An approach for designing on-line testable state machines

P. K. Lala
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701

M. G. Karpovsky
Department of Electrical and Computer Engineering
Boston University
Boston, MA 02215

1. Introduction

Synthesis of state machines have attracted the attention of researchers for more than two decades. Several state assignment techniques that result in efficient implementation of the next state logic have been developed [1-3]. However, none of these addresses the testability of an implemented machine. A popular approach for enhancing the testability of a state machine is to modify its design by using the scan path technique e.g. LSSD [4]. A number of techniques for synthesizing testable state machine directly from their specifications have also been proposed [5-6]. The goal of these techniques is to make state machines fully testable for all single stuck-at faults, and to derive test sequences for them by using test generation techniques for combinational circuits. The major problem with these techniques as well as the LSSD is that they can only increase the off-line testability i.e. they simplify the detection of permanent faults. Recent studies show that transient faults will be the dominant faults in systems designed using deep submicron technology [7]. Traditional off-line testability approaches cannot guarantee the detection of transient faults; they have to be detected during normal operation of a circuit. This in turn requires that circuits be designed so that they have built-in mechanisms for on-line fault detection. Over the years some techniques have been proposed for designing state machines with on-line fault detection capability [8-10]. However, these techniques concentrate mainly on the post design modifications rather than designing machines that are on-line testable by design. This paper proposes a new approach for designing state machines that have built-in capability for enhancing on-line and off-line testability.

2. Proposed approach:

The architecture of the proposed state machine is shown in Fig.1. It is assumed that that the states of a machine are encoded using 1-hot code. The code for the initial state is first loaded into Reg. R by using the system clock. The next state corresponding to an input is determined by the next state logic. The next state logic is designed using transmission gates (TX gates) and tri-state buffers only. The next state is stored in Reg. Q. However, the outputs of the next state logic are transferred into Q by using the preset and the reset inputs associated with each bit of Q, instead of using the system clock. In other words, Reg. Q operates in asynchronous mode.

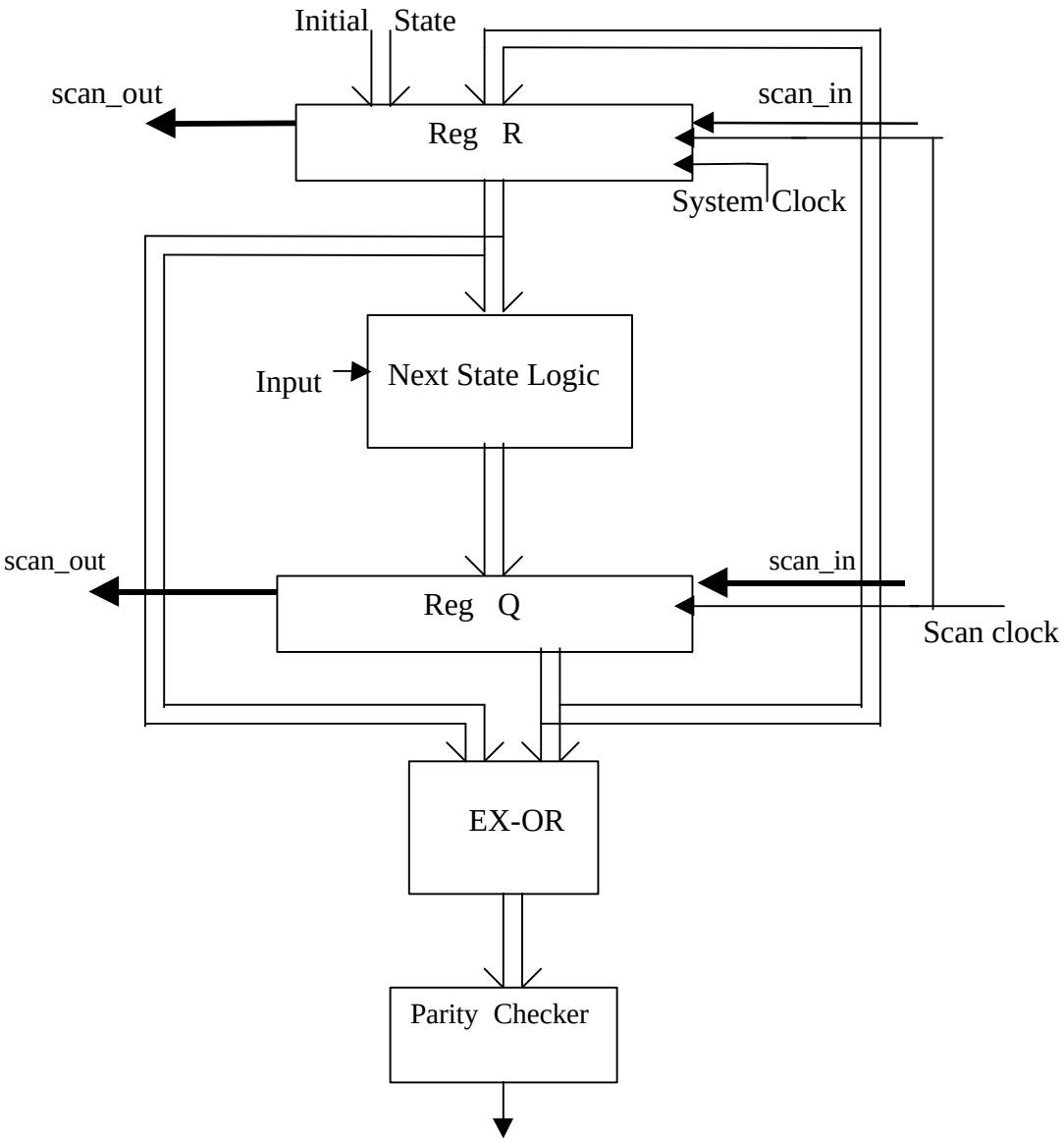

Fig. 1 Proposed architecture of state machines

The contents of Regs. R and Q are EX-ORed, and the result is checked for even parity. In the absence of any circuit fault, the outputs of the EX-OR block will either be all 0s or contain two 1s. An all 0s output indicates that the current state and the next are identical. On the other hand, if the next state is different from the current state, the EX-OR outputs will have exactly two 1's. Any other output pattern will indicate the presence of a fault in

the state machine. Thus the proposed design approach guarantees the on-line testability of a state machine.

As can be seen in Fig.1 both reg. R and Q have scan-in/scan-out features. Thus, any pattern can be serially loaded into R to test the TX gates in the next state logic block. The output of the block is loaded into Q by using the individual preset and reset inputs of the register, and then scanned out. Also, appropriate input patterns can be loaded into R and Q by using the scan clock to test the EX-OR gates and the parity checker circuit. Thus, the off-line testing of state machines based on the model of Fig.1 is considerably simplified.

The next state logic is implemented by using TX gates and tri-state buffers only. Since each state is 1-hot encoded, the state transitions can be implemented in reg. Q by deactivating the state bit corresponding to the current state, and at the same time activating the state bit corresponding to the next state. If the current and the next state are identical i.e. there is no state change produced by an input, the state bits in Q are not deactivated.

Let us illustrate the proposed approach by designing the state machine(an MCNC benchmark circuit) specified in Table 1. We consider only the next states, the inclusion of the output logic is straightforward.

	$\bar{x}_1 \bar{x}_2$	$\bar{x}_1 x_2$	$x_1 x_2$	$x_1 \bar{x}_2$
A	A	–	–	B
B	A	–	C	B
C	–	D	C	B
D	E	D	C	–
E	E	D	–	F
F	E	–	G	F
G	–	H	G	F
H	I	H	G	–
I	I	H	–	–

Table 1. State machine

The states are encoded using 1-out-of-9 code as shown below:

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
A	1	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	0
C	0	0	1	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0

F	0	0	0	0	0	1	0	0	0
G	0	0	0	0	0	0	1	0	0
H	0	0	0	0	0	0	0	1	0
I	0	0	0	0	0	0	0	0	1

The implementation of reg Q is shown in Fig.2 . During normal operation the content of each register bit is determined by the values assigned to its *p* (preset) and *r* (reset) inputs.
The contents of Q are transferred to R by the system clock.

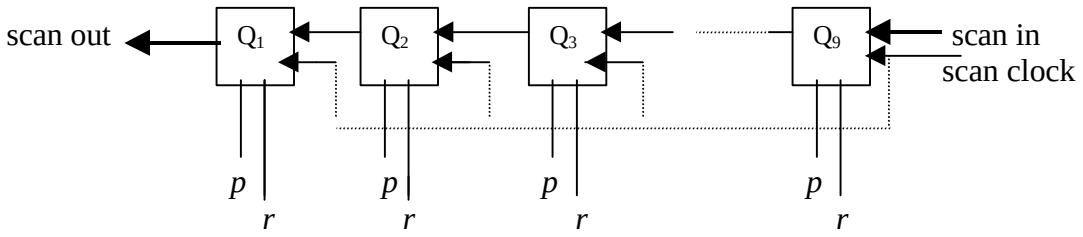

Fig. 2 Structure of Reg.Q

As stated previously state the next state corresponding to a current state and an input is determined by setting and resetting the appropriate state bits in reg. Q . Table 2 shows the setting and resetting of bits in reg.Q for implementing the state machine of Table 1.

	$\bar{x}_1 \bar{x}_2$	$\bar{x}_1 x_2$	$x_1 x_2$	$x_1 \bar{x}_2$
A	pQ_1	–	–	$rQ_1\ pQ_2$
B	$rQ_2\ pQ_1$	–	$rQ_2\ pQ_3$	pQ_2
C	–	$rQ_3\ pQ_4$	pQ_3	$rQ_3\ pQ_2$
D	$rQ_4\ pQ_5$	pQ_4	$rQ_4\ pQ_3$	–
E	pQ_5	$rQ_5 pQ_4$	–	$rQ_5\ pQ_6$
F	$rQ_6\ pQ_5$	–	$rQ_6\ pQ_7$	pQ_6
G	–	$rQ_7\ pQ_8$	pQ_7	$rQ_7\ pQ_6$
H	$rQ_8\ pQ_9$	pQ_8	$rQ_8\ pQ_7$	–
I	pQ_9	$rQ_9\ pQ_8$	–	–

Table 2 Presetting and resetting of bits in Reg.Q

The first row in Table 2 ,for example, indicates that for current state A and input $\bar{x}_1 \bar{x}_2$, bit Q_1 in Fig.2 is preset, thus keeping $Q_1\ Q_2\ Q_3$ …..Q_9 = 1 0 0….. 0 i.e. there is no change in the state bits bits. When the input is $\bar{x}_1 x_2$ the next state is B. This can be implemented by resetting Q_1 and

presetting Q_2. For inputs $\bar{x}_1 x_2$ and $x_1 x_2$, the next states are unspecified, therefore none of the state bits is preset or reset. The circuit for generating an enable signal corresponding to each input combination is shown in Fig. 3.

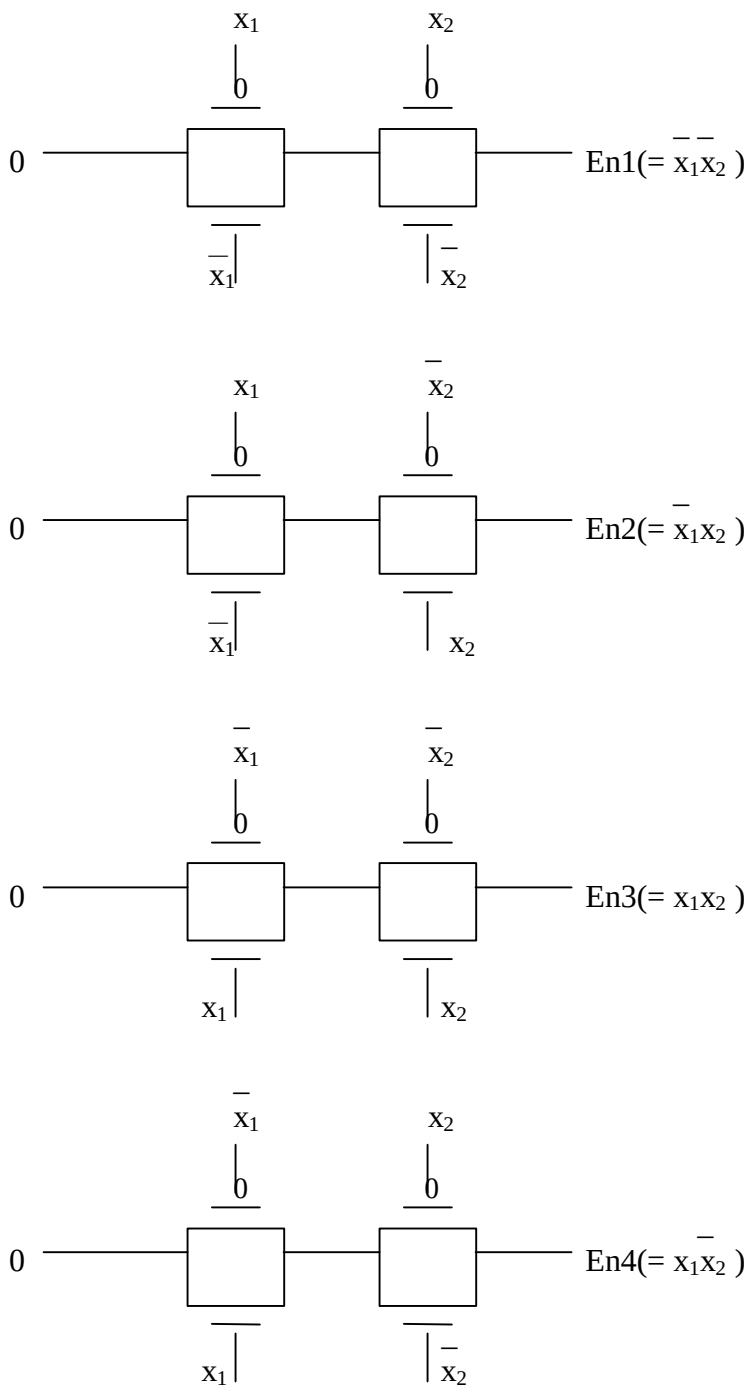

Fig. 3 Generation of enable signals

Fig. 4 shows the circuit needed to generate set and reset signals for activating transitions from state B for various input combinations. Current state B is identified by R_2 = 1 and $R_1..R_3$ ….R_9 = 0 0….0. Therefore in Fig. 4, R_2 (= 1) is used to enable the tri-state buffer. The next state logic for implementing other state transitions can be derived in a similar manner. Note that at any time only one enable signal is activated. Also only tri-state buffer is enabled at a time, thus the outputs of the TX gates can be connected together if they drive the same preset and reset inputs of the state bits in Q register.

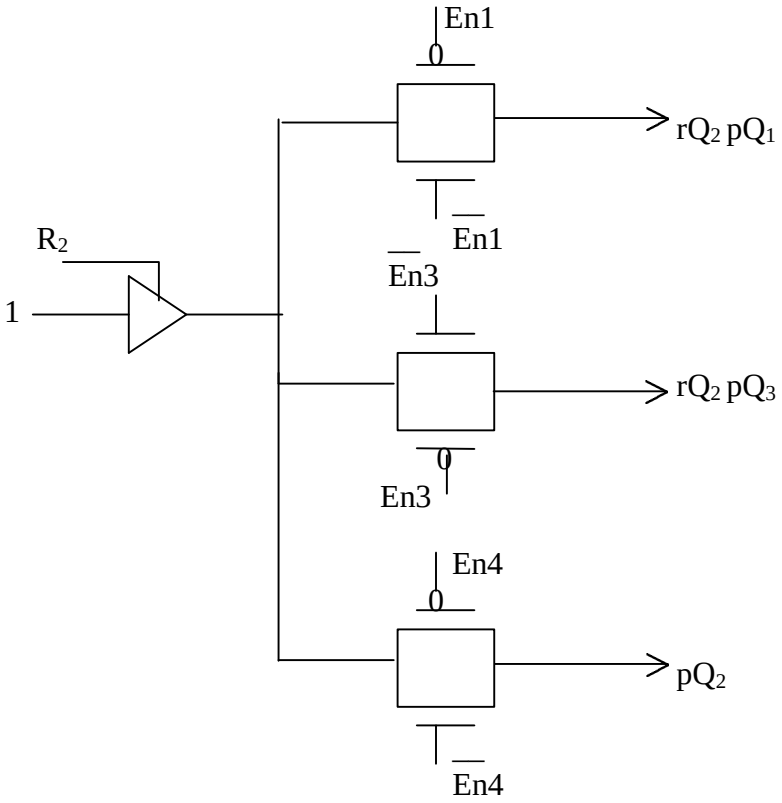

Fig. 4 Transitions from state B

The proposed technique has been applied to several MCNC benchmarks circuits. The number of registers, transmission gates and buffers needed to implement each circuit are shown in Table 3. As mentioned previously some additional circuit are also needed to generate appropriate enable signals (corresponding to input combinations) for the transmission gates.

Circuit	Length of reg. R and Q	# of TX gates	# of tri-state buffers	# of EX-OR gates
bbara	10-bit	60	10	19
bbsse	16-bit	56	16	31
bbtas	6-bit	24	6	11
beecount	7-bit	28	7	13
cse	16-bit	91	16	31
dk14	7-bit	56	7	13
dk27	7-bit	14	7	13
ex2	19-bit	72	19	37
ex5	9-bit	32	9	17
modulo12	12-bit	12	12	23
opus	10-bit	15	10	19

Table 4. Components needed for implementing MCNC benchmark circuits

3. Conclusion:

A new approach for designing state machines that have built-in on-line and off-line testability has been proposed. The resulting machines have scan_in/scan_out capability that allows the testing of the next state logic composed of tri-state buffers and TX gates only. The number of buffers required is equal to the number of states in a machine, and the number of TX gates is equal to the number of specified next states. The on-line testing capability is achieved by EX-ORing the outputs of the two registers in a machine, and checking for even parity at the outputs of the EX-OR gates. The number of EX-OR gates equals the length of the registers, plus those needed for parity checking.

4. References:

1.T.Villa and A. Sangiovanni-Vincenetelli, “NOVA: State Assignment of finite state machines for optimal two-level logic implementations”, IEEE Trans.on Computers, Sept., 1990, pp.1326-1334.

2. S.Devadas, B.Ma, R.Newton, and A. Sangiovanni-Vincenetelli, “MUSTANG: State assignment of finite state machines targeting multi-level logic implementations”, IEEE Trans.on Computer-Aided Design, Dec.1988, pp.

3. B.Lin and R.Newton, “Synthesis of multiple level logic from symbolic high-level description languages”, Proc. VLSI’89 Conf., Munich, Germany, 1989, pp.

4. E.B.Eichelberger and T.W.Williams, “A logic system structure for LSI testability”, Pro.ACM/IEEE Design Automation Conf., 1978, pp.462-468.

5. S.Devadas and K.Kreutzer, “Boolean minimization and algebraic factorization procedures for fully testable sequential machines “ IEEE Conf on CAD, 1989, pp.208-211

6. S. Devadas et.al, “A synthesis and optimization procedure for fully and easily testable sequential machines”, IEEE Trans. on CAD, Oct.1989, pp.110-1107.

7. M. Nicolaidis, “Time redundancy based soft error tolerance to rescue nanometer technologies”, IEEE VLSI Test Symposium, April 1999, pp.86-94.

8. N.K. Jha and S.J.Wang, “Design and synthesis of self-checking VLSI circuits and systems”, IEEE Trans. on CAD, June 1993, pp. 878-887.

9. A. Matrosova and S. Ostanin, “Self-checking FSM design with observing only FSM outputs”, Proc.6th IEEE On-line Testing Workshop, Palma de Mallorca, Spain, 2000, pp.153-154.