A Complete Memory Address Generator for Scan Based March Algorithms

Wei-Lun Wang[1] and Kuen-Jong Lee[2]
[1]*Department of Electronic Engineering, Cheng Shiu University*
Kaohsiung, Taiwan, ROC
E-mail: wlwang@csu.edu.tw
[2]*Department of Electrical Engineering, National Cheng Kung University*
Tainan, Taiwan, ROC
E-mail: kjlee@mail.ncku.edu.tw

Abstract

The march algorithm based built-in self-test (BIST) schemes have been widely used to test memory chips (cores). Conventional methods which use binary counters to generate the addresses may require large routing area when the addresses are to be broadcast to multiple memory cores. In this paper we propose to use linear feedback shift registers (LFSRs) to generate the memory addresses which can be serially applied to the memory cores under test and thus the routing area overhead can be greatly reduced. We have designed a complete up/down LFSR which can generate complete march addresses, including all 2^n up and 2^n down sequences. Also theoretic analysis has been done which guarantees the transitions from up to down and down to up sequences can all be smoothly carried out such that the memory under test can receive a different address per clock cycle even during the transitions.

1. Introduction

With the advance of deep submicron manufacturing technology and system-on-a-chip (SOC) design methodology, a large number of cores, especially the memory cores, are now integrated into a single chip. It has been predicted that by the year of 2014, memory cores may occupy 94% area of a typical SOC [1]. Embedded memory cores are thus essential to the performance of an SOC. However due to the limit of the I/O pins, it is difficult to test embedded memory cores externally. Thus the built-in self-test (BIST) schemes have been widely used to solve this problem [2].

Due to the high test efficiency and low implementation complexity, the march algorithms have been widely used in the BIST design for memory cores. A general testing scheme for testing the memory is shown in Fig. 1. In a conventional BIST environment, the address bus, data bus, and read/write control signals generated from the test pattern generator of the BIST are applied to the memory under test in parallel. In case where a large number of memory cores are to be tested in parallel, it is often required to share a BIST controller so as to reduce the area overhead. The routing area for the address bus, data bus, and read/write control signals from the BIST controller to all the memory cores thus may occupy a large area and hence should be carefully dealt with if parallel testing with minimized resources is targeted.

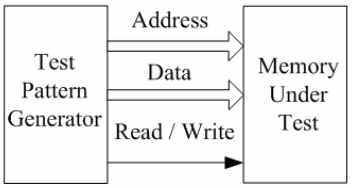


Fig. 1 A general testing for the memory.

In [3], a serial interfacing method for embedded-memory testing is proposed which allows only one line to provide the test data to the memory cores and hence can greatly reduce the required data bus area. However in [3] only the data line problem is considered while the address lines which may also occupy a large area is not attacked. In this paper we will focus on the reduction of the routing area of the address bus. A similar work has been done in [4] where using an LFSR to generate both the "down" and "up" sequences has been proposed. However in this work only the separate up or down sequence are generated but how to make smooth transitions from the up to down sequences and from the down to up sequences, which are commonly used in most march algorithms, are not

Proceedings of the 2005 IEEE International Workshop on Memory Technology, Design, and Testing (MTDT'05)
1087-4852/05 $20.00 © 2005 **IEEE**

addressed. As a result, the memory addressing must be paused by a control circuitry for several clock cycles during the transitions, which can cause some delay on the testing as well as requires extra hardware control. To solve this problem, in this paper we develop a complete up/down LFSR which not only can generate complete march addresses, including all the 2^n up and 2^n down sequences, but also guarantee that all the transitions (up-up, up-down, down-up, and down-down) can be smoothly carried out such that the memory under test can receive a different address per clock cycle even during the transitions. We also provide extensive theoretic analysis to validate our design.

2. The complete up/down LFSR

Due to the regularity of structure and lower hardware overhead, the linear feedback shift registers (LFSRs) and its extended variation, the complete LFSR (CLFSR) [5], have been widely used in the testing of digital chips/systems [2, 6, 7]. In this paper we will use the CLFSR based memory address generator (MAG) to generate the up/down sequences and then apply them serially to the memory address register(s) (MAR) to generate the proper up/down memory addresses of the march algorithm to test the memories. Figure 2 shows the basic idea of this method, where the up and down sequences are generated from the MAG and applied to the MAR of all the memory cores to be tested through a single line. As will become clear later, the up and down sequences will be applied to the MAR from different directions. It is easy to see that if more memory cores are to be tested at the same time, the routing area that can be saved will be quite significant.

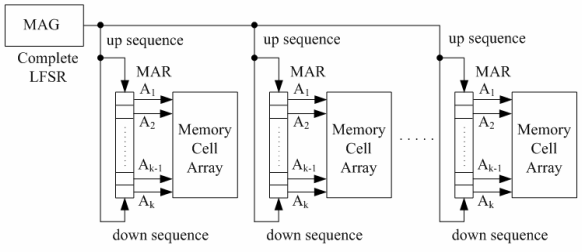


Fig. 2 Block diagram of scan based memory address generation.

We start with some basic LFSR theory to prove the correction of our design. Fig. 3 shows a *k*-stage complete LFSR with a primitive polynomial g(X) which is expressed as follows:

$$g(X) = \sum_{i=0}^{k} c_i * X^i \ , \ c_0 = c_k = 1. \qquad \textbf{(1)}$$

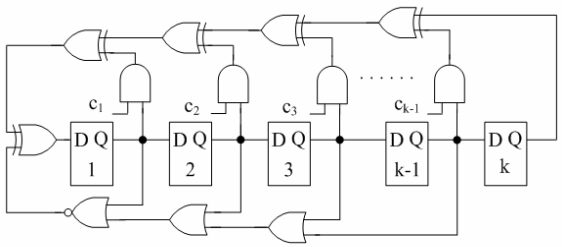


Fig. 3 The *k*-stage external type complete LFSR with g(X).

Let the *i*-th internal state of the *k*-stage CLFSR be S_i ($0 \le i \le (2^k - 1)$) which is expressed as:

$$S_i = (q_1, q_2, q_3, \ldots, q_{k-1}, q_k)_i \qquad \textbf{(2)}$$

The *k*k* characteristic matrix *G* is expressed as:

$$G = \begin{pmatrix} c_1 & 1 & 0 & \ldots & 0 \\ c_2 & 0 & 1 & \ldots & 0 \\ : & : & : & & : \\ c_{k-1} & 0 & 0 & \ldots & 1 \\ c_k & 0 & 0 & \ldots & 0 \end{pmatrix}. \qquad \textbf{(3)}$$

The relationship between the *i*-th state S_i and the *j*-th state S_j in Fig. 3 is:

$$S_i = S_j * G^{i-j} + T_j, \ i \ge j. \qquad \textbf{(4)}$$

where $T_j = (a, 0, 0, \ldots, 0)$ and

$$a = \begin{cases} 1, & \text{if } S_j = (0,\ 0,\ 0,\ ...,\ 0) \\ 0, & \text{elsewhere} \end{cases}.$$

For example, the internal states of a 4-stage CLFSR with a primitive polynomial of $g(X) = X^4 + X + 1$ are tabulated in Table 1.

It has been shown in [7] and is now well-known that if an up sequence can be generated by a CLFSR with a primitive polynomial, then the corresponding down sequence can be generated by a CLFSR with its reciprocal polynomial. The reciprocal polynomial of g(X), h(X), as defined in [7] can be expressed as follows:

$$h(X) = \sum_{i=0}^{k} c_i * X^{k-i} \ , \ c_0 = c_k = 1. \qquad \textbf{(5)}$$

The CLFSR with h(X) is shown in Fig. 4. Let the *i*-th internal state of the *k*-stage CLFSR be Y_i ($0 \le i \le (2^k - 1)$), i.e.,

$$Y_i = (q_1, q_2, q_3, \ldots, q_{k-1}, q_k)_i. \qquad \textbf{(6)}$$

The relationship between the *i*-th state Y_i and the *j*-th state Y_j in Fig. 4 is:

$$Y_i = Y_j * H^{i-j} + Z_j, \ i \ge j. \qquad \textbf{(7)}$$

where $Z_j = (0,\ 0,\ 0,\ ...,\ b)$,

$$b = \begin{cases} 1, & \text{if } Y_j = (0,\ 0,\ 0,\ ...,\ 0) \\ 0, & \text{elsewhere} \end{cases}, \text{ and}$$

$$H = \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & c_1 \\ 0 & 1 & \dots & 0 & c_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & c_{k-1} \end{pmatrix}. \quad \textbf{(8)}$$

Table 1 The internal states of CLFSR with $g(X) = X^4 + X + 1$

$S_0 = \{0, 0, 0, 0\}$	$S_8 = \{1, 0, 1, 0\}$
$S_1 = \{1, 0, 0, 0\}$	$S_9 = \{1, 1, 0, 1\}$
$S_2 = \{1, 1, 0, 0\}$	$S_{10} = \{0, 1, 1, 0\}$
$S_3 = \{1, 1, 1, 0\}$	$S_{11} = \{0, 0, 1, 1\}$
$S_4 = \{1, 1, 1, 1\}$	$S_{12} = \{1, 0, 0, 1\}$
$S_5 = \{0, 1, 1, 1\}$	$S_{13} = \{0, 1, 0, 0\}$
$S_6 = \{1, 0, 1, 1\}$	$S_{14} = \{0, 0, 1, 0\}$
$S_7 = \{0, 1, 0, 1\}$	$S_{15} = \{0, 0, 0, 1\}$

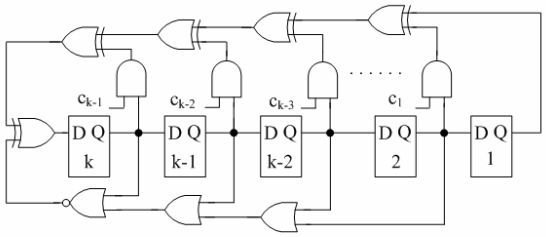


Fig. 4 The *k*-stage external type complete LFSR with h(X).

Compared to the LFSR, all the 2^k states including an all-zero state are generated by a *k*-stage CLFSR. In Fig. 3 and 4, the next states of the all-zero state are (1, 0, 0, …, 0) and (0, 0, …, 0, 1), respectively. The state transitions of the remaining non-zero states in the CLFSR are the same as those in the LFSR. Therefore, for any two non-zero states in the CLFSR with g(X) and h(X), S_i and S_j, their relationships are reduced to:

$$S_i = S_j * G^{i-j} \quad \textbf{(9)}$$

$$Y_i = Y_j * H^{i-j} \quad \textbf{(10)}$$

and the relationship between *G* and *H* is

$$G*H = \begin{pmatrix} c_1 & 1 & 0 & \dots & 0 \\ c_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ c_{k-1} & 0 & 0 & \dots & 1 \\ c_k & 0 & 0 & \dots & 0 \end{pmatrix} * \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & c_1 \\ 0 & 1 & \dots & 0 & c_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & c_{k-1} \end{pmatrix} \quad \textbf{(11)}$$

$$= \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}.$$

Therefore the *H* is the inverse of *G* ($H = G^{-1}$) and Eq. (7) can be rewritten as

$$S_i = S_j * (G^{-1})^{j-i} + Z_j, \; i < j. \quad \textbf{(12)}$$

Table 2 tabulates the internal states of the CLFSR with the reciprocal polynomial of $g(X) = X^4 + X + 1$, i.e., $h(X) = X^4 + X^3 + 1$. Compared to the state transitions in Tables 1 and 2, the up state transitions are opposite to the down state transitions.

Table 2 The internal states of CLFSR with $h(X) = X^4 + X^3 + 1$

$S_{14} = \{0, 0, 1, 0\}$	$S_6 = \{1, 0, 1, 1\}$
$S_{13} = \{0, 1, 0, 0\}$	$S_5 = \{0, 1, 1, 1\}$
$S_{12} = \{1, 0, 0, 1\}$	$S_4 = \{1, 1, 1, 1\}$
$S_{11} = \{0, 0, 1, 1\}$	$S_3 = \{1, 1, 1, 0\}$
$S_{10} = \{0, 1, 1, 0\}$	$S_2 = \{1, 1, 0, 0\}$
$S_9 = \{1, 1, 0, 1\}$	$S_1 = \{1, 0, 0, 0\}$
$S_8 = \{1, 0, 1, 0\}$	$S_0 = \{0, 0, 0, 0\}$
$S_7 = \{0, 1, 0, 1\}$	$S_{15} = \{0, 0, 0, 1\}$

The up/down memory addresses of the march algorithms are generated in distinct testing phases. To save the hardware cost, the flip-flops of the CLFSR with a primitive polynomial and its reciprocal polynomial, g(X) and h(X), can be shared. Fig. 5 shows an integrated up/down CLFSR with $g(X) = X^4 + X + 1$ and $h(X) = X^4 + X^3 + 1$. In Figs. 3 and 4, the data flow direction from the previous stage to the next stage is opposite, therefore a multiplexer (MUX) is required for each flip-flop.

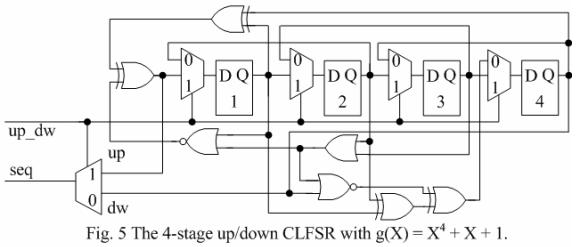
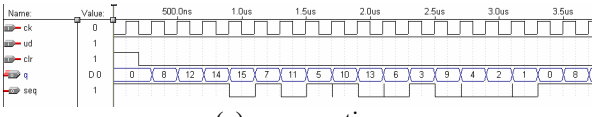
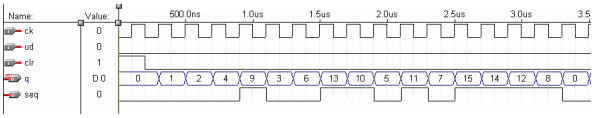


Fig. 5 The 4-stage up/down CLFSR with $g(X) = X^4 + X + 1$.

For convenience, in this paper we assume all the initial states of the CLFSRs can be set to all-zero by a reset signal (for simplicity, the reset signal is not shown in all figures). The internal state transition order of the CLFSR is going up (down) when the value of the *up_dw* controlled line is one (zero). The up/down sequence will appear serially at the output line of *seq*. To verify that the design in Fig. 5 can meet the requirements of the up/down address generation of the march algorithm, i.e., the up addresses have a reverse order of that of the down addresses, in the up (down) operation of the CLFSR, the timing diagram of the up/down CLFSR is shown in Fig. 6. The state transitions of the up CLFSR are opposite to those in the down CLFSR.



(a) up operation



(b) down operation

Fig. 6 The timing diagram of the 4-stage up/down CLFSR.

For generality, Fig. 7 shows a *k*-stage CLFSR based up/down sequence generator with a primitive polynomial of g(X). To generate the up/down memory addresses of the march algorithm, for a *k*-stage CLFSR, the up (down) sequence must be extracted from the terminal of D_1 (Q_k). This will be explained in the next section.

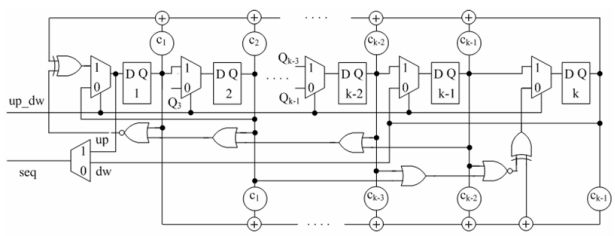


Fig. 7 The *k*-stage CLFSR based up/down sequence generator.

3. The memory address register

We have described how to generate an up/down sequence by an up/down CLFSR as the memory address generator (MAG). We now describe how to feed the up/down serial sequence into the serial memory address register (MAR) to generate the *k*-bit up/down addresses ($A_1, A_2, ..., A_{k-1}, A_k$).

Again, we will use the LFSR in Figure 5 to illustrate how to apply the up/down sequence to the memory address register (MAR). Both the internal states of the 4-bit CLFSR and the MAR are tabulated in Table 3. The symbols, O_i, S_i, U_i, and V_i (i = 0, 1, 2, …, 15), are the i-th output bit of the complete LFSR, the i-th internal state of the CLFSR, the i-th up memory address, and the i-th down memory address, respectively.

In the march algorithm, the i-th up memory address U_i is equal to the (15-i)-th down memory addresses V_{15-i} (i = 0, 1, 2, …, 15). To save the hardware overhead of the control circuitry, let the initial states of the CLFSR and MAR, $\{O_3, O_2, O_1, O_0\}$, be all-zero, thus $S_0 = U_0 = V_{15} = \{0, 0, 0, 0\} = \{O_3, O_2, O_1, O_0\}$. Since the i-th output bit O_i ($0 \le i \le 15$) is generated by the 4-stage CLFSR, thus the relationship between the i-th state S_i and O_i is expressed as:

$$O_i = RB(S_i) = RB(\{O_{(i+3) \bmod 16}, O_{(i+2) \bmod 16}, O_{(i+1) \bmod 16}, O_{i \bmod 16}\}) \quad \textbf{(13)}$$

where $RB(S_i)$ is the function to take the rightmost bit of S_i.

Table 3 The internal states of the 4-stage CLFSR and MAR

Order	CLFSR (Q_1, Q_2, Q_3, Q_4)	MAR (A_1, A_2, A_3, A_4)
Up	$S_0 = \{O_3, O_2, O_1, O_0\}$	$U_0 = \{O_3, O_2, O_1, O_0\}$
	$S_1 = \{O_4, O_3, O_2, O_1\}$	$U_1 = \{O_4, O_3, O_2, O_1\}$
	$S_2 = \{O_5, O_4, O_3, O_2\}$	$U_2 = \{O_5, O_4, O_3, O_2\}$
	$S_3 = \{O_6, O_5, O_4, O_3\}$	$U_3 = \{O_6, O_5, O_4, O_3\}$
	$S_4 = \{O_7, O_6, O_5, O_4\}$	$U_4 = \{O_7, O_6, O_5, O_4\}$
	$S_5 = \{O_8, O_7, O_6, O_5\}$	$U_5 = \{O_8, O_7, O_6, O_5\}$
	$S_6 = \{O_9, O_8, O_7, O_6\}$	$U_6 = \{O_9, O_8, O_7, O_6\}$
	$S_7 = \{O_{10}, O_9, O_8, O_7\}$	$U_7 = \{O_{10}, O_9, O_8, O_7\}$
	$S_8 = \{O_{11}, O_{10}, O_9, O_8\}$	$U_8 = \{O_{11}, O_{10}, O_9, O_8\}$
	$S_9 = \{O_{12}, O_{11}, O_{10}, O_9\}$	$U_9 = \{O_{12}, O_{11}, O_{10}, O_9\}$
	$S_{10} = \{O_{13}, O_{12}, O_{11}, O_{10}\}$	$U_{10} = \{O_{13}, O_{12}, O_{11}, O_{10}\}$
	$S_{11} = \{O_{14}, O_{13}, O_{12}, O_{11}\}$	$U_{11} = \{O_{14}, O_{13}, O_{12}, O_{11}\}$
	$S_{12} = \{O_{15}, O_{14}, O_{13}, O_{12}\}$	$U_{12} = \{O_{15}, O_{14}, O_{13}, O_{12}\}$
	$S_{13} = \{O_0, O_{15}, O_{14}, O_{13}\}$	$U_{13} = \{O_0, O_{15}, O_{14}, O_{13}\}$
	$S_{14} = \{O_1, O_0, O_{15}, O_{14}\}$	$U_{14} = \{O_1, O_0, O_{15}, O_{14}\}$
	$S_{15} = \{O_2, O_1, O_0, O_{15}\}$	$U_{15} = \{O_2, O_1, O_0, O_{15}\}$
Down	$S_{14} = \{O_1, O_0, O_{15}, O_{14}\}$	$V_0 = \{O_1, O_0, \overline{O_{15}}, O_{15}\}$
	$S_{13} = \{O_0, O_{15}, O_{14}, O_{13}\}$	$V_1 = \{O_0, \overline{O_{15}}, O_{15}, O_{14}\}$
	$S_{12} = \{O_{15}, O_{14}, O_{13}, O_{12}\}$	$V_2 = \{\overline{O_{15}}, O_{15}, O_{14}, O_{13}\}$
	$S_{11} = \{O_{14}, O_{13}, O_{12}, O_{11}\}$	$V_3 = \{O_{15}, O_{14}, O_{13}, O_{12}\}$
	$S_{10} = \{O_{13}, O_{12}, O_{11}, O_{10}\}$	$V_4 = \{O_{14}, O_{13}, O_{12}, O_{11}\}$
	$S_9 = \{O_{12}, O_{11}, O_{10}, O_9\}$	$V_5 = \{O_{13}, O_{12}, O_{11}, O_{10}\}$
	$S_8 = \{O_{11}, O_{10}, O_9, O_8\}$	$V_6 = \{O_{12}, O_{11}, O_{10}, O_9\}$
	$S_7 = \{O_{10}, O_9, O_8, O_7\}$	$V_7 = \{O_{11}, O_{10}, O_9, O_8\}$
	$S_6 = \{O_9, O_8, O_7, O_6\}$	$V_8 = \{O_{10}, O_9, O_8, O_7\}$
	$S_5 = \{O_8, O_7, O_6, O_5\}$	$V_9 = \{O_9, O_8, O_7, O_6\}$
	$S_4 = \{O_7, O_6, O_5, O_4\}$	$V_{10} = \{O_8, O_7, O_6, O_5\}$
	$S_3 = \{O_6, O_5, O_4, O_3\}$	$V_{11} = \{O_7, O_6, O_5, O_4\}$
	$S_2 = \{O_5, O_4, O_3, O_2\}$	$V_{12} = \{O_6, O_5, O_4, O_3\}$
	$S_1 = \{O_4, O_3, O_2, O_1\}$	$V_{13} = \{O_5, O_4, O_3, O_2\}$
	$S_0 = \{O_3, O_2, O_1, O_0\}$	$V_{14} = \{O_4, O_3, O_2, O_1\}$
	$S_{15} = \{O_2, O_1, O_0, O_{15}\}$	$V_{15} = \{O_3, O_2, O_1, O_0\}$

Since the first up address of the MAR is $\{O_3, O_2, O_1, O_0\}$, then the next bit to be fed into the MAR is O_4, thus the 16-bit up sequence, $\{O_4, O_5, O_6, …, O_{14}, O_{15}, O_0, O_1, O_2, O_3\}$ (where the O_4 is generated firstly, then O_5, and so on), is extracted from the terminal of D_1, the input of the first flip-flop, and fed serially into the first flip-flop of the 4-bit MAR shown in Fig. 8. During the up address generation, the data in the MAR is from the i-th flip-flop to the (i+1)-th flip-flop (i = 1, 2, 3). All the internal states of the up CLFSR and the 4-bit up memory addresses are tabulated in Table 3. Since the internal states of the up CLFSR can be repeated every 16 clock cycles, therefore all the up memory addresses for any two consecutive up address orders are also produced repeatedly.

If the memory address order of the march algorithm is from the up order changed to the down order. To meet the requirements of the march algorithm, the i-th up memory address U_i is equal to the (15-i)-th down memory addresses V_{15-i} (i = 0, 1, 2, …, 15), thus we

can obtain the 16 simultaneous equations for the down operation as follows:

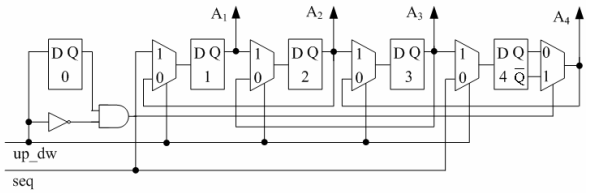


Fig. 8 The 4-bit memory address register (MAR).

$V_0 = U_{15} = \{0, 0, 0, 1\} = \{O_2, O_1, O_0, O_{15}\}$ **(14)**
$V_1 = U_{14} = \{0, 0, 1, 0\} = \{O_1, O_0, O_{15}, O_{14}\}$ **(15)**
$V_2 = U_{13} = \{0, 1, 0, 0\} = \{O_0, O_{15}, O_{14}, O_{13}\}$ **(16)**
$V_3 = U_{12} = \{1, 0, 0, 1\} = \{O_{15}, O_{14}, O_{13}, O_{12}\}$ **(17)**
$V_4 = U_{11} = \{0, 0, 1, 1\} = \{O_{14}, O_{13}, O_{12}, O_{11}\}$ **(18)**
$V_5 = U_{10} = \{0, 1, 1, 0\} = \{O_{13}, O_{12}, O_{11}, O_{10}\}$ **(19)**
$V_6 = U_9 = \{1, 1, 0, 1\} = \{O_{12}, O_{11}, O_{10}, O_9\}$ **(20)**
$V_7 = U_8 = \{1, 0, 1, 0\} = \{O_{11}, O_{10}, O_9, O_8\}$ **(21)**
$V_8 = U_7 = \{0, 1, 0, 1\} = \{O_{10}, O_9, O_8, O_7\}$ **(22)**
$V_9 = U_6 = \{1, 0, 1, 1\} = \{O_9, O_8, O_7, O_6\}$ **(23)**
$V_{10} = U_5 = \{0, 1, 1, 1\} = \{O_8, O_7, O_6, O_5\}$ **(24)**
$V_{11} = U_4 = \{1, 1, 1, 1\} = \{O_7, O_6, O_5, O_4\}$ **(25)**
$V_{12} = U_3 = \{1, 1, 1, 0\} = \{O_6, O_5, O_4, O_3\}$ **(26)**
$V_{13} = U_2 = \{1, 1, 0, 0\} = \{O_5, O_4, O_3, O_2\}$ **(27)**
$V_{14} = U_1 = \{1, 0, 0, 0\} = \{O_4, O_3, O_2, O_1\}$ **(28)**
$V_{15} = U_0 = \{0, 0, 0, 0\} = \{O_3, O_2, O_1, O_0\}$ **(29)**

According to Eqs. (14) to (29), during the down memory address generation, the data flow of the MAR is from the (i+1)-th flip-flop of A_{i+1} to the i-th flip-flop of A_i (i = 1, 2, 3), the down sequence, {O_{15}, O_{14}, O_{13}, …, O_2, O_1, O_0} (where the O_{15} is generated firstly, then O_{14}, and so on), is extracted from the output of the last stage of CLFSR (Q_4) and applied to the last flip-flop of the MAR.

Moreover, we have also noticed that when the up address order is changed to the down address order, the last up address of the MAR in Eq. (14) is {O_2, O_1, O_0, O_{15}} (= {0, 0, 0, 1}) which is also the first down address, however due to the data flow of the MAR, the first down address is supposed to be {O_1, O_0, O_{15}, O_{15}} (= {0, 0, 1, 1}), this may cause contradiction and hence a simple circuitry is required to compensate such a situation.

By Eqs. (4) and (7), in the 4-stage up/down CLFSR, the next bit of the four consecutive zero bits {O_3, O_2, O_1, O_0}, O_4 / O_{15}, is one. To solve this problem, a little modification is required to replace {O_1, O_0, O_{15}, O_{15}} by {O_1, O_0, $\overline{O_{15}}$, O_{15}} during the up-to-down address order:

$V_0 = U_{15} = \{0, 0, 0, 1\} = \{O_2, O_1, O_0, O_{15}\}$
$= \{O_1, O_0, \overline{O_{15}}, O_{15}\}$ **(30)**

As shown in Fig. 8, the compensation circuitry is composed of a flip-flop and two logic gates. The *0*-th flip-flop is used to store the value of the previous address order, the present address order is firstly complemented via the inverter then through the AND gate with the previous address order. When the address order is changed from the up order to the down order, the output of the AND gate is going from low to high then the value of A_4 is switched from Q_4 to $\overline{Q_4}$ such that the Eq. (30) can be compensated. At the next clock, the output of the AND gate is going to low level, then the value of A_4 is switched back to Q_4.

During the down-to-down address order, all the down addresses in Fig. 8 are generated repeatedly. However, during the down-to-up address order, according to Eq. (29), the last down address of the MAR is {O_3, O_2, O_1, O_0} (= {0, 0, 0, 0}), but due to the data flow direction of the MAR, the first up address is supposed to be {O_3, O_3, O_2, O_1}. Fortunately, the values of {O_3, O_2, O_1, O_0} are all-zero, thus

$V_{15} = U_0 = \{0, 0, 0, 0\} = \{O_3, O_2, O_1, O_0\}$
$= \{O_3, O_3, O_2, O_1\}$, **(31)**

and no compensations are required.

Similarly, the 4-bit MAR can be expanded to the k-bit MAR which is shown in Fig. 9.

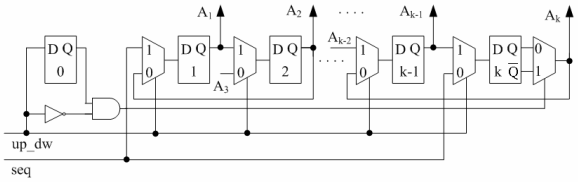


Fig. 9 The k-bit MAR.

The internal states of the k-stage CLFSR and MAR are tabulated in Table 4. The initial states of the k-stage CLFSR and MAR, S_0 and U_0, are all-zero, i.e., {O_{k-1}, O_{k-2}, …, O_0} = {0, 0, …, 0}. By Eqs. (4) and (12), the values of the O_k and O_{L-1} are all one. In the CLFSR with a primitive polynomial of degree k, the output sequence is repeated every 2^k clock cycles. Because the up sequence, {O_k, O_{k+1}, …, O_{L-1}, O_0, O_1, …, O_{k-1}}, is lagged by the down sequence, {O_{L-1}, O_{L-2}, …, O_1, O_0}, with k clock cycles, thus the up and down sequences are extracted at the terminals of D_1 and Q_k, respectively. The internal states of the up/down CLFSR are expressed in Eq. (4) and (12), respectively. For any two successive up/down address order, there are four pairs of address orders. In each pair, the i-th up (dw) memory address, U_i (V_i) (where $0 \le i \le (2^k - 1)$ and $L = 2^k$) is discussed in the following.

(1) up-to-up:

The i-th up memory address U_i is the same as the internal state of the up CLFSR, i.e.,

$U_i = S_i = \{O_{(i+k-1) \bmod L}, O_{(i+k-2) \bmod L}, ..., O_{i \bmod L}\}$
$i = 0, 1, ..., (L-1)$ **(32)**

Since the internal states of the CLFSR with a primitive polynomial of degree k can be repeated every

2^k clock cycles, therefore the up addresses can also be repeated.

Table 4 The internal states of the CLFSR and MAR

Order	CLFSR	MAR
Up	$S_0 = U_0$	(1) up-to-up: $U_0=\{O_{k-1}, O_{k-2}, \ldots, O_0\}$ (2) dw-to-up: $U_0=\{O_{k-1}, O_{k-1}, \ldots, O_1\}$
	$S_1 = U_1$	$U_1 = \{O_k, O_{k-1}, \ldots, O_1\}$
	$S_2 = U_2$	$U_2 = \{O_{k+1}, O_k, \ldots, O_2\}$
	:	:
	$S_{L-2} = U_{L-2}$	$U_{L-2} = \{O_{k-3}, O_{k-4}, \ldots, O_{L-1}, O_{L-2}\}$
	$S_{L-1} = U_{L-1}$	$U_{L-1} = \{O_{k-2}, O_{k-3}, \ldots, O_0, O_{L-1}\}$
Down	S_{L-2}	(1) up-to-dw: $V_0 = \{O_{k-3}, \ldots, O_0, \overline{O_{L-1}}, O_{L-1}\}$ (2) dw-to-dw: $V_0 =\{O_{k-2}, \ldots, O_0, O_{L-1}\}$
	S_{L-3}	$V_1 = \{O_{k-4}, O_{k-5}, \ldots, O_{L-1}, O_{L-2}\}$
	S_{L-4}	$V_2 = \{O_{k-5}, O_{k-6}, \ldots, O_{L-2}, O_{L-3}\}$
	:	:
	S_0	$V_{L-2} = \{O_k, O_{k-1}, \ldots, O_2, O_1\}$
	S_{L-1}	$V_{L-1} = \{O_{k-1}, O_{k-2}, \ldots, O_1, O_0\}$

(2) up-to-dw:

When the address order is changed from the up to the down, the last up address, $U_{L-1} = \{O_{k-2}, O_{k-3}, \ldots, O_0, O_{L-1}\}$, is equal to the first down address, thus the compensation circuitry is triggered to replace $\{O_{k-3}, O_{k-4}, \ldots, O_0, O_{L-1}, O_{L-1}\}$ by $\{O_{k-3}, O_{k-4}, \ldots, O_0, \overline{O_{L-1}}, O_{L-1}\}$ $(= V_0)$. Because $\{O_{k-1}, O_{k-2}, \ldots, O_0\} = \{0, 0, \ldots, 0\}$ and O_{L-1} is one, thus $U_{L-1} = V_0 = \{0, 0, \ldots, 0, 1\}$. The i-th memory address U_i during the consecutive up and down address orders is expressed as:

$$U_i = V_{L-1-i} = \begin{cases} \{O_{k-3}, O_{k-4}, \ldots, O_0, \overline{O_{L-1}}, O_{L-1}\}, \\ \qquad i = (L-1) \text{ and transition occurs} \\ S_i, \ 0 \le i \le (L-1), \text{ elsewhere} \end{cases} \quad \textbf{(33)}$$

(3) dw-to-up:

When the address order is changed from down to up, due to the data flow of the MAR, the last down address, $V_{L-1} = \{O_{k-1}, O_{k-2}, \ldots, O_1, O_0\}$, is equal to $\{O_{k-1}, O_{k-1}, \ldots, O_2, O_1\}$. Since $\{O_{k-1}, O_{k-2}, \ldots, O_0\} = \{0, 0, \ldots, 0\}$, thus no corrections are required and the i-th memory address is $U_i = V_{L-1-i} = S_i$ $(0 \le i \le (L-1))$.

(4) dw-to-dw:

The down CLFSR is used to generate the down sequence which is applied to the MAR. The down sequence is repeated every 2^k clock cycles. Therefore the down addresses can also be repeated smoothly and the i-th address is expressed as $U_i = V_{L-1-i} = S_i$ $(0 \le i \le (L-1))$.

As explained above, a simple compensation circuitry is triggered only when the address order is changed from up to down and the hardware overhead of the compensation circuitry is very low for any size of memory.

4. Conclusions

To generate the up/down memory addresses of the march algorithm with a lower area overhead, a scan based memory address generator based on an up/down complete LFSR has been proposed in this paper. We have analyzed mathematically the structure of the CLFSR and proved that the up/down sequence can be generated by the complete LFSR with a primitive polynomial and its reciprocal polynomial. The architecture of the memory address generator (MAR) has also been concerned in this paper. The relationship between the state transitions of the CLFSR and the memory addresses of the MAR has been clearly investigated in this paper.

The memory address generation for any two consecutive memory address orders, up-to-up, up-to-dw, dw-to-up, and dw-to-dw, has been completely discussed in this paper. For the CLFSR based serial memory address generation, only the address order is switched from up to down is required to compensate by a simple compensation circuitry in the MAR. The hardware overhead of the compensation circuitry is very low for any size of memory under test.

5. References

[1] E. J. Marinissen, B. Prince, D. Keitel-Schulz, Y. Zorian, "Challenges in Embedded Memory Design and Test," *Proc. of Design, Automation and Test in Europe*, 2005, pp. 722 – 727.

[2] A. J. van de Goor, *Testing Semiconductor Memories, Theory and Practice,* ComTex Publishing, Gouda, The Netherlands, 1998.

[3] B. Nadeau-Dostie, A. Silburt, and V. K. Agarwal, "Serial Interfacing for Embedded-Memory Testing," *IEEE Design & Test of Computers*, Vol. 7, No. 2, April 1990, pp. 52 - 63.

[4] M. Nicolaidis, V. C. Alves, and O. Kebichi, "Trade-Offs In Scan Path and BIST Implementations for RAMs," *Proc. of European Test Conf.*, 1993, pp. 169-178.

[5] E. McCluskey and S. Bozorgui-Nesbat, "Design for Autonomous Test," *IEEE Transactions on Circuits and Systems*, Vol. 28, No. 11, Nov. 1981, pp. 1070 - 1079.

[6] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design,* New York: Computer Science Press, New York, 1990.

[7] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, Inc., 1987.