

ECE 512 Digital System Testing and Design for Testability

Model Solutions for Assignment #3

- 14.1) In a fault-free instance of the circuit in Fig. 14.15, holding the input low for two clock cycles should keep the output at 0; holding the input high for two clock cycles should produce outputs of 0 followed by 1, or 1 followed by 0.

In the presence of the indicated stuck-at fault, the output of the circuit will keep on toggling between 0 and 1, regardless of the input signal that is applied. A test for the fault is therefore to hold the input low, and to verify that the output does not toggle. This test will therefore take two clock cycles since the initial state of the flip-flop is unknown. To make the test deterministic, the circuit must be modified to all the initial flip-flop state to be forced. This could be done by inserting a 2-input AND gate just before the D input of the flip-flop, with the second input controlled by a new input test signal. In normal mode, this new input would be held at 1; to force initialization, this input would be held at 0 while the clock is pulsed once. A second modification would be to use a flip-flop with a reset input signal.

14.3) Scan test length = $(n_{\text{comb}} + 3) n_{\text{sff}} + 4$ (14.1)

$$\text{Gate overhead of scan} = (4 \times n_{\text{sff}} / n_g) \times 100\% \quad (14.2)$$

Now assume that the flip-flops are split into n_{chain} scan chains of equal length, which can be accessed in parallel with no time penalty. The scan chain then directly reduces in effective length to n_g / n_{chain} . So we have the following new equations:

$$\text{New scan test length} = (n_{\text{comb}} + 3) (n_{\text{sff}} / n_{\text{chain}}) + 4$$

There should be minimal extra area since the number of scannable flip-flops has not changed, and since the 20 scan chains can be accessed using the existing 20 primary inputs and 20 primary outputs. The test enable input must be present regardless of the number of scan chains. There will be extra routing required to connect the 9 additional scan chains to the input and output pins, but the additional required area is likely to be small if one assumes a modern CMOS process with multiple layers of metal interconnect as well as over-the-cell routing. So equation (14.2) stays the same.

- 14.4) Original chip specs: $n_g = 100,000$ gates & $n_{\text{sff}} = 2,000$ flip-flops & $n_{\text{comb}} = 500$ test vectors

$$\begin{aligned} \text{Scan test length} &= (n_{\text{comb}} + 3) n_{\text{sff}} + 4 \\ &= (500 + 3) 2000 + 4 \\ &= 1,006,004 \text{ clock periods (assuming only one scan chain)} \end{aligned}$$

$$\begin{aligned}
\text{Gate overhead of scan} &= (4 \times n_{\text{sff}}) / n_g \times 100\% \\
&= (4 \times 2000) / 100000 \times 100\% \\
&= 8\%
\end{aligned}$$

Now assume that the scan chain is broken up into 20 equal-length parallel scan chains. Each scan chain now has 100 flip-flops.

$$\begin{aligned}
\text{Scan test length} &= (n_{\text{comb}} + 3) (n_{\text{sff}} / n_{\text{chain}}) + 4 \\
&= (500 + 3) 100 + 4 \\
&= 50,304 \text{ clock periods (assuming 20 scan chains)}
\end{aligned}$$

The area overhead should be about the same, according to the earlier argument.

- 15.4) The so-called “standard LFSR” has an external XOR gate network. We are given that the characteristic polynomial is $f(x) = x^8 + x^7 + x^2 + 1$. Let the outputs of the eight LFSR flip-flops, in the (rightward) direction of shifting, be labeled x^7, x^6, \dots, x^0 . According to $f(x)$, the leftmost flip-flop input is determined by the XOR of x^7, x^2 and x^0 . The first eight patterns, starting with an initial pattern of 00000001 are given below. The bold labels indicate which values must be XORed together to obtain the next value of x^7 . The new values of x^6, \dots, x^0 are obtained by right shifting the old values of x^7, \dots, x^1 .

| x7 | x6 | x5 | x4 | x3 | x2 | x1 | x0 |
|-----------|----|----|----|----|-----------|----|-----------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

- 15.5) The so-called “modular LFSR” has internal XOR gates located in between some of the flip-flops. We are given that the characteristic polynomial is $f(x) = x^3 + x + 1 = 1 + x + x^3$. Let the outputs of the three LFSR flip-flops, in the (rightward) direction of shifting, be labeled x^0, x^1 , and x^2 . According to the definition of such LFSRs, the new value of x^0 is simply the old value of x^2 ; and the new value of x^1 is the XOR of the old value of x^0 and the old value of x^2 ; and all other flip-flop states are obtained by right-shifting. The first eight patterns, starting with an initial pattern of $(x^0, x^1, x^2) = (1,0,0)$ are given below.

| | | | | | | | | | | |
|----|----|----|--|----|----|-----------|--|----|----|----|
| X0 | <- | X2 | | X1 | <- | X0 XOR X2 | | X2 | <- | X1 |
| X0 | X1 | X2 | | X1 | | | | | | |
| 1 | 0 | 0 | | | | | | | | |
| 0 | 1 | 0 | | 1 | = | 1 XOR 0 | | | | |

| | | | |
|---|---|---|-------------|
| 0 | 0 | 1 | 0 = 0 XOR 0 |
| 1 | 1 | 0 | 1 = 0 XOR 1 |
| 0 | 1 | 1 | 1 = 1 XOR 0 |
| 1 | 1 | 1 | 1 = 0 XOR 1 |
| 1 | 0 | 1 | 0 = 1 XOR 1 |
| 1 | 0 | 0 | 0 = 1 XOR 1 |

Note that all seven nonzero patterns are generated because the characteristic polynomial $f(x)$ happens to be primitive.

15.7) The generated unweighted and weighted outputs from the LFSR in Fig. 15.15 are as follows:

| x7 | x6 | x5 | x4 | x3 | x2 | x1 | x0=1/2 | 1/4 | 1/8 | 1/16 |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---------------|------------|------------|-------------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

It is helpful to construct a truth table which shows the output of the good 4-input XOR circuit as well as the outputs of all of the considered faulty circuits. In the following table node “g” is the XOR of inputs “a” and “b”, and node “h” is the XOR of inputs “c” and “d”.

| a | b | c | d | g | h | f | a0 | a1 | b0 | b1 | c0 | c1 | d0 | d1 | g0 | g1 | h0 | h1 | f0 | f1 |
|----------|----------|----------|----------|----------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

```

1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0 1 0 1
1 0 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1
1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1
1 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

```

The shortest possible set of test vectors can be found by selecting the minimum number of rows in the truth table that ensure that each faulty behaviour is observed at least once (i.e., the output differs from the good output). Certainly no more than four vectors are required since 0101, 1010, 0001 and 0100 together detect all of the faults. However, the vectors must be generated by the LFSR, and the most effective test vectors may not be generated very soon. If one makes the connections $a=X5$, $b=X7$, $c=X6$ and $d=X0$ then all of the faults are detected in seven clocks cycles using the above LFSR. The weights are not particularly useful in this circuit since there are no high fan-in AND or OR gates. It may be possible to detect all of the faults in fewer than seven clock cycles, but it would take quite a bit of trial and error to find a better connection from the LFSR outputs to the a, b, c and d inputs of the circuit under test.

- 15.9) A rule 150 cellular automaton determines the next state of a flip-flop as the XOR of the old state of the same flip-flop along with the states of the left and right neighbours. There are several choices for handling the leftmost and rightmost flip-flops. One choice is to include permanent 0 signals from the missing neighbours (this is the approach illustrated in Fig. 15.17 in the textbook). Another choice is to treat the leftmost and rightmost flip-flops as neighbours. If one seeds these two possible designs with “0001” then one obtains the following two states sequence:

0s at the ends

```

A  B  C  D
0  0  0  1
0  0  1  1
0  1  0  0
1  1  1  0
0  1  0  1
1  1  0  1
0  0  0  1

```

A and D are neighbours

```

A  B  C  D
0  0  0  1
1  0  1  1
0  0  0  1

```

Given the characteristic polynomial $f(x) = 1 + x^4$, both a standard (external XOR) LFSR and a modular (internal XOR) LFSR have the same structure. This LFSR has no taps, and has a feedback wire that routes the value of the rightmost flip-flop back to the leftmost flip-flop. The resulting state sequences are given below:

External XOR

Internal XOR

| X3 | X2 | X1 | X0 |
|----|----|----|----|
| 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |

| X0 | X1 | X2 | X3 |
|----|----|----|----|
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 |

Note that both LFSR sequences only go through four distinct states before repeating. The state sequence would be extended to 15 states, without repetition, if the LFSRs were to be based on a primitive characteristic polynomial instead of $f(x) = 1 + x^4$.

In this example, the first of the two cellular automata (CA) is slightly better than the two LFSRs because the state states are more “random” and the state sequence contains six distinct states (as opposed to four distinct states in the LFSRs). The second CA design, however, produces only two distinct states, and is therefore inferior to the LFSRs. The moral of this story is that the design of the LFSRs and CAs must be done carefully to ensure that a maximum-length state sequence is obtained.

- 15.10) The maximum-length LFSR from question 15.5 can be readily modified to solve this problem. An inverter can be added to the output of flip-flop x1. This will remap the states a bit; in particular, state 010 will be mapped to state 000. State 010 will disappear from the sequence (but we are told that this vector is not useful for detecting any faults). All of the other nonzero states will still be present, but will appear in a different order from before.
- 15.11) Error vector e has an error probability $p = 0.3$. The number of bits in the LFSR that is used for response compaction is given as $k = 15$. In that case, the probability of aliasing lies between the lower bound $p^k = 0.3^{15} = 0.000000014348907 = 0.0143$ ppm, and the upper bound $(1 - p)^k = 0.7^{15} = 0.00474756151 = 4747.56$ ppm. Obviously this is a very wide range of possible aliasing probabilities. The range would be narrower if p were to be closer to 0.5. Also, the aliasing probability is lowered by simply increasing the LFSR width k .
- 15.14) Solution omitted.
- 15.15) Solution omitted.