A technique for designing self-checking state machines with low power consumption

P. K. Lala [1]

Abstract

This paper presents a state assignment technique for on-line detection of a single bit error or a unidirectional multibit error in a state code during state transitions in finite state machines. The state assignment is based on *k-out-of-2k* (*2k+1*) code; the code words are assigned such that only a minimum number state bits change their values during state transitions, resulting in lower power consumption .

Introduction

Traditionally the goal of the state assignment process in a finite state machine design has been to reduce the area required to implement the machine. In the past many researchers have proposed state assignment techniques for minimum area implementation of two-level and multi-level implementation of next state logic [1-5]. As the complexity of digital systems implemented in CMOS grows, a major concern is the reduction of power dissipation. In general, the power consumption in a CMOS circuit can be represented by the following expression [6]:

$$P = \tfrac{1}{2} CV_{DD}^2 f N + Q_{SC}V_{DD} f N + I_{leak} V_{DD}$$

where, $\tfrac{1}{2} CV_{DD}^2$ = energy involved in charging or discharging a circuit node with capacitance
N = average number of times the nodes switch
f = frequency of operation
Q_{SC} = quantity of charge carried by the short-circuit current per transition
$I_{leak} V_{DD}$ = static power dissipation due to leakage current I_{leak}

It has been shown in [7] that the switching activity produces more than 90% of the total power consumption in CMOS circuits. Thus, a state assignment that reduces the number of state bit transitions while moving from one state to another will result in a reduction of power consumption in state machines implemented in CMOS. A few techniques have been proposed in recent years for state assignment that result in low power consumption e.g. SYCLOP [8,9] , SABSA [10].

In addition to minimum area and low power consumption , testability has become a major issue in digital system design. Several techniques have been developed over the years to enhance the testability of state machines[11]. However, not much work has been reported to improve the on-line testability of state machines. The on-line testability of a state machine can be improved by ensuring that an erroneous state transition is detected during the normal operation of the state machine. In order to guarantee that a fault in the next state logic cannot change a valid state into a non-valid state , it is necessary to encode the states. If each state corresponds to **a correct** code word, the next state logic

[1] Department of Electrical Engineering, University of South Florida, 4202 E. Fowler Avenue ENB 118, Tampa, FL 33620, e-mail: lala@eng.usf.edu

can be considered to be fault-free. Also, **if the** next state logic is **fault-secure**, then a state code represents a valid and correct state; in other words, a fault in the next state logic does not result in a valid but an incorrect state.

This paper presents a technique for generating a state encoding that guarantees on-line detection of erroneous state transitions in finite state machines, and minimizes the number of bits that change during a state transition. Thus, a state machine designed using the proposed state assignment technique will consume less power, and will also have on-line error checking capability.

2. State Assignment Technique

A **finite** state machine is usually represented by a state transition graph (STG). Fig.1 shows the STG of a state machine. The proposed state assignment technique consists of two separate tasks. In the first task the probability of transition from a present state $\mathbf{S_i}$ to a next state $\mathbf{S_j}$, denoted as $\mathbf{P(S_i \rightarrow S_j)}$, is computed. During the second task , code words from a selected ***k-out-of-2k*** (or *2k +1*) code is assigned to each state. This ensures that a single bit error or a unidirectional multibit error in a state code can be detected on-line. Also, this assignment is done such that during the majority of the state transitions only the minimum number of bits changes their logic values.

The ***k-out-of-2k*** **(*2k*+1)** code with minimum value of *k* for a STG can be derived using the procedure shown in Fig.2. For example, for the STG of Fig. 1 the **2-out-of-5** code is necessary for the state assignment. The $\mathbf{P(S_i \rightarrow S_j)}$ values for the STG of Fig.1 are shown in Table 1.

It can be seen from Table 1 that the state machine always moves to state A from state E, to state G from state F, and to state A **from** state G. These states are assigned code words belonging to 2-out-of-5 code such that the minimum Hamming distance between these code words is 2. A possible assignment for states A, E, F and G is shown in column 1 of Table 2.

P(A→B) = 0.5	**P(D→E)** = 0.5
P(A→C) = 0.5	**P(D→G)** = 0.5
P(B→D) = 0.5	**P(E→A)** = 1
P(B→F) = 0.5	**P(F→G)** = 1
P(C→D) = 0.5	**P(G→A)** = 1
P(C→F) = 0.5	

Table 1. State transition probability

Next states **B**, **C**, and **D** are assigned code words from the remaining code words of the **2-out-of-5** code. These code words are assigned such that transitions from the newly assigned states to those already assigned or vice-versa, require change in the minimum number of state bits; however, this may not always be possible. Column 2 of Table 2 shows a possible code word assignment for states for B, C, and D. Note that this assignment ensures minimum transitions from A to B, A to C, C to D and C to F, but four bits change during transition **from** B to D.

This state assignment may be refined to ensure that all transitions have minimum bit changes. The refinement process involves selective changing and/or exchanging assigned code words to the states. Column 3 of Table 2 shows the refined state assignment obtained by changing the code words for states A, B and C in column 2 to 01010, 10010 and 01100 respectively. It should be pointed out that in this assignment all valid state transitions have only two state bit changes.

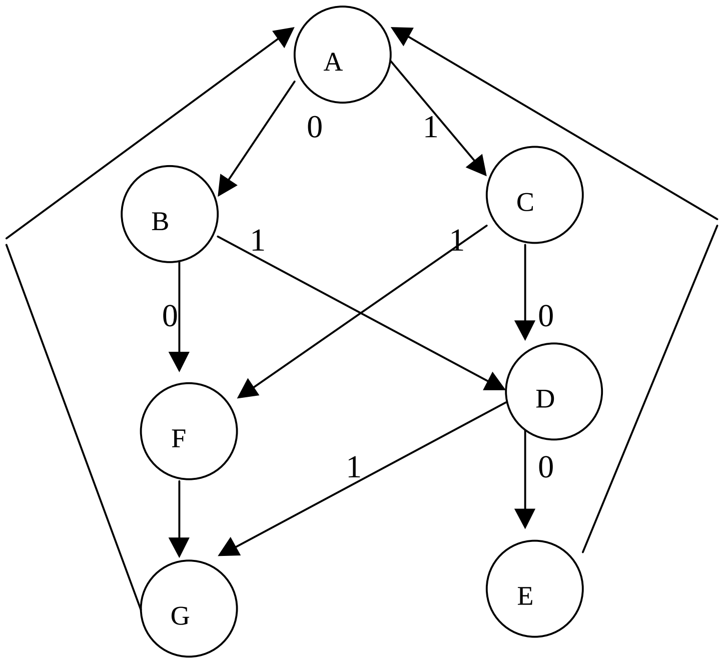


Fig.1 A state transition graph

```
Set $k = 2$;
    |
If $\binom{2k}{k} \geq$ the number of states, exit;
    |
If $\binom{2k+1}{k} \geq$ the number of states, exit;
    |
$k = k + 1$;
    |
Go to step 1;
```

Fig. 2 Selection of *k-out-of-*2*k* (or 2*k* +1) code

The refinement process is heuristic, and cannot guarantee an assignment with minimum state bit changes for all possible state transitions in an STG.

State	**Code Words**		
A	10010	10010	01010
B		10001	10010
C		01010	01100
D		00110	00110
E	00011	00011	00011
F	11000	11000	11000
G	10100	10100	10100

Table 2. State assignment

The step-by-step description of the proposed state assignment technique is given in Fig.3.

Code Word Assignment:

1. Derive Wij = P(Si $\rightarrow$ Sj) for each state in a state transition graph.

2. Assign codes to all state pairs (Si | Sj) with highest Wij values first, such that H(Si,Sj) is minimum. Then assign codes to state pairs with the next highest Wij values , satisfying the minimum distance requirement. Continue this process till all states have been assigned unique codes.

Refinement

The following steps are applied repeatedly

3. Interchange the assigned code words of any two states.

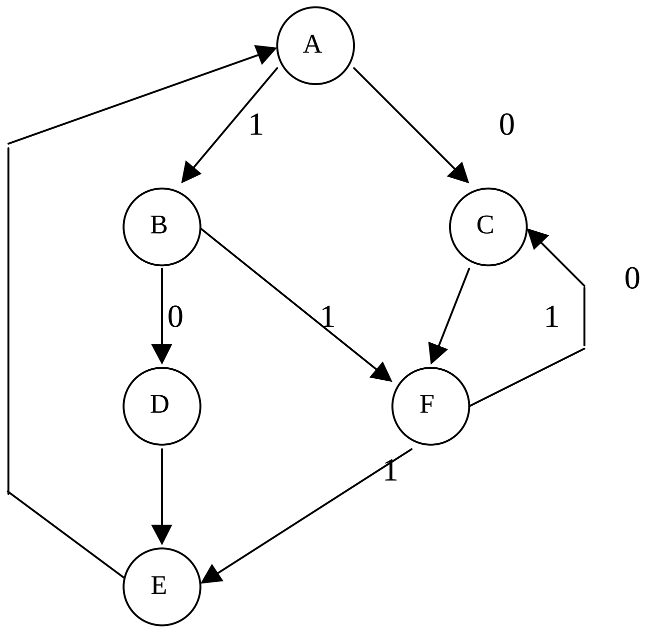
4. Replace the code word assigned to a state by a randomly selected new code word.

Fig. 3. State Assignment Technique

3. Validation of the technique

We illustrate the self-checking capability of the proposed technique by using UC-Berkeley SIS1.3 tool , and applying it to the state transition graph shown in Fig. 4. The state codes for the STG are derived as discussed above, and are also shown in Fig.4. The

ESPRESSO input file (in PLA format) that corresponds to the state assignment of Fig.4 is shown in Fig 5(a), and the resulting logic expressions are shown in Fig. 5(b).



	q3	q2	q1	q0
A =	0	1	1	0
B =	0	0	1	1
C =	0	1	0	1
D =	1	0	1	0
E =	1	1	0	0
F =	1	0	0	1

Fig. 4

```
.i 5
.o 4
.ilb x q3 q2 q1 q0
.ob s3 s2 s1 s0
.p 12
1 0110  0011
0 0110  0101
1 0011  1001
0 0011  1010
1 0101  1001
0 0101  0101
1 1010  1100
0 1010  1100
1 1100  0110
0 1100  0110
1 1001  1100
0 1001  0101
.e
```

(a) PLA input file for ESPRESSO

```
INORDER = x q3 q2 q1 q0;
OUTORDER = s3 s2 s1 s0;
[128] = q3*q2*!q1*!q0;     ←
[129] = x*!q3*q2*q1*!q0;
[130] = !x*!q3*!q2*q1*q0;
[131] = q3*!q2*q1*!q0;
[132] = x*q3*!q2*!q1*q0;
[133] = x*!q3*q2*!q1*q0;
[134] = x*!q3*!q2*q1*q0;
[135] = !x*!q3*q2*q1*!q0;
[136] = !x*q3*!q2*!q1*q0;
[137] = !x*!q3*q2*!q1*q0;
[138] = ![130]*![131]*![132]*![133]*![134];
s3 = ![138];
[140] = ![128]*![131]*![132]*![135]*![136]*![137];
s2 = ![140];
[142] = ![128]*![129]*![130];
s1 = ![142];
[144] = ![129]*![133]*![134]*![135]*![136]*![137];
s0 = ![144];
```

(b) Two-level expressions produced by ESPRESSO

Fig. 5

Next we introduce a fault in one of the nodes (arbitrarily chosen) in Fig. 4(b); assume node 128 is set to logic 1. The PLA file for the faulty machine is then derived from the expressions, and is shown in Fig. 6.

```
.i 5
.o 4
.ilb x q3 q2 q1 q0
.ob s3 s2 s1 s0
.p 10
-0011 1000
-1010 1000
10101 1000
11001 1000
-----   0100
-----   0010
-0101 0001
-0110 0001
10011 0001
01001 0001
.e
```

Fig. 6 PLA file with a fault

The state transitions corresponding to this PLA file are as follows:

		x =0	x =1
	q3 q2 q1 q0	s3 s2 s1 s0	s3 s2 s1 s0
A (0110)	0111	0111	
B (0011)	1110	1111	
C (0101)	0111	1111	
D (1010)	1110	1110	
E (1100)	A (0110)	A (0110)	
F (1001)	0111	1110	

Notice that all next states except for state E, are non-code words; state E makes the correct next state transitions in spite of the assumed fault. A self- checking checker for

2-out-of-4 code will detect an erroneous state transition on–line. All faults except those create bidirectional errors at the outputs of the flip-flops can be detected by the proposed technique.

4.Conclusion

A heuristic state assignment technique that results in minimum transitions in state bits in finite state machines implemented in CMOS is presented in this paper. It also guarantees on-line detection of a single bit error or unidirectional multibit errors in a state code. Although the proposed technique uses more state bits than that can be obtained by using conventional state assignment techniques, the advantages of state machines designed using the technique are lower dynamic power consumption and on-line checking capability.

References

1. G. De Micheli, R.K. Brayton and A.Sangiovanni-Vincentelli, " Optimal state assignment of finite state machines", IEEE Trans. CAD, vol.4, 1985, pp. 269-285.

2. T. Villa and A. Sangiovanni-Vincentelli, " NOVA: State assignment of finite state machines for optimal two-level logic implementations", ", IEEE Trans. CAD, vol.9, 1990, pp. 905-924

3. S. Yang and M. Ciesielski, "On the relationship between input encoding and logic minimization", Proc. 23rd Hawaii Int. Conf. on System Sciences, January 1990, pp.377-386.

4. S.Devadas, H.-K T. Ma , R. Newton and A. Sangiovanni-Vincentelli, "MUSTANG: State assignment of finite state machines targeting multilevel logic implementations", IEEE Trans. CAD, vol.7, 1988, pp. 1290-1300

5 B.Lin and A.R. Newton, " Synthesis of multiple-level logic from symbolic high-level

description languages", International Conf. on Very Large Scale Integration, August 1989, pp.187-196.

6. N. Weste and K. Eshragian, *Principles of CMOS VLSI Design*, Addision-Wesley, 1994.

7. A. Chandrakashan, T. Sheng, and R. Broderson, "Low power CMOS digital design", IEEE Journal of Solid-State Circuits, vol.27, no.4, 1987, pp.473-484.

8. K. Roy and S. Prasad, "SYCLOP: Synthesis of CMOS logic for low power applications", Proc. IEEE Conf. on Computer Design, 1992, pp. 464-467.

9. K. Roy and S. Prasad, " Circuit activity based logic synthesis for low power reliable operations", IEEE Trans. on Very Large Scale Integration Systems, vol.1, 1993, pp.503-513.

10. S. Washabaugh, P. Franzon, H. Troy Nagle, "SABSA: Switching –activity-based state assignment", International Jour. of High Speed Electronics and Systems, vol.5, no.2, 1994, pp.203-212.

11. P.K. Lala, *Digital Circuit Testing and Testability*, Academic Press, 1997.

Parag K. Lala

Parag K. Lala is a Professor in the Department of Electrical Engineering , University of South Florida. He received a Ph.D degree from the City University of London, and a D.Sc(Eng.) degree from the University of London. He is the author/co-author of more than eighty-five papers. He is also the author of four books: ***Fault-Tolerant & Fault-Testable Hardware Design*** (Prentice Hall, 1985), ***Digitl System Design using PLDs*** (Prentice Hall,1990), ***Practical Digital Logic Design and Testing*** (Prentice Hall,1996), and ***Digital Circuit Testing and Testability*** (Academic Press, 1997). He is a Fellow of the IEE, and a Senior Member of the IEEE.