International Journal
of Applied Information Technology

http://journals.telkomuniversity.ac.id/ijait

Scrambling and De-Scrambling Implementation Using Linear Feedback Shift Register Method on FPGA

Manda Lurina [a], Sugondo Hadiyoso [b, *], Rina Pudji Astuti [c]

[a,c] *School of Electrical Engineering, Telkom University, Indonesia*
[b] *School of Applied Science, Telkom University, Indonesia*

ARTICLE INFO

Received June 7, 2017
Revised July 25 2017
Accepted August 9, 2017
Available online August 14, 2017

Keywords
Scrambling, De-scrambling, LFSR, PRBS

ABSTRACT

Digital broadband communications require a fast, functional and efficient system. In a digital communication system, a long sequence of bits '0' or '1' will inherits the loss of bit synchronization, and hence it can cause the false detection on the receiver. To avoid this, long sequence of bits will be randomized first so that long sequence of bits '0' or '1' can be removed. This randomization process is called scrambling and the circuit that works for the process is a scrambler. In the receiver there is a descrambler that serves to return the bits to their original information. This paper presents a design of scrambler and descrambler using a combination of Linear Feedback Shift Register (LFSR) with 15 registers, XOR logic gates, and Pseudo Random Binary Sequence (PRBS) generator structure with polynomial $1 + x^{14} + x^{15}$. One of the two main parts of LFSR is the shift register while the other is the feedback. In LFSR, the bits contained within the selected position in the shift register will be combined in a function and the result will be put back into this register's input bit. Feedback also makes the system more stable and no error occurrence. Then special tap is taken from a certain point in XOR and returned as a feedback register. The system is implemented on FPGA board Altera De0-Nano EP4CE22F17C6 Cyclone IV E. Resource memory required <1% of available memory. Bit rate that can be achieved with clock speed 50MHz is 335570.47 bps.

* Corresponding author at:
School of Applied Science, Telkom University,
Jl. Telekomunikasi No. 1, Terusan Buah Batu, Bandung, 40257
Indonesia.
E-mail address: sugondo@tass.telkomuniversity.ac.id

ORCID ID:
Second Author: 0000-0002-2086-2156

https://doi.org/10.25124/ijait.v1i02.876
Paper registration number IJAIT000010208 2017

1. Introduction

Digital communication has grown so rapidly in line with the growth and development of the era of computerization and telecommunications. To facilitate the integration of a reliable telecommunication system, digital communication has many advantages over analog communications such as signal reinvention much easier, more resistant to distortion and interference and minimum error rate [1]. In broadband communications with high speed greater than 256Kbps [2], error rate becomes the main concentration to ensure reliability and communication quality. It requires careful synchronization between the sender and receiver so that data can be represented correctly without any errors. On the theory of data [3] in high speed data communications has the problem of bit loss synchronization resulting from long sequences of '0' or '1' bits, it can make errors data representation on the receiver. Beside using channel coding techniques, to solve this problem, long sequence of bits will be randomized first so that long sequence of bits ‘0’ or ‘1’ can be removed. This randomization process is called scrambling and the circuit that works for the process is a scrambler. In the receiver there is a descrambler that serves to return the bits to their original information.

Scrambling techniques are widely used for information and data security, as in research conducted by Niu Jiping. The paper discusses an AES-based digital data scrambling and recovery scheme [4]. Another related research is image encryption based on the Linear Feedback Shift Register method [5] that successfully implements complex methods for the security of a digital image.

Scrambling techniques on digital communications systems are also widely applied to information security, error handling and provide solutions to the problem of bit synchronization due to long bits of '0' or '1' in communication with high data rate. The scrambling method that can be used is the Linear Feedback Shift Register (LFSR). This method offers computation efficiency and high performance. Some researchers have simulated this method on communication systems. Research by Dhiraj, designed a scrambling and de-scrambling system on ALTERA CYCLONE II FPGA [6]. The use of LSFR in image encryption is also used by Allawi [7]. In his research new LFSR method of color image encryption to reorder position of the image pixels. The use of scrambling to suppress bit error was presented in the study [8] by Javaid who successfully simulated the scrambling method to increase BER.

From the literature review that has been described, a data scrambling scheme is indispensable for the reasons already mentioned above. Some simulations that have been done in previous research need to be implemented on level hardware so that it can be directly applied to communication system based on embedded application. Therefore, in this paper we present a design of scrambler and descrambler machines using 15 bit LFSR and Pseudo Random Binary Sequence (PRBS) generator structure with polynomial $1 + x^{14} + x^{15}$. The simulation results from the design, implemented on the FPGA Altera De0-Nano EP4CE22F17C6 Cyclone IV E.

The main contents of this paper are organized as follows; Section II presents design methodology. Section III presents implementation of the system. Section IV provides the brief description of the test result then the conclusions at section V.

2. Design Methodology

The design stage is done by designing system and simulation on Matlab, input and output simulation using Microsoft Excel, create and simulate VHDL language using ModelSim, the last VHDL program is implemented on FPGA using Quartus.

Overall, the system block diagram implemented in this research is shown in Figure 1.

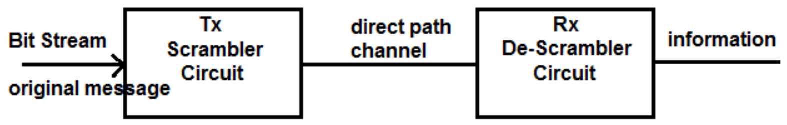


Figure 1 Design System

In the transmitter, the scrambler circuit will randomize the original information signal. In the receiver, there is a descrambler that serves to return the bits to their original information. This work using a combination of LFSR with 15 registers, XOR logic gate, and PRBS generator structure with polynomial $1 + x^{14} + x^{15}$.

2.1. Data Generator

The data generator is a circuit that generates the bit stream as a representation of the transmitted information. In this research, data generator will generate binary numbers with randomly ‘0’ and ‘1’ values.

2.2. Pseudo Random Binary Sequences (PRBS)

PRBS is a pseudo random binary sequence that performs its own repetition. The original random sequence will not repeat itself [9], but the original random sequences are difficult to generate. But PRBS with a long sequence (billions of bits) shows similarities with the original random signal, and sufficient for testing purposes. The PRBS pattern can produce all possible combinations 1 and 0 over a given time period. PRBS can be generated by sliding bits through a number (N) in the cascade register, where some of the output registers (called as tap sets) are modulo sum and feedback to the input of the first register.

2.3. Linear Feedback Shift Register (LFSR)

PRBS is implemented using LFSR or Linear Feedback Shift Register. LFSR has two main parts, namely shift register and feedback action [10]. The function of shift register is to shift the contents of the register to the adjacent position in the register. If the bit position is at the end of the register, then the contents will exit the register. The position of the bits on the other end will be left empty unless a new input enters the register. The linear function of the single bits is only XOR and inverse-XOR, thus, LFSR is the shift register whose input bit is generated from the XOR process some bits of the overall shift register value.

The initial value of LFSR is called seed [9]. The LFSR operation is deterministic, the stream of values generated by the register is actually determined from the previous state. Since the register has a limited number of state possibilities, there will be repeated cycles. LFSR with well-chosen feedback function generates a series of bits and has a long cycle. Diagram block of LFSR operation consist of D-Flip Flop and XOR gate as shown in Figure 2.

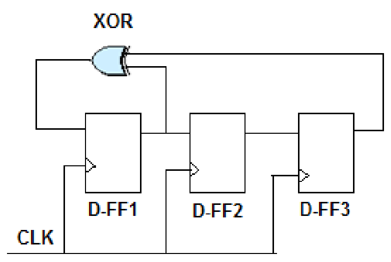


Figure 2 LFSR Operation

3. Implementation

3.1. Data Generator

The generator data will produce bit data as a representation of the source information. This data generator is actually pseudo random because the resulting random bit is the result of computing two variables. Here is the data generator block on the RTL level design shown in Figure 3.

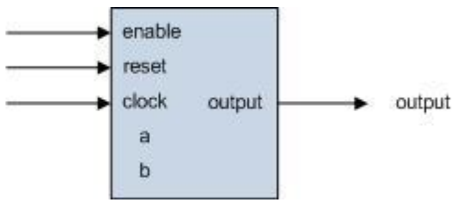


Figure 3 RTL Design of Data Generator

Two variables are namely a and b, each variable consists of 75 bits with a randomly generated pattern as follows.

```
a      =     1100110110011100101010100001111010100010111111001010101011
              11110101010101010111
b      =     110100101010010101011110000010101000101010101100000111101
              010101000011111011.
```

The random data pattern that implemented in this work follows the following explanation bellow.

1. In the first 75 bits (1-75), the generator output is same with value of a.
2. In the second 75 bits (76-150), the generator output is the sum of a and b, where the carry of the sum is not included as the output of the generator.
3. In the third 75 bits (151-225), the generator output is result of sum in the second 75 bit with variable b, where the carry of the sum is not included as the generator output.
4. In the fourth 75 bits (226-300), the generator output is result of sum in the third 75 bit with variable b, where the carry of the sum is not included as the generator output.

If procedure above applied continuously, the data generator will generate infinite and different bits. Output sample of data generator can be seen on Figure 4.

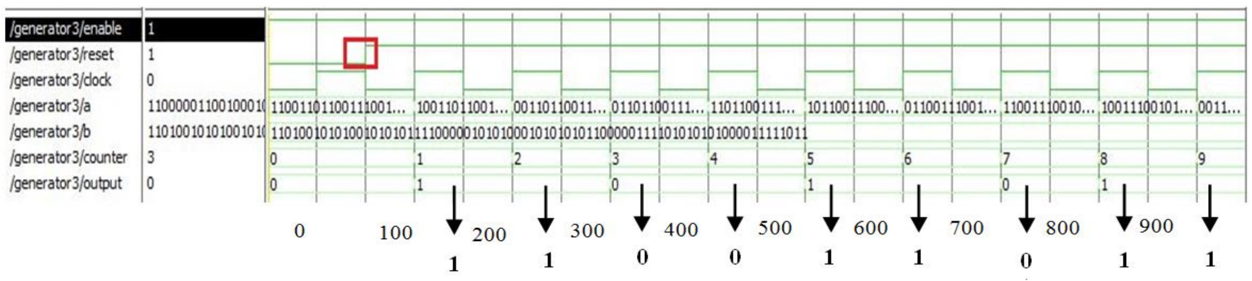


Figure 4 Output Sample of Data Generator

3.2. Scrambler and De-Scrambler

The design and implementation of scramblers and de-scramblers is the main work of this research. The design of this scrambler and descrambler uses PRBS technique with 15 bit LFSR. So the polynomial $1 + x^{14} + x^{15}$ is obtained which means to obtain the scrambler output need summing operation of modulo-2 (XOR) between the input, the contents of register 14, and the contents of register 15. The same process is also done to obtain descrambler output. Since 15 bits of LFSR are used so that the number of states can be determined by calculation where n is the number of LFSR bits used:

$$Number\ of\ states = 2^n - 1 = 2^{15} - 1 = 32768 - 1 = 32767$$

Based on the characteristics of pseudo random, in each state the composition between 0 and 1 should be appropriate. Since 15 bits of LFSR are used, the comparison between 0 and 1 must be 7 bits 0 and 8 bits 1 or 7 bits 1 and 8 bit 0, besides from these two conditions the state can't use 15 bits LFSR. Here is a sample state of 15 bit LFSR with pseudo random properties:

```
000000011111111 (7 bit with value '0' and 8 bit with value '1')
010101010101010 (8 bit with value '0' and 7 bit with value '1')
000100110100111 (8 bit with value '0' and 7 bit with value '1')
110010111000011 (7 bit with value '0' and 8 bit with value '1')
100000110101111 (7 bit with value '0' and 8 bit with value '1')
```

One of the two main parts of LFSR is the shift register (the other is feedback). Shift register is a device that serves to shift a contents of the register. When it reaches final condition, it will cross the previous series. The contents of a shift register are usually binary numbers, one and zero. If the contents of a shift register are 1101, a shift (in this case the shift is done to the right) will produce a new register content of 0110; then will generate 0011. After a continuous shift the shift register will generate 0000. In LFSR, the bits contained in the selected position in the shift register will be combined in a function and the result will be re-entered to this register's input bit. Feedback also makes a system more stable and error free. A special tap is taken from a certain point then in XOR and then returned as a feedback register.

Figure 5 shows the scrambler block, when the first input goes into the scrambler the register condition is 100101010000000 (initial state). Then the next step is sum of modulo-2 (XOR) between the contents of register 14 and the contents of register 15, the result will be entered into the register and become the contents of the register 1. The bits that previously exist in the register 1 will shift to register 2. The next process is the sum of modulo-2 XOR) between the inputs, the contents of register 14, and the contents of the register 15 where the sum is the scrambler output. The scrambler output will be the descrambler input.

In the descrambler block (see Figure 6), the circuit will reorder the previously randomized data in the scrambler to allow the data to be received and read by the receiver. Data that has been through the direct transmission channel will go into the shift register at the receiver. The register condition is 100101010000000 (initial state). Then the summation of modulo-2 (XOR) between the contents of register 14 and the contents of register 15, the result will be entered into the register and become the contents of the register 1. The bits that previously exist in the register 1 will shift to register 2. The next process is summation modulo-2 XOR) between the inputs, the contents of register 14, and the contents of the register 15 where the sum is the descrambler output. The descrambler output should be the same as the scrambler input.

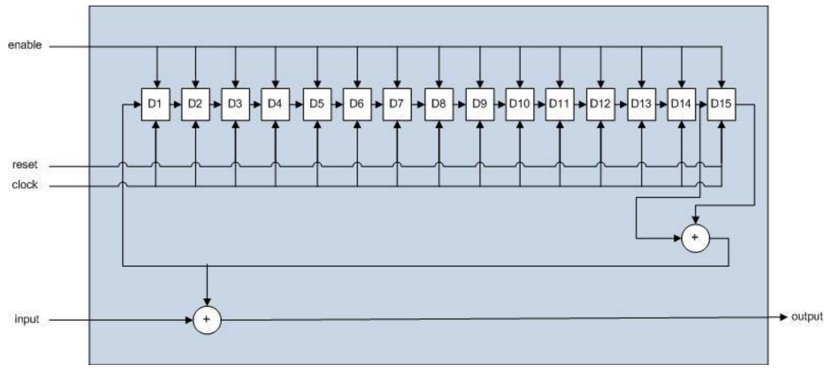


Figure 5 Scrambler Diagram

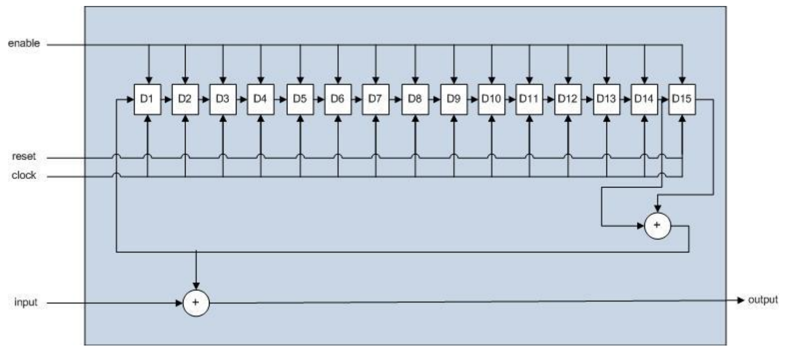
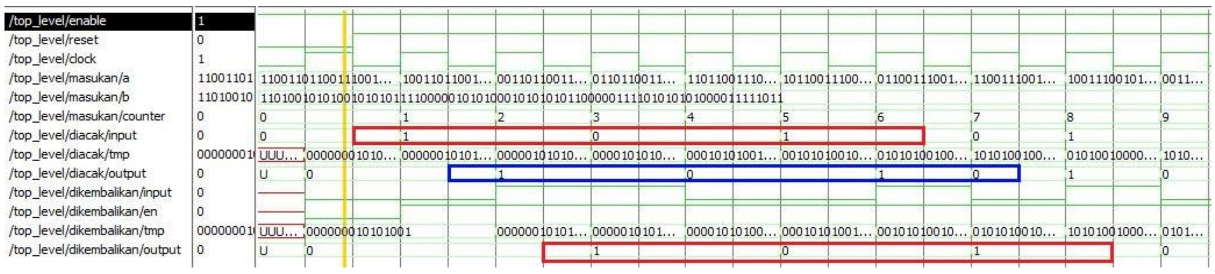


Figure 6 De-scrambler Diagram

The simulation results in ModeSim for scrambler and de-scrambler are shown in Figure 7.



```
/top_level/enable                1
/top_level/reset                 0
/top_level/clock                 1
/top_level/masukan/a             11001101
/top_level/masukan/b             11010010
/top_level/masukan/counter       0
/top_level/diacak/input          0
/top_level/diacak/tmp            00000001
/top_level/diacak/output         0
/top_level/dikembalikan/input    0
/top_level/dikembalikan/en       0
/top_level/dikembalikan/tmp      00000001
/top_level/dikembalikan/output   0
```

Figure 7 The Results of the Scrambler and De-scrambler Simulations on ModelSim

The resulting output of scrambler is the result of XOR between the inputs, the contents of register 14, and the contents of register 15. In the simulation shows that the scrambler produces the output after a delay of 100ps from the input value into the scrambler (derived from the input generator output) and descrambler produces output after the delay 100ps of input value into descrambler (derived from scrambler output). Figure 7 shows that the process of input and output validation of the system with the input variables a and b is in accordance with the theory of scrambler and descrambler where the output descrambler has the same value as the scrambler input.

4. Result

The design and simulation results on ModelSim are tested on FPGA hardware level to ensure the system design works well. The tools used to view the output are Logic Analyzer (LA) connected to the output pins of the random generator, scrambler and de-scrambler. In addition, the output can also be viewed through the I/O LED lights on the FPGA board. The target device used is Altera De0-Nano FPGA EP4CE22F17C6 Cyclone IV E. In this work used 3 pieces of FPGA board each as data generator, scrambler and de-scrambler. The results of the implementation are shown in Figure 8.

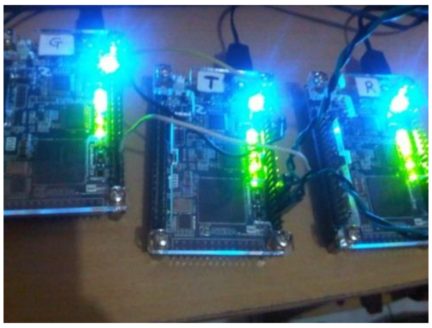


Figure 8 Implementation on FPGA

Detailed explanations of the implementation above are:

4.1. Data Generator Output

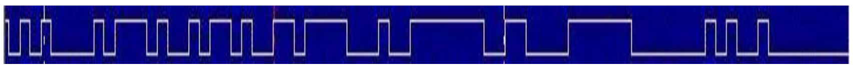


Figure 9 Testing the input generator block on logic analyzer in real-time

From the output of logic analyzer in Figure 8 can be analyzed that the data generator works. The generator generated output is a random bit stream with value is between 0 and 1.

4.2. Scrambler Output

The output generated by the random generator will be the scrambler input. From the observation result on logic analyzer Figure 10 can be analyzed that the input on the scrambler experience randomization process so that the signal form in Figure 9 and Figure 10 become different.

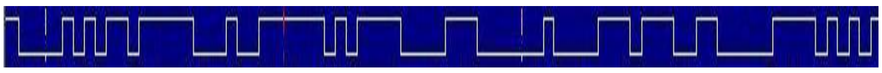


Figure 10 Testing scrambler blocks on logic analyzer in real-time

4.3. De-Scrambler Output

The resulting output of scrambler will be the descrambler input. From the observation on the logic analyzer can be analyzed that the process of returning data scrambled by scrambler can run according to given algorithm. Figure 11 shows that the output signal form descrambler has the same shape as the Figure 9 which is the output signal form generated by the generator data. The result of integration for all system blocks can be seen in Figure 12.

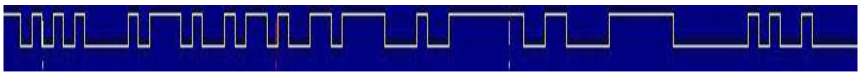


Figure 11 Testing of de-scrambler blocks on logic analyzer in real-time

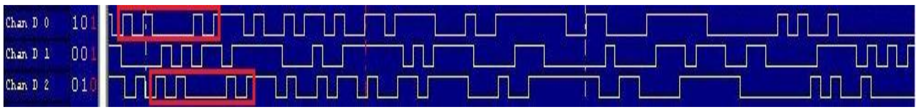


Figure 12 Results of System Integration

4.4. Memory Resource

The implementation of scrambler and de-scrambler blocks uses 99 and 101 logic elements of 22,320 logic elements (<1%) provided by the Cyclone IVE FPGA. This logic element consists of 99 combinational functions and 56 dedicated logic registers. From the results of the implementation can be concluded that the resources used in each block less than 1% so that this application can implemented on embedded application with a small memory resource.

4.5. Delay and Data Rate

Based on the implementation, the scrambler output has a delay process of 1 clock and the descrambler output also has a delay process of 1 clock. The clock frequency implemented on the FPGA is 50 MHz as the basis of bit rate calculation.

$$f_{clock} = \frac{50\ x\ 10^6}{75\ x\ 10^6} = 0{,}67\ x\ 10^6\ Hz$$

$$T_{clock} = \frac{1}{f_{clock}} = \frac{1}{0{,}67\ x\ 10^6} = 1{,}49\ x\ 10^{-6}\ s$$

$$Delay\ process = 2\ x\ T_{clock} = 2\ x\ 1{,}49\ x\ 10^{-6} = 2{,}98\ x\ 10^{-6}\ \mathrm{s}$$

$$Bit\ rate = \frac{1}{Delay\ Process} = \frac{1}{2{,}98\ x\ 10^{-6}} = 335570.47\ bps$$

From the calculation above, we can get the value of the delay process of the system is $2{,}98\ x\ 10^{-6}s$ with bit rate of 335570.7 bps.

5. Conclusion

In this study successfully designed and implemented scrambler and descrambler using combination of LFSR (Linear Feedback Shift Register) with register count of 15 registers, XOR logic gate, and PRBS (Pseudo Random Binary Sequence) and PRBS generator structure with polynomial $1 + X^{14} + X^{15}$. The FPGA resource used in each block is less than 1%. Delay process system is $2{,}98\ x\ 10^{-6}$s with bit rate was reached 335570.47 bps at clock speed 50 MHz. This system can be implemented for embedded application on communication system.

Bibliography

[1] K.M.L Sai Indrani and P. Ramesh, "A Study on the Performance of an AWGN Channel in a Communication System," *International Journal of Electronics & Communication Technology*, Vol. 4, Issue 2, pp. 89–90, June 2013.

[2] ___________," How ITU's Broadband Standards Improve Access to the Internet," [online] available at http://www.itu.int/osg/spu/ip/chapter_seven.html [accessed, June, 2017]

[3] N. Vlajic, "Digital Transmission of Digital Data: Line and Block Coding, Digital Transmission Modes," course material, 2010.

[4] N. Jiping, Z. Yongchuan, H. Zhihua， and Y. Zuqiao, "A Digital Image Scrambling Method Based on AES and Error-correcting Code," *International Conference on Computer Science and Software Engineering*., pp. 677–680, 2008.

[5] Rohith S., K. N. H. Bhat, and A. N. Sharma, "Image Encryption and Decryption using Chaotic Key Sequence Generated by Sequence of Logistic Map and Sequence of States of Linear Feedback Shift Register," *International Conference on Advance in Electronics, Computers and Communications*, 2014.

[6] D. S. Bhojane, S. S. Oak, and A. S. Joshi, "Design of Logical Scrambling and De-Scrambling System for High Speed Application," *International Conference on Computing for Sustainable Global Development*, pp. 617-622, 2016.

[7] S. T. Allawi and J. H. Al-A'meri, "Image Encryption Based on Linear Feedback Shift Register Method," *Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications*, 2016.

[8] J. A. Sheikh, Uzma, S. A. Parah, and G. M. Bhat, "Bit Error Rate (BER) Improvement of Multiple Input Multiple Output Orthogonal Frequency Division Multiplexing (MIMO-OFDM) System Using Bit Level Scrambling," *IEEE INDICON*, 2015

[9] K. Amandeep, "Linear Feedback Shift Registers in Wireless Communication Systems," India : Thapar Institute of Engineering and Technology, 2006.

[10] F. Masoodi, S. Alam, and M. U. Bhokari, "An Analysis of Linear Feedback Shift Registers in Stream Ciphers," *International Journal of Computer Applications* Volume 46, No.17, pp 46-49, May 2012 .