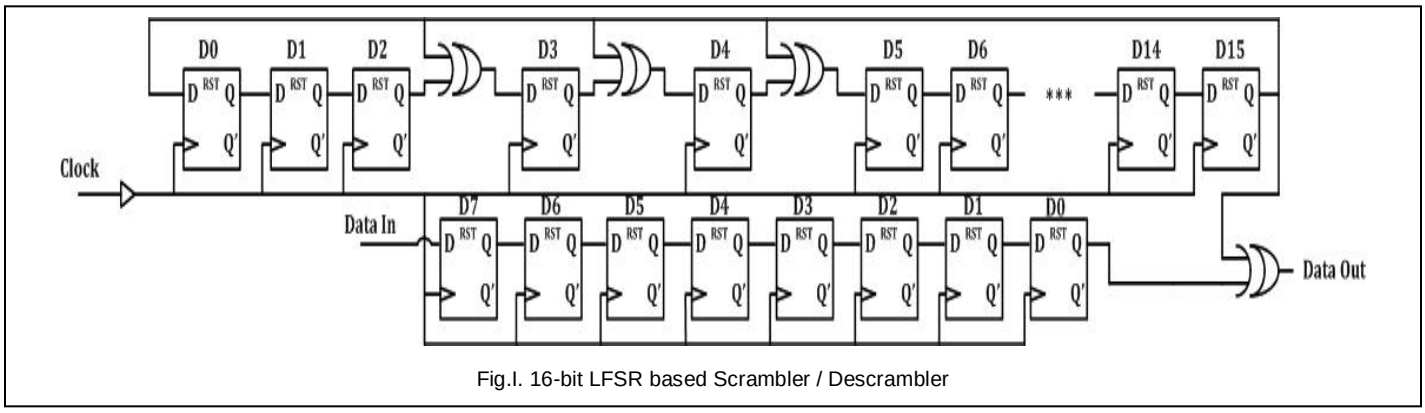

Fig.I. 16-bit LFSR based Scrambler / Descrambler

The structure of scrambler/descrambler system is shown in Figure I. It consists of 16 D flip-flops (D0-D15) for LFSR and 8 (D0-D7) for Input Data. At D3, D4, D5 in LFSR the input is XORed with D15 value. With every clock cycle the Data and the LFSR value is shifted right. After 8 cycle , *Data Out* started to generate depend on the *Data In*. The initial value of the LFSR is set to FFFFh.

23-bit LFSR –

The LFSR uses the following polynomial for data rate 8.0 GT/s or higher and is demonstrated in Figure II.

$$G(X) = X^{23} + X^{21} + X^{16} + X^8 + X^5 + X^2 + 1 \quad (2)$$

The data is descrambled at the receiver using the inverse of the scrambling function. The purpose is to generate a more randomized content of 0's and 1's and generating a frequency spectrum which more closely approximates a Gaussian distribution.

3 Results And Discussions

The analysis of the module is executed using **modelsim** and **quartus** software tool. The simulation graphs and list are elaborated here.

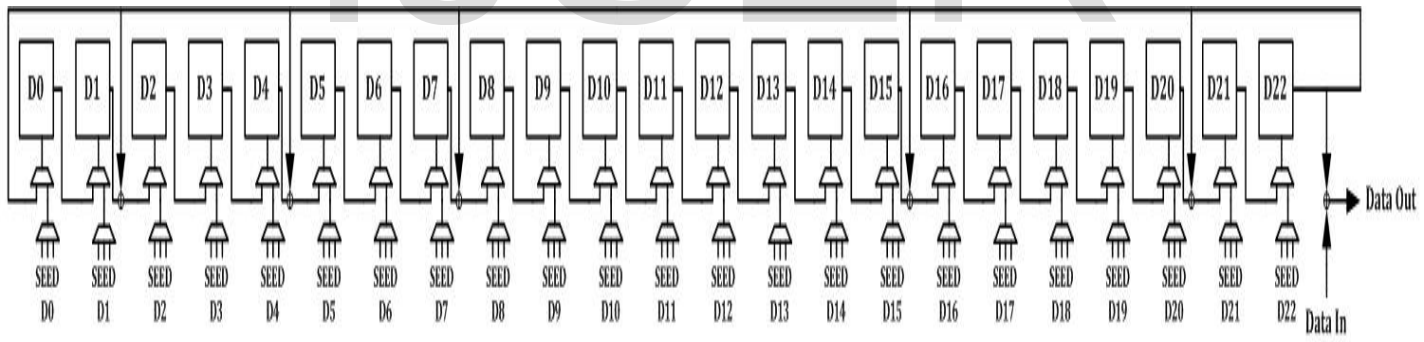

Fig.II. 23-bit LFSR based modulo 8 Scrambler / Descramble

The scrambler/descrambler system in Figure II consists of 23 D flip-flops for LFSR. In each cycle the *Data In* is XORed with the output of D22 flip-flop and thus generates the *Data Out*. At D2, D5, D8, D16, D21, the input is XORed with the output of D22. The seed value of the LFSR is dependent on the usage of the LFSR in the chip. The seed values for modulo 8 are:

0: 1DBFBCh, 1: 0607BBh, 3: 18C0DBh, 4: 010F12h, 5: 19CFC9h, 6: 0277CEh, 7: 1BB807h

For modulo 8 scrambling the initial value proposed here.

Scrambler – The LFSR is advanced 8 serial clocks which reduce the timing delay. For indicating the control bit a special input *k_code* is used and for the ordered sets in the PHY chip *training_sequence* is used. Input *inbyte* is a 8 bit value and the following scrambled output *outbyte* list is given here sequentially. In the case of *k_code* and *training_sequence* the output will be unscrambled. Other cases Scrambling is done depending on the LFSR value.

Simulation results :

0 k_code = 0 training_sequence = 0 inbyte = 00000000

outbyte = 00000000
# 20 k_code = 0 training_sequence = 1 inbyte =10111100 outbyte = 10111100
# 30 k_code = 0 training_sequence = 0 inbyte =10111101 outbyte = 01000010
# 40 k_code = 0 training_sequence = 0 inbyte =01011111 oubyte = 01001000
# 50 k_code = 0 training_sequence = 0 inbyte =10011101 outbyte = 01011101
# 60 k_code = 0 training_sequence = 0 inbyte =00011100 outbyte = 00011100
# 70 k_code = 0 training_sequence = 0 inbyte =11101100 outbyte = 11111000
# 80 k_code = 1 training_sequence = 0 inbyte =01011100 outbyte = 01011100
# 90 k_code = 0 training_sequence = 0 inbyte =01010101 outbyte = 11100111
# 100 k_code = 1 training_sequence = 0 inbyte = 00011100 outbyte = 00011100
# 110 k_code = 0 training_sequence = 0 inbyte = 01010101 outbyte = 10110010
# 120 k_code = 0 training_sequence = 0 inbyte = 10001100 outbyte = 10001110

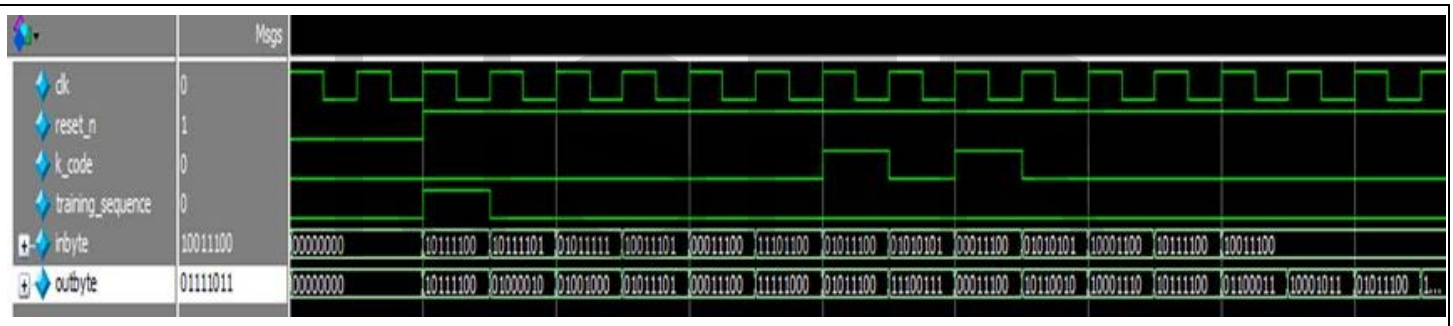

Fig.III. Scrambled output in the transmitter

130 k_code = 0 training_sequence = 0 inbyte = 10111100 outbyte = 10111100
140 k_code = 0 training_sequence = 0 inbyte = 10011100 outbyte = 01100011

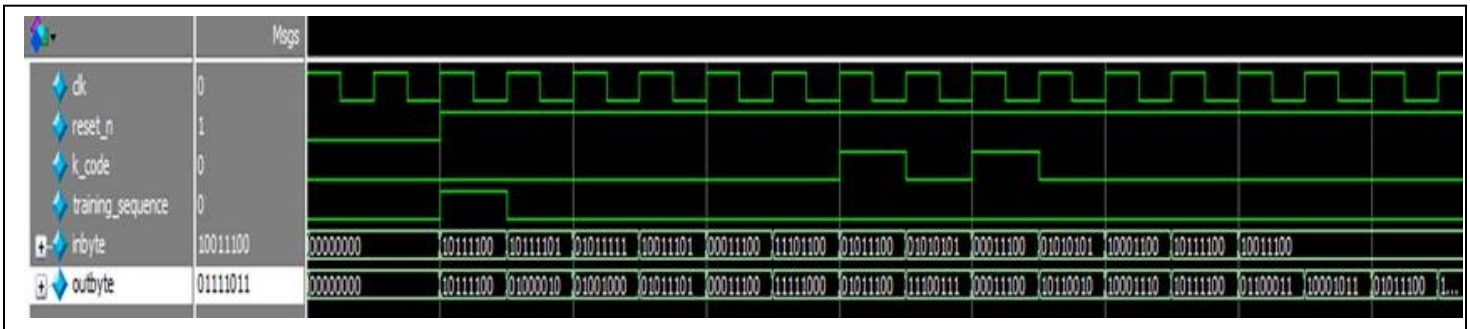

Fig.IV. Descrambled output in the receiver

150 k_code = 0 training_sequence = 0 inbyte = 10011100 outbyte = 10001011
160 k_code = 0 training_sequence = 0 inbyte = 10011100 outbyte = 01011100
170 k_code = 0 training_sequence = 0 inbyte = 10011100 outbyte = 10001000
180 k_code = 0 training_sequence = 0 inbyte = 10011100 outbyte = 00101110
190 k_code = 0 training_sequence = 0 inbyte = 10011100 outbyte = 01111011

De-scrambler – In the receiver side the function is applied with the scrambled output as a input to *inbyte* .In the similar way the descrmbled *outbyte* list is shown here. The output here is the input of the scrambler.That means data is transmitted and received with proper order.

Simulation results:

0 k_code = 0 training_sequence = 0 inbyte = 00000000 outbyte = 00000000
20 k_code = 0 training_sequence = 1 inbyte =10111100 outbyte = 10111100
30 k_code = 0 training_sequence = 0 inbyte =01000010 outbyte = 10111101
40 k_code = 0 training_sequence = 0 inbyte =01001000 outbyte = 01011111
50 k_code = 0 training_sequence = 0 inbyte =01011101 outbyte = 10011101
60 k_code = 0 training_sequence = 0 inbyte =00011100 outbyte = 00011100
70 k_code = 0 training_sequence = 0 inbyte =11111000 outbyte = 11101100
80 k_code = 1 training_sequence = 0 inbyte =01011100

outbyte = 01011100
# 90 k_code = 0 training_sequence = 0 inbyte =11100111 outbyte = 01010101
# 100 k_code = 1 training_sequence = 0 inbyte = 00011100 outbyte = 00011100
# 110 k_code = 0 training_sequence = 0 inbyte = 10110010 outbyte = 01010101
# 120 k_code = 0 training_sequence = 0 inbyte = 10001100 outbyte = 10001110
# 130 k_code = 0 training_sequence = 0 inbyte = 10111100 outbyte = 10111100
# 140 k_code = 0 training_sequence = 0 inbyte = 01100011 outbyte = 10011100
# 150 k_code = 0 training_sequence = 0 inbyte = 01100011 outbyte = 01110100
# 160 k_code = 0 training_sequence = 0 inbyte = 01100011 outbyte = 10100011
# 170 k_code = 0 training_sequence = 0 inbyte = 01100011 outbyte = 01110111
# 180 k_code = 0 training_sequence = 0 inbyte = 01100011 outbyte = 11010001
# 190 k_code = 0 training_sequence = 0 inbyte = 01100011 outbyte = 10000100

4 Conclution And Future Scope

Scrambling has no overhead, unlike block codes which always add overhead bits to the data. In the higher data rates, 8 GT/s or higher overhead is minimized using scrambling in the data stream. The 23-bit LFSR used for higher data rates reduced the overhead to approximately 1.54%. This implementation of the verilog modeling of the scrambler requires fewer than 25 XOR gates. Scrambling does not guarantee that a long run length of 0's or 1's does not occur but the probability of such a pattern occurring is astronomically low and can be ignored if the probability is well below the specified BER (BER = 10^{-12}) of the system. For Generation 4 data rate multiplicative scrambler (23bit or higher) can be implemented in the protocol.

Acknowledgment

The authors wish to thank Fastrack Anontex Ltd. This work was inspired from a Project work.

References

[1] G.M. Bhat, M. Mustafa, Shabir Ahmad and Javaid Ahmad "VHDL modeling and simulation of data scrambler and descrambler for secure data communication" , Indian Journal of Science and Technology , Vol.2 No. 10 (Oct 2009) ISSN: 0974- 6846
[2] Bhat, G. M and Ahmad W. "Reliable and Secure Data Transmission". Electronics Engineering, Vol. 68, No. 832, pp. 32-34, April 1996, London, U. K.
[3] Standaert, F. Piret, G. Rouvroy, G and Quisquater J.J, "FPGA Implementations of the ICEBERG Block Cipher", in the proceedings of ITCC 2005, vol 1, pp 556561,Las Vegas, Nevada, April 2005
[4] Wasim Ahmad (2001- 2002) Development of low-cost secure communication techniques. AICTE (R&D) Project. Deptt. of Electronics Engg., AMU, Aligarh
[5] PCI Express Base Specification Revision 3.0
[6] Based on the "interference" entry of The Concise Oxford English Dictionary, 11th edition, online
[7] Sue, M.K... "Radio frequency interference at the geostationary orbit". *NASA*. Jet Propulsion Laboratory. Retrieved 6 October 2011
[8] "PCI Express 3.0 Frequently Asked Questions" PCI-SIG. Retrieved 23 November 2010
[9] David Robert Stauffer "High Speed Serdes Devices and Applications"
[10] MUNIR A. AL-ABSI "Flexible Digital Scrambler/De-Scrambler System", Proceedings of the 10th WSEAS International Conference on CIRCUITS, Vouliagmeni, Athens, Greece, July 10-12, 2006 (pp17-21)
[11] K.Sam Shanmugam "Digital and analogue communication" John Wily Sons Inc, Canada 1979
[12] Pere Daielson "A variable-length shift Register" IEEE Trans on Computers, Vol. C-32 No 11. November 1982