SCAN SYNTHESIS FOR ONE-HOT SIGNALS

Subhasish Mitra, LaNae J. Avra and Edward J. McCluskey

Center for Reliable Computing
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

ABSTRACT

Tri-state buses and pass transistor logic are used in many complex applications to achieve high performance and small area. Such circuits often contain logic requiring one-hot signals. In a scan-based design, one-hot values on these signals may not be maintained during the scan-in and scan-out operations. Also, the presence of faults, the existence of don't care conditions and the use of random patterns for testing the circuit in a scan or BIST environment may lead to non-one-hot values on these one-hot signals, resulting in abnormal circuit behavior and possible circuit damage. In this paper, we present new techniques for synthesizing scan-based designs so that one-hot values are maintained on the one-hot signals during all modes of operation. One of our synthesis techniques often generates designs with no area overhead — the designs are smaller than those that do not ensure safe scan operation. In addition, we propose a scan path design that has no performance overhead during the normal mode of operation and ensures that only valid states appear on the bistables during test mode, thus guaranteeing safe scan operations.

1. INTRODUCTION

Among the different design for testability (DFT) techniques available today, scan path based methods [McCluskey 86] are probably the most basic and most widely used. In a scan-path based design, the circuit has two operating modes: normal functional mode, and scan mode, during which the circuit bistables are interconnected into a shift register. In the scan mode, it is possible to shift an arbitrary test pattern into the bistables. By returning the circuit to normal mode for one clock period, the outputs of the combinational circuitry are stored in the bistables. If the circuit is then placed into scan mode, it is possible to shift out the contents of the bistables and compare these contents with the correct response. Thus, a sequential circuit is transformed into a combinational one during testing thereby making test generation simpler.

Many circuits contain logic that is controlled by one-out-of-*n* (one-hot) input signals. A typical example is an *n*-to-*1* multiplexer implemented with *n* transmission gates, each enabled by a different control signal. Tri-state buses and logic implemented with pass transistors are typically used in complex microprocessors to achieve high performance in a small area. Examples of this design style have been reported in [Shoji 86], [Suzuki 93], [Yano 90], [Yano 94]. Unfortunately, the presence of tristate buses and pass transistor logic poses a problem in a scan or BIST design. At some point during the testing process, the bistables may contain a state that does not occur during functional operation, and this can result in non-one-hot values on the one-hot signals. Bistables may contain invalid states while patterns are scanned in and out or when pseudo-random patterns are applied to the circuit during BIST operation. This can result in abnormal and unpredictable circuit behavior. In designs containing tristate buses, the presence of non-one-hot values in the tristate control inputs may cause circuit damage.

The goal of our synthesis techniques is to generate one-hot signals that are safe for scan and BIST operations. For many cases, one of our techniques generates designs with less area than the conventional technique of synthesizing such designs without considering safety during scan or BIST operations. Section 2 of this paper surveys some existing techniques to handle one-hot bistables in scan path-based designs. In Sec. 3, we describe our synthesis schemes for generating one-hot signals. The experimental results for the area and the delay values of the designs synthesized with our schemes (Sec. 3.1 and 3.2) are reported in Sec. 4. Section 5 discusses the testability of the logic generated by the scheme discussed in Sec. 3.2. Section 6 describes a scan path design to ensure safe scan in delay critical circuits. Finally, we conclude in Sec. 7.

2. Existing Approaches for Handling One-Hot Bistables

One-hot signals are often generated directly from the bistable outputs of a design. In that case, the bistables store one-hot values. Such bistables are called *one-hot bistables*. Several approaches have been proposed for handling one-hot bistables in a scan path based design. The simplest is to remove such bistables from the scan path, resulting in a partial-scan design [Cheng 90], [Lee 90], [Abramovici 91]. In that case, sequential test generation techniques must be used to generate test vectors. Another approach is to gate the output of the one-hot bistables during scan with a *scan* signal, resulting in a particular one-hot value enforced on the bistable outputs irrespective of their contents. Only one of the bistable outputs should be or-ed with the *scan* signal while the other latch outputs should be and-ed with *(scan)´* signal. When *scan* is 1 during

scanning, a particular one-hot value is enforced. This technique ensures safety during scan-in and scan-out operations, but non one-hot values may appear on the one-hot signals if pseudo-random patterns are loaded into the bistables. A generalization of this approach is to enforce a particular one-hot value on the one-hot signals throughout the test mode of operation by using a special signal. Although this solution ensures safety during scan-in and scan-out operations and also when pseudo-random patterns are used to test the circuit, the fault coverage can fall drastically because the logic may not be sufficiently tested since the enforced one-hot value does not change during testing.

In [Levitt 95], a special tristate enable signal is used for all multiplexer selects that are not proven to be exclusive under all possible flip-flop states. The L3 latch based Level Sensitive Scan Design (LSSD) technique presented in [Das Gupta 81] can also be used to take care of the one-hot latch problem. The L3 latch is used to break the path from an LSSD to a non-LSSD network so that the non-LSSD does not interfere with the testing of the LSSD logic. With an L3 latch-based scheme, the desired pattern is first scanned into the L1-L2 latches. Next, the pattern is loaded into the L3 latches. Thus, the non-LSSD logic receives the final scanned-in value (a valid state) and not the intermediate values (invalid states). Neither of these schemes ensure safety when pseudo-random patterns are used to test the circuit under test (CUT) during test mode because the final scanned-in value may violate the one-hot condition (because it is a random pattern). The encoder/decoder based scheme for solving the one-hot latch problem discussed in [Pateras 95] overcomes these problems. However, the scheme has limitations due to the fact that it assumes that one-hot signals are produced directly from the one-hot latches. Moreover, the scheme requires additional latches when the number of one-hot signals is not a power of 2, and to eliminate fault coverage losses. We consider a more general problem, where any set of signal lines (not necessarily the bistable outputs) can contain one-hot values. Thus, we target a more general class of circuits. Our aim is to synthesize these circuits such that one-hot values are maintained regardless of the state of the bistables. Our schemes guarantee safety during the test mode of operation not only while the patterns are scanned in or out but also when pseudo-random patterns are used to test the circuit. Thus our schemes are also applicable in pseudo-random BIST environments such as when pseudo-random patterns are scanned into the system bistables (e.g., STUMPS), or when the system bistables are reconfigured to generate pseudo-random patterns for the BIST operation (e.g., circular BIST).

3. Proposed Synthesis Technique

In this section, we describe two approaches for synthesizing designs containing one-hot signals. One of the approaches is to insert encoding logic between the logic generating and the logic using the one-hot signals. The other approach assumes that a *finite state machine* (FSM) generates the one-hot signals and synthesizes the FSMs such that the one-hot signals always have one-hot values regardless of the state of the FSM.

3.1 Encoder-Based Technique

The most straightforward and general way to guarantee the one-hot condition on a set of signals is to insert encoding logic between the logic generating the one-hot signals and the logic requiring the one-hot condition as shown in Fig. 1. The encoding logic has *n* inputs corresponding to the *n* one-hot signals and produces *n* one-hot output signals. If the input to the encoder is one-hot, then the encoder output should be the same as the input. If the encoder input is not one-hot, the encoder can map it to any one-hot value. A typical example of such encoding logic is a *priority encoder* which produces a *'1'* on output *i* and *'0'* on the remaining outputs if all inputs *1 .. i-1* are *'0'*, input *i* is *'1'*. Thus, the priority encoder ensures that, as long as it is fault-free, its outputs are one-hot. Table 1 shows the truth table for an eight input priority encoder. The response of the encoder to 00...0 may vary — if 00..0 (*none hot*) is a valid pattern then the response to 00...0 is 00...0. Otherwise, it has to be mapped to some one-hot value. Truth Table 1 assumes that the none-hot condition is not valid. The greatest advantage of such a scheme is that it is independent of any assumption about the general structure of the circuit that produces one-hot signals. Moreover, it ensures one-hot values on the signal lines both in the normal and in the test mode of operation.

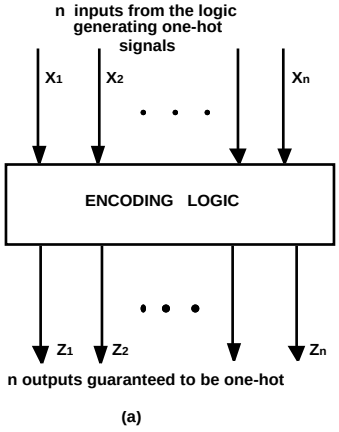

Table 1

X_1X_2 ....X_8	Z_1Z_2 ..... Z_8
1- - - - - - -	10000000
01- - - - - -	01000000
001- - - - -	00100000
0001- - - -	00010000
00001- - -	00001000
000001- -	00000100
0000001-	00000010
0000000-	00000001

(b)

Figure 1. Encoder based scheme to guarantee one-hot values (a) block diagram (b) Truth table for 8-bit priority encoder.

However, this scheme has an area and delay overhead. Section 4 reports the area and delay values after synthesizing priority encoders with different input sizes using *sis* [Sentovich 92]. As discussed earlier, to set a particular output bit *i* to 1, an *n*-bit priority encoder has to check that all the input bits 0 ... *i-1* are 0 and input bit *i* is 1. This checking procedure is the main source of delay in the priority encoder. We are currently investigating schemes to implement the priority encoder with minimum delay overhead. This delay problem has a close resemblance

to the inherent delay in generating the final carry (*carry out*) in ripple-carry adders. Various adder designs, such as carry-lookahead, carry-skip, and carry-select, have been proposed to speed up integer addition [Hennessy 95]. We can use similar techniques to reduce the delay associated with priority encoders.

3.2 FSM Synthesis for one-hot signals

The general model of a *finite state machine* (FSM) is shown in Fig. 2. As mentioned in [McCluskey 86], a finite state machine has next state logic and output logic associated with it. The next state logic generates the *next state* of the finite state machine and the output logic generates the outputs dependent on the *current state* and the current input. We assume that there are two types of outputs generated by the output logic: one-hot outputs and non-one-hot outputs. We must ensure that the one-hot outputs always contain one-hot values regardless of the state of the finite state machine. Non-one-hot values could arise on the one-hot outputs of the finite state machine due to one or more of the following reasons :

(i) *Don't care* conditions in the output logic generating the one-hot outputs. A non-one-hot value may occur on the one-hot outputs if the FSM output logic has don't-care conditions and the FSM reaches an invalid state. Since this invalid state is a don't care input for the output logic, two outputs corresponding to one-hot signals may simultaneously be set to 1. Invalid states may appear in the FSM bistables:

(a) During the scan-in and scan-out operations,

(b) When pseudo-random test techniques are used,

(c) When there are faults in the next-state logic.

(ii) Faults in the output logic.

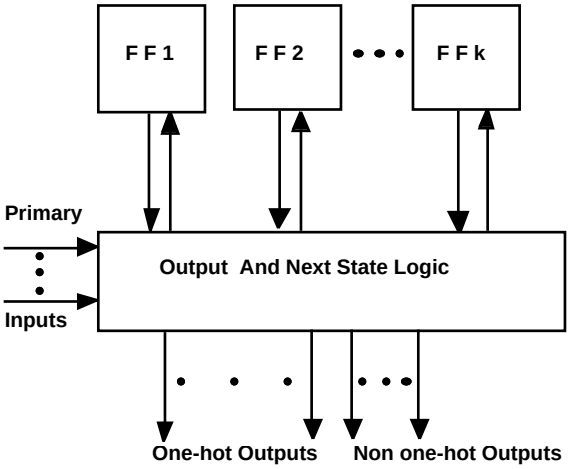

Figure 2. General structure of FSM that generates one-hot signals.

To guarantee one-hot values on the one-hot outputs, we alter the original FSM specification to one where the outputs that are supposed to be one-hot are fully encoded using the minimum number of bits. We call these outputs *encoded one-hot outputs.* Next, we synthesize the altered FSM and add a decoder whose inputs are the encoded one-hot outputs and whose outputs are the one-hot signals. The rationale behind encoding the one-hot outputs is that, irrespective of any fault in the output or next state logic or presence of any invalid state in the flip-flops of the FSM, we can always decode the encoded output into some valid one-hot signal as long as the decoder circuitry is fault-free. Thus, one-hot values are guaranteed on one-hot signals not only during the scan-in/scan-out operations but also during the normal mode of operation and also when random patterns are used to test the circuit in a scan or a BIST environment. Figure 3 shows our modification to the FSM of Fig. 2, ensuring one-hot values on the one-hot output signals. The flow of the FSM synthesis scheme is given in Fig. 4.

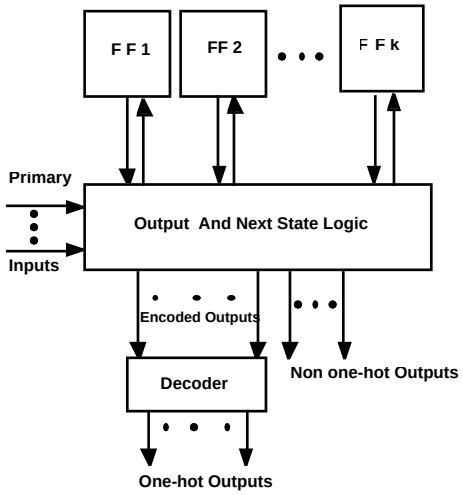

Figure 3. FSM with guaranteed one-hot outputs.

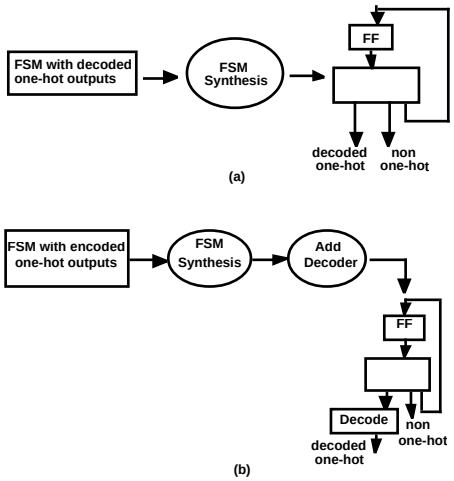

Figure 4. FSM synthesis: (a) typical flow (b) modified flow ensures one-hot values.

Let us consider the example FSM specification, FSM-1, in Table 2 that we fabricated to illustrate our FSM synthesis scheme. FSM-1 has 4 primary inputs, 5 states

and 7 outputs. The first 4 of the 7 outputs are one-hot. The first step in our synthesis process is to alter the specification of FSM-1 by encoding the one-hot outputs as shown in Table 3. The first four outputs of FSM-1 (the one-hot outputs) are encoded using two bits. The truth table for the output decoder is shown in Table 4. Under fault-free conditions, the decoder produces one-hot values on its outputs.

Table 2. Specification of an example FSM with one-hot outputs (FSM-1).

Input	Present State	Next State	Output
0100	state 0	state 1	1000 101
1100	state 1	state 2	0100 000
0001	state 2	state 0	0010 110
0100	state 2	state 3	0100 010
0100	state 3	state 4	0001 011
1111	state 2	state 1	1000 111
0111	state 4	state 3	0001 001

Table 3. Modified specification of FSM-1 with encoded one-hot outputs.

Input	Present State	Next State	Output
0100	state 0	state 1	00 101
1100	state 1	state 2	01 000
0001	state 2	state 0	10 110
0100	state 2	state 3	01 010
0100	state 3	state 4	11 011
1111	state 2	state 1	00 111
0111	state 4	state 3	11 001

We could encode the one-hot outputs such that the logic generated during FSM synthesis is minimized. This adds a degree of freedom which can be effectively utilized to generate a minimal-area implementation of the FSM. For example, if we encoded 1000 as 11, 0100 as 00, 0010 as 10 and 0001 as 01, then we would have to synthesize only three distinct output functions in the modified FSM specification with encoded one-hot outputs. We have developed an algorithm that encodes the one-hot signals such that the total number of output functions to be synthesized is minimum [Mitra 97]. Note, however, that changing the encoding of the one-hot signals changes the decoder specification but does not change the size of decoder implementation in this case.

Table 4. Decoder that generates one-hot outputs.

Input	Output
00	1000
01	0100
10	0010
11	0001

FSM-1 is a specific case where the number of one-hot signals is a power of 2. When the number of one-hot signals is not a power of 2, we can use our synthesis scheme with minor modifications to the decoder specification. Suppose we have m one-hot signals which have been encoded using $n = \lceil \log_2 m \rceil$. Not all the 2^n combinations will appear on the output of the FSM. Hence, we map the unused combinations to particular decoded values in order to minimize the size of the decoder. Table 5 shows the truth table for an efficient decoder for five one-hot outputs.

Table 5. Decoder for five one-hot outputs.

Input	Output
000	10000
-01	01000
-10	00100
-11	00010
100	00001

While we have explained our technique for generating one-hot signals in the context of FSM synthesis, it can be applied to any combinational or sequential logic that generates one-hot signals. For FSMs, our scheme is a straight-forward solution that can be used with existing FSM synthesis techniques and tools.

4. Experimental Results

In this section, we first report the area and delay of the priority encoders of different sizes, synthesized using *sis* [Sentovich 92] that are used to guarantee one-hot values on one-hot signals, as discussed in Sec. 3.1. The underlying library used for technology mapping is the LSI Logic g10p library [LSI 96]. These area and delay values provide insight into the area and delay overheads incurred when the encoder-based scheme, presented in Section 3.1, is used to handle one-hot signals. Table 6 reports the area and delay values obtained after optimizing the priority encoders for area. Table 7 reports the area and delay values obtained after optimizing the priority encoders for delay. Thus, encoders synthesized in Table 7 may be used in performance constrained applications while those synthesized in Table 6 may be used for designs where area is a major constraint.

Table 6. Area optimized priority encoders.

Number of Inputs	Area (LSI cell units)	Delay (ns)
4	20	0.29
6	44	0.59
8	68	0.94
9	80	1.11
10	92	1.28
11	104	1.45
12	116	1.62
13	128	1.79
14	140	1.96
15	152	2.13
16	164	2.3

Now, we present the area and delay results obtained after applying our FSM-based synthesis scheme for generating one-hot signals (Sec. 3.2) to benchmark designs. For each of the MCNC FSM benchmarks, shown in Table 8, we added 8, 12 and 16 one-hot outputs and used *sis* [Sentovich 92], a logic synthesis tool developed at University of California at Berkeley, to synthesize the FSM benchmarks. We performed multi-level logic minimization using the standard script (called *rugged*) that is supplied with *sis*. For technology mapping, we used the LSI Logic g10p library [LSI 96]. For FSM state encoding, we used two options: one-hot state encoding and binary state encoding (NOVA based). NOVA [Villa 90] encodes the states of an FSM having n states using $\lceil \log_2 n \rceil$ bits (flip-flops). Next, we used our scheme described in Sec. 3.2 to encode the one-hot outputs, synthesized the FSM with encoded one-hot outputs, and connected the encoded outputs to the inputs of a decoder to generate the one-hot signals.

Table 7. Delay optimized priority encoders.

Number of Inputs	Area (LSI cell units)	Delay (ns)
4	20	0.29
6	50	0.41
8	87	0.61
9	104	0.61
10	127	0.63
11	150	0.66
12	178	0.68
13	208	0.70
14	240	0.69
15	271	0.72
16	310	0.86

We compared the area and delay of the designs generated by our scheme to that of the general scheme of synthesizing FSMs with one-hot outputs without taking special care to guarantee one-hot values on one-hot outputs during scan (as shown in Fig. 4a). Tables 9 and 10 show the area and the delay results, respectively, for some of the MCNC FSM benchmark circuits for which we performed the state encoding using the one-hot state encoding option.

Table 8. MCNC logic synthesis FSM benchmark circuits.

Name	# Inputs	# Outputs	# States
bbara	4	2	10
bbtas	2	2	6
beecount	3	4	7
dk14	3	5	7
dk16	2	3	27
dk512	1	3	15
donfile	2	1	24
ex3	2	2	10
ex6	5	8	8
ex7	2	2	10

Tables 11 and 12 report the area and delay results, respectively, for the FSM benchmarks using binary (NOVA based) state encoding. For both one-hot and binary state encoding, our synthesis scheme not only ensures a safe scan operation, it often results in the smallest area implementations. This is due to the fact that, with our scheme, the FSM has to generate a smaller number of outputs (since the one-hot outputs are encoded). So, *sis* can perform different optimizations more efficiently (these optimizations are heuristic) and hence generate better designs. It may be noted that the area results of the FSMs synthesized using our approach to handle one-hot signals include the area of the decoder that is required to generate the one-hot outputs from the encoded one-hot signals.

5. Testability Issues

In this section, we discuss some issues related to the testability of the logic generated by the technique reported in Sec. 3.2. The following lemma gives some insight into the testability of the logic.

Lemma 1: If the number of one-hot signals is a power of 2, the logic generated by our scheme (logic generating the encoded one-hot outputs followed by the decoder) is *irredundant* [Abramovici 90] as long as the logic generating the encoded one-hot outputs (output and next state logic in Fig. 3) and the decoding logic are themselves irredundant.

Proof: Let us consider any single stuck-at fault in the logic generated by the scheme described in Sec. 3.2. The fault can be either in the logic generating the encoded one-hot signals (we call it L) or in the decoding logic. If the fault is in the decoding logic, then there exists a vector v which, when applied to the decoding logic, detects the fault because the decoding logic is irredundant. Since the number of one-hot signals is a power of *2*, each encoded one-hot signal corresponds to one and only one one-hot signal; hence, there always exists a vector u which, when applied to the input of L, generates v at the encoded one-hot outputs (which is the input of the decoding logic) — thus the fault is detected.

Now, let us consider a single stuck-at fault, f, in L, the logic generating the encoded one-hot signals. Since L is irredundant, there exists an input vector v such that $L(v) \neq L_f(v)$, where $L_f(v)$ is the response of L in presence of the single stuck-at fault f. Since the number of one-hot signals is a power of 2, we have a *complete decoder* at the output of the encoded one-hot logic. As a result, the output of the decoder in response to $L(v)$ will be different from the output of the decoder in response to $L_f(v)$. Hence, the logic generated by our scheme is fully testable with respect to single stuck-at faults when the number of one-hot signals is a power of 2.

When the number of one-hot signals is not a power of 2, we have an incomplete decoder, i.e., the number of outputs of the decoder is less than 2^n where n is the number of

inputs to the decoder. If there is a single stuck-at fault, f, in L, then there exists some input vector v such that $L(v) \neq L_f(v)$, where $L_f(v)$ is the response of L in the presence of f. But, due to the presence of the incomplete decoder at the output of L, the decoder output in response to $L(v)$ may be the same as the decoder output in response to $L_f(v)$. If this happens for every v that detects f in L, then f is undetectable when the entire network is considered. We can remedy this situation in two ways. The first way is to use decoders with false-data rejection [McCluskey 86]. These decoders do not have any output energized when the input to the decoder does not belong to a valid set of decoder inputs. The second way is to apply standard redundancy removal techniques to remove f from L as described in [DeMicheli 94]. If there is a single stuck-at fault, f, in the decoding logic, then there exists a vector, v, which, when applied to the decoder input, detects f. But since it is an incomplete decoder, there may not exist a vector, u, which, when applied to the input of L, generates v.

Table 9. Area results for FSM benchmarks using one-hot state encoding option.

FSM Name	8 one-hot outputs added			12 one-hot outputs added			16 one-hot outputs added		
	Area * (conventional)	Area # (guaranteed one-hot)	%-age reduction	Area * (conventional)	Area # (guaranteed one-hot)	%-age reduction	Area * (conventional)	Area # (guaranteed one-hot)	% -age reduction
bbara	666	557	16 %	592	617	- 4 %	953	745	22 %
bbtas	211	147	30 %	200	214	- 7 %	221	222	- 0.04 %
beecount	429	352	17 %	421	409	3 %	429	419	2 %
dk16	1620	1026	36 %	1520	1189	22 %	1291	1259	2 %
dk512	375	479	- 21 %	333	499	- 49 %	347	545	- 36 %
donfile	1275	713	44 %	646	751	- 16 %	678	821	- 21 %
ex3	360	328	9 %	385	385	0 %	412	408	1 %
ex6	506	640	- 26 %	500	730	- 46 %	512	700	- 36 %
ex7	271	267	1 %	333	342	- 2 %	369	337	9 %

*. Conventional Synthesis: one-hot requirement is not guaranteed during scan.

#. Our synthesis technique: one-hot requirement is guaranteed during scan; includes area of the decoder.

Table 10. Delay results for FSM benchmarks using one-hot state encoding option.

FSM Name	8 one-hot outputs added			12 one-hot outputs added			16 one-hot outputs added		
	Delay * (conventional)	Delay # (guaranteed one-hot)	%-age difference	Delay* (conventional)	Delay # (guaranteed one-hot)	%-age difference	Delay* (conventional)	Delay # (guaranteed one-hot)	%-age difference
bbara	1.69	2.51	- 48 %	1.42	2.03	- 30 %	1.69	2.46	- 45 %
bbtas	0.79	0.93	- 17 %	0.64	0.59	8 %	0.79	0.93	- 17 %
beecount	1.24	1.81	- 45 %	1.08	1.43	- 32 %	1.24	1.63	- 31 %
dk16	1.83	1.77	3 %	2.76	1.73	37 %	1.91	1.76	8 %
dk512	1.19	2.03	- 70 %	1.41	1.64	-16 %	1.46	1.64	- 12 %
donfile	2.05	2.10	- 2 %	1.36	2.19	- 37 %	1.32	1.94	- 46 %
ex3	1.34	1.21	10 %	1.34	1.22	9 %	1.34	1.37	- 2 %
ex6	1.43	2.31	- 61 %	1.47	2.47	- 68 %	1.43	2.30	- 60 %
ex7	1.02	0.93	9 %	1.07	1.25	- 16 %	1.05	1.27	- 20 %

*. Conventional synthesis: one-hot requirement is not guaranteed during scan.

#. Our synthesis technique: one-hot requirement is guaranteed during scan.

Table 11. Area results for FSM benchmarks using binary encoding (NOVA).

FSM Name	8 one-hot outputs added			12 one-hot outputs added			16 one-hot outputs added		
	Area * (conventional)	Area # (guaranteed one-hot)	%-age reduction	Area * (conventional)	Area # (guaranteed one-hot)	%-age reduction	Area * (conventional)	Area # (guaranteed one-hot)	% -age reduction
bbara	450	367	18 %	339	359	- 5 %	639	478	25 %
bbtas	176	145	21 %	215	181	16 %	192	208	- 8 %
beecount	247	236	4 %	284	319	- 12 %	305	343	- 12 %
dk14	718	621	13 %	728	676	7 %	783	705	10 %
dk512	347	402	- 15 %	449	458	- 2 %	467	492	- 5 %
donfile	690	585	15 %	766	595	22 %	876	626	28 %
ex3	298	312	- 4 %	319	358	- 12 %	364	346	5 %
ex6	638	569	11 %	647	607	6 %	644	665	- 3 %
ex7	274	272	0.07 %	325	336	- 3 %	298	321	- 7 %

*. Conventional synthesis : one-hot requirement is not guaranteed during scan.

#. Our synthesis technique : one-hot requirement is guaranteed during scan. The area of the decoder is also included.

Table 12. Delay results for FSM benchmarks using binary state encoding (NOVA).

FSM Name	8 one-hot outputs added			12 one-hot outputs added			16 one-hot outputs added		
	Delay * (conventional)	Delay # (guaranteed one-hot)	%-age difference	Delay * (conventional)	Delay # (guaranteed one-hot)	%-age difference	Delay * (conventional)	Delay # (guaranteed one-hot)	%-age difference
bbara	3.09	3.23	- 4 %	2.22	3.24	- 45 %	2.41	3.09	- 28 %
bbtas	1.19	1.02	1 %	1.06	0.92	13 %	0.95	1.09	- 14 %
beecount	1.88	2.46	- 30 %	1.40	2.30	- 64 %	1.65	2.32	- 40 %
dk14	4.06	5.74	- 40 %	4.53	5.03	- 11 %	4.41	4.83	- 9 %
dk512	4.25	3.26	24 %	2.28	3.35	- 46 %	2.67	3.54	- 29 %
donfile	4.30	4.10	5 %	3.84	4.01	- 4 %	4.31	3.99	8 %
ex3	2.68	2.22	17 %	1.99	2.68	- 34 %	1.94	2.22	- 14 %
ex6	3.65	3.34	9 %	2.98	2.92	2 %	2.86	4.22	- 47 %
ex7	2.49	2.3	8 %	1.89	2.74	- 44 %	1.57	2.69	- 70 %

*. Conventional synthesis: one-hot requirement is not guaranteed during scan.

#. Our synthesis technique: one-hot requirement is guaranteed during scan.

Thus, *f* is redundant and can be removed from the decoding using standard redundancy removal techniques.

6. A Shadow Register-Based Approach to Ensure Safe Scan

The encoder-based approach reported in Sec. 3.1 is simple to implement but has extra area and delay overhead. The overhead can be reduced by synthesizing the logic appropriately but cannot be fully eliminated. The approach of synthesizing logic with encoded one-hot signals and then decoding them, as reported in Sec. 3.2, has the advantage that one-hot signals can be encoded efficiently so that the area of the logic generating the encoded one-hot signals is minimized. However, a decoder is required to decode the encoded one-hot signals. The results in Sec. 4 show that with existing logic synthesis tools, the area of the entire logic (including the decoder) is less than the area of the logic generating the one-hot signals for many cases.

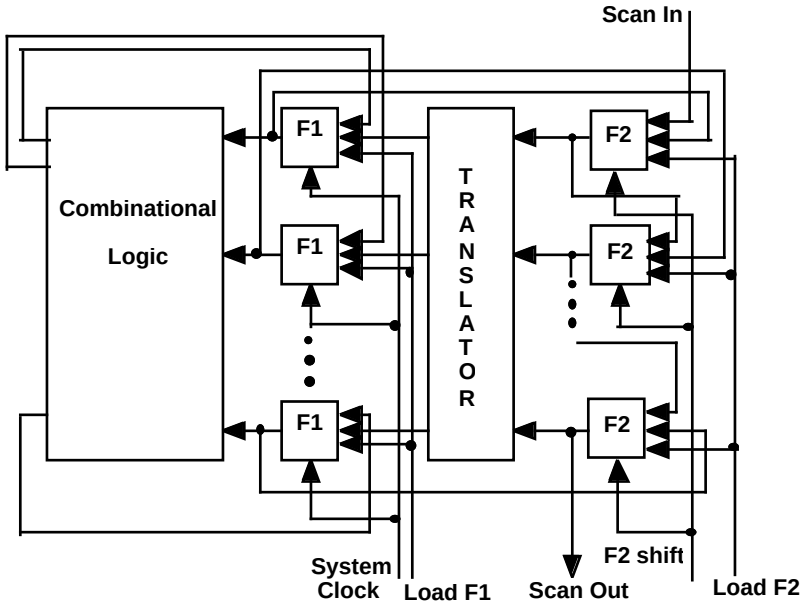

Figure 5 A shadow-register based technique to handle one-hot signals during scan

However, the decoder may add extra delay to the signal paths during all modes of operation. If the decoder is on a critical path, the system performance will be reduced. Hence, for delay-critical applications, our previous approaches may be unacceptable to the designers.

In this section, we propose another approach to handle one-hot signals in a scan-based design which does not affect the system performance during normal operation but adds extra delay in the logic paths during test mode. We propose a shadow register-based scheme, which is an extension of the L3 latch based LSSD scheme described in [Das Gupta 81]. Figure 5 shows the basic implementation of our method. The scheme requires two levels of flip-flops, F1 and F2, and extra logic between F1 and F2 called a *translator*. If F2 contains a *valid* state, the translator passes it to F1 unchanged. However, if F2 contains an *invalid* state, the translator logic transforms it into a valid state pattern. For example, if the state variables were encoded using a one-hot encoding scheme, the truth table shown in Table 1 could serve as a truth table for the translator between the F1 and the F2 flip-flops.

The functionality of the F1 flip-flops is the same as in basic scan design [McCluskey 86], [Eichelberger 77]. The F2 flip-flops correspond to the shadow register. In test mode, patterns are first scanned in using the F2 flip-flops. This pattern is fed into the translator, the output of which is loaded into the F1 flip-flops. The response of the combinational logic may be loaded into the F1 flip-flops or into both F1 and F2 flip-flops. In the first case, we require a separate control signal to load the contents of the F1 flip-flops into the F2 flip-flops so that the data can be scanned out. In the latter case this signal is not required and the response of the combinational logic can be directly scanned out of the F2 flip-flops. Moreover, depending on the technology, the designer may not want to have additional fanouts on the F1 flip-flop so that the performance of the circuit in the normal mode of operation is not degraded. Thus, safety (non-occurrence of invalid states) is ensured during testing. With this scheme, one-hot values are guaranteed even when random patterns are used for testing since the patterns are first loaded into F2. Moreover, the scheme adds no delay during the normal mode of operation.

This scheme is similar to the scan-set structures discussed in [McCluskey 86], [Stewart 77]. However, unlike the scan-set architecture, this scheme ensures safe scan operation due to the presence of the translator logic and also ensures that the presence of extra flip-flops and translator logic does not add to the signal delay during the normal mode of operation. The scheme has an extra overhead of one flip-flop per one-hot signal over normal scan plus the cost of the translator logic. The overhead due to the translator logic may be minimized by appropriate logic synthesis techniques but cannot be fully eliminated. To minimize the area overhead due to the shadow register, F2 flip-flops may be added only at places where the circuit under test contains logic generating or using one-hot signals. However, this may complicate the test control logic.

7. Conclusions

This paper addresses a critical problem of synthesizing testable designs containing logic that is controlled by one-out-of-n signals. Our schemes ensure one-hot values not only during scan-in or scan-out operation, but also when pseudo-random patterns are present in the system bistables.

Thus, our schemes are equally applicable to different pseudo-random BIST schemes. The encoder-based technique reported in Sec. 3.1 is simple to implement but has overhead in terms of area and delay. It can be used as a simple solution to the one-hot signal problem where the area and delay overheads are not of major concern. The general scheme described in Sec. 3.2 has smaller area overhead and it often generates designs with smaller area than the original synthesized designs. Hence, for area constrained applications this scheme provides a feasible solution. For delay critical circuits with tight performance constraints, the shadow register-based technique described in Sec. 6 can be used. It does not degrade the performance of the circuit in the normal mode of operation. However, the area overhead of this scheme is higher than the other two techniques. Moreover, if the system enters an invalid state during its normal mode of operation due to the presence of transient faults, the schemes reported in Sec. 3.1 and 3.2 ensure one-hot values on one-hot signals as long as the encoder (Sec. 3.1) or the decoder (Sec. 3.2) are fault-free. We have implemented the techniques reported in this paper and are currently incorporating them into **TOPS**, the Stanford CRC totally optimized synthesis for test tool. We are also investigating new techniques that allow specification of one-hot signals in FSM descriptions.

8. Acknowledgments

This work was supported by the Advanced Research Projects Agency under prime contract No. DABT63-94-C-0045. Thanks to Robert B. Norwood of Stanford CRC for his helpful comments regarding this work.

9. References

[Abramovici 90] Abramovici, M., M. A. Breuer and A. D. Friedman, *Digital Systems Testing Testing and Testable Design*, W. H. Freeman and Company, New York, USA, 1990.

[Abramovici 91] Abramovici, M., J. J. Kulikowski and R. K. Roy, "The Best Flip-Flops to Scan," *International Test Conference Proc.*, pp. 166-173, Oct. 1991.

[Cheng 90] Cheng, K. T. and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits," *IEEE Transactions on Computers*, pp. 544-548, April 1990.

[Das Gupta 81] Das Gupta, S., R. G. Walther, T. W. Williams and E. B. Eichelberger, "An Enhancement to LSSD and some Applications of LSSD in Reliability, Availability and Serviceability," *11th International Symposium on Fault-Tolerant Computing Proc.*, pp. 32-34, June, 1981.

[De Micheli 94] De Micheli, G, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, Inc., 1994.

[Eichelberger 77] Eichelberger, E. B. and T. W. Williams, "A Logic Design Structure for LSI Testability," *Design Automation Conference Proc.*, pp. 462-468, 1977.

[Eichelberger 91] Eichelberger, E. B., E. Lindbloom, J. A. Waicukauski and T. W. Williams, *Structured Logic Testing*, Prentice-Hall, Eaglewood Cliffs, NJ, USA, 1991.

[Hennessy 95] Hennessy, J. L. and D. A. Patterson, *Computer Architecture - A Quantitative Approach, 2nd Edition*, Morgan Kaufmann Publishers, San Mateo, California, USA, 1995.

[Lee 90] Lee, D. H. and S. M. Reddy, "On Determining Scan Flip-Flops in Partial Scan Designs," *International Conference on Computer Aided Design Proc.*, pp. 322-325, 1990.

[Levitt 95] Levitt, M., *et. al.*, "Testability, Debuggability, and Manufacturability Features of the UltraSPARC^{TM}-I Microprocessor," *International Test Conference Proc.*, pp. 157-166, 1995.

[LSI 96] *G10-p Cell-Based ASIC Products Databook*, LSI Logic, May 1996.

[McCluskey 86] McCluskey, E. J., *Logic Design Principles with Emphasis on Testable Semicustom Circuits*, Prentice-Hall, Eaglewood Cliffs, NJ, USA, 1986.

[Mitra 97] Mitra, S., L. J. Avra and E. J. McCluskey, "A New Output Encoding Problem and Its Exact Solution," To appear in *Proc. of International Conference on Computer-Aided Design*, Nov., 1997.

[Pateras 95] Pateras, S. and M. S. Schmookler, "Avoiding Unknown States when Scanning Mutually Exclusive Latches," *International Test Conference Proc.*, pp. 311-318, 1995.

[Sentovich 92] Sentovich, E. M., *et. al.*, "SIS: A System for Sequential Circuit Synthesis,"*Electronics Research Laboratory Memo. No. UCB/ERL M92/41*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, 1992.

[Shoji 86] Shoji, M., "Elimination of Process-Dependent Clock Skew in CMOS VLSI," *IEEE Journal of Solid-State Circuits*, pp. 875-880, 1986.

[Stewart 77] Stewart, J. H., "Future Testing of Large LSI Circuit Cards," *Dig. 1977 IEEE Semicounductor Test Conf.*, pp. 6-15, 1977.

[Suzuki 93] Suzuki, M., *et. al.*, "A 1.5 ns 32b CMOS ALU in Double Pass-Transistor Logic," *IEEE International Solid-State Circuits Conference*, 1993.

[Villa 90] Villa, T. and A. Sangiovanni-Vincentelli, "NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementations," *IEEE Trans. on CAD*, Vol. 9, No. 9, pp. 905-924, Sept. 1990.

[Yano 90] Yano, K., *et. al.* , "A 3.8-ns CMOS 16x16-b Multiplier using Complementary Pass-Transistor Logic," *IEEE Journal of Solid-State Circuits*, vol. 25, No. 2, pp. 388-395, April 1990.

[Yano 94] Yano, K., Y. Sasaki, K. Rikino and K. Seki, "Lean Integration: Achieving a Quantum Leap in Performance and Cost of Logic LSIs," *IEEE Custom Integrated Circuits Conference Proc.*, pp. 603-606, 1994.