1992

DynaTAPP Dynamic Timing Analysis With Partial Path Activation in Sequential Circuits

Prathima Agrawal
AT&T Bell Laboratories

Vishwani Agrawal
AT&T Bell Laboratories

Sharad C. Seth
University of Nebraska-Lincoln, seth@cse.unl.edu

European Design Automation Conference, 1992. EURO-VHDL '92, EURO-DAC '92.
doi: 10.1109/EURDAC.1992.246252

DynaTAPP: Dynamic Timing Analysis With Partial Path Activation in Sequential Circuits

Prathima Agrawal and Vishwani D. Agrawal
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Sharad C. Seth*
Dept. of Computer Science and Eng.
University of Nebraska
Lincoln, NE 68588

Abstract – This paper gives a method of finding all sensitizable paths in a non-scan synchronous sequential circuit. Path activation conditions of the circuit are mapped onto a single stuck type fault by adding a few modeling gates to the netlist. Only if the corresponding stuck type fault is found detectable by a sequential circuit test generator is the path considered sensitizable. A depth-first analysis of circuit topology, that determines all paths between primary inputs, primary outputs and flip-flops, employs a partial path hierarchy. Thus, all paths with a common unsensitizable segment need not be examined separately. Results on benchmark circuits show that (1) the number of sensitizable paths can be significantly smaller than that found by a static timing analyzer and (2) the partial path analysis adds to efficiency when the number of sensitizable paths is less than 20 percent.

1. Introduction

Since early 1980s, static timing analysis has been routinely used by designers. The tools developed for internal use in large corporations [1] and others commercially available have been used to design thousands of VLSI circuits. In static timing analysis, the total delay of a path in the circuit is analyzed without regard to the sensitizability of the path. Designers often complain about the voluminous path data produced. Clearly, many paths, reported as *failing* by the static analysis, are non-functional and their consideration can lead to unnecessary overdesign.

A path delay is considered relevant to the proper functioning of the circuit if a rising or a falling signal can be propagated through the path and the result at the destination of the path can be observed. Both delay test generation and path analysis problems have been studied extensively for combinational circuits where the sources and destinations of paths are fully accessible. In this paper, we present a dynamic path analysis technique for non-scan synchronous sequential circuits. Thus, for a path of interest, there must exist an input vector sequence that will set up a transition at the origin (primary input or a flip-flop), propagate the transition through the path to its destination (a flip-flop or a primary output), and if the destination is a flip-flop, propagate its latched state to an observable output. If a path is sensitizable, the delays of elements along the path can be added up to estimate the path delay.

The dynamic timing analysis differs from delay testing. For ease of delay-fault test generation, the system clock is assumed to be slower during initialization and fault propagation phases and runs at the rated speed only during the path activation phase [2, 3]. Such an operation, though useful in test generation and fault diagnosis, differs from the normal operation of the circuit. In timing analysis, we assume the clock runs at rated speed throughout.

The result of both test generation and timing analysis should be independent of any gate delays. Testing requires that the circuit output must produce the fault effect for any set of arbitrary delays in the circuit. The requirements for a test sequence can be broken down into a *necessary* condition (the signal transition produces a fault effect at a circuit output) and a *sufficient* condition (the transition has no hazard). For timing analysis, on the other hand, as long as there exists a possible set of delays in the circuit that will make the primary output sensitive to the path delay, the path is considered relevant. Therefore, only the necessary condition is used. Obviously, the vector sequences generated as a byproduct of our timing analysis only verify the existence of functional paths but may otherwise be non-robust tests for verifying path delays.

2. Paths in a Non-Scan Circuit

In a synchronous sequential circuit, the boundaries of combinational logic consist of primary inputs, primary outputs and flip-flops. All flip-flops are synchronized by a common clock signal of some given frequency (or period). Proper operation requires that any signal changes, occurring at the inputs of the combinational logic and propagating to the outputs, must do so within the clock period. The relevant paths will all be sensitizable from inputs to outputs of the combinational logic. For each path, we have two potential cases based on the times that rising and falling transitions take to propagate through the path.

Sensitizability analysis requires the initialization of the circuit to an appropriate state such that a transition can be propagated through the path and the resulting transition captured in the destination flip-flop. If the delay of the path exceeds the clock period, then the state of the destination latch will be incorrect. This state must be observable at a primary output. In order to apply and propagate a transition through a path, we require two vectors applied to the combinational logic. Following Lin and Reddy [4], we specify the combination of signal values during the application of these two vectors as: $u0 = x0$, $u1 = x1$, $s0 = 00$, $s1 = 11$, $R = 01$, $F = 10$ and $X = xx$. The hazard conditions associated with signals, as discussed elsewhere [2], do not concern us here.

* Supported during this work by AT&T Bell Laboratories and by a Collaborative Research Grant from NATO.

The signals on the path assume R and F values. The *off-path signals* feeding the gates in the path assume values among $u0$, $u1$, $s0$ and $s1$. During the second vector, the off-path signals must sensitize the path. However, in the first vector, the off-path signals are left in the *don't care* state if the path signal applies the controlling value (0 for AND and NAND, 1 for OR and NOR). If the signal arriving at the destination flip-flop is a rising transition, the correct value latched in this flip-flop will be 1. Whenever the path delay exceeds the rated clock period, a 0 will be latched. Thus the state of the destination flip-flop can be denoted by D, which has the same meaning as in D-algorithm [5]. Similarly, for a falling transition arriving at the destination flip-flop, the state will be $\bar{D}$.

3. Path Activation Analysis

Given a path and a transition at its source, we create a modified circuit in which a test for a specified single stuck fault will activate the path. The fault must be such that a D ($\bar{D}$) value is injected into the destination flip-flop whenever the two path activation vectors produce a falling (rising) transition at the input to the flip-flop. Otherwise, the destination flip-flop should be set according to the normal operation of the circuit.

Figures 1 and 2 implement the above requirements. For the path *ace* (shown in bold) between flip-flops FFS and FFD, we insert the circuit enclosed within dashed lines. The signal requirements for a falling (rising) transition at the destination flip-flop are captured by the gates *AND1F (AND1R)* and *AND2F (AND2R)*. The output of *AND1F (AND1R)* feeds a flip-flop (*FF1* initialized to 0) whose output feeds *AND2F (AND2R)*. *AND2F (AND2R)* is 1 iff the preceding and the current vectors launch a falling (rising) transition at the input to the destination flip-flop. The *AND2F (AND2R)* output signal is used to gate a D ($\bar{D}$) into the destination flip-flop according to the above requirement. When the *AND2F (AND2R)* output is 0, the circuit remains unaffected by the added logic.

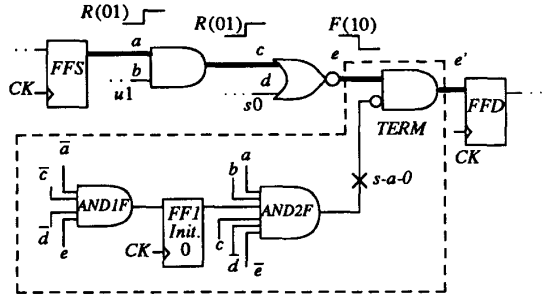


Fig. 1 Logic model for path with falling transition at destination.

The above model can be incorporated into a sequential circuit test generator program by modifying the netlist according to the path under consideration [2]. If the fault in Figure 1 (Figure 2) is not detectable, we can conclude that a falling (rising) transition on the selected path cannot be observed at a primary output of the circuit, hence the path need not be considered in timing analysis.

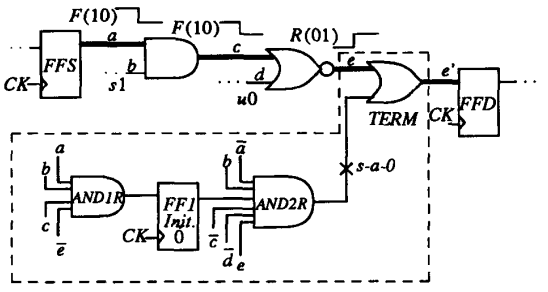


Fig. 2 Logic model for path with rising transition at destination.

Paths are considered *one* at a time. Thus, only two gates and one flip-flop are added to the netlist that is supplied to the test generator. The connectivity of these gates is changed automatically by the program for each run of the test generator.

4. Implementation

We have implemented a dynamic timing analysis system, DynaTAPP. The program modules are similar to those used in a path delay test generator [2].

The main algorithms are shown in Fig. 3. There are three path generation functions, *newpath*, *forward*, and *back*. From the netlist, an input routine (not shown) builds the internal circuit structure in the form of a bipartite graph whose nodes represent the gates and the signals (wires). Source and destination wires are marked in this graph. A new path is generated each time the function *newpath* is called. Thus, DynaTAPP repeatedly calls *newpath* until all paths are *done*. Within *newpath*, the search for paths moves under the control of *forward* and *back*. The algorithm *forward* traces the circuit topology in the forward direction until a partial or a full path destination is reached. It then calls *sensitize_path* that performs the analysis of Section 3 using the Steed [6] test generator. *Forward* returns a partial or a full path with its sensitization status.

When *newpath* is called such that the previous path was either a full path or not sensitizable, *back* is called to move back to a node with unused forward paths. If the backward movement returns to a source, *back* will initialize to another source node unless all sources have been exhausted. When a partial path is found to be sensitizable, *newpath* calls *forward*. The preprocessor, mark_nodes, counts the number of paths from any gate to all destinations. For a gate g, this number is denoted by $N(g)$. This information is helpful for the partial path analysis. Also, the data is used to sort the fanout list of each gate output in the decreasing order of path counts.

Experience shows that a large number of potential paths cannot be sensitized during the normal operation of a VLSI circuit. The idea of hierarchically analyzing segments of combinational paths has been used to advantage both for timing analysis [7] and for delay test generation in combinational circuits [8]. We define a *partial path* as a

```
mark_nodes( ) /* mark nodes with number of forward paths */
newpath( ) /* generate next sensitizable path */
{
  if (first_call) {
    initialize; /* create a partial path of first source wire */
    forward( );
  }
  else
  if (not_done) {
    if (last path is partial and sensitizable)
      forward( );
    else
      back( );
    return(path);
  }
}
forward( ) /*go forward to a partial or full path destination*/
{
  if (current wire not a destination)
    forward_one( ); /* move forward to the next wire */

  while (current wire not a partial or full path destination)
    forward_one( );

  sensitize_path( );
}
back( ) /* go back until a wire with unused fanout */
{
  if (the current wire is not a source)
    back_one( ); /* move back to the previous wire */
  while (current wire not source AND no unused fanouts)
    back_one( );
  if (current wire has unused fanouts)
    forward( );
  else /* at a source node whose all paths have been traced */
    if (there are unprocessed source nodes) {
      initialize;/*create a partial path with next source wire*/
      forward( );
    }
    else
      done = TRUE;
}
```

Fig. 3 Path generation algorithms in DynaTAPP.

path from a source (primary input or flip-flop) to any gate that is not a primary output or a flip-flop input. Thus, if a partial path terminating on gate g is not sensitizable, then so are the $N(g)$ paths that include this partial path.

Suppose the probability of sensitizing a partial path to gate g is p. If this partial path is found to be unsensitizable, then a single run of path sensitization analysis saves us $N(g)$ runs which we would need without the partial path analysis. Thus, the saving in the runs of sensitization analysis due to a failed partial path is $N(g)-1$. Since the average number of failing paths to g is $1-p$, we estimate the average saving of $(1-p)(N(g)-1)$ runs due to the partial path analysis. On the other hand, if the sensitization analysis finds the partial path to be sensitizable, then we incur a cost of one extra run. The average cost is p runs. A proper selection of partial paths should ensure that *saving* > *cost*. Therefore, we get

$$N(g) \geq \frac{1}{1-p}$$

For very small values of p, we notice that any value of $N(g)$ greater than 1 is beneficial. However, for $p \approx 1$, $N(g)$ should be very large.

In circuits where most paths are sensitizable, i.e., $p \approx 1$, the analysis of partial path will not be beneficial unless $N(g)$ is very large. Notice that $N(g)$ depends only on the circuit fanout structure and is easily determined. The benefit of partial path analysis is easier to realize in circuits where a small fraction of paths is sensitizable. However, to maximize the benefit, a partial path should have a large $N(g)$ and a small p. If we assume that the sensitization probability of a path decreases with its length then a partial path should not be too short. A good heuristic is to check for partial path sensitization only if the *product* of path length and $N(g)$ exceeds some threshold.

The partial path heuristic employed in DynaTAPP differs slightly from the above suggestion and works without a threshold value. We monitor the product during the path search. A drop in the product value signals the need to check for sensitization of the partial path up to (but not including) the node at which the drop occurs.

5. Results

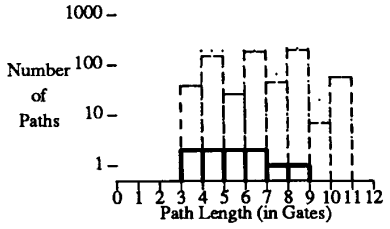
Table 1 shows the results of DynaTAPP. For each physical path, two cases of rising and falling transitions were considered. Thus, the number shown as *Total Paths* is twice the number of physical paths. The *Dynamic Paths %* is the percentage of paths that were found sensitizable. The program provides a vector sequence for each sensitizable path. Since our sensitization analysis does not consider *combinational hazards*, the sequence is not a *robust* test. In the present context, we only analyze the *necessary* condition for sensitization.

Table 1 – DynaTAPP Results

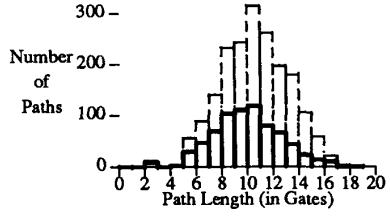
Circuit	Paths		Partial Path Analysis		Full Path
Name	Static	Dynamic %	Full+Partial	CPU s	CPU s
s27	56	37.5	54 + 9	2.6	2.2
s208	290	16.6	141 + 40	11.7	18.8
s298	462	20.8	388 + 48	144.2	150.4
s344	710	25.9	625 + 170	1302.5	1155.9
s349	730	25.2	633 + 175	1296.3	1168.1
s382	800	1.3	309 + 79	1687.7	34962.6
s420	738	6.4	194 + 75	122.2	334.0
s444	1,070	5.6	309 + 79	5180.0	14209.6
s510	738	23.4	708 + 110	126.4	113.6
s526	820	3.8	312 + 72	19915.0	42521.2
s526n	816	3.8	312 + 72	20012.4	42564.4
s820	984	37.4	945 + 144	2970.0	2952.0
s832	1,012	36.4	993 + 159	3625.6	3162.6
s953	2,266	40.2	2228 + 560	1550.7	1287.7
s1488	1,924	37.5	1831 + 419	1053.7	901.3
s1494	1,952	37.0	1857 + 432	1085.6	928.8

The CPU time of DynaTAPP was measured on SUN Sparc 2. For a given circuit, the run time is roughly proportional to the number of paths that are analyzed. For example, for the circuit s208, a total of 181 (141 full and 40 partial) paths were analyzed. Without the partial path

analysis, we would analyze all 290 paths. For s344, however, the partial path analysis processes more paths (a total of 625 + 170 = 795) as compared to the 710 actual paths. We may conclude that our analysis is faster whenever the fraction of sensitizable paths is below 20%. This result is dependent on the heuristic used in the program and there is scope for improvement.



(a) s382



(b) s1488

Fig. 4 Path lengths obtained by DynaTAPP (Non-Scan: Solid; Scan/Hold: Dashed) and Static analysis (Dotted).

In general, simulators accumulate estimated values of rise and fall delays for gates and propagation delays for interconnections to determine path delays. However, for simplicity, we will assume that the path length is given simply by the number of gates in a path. Figure 4 shows the histograms of path data for the circuits s382 and s1488. The dotted lines show the data from *static* analysis, obtained by DynaTAPP without using *sensitize_path*. The dashed lines are the results of running the program on the *combinational* part of the circuits. The solid lines show the results for non-scan sequential mode. Notice that for s1488, the static and the combinational data almost overlap and, for both circuits, the longest path length is unchanged under these two categories. The non-scan circuits show a marked reduction in the number of paths. The longest path is also shortened by two gates in both circuits.

The results clearly show that the reduction in the number of functional paths is significant for non-scan circuits. We found that the unsensitizable paths were impossible to activate in the sequential mode and are not due to the inability of the test generator to find a test. Further experiments showed that the number of paths activaed by another set of input vectors, generated to have a high stuck type fault coverage, can be even lower than that determined by our timing analysis. To our knowledge, such results have not been reported before and should help improve the design and test methodology.

Since *sensitize_path* generates an input sequence that sensitizes the path, circuit level simulation can be readily used for accurate verification. Use of functional paths in layout-level timing optimization [9] will lead to more meaningful result. We must caution, however, that the vectors generated by *sensitize_path* may not be appropriate for path delay testing since the hazards that depend upon the actual delays of gates can invalidate the testing of a path [2]. Just the hazard-free or robust delay tests will not be sufficient for testing if the path coverage is low, since the stuck-type coverage is likely to be unacceptably low. The fact that in some circuits only a small fraction of paths is sensitized needs further investigation.

6. Conclusion

This is perhaps the first time potentially sensitizable paths of sequential circuits have been analyzed by an automated analysis tool. The results show that the *false path* problem may be more significant in non-scan circuits than in combinational circuits. Due to the low sensitizability of paths in the sequential mode of operation, partial path analysis can provide greater efficiency. Our path analysis is based on a necessary condition of path delay testing.

References

[1] V.D. Agrawal, "Synchronous Path Analysis in MOS Circuit Simulator," *Proc. 19th Design Autom. Conf.*, pp. 629-635, June 1982.

[2] P. Agrawal, V.D. Agrawal, and S.C. Seth, "A New Method for Generating Tests for Delay Faults in Non-Scan Circuits," *Proc. 5th Intl. Conf. VLSI Design*, pp. 4-11, January 1992.

[3] T.J. Chakraborty, V.D. Agrawal, and M.L. Bushnell, "Delay Fault Models and Test Generation for Random Logic Sequential Circuits," *Proc. Design Autom. Conf.*, June 1992.

[4] C.J. Lin and S.M. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. CAD*, Vol. CAD-6, pp. 694-701, September 1987.

[5] J.P. Roth, W.G. Bouricius, and P.R. Schneider, "Programmed Algorithms to Compute Tests and to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Trans. Electronic Computers*, Vol. EC-16, pp. 567-580, October 1967.

[6] A. Ghosh, S. Devadas, and A.R. Newton, "Test Generation and Verification for Highly Sequential Circuits," *IEEE Trans. CAD*, Vol. 10, pp. 652-667, May 1991.

[7] J. Benkoski, E.V. Meersch, L. Claesen, and H. DeMan, "Efficient Algorithms for Solving the False Path Problem in Timing Verification," *Proc. Int'l Conf. Computer Aided Design*, pp. 44-47, November 1987.

[8] F. Fuchs, F. Fink, and M.H. Schulz, "DYNAMITE: An Efficient Automatic Test Pattern Generation System for Path Delay Faults," *IEEE Trans. CAD*, Vol. 10, pp. 1323-1335, October 1991.

[9] A.E. Dunlop, V.D. Agrawal, D.N. Deutsch, M.F. Jukl, P. Kozak, and M. Wiesel, "Chip Layout Optimization Using Critical Path Weighting," *Proc. 21st Design Autom. Conf.*, pp. 133-136, June 1984.