IEEE VLSI Design Conference, 2001, pp. 103-108

Error Diagnosis of Sequential Circuits Using Region-Based Model *

Anand L. D'Souza[†] and Michael S. Hsiao[††]
(anand.dsouza@eng.sun.com, mhsiao@caip.rutgers.edu)
[†]Sun Microsystems, Palo Alto, CA
[††]Dept. Electrical & Computer Engineering, Rutgers University, Piscataway, NJ

Abstract

Algorithms to locate multiple design errors using region-based model are studied for both combinational and sequential circuits. The model takes locality aspect of errors and is based on a 3-value, non-enumerative analysis technique. Studies show the effectiveness of the region based model for gate connection and gate substitution errors. For sequential circuits, we try to locate the time frame at which the error was first excited, by re-simulating as few vectors as possible preceding the erroneous vector in a fully initialized circuit to carry out the diagnosis. Experimental results on benchmark circuits are used to demonstrate rapid and accurate locating of multiple errors.

1 Introduction

Error diagnosis is critically helpful in providing feedback to the designer, especially after a failed design verification, to suggest potential error sites. Huang et.al. [5] have presented techniques for diagnosis for both sequential and combinational circuits by extensive enumeration and simulation. Tomita *et al.* [6, 7] proposed the use of input pattern for locating design errors for both single and multiple errors. Simulation-based techniques for diagnosing errors were presented in [8], with error location and correction for macro-based circuits in [9]. Those techniques rely on analyzing nodes one at a time. Synthesis based methods for error location and correction have been proposed [10, 11] using BDD's on limited error models. A general model for both fault and error diagnosis was proposed by Boppana *et al.* [1] and has been used to diagnose single errors in combinational circuits. The effectiveness of the model was studied by injecting random gate substitution errors in ISCAS 85 combinational benchmark circuits. The model has been extended to multiple errors which utilizes the locality of errors (Region-based error model) [2, 4].

In this work, we have used the region-based model for diagnosing gate connection errors in combinational circuits. We have also used the region-based model to diagnose errors in sequential circuits by isolating the time frame for which the error is first observed. This is done by simulating a few vectors before the vector at which the circuit failed and then use the above algorithm to carry on a diagnosis of the sequential circuit in full-scan mode. Our experiments performed on the ISCAS 89 benchmark circuits by introduction of single and multiple stuck faults show that the diagnosis algorithm is indeed effective in diagnosing the errors.

*This research was supported in part by the NJ Commission on Science and Technology under contract 00-2042-007-14.

2 Region-Based Model

Boppana *et al.* [2] have proposed two logic level, structural, error models to capture the locality of errors for application in diagnosis: topologically-bounded error model and the region-based error model. We use the region-based error model for capturing the effects of errors in the vicinity of any gate in the circuit. Such groups are called "regions" and form the basic mechanism to model, introduce and simulate errors. These objects provide a logic-level alternative to capture locality of errors. Each region is formed by including all the gates within a fixed structural distance called the *radius* from a single gate (called "center") of the circuit. A region of radius 1 around gate G consists of G, its fanouts and fanins. Regions are formed for every gate in the circuit, so there can be as many **overlapping** regions as the number of gates in the circuit.

During simulation, all the output nodes of the region are first set to unknown value X to cover any arbitrary error that may occur in the region. If no X propagates to a primary output for a given vector V, we can conclude, for this vector, that any error within this region will **not** be detectable. Otherwise, we say that the region is a possible candidate and may have caused the erroneous behavior. This concept is illustrated in Figure 1. The specification specifies that the primary output (PO) #2 should have a value 0 for the given vector, but due to faulty implementation we get an erroneous output of 1. Now we simulate region A with the fanouts of region A set to X's, and if one of the X's propagate at the PO #2, then we can say that region A may contain the error. On the other hand, if no X propagates to PO #2 when simulating region B, then we can assert that the error is not contained in region B.

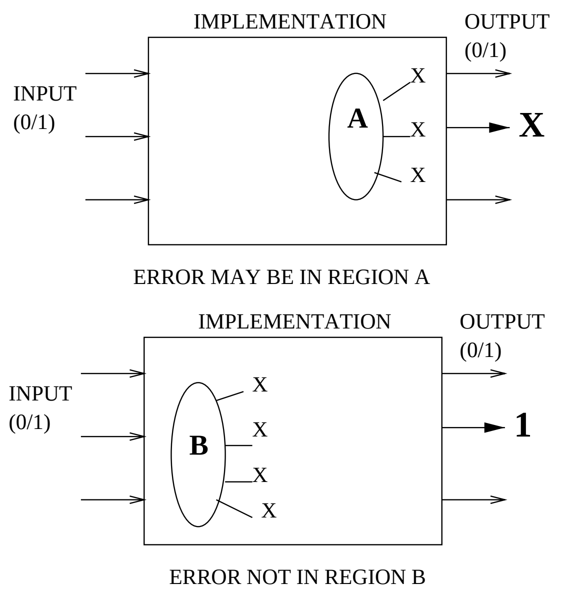

Figure 1: Diagnosis using Regions.

3 Diagnosis of Design Errors

We propose two cases for studying the behavior of region based model for gate connection errors. In case 1, we considered the diagnosis algorithm similar to [2], with the exception that only the *erroneous* vectors are considered for the matching procedure. An erroneous vector is one at which the output response of the implementation did not match that of the specification. The primary output at which the error is observed is referred to as *differing output*. The algorithm is shown in Figure 2.

The first step in the algorithm is to extract the regions of specified radius from the faulty gate level implementation. The Xlist-based algorithm then uses the X-Simulate(V,L) function to place an X on all the output nodes of the region being simulated. By the definition of region-based model if X propagates to the primary outputs at which the error was observed, we need to make that region a candidate. Diagnosis is conducted for each erroneous vector, and for every region, we observe the number of matches (0/0), (1/1) value combinations, mismatches (0/1), (1/0) value combinations and partial matches (0/X), (1/X) value combinations on the primary outputs. So the score of a region is incremented (decremented) depending on whether there is match/partial match (mismatch). A typical run would use 10, 5 and 1000 for the three quantities MATCHCOUNT, PARTIALMATCHCOUNT, and MISMATCHPENALTY, respectively [2]. Finally, a ranked candidate list is generated. A candidate with rank 1 indicates higher probability that it fully contains the error.

In case 2, we propose an algorithm shown in Figure 3. The conditions for candidacy are changed as follows. A region is a candidate if and only if X propagates to **each** of the erroneous outputs, for every erroneous vector. Score

```
begin
 for each erroneous vector
   for each region L in circuit
     X-Simulate(V, L) // Simulate the circuit with all output
                  //nodes of L set to X
     for each primary output do
       if ( Match (X-response, Correct response)
         Score[L] += MATCHCOUNT;
       else if (PartMatch (X-response, Correct response)
         Score[L] += PARTIALMATCHCOUNT;
       else
         Score[L] -= MISMATCHPENALTY;
end
```

Figure 2: Diagnosis Algorithm for Case 1

is calculated only for the erroneous outputs.

Consider the two examples shown in Table 1 for a circuit with four primary outputs. The error-free response is 0000, while the erroneous implementation yielded **1100**. Three regions are being considered for diagnosis using parameters MATCHCOUNT, PARTIALMATCHCOUNT and MISMATCHPENALTY of 10, 5 and 1000 respectively. In region 1, the error in the circuit is observed at primary outputs 1 and 2. So From the algorithm for case 1 we have two partial matches on first 2 bits and two definite matches on last 2 bits, resulting in a score of 30. Similarly the score for region 2 is 25. For case 2, we consider only the outputs at which the error was observed and hence the scores for both regions 1 and 2 are 10. In region 3, we see that in case 1 there is mismatch at primary output #1 and hence it will incur a mismatch penalty resulting in a score of -975. But for case 2, since **X** did not propagate to primary output 1, the region is dropped from subsequent simulations and removed from the candidate list.

Table 1: **Score calculation Example**

Region	X-simulation	Case	Score
1	XX00	1	30
		2	10
2	XXX0	1	25
		2	10
3	0X00	1	-975
		2	drop

Correct response: **0000**; erroneous response: **1100**

Finally, all the regions in the candidate list are given a rank, which in case 2 will all be 1. This is because all the regions that survived in the candidate list have had X's observed at all erroneous outputs for all erroneous vectors. This ranking procedure is compatible to the case 1 algorithm.

Such a stringent condition can be placed in case 2 based on the assumption that the error is *fully* contained within the region of a given radius. If the region is suf-

```
begin
  for each region L in circuit
    isCandidate[L] = TRUE;
  for each **erroneous vector**
    for each region L in circuit && isCandidate[L]==TRUE
      X-Simulate(V, L) // Simulate the circuit with all output
                  //nodes of L set to X
      for each erroneous output
        if (PartMatch (X-response, Correct response)
          Score[L] += PARTIALMATCHCOUNT;
        else
          isCandidate[L] = FALSE;
end
```

Figure 3: Diagnosis Algorithm for Case 2

ficiently large, our assumption of full containment of the error is very reasonable, since most errors and faults are localized in nature. In case of single errors, i.e., only one gate being in error, the assumption of full containment is always satisfied. In the case of multiple errors, regions of small radii may not be sufficient to get a good diagnosis [2]. So if we can assure that the error is contained in at least one of the regions of the given radius, our algorithm will guarantee successful diagnosis and the region which fully contains the multiple error will be returned in the candidate list. If we increase the radius of regions, then more than one region can contain the error of the previous smaller size, and hence the candidate list may be longer. If the error in the circuit is contained in a region of larger radius, but not in the smaller radius, we will get a candidate list which will have partial hits of the error for the smaller radius. That is, more than one region will contain a part of the error and this information will be helpful in diagnosis. But if we conduct diagnosis with regions of the larger radius, we will definitely get a region in the candidate list which fully contains the error. Since the candidate list will contain more than one candidate, it signifies that along with a exact hit, there will be considerable amount of partial hits.

4 Sequential Circuits

An error in a sequential circuit may be excited several time frames prior to propagation to the output. So we need to isolate the time frame at which the error is first excited and propagated to either a flip-flop or a PO. For error diagnosis, this is possible because the designer will have full access to the flip-flops in the circuit. The vector at which the error is detected may be deep in the test set and hence it would not be feasible to simulate the entire test set till the error was observed.

We need to get to the state of the machine at which the error was first activated. We can do so by simulating a few vectors preceding the erroneous vector (where the error has been observed at the PO). This is illustrated in Figure 4. Consider that we have the erroneous vector

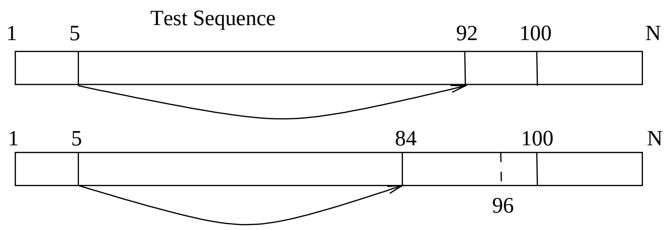

Figure 4: Time Frame Identification

number to be 100, and the error was first excited and propagated in the time frame 98. Instead of simulating the first 97 vectors to reach this state, we want to reach a state that activates the error by re-simulating as few vectors as possible. Let us assume that only the first 5 vectors are initializing vectors. So we first run the 5 initialization vectors and then jump to vector number 92. We may see that no error is detected at vector number 100. So we now reinitialize the circuit with the first 5 vectors and jump to vector number 84. This time we find that at time frame 96, we observe the error effect. So time frame 96 will be used for diagnosis. Note that, since we are using a different set of vectors to simulate the circuit, the machine may go through a different set of states and this can cause the error to be activated in a different time frame than the one in which it was observed when the entire test set was simulated.

The algorithm is shown in Figure 5. This algorithm takes the index number of the vector at which the design failed as an input. The first step is to initialize all the flip-flops in the circuit to a known state. This can be achieved by using the initialization sequence (which will be known to the designer) or by scanning in the reset state. Then, we simulate certain number of vectors, specified as JUMPNUM by the user (e.g. 8), before the erroneous vector and look for a time frame in which the error is observed at a flip-flop or a PO. If such a time frame is found, then a combinational diagnosis can be carried out on the circuit using the case 2 algorithm. If not, we increase the parameter JUMPNUM and repeat the procedure. In the worst

```
//User defined parameters : JUMPNUM, ERRORVECTOR
begin
  Initialize the flops;
  errorObserved = 0;
  while (errorObserved == 0)
    StartVector = ERRORVECTOR − JUMPNUM;
    while (StartVector <= ERRORVECTOR)
      Simulate(StartVector);
      if error observed at flop or PO
        errorObserved = 1;
        break;
      StartVector++;
    JUMPNUM = JUMPNUM × 2;
  candidateList = runDiagnosisCase2();
  return candidateList;
end
```

Figure 5: Sequential Diagnosis Algorithm

Table 2: Comparison between Algorithms for Gate Connection Errors for Radius 1

Ckt	Vec	Err Vec	Rgn	Case 1				Case 2			
				T	Cand	E	R	T	Cand	E	R
c432	100	9.6	203	0.07	38.8	1.1	3.8	0.04	38.8	1.1	0.5
c499	184	29.4	275	0.32	56	1.3	6.2	0.13	34.9	1.3	0.5
c880	128	10.9	469	0.18	31.6	1.6	8	0.08	24.9	1.6	0.5
c1355	198	36.5	619	1.26	111.9	1.5	12.7	0.30	75	1.5	0.5
c1908	280	55.8	938	2.91	90.1	1.4	6.8	0.64	83.3	1.4	0.5
c2670	102	28.3	1566	2.14	739.3	1.7	1.1	0.56	104.3	1.7	0.5
c3540	350	48.9	1741	7.65	69.5	1.9	6.7	1.19	60.2	1.9	0.5
c5315	264	39.1	2608	6.99	924.2	1.6	2.7	0.91	83.3	1.6	0.5
c6288	46	18.8	2480	16.71	711.9	1.5	27.3	8.67	354.6	1.5	0.5
c7552	450	90.6	3827	23.41	2612.3	1.5	1.8	2.51	97.7	1.5	0.5
s5378sc	331	33.9	3221	5.54	2305.1	1.5	8.8	0.90	26	1.5	0.5

Ckt - Circuit; Vec - No. of Vectors; Err Vec - No. of Erroneous Vectors; Rgn - No. of Regions; T - Time in seconds; Cand - Candidate Regions; E - Error Fully Contained; R - Rank

Table 3: Comparison between Algorithms for Gate Connection Errors for Radius 2

Ckt	Vec	Err Vec	Rgn	Case 1				Case 2			
				T	Cand	E	R	T	Cand	E	R
c432	100	9.6	203	0.12	101	6.9	26.4	0.09	101	6.9	1
c499	184	29.4	275	0.59	140.3	9.7	35.4	0.40	106.7	9.7	1
c880	128	10.9	469	0.31	95.6	7.4	24.4	0.15	76.8	7.4	1
c1355	198	36.5	619	2.09	206.8	6.4	15.1	0.65	140	6.4	1
c1908	280	55.8	938	4.60	175.8	7.2	27.2	1.18	153.1	7.2	1
c2670	102	28.3	1566	3.32	798.6	9.1	26.4	1.14	185.7	9.1	1
c3540	350	48.9	1741	15.49	201.5	9.6	41.7	3.86	182.3	9.6	1
c5315	264	39.1	2608	12.32	1051.7	10.7	48.6	1.57	177.7	10.7	1
c6288	46	18.8	2480	25.88	1087.5	6.7	66.9	18.38	790	6.7	1
c7552	450	90.6	3827	44.45	2621.6	7.8	11.5	5.87	213.7	7.8	1
s5378sc	331	33.9	3221	7.46	2339	7.7	22.4	1.20	69	7.7	1

Ckt - Circuit; Vec - No. of Vectors; Err Vec - No. of Erroneous Vectors; Rgn - No. of Regions; T - Time in seconds; Cand - Candidate Regions; E - Error Fully Contained; R - Rank

case, the entire test set is needed to find the frame.

5 Experimental Results

The diagnosis procedures were implemented in C++; ISCAS85 and ISCAS89 benchmark circuits were used for our study. All experiments were performed on Sun ULTRA 10 workstations with 256MB of memory using STRATEGATE [13] test sets.

5.1 Diagnosis of combinational circuits

We performed diagnosis for gate connection errors, where extra gate connection and wire exchange errors were considered [12]. Tables 2 and 3 show the results for case 1 and case 2 algorithms, for regions of radius 1 and 2, respectively. For each circuit, the average over 10 runs with extra gate connection and wire exchange errors injected at random are listed. The circuit s5378sc is a full scan version of the ISCAS89 benchmark circuit s5378. The columns in the table represent for each circuit, the number of vectors in the given vector set, the number of erroneous vectors, the number of regions of radius 1 or 2, time taken in seconds (T), number of candidates returned that may contain the error (Cand), the number of candidate regions that fully contained the error injected (E) and the rank of the region containing the error (R) for both case 1 and case 2. The rank given to a candidate with the highest score was 1, indicating that the region is the most likely region in which the error resides. If the injected error was not fully contained in any region, a value of 0 was used when calculating the average over the 10 runs.

The regions considered in this analysis were overlapping and hence the number of regions is equal to the number of gates in the circuit. For example in circuit c5315, for radius 1, the total number of vectors in the set was 264, the average index number of erroneous vectors was 39.1, the total number of regions was 2608, for case 1, the average time taken was 6.99 sec, the average number of candidates found was 924.2, the average error hit was 1.6 and the average rank given to the region that contained the actual error was 2.7. On the other hand, for case 2, the time taken was 0.91 sec and only 83.3 candidates were returned with an error hit of 1.6 and average rank of 0.5. Because not all errors were fully containable in regions of radius 1, out of the 10 errors injected, 5 were contained within regions of radius 1 and the rest were contained in regions of radius 2. So the algorithms will return a rank of 1 for 5 instances and 0 for the rest resulting in an average rank of 0.5. One more important point to be noted is that even though the number of candidates returned in case 1 is less than in case 2, the accuracy is not sacrificed as can be seen from the equal numbers seen in the error hit column for both the methods. An observation was made that in-spite of injecting the error in the circuit, we

Table 4: Diagnosis of Sequential circuits with Single Stuck-at Errors using Regions of Radius 0

Circuit	**No. of Vectors**	**No. of Regions**	**Error Vector**	**Error Frame**	**Jump Num**	**Time (sec)**	**Cand**	**E**	**R**
s298	194	142	18.8	17.8	10.4	0.01	13.4	1	1
s344	86	195	13.6	12.1	8	0.02	8	1	1
s382	1486	188	91.6	82.3	54.4	0.01	9.5	1	1
s400	2424	194	77.1	70.6	50.4	0.02	7.7	1	1
s444	1945	211	125.6	116.8	51.2	0.02	9	1	1
s526	2642	223	46.6	44.5	21.6	0.02	5.8	1	1
s641	166	457	23.7	22.6	8.8	0.03	11.9	1	1
s713	176	470	28.8	28.4	8	0.04	14.4	1	1
s820	590	331	201.3	200.6	21.6	0.03	14.7	1	1
s832	701	329	172.7	171.9	20	0.02	16	1	1
s1196	574	575	70.3	69.8	8	0.04	12.9	1	1
s1238	625	554	85.4	84.2	8	0.04	12.6	1	1
s1423	3943	753	60.4	53.7	20.8	0.07	6.4	1	1
s1488	593	686	36.5	35.5	11.2	0.06	17.5	1	1
s1494	540	680	43.6	42.2	24	0.05	14.6	1	1
s5378	11481	3042	87.1	84.6	11.2	0.41	44.8	1	1
s35932	257	18148	17.3	13.6	8	13.52	967	1	1

Cand - Candidate Regions; E - Error Fully Contained; R - Rank

Table 5: Diagnosis of Sequential circuits with Multiple Stuck-at Errors

Ckt	**Vec**	**Rgn**	**Err Vec**	**Frm**	**JNUM**	**Radius 1**				**Radius 2**			
						T	**Cand**	**E**	**R**	**T**	**Cand**	**E**	**R**
s298	194	142	8.5	5.4	8.8	0.01	16.2	0.5	0.5	0.02	39.5	5.3	1
s344	86	195	6	5.3	8	0.02	13.9	0.6	0.5	0.03	32.2	4.7	1
s382	1486	188	38.7	29.7	20.8	0.02	26.1	0.6	0.5	0.03	62.1	9	1
s400	2424	194	30.5	26.3	8.8	0.02	25.4	0.6	0.5	0.03	64.1	7.1	1
s444	1945	211	23	15.5	13.6	0.02	21.5	0.6	0.5	0.03	61.1	8.3	1
s526	2642	223	36.5	31.2	10.4	0.03	27.7	0.6	0.5	0.05	72.8	7.7	1
s641	166	457	8.3	7	8	0.04	12.2	0.6	0.5	0.06	28.3	6.3	1
s713	176	470	10.8	10.4	8	0.04	23.1	0.5	0.5	0.07	44.7	6.9	1
s820	590	331	23.1	22.6	10.4	0.06	72.3	0.6	0.5	0.21	208.9	16.3	1
s832	701	329	39.9	38.4	10.4	0.06	65.4	0.6	0.5	0.22	181.1	24.7	1
s1196	574	575	18.9	17.4	8	0.07	43.9	0.6	0.5	0.14	125.8	6.3	1
s1238	625	554	9.4	8.2	8	0.06	44.6	0.5	0.5	0.16	147.7	6	1
s1423	3943	753	22.7	20.2	10.4	0.10	31	0.5	0.5	0.15	96.8	11.4	1
s1488	593	686	5.2	4.7	9.6	0.14	100.8	0.5	0.5	0.53	376.9	12.2	1
s1494	540	680	6.1	5.6	9.6	0.13	94.1	0.5	0.5	0.54	310.2	16.7	1
s5378	11481	3042	193.6	192.7	9.6	0.49	53.4	0.5	0.5	0.67	95.1	4.8	1
s35932	257	18148	10.7	8.3	8	17.7	1002	0.5	0.5	197.06	2294.9	160.1	1

Ckt - Circuit; Vec - No. of Vectors; Rgn - No. of Regions; Err Vec - Index No. of Erroneous Vector; Frm - Error Frame; JNUM - JUMPNUM; T - Time in seconds; Cand - Candidate Regions; E - Error Fully Contained; R - Rank

could get any number of erroneous vectors.

For radius 2, it can be seen that all the errors were fully diagnosed for case 2, since all errors were fully containable. It is clear from the rank column that for region-based algorithm, the candidate which fully contains the error is always given a rank 1. For both radii, diagnostic resolution was improved significantly. For example, in circuit s5378sc, the number of candidates was reduced from more than 2300 down to less than 100. Similar results are observed in other circuits as well. Not only did the case 2 algorithm show better diagnostic resolution than case 1, the execution time was also greatly reduced over the case 1 algorithm.

5.2 Diagnosis of sequential circuits

We have applied this diagnosis algorithm for sequential circuits for both single and multiple stuck faults using regions of radius 0, 1 and 2. Table 4 shows the results for diagnosis of sequential circuits with single stuck-at errors injected at random using regions of radius 0. The columns in the table represent for each of the circuit, average over 10 runs, the number of vectors in the given test set, the number of regions of radius 0 which is equal to the number of gates in the circuit, the vector at which the circuit failed, the time frame which was used for diagnosis, the JUMPNUM as explained earlier, the time taken in seconds, the number of candidate regions returned, number of regions fully containing the error and the best rank of the region containing the error. For example, in circuit s1423, the number of vectors in the test set was 3943, the total number of regions in the circuit was 753, the erroneous vector was 60.4, the time frame at which diagnosis was carried was 53.7, the JUMPNUM was 20.8, the time taken was 0.07 seconds, the number of candidates returned were 6.4 with error hit of 1 and rank of 1. From the results we can see that all the errors were diagnosed with very good accuracy as the reading of 1 in the error hit column indicates. The results show that JUMPNUM is usually small. For example, in circuit s820, the erroneous vector number was 201.3 on an average, the JUMPNUM was 21.6 and the time frame was 200.6. This means that we had to re-simulate only 21.6 out of 200 vectors on an average to excite and propagate the error to a flip-flop. We can observe that JUMPNUM for circuits s382, s400 and s444 are larger than other circuits and this may be due to the fact that the states may be difficult to achieve without re-simulating a larger number of vectors. For all circuits, the error was always diagnosed and the number of candidates were very small.

Table 5 shows the results for diagnosis of sequential circuits in the presence of *multiple* stuck-at errors using regions of radius 1 and 2. The table shows the average over 10 runs for each circuit with multiple stuck-at errors which varied from 2 to 4 in number, chosen at random. Out of the 10 errors injected, some may be contained within regions of radius 1 and the rest were contained in regions of radius 2. For radius 2, all the errors were diagnosed and since errors within radius 1 may be contained in multiple regions of radius 2, error hit is greater than 1. So this method works effectively for multiple errors in sequential circuits as well. We see that the JUMPNUM is smaller than in case of single error diagnosis. This is because the error effects with multiple errors is increased and it is easier to find a time frame which shows an error effect. The number of candidate regions returned for s35932 is large and this is because the starting candidate list is very large. If diagnosis can be carried over multiple time frames, better diagnostic resolution results, and this can be achieved by generating erroneous sequences differentiating implementation and the specification.

6 Conclusions

Two approaches for diagnosis using region-based model were studied for design errors. Results show the efficiency of the second approach over the first both in terms of time taken and number of candidates regions. We also applied the region-based model to diagnosis of gate connection errors, which is different from traditional stuck-type models. This strengthens the original premise that region-based model is not restricted to a specific fault like stuck-at or gate substitution and can handle arbitrary types of error. The region-based model was extended for diagnosis of sequential circuits by isolating the time frame at which the error was first observed. The isolation was achieved by re-simulating the minimum number of possible vectors, even if the vector at which the design failed was deep in the test set. Experiments show very good accuracy in diagnosis in the presence of single or multiple stuck faults.

References

[1] V. Boppana and M. Fujita, “Modeling the unknown! towards model-independent fault and error diagnosis”, in *Proc. Intl. Test Conf.*, 1998, pp. 1094–1101.

[2] V. Boppana, R. Mukherjee, J. Jain and M. Fujita, “Multiple error diagnosis based on xlists”, in *Proc. Design Automation Conf.*, 1999, pp. 100–110.

[3] A. Jain, V. Boppana, M. S. Hsiao, M. Fujita, “On the evaluation of arbitrary defect coverage of test sets”, in *Proc. VLSI Test Symp.*, 1999, pp. 426–432.

[4] A. Jain, V. Boppana, R. Mukherjee, J. Jain, M. S. Hsiao, M. Fujita, “Testing, verification, and diagnosis in the presence of unknowns,” in *Proc. IEEE VLSI Test Symp.,* 2000, pp. 263-269.

[5] S.-Y. Huang and K.-T Cheng, “ErrorTracer: design error diagnosis based on fault simulation techniques”, in *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 1341–1352, Sept. 1999.

[6] M. Tomita and Hong-Hai Jiang, “An algorithm for locating logic design errors”, in *Proc. Intl. Conf. Computer-Aided Design*, 1990, pp. 468–471.

[7] M. Tomita, T. Yamamoto, Sumikawa F and K. Hirano, “Rectification of multiple logic design errors in multiple output circuits”, in *Proc. Design Automation Conf.*, 1994, pp. 212–217.

[8] I. Pomeranz and S. M. Reddy, “On diagnosis and correction of design errors”, in *Proc. Intl. Conf. Computer-Aided Design*, 1993, pp. 500–507.

[9] I. Pomeranz and S. M. Reddy, “On error correction in macro-based circuits”, in *Proc. Intl. Conf. Computer-Aided Design*, 1994, pp. 568–575.

[10] P-Y. Chung, Y-M. Wang and I. N. Hajj, “Diagnosis and correction of logic design errors in digital circuits”, in *Proc. Design Automation Conf.*, 1993, pp. 503–508.

[11] H-T. Liaw, J-H Tsaih and C-S. Lin, “Efficient automatic diagnosis of digital circuits”, in *Proc. Intl. Conf. Computer-Aided Design*, 1990, pp. 464–467.

[12] M. S. Abadir, J. Ferguson and T. E. Kirkland, “Login design verification via test generation”, in *IEEE Transactions on Computer-Aided Design*, Vol. 7, No. 1, January 1988, pp. 138–148.

[13] M. S. Hsiao, E. M. Rudnick and J. H. Patel, “Sequential circuit test generation using dynamic state traversal”, in *Proc. IEEE Euro Design and Test Conf.*, 1997, pp. 22–28.

[14] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital System Testing and Testable Design,* New York, NY: Computer Science Press, 1990.