Innovational Stack Pointer

Ing-Heng Shih
BENQ Mobile System Inc.
No. 23 Li-Hsin Rd., Science Based Industrial Park, Hsin-Chu, Taiwan
Tel: +886-3-611-8877 ext 6587 Fax: +886-3-612-8188
E-Mail: stoneshih@benqms.com

Abstract

A method and apparatus to generate the read/write pointer of the stack or the last in first out memory (LIFO) is proposed. We use the linear feedback register (LFSR) to replace the traditional counter to generate the read/write address for the stack. Its' advantage is more efficient and costs lower circuit area overhead. Besides, due to the pseudo random sequence of the LFSR, we make encryption to the data written to the stack (PUSH) and the burst error can avoid when we read data from the stack (POP).

Introduction

The stack and the LIFO memory are widely use in VLSI design and general digital system design. The common feature of the stack and the LIFO memory is that the last written data will be first read. Therefore, the design philosophy of the address generator of the stack is simple. When the data is written, the address increases 1. On the contrary, when the data is read, the address decreases 1. Traditionally, we use the up-down counter for the implementation of the read/write address generator. But when the size of the memory is getting large, the counter will cost large area and the operation frequency will be reduced. We know that the N stage linear feedback shift register (LFSR) with primitive characteristic polynomial can generate 2^N-1 patterns (we can use one NOR-gate to generate the missing pattern). The LFSR costs the lower area overhead than counter. However, in order to replace the up-down counter, we have to use the LFSR to generate the up-down sequence. Therefore, we use the feature that the external-type LFSR with the characteristic polynomial generates the opposite state sequence to the LFSR with reciprocal polynomial to design the LFSR with up-down sequence. We can prove it in the following paragraph.

Mathematically Derivation

Figure 1 shows the general structure of the normal LFSR. The parameters C_0~C_n represent the connection between the register and external exclusive-or gate. According to the characteristic polynomial we can decide where has a connection between the register and external exclusive-or gate. If there is a connection, the parameter C_i is 1. Otherwise, the parameter C_i is 0. The parameters D_1~D_{n-1} represent the output of the register is from Q or $\overline{Q}$. If it is 0, it means the output of the register is Q. Otherwise the output of the register is $\overline{Q}$.

We assume that the value of register out put is Q_1~Q_n. Assume that C_0 and C_n are always 1. It means that there is always a connection in C_0 and C_n. We can derive the state equation over GF(2) as follows:

$(Q_1)_{i+1} = \{C_1*(Q_1)_i + C_2*(Q_2)_i + \ldots + C_{n-1}*(Q_{n-1})_i +$

$C_n*(Q_n)_i \}*C_0+D_1$

$(Q_2)_{i+1} = (Q_1)_i + D_2$

$(Q_3)_{i+1} = (Q_2)_i + D_3$

$\vdots$

$(Q_{n-1})_{i+1} = (Q_{n-2})_i + D_{n-1}$

$(Q_n)_{i+1} = (Q_{n-1})_i + D_n$

The state "*i*" means the current state, and the state "*i+1*" means the next state. We can also express the state equation in the term of the matrix form.

$$[Q_1 \quad Q_2 \quad \cdots \quad Q_n]_{i+1} = [Q_1 \quad Q_2 \quad \cdots \quad Q_n]_i * \begin{bmatrix} C_1 & 1 & 0 & \cdots & 0 \\ C_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{n-1} & 0 & 0 & \cdots & 1 \\ C_n & 0 & 0 & \cdots & 0 \end{bmatrix} + [D_1 \quad D_2 \quad \cdots \quad D_n]$$

We define the matrix $\boldsymbol{Q_{i+1}}$ as $[Q_1 \quad Q_2 \quad \cdots \quad Q_n]_{i+1}$, and the matrix $\boldsymbol{Q}_i$ as $[Q_1 \quad Q_2 \quad \cdots \quad Q_n]_i$, and the matrix $\boldsymbol{C}$ as $\begin{bmatrix} C_1 & 1 & 0 & \cdots & 0 \\ C_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{n-1} & 0 & 0 & \cdots & 1 \\ C_n & 0 & 0 & \cdots & 0 \end{bmatrix}$ and the matrix $\boldsymbol{D}$ as $[D_1 \quad D_2 \quad \cdots \quad D_n]$. Therefore, the original state function can be expressed as $\boldsymbol{Q}_{i+1} = \boldsymbol{Q}_i * \boldsymbol{C} + \boldsymbol{D}$. (1.1)

As shown in Figure 2, the structure of the LFSR with reciprocal polynomial (inverse-type LFSR) has inverse parameters $C_{n-1} \sim C_0$ and $D_1 \sim D_{n-1}$ to the normal type. We define the register outputs as $Q'_1 \sim Q'_n$. Then, the state equation of inverse-type LFSR can be derived with the same method in the following:

$(Q'_1)_{i+1} = (Q'_2)_i + D_2$

$(Q'_2)_{i+1} = (Q'_3)_i + D_3$

$(Q'_3)_{i+1} = (Q'_4)_i + D_4$

$\vdots$

$(Q'_{n-1})_{i+1} = (Q'_n)_i + D_n$

$(Q'_n)_{i+1} = \{ C_n * [(Q'_1)_i + D_1] + C_1 * [(Q'_2)_i + D_2] + C_2 * [(Q'_3)_i + D_3] + \ldots + C_{n-2} * [(Q'_{n-1})_i + D_{n-1}] + C_{n-1} * [(Q'_n)_i + D_n] \}$

$= \{ C_n * (Q'_1)_i + C_1 * (Q'_2)_i + C_2 * (Q'_3)_i + \ldots + C_{n-2} * (Q'_{n-1})_i + C_{n-1} * (Q'_n)_i \} + \{ C_n * D_1 + C_1 * D_2 + \ldots + C_{n-2} * D_{n-1} + C_{n-1} * D_n \}$

We can express the state equation in the term of the matrix form in the following:

$$[Q'_1 \quad Q'_2 \quad \cdots \quad Q'_n]_{i+1} = [Q'_1 \quad Q'_2 \quad \cdots \quad Q'_n]_i * \begin{bmatrix} 0 & 0 & \cdots & 0 & C_n \\ 1 & 0 & \cdots & 0 & C_1 \\ 0 & 1 & \cdots & 0 & C_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & C_{n-1} \end{bmatrix} + [D_2 \quad D_3 \quad \cdots \quad D_n \quad (C_n * D_1 + C_1 * D_2 + \ldots + C_{n-1} * D_n)]$$

We define the matrix $\boldsymbol{Q'_{i+1}}$ as $[Q'_1 \quad Q'_2 \quad \cdots \quad Q'_n]_{i+1}$, and the matrix $\boldsymbol{Q'_i}$ as $[Q'_1 \quad Q'_2 \quad \cdots \quad Q'_n]_i$, and the matrix $\boldsymbol{C'}$ as $\begin{bmatrix} 0 & 0 & \cdots & 0 & C_n \\ 1 & 0 & \cdots & 0 & C_1 \\ 0 & 1 & \cdots & 0 & C_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & C_{n-1} \end{bmatrix}$, and the matrix $\boldsymbol{D'}$ as $[D_2 \quad D_3 \quad \cdots \quad D_n \quad (C_n * D_1 + C_1 * D_2 + \ldots + C_{n-1} * D_n)]$. Therefore, the original state function can be expressed as $\boldsymbol{Q'_{i+1}} = \boldsymbol{Q'_i} * \boldsymbol{C'} + \boldsymbol{D'}$ (1.2)

According to the equation 1.1 and 1.2, we consider the condition when state $\boldsymbol{Q_i} = \boldsymbol{Q'_i}$:

$$\boldsymbol{Q'_{i+1}} = \boldsymbol{Q'_i} * \boldsymbol{C'} + \boldsymbol{D'} = (\boldsymbol{Q_{i-1}} * \boldsymbol{C} + \boldsymbol{D}) * \boldsymbol{C'} + \boldsymbol{D'} = \boldsymbol{Q_{i-1}} * \boldsymbol{C} * \boldsymbol{C'} + \boldsymbol{D} * \boldsymbol{C'} + \boldsymbol{D'} \qquad (1.3)$$

We calculate the matrix multiplication of $\boldsymbol{C} * \boldsymbol{C'}$ and $\boldsymbol{D} * \boldsymbol{C'} + \boldsymbol{D'}$**:**

$\boldsymbol{C} * \boldsymbol{C'} =$

$$\begin{bmatrix} C_1 & 1 & 0 & \cdots & 0 \\ C_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{n-1} & 0 & 0 & \cdots & 1 \\ C_n & 0 & 0 & \cdots & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & \cdots & 0 & C_n \\ 1 & 0 & \cdots & 0 & C_1 \\ 0 & 1 & \cdots & 0 & C_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & C_{n-1} \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & C_1 * (C_n + 1) \\ 0 & 1 & \cdots & 0 & C_2 * (C_n + 1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & C_{n-1} * (C_n + 1) \\ 0 & 0 & \cdots & 0 & C_n * C_n \end{bmatrix}$$

In our assumption, C_n is always 1. Therefore, $\boldsymbol{C} * \boldsymbol{C'} = \boldsymbol{I}$.

$$\boldsymbol{D} * \boldsymbol{C'} = \begin{bmatrix} D_1 & D_2 & \cdots & D_n \end{bmatrix} \begin{bmatrix} 0 & 0 & \cdots & 0 & C_n \\ 1 & 0 & \cdots & 0 & C_1 \\ 0 & 1 & \cdots & 0 & C_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & C_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} D_2 & D_3 & \cdots & D_n & (C_n * D_1 + C_1 * D_2 + \ldots + C_{n-1} * D_n) \end{bmatrix}$$
$$= \boldsymbol{D'}$$

Therefore, $\boldsymbol{D} * \boldsymbol{C'} + \boldsymbol{D'} = \boldsymbol{D'} + \boldsymbol{D'} = \boldsymbol{0}$ in GF(2). According to this calculation, we can reduce the equation 1.3:

$$\boldsymbol{Q'_{i+1}} = \boldsymbol{Q_{i-1}} * \boldsymbol{C} * \boldsymbol{C'} + \boldsymbol{D} * \boldsymbol{C'} + \boldsymbol{D'} = \boldsymbol{Q_{i-1}} * \boldsymbol{I} + \boldsymbol{0} = \boldsymbol{Q_{i-1}} \qquad (1.31)$$

Equation 1.31 implies that when there is a certain state makes $\boldsymbol{Q'_i}$ equal to $\boldsymbol{Q_i}$, the next state of $\boldsymbol{Q'_i}$ will be equal to the previous state of $\boldsymbol{Q_i}$.

Circuit Design

Figure 3 shows the block diagram of the general stack. The function of the stack is very simple. It only contains two main functional blocks. One is memory module and the other is address generator. The memory module can be asynchronous or synchronous. We illustrate the asynchronous memory in this example. The address generator produces the address for the memory module according the read/write enable signal.

The primary input of the stack is the data bus "data_in" and read/write enable. Maybe the read/write enable signal is decoded from "**PUSH**" or "**POP**" instructions. The users needn't care which address they write or read. The operation that users want is to give "**PUSH**" instruction and the data will be written into the memory. In the same way, they give "**POP**" instruction to read the latest written data. Therefore, the address generator is also responsible for address maintenance.

Figure 4 shows the general read/write operation of the stack. 4(a) shows the write operation of the stack. When the "data 5" is written into the stack, it will be stored in the "address 5" of the physical memory. 4(b) shows the read operation. After the "data 5" is read from the stack, the pointer will decrease 1 and point to the "address 4". Therefore, in 4(c), if we want to read stack again, the "data 4" store in the "address 4" will be read.

If we use LFSR to be the address generator, the address will not be in order. Figure 5 shows the operation of the stack with the new address generator. Although the sequence of address is not in order, it still can give up (normal) and down (reverse) sequence. The address generator is illustrated in Figure 6. The 4-stage LFSR is used to generate 16 addresses in this example. The characteristic polynomial is X^4+X+1 and the reciprocal polynomial is X^4+X^3+1.

In Figure 6, the example of up-down address generator is proposed. We know that N-stage LFSR can generate 2^N-1 patterns. The NOR-gate is used to generate the pattern "0" we lack. Under the proposed architecture, we can generate up-down sequence (not in order) we want from the register outputs.

There are several reversible components in this architecture. The detailed circuit of the reversible register is shown in Figure 7. It uses different direction tri-state buffers to construct normal and reverse path. When the signal "*up_dn*" is 0, the tri-state buffers to the right direction are asserted. The path begins from the port *"inout_left"* to the port *"inout_right"*. On the contrary, when the signal "*up_dn*" is 1, the tri-state buffers to the left direction are asserted and the tri-state buffers to the right direction are de-asserted. The path will begin from the port *"inout_right"* to the port *"inout_left"*.

There is the other reversible component – exclusive or gate (XOR) in Figure 6. The operation principle is as the same as the reversible register. The detailed circuit is shown in Figure 8. It has to be mentioned that there is something different in the "One-way XOR". There is only one path through exclusive-or gate and another path is bypass. Figure 9 shows the "One-way XOR" on the left side in Figure 6. When the signal "*up_dn*" is 0, the path will be through the exclusive-or gate. In Figure 10, if the path wants to go through the exclusive-or gate, the signal "*up_dn*" should be 1.

Therefore, we can consider the circuit of the up-down sequence LFSR in Figure 6 as two different circuit configurations under the different value of the signal *"up_dn"*. When the signal *"up_dn"* is 0, we can consider it as the circuit in Figure 11. It is the normal LFSR with the characteristic polynomial X^4+X+1. If the signal *"up_dn"* is 1, it can consider as the LFSR with the characteristic polynomial X^4+X^3+1. But as shown in Figure 12, the definition of the LSB and MSB of the address is opposite to Figure 11. According to the mathematical proven previous paragraph, we know that these two circuits can generate contrary address sequence to each other. The simulation result is shown in Figure 13. When the up-down signal changes, the address sequence becomes reversed to the original.

Conclusion

This paper proposes a method that can generate read/write address for the stack and LIFO memory by use the linear feedback shift register. This circuit costs lower area overhead and can make an encryption to the data stored in the memory.

Reference:

[1] M. Nicolaidis.V. C. Alves. "Trade-Offs In Scan Path and BIST Implementations for RAMs", *European Test Conference*, page 169-178, 1993.

[2] Miron Abramovici, Melvin A. Breuer, Arthur D. Friendman. "Digital System Testing and Testable Design", *IEEE PRESS.*

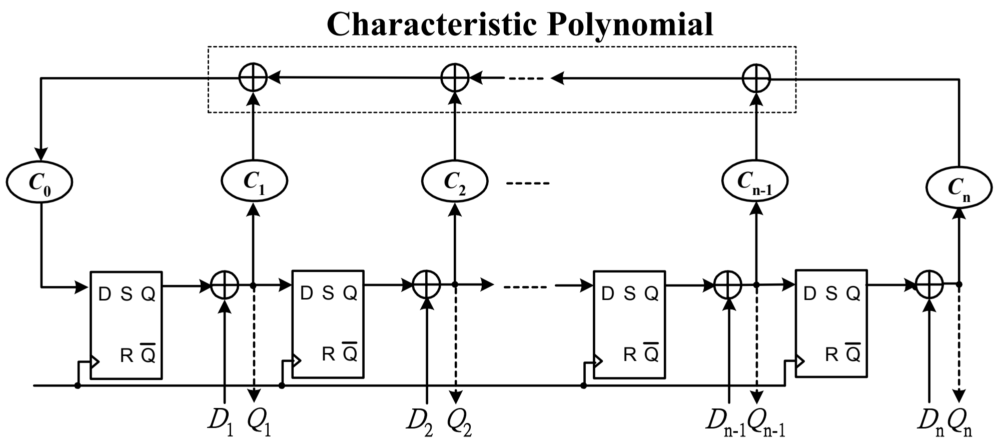

Figure 1. Scheme of the Mixed Type Linear Feedback Shift Register (MFSR)

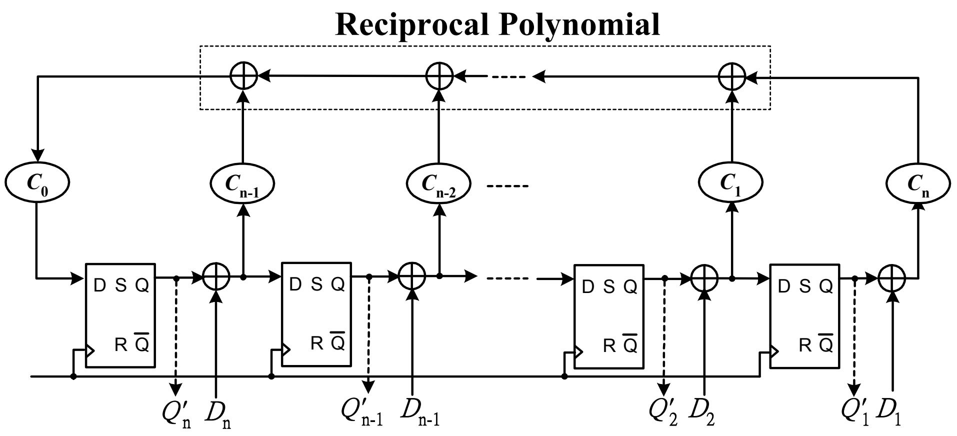

Figure 2. Scheme of the MFSR with the Reciprocal Polynomial

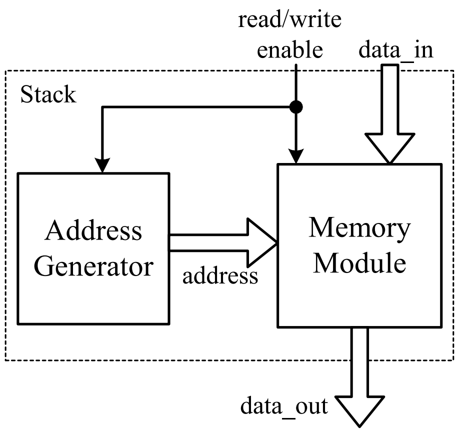

Figure 3. Functional Block Diagram of the Stack

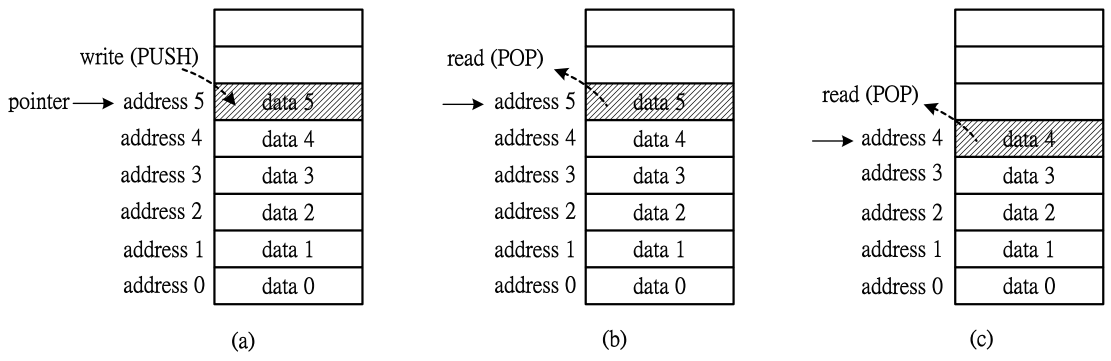

Figure 4. The illustration of the general read/write operation of the stack: (a) write one data to the stack (PUSH), (b) read one data from the stack (POP), (c) after (b), continue to read data for the stack (POP).

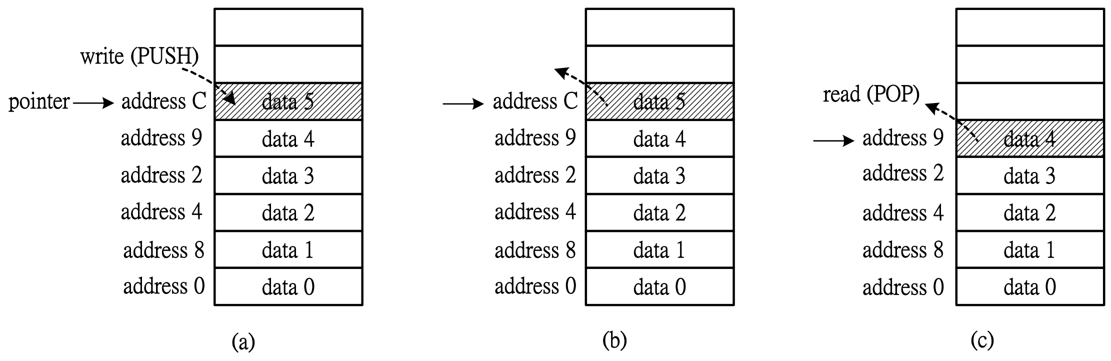

Figure 5. The illustration of the modified read/write operation of the stack: (a) write one data to the stack (PUSH), (b) read one data from the stack (POP), (c) after (b), continue to read data for the stack (POP).

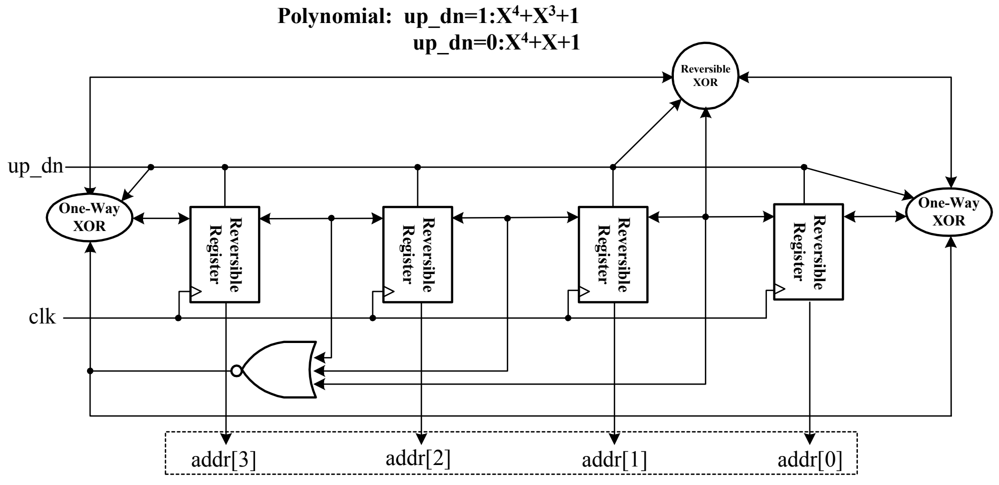

Figure 6. The example of up-down sequence LFSR

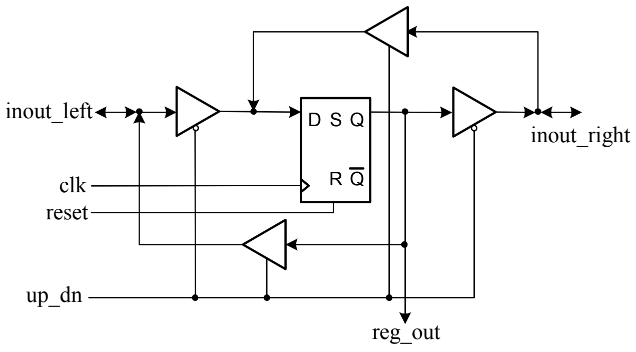

Figure 7. The illustration of the reversible register

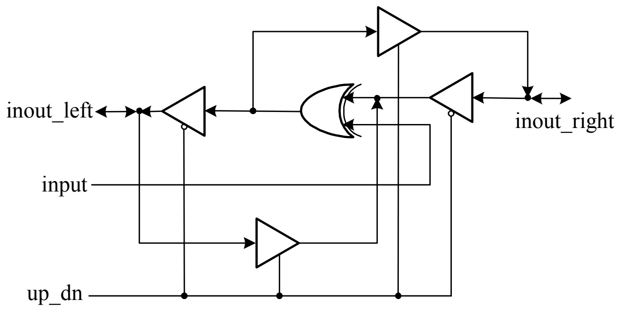

Figure 8. The illustration of the reversible XOR

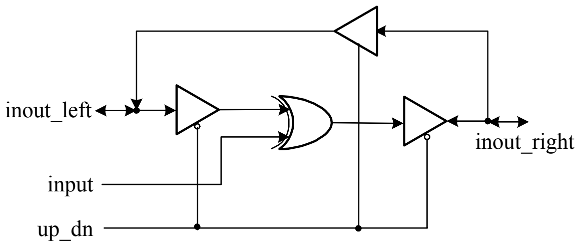

Figure 9. The illustration of the One-way XOR on the left side in Fig 6

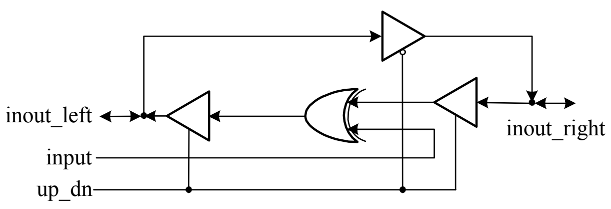

Figure 10. The illustration of the One-way XOR on the right side in Fig. 6

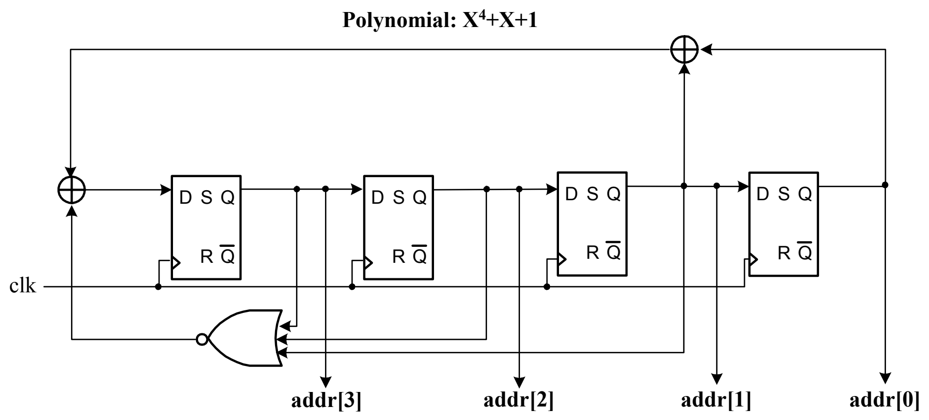

Figure 11. The architecture of the LFSR when up_dn = 0.

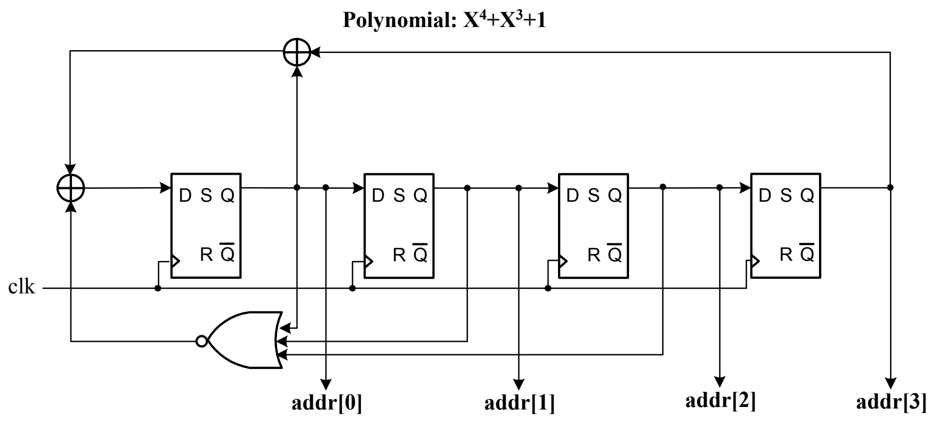

Figure 12. The architecture of the LFSR when up_dn = 1

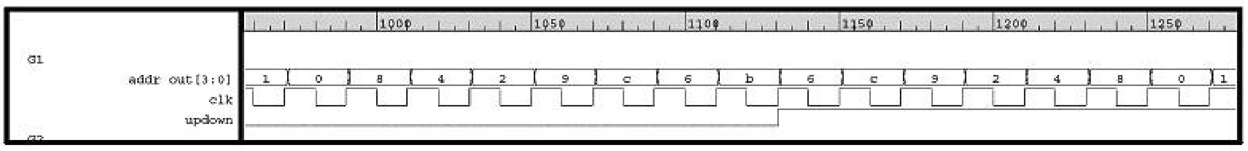

Figure 13. The simulation result of the example in Fig. 6