Dynamical Identification of Critical Paths for Iterative Gate Sizing *

How-Rern Lin and TingTing Hwang

Department of Computer Science, National Tsing Hua University
Hsinchu, Taiwan 30043, R.O.C.

Abstract

Since only sensitizable paths contribute to the delay of a circuit, false paths must be excluded in optimizing the delay of the circuit. Just identifying false paths in the first place is not sufficient since during iterative optimization process, false paths may become sensitizable, and sensitizable paths false. In this paper, we examine cases for false path becoming sensitizable and sensitizable becoming false. Based on these conditions, we adopt a so-called loose sensitization criterion [ChD91] which is used to develop an algorithm for dynamically identification of sensitizable paths. By combining gate sizing and dynamically identification of sensitizable paths, an efficient performance optimization tool is developed. Results on a set of circuits from ISCAS benchmark set demonstrate that our tool is indeed very effective in reducing circuit delay with less number of gate sized as compared with other methods.

1 Introduction

Performance optimization is commonly involved in design process to enforce the satisfaction of long path constraints. The delay of a combinational circuit is the delay of the longest sensitizable path in the circuit. False paths must be excluded in optimizing the delay of the circuit. Studies on path sensitizability problem [DYG89, McB89, DKM91, ChD91, BMG87] are abundant in the literature in which various sensitizability criteria were proposed to eliminate the false paths in a circuit.

Just identifying false paths in the first place is not sufficient since during optimization process, false paths may become sensitizable. Therefore, two approaches [JoF93, CDL91, HPS93] incorporating the optimization effect into false paths identification were proposed. In the first approach [JoF93], an iterative optimization scheme is adopted to the effect that sensitizability analysis and gate sizing routines are called iteratively. To avoid iterative sensitizability analysis, the second approach [CDL91, HPS93] performs sensitizability analysis only once and reports a set of paths for optimization. The path set includes not only sensitizable paths but also some false ones which may become sensitizable and dominate the circuit delays during optimization. Figure 1 depicts the flow of these two methods.

However, we find that not only the false paths may become sensitizable, the sensitizable paths may also become false during performance optimization phase.

*Supported by a grant from the National Science Council of R.O.C. under contract no. NSC-82-0404-E-007-129

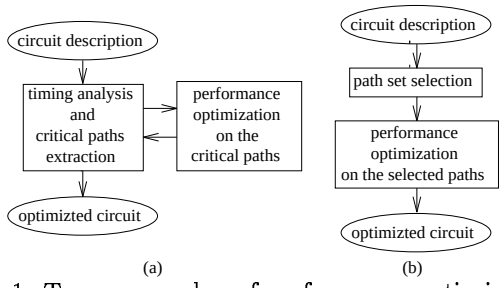

Figure 1: Two approaches of performance optimization.

Consider the example circuit shown in Figure 2. For simplicity, assume the rising/falling delay of AND, OR, NAND, NOR gates are 4, and inverter is 1. Consider the four logical paths $P_1 = \overline{i_2} - g_1 - s_1 - g_2 - s_2 - g_3 - \overline{s_3}$, $P_2 = \overline{i_2} - g_1 - s_1 - g_6 - s_6 - g_7 - \overline{s_7}$, $P_3 = i_1 - g_4 - s_4 - g_5 - s_5 - g_3 - \overline{s_3}$ and $P_4 = i_1 - g_4 - s_4 - g_5 - s_5 - g_7 - \overline{s_7}$ of the example circuit. In order to allow signal $\overline{i_2}$ propagate to output node s_3 along path P_1, we must set signal $i_1 = 1$ as a *non-controlling value* to g_2. Signals s_4 and s_5 will stable with logic value 1 since i_1 is set to logic value 1. As a result, s_2 and s_5 are both with the *non-controlling value* of gate g_3. However, the signal cannot propagate from s_2 to s_3 since the stable time of signal s_5 is later than signal s_2. That is P_1 is a false path. Similarly, in order to allow signal i_1 propagate to output node s_7 along path P_4, we must set signals $i_2 = 0$ as a *non-controling value* to g_5. Signals s_1 and s_6 will both stable with logic value 1 since i_2 is set to logic value 0. Therefore, s_5 and s_6 are both with the *controlling value* of gate g_7. However, the signal cannot not propagate from s_5 to s_7 since the stable time of signal s_6 is earlier than signal s_5. That is, P_4 is a false path. P_2 is sensitizable if signal $i_1 = 1$ is set. Similarly, P_3 is sensitizable if signal $i_2 = 0$ is set. However, if the delay of g_4, and g_5 are both reduced from 4 to 2, the sensitizable paths P_2 and P_3 become false while the false paths P_1 and P_4 become sensitizable.

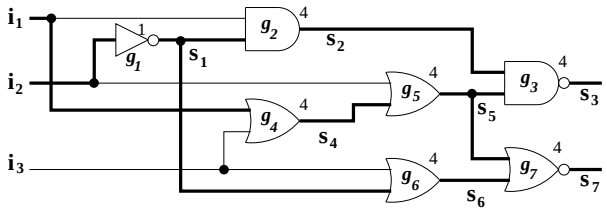

Figure 2: An example circuit.

Moreover, we observe that some false paths can n-

ever become sensitizable during timing optimization, e.g., one that results from incompatibility in propagation conditions, from the existence of redundancy in a circuit, from designs for special features such as carry lookahead adder etc. We refer to these type of false paths as *function-false paths*. To increase the efficiency of optimization process, *function-false paths* should be distinguished from other false paths and put aside since they can never become sensitizable.

Based on the above observation, we examine the conditions for sensitizable path becoming false and false path becoming sensitizable and adopt a so-called loose sensitization criterion [ChD91]. With the criteria, a performance optimization algorithm is developed to dynamically update sensitizable paths. In our algorithm, *function-false paths* are eliminated first. The circuit is then optimized by selecting a gate and replacing it by a faster template from cell library iteratively until the performance requirements are met. In order to ensure that the optimization effort is made on the real critical paths, the critical paths are dynamically identified in each iteration. To avoid a full-scale sensitizability analysis during each iteration, data structures are used to keep necessary information for identification of sensitizable paths.

The remaining of this paper is organized as follows. In Section 2, we review the definitions of path sensitization. The conditions for a false path becoming sensitizable and a sensitizable path becoming false are presented in Section 3. To avoid the time consuming sensitizability analysis, some data structures are utilized. Section 4 discusses our performance optimization algorithm. In Section 5, we give benchmark results on a set of circuits from ISCAS benchmark set.

2 Path sensitization

In order to ensure optimization effort is made on the real critical paths, sensitizable paths must be identified. The sensitization criteria is basically taken from [DYG89] with modification in which rising transition and falling transition are dealt with separately.

A sensitizable path is a path that can be activated by at least one input vector. The output transition of a logic gate g is decided by either the earliest input signal with controlling value of g, if any, or the latest input with *non-controlling value* of gate g if all input signals with *non-controlling value*.

Definition 2.1: A path $P = s_0 - g_0 - s_1 - g_1 - ... - s_{k-1} - g_{k-1} - s_k$ in a circuit C is a *sensitizable path* if and only if there exists at least one input vector I such that all signals along path P satisfy the condition,

$$T_S(s_i, I, C) = PD(s_i, P, C), \ for \ 0 \leq i \leq k,$$

where path delay $PD(s_i, P, C)$ of signal s_i is mere summation of gate and signal delays, from s_0 up to s_i, on path P in circuit C and $T_S(s_i, I, C)$ is the signal stable time of s_i by applying input vector I to the primary inputs at time 0.

Definition 2.2: A path P which is not a *sensitizable path* is a *false path*.

3 Dynamical Path Sensitization Criterion

During the iterative gate resizing, a sensitizable path may change to false and vice versa. However, only sensitizable paths contribute to the delay of a circuit. In order to ensure the optimization effort is made on the real critical paths, dynamically identifying the critical paths is required.

We first show the conditions for a false path becoming a sensitizable one. Let C and C' denote the circuits before and after one iteration of gate sizing, respectively. Let $P_{i+1} = s_0 - g_0 - ... - s_i - g_i - s_{i+1}$ be a false partial/complete path, and $P_i = s_0 - g_0 - ... - g_{i-1} - s_i$ be a subpath of P_{i+1} and a sensitizable path in circuit C. Two cases change false path P_{i+1} in C to sensitizable one in C'.

case 1. s_i with a *controlling value* of gate g_i.
This case may occur when a gate sitting on path P_i, rather than on any other path reaching the side inputs of gate g_i, is sized.

case 2. s_i with a *non-controlling value* of gate g_i.
This case may occur when a gate sitting, except on path P_i, on any path reaching the side inputs of gate g_i is sized.

The conditions for a sensitizable partial/complete path becoming a false one is shown as follows. Let $P = s_0 - g_0 - ... - s_i - g_i - ...s_{k-1} - g_{k-1} - s_k$ be the path which is sensitizable in circuit C and false in C', and both $P_i = s_0 - g_0 - ... - g_{i-1} - s_i$ and $P_{i+1} = s_0 - g_0 - ... - g_i - s_{i+1}$ be subpaths of P. Two cases change sensitizable path P in C to false one in C'. Suppose that path P be blocked at gate g_i in circuit C'.

case 3. s_i with a *controlling value* of gate g_i.
This case may occur when a gate sitting, except on path P_i, on any path reaching the side inputs of gate g_i is sized.

case 4. s_i with a *non-controlling value* of gate g_i.
This case may occur when a gate sitting on path P_i, rather than on any other path reaching the side inputs of gate g_i, is sized.

The above-mentioned four cases give us the exact sensitization criteria to identify critical paths for sizing in each optimization iteration. However, certain paths change from false to sensitizable and from sensitizable to false frequently during the optimization process. If the exact criteria is used, it may results in using more hardware for a specified constraint. Consider the example circuit shown in figure 1 where path P_1 is false and path P_3 is sensitizable in the original circuit. Suppose that the required time of output node s_3 is set to 8. Let both the delay of gates g_4 and g_5 on path P_3 are reduced from 4 to 2 in the first iteration. As a result, path P_1 becomes sensitizable while P_3 becomes false. In a later iteration, to satisfy the timing constraint, the delay of gate g_2 on path P_1 is reduced from 4 to 2. Path P_1 becomes false again and path P_3 becomes sensitizable. To satisfy the timing constraint, three cells may be needed for sizing. If these two paths are both included in the path set for sizing, the gates along the intersection of the two paths, such as gate g_3, will be chosen for sizing and the delay of two paths are reduced simultaneously. Consequently, only two cells may be chosen for resizing. Therefore, in each iteration, the inclusion of some false paths in the critical path set could lead to a better decision in cell selection.

The paths whose change shuttles between false and sensitizable should be included in path set for sizing. To detect this type of false paths, we define the follow term.

Definition 3.1: Let $P = s_0 - g_0 - s1 - g_1 - ... - s_{k-1} - g_{k-1} - s_k$ be a false path in circuit C, and s_i is a signal along path P. Signal s_i is defined as a *false point* if

$PD(s_i, P, C) \neq T_S(s_i, I, C)$, for a given input vector I.

Definition 3.2: A path $P = s_0 - g_0 - s1 - g_1 - ... - s_{k-1} - g_{k-1} - s_k$ is a *shuttle path* in circuit C if and only if P is a delay-false path and there exists at least one input vector I such that for all *false points* s_i along path P, the stable value of s_{i+1} under input vector I is the *non-controlling value* of g_i.

The *shuttle paths* are those paths whose change shuttles most frequently between sensitizable and false. They should be included in the critical path set for sizing. Based on the above discussion, the criteria for identifying the critical paths in each iteration of gate sizing is presented as follows.

Dynamical path sensitization criterion: Given a path $P = s_0 - g_0 - s1 - g_1 - ... - s_{k-1} - g_{k-1} - s_k$. Path P is a critical path if and only if the path delay of P is longer than the given delay constraint and there exists at least one input vector I such that for each signal s_i along P satisfies the following conditions :
(1) if s_i is the *controlling value* of gate g_i, and for all $s_j \in fanin(g_i)$ and s_j is the *controlling value* of g_i, then $PD(s_i, P, C) \leq T_S(s_j, I, C)$, or
(2) if s_i is the *non-controlling value* of gate g_i, then the stable value of all input signals of gate g_i is the *non-controlling* value of gate g_i.

This sensitization criterion is classified as a looser one, $SENV_{loose}$, in [ChD91].

4 Performance optimization algorithm

Our performance optimization algorithm is presented in this section, which is a two-phase procedure. Phase one is to identify function-false paths from the long paths and extract critical paths. If the critical path set is not empty, the algorithm proceeds to phase two. Phase two is an iterative procedure. In each iteration, a gate is selected and replaced by a fast template from cell library. The critical path set is then updated. If the timing constraint is still not satisfied, the iteration restarts. Figure 3 shows the flow of our algorithm.

```
Algorithm performance_optimization(circuit, delay_constraint)
  Phase 1 :
     function-false paths elimination ;
     critical path set extraction;
  Phase 2 :
     while (delay constraints are not satisfied)
        selected_gate = gate_selection(critical_paths);
        if (selected_ gate = ∅ )
           report failure to meet delay constraints;
           exits ;
        endif
        change the template of the selected_gate;
        dynamically updating the critical path set;
     endwhile
endalgorithm
```

Figure 3: performance optimization algorithm

In the following, we give a more detailed description of each procedure.

Functional false path elimination :
Functional false paths are those paths resulting from incompatibility in propagation conditions, from the existence of redundancy, from design for special features such as carry lookahead adder, etc. These paths can never become sensitizable during time optimization. Therefore, to increase the efficiency and accuracy, *function-false paths* should be distinguished from other false path and put aside during the iteration process. Functional false paths are detected by a procedure similar to D-algorithm.

Critical path extraction :
Critical paths are extracted using the sensitization criterion presented in Section 3. A procedure similar to the false path detection algorithm of [DYG89] is designed.

Gate selection :
The sensitivity of a gate, which is defined as the delay savings per increment in area, is used as a selection criterion. Moreover, in order to gain the speed-up of multiple paths simultaneously by sizing only a single gate, the gate passed by many critical paths should be selected. Therefore, we define the following *gain* function to reflect the sensitivity of a cell and the degree of simultaneous speed-up of multiple paths.

$$gain(g_i) = sensitivity(g_i) \times \sum_{\forall P \in CP_i} slack(P),$$

where CP_i is the set of critical paths passing gate g_i and $slack(P)$ is the difference of the required time and the actual arrival time of path P. The gate with largest gain value is selected for resizing.

Dynamically updating critcal path set :
The critical path set is updated based on the criterion proposed in Section 3. It is inefficient to update the critical paths by a full-scale critical path analysis in each iteration. Since delay-change occurs only in a small portion of the entire circuit after sizing a gate, the updating of critical path set is confined to the paths passing this region. Let g_i be the gate resized in an iteration. *Influenced gates* are referred to the gates s_j that either the output of s_j fan out to s_i or there exists at least one signal path from s_i to s_j. Only the paths passing through the *influenced gates* are checked if there is any change in timing.

One more problem needs to be discussed as to recording the critical path set for dynamically updating. Since there are a large number of paths in a circuit, it is impractical to enumerate all of them. Instead, we store a path by three values : the primary input, the level, and the path delay of this path. For a critical path that is a complete one, the values are stored in the corresponding primary output node. For a false path that is a partial one, the values are stored in the first *false point*. A path can be easily traced back by these three values.

5 Experimental Results

The algorithm *performance_optimization* described in the previous section has been implemented as DYNA (DYNAmical identification of critical paths for iterative gate sizing) in C language on a SUN SPARC-10 workstation. Benchmarking process is performed on circuits from the ISCAS benchmark set. First, the circuits are technology mapped to a standard cell library using MIS mapper [Bra87] with fanout optimization option. Without considering path sensitizability, the maximum path delay (*maximum_path_delay*) of the mapped circuit is extracted to be used for setting timing constraints. The delay constraints are set to $R \times maximum\_path\_delay$, where different R is set for different cases.

The benchmarking process is conducted in three parts. In order to investigate the number of *function-false paths* in a circuit, the *function-false paths* are extracted according to different delay constraints. $R = 0.9, 0.8$ and 0.7 are set for different delay constraints. First, the long paths in a circuit are extracted from which the *function-false paths* are identified. Table I shows the results. For some circuits, the number of *function-false paths* is very large, e.g., circuits c432, c1355 and c5315. Moreover, the number of long *function-false paths* grows rapidly when a tighter delay constraint is set.

Table I Long function-false paths.

circuit	R=0.9	R=0.8	R=0.7
c432	201737	457449	782978
c499	80800	288144	521040
c880	0	66	184
c1355	1596608	5692016	10031824
c1908	83460	441855	893450
c2670	14125	51560	92313
c5315	192471	1137697	2768251

The second experiment is to optimize a circuit under various delay constraints. In addition to DYNA, two other algorithms, FPE (False Path Elimination) and P-S (Path Set), are implemented. The FPE is an iterative gate sizing tool. In each iteration, the critical paths are extracted by eliminating the false paths from the long paths using the false path detection algorithm in [DYG89]. Two major differences between DYNA and FPE are that a looser path sensitization criterion is used in DYNA, and sensitizable paths in DYNA are updated dynamically rather than analyzed by a full-scale sensitizability analysis. The PS is implemented along the algorithm presented in [HPS93]. A set of paths is selected in PS according to the path selection criterion in [CDL91]. They constitutes a subcircuit of the original circuit. The optimization is made on this critical subcircuit until all paths in this subcircuit meet the delay constraint. The circuit c880 is chosen to be optimized using the three algorithms with $R = 0.9$, 0.8 and 0.7. The results are shown in Table II. The column labeled "G" is the number of gates been resized, and the column "ΔA" is the ratio of total increased area to the total initial area. As shown in Table II, for a loose delay constraint, R=0.9, the difference among the results of these three algorithms is not significant. However, when a tighter constraint is set, the result of PS is getting less satisfactory. This is because many false paths are included in the path set. The PS wastes additional effort to optimize these false paths which may never become sensitizable.

Table II Optimization result under various delay constraints for one circuit.

R	DYNA		FPE		PS	
	G	ΔA	G	ΔA	G	ΔA
0.9	3	0.38%	3	0.38%	3	0.38%
0.8	12	1.27%	14	1.58%	19	2.05%
0.7	49	5.85%	54	6.50%	58	6.86%

We continue our benchmarking effort to optimize the circuits from ISCAS benchmark set. According to the long paths in a circuit, the delay constraint parameter, R, is properly set for each circuit. The circuits are then optimized by all three algorithms. The results are shown in Table III and Table IV.

Table III shows the number of resized gates and the increased area. As shown in Table III, DYNA outperforms FPE and PS for all these circuits. For the circuit c880, much larger area overhead is required by FPE as compared with DYNA. By observing the optimization process of FPE, we find the critical paths change drastically in this circuit. However, this thrashing phenomenon on critical paths is not so serious in DYNA because a looser path sensitization criterion is used.

Table IV is the statistics of critical paths. From Table IV, we can see that a number of *false paths* are included in path set by PS in circuits c499, c1355, c1908 and c5315. Moreover, the critical paths selected in the path set of PS are not guaranteed to be sensitizable all the time during the optimization process. This is the reason that the results of PS in these four circuits are less satisfactory.

Table III Comparisons on area overhead.

circuit	R	DYNA		FPE		PS	
		G	ΔA	G	ΔA	G	ΔA
c432	0.95	13	1.35%	18	1.82%	18	1.82%
c499	0.90	30	2.02%	32	2.11%	64	3.03%
c880	0.60	130	13.89%	153	16.31%	137	14.54%
c1355	0.90	16	1.24%	36	2.86%	68	4.45%
c1908	0.85	33	1.04%	52	1.61%	108	3.95%
c2670	0.50	31	0.89%	45	1.35%	45	1.23%
c5315	0.90	43	0.52%	45	0.56%	53	0.69%

Table IV Statistics of critical paths.

circuit	R	long paths	DYNA CP	FPE CP	PS CP
c432	0.95	32954	18467	18467	18467
c499	0.90	117760	117081	117081	117152
c880	0.60	4132	4132	4132	4132
c1355	0.90	899072	183938	183938	190256
c1908	0.85	298301	15697	14618	15709
c2670	0.50	44801	29876	28548	29876
c5315	0.90	112207	27263	27251	27417

6 Conclusions

In this paper, we examine the cases for false path becoming sensitizable and sensitizable false. An algorithm to dynamically update the sensitizable set is proposed. By Combining gate sizing and dynamically identification of sensitizable paths, an efficient performance optimization tool is developed. Results on a set of circuits from ISCAS benchmark set demonstrate that our tool is indeed very effective in reducing circuit delay with less number of gate sized as compared with other methods.

References

[BMG87] J. Benkoski, E.V. Meersch, L. Glaesen and H. De Man, "Efficient Algorithms for Solving the Fasle Path Problem in Timing Verification," *Proc. ICCAD'87*,pp. 44-47, Nov. 1987.

[Bra87] R. K. Brayton, *et al.*, "MIS: A multiple-level logic optimization system," *IEEE Trans. on CAD*, Vol. CAD-6, pp. 1062-1081, Nov. 1987.

[CDL91] H.C. Chen, H.C. Du and L.R. Liu, "Critical Path Selection for Performance Optimization," *Proc. of 28th Design Automation Conf.,* pp.547-550, June 1991.

[ChD91] H.C. Chen and H.C. Du, "Path Sensitization in Critical Path Problem," *Proc. of ICCAD'91,*pp. 208-211, Nov. 1991.

[DKM91] S. Devadas, K. Keutzer and S. Malik, "Delay Computation in Combinational Logic Circuits: Theory and Algorithms," *Proc. of ICCAD'91,* pp. 176-179, Nov. 1991.

[DYG89] H.C. Du, H.C. Yen and S. Ghanta, "On the General False Path Problem in Timing Analysis," *Proc. of 26th Design Automation Conf.,* pp.555-560, June 1989.

[HPS93] S.T. Huang, T.M. Parng and J.M. Shyu, "A New Method of Identifying Critical Paths for Performance Optimization," *Proc. of EDAC-93,* pp. 455-459, Feb. 1993.

[JoF93] W.B. Jone and C.L. Fang, "Timing Optimization By Gate Resizing And Critical Path Identification," *Proc. of 30th Design Automation Conf.,* pp. 135-140, June 1993.

[McB89] P.C. McGeer and R.K. Brayton, "Efficient Algorithms for Computing the Longest Viable Path in a Combination Circuit," *Proc. of 26th Design Automation Conf.,* pp. 561-567, June 1989.