1.

Properties of Faults in Synchronous Sequential Circuits and Their Application to Test Generation, Logic Optimization and Testable Designs

Sudhakar M. Reddy
Electrical and Computer Engineering Department
University of Iowa
Iowa City, IA 52242

In collaboration with Irith Pomeranz and Xijiang Lin

OVERVIEW

- Objectives of the Works Described
- The Importance of Synchronizing Sequences
- Properties of Synchronizable Synchronous Sequential Circuits
- Faulty Circuits with Easily Computed Synchronizing Sequences
- Undetectable and Redundant Fault Identification
- Logic Optimization
- Experimental Results for Each Area
- Conclusion

OBJECTIVES

Facilitate test generation.

- Find synchronizing sequences for fault free and faulty circuits
- Identify undetectable faults.

Obtain fully testable circuits.

- Identify and remove redundant faults.
- Use design-for-testability techniques for undetectable faults that remain.

4. THE IMPORTANCE OF SYNCHRONIZING SEQUENCES

A *synchronizing sequence* of a circuit C is an input sequence that brings C to a known state S independent of the initial state or the output sequence.

The state S is called a *synchronization state* of C.

A *test sequence* for a fault f distinguishes every state of the fault free circuit from every state of the faulty circuit.

A fault is detectable if it has a test sequence;
otherwise, it is undetectable.

5.

Using Synchronizing Sequences to Facilitate Test Generation

- Reduce the number of states that need to be distinguished by the test sequence. Only the synchronization states need to be considered.

- Alleviate the loss of accuracy due to unspecified values.

Used in the following procedures:

[Cho, Jeong, Somenzi and Pixley, JETTA 93]
[Wehbeh and Saab, VTS97]
[Lin, Pomeranz and Reddy, VLSI Design98]

6. Using Synchronizing Sequences to Remove Undetectable Faults

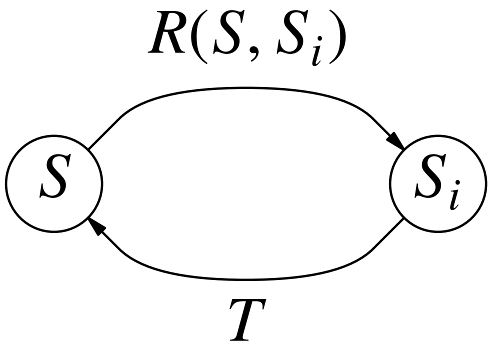
In a synchronizable circuit, an undetectable fault that leaves the circuit synchronizable can be removed without affecting normal circuit operation [Pomeranz and Reddy, IEEE TC 96].

7.

PROPERTIES AND FAULT CLASSIFICATION FOR SYNCHRONIZABLE SYNCHRONOUS SEQUENTIAL CIRCUITS

[Pixley, IEEE TCAD 92], [Pomeranz and Reddy IEEE TC 96]

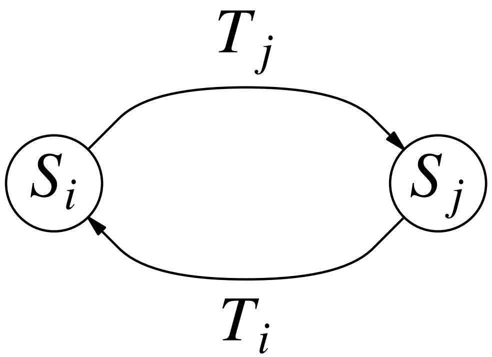
Lemma 1: Let S be a synchronization state of a synchronous sequential circuit C.
All the states reachable from S belong to the same strongly-connected component of the state diagram of C.



$R(S, S_i)$ - a transfer sequence from S to S_i.
T - a synchronizing sequence.

Lemma 2

If a circuit has synchronization states $S_1, S_2, \cdots, S_i$ with synchronizing sequences $T_1, T_2, \cdots, T_i$, respectively, then $S_1, S_2, \cdots, S_i$ are in the same strongly-connected component.



T_i - a synchronizing sequence into S_i.

9.

The Main Strongly-Connected Component

The strongly-connected component that contains all the synchronization states.

It is denoted $M - SCC$.

The other strongly-connected components are called secondary strongly-connected components and denoted $S - SCC$.

Lemma 3

Informal statement: Once a circuit with a synchronizing sequence enters its $M - SCC$, it cannot leave it until it is powered-up again.

Formal statement:

The graph of strongly-connected components $G(V, E)$:
V contains a vertex v_i for every strongly-connected component SCC_i.
There is an edge $e = (v_i, v_j)$ in E iff there is an edge from a state $S_i \in SCC_i$ to a state $S_j \in SCC_j$.

11.

Formal Statement of Lemma 3

In $G(V, E)$, the vertex corresponding to the $M - SCC$ is a sink (i.e., it has no outgoing edges).

Normal circuit operation after synchronization is limited to the $M - SCC$.

12.

Fault Removal

We use the term *redundancy removal* to describe the removal of logic due to a redundant fault.

We use the term *fault removal* to describe the removal of logic due to a given fault.

Conventionally, only redundant faults are considered for removal.

In synchronous sequential circuits with synchronizing sequences, there exist undetectable faults which are not redundant (partially detectable faults) that can be removed.

13.

Removal of Undetectable Faults

Type 1 Faults:

The faulty circuit does not have a synchronizing sequence.

Example:

$$Y_1 = y_1 y_2 + x \bar{y}_1$$
$$Y_2 = y_1 \bar{y}_2 + x y_1 \;/\; y_1 \bar{y}_2 + y_1$$
$$z = \bar{x} y_1 \bar{y}_2 + x \bar{y}_1 y_2$$

The fault is partially detectable.

14.

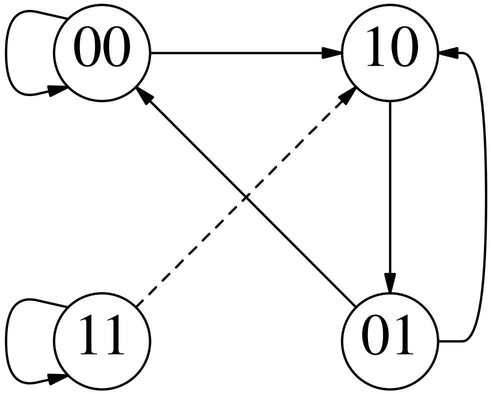
The Fault Free and Faulty State-Tables

$y_1 y_2$	$Y_1 Y_2, z$ $x = 0$	 $x = 1$
00	00,0	10,0
01	00,0	10,1
10	01,1	01,0
11	10/11,0	11,0

Synchronizing sequence:
Fault free - 000 to state 00
Faulty - no synchronizing sequence

15.

*SCC*s of the Fault Free and Faulty Circuits



$M - SCC$ - {00,01,10}
$S - SCC$ - {11}

The fault disconnects the $S - SCC$ from the $M - SCC$.
As a result, no synchronizing sequence exists for the faulty circuit.

16.

Removal of Type 1 Faults

Since fault free circuit operation

relies on the existence of a synchronizing sequence,

the fault cannot be removed.

Type 2 Faults

The faulty circuit has a synchronizing sequence.

Theorem: Let f be a partially detectable fault in a synchronous sequential circuit with a synchronizing sequence.

Let f create a faulty circuit that has a synchronizing sequence, and hence has an $M-SCC$.

The $M-SCC$ of the faulty circuit is equivalent to the $M-SCC$ of the fault free circuit.

18.

Example

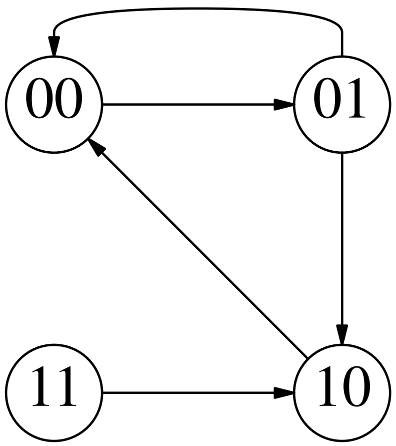
$$Y_1 = \bar{x} y_2 + y_1 y_2$$
$$Y_2 = \bar{y}_1 \bar{y}_2$$
$$z = \bar{x}\ \bar{y}_2 + x y_1 y_2 \,/\, \bar{x}\ \bar{y}_2$$

	$Y_1 Y_2, z$	
$y_1 y_2$	$x = 0$	$x = 1$
00	01,1	01,0
01	10,0	00,0
10	00,1	00,0
11	10,0	10,1/0

Synchronizing sequence -
0101 to state 00 (fault free and faulty)

19.

Fault Removal



$M-SCC$ - {00,01,10}
$S-SCC$ - {11}

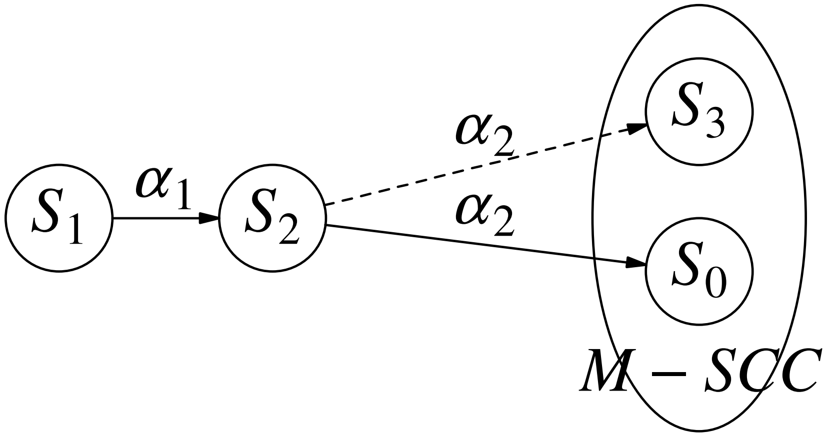
The fault affects only the output value of the transition from 11 to 10.

The fault can be removed.

20.

FAULTS WITH EASILY COMPUTED SYNCHRONIZING SEQUENCES

Example 1:



$T_{sync} = (\alpha_1 \alpha_2)$. Synchronization state is S_0.
After applying T_{sync}: $S \neq S_1 \rightarrow S_0$. $S_1 \rightarrow S_3$.
Apply T_{sync} a second time: All the states reach S_0.

Generalization of the Example

A fault is represented by a set of state-transitions of the form $(\alpha, S_1, S_2/S_2', \beta/\beta')$.

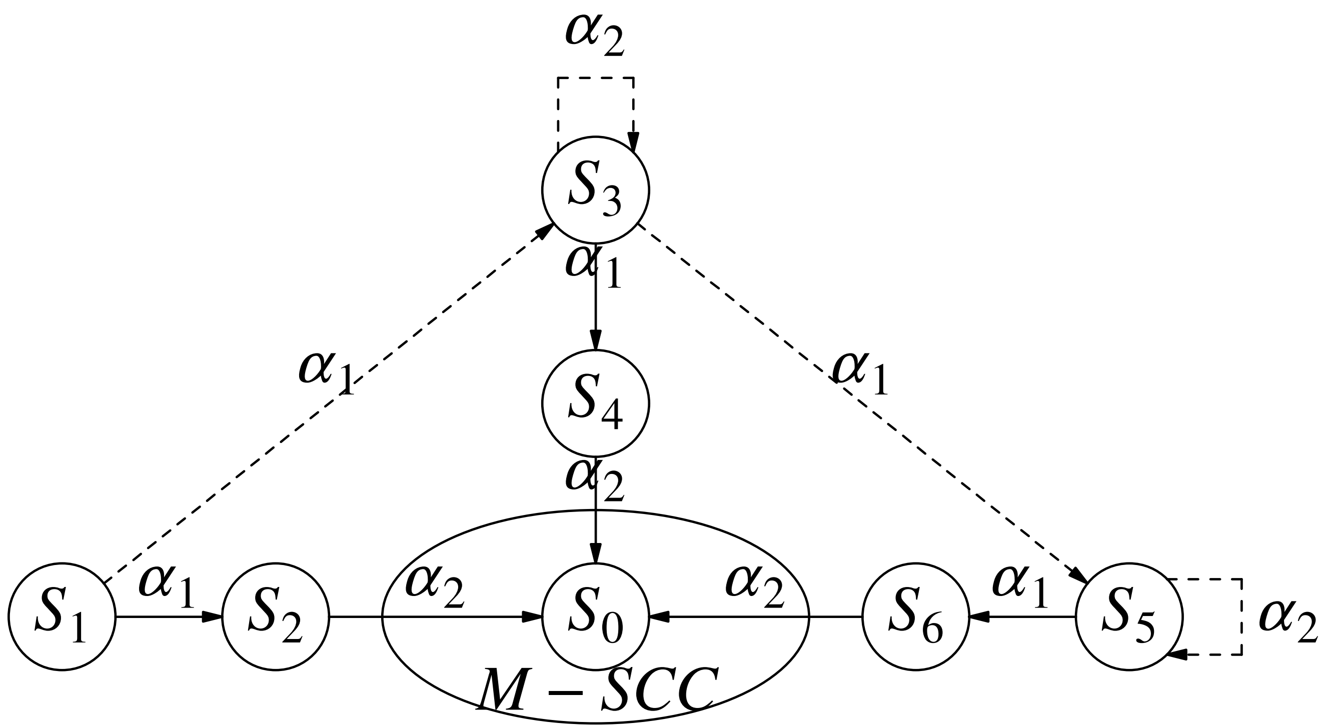
Let f be a fault in a circuit C with a synchronizing sequence $T_{sync} = (\alpha_1 \alpha_2 \cdots \alpha_m)$.

Suppose that all the state-transitions affected by f that have α_i, $1 \leq i \leq m$, as the input vector are of the form $(\alpha_i, S_{i1}, S_{i2}/S_{i2}', \beta_i/\beta'_i)$ where:

- S_{i1} is in an $S-SCC$ of the fault free circuit.
- S_{i2} and S_{i2}' are in the $M-SCC$ of the fault free circuit.

For f, $T_{sync}T_{sync}$ is a synchronizing sequence of the faulty circuit.

Example 2



$T_{sync} = (\alpha_1\alpha_2)$. Synchronization state S_0.
After applying T_{sync}: $S \neq S_1, S_3 \longrightarrow S_0$. $S_1 \longrightarrow S_3$. $S_3 \longrightarrow S_5$.
Apply T_{sync} a second time: $S_0 \longrightarrow S_0$. $S_3 \longrightarrow S_5$. $S_5 \longrightarrow S_0$.
Apply T_{sync} a third time: All the states reach S_0.

Experimental Results using Duplicated Synchronizing Sequences

		1rep						
circuit	flts	len	flts	2rep	3rep	4rep	5rep	6rep
s208	215	1	186	193				
s298	308	2	293	296				
s344	342	2	311	323			325	327
s382	399	1	361	383				
s386	384	2	349	380				
s400	421	1	385	405				
s420	430	1	373	388				
s641	467	1	439	442				
s820	850	1	844	848				
s1196	1242	1	1228	1242				
s1423	1515	3	1420					
s5378	4603	13	4480	4542	4546			
s35932	39094	1	37356	39084				
am2910	2573	6	2330	2347	2348	2349	2361	
mp1_16	1708	33	1582	1617		1618		
pcont2	11272	1	11247	11256				
piir8	29689	1	29640	29686				

Extension to Multiple Synchronizing Sequences

Even if T_{sync_1} is destroyed by a fault,

T_{sync_2} may synchronize the faulty circuit, or

$(T_{sync_2})^k$, $k > 1$, may synchronize the faulty circuit.

Use disjoint synchronizing sequences

$$T_{sync_1}, T_{sync_2}, \cdots, T_{sync_m}.$$

Apply $(T_{sync_1})^k (T_{sync_2})^k \cdots (T_{sync_m})^k$.

25.

Experimental Results using Multiple Duplicated Sequences

circuit	1seq 1rep len	1seq 1rep flts	1seq mrep	mseq 1rep len	mseq 1rep flts	mseq mrep
s208	1	186	193	5	195	195
s298	2	293	296	6	296	296
s344	2	311	327	10	325	329
s382	1	361	383	4	383	383
s386	2	349	380	10	380	380
s400	1	385	405	4	405	405
s420	1	373	388	5	389	390
s641	1	439	442	5	451	455
s820	1	844	848	5	849	849
s1196	1	1228	1242	5	1242	1242
s1423	3	1420	1420	15	1439	1439
s5378	13	4480	4546	25	4542	4546
s35932	1	37356	39084	5	39084	39084
am2910	6	2330	2361	30	2413	2413
mp1_16	33	1582	1618	76	1624	1628
pcont2	1	11247	11256	5	11256	11256
piir8	1	29640	29686	5	29686	29686

26.

UNDETECTABLE AND REDUNDANT FAULTS

Efficient methods to identify subsets of undetectable faults:

- Analysis of values that cannot be assigned to lines in the circuit:
  [Long, Iyer and Abramovici, VTS95, DAC96],
  [Cao and Pradhan, ICCAD96]

- An iterative logic array of limited length:
  [Agrawal and Chakradhar, ETC93, IEEE TCAD 95],
  [Pomeranz and Reddy, VTS94]

- Uninitializable state-variables and uncontrollable values:
  [Liang, Lee and Chen, ATS94, D&T95]

Use in Test Generation

Step 1: Identify a subset of undetectable faults.

Step 2: Test generation for the remaining faults.

The Proposed Procedure

Step 1: Identify a subset of faults that will not be proved undetectable.

Step 2: Identify a subset of undetectable faults out of the remaining faults.

Step 3: Test generation for the faults not proved to be undetectable.

Techniques Used in the Proposed Procedure

An iterative logic array of small length compared to previous methods (1 or 2).

Sequential static learning over several time units to compensate for the small array size.

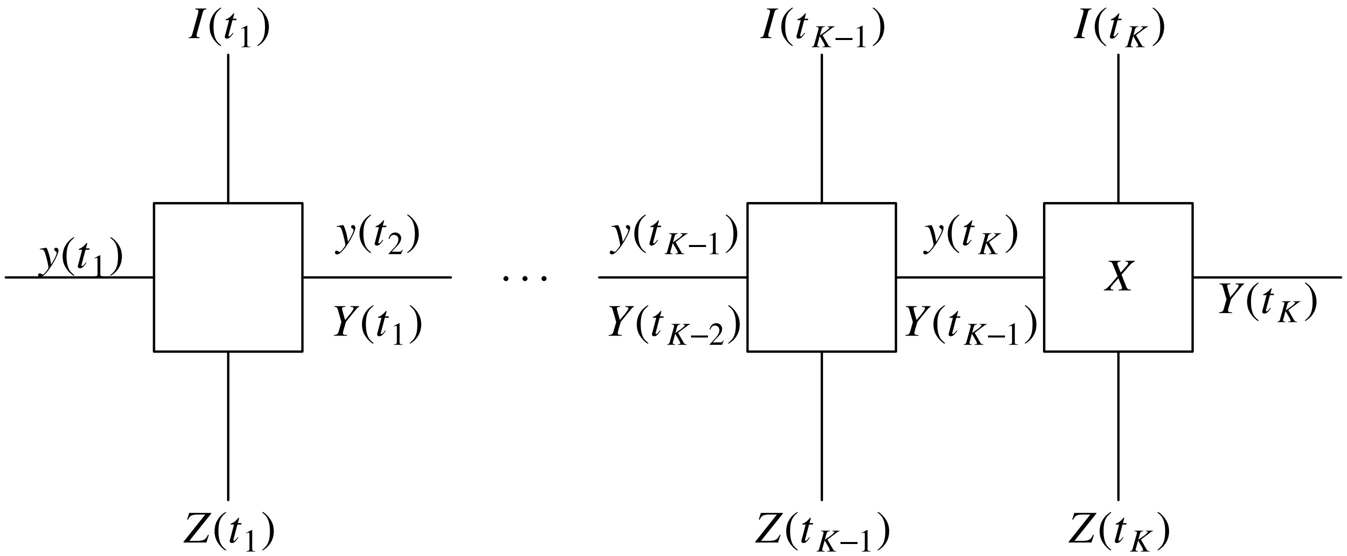
Provides information about line values that cannot be obtained for reachable states (states used during normal circuit operation, included in sink-SCCs).

If the detection of a fault requires such values, then the fault is undetectable.

Identifies more undetectable faults than previous procedures.

30.

Using an Iterative Logic Array



31.

Undetectable Faults based on the Iterative Logic Array

Inject the fault *only* into the last cell of the iterative logic array.

Assume that all the primary inputs and the present state variables of Cell 1 are controllable.

Assume that all the primary outputs and the next-state variables of Cell K are observable.

Use a combinational test generation procedure for single stuck-at faults.

If a test cannot be found, the fault is undetectable.

32.

Sequential Static Learning

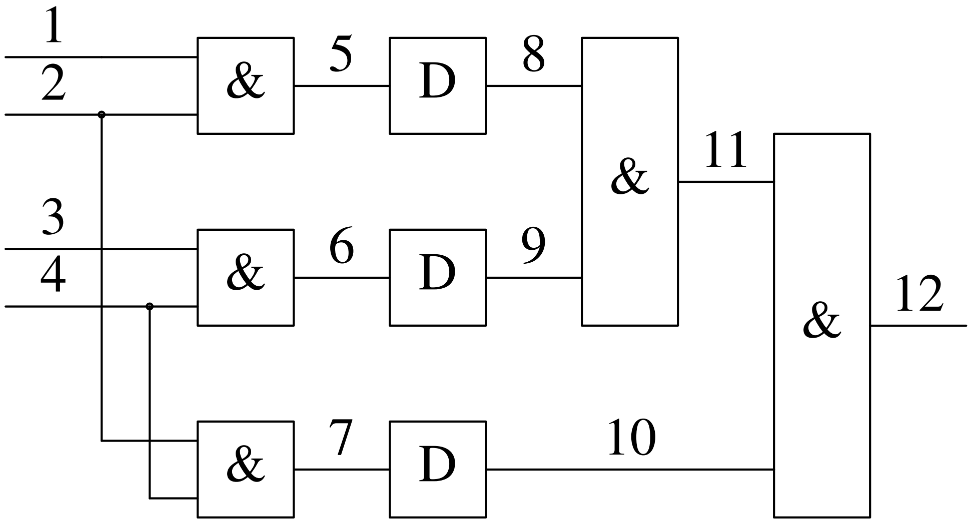
Goal: Find relations among values that pairs of lines can assume.

Use them during test generation to avoid conflicting assignments to lines that can only carry specific combinations of values.

Sequential static learning is done over several time units.

33.

Example of Sequential Static Learning



Line 11 = 1 at i Lines 8 and 9 = 1 at i

Lines 5 and 6 = 1 at $i-1$ Lines 1, 2, 3 and 4 = 1 at $i-1$

Line 7 = 1 at $i-1$ Line 10 = 1 at i

Sequential Static Learning in the Proposed Procedure

Only the fault free circuit.

An iterative logic array with n_L time units labeled

$$-n_L/2, \cdots, -1, 0, 1, \cdots, n_L/2.$$

Implications are stored as $g = \alpha$ at 0 => $h = \beta$ at i.

Observation: The implications are valid for states reachable during normal circuit operation (after synchronization, or in a sink-SCC).

35.

Identifying Undetectable Faults

Test generation is carried out in the limited-length iterative logic array.

A fault is declared undetectable if it requires both $g = \alpha$ at j and $h = \bar{\beta}$ at $(j + i)$ for some j, and the implication $g = \alpha$ at 0 => $h = \beta$ at i was derived during static learning.

Observation: It is sufficient to consider reachable states to show that a fault is undetectable. The results of sequential static learning are valid for reachable states.

Experimental Results

		K = 1			K = 2			incremental		
circuit	flts	und	cpu	red	und	cpu	red	und	cpu	red
s5378	4603	690	20.5	690	731	34.7	731	731	37.6	731
s9234	6927	3229	241	2993	3235	454	2993	3235	593	2993
s13207	9815	1263	199	368	1535	736	368	1535	830	368
s15850	11725	762	1684	212	771	3122	212	774	3641	212
s35932	39094	3984	362	3984	3984	321	3984	3984	422	3984
s38417	31180	1064	1162	720	1126	3808	720	1130	4566	720
s38584	36303	1955	1241	2	2142	2833	2	2142	3522	2
s499	583	83	1.79	0	144	2.48	0	144	2.37	0
s967	1066	3	0.92	0	19	1.89	0	19	2.05	0
s1269	1343	1	0.31	0	1	0.59	0	1	0.74	0
s1512	1357	7	0.74	0	7	5.45	0	7	5.83	0
s3271	3270	0	1.58	0	0	2.20	0	0	3.55	0
s3330	2870	408	3.83	408	421	7.69	421	421	9.56	421
s3384	3380	1	1.39	1	1	2.36	1	1	3.72	1
s4863	4764	82	3.13	82	126	3.88	126	126	5.78	126
s6669	6684	0	1.38	0	0	1.83	0	0	2.93	0
prolog	3305	630	5.8	630	643	10.7	643	643	12.5	643

Effect on Test Generation Time

	w/o und		with und	
circuit	det	cpu	det	cpu
s5378	3646	6052	3646	3919
s3330	2124	440	2124	336
s4863	4638	429	4637	171
s35932	35110	10843	35110	10409
prolog	2325	4293	2325	3942

Comparison with Other Methods

circuit	proposed und	proposed cpu	[A&C] und	[A&C] cpu	FIRES[L,I&A] und	FIRES[L,I&A] cpu
s5378	731	37.6	335	174	367	52
s9234	3235	593	520	846	284	114
s13207	1535	830	924	886	893	98
s15850	774	3641	440	901	332	313
s35932	3984	422	3984	81	3984	557
s38417	1130	4566	203	1555	147	318
s38584	2142	3522	2131	3573	1437	272
s499	144	2.37	-	-	5	12.2
s1269	1	0.74	-	-	1	0.9
s3330	421	9.56	-	-	162	6.9
prolog	643	12.5	-	-	253	5.9

LOGIC OPTIMIZATION

Synchronizable circuits (extendible to weak-synchronizable circuits).

Identify undetectable faults.

An undetectable fault can be removed if the faulty circuit is synchronizable.

Other methods:

[Cheng, DAC91], [Cheng, ICCAD91],
[Cho, Hachtel and Somenzi, IEEE TCAD 93],
[Entrena and Cheng, ICCAD93].

Procedure

Identify removable faults (undetectable and synchronizable).

Identify a maximal subset of removable faults that form an undetectable multiple fault F_m.

If the circuit remains synchronizable in the presence of F_m:

  Remove the logic corresponding to the faults in F_m.

  Repeat the process.

The Set F_m

Initially, $F_m = \phi$.

At every iteration, a single fault f is considered.

If $F_m \cup \{ f \}$ is undetectabe, f is added to F_m.

To check whether $F_m \cup \{ f \}$ is undetectabe, test generation is carried out in an iterative logic array of limited length with all the faults injected in the last time frame.

Experimental Results

Combinational redundancies:

circ	original gates	original DFFs	original lines	simplified gates	simplified DFFs	simplified lines	after removal gates	after removal DFFs	after removal lines	iter	time
c1355	546	0	1355	514	0	1323	514	0	1315	2	1.8
c1908	880	0	1908	880	0	1908	872	0	1888	4	2.1
c2670	1193	0	2670	1161	0	2638	879	0	2040	12	9
c3540	1669	0	3540	1669	0	3540	1546	0	3247	8	11.8
c5315	2307	0	5315	2290	0	5298	2262	0	5198	6	16.2
c6288	2416	0	6288	2416	0	6288	2382	0	6201	3	2.2
c7552	3512	0	7552	3466	0	7506	3207	0	6988	37	250
s9234	5597	228	9234	1577	160	3609	1216	160	2877	16	62
s13207	8027	669	13255	3393	649	8435	2986	647	7353	55	1106
s15850	9786	597	15861	4695	586	10655	4165	583	9466	21	285
s35932	16385	1728	35932	14283	1728	33830	11579	1728	28166	44	1823
s38417	22397	1636	38557	13701	1564	29493	13498	1564	29059	8	941
s38584	19407	1452	38586	16357	1451	35535	14323	1434	31520	15	1574

Sequential Circuits with $K = 1$

circ	original			simplified			after removal			iter	time
	gates	DFFs	lines	gates	DFFs	lines	gates	DFFs	lines		
s208	96	8	208	96	8	208	59	5	122	3	0.24
s344	169	15	344	158	15	333	151	15	315	2	0.08
s349	170	15	349	159	15	338	152	15	317	3	0.12
s386	159	6	386	133	6	360	123	6	307	4	0.48
s400	162	21	400	143	21	381	139	21	362	3	0.33
s420	196	16	420	196	16	420	63	5	126	3	0.57
s444	181	21	431	168	21	431	159	21	396	3	0.43
s713	393	19	713	235	19	555	189	19	447	3	0.36
s832	287	5	832	287	5	832	281	5	812	5	1.87
s838	390	32	838	390	32	838	61	5	124	3	2
s1238	508	18	1238	508	18	1238	475	18	1143	7	5.4
s1423	657	74	1423	630	74	1396	619	74	1361	3	1.9
s1494	647	6	1494	597	6	1444	589	6	1424	2	2.2
s5378	2779	179	5295	1900	179	4416	1313	133	3084	10	43
s3330	1798	132	3330	1149	132	2690	814	66	1985	2	4.2
s4863	2342	104	4799	2261	104	4718	2169	104	4471	28	42
prolog	1606	136	3371	1406	136	3171	816	66	1979	3	4.8
am2910	930	87	2253	925	87	2248	825	87	1953	23	22
mult16	645	55	1514	627	55	1496	624	55	1481	3	1.8
div16	833	50	1998	812	50	1977	661	50	1580	26	33
pcont2	4010	24	9781	4002	24	9773	1512	24	3471	32	125
piir8	10712	56	25926	10704	56	25926	3472	56	7703	99	3914

Sequential Circuits with $K = 2$

	original			simplified			after removal				
circ	gates	DFFs	lines	gates	DFFs	lines	gates	DFFs	lines	iter	time
s208	96	8	208	96	8	208	56	5	118	3	0.26
s298	119	14	298	104	14	283	99	14	256	5	0.32
s349	170	15	349	159	15	338	151	15	315	4	0.17
s386	159	6	386	133	6	360	113	6	280	13	1.6
s400	162	21	400	143	21	381	139	21	359	5	0.62
s420	196	16	420	196	16	420	60	5	122	3	0.64
s444	181	21	431	168	21	431	159	21	395	4	0.69
s526	193	21	526	178	21	511	175	21	486	8	1.4
s820	289	5	820	281	5	812	271	5	767	15	6.7
s832	287	5	832	287	5	832	271	5	767	18	8.2
s838	390	32	838	390	32	838	58	5	120	3	2.1
s1488	653	6	1488	589	6	1424	566	6	1355	21	21
s1494	647	6	1494	597	6	1444	575	6	1380	17	19
s5378	2779	179	5295	1900	179	4416	1292	123	3025	13	56
s4863	2342	104	4799	2261	104	4718	2135	104	4382	29	51

Effects on Test Generation

	original						after removal						
circ	faults	det	und	abt	vec	time	K	faults	det	und	abt	vec	time
s208	215	150	65	0	98	0.75	2	121	121	0	0	105	0.36
s298	308	273	26	9	206	11	2	279	264	14	1	180	2.6
s344	342	335	5	2	83	1.8	1	321	317	1	3	88	1.8
s349	350	341	7	2	104	1.4	1	322	317	3	2	86	2.5
s386	384	314	70	0	226	3.5	2	304	290	14	0	196	1.8
s420	430	204	226	0	100	5	2	121	121	0	0	117	0.33
s820	850	815	35	0	838	22	2	802	800	2	0	747	17.2
s832	870	819	51	0	881	29	2	802	800	2	0	734	14.1
s838	857	303	501	53	109	69	2	119	119	0	0	95	0.29
s1238	1355	1283	72	0	347	5.9	1	1243	1241	3	0	315	4.4
s1488	1486	1446	40	0	918	65	2	1416	1416	0	0	870	25.8
s1494	1506	1455	50	1	759	65	2	1441	1427	14	0	810	37.3
s3330	2870	2124	709	37	758	595	1	2130	1878	232	20	802	236
s4863	4764	4638	126	0	671	430	2	4386	4386	0	0	711	85
s5378	4603	3643	753	207	1499	7194	2	3206	3160	24	22	1214	962
s35932	39094	35110	3984	0	235	10382	1	33174	33174	0	0	237	7091
prolog	3305	2325	694	286	1003	4358	1	2114	1862	233	19	974	281
am2910	2391	2207	161	23	1034	567	1	2056	2046	4	6	932	163
mult16	1708	1684	15	9	371	394	1	1687	1669	8	10	284	360
pcont2	11272	6830	3849	593	329	4402	1	3856	3663	0	193	194	300
piir8	29689	18077	10799	813	488	18704	1	8366	8344	18	4	390	145

46.

Application of the Test Sequences to the Original Circuits

circ	faults	original det	original vec	original time	simulation det	simulation vec	diff	total time
s208	215	150	98	0.75	147	105	3	0.71
s298	308	273	206	11	269	180	4	3.3
s344	342	335	83	1.8	335	88	0	2
s349	350	341	104	1.4	341	86	0	2.8
s386	384	314	226	3.5	312	196	2	3.5
s420	430	204	100	5	203	117	1	1.3
s820	850	815	838	22	811	747	4	25
s832	870	819	881	29	814	734	5	24
s838	857	303	109	69	297	95	6	3.3
s1238	1355	1283	347	5.9	1268	315	15	10
s1488	1486	1446	918	65	1445	870	1	49
s1494	1506	1455	759	65	1455	810	0	59
s3330	2870	2124	758	595	2124	802	0	246
s4863	4764	4638	671	430	4636	895	2	139
s5378	4603	3643	1499	7194	3649	1214	-6	1070
s35932	39094	35110	235	10382	35109	237	1	9364
prolog	3305	2325	1003	4358	2325	974	0	297
am2910	2391	2207	1034	567	2206	932	1	189
mult16	1708	1684	371	394	1684	284	0	365
pcont2	11272	6830	329	4402	6770	194	60	444
piir8	29689	18077	488	18704	17962	390	115	4225

CONCLUSION

We considered methods to facilitate test generation and derive fully testable synchronous sequential circuits.

- The importance of synchronizing sequences.
- Synchronizing sequences for faulty circuits.
- Identification of undetectable and redundant faults.
- Removal of redundant faults for logic optimization.