14.3 Multiple scan chains (10 points)

Number of flip-flops in each chain $= \frac{n_{sff}}{n_{chain}}$ and, therefore, Equation 14.1 will change to:

$$\text{Scan test length} = \frac{n_{sff}}{n_{chain}} + 4 + (\frac{n_{sff}}{n_{chain}} + 1) \times n_{comb} + \frac{n_{sff}}{n_{chain}}$$

To compute the gate overhead, we notice that we will need to add a MUX at each scan chain output to multiplex the scan and the normal output. Input pins will not require any additional MUXes as the MUXes added at the first flip-flop of each scan chain can be used to multiplex the corresponding input also. Hence,

$$\text{Gate overhead} = \frac{4 \times n_{chain} + 4 \times n_{sff}}{n_g + 10 \times n_{ff}} \times 100\%$$

(Bushnell and Agrawal) Problem 14.4

Assume 20 equal length scan chains, each having $2000/20 = 100$ flip-flops. Scan sequence test length is given by:

$$\begin{aligned} \text{Scan test length} &= (n_{comb} + 2) \times n_{chain} + n_{comb} + 4 \\ &= (500 + 2) \times 100 + 500 + 4 \\ &= 50,704 \text{ clock cycles} \end{aligned}$$

where n_{comb} = number of combinational vectors, and n_{chain} = number of flip-flops in the longest scan chain.

Gate Overhead: All scanin inputs are obtained as fanouts of normal PIs. A multiplexer is inserted between each PO and its normal output signal. The other data input of the multiplexer is a scanout and control is the test control (TC) PI. Assuming normal data flip-flops of 10 gates, the overheads are:

$$\text{Overhead (single chain)} = 4n_{sff} + 4 = 4 \times 2000 + 4 = 8,004 \text{ gates}$$

where n_{sff} = total number of scan flip-flops. A multiplexer is assumed to have 4 gates. Only one multiplexer is added for the scanout.

$$\text{Extra overhead (20 chains)} = 4(N_{chain} - 1) = 4 \times 19 = 76$$

$$\text{Total gates in pre} - \text{scan circuit} = 100,000 + 10n_{ff} = 120,000$$

where n_{ff} = total number of flip-flops.

$$\text{Extra overhead of 20 chains} = \frac{76}{120,000 + 8,004} \times 100 = 0.06\%$$

An overhead of 0.06% is incurred to reduce the test length by a factor of $\sim$ 20.

14.4 Scan tests (10 points)

Assume 20 equal length scan chains, each having 2000/20 = 100 flip-flops. Scan sequence test length is given by:

$$\begin{aligned}\text{Scan test length} &= (n_{comb} + 2) \times n_{chain} + n_{comb} + 4 \\ &= (500 + 2) \times 100 + 500 + 4 \\ &= 50,704 \text{ clock cycles}\end{aligned}$$

where n_{comb} = number of combinational vectors, and n_{chain} = number of flip-flops in the longest scan chain.

Gate Overhead: All scanin inputs are obtained as fanouts of normal PIs. A multiplexer is inserted between each PO and its normal output signal. The other data input of the multiplexer is a scanout and control is the test control (TC) PI. Assuming normal data flip-flops of 10 gates, the overheads are:

$$\text{Overhead (single chain)} = 4n_{sff} + 4 = 4 \times 2000 + 4 = 8,004 \text{ gates}$$

where n_{sff} = total number of scan flip-flops. A multiplexer is assumed to have 4 gates. Only one multiplexer is added for the scanout.

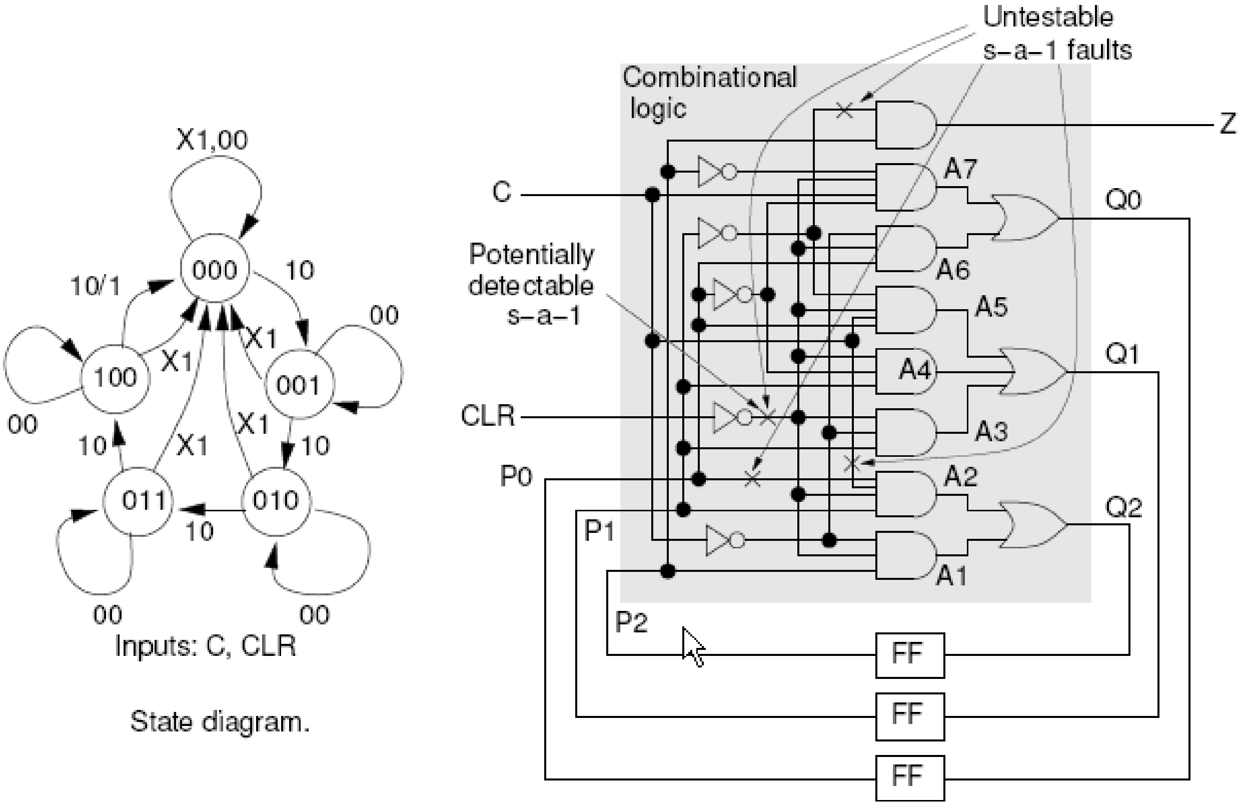
$$\text{Extra overhead (20 chains)} \quad = \quad 4(N_{chain} - 1) = 4 \times 19 = 76$$

Problem 14.5 Modulo-5 counter circuit

The following figure shows a modulo-5 counter circuit. As shown in the state diagram, the states are encoded as 000, 001, 010, 011 and 100. The input $CLR = 1$ initializes the circuit to the 000 state. Input $C = 1$, $CLR = 0$ advances the state at every clock. The clock signal applied to the three D flip-flops is not shown.

The output Z remains 0 with the exception of the state 100, which produces a $Z = 1$ output.

The combinational circuit (shown in the grey box) is made completely single-fault testable by removing redundant faults that were identified by an ATPG program.



State diagram.

Circuit.

Modulo–5 counter.

For the sequential counter, a sequential circuit ATPG program produced 62 vectors to obtain a coverage of $(57/62) \times 100 = 92.98\%$. The five untestable faults were all s-a-1 type and are shown in the figure. Among these the s-a-1 fault on the CLR signal was potentially detected by the test set.

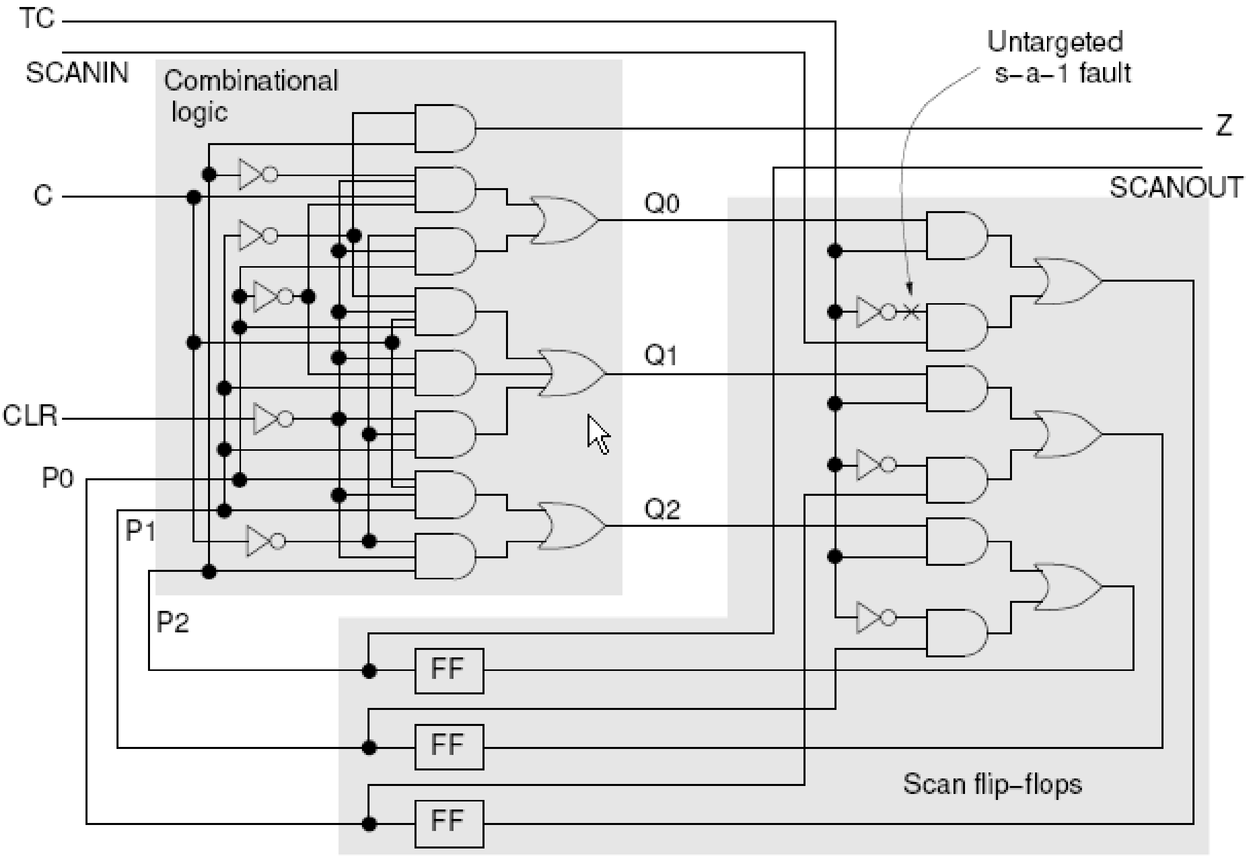
Problem 14.6 Full-scan design

The following figure shows the scan design of the modulo-5 counter.

The number of vectors obtained may vary depending on the ATPG program used. These results were obtained from Bell Labs' Gentest program. The combanitional circuit, whose inputs are C, CLR, $P0$, $P1$ and $P2$, and outputs are Z, $Q0$, $Q1$ and $Q2$, has a collapsed set of 57 faults. All of these faults were detected by 16 vectors.

A complete scan sequence consists of 74 vectors (see Equation 14.1 in the book), which includes 7 vectors for testing the scan register. The scan circuit contains a collapsed set of 79 faults. Fault simulation of the 74-vector sequence showed that 78 faults were detected. The undetected s-a-1 fault is marked on the circuit diagram. It is at the output of the test control (TC) inverter in the first multiplexer.

The reason this fault is not detected is that it was never targeted. Since the scan register test holds TC to 0 for a continuous scan mode, this fault was not activated. The fault is, however, activated every time the circuit is set in the normal mode during the application of the scan sequence. Since in the normal mode the state of $SCANIN$ is considered irrelevant, $SCANIN$ was arbitrarily set to 0. That



Scan–testable modulo–5 counter.

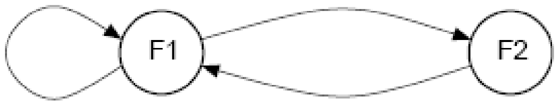
prevented the propagation of the fault effect. A suitable strategy for detecting this fault is to set $Q0$ outputs of the combinational logic as 0 by applying $CLR = 1$. At the same time, the circuit is set in the normal mode by applying $TC = 1$. The fault effect is now propagated to the flip-flop and can be scanned out.

We notice that similar faults in the other two multiplexers were detected by our scan sequence. This is due to the chance occurrence of normal data as 0 and scan data as 1 when $TC = 1$, which would place the fault effect in the flip-flop. $TC = 1$ was always followed by scanout that detected the fault.

In general, it can be recommended that $SCANIN$ is set to 1 whenever the circuit goes to the normal mode (TC=1), provided the AND-OR type of multiplexer is used.

14.8 Partial-scan (10 points)

The s-graph of the circuit in Figure 14.16 is given below.

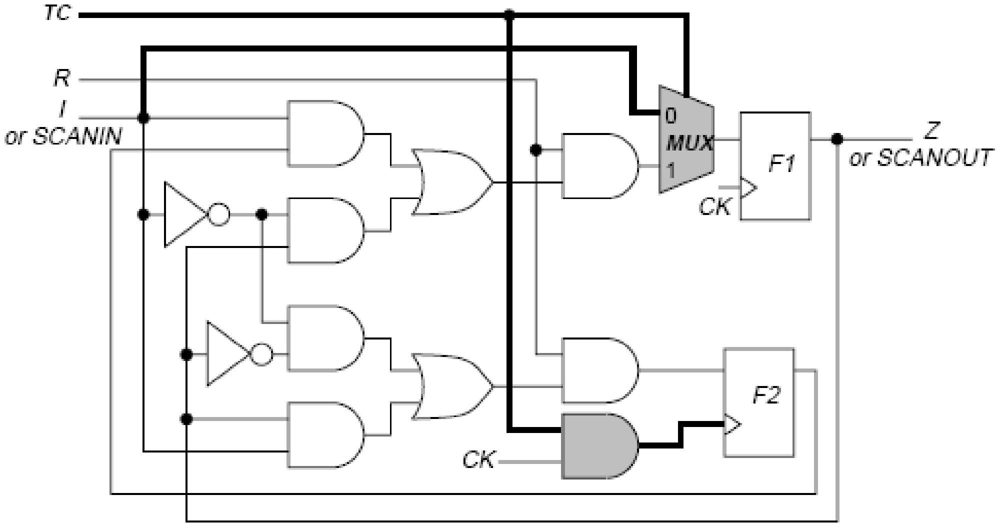


s-graph for the circuit of Figure 14.16.

By scanning F1 all cycles can be eliminated.

Problem 14.9 Partial-scan

The partial-scan circuit is given below. Added circuitry is shown in grey and wiring, in bold lines. We insert a multiplexer at the input of F1. One input of this multiplexer is the normal input of F1. The other input is a fanout of PI I, which is now also used as $SCANIN$. The control input of the multiplexer is a new PI, TC. $TC = 0$ is scan mode and $TC = 1$ is the normal mode. $TC = 0$ also inhibits the clocking of the non-scan flip-flop $F2$ such that it holds its state during the scan operation. This is accomplished by using the grey-shaded AND gate. PO Z also acts as $SCANOUT$.

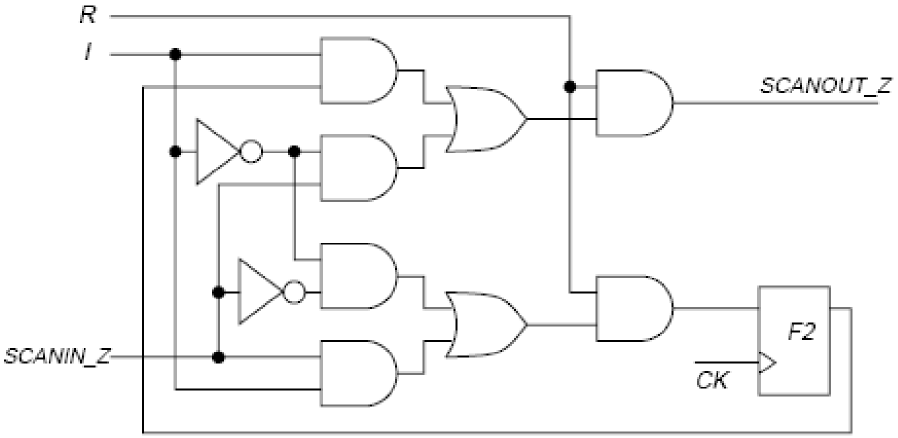


Partial-scan design of the circuit of Figure 14.16.

The ATPG circuit is obtained by removing the MUX and $F1$, making Z a new PI $SCANIN\_Z$, and making the AND gate output feeding into the MUX a new PO $SCANOUT\_Z$. This circuit is shown in the next figure. A sequential circuit ATPG program, GENTEST[1], produced 11 vectors to detect all faults in this circuit. These vectors were converted into scan sequences (see Chapter 14 of the book.) Thus, a set of 28 vectors was produced, which also includes 5 vectors for testing the scan register. The following table shows the test sequence. When the partial-scan circuit

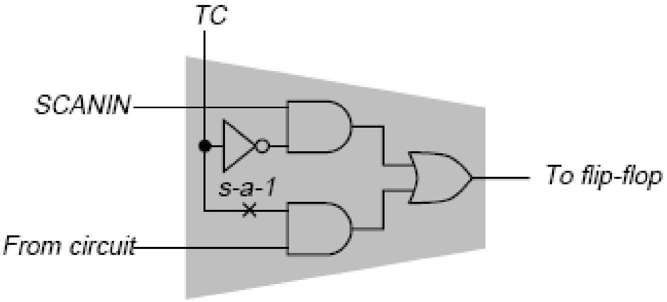
Test sequence for partial-scan design of circuit of Figure 14.16.

Vector	Input bits			Functions
number	TC	R	I	performed
1	0	X	0	Vectors 1 through 5 test scan register in scan mode ($TC = 0$). They apply a 0011 bit stream to I and observe it at Z.
2	0	X	0	
3	0	X	1	
4	0	X	1	
5	0	X	0	
6	0	X	0	Scan-in 0
7	1	0	0	Apply 00 to R and I in normal mode
8	0	X	1	Scan-out Z and scan-in 1
9	1	1	1	Apply 11 to R and I in normal mode
10	0	X	0	Scan-out Z and scan-in 0
11	1	1	0	Apply 10 to R and I in normal mode
12	0	X	1	Scan-out Z and scan-in 1
13	1	1	0	Apply 10 to R and I in normal mode
14	0	X	0	Scan-out Z and scan-in 0
15	1	1	0	Apply 10 to R and I in normal mode
16	0	X	0	Scan-out Z and scan-in 0
17	1	1	1	Apply 11 to R and I in normal mode
18	0	X	1	Scan-out Z and scan-in 1
19	1	1	0	Apply 10 to R and I in normal mode
20	0	X	1	Scan-out Z and scan-in 1
21	1	1	1	Apply 11 to R and I in normal mode
22	0	X	0	Scan-out Z and scan-in 0
23	1	1	1	Apply 11 to R and I in normal mode
24	0	X	0	Scan-out Z and scan-in 0
25	1	1	1	Apply 11 to R and I in normal mode
26	0	X	1	Scan-out Z and scan-in 1
27	1	0	0	Apply 00 to R and I in normal mode
28	0	X	0	Scan-out Z and scan-in 0



ATPG circuit for the partial-scan design of the circuit of Figure 14.16.

was simulated in the sequential mode, these 28 vectors detected all faults, except one fault that was potentially detected. That fault was a s-a-1 fault in the MUX circuit and is shown in the next figure. This happened because we left the input R in the unknown state (X) during the scan mode. If $R = 0$ was used instead, the s-a-1 fault in the MUX would not be detected. However, if $R = 1$ was used, then that fault would have been detected. The detection of such faults is not guaranteed since they are not targeted by the ATPG. Being a part of the scan structure, the MUX is not included in the ATPG circuit. This is a typical situation for scan design.



Potentially detected fault in the scan multiplexer.

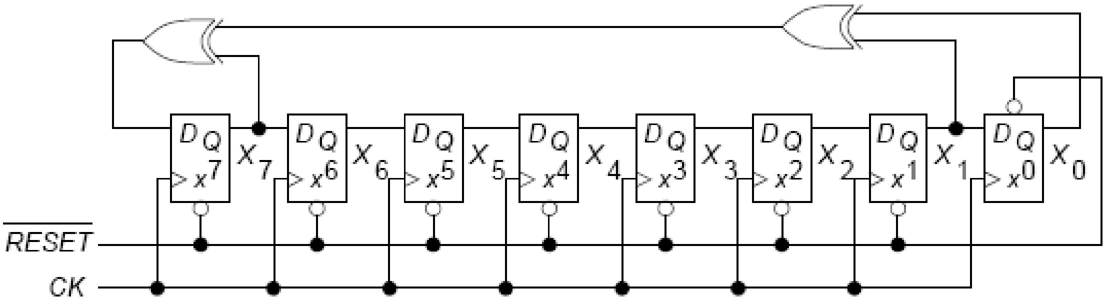
14.10 Partial-scan (15 points)

Suppose we arbitrarily select one non-scan flip-flop and scan all other flip-flops. Since there are no self-loops in the original s-graph, this partial scan circuit has no cycles. We will prove the optimality of this design by showing that no flip-flop in this design can be dropped from scan without creating a cycle. Suppose we were to drop one flip-flop from scan. Because the s-graph is fully connected, the two non-scan flip-flops will form a cycle of length two. By a similar argument, no flip-flop can be dropped from scan without creating a cycle. Thus, the single non-scan flip-flop design is optimal. ■

15.2 Standard LFSR (10 points)

Consider the polynomial for a standard LFSR shown in the figure:

$$f(x) = x^8 + x^7 + x^2 + 1$$



A standard LFSR

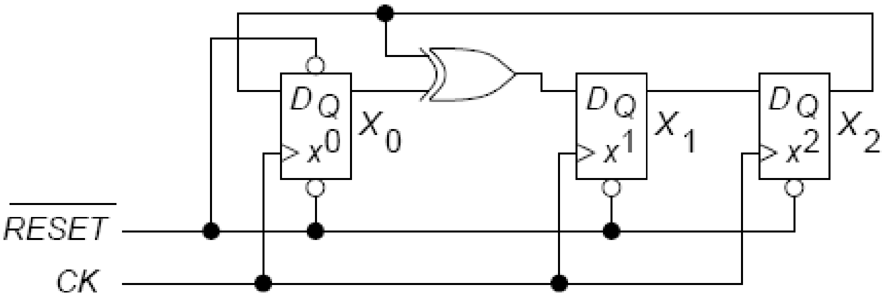
$$\mathbf{X}(t+1) = \mathbf{T_s}\mathbf{X}(t)$$

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ X_3(t+1) \\ X_4(t+1) \\ X_5(t+1) \\ X_6(t+1) \\ X_7(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \\ X_5(t) \\ X_6(t) \\ X_7(t) \end{bmatrix}$$

15.3 Modular LFSR (10 points)

For the modular LFSR shown in the figure, consider the polynomial:

$$f(x) = x^3 + x + 1$$



Modular LFSR.

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix}(t+1) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix}(t)$$

15.4 Standard LFSR

Pattern #	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0
1.	0	0	0	0	0	0	0	1
2.	1	0	0	0	0	0	0	0
3.	1	1	0	0	0	0	0	0
4.	1	1	1	0	0	0	0	0
5.	1	1	1	1	0	0	0	0
6.	1	1	1	1	1	0	0	0
7.	1	1	1	1	1	1	0	0
8.	0	1	1	1	1	1	1	0

15.6 MISRs (20 points)

Equations representing MISR:

$$\mathbf{X}(t+1) = \mathbf{T_s}\mathbf{X}(t) + \mathbf{I}(t)$$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} (t+1) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} (t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ B \\ A \end{bmatrix}$$

Equations for modular MISR:

$$\mathbf{X}(t+1) = \mathbf{T_s^T}\mathbf{X}(t) + \mathbf{I}(t)$$

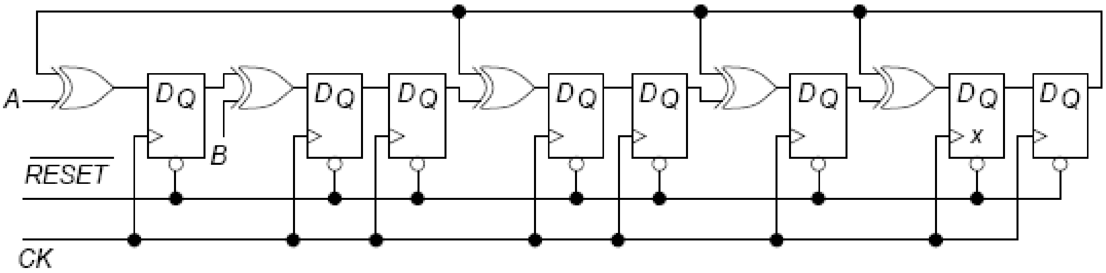
Standard equation: $\mathbf{X}(t+1) = \mathbf{T_s}\mathbf{X}(t)$

Transpose: $\mathbf{X^T}(t+1) = \mathbf{X^T}(t)\mathbf{T_s^T}$

Post-multiply both sides by $\mathbf{X^T}$ and pre-multiply both sides by $\mathbf{X_2}$, to get

$$\mathbf{X_2}(t+1) = \mathbf{T_s^T}\mathbf{X_2}(t)$$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} (t+1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} (t) + \begin{bmatrix} A \\ B \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



Taps: $h_3,\ h_5,\ h_6$

The modular LFSR gives the true remainder of the

$$\sum_{A,B}\left(\frac{\text{output sequence}}{\text{primitive polynomial}}\right)$$

where $\sum$ is the XOR operator.

$$\begin{aligned}\mathbf{X}(t+1) &= \mathbf{T_s X}(t)\\ \mathbf{X^T}(t+1) &= (\mathbf{T_s X}(t))^{\mathbf{T}}\\ &= \mathbf{X^T}(t)\mathbf{T_s^T}\\ &= \mathbf{X^T}(t)\mathbf{T_M}\end{aligned}$$

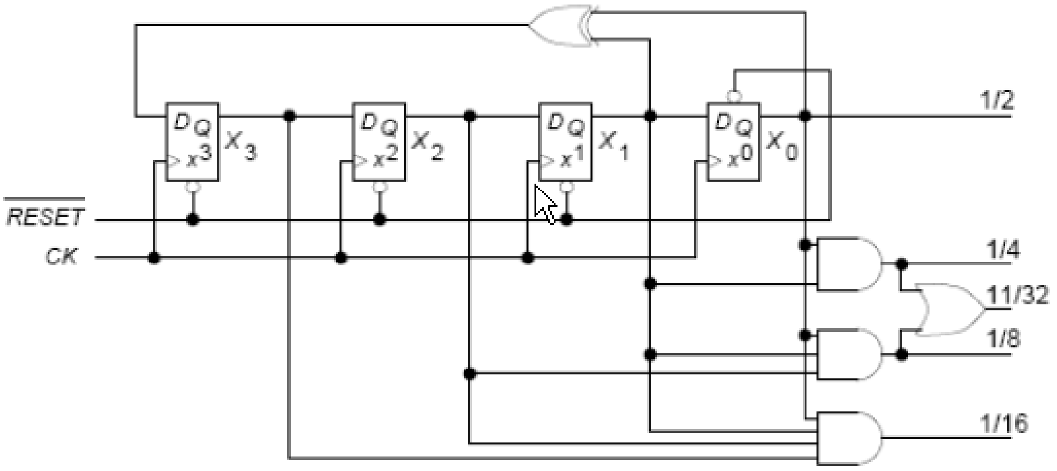
The standard signature is a different state table realization of the modular MISR signature.

15.8 Weighted random pattern generator

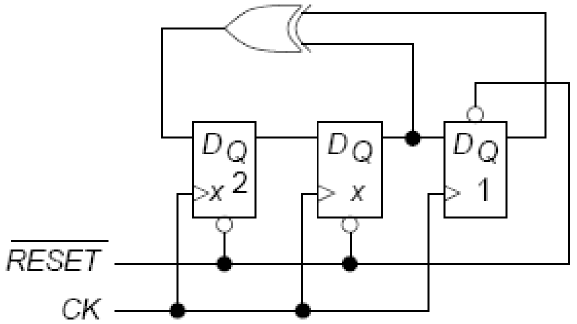
Use a 4-bit pattern generator. From Appendix B of the book, the primitive polynomial is:

$$x^4 + x + 1$$

A circuit to generate the required weights is shown below.



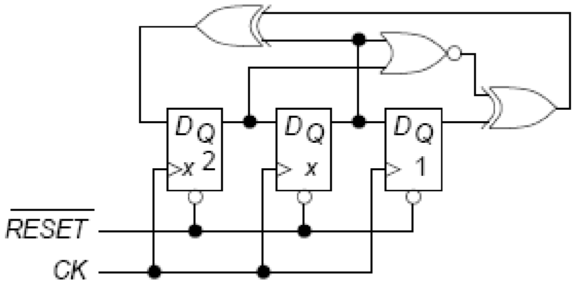
(Bushnell and Agrawal) Problem 15.13.
A standard LFSR and its patterns are shown below.



Standard LFSR.

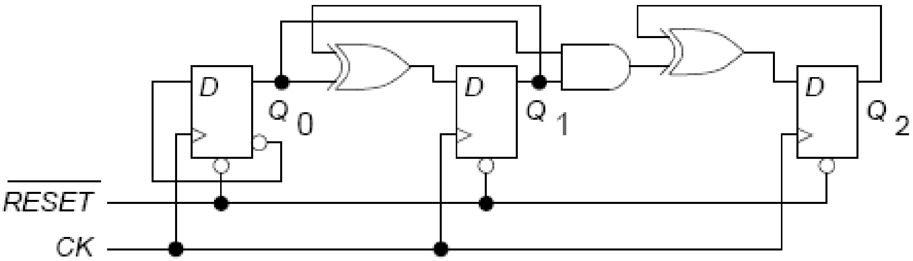
1	1	0	0	1	0	1	1	1
x	0	0	1	0	1	1	1	0
x^2	0	1	0	1	1	1	0	0

The next figure gives an augmented LFSR and the patterns it produces. This definitely uses less hardware than a counter, which needs more complex gates. It gets comparatively simpler as the counter width increases. A counter and its patterns are also shown below.



Augmented LFSR.

1	0	0	0	1	0	1	1	1	0
x	0	0	1	0	1	1	1	0	0
x^2	0	1	0	1	1	1	0	0	0



Counter.

Q_0	0	1	0	1	0	1	0	1	0
Q_1	0	0	1	1	0	0	1	1	0
Q_2	0	0	0	0	1	1	1	1	0

15.14 Aliasing analysis

$$Z = Y(B \oplus C) \oplus B$$

Results of circuit simulation are as follows:

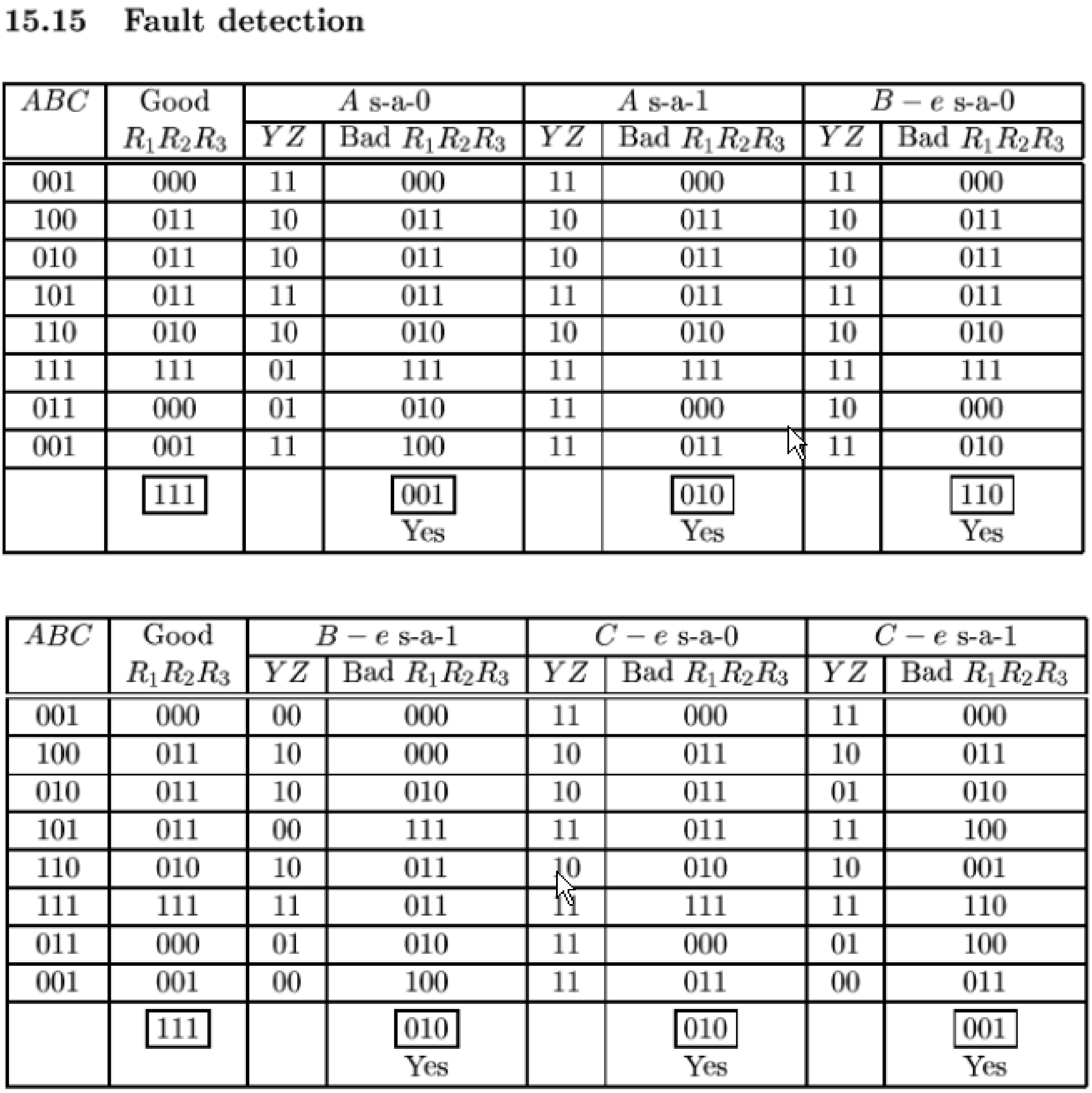
A	B	C	Y	Z	Good machine	Failing machine, e sa0
					$R_1R_2R_3$	$R_1R_2R_3$
0	0	1	D	D	000	000
1	0	0	D	0	011	000
0	1	0	D	$\overline{D}$	011	000
1	0	1	D	D	011	001
1	1	0	1	0	010	100
1	1	1	1	1	111	000
0	1	1	0	1	000	011
0	0	1	D	D	001	000
1	0	0	D	0	$\boxed{111}$	$\boxed{000}$

For output Y, the fault effect is XORed four times, while the fault effect is XORed into Z three times, during the first 7 clock periods. Repeating the first LFSR pattern during the 8th clock period XORs the fault effect in one additional time frame on each output.

The error vector is set to 1 on an output when it differs from a good machine. Here are the other error vectors:

	1	2	3	4	5	6	7	8
Y	1	1	1	1	0	0	0	1
Z	1	0	1	1	0	0	0	1

Even with the repeated pattern, the cumulative # of 1's in the error vector remains odd. This is why aliasing does not occur. If the total # of 1's in the error vector becomes even, then aliasing might occur.



15.15 Fault detection

ABC	Good	A s-a-0		A s-a-1		$B-e$ s-a-0	
	$R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$
001	000	11	000	11	000	11	000
100	011	10	011	10	011	10	011
010	011	10	011	10	011	10	011
101	011	11	011	11	011	11	011
110	010	10	010	10	010	10	010
111	111	01	111	11	111	11	111
011	000	01	010	11	000	10	000
001	001	11	100	11	011	11	010
	$\boxed{111}$		$\boxed{001}$ Yes		$\boxed{010}$ Yes		$\boxed{110}$ Yes

ABC	Good	$B-e$ s-a-1		$C-e$ s-a-0		$C-e$ s-a-1	
	$R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$
001	000	00	000	11	000	11	000
100	011	10	000	10	011	10	011
010	011	10	010	10	011	01	010
101	011	00	111	11	011	11	100
110	010	10	011	10	010	10	001
111	111	11	011	11	111	11	110
011	000	01	010	11	000	01	100
001	001	00	100	11	011	00	011
	$\boxed{111}$		$\boxed{010}$ Yes		$\boxed{010}$ Yes		$\boxed{001}$ Yes

15.16 Fault detection

ABC	Good	B s-a-0		B s-a-1		$B-g$ s-a-0	
	$R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$
001	000	11	000	01	000	11	000
100	011	10	011	10	001	10	011
010	011	10	011	10	110	11	011
101	011	11	011	11	101	11	010
110	010	10	010	10	101	11	110
111	111	11	111	11	100	10	100
011	000	11	000	01	001	01	000
001	001	11	011	01	101	11	001
	$\boxed{111}$		$\boxed{010}$ Yes		$\boxed{111}$ No		$\boxed{111}$ No

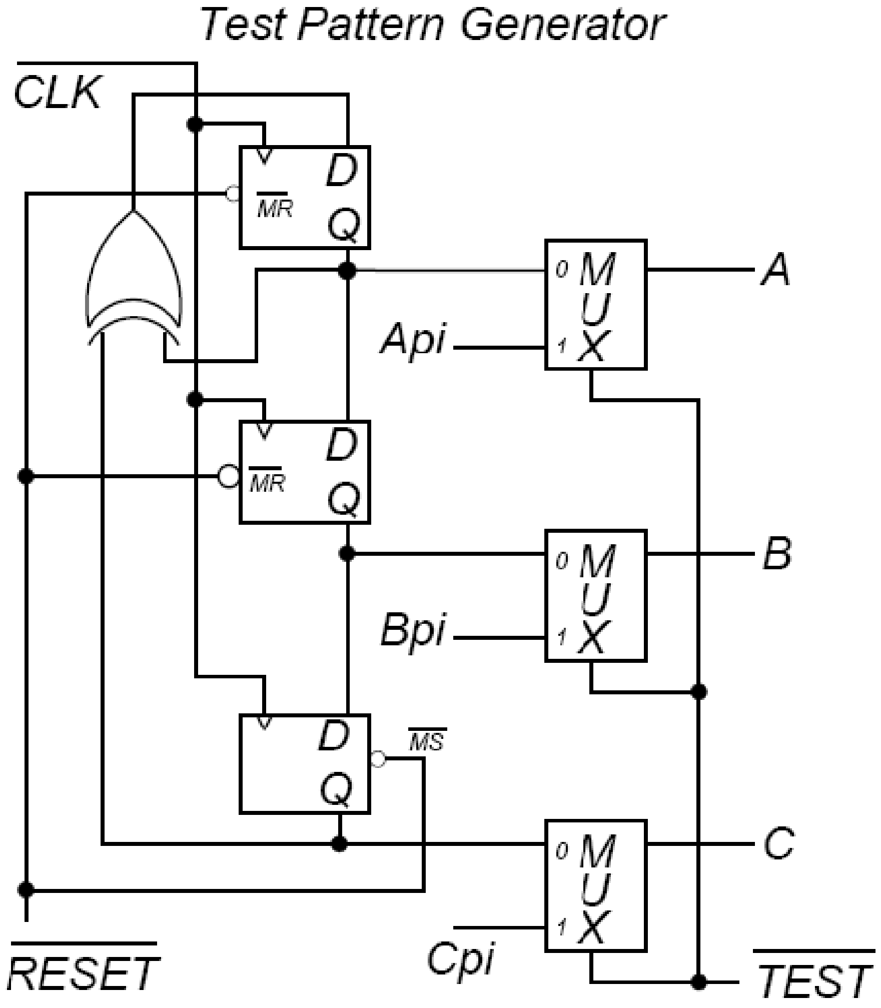
Problem 15.17 Fault detection

ABC	Good	C s-a-0		C s-a-1		$C-g$ s-a-0	
	$R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$
001	000	10	000	11	000	10	000
100	011	10	010	11	011	10	010
010	011	10	111	01	010	10	111
101	011	10	001	11	100	10	001
110	010	10	110	11	001	10	110
111	111	10	101	11	111	10	101
011	000	10	100	01	000	01	100
001	001	10	000	11	001	10	011
	$\boxed{111}$		$\boxed{010}$ Yes		$\boxed{111}$ No		$\boxed{011}$ Yes

ABC	Good	$C-g$ s-a-1		$f-Y$ s-a-0		$f-Y$ s-a-1	
	$R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$	YZ	Bad $R_1R_2R_3$
001	000	11	000	00	000	10	000
100	011	11	011	00	000	10	010
010	011	11	010	00	000	10	111
101	011	11	110	01	000	11	001
110	010	11	100	00	001	10	111
111	111	11	001	01	100	11	001
011	000	01	111	01	011	11	111
001	001	11	010	01	000	11	000
	$\boxed{111}$		$\boxed{110}$ Yes		$\boxed{001}$ Yes		$\boxed{011}$ Yes

15.19 Signature computation

(a) The hardware is shown in following figure.

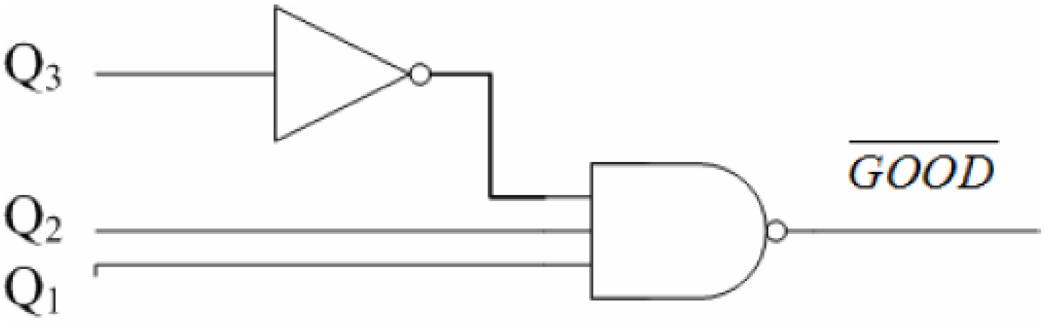


Circuit for Problem 15.19 with BIST pattern generator and input MUX.

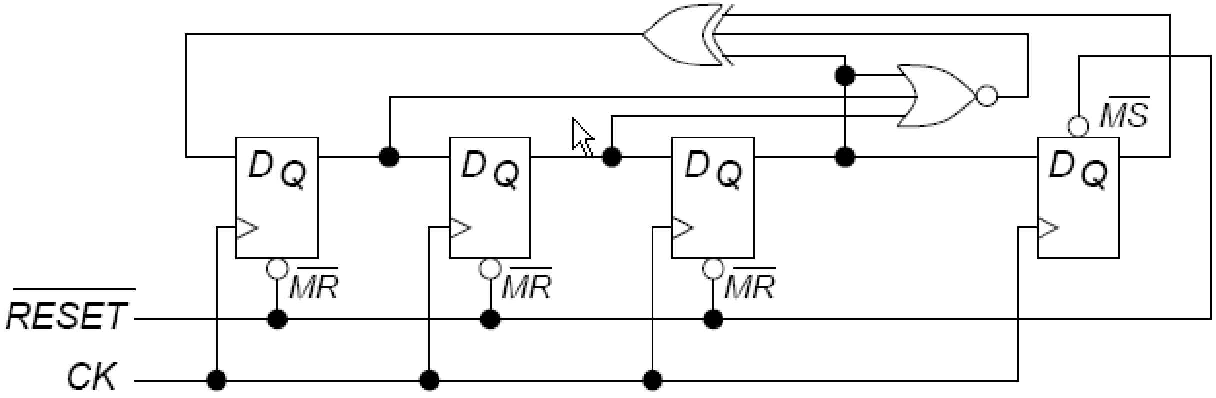
(b)

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}(t+1) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}\begin{bmatrix} a \\ b \\ c \end{bmatrix}(t)$$

(c) The initial state of the flip-flops is assumed to be $Q_1Q_2Q_3 = 000$. Good signature is $Q_1Q_2Q_3 = 110$. The NAND gate-signature comparator should be designed as following: only $Q_1Q_2Q_3 = 110$ can make $\overline{GOOD} = 0$, other values have to make $\overline{GOOD} = 1$.



15.24 Up/Down LFSR (BONUS – 100 points) From Appendix B, a primitive polynomial for a 4-bit LFSR is $f(x) = 1 + x + x^4$. The following circuit produces all patterns including 0000, which appears immediately after the initialization pattern, 0001. A NOR gate has been added to the basic LFSR to produce the 0000 pattern. The pattern sequence is shown after the circuit diagram.



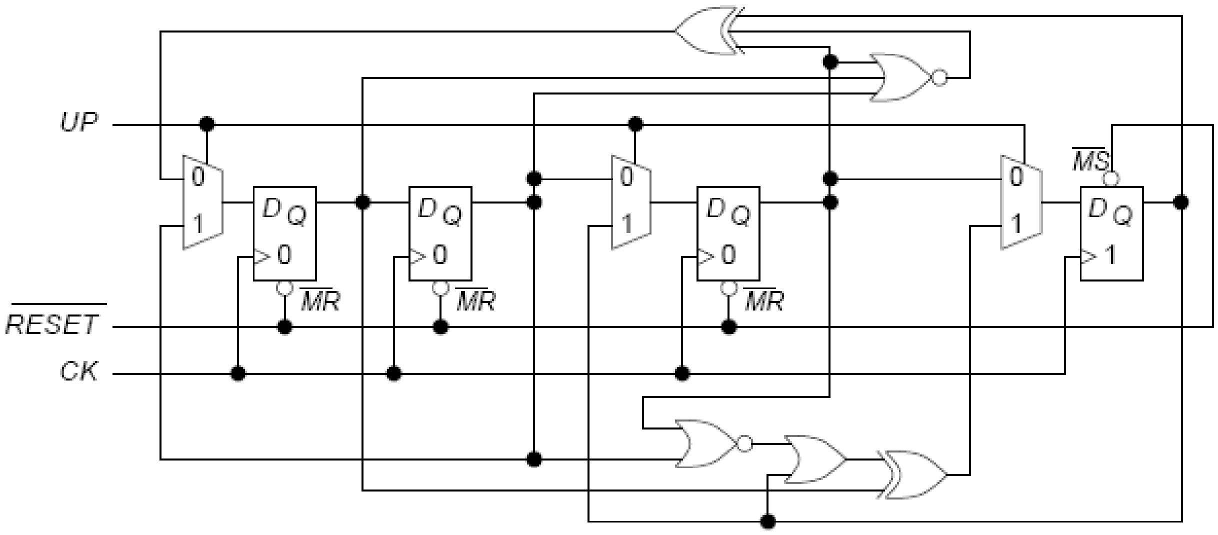
Pattern No.	Pattern	Decimal value	Remarks
1	0001	1	Initialization pattern
2	0000	0	Forced by NOR gate
3	1000	8	
4	0100	4	
2	0010	2	
6	1001	9	
7	1100	12	
8	0110	6	
9	1011	11	
10	0101	5	
11	1010	10	
12	1101	13	
13	1110	14	
14	1111	15	
15	0111	7	
16	0011	3	
17	0001	1	Sequence starts repeating

To find the inverse LFSR, we compute the inverse characteristic function:

$$\frac{1+x+x^2+x^3+x^4+x^5+x^6+x^7+x^8}{f(x)}$$

$$
\begin{array}{r|ccccccccc}
 & 1 & & x^2 & & x^4 & & & & \\
\cline{2-10}
1+x+x^4 & 1 & +x & +x^2 & +x^3 & +x^4 & +x^5 & +x^6 & +x^7 & +x^8 \\
 & 1 & +x & & & +x^4 & & & & \\
\cline{2-10}
 & & & x^2 & +x^3 & & +x^5 & +x^6 & +x^7 & +x^8 \\
 & & & x^2 & +x^3 & & & +x^6 & & \\
\cline{4-10}
 & & & & & & x^5 & & +x^7 & +x^8 \\
 & & & & & x^4 & +x^5 & & & +x^8 \\
\cline{6-10}
 & & & & & x^4 & & & +x^7 &
\end{array}
$$

This does not evenly divide the all 1's polynomial and we get a remainder of $x^4 + x^7$. We conclude that the inverse LFSR does not exist, so we must synthesize it as a finite state machine. The following circuit is based on a design synthesized by Synopsys.



Up/Down LFSR of Problem 15.24

(Bushnell and Agrawal) Problem 16.2.

Boundary scan uses two MUXes and two FF's per pin. With four gates (14 transistors) per MUX and ten gates (44 transistors) per master-slave FF, we get

$$\text{Transistors per I/O} = 2 \times 14 + 2 \times 44 = 116$$

$$\begin{aligned}\text{Hardware cost} &= \#pins \times \frac{\#transistors}{\#\ of\ I/O} \times \frac{cost}{transistor} + TAP \\ &= (256 \times 116 + 262) \times 525 \times 10^{-6} \text{ cents} = 15.73 \text{ cents}\end{aligned}$$

$$\begin{aligned}\text{Test time in TCK's} &= 5 + \#pins \times \#vectors \\ \text{Test time cost} &= \frac{5 + 256 \times 512,000}{200 \times 10^{6}\ Hz} \times (4.5 \text{ cents/s}) \\ &= 2.94912 \text{ cents}\end{aligned}$$